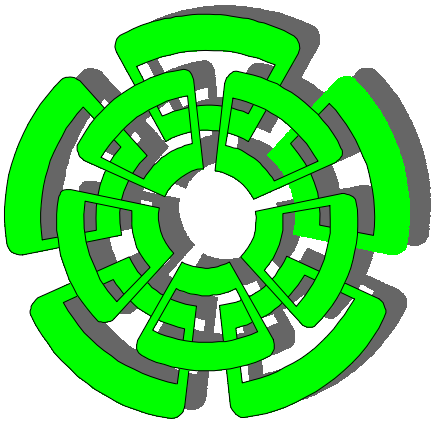

CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS AVANZADOS
DEL
INSTITUTO POLITÉCNICO NACIONAL

DEPARTAMENTO DE CONTROL AUTOMÁTICO

Modelado y Control de Sistemas Híbridos con Redes de Petri Difusas y Redes Neuronales

TESIS QUE PRESENTA
Sergio Antonio Pérez Moo*

PARA OBTENER EL GRADO DE
MAESTRO EN CIENCIAS

EN LA ESPECIALIDAD DE
CONTROL AUTOMÁTICO

DIRECTORES DE TESIS:
Dr. Wen Yu Liu
Dra. Xiaoou Li Zhang

México, D.F., 12 de Junio del 2002.

---

* Becario del CONACyT

There is not Knowledge,

that is not power.

M.K. Trilogy.

A mis papis Mónica y Toño, que me volvieron a dar su apoyo incondicional para realizar la maestría.

A toda la familia, Barbis, Chilo, Rorro, Xioma, Fany, Claudia, Cris y de manera muy especial, a Omaris la miembro mas reciente de la familia junto a mi querida novia Klaus por su constante apoyo durante toda la maestría.

A los cuates y buenos compañeros Toño, Huante, Fonseca, Gerardo, Alberto, Aarón, Eduardo, el H. P., Javier y Raúl por estar en las buenas y en las malas.

A la gente honrada de México que con sus impuestos y la administración del Consejo Nacional de Ciencia y Tecnología (CONACyT), dan el soporte económico para la realización de éste y otros trabajos científicos.

Contenido

1 Introducción **8**

1.0.1 Definición del problema 9

1.1 Estructura del trabajo 9

2 Introducción a las redes Neuronales, redes de Petri y sistemas híbridos **11**

2.1 Redes Neuronales 11

2.1.1 Modelo de una Neurona. 13

2.1.2 Estructuras de las Redes Neuronales. 16

2.2 Redes de Petri. 19

2.2.1 Modelado. 20

2.3 Sistemas Híbridos. 26

3 Representación de conocimientos **30**

3.1 Introducción 30

3.2 Red de Petri Difusa 32

3.2.1 Representación del Conocimiento. 32

3.3 Red de Petri Difusa Adaptable 37

3.3.1 Aprendizaje y entrenamiendo de la RPDA 43

3.4 Simulación 46

3.5 Conclusión 54

4 Modelado de Sistemas Híbridos mediante Redes de Petri Difusas y Redes Neuronales **56**

4.1 Introducción 56
4.2 Modelado de la dinámica continua con Redes Neuronales 57
4.3 Modelado del sistema híbrido 62
4.4 Simulación 64
4.5 Conclusión 73

5 Modelado de sistemas híbridos mediante Redes de Petri Continuas 74
5.1 Introducción 74
5.2 Red de Petri Continua 75
5.3 Modelado de sistemas híbridos via Redes de Petri Híbridas. 76
5.4 Ejemplos de redes de Petri híbridas 79
5.5 Conclusión 83

6 Control de Sistemas Híbridos mediante Redes de Petri Híbridas 84
6.1 Introducción 84
6.2 Control supervisado. 85
6.2.1 Modelo lógico para DEDS 85
6.2.2 Concepto de control supervisado y controlabilidad 86
6.3 Control supervisado mediante Redes de Petri 87
6.3.1 Variables y eventos booleanos 90
6.3.2 Uso de eventos en un modelo híbrido 92
6.3.3 Especificaciones para el control 93
6.3.4 Especificaciones para la supervisión 98
6.4 Ejemplo de control de sistemas híbridos 99
6.4.1 El proceso extendido bajo supervisión 100
6.4.2 Obtención del modelo del proceso extendido bajo supervisión 101
6.4.3 Implementación del control supervisado 104
6.5 Conclusión 105

7 Conclusiones y trabajos futuros 107

Indice de Figuras

2-1 Anatomia de una Neurona. La sinapsis es el espacio entre las neuronas para el intercambio de neurotransmisores (sucede en las dendritas y los axones). . 12

2-2 Modelo matemático de una neurona. 13

2-3 Perceptrón multicapa. 18

2-4 Red de Petri marcada (esta red es la gráfica de la estructura dada en 2.9). La marcación es $(1, 2, 0, 0, 1)$. 23

2-5 Ilustración de como cambia la marcación de un lugar. 24

2-6 Una red de Petri marcada. Marcado inicial $\boldsymbol{\mu} = (1, 0, 0)$. 26

2-7 Esquema de una planta de mezclado. 28

3-1 Red de Petri Difusa 35

3-2 Disparo de una red de Petri difusa. (a) Antes de disparar la transición t_i. (b) Despues de disparar la transición t_i. 36

3-3 Red de Petri difusa marcada del tipo 1. (a) Antes de disparar la transición t_i. (b) Despues de disparar t_i. 36

3-4 Red de Petri difusa marcada del tipo 2. (a) Antes de disparar la transición t_i. (b) Despues de disparar t_i. 37

3-5 Red de Petri difusa marcada del tipo 3. (a) Antes de disparar la transición t_i. (b) Despues de disparar t_i. 38

3-6 Red de Petri difusa adaptable del tipo 1. 39

3-7 Red de Petri difusa adaptable del tipo 2. 39

3-8 Red de Petri difusa adaptable del tipo 3. 39

3-9 RPDA del ejemplo 3.1. 40

3-10 Funciones sigmiodes del Ejemplo 3.1. 44
3-11 La translación de la red neuronal de la parte de aprendizaje del Ejemplo 3.1. 49
3-12 Resultados de una capa de aprendizaje del Ejemplo 3.1. 50
3-13 RPDA del ejemplo 3.2. 51
3-14 La translación a red neuronal de la RPDA de la figura 3-13. 52
3-15 Resultados del aprendizaje del PMC del Ejemplo 3.2. 54

4-1 Estructura de la red neuronal dinámica. 58
4-2 Estructura para el modelado híbrido. 63
4-3 Gráfica de la red híbrida. 64
4-4 Sistema de dos tanques. 65
4-5 Modelo lógico del sistema de dos tanques. 66
4-6 Modelo híbrido de los dos tanques. 67
4-7 Comportamiento del Tanque 1. 68
4-8 Comportamiento del Tanque 2. 69
4-9 Comportamiento de la red neuronal del Tanque 1. 69
4-10 Comportamiento de la red neuronal del Tanque 2. 70
4-11 Simulación del comportamiento lógico de los dos tanques. 70
4-12 Modelo híbrido de la regla del tipo 2. 71
4-13 Esquema de una celda del proceso de llenado. 72
4-14 Modelo híbrido de una celda de llenado. 72

5-1 (a) Red de Petri Continua. (b) Red de Petri continua con transición de salida discreta. 76
5-2 Gráfica de una red de Petri híbrida. 78
5-3 Red de Petri híbrida de la regla de producción difusa. 80
5-4 Gráfica de la red de Petri híbrida del ejemplo *5.2.* 83

6-1 Control supervisado. (a) Proceso P en aislamiento. (b) Proceso P con comportamiento restringido. (c) Esquema del control supervisado. 86

6-2 Control supervisado. (a) El proceso extendido. (b) El esquema del supervisor. (c) El esquema del control supervisado. 88
6-3 Ejemplo 2: Llenado de tanques. 91
6-4 Dos modelos correspondientes a las misma especificaciones. (a) Las entradas son eventos y variables Booleanas, las salidas son variables Booleanas (Red de Petri). (b) Todas las entradas y salidas son eventos (Red Híbrida). . . . 91
6-5 Un sistema de manufactura. 93
6-6 Modelos del sistema de manufactura. (a) Modelo de M_1. (b) Modelo de M_2. 94
6-7 Especificaciones del control para el sistema de manufactura. (a) Modelo híbrido para M_1. (b) Modelo híbrido para M_2. 95
6-8 Árbol de estados del proceso extendido. (a) Árbol global. (b) Árbol para M_1. 96
6-9 Especificaciones para el supervisor. 99
6-10 Proceso extendido bajo supervisión. (a) Modelo híbrido de M_1. (b) Modelo híbrido de M_2. 100
6-11 Ejecución de una acción impulso condicional. 101
6-12 Modelos equivalentes a los de la Fig.6-10. 102
6-13 Árboles de el proceso extendido bajo supervisión, obtenidos de la Fig.6-12. (a) Árbol para M_1. (b) Árbol para M_2. 103
6-14 Árbol global de el proceso extendido bajo supervisión. 104
6-15 Esquema del control supervisado de el ejemplo del sistema de manufactura. . 105

Capítulo 1

Introducción

La concepción básica de un modelo es la representación de un sistema al cual se le aplica un experimento para responder preguntas sobre el mismo. Por lo que generalmente la definición de modelo se da apartir de la representación, a menudo en término matemáticos, de las caractéristicas importantes del objeto o sistema bajo estudio. Por la manipulación de la representación, se espera que nuevo conocimiento acerca del fenómeno modelado pueda ser obtenido sin peligro, costo o inconveniente de manipular el mismo fenómeno real.

El desarrollo de computadoras de alta velocidad ha hecho posible incrementar las aplicaciones del modelado, mediante la representación de un sistema como un modelo matemático, convirtiendo al modelo en instrucciones para una computadora. Actualmente es posible modelar sistemas más grandes y complejos que antes. Esto ha resultado en considerar el estudio de las técnicas de modelado para computadoras, es decir, las computadoras están involucradas en el modelado en dos maneras: como herramienta computacional para modelado y como sujetos de modelado.

Existen varias técnicas de modelado que caen en los casos mencionados arriba, en el trabajo de esta tesis nos ocupamos de las redes neuronales y las redes de Petri (donde esta última se encuentra en ambos casos). Estas redes desempeñan un papel importante ya que se pretende fusionarlas para crear nuevas estructuras para la representación del mundo real.

1.0.1 Definición del problema

La motivación principal para la elaboración de esta tesis, ha sido el estudio y desarrollo de distintas técnicas de modelado de sistemas denominados híbridos, principalmente se enfoca a una mejor representación del conocimiento de los denominados sistemas expertos. Los sistemas se pueden clasificar de manera general en; sistemas continuos, sistemas discretos y sistemas a eventos discretos. Los sistemas híbridos se pueden considerar como la combinación de un parte continua y una parte discreta. Junto a este tema central, se añade la necesidad de presentar un esquema de control para dichos sistemas. Siendo así, se desarrollarán modelos para sistemas híbridos y un esquema de control en el que se apliquen dichos modelos.

De la diversidad de modelos que se han desarrollado en el pasado siglo, es de interés para el desarrollo de esta tesis las Redes Neuronales (RN) y las Redes de Petri (RP), dadas sus capacidades de representar sistemas continuos y discretos respectivamente.

Como se mencionó al principio, la tesis se desarrolla en la representación del conocimiento en sistemas expertos (particularmente sistemas híbridos), por esta razón se utilizan nociones de lógica difusa (a nivel básico) con lo cual se propone lograr una mejor representación del conocimiento, es decir, poder procesar mejor la información del mundo real.

1.1 Estructura del trabajo

Para su presentación este trabajo ha sido organizado de la siguiente forma:

El capítulo 2 se da la introducción a los conceptos básicos de las *Redes Neuronales, Redes de Petri* y la definición de *Sistemas Híbridos*.

En el capítulo 3, se dan las nociones básicas de lógica difusa con la cual se establecen las reglas de producción difusas para la representación del conocimiento. Se construye un modelo de tales reglas con la ayuda de las redes de Petri obteniendo una nueva estructura denominada Red de Petri Difusa (RPD), se obtiene entonces una red de Petri que modela reglas de conocimiento difusas, esta estructura está propuesta en [11]. La segunda parte del capítulo presenta una concepto original obtenido de las RPD el cual es una estructura de red de Petri la cual es adaptiva (RPDA), esta red es propuesta en [33].

El capítulo 4 introduce un nuevo arreglo para el modelado de los sistemas híbridos,

este arreglo se derivó de la idea de la red de Petri adaptable. Este arreglo consta de una combinación directa de las redes neuronales y las redes de Petri, donde en los lugares de la red de Petri se lleva acabo la representación continua del sistema con la ayuda de una red neuronal. Este nuevo modelo se presentó en el artículo [44].

En el capítulo 5 se presenta un modelo híbrido de redes de Petri, el cual utiliza el concepto de Red de Petri continuas. Utilizando una red de Petri difusa y una red de Petri continua obtenemos una nueva estructura la cual se denomina red de Petri híbrida.

El capítulo 6 se da un nuevo enfoque para el desarrollo de un control supervisado para los sistemas híbridos utilizando de manera general los modelos obtenidos en los capítulos 3, 4 y 5. Esta propuesta tiene el inconveniente de que poco estudios se han hecho sobre el tema y los que se encuentran en la literatura se basan en modelos autómatas para la prueba de controlabilidad. Este esquema de control supervisado se presentó en el artículo [43].

El capítulo 7 se dan las conclusiones del trabajo desarrollado en la tesis y sus posibles lineas de investigación a seguir a futuro.

Capítulo 2

Introducción a las redes Neuronales, redes de Petri y sistemas híbridos

2.1 Redes Neuronales

Las redes neuronales han sido desarrolladas en base a procesos biológicos para el procesamiento de información, basados específicamente en el sistema nervioso y su unidad básica, la *neurona*. Las señales son propagadas mediante las *denditras* en la forma de diferencias de potencial entre el interior y exterior de la neurona. Las partes principales de una célula neuronal se muestran en la Figura 2-1.

Las dendritas traen señales desde otras neuronas dentro del cuerpo de la célula o *soma*, posiblemente multiplicando cada señal entrante por un *coheficiente de pesos* de transferencia. En el soma, la capacitancia celular integra las señales las cuales colecta en el axón Hillock. Una vez que la señal compuesta excede un *umbral* celular, una señal, el potencial de acción, es transmitido a lo largo del axón. En la conducción de las señales las nolinealidades celulares hacen del potencial de acción compuesto una *función nolineal* de la combinación de las señales que arrivan al soma. El axón se conecta en el hueco de las sinapsis con las dendritas de las neuronas subsecuentes. La sinapsis opera mediante la descarga de neurotransmisores químicos a través de brechas intercelulares, y puede ser excitatoria (tendiendo a disparar cuando se exceda el umbral de las entradas de la siguiente neurona) o inhibitoria (tendiendo

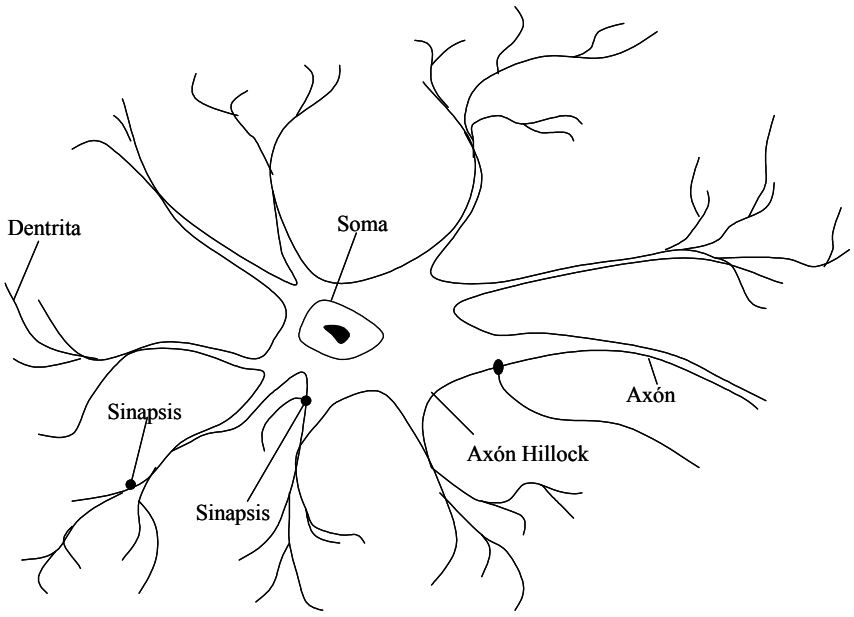

Figura 2-1: Anatomia de una Neurona. La sinapsis es el espacio entre las neuronas para el intercambio de neurotransmisores (sucede en las dendritas y los axones).

a prevenir el disparo a la siguiente neurona).

Los sistemas biológicos proveen de muchas pistas para el desarrollo del aprendizaje *robusto* (altamente estable) y de algoritmos adaptables. Dichos sistemas procesan la información en forma diferente a los esquemas de control convencionales, ya que no están basados en ningún modelo, sin embargo son muy eficientes para tratar con incertidumbres y complejidades, no requieren del desarrollo de un modelo matemático para ejecutar tareas complejas. Ciertamente, pueden aprender a ejecutar nuevas tareas y adaptarse facilmente a cambios en el ambiente.

Si los principios fundamentales de la computación encajaran en los sistemas biológicos (por ejemplo: el cerebro), entonces una generación totalmente nueva de métodos de control podría ser desarrollada más alla de las capacidades de las técnicas actuales, basadas en un modelo matemático explícito.

Se dice que un sistema de control tiene la habilidad de aprender si adquiere información durante la operación de comportamientos desconocidos de la planta y de su ambiente, tal que la ejecución completa sea mejorada. Con este enriquecimiento el controlador podría

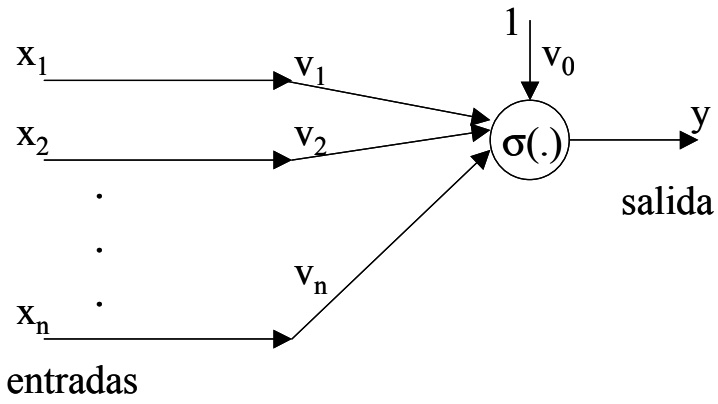

Figura 2-2: Modelo matemático de una neurona.

expandir su región de operación, de manera que pueda resolvar estos problemas en forma autónoma.

Una clase de modelos, con la potencialidad de implementar aprendizaje, son las redes neuronales artificiales. Y aunque la fisiología neuronal del sistema nervioso es mucho mas compleja, una analogía simplificada puede ser desarrollada, la cual podría ser utilizada en aplicaciones de ingeniería.

2.1.1 Modelo de una Neurona.

Un modelo matemático de una neurona está descrito en la Figura 2-2, la cual muestra los pesos de la dendrita v_j, el umbral de disparo v_0 (también llamado la *bias*), la suma de las señales de entrada pesadas, y una función nolineal $\sigma(\cdot)$. Las entradas a la célula son las n señales temporales $x_1(t), x_2(t), ...x_n(t)$ y la salida es el escalar $y(t)$, la cual puede ser definida como

$$y(t) = \sigma(\sum_{j=1}^{n} w_j x_j(t) \pm v_o).$$

Los pesos positivos ω_j corresponden a sinapsis excitatorias y pesos negativos a sinapsis inhibitorias. Esto fue llamado el *perceptrón* por Rosenblatt en 1959 [29].

La función celular $\sigma(\cdot)$ es conocida como la *función de activación*, es seleccionada depen-

diendo de las aplicaciones (ver la Tabla *2.1*). La función de activación modela el comportamiento de la célula donde no se tiene salida por debajo de un cierto valor del argumento de $\sigma(\cdot)$ y la salida toma una magnitud específica por encima del valor del argumento. De la Tabla *2.1* se tiene una clase general de funciones monotónicamente no decrecientes que toman valores acotados en $-\infty$ y $+\infty$ y son conocidas como las funciones *sigmoides*. Para muchos algoritmos de entrenamiento (incluyendo retropropagación), la derivada de $\sigma(\cdot)$ es necesaria en al demostración de su convergencia por lo que la función de activación seleccionada debe ser diferenciable dependiendo del caso.

Una Red Neuronal Artificial (RNA) [22] es un elemento capaz de procesar gran cantidad de información en forma paralela y distribuida, las cuales pueden almacenar conocimiento experimental y tenerlo disponible para su uso [23]. Algunas de sus similaridades con el cerebro son:

1. El conocimiento es adquirido a través del proceso de aprendizaje.

2. La conectividad entre neuronas es llamada pesos sinápticos y estos son utilizados para almacenar el conocimiento.

El procedimiento para el proceso de aprendizaje es conocido como algoritmo o leyes de aprendizaje. Su función es modificar los pesos sinápticos de la red neuronal para alcanzar una meta preestablecida. La modificación de los pesos se basa el método tradicional para el diseño e implementación de las redes neuronales.

La neurona es la unidad fundamental para la operación de la red neuronal. La Figura 2-2 muestra el modelo de una neurona.

Existen tres elementos básicos de la RNA:

1. Un conjunto de uniones sinápticas, en cada elemento, caracterizado por su propio peso.

2. Una función que relaciona las señales de entrada con los respectivos pesos sinápticos.

3. Una función de activación no lineal que transforma la salida del sumador en la entrada de la siguiente neurona.

4. Un umbral externo para reducir la entrada de la función de activación.

En términos matemáticos, el modelo de la i-ésima neurona se describe como:

$$\begin{aligned} u_i &= \sum_{j=1}^{n} w_{ij} x_j \\ y_i &= \sigma(u_i - p_i), \end{aligned} \tag{2.1}$$

donde:

x_j j-ésimo componente de la entrada,

w_{ij} peso de la conexión entre la j-ésima componente de la entrada y la i-ésima neurona,

u_i salida del sumador,

p_i umbral,

$\sigma(\cdot)$ función de activación no lineal,

y_i salida de la i-ésima neurona.

La función de activación no lineal es denotada por $g(\cdot)$ y genera el elemento de la salida y_i, recibiendo como entrada x_i:

$$y_i = g(x_i). \tag{2.2}$$

Podemos clasificar este tipo de funciones de la manera siguiente: Diferenciable y No-diferenciable, se distingue por tener funciones suaves y discontinuas. Las funciones suaves son necesarias para algunos algoritmos de adaptación como el de propagación hacia atrás (*backpropagation*), mientras que las funciones discontinuas (por ejemplo: las funciones de umbral) son necesarias para generar una salida binaria. Tipo pulso y Tipo escalón, se distingue por tener funciones con un solo valor significativo de salida cuando las entradas están cerca del cero, porque las funciones solo cambian significativamente alrededor del cero. Positiva y Promedio cero, se refiere a las funciones positivas que cambian de 0 en $-\infty$ a 1 en ∞ y a las funciones de promedio cero que cambian de -1 en $-\infty$ a 1 en ∞.

Algunas funciones comunes pueden observarse en la Tabla *2.1*:

Nombre	Fórmula $\sigma(\cdot)$	Características
Umbral	$+1$ si $x > 0$, 0 si $x \leq 0$	No-diferenciable, tipo escalón, positiva
Umbral	$+1$ si $x > 0$, -1 si $x \leq 0$	No-diferenciable, tipo escalón, prom. cero
Sigmoide	$\frac{1}{1+e^{-x}}$	Diferenciable, tipo escalón, positiva
Tangente hiperbólica	$\tanh(x)$	Diferenciable, tipo escalón, prom. cero
Gaussiana	$e^{-\frac{x^2}{\sigma^2}}$	Diferenciable, tipo pulso

Tabla *2.1* Distintas funciones de activación.

2.1.2 Estructuras de las Redes Neuronales.

La forma en que las neuronas de una red neuronal están interconectadas determina su estructura. Para propósitos de identificación y control, las estructuras mas usadas son:

1. Redes de alimentación hacia adelante de una capa.
2. Redes de alimentación hacia adelante multicapa.
3. Redes de funciones radiales básicas.
4. Redes neuronales dinámicas.

Redes de alimentación hacia adelante de una capa.

Esta es la forma mas simple de una red neuronal, tiene solo una capa de neuronas. La mas conocida es llamada *Perceptrón (*ver Figura 2-2), que básicamente consta de una neurona con pesos sinápticos ajustables y de una función de activación. El perceptrón básico es el llamado modelo de McCulloch-Pitts [27]. El algoritmo de aprendizaje que ajusta los pesos de estas redes neuronales apareció por primera vez en [29], [30]. Ahi es probado que los vectores de información, usados para entrenar el perceptrón, son tomados de dos clases lineales separables, entonces el algoritmo de aprendizaje del perceptrón converge y define una superficie de decisión: un hiperplano que separa las dos clases. La prueba de convergencia, respectiva, es conocida como el teorema de convergencia del perceptrón. El propósito del perceptrón es el de clasificar la señal de entrada, con componentes: u_1, u_2, ..., u_n, en una

o dos clases: C_1 o C_2. La regla de decisión para la clasificación consiste en asignar a cada punto, que corresponde a una entrada u_1, u_2, ..., u_n, la clase C_1 si la salida del perceptrón y es igual a $+1$ y la clase C_2 si es -1.

Usualmente el umbral p es tratado como un peso sináptico conectado a una entrada fija en -1, asi el vector de entrada queda definido por:

$$(\mathbf{u}(k))^T = (-1, u_i(k), u_2(k), ..., u_n(k)), \tag{2.3}$$

donde k es la k-ésima entrada.

El sumador que produce la salida es calculado por:

$$\mathbf{v}(k) = \mathbf{w}^T(k)\,\mathbf{u}(k) = \mathbf{u}^T(k)\,\mathbf{w}(k), \tag{2.4}$$

donde $\mathbf{w}$ es el vector de los pesos sinápticos.

Existe un vector $\mathbf{w}$ tal que para cualquier k, en el espacio n-dimensional, la ecuación $\mathbf{w}^T\mathbf{u}$, con coordenadas u_1, u_2, ..., u_n, define un hiperplano que separa las entradas en dos clases: C_1 y C_2:

$$\mathbf{w}^T\mathbf{u} \geq 0, \nabla\mathbf{u} \in C_1 \quad \text{y} \quad \mathbf{w}^T\mathbf{u} < 0, \nabla\mathbf{u} \in C_2.$$

El algoritmo de aprendizaje adapta el vector de los pesos como sigue:

$$w_{ij}[k+1] = w_{ij}[k] + \Delta w_{ij}, \tag{2.5}$$

donde

$$\Delta w_{ij} = \eta[d_i - sgn(w_i^T x)]x_j = \begin{cases} 2\eta d_i x_j & \text{si } y_i \neq d_i \\ 0 & \text{de otra forma} \end{cases}$$

La convergencia de este algoritmo puede ser demostrada utilizando un argumento por contradicción (ver [29]).

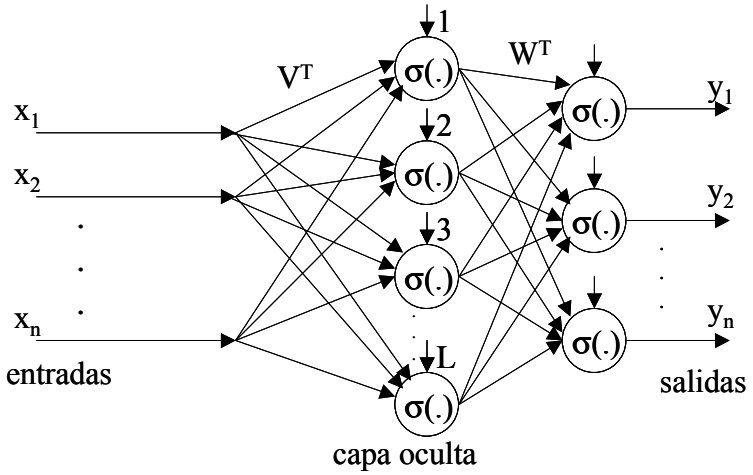

Figura 2-3: Perceptrón multicapa.

Si se observa únicamente la salida $v(k)$ y se define el error como:

$$\mathbf{e}(k) = \mathbf{d}(k) - \mathbf{v}(k). \tag{2.6}$$

Se utiliza el método del gradiente descendiente para minimizar el cuadrado del error, esto da por resultado la siguiente ley de aprendizaje:

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \eta \mathbf{e}(k)\,\mathbf{u}(k). \tag{2.7}$$

Como el error depende linealmente de los pesos, este algoritmo asegura la obtención de un mínimo global.

Redes de alimentación hacia adelante multicapa.

Las redes de alimentación hacia adelante multicapa se distinguen, por la presencia de una o mas capas ocultas (ver la Figura 2-3).Cuyos nodos computacionales internos se llaman neuronas ocultas. Típicamente, las neuronas en cada capa tienen como señal de entrada las señales de salida de la capa precedente. Si cada neurona, en cada capa, es conectada con todas las neuronas de las capas adyacentes la red neuronal se dice que está totalmente conectada, en el caso opuesto, es llamada parcialmente conectada.

El perceptrón multicapa tiene las siguientes tres características:

1. La función de activación para cada neurona es suave, en oposición a la de límites extremos usada en el perceptrón de una sola capa. Usualmente, esta función no lineal es una sigmoide:

$$\sigma_i(x_i) = \frac{1}{1+e^{-x_i}}. \tag{2.8}$$

2. La red está compuesta por una o mas capas ocultas de neuronas.

3. La red presenta un alto grado de conectividad, es decir, el número de concecciones entre las capas ocultas aumenta drásticamente al aumentar el número de neuronas en las capas (su complejidad aumenta).

El perceptrón multicapa obtiene su poder computacional de la combinación de estas características y su capacidad de aprender de la experiencia. Sin embargo, la presencia de no-linealidades distribuidas y la alta conectividad de la red hacen el análisis teórico difícil de realizar. En suma a mayor número de capas la complejidad en la red aumenta, dependiendo de los requeriemientos del diseñador. En el caso de la tesis, el trabajo experimental solo se utilizaron redes neuronales de una capa.

2.2 Redes de Petri.

Las *redes dePetri* son herramientas para el estudio de sistemas que presentan características como son: Estar compuestos de partes que interactuan por separado, donde cada componente puede ser en si un sistema, pero su comportamiento puede ser descrito independientemente de las otras partes del sistema. Dichos componentes presentan *concurrencia* o *paralelismo*, esto es que sus actividades se realizan de manera simultánea con las actividades de otros componentes del sistema.

Estas características crean un problema para modelar al sistema ya que se presenta el fenómeno de *sincronización,* en el cual la transferencia de información o materiales de una parte a otra parte del sistema requiera que las actividades de las otras partes estén en

sincronía, para que no se tenga que un componente esté en espera de otro o se llegue a un punto muerto donde las actividades cesen.

La teoría de redes de Petri permite a un sistema con estas características ser modelado mediante una red de Petri, el análisis de la red de Petri puede entonces, ayudar a revelar información importante acerca de la estructura y el comportamiento dinámico del sistema modelado. Esta información puede ser utilizada para evaluar el sistema modelado y sugerir mejoras o cambios. Asi pues, el desarrollo de una teoría de redes de Petri está basada en la aplicación de redes de Petri en el modelado y diseño de dichos sistemas.

2.2.1 Modelado.

La aplicación de las redes de Petri es a través del *modelado*. En muchos campos de estudio, un fenómeno no es estudiado directamente, pero si indirectamente utilizando un *modelo* de el fenómeno. Las redes de Petri fueron diseñadas principalmente para *modelar* los sistemas descritos anteriormente, en particular, pueden modelar el flujo de información u otros recursos dentro de un sistema. estos pueden ser de distintas clases: hardware computacional, software, sistemas físicos, sociales, etc. Las redes de Petri son usadas para modelar la ocurrencia de varios eventos y actividades en un sistema.

Origen de las redes de Petri.

Las redes de Petri son diseñadas específicamente para modelar sistemas con componentes concurrentes interactivos. Las redes de Petri fueron desarrolladas por los primeros trabajos de Carl Adam Petri [1962]. En su disertación doctoral, *Kommunikation mit Automaten*, formuló las bases para una teoría de comunicación entre los componentes asíncronos de un sistema computacional. Petri estaba particularmente enfocado con la descripción de las relaciones casuales entre eventos. Su disertación fue primordialmente un desarrollo teórico de los conceptos básicos para el desarrollo de las redes de Petri.

Una de las características principales de las redes de Petri es su naturaleza *asíncrona*, es decir, no hay una medida inherente del tiempo ó de el flujo de tiempo en una red de Petri. Esto refleja una filosofía del tiempo la cual establece que su única propiedad importante desde

un punto de vista lógico, es una definición de un ordenamiento parcial de la ocurrencia de eventos.

Estructura de las redes de Petri.

Una red de Petri está compuesta de cuatro partes: un conjunto de *lugares* P, un conjunto de *transiciones* T, una función de *entrada* I y una función de *salida* O. Las funciones de entrada y salida relacionan transiciones y lugares . La función de entrada I es un mapeo de una transición t_j a una colección de lugares $I(t_j)$ conocida como los *lugares de entrada* de la transición. La función de salida O mapea una transición t_j a una colección de lugares $O(t_j)$ conocida como los *lugares de salida* de la transición.

Definición 2.1 *[39] Una estructura de Petri C es una cuátrupla*

$$C = (P, T, I, O),$$

donde $P = \{p_1, p_2, ..., p_n\}$ es un conjunto finito de lugares, $n \geq 0$. $T = \{t_1, t_2, ..., t_m\}$ es un conjunto finito de transiciones, $m \geq 0$. El conjunto de lugares y el conjunto de transiciones son disjuntos, $P \cap T = \emptyset$. $I : T \rightarrow P^\infty$ es la función de entrada, un mapeo de transiciones a una bolsa de lugares[1]. $O : T \rightarrow P^\infty$ es la función de salida, un mapeo de transiciones a una bolsa de lugares.

La cardinalidad del conjunto P es n, y la cardinalidad de el conjunto T es m. Denotamos un elemento arbitrario de P por p_i, $i = 1, ..., n$, y un elemento arbitrario de T por t_j,

[1] *Una bolsa es una colección de elementos sobre un dominio, donde se permite la ocurrencia múltiple de elementos. En la teoría de bolsas, un elemento puede estar dentro de una bolsa cero veces, una vez, dos veces, o un número de veces determinado.*

$j = 1, ..., m$. Una estructura de red de Petri se muestra a continuación:

$$\begin{aligned} C &= (P, T, I, O), \\ P &= \{p_1, p_2, p_3, p_4, p_5\}, \\ T &= \{t_1, t_2, t_3, t_4\}, \\ I(t) &= \{p_1\}, \qquad O(t) = \{p_2, p_3, p_5\}, \\ I(t) &= \{p_2, p_3, p_5\}, \quad O(t) = \{p_5\}, \\ I(t) &= \{p_3\}, \qquad O(t) = \{p_4\}, \\ I(t) &= \{p_4\}, \qquad O(t) = \{p_2, p_3\}. \end{aligned} \tag{2.9}$$

Representación gráfica de las redes de Petri.

Una *gráfica* de la red de Petri es una representación de la estructura de la red de Petri como un digrafo bipartita direccionado, consiste de lugares y transisciones, representados como dos tipos de nodos: un *círculo* ◯ representa un lugar; una *barra* | representa una transición.

Los arcos direccionados (flechas →) conectan los lugares y las transiciones, con algunos arcos direccionados de los lugares a las transiciones y otros arcos direccionados de las transiciones a los lugares. Un arco direccionado de un lugar p_i a una transición t_j define al lugares para ser una entrada de la transición. Múltiples entradas a una transición son indicadas por múltiples arcos desde los lugares de entrada a la transición. Un lugar de salida es indicado por un arco desde la transición al lugar. Igualmente múltiples salidas son representadas por arcos múltiples. Un ejemplo típico está dado en la Figura2-4.

Definición 2.2 *[39] Una gráfica de Red de Petri G es un digrafo bipartita direccionado,*

$$G = (V, A),$$

donde $V = \{v_1, v_2, ..., v_s\}$ es un conjunto de vértices y $A = \{a_1, a_2, ..., a_r\}$ es una bolsa de arcos direccionados, $a_i = (v_j, v_k)$, con $v_j, v_k \in V$. El conjunto V puede ser particionado en dos conjuntos disjuntos P y T tales que $V = P \cup T$, $P \cap T = 0$, y por cada arco direccionado, $a_i \in A$, si $a_i = (v_j, v_k)$, entonces $v_j \in P$ y $v_k \in T$ o $v_j \in T$ y $v_k \in P$.

en especificaciones continuas y discretas. En la primera, el modelo de la planta híbrida es reemplazado por un modelo de eventos discretos solamente, relacionado con las especificaciones continuas y se utiliza un supervisor diseñado de la teoría de control supervisoria sugerida por Wonham-Ramadge. Los otros dos métodos utilizan directamente la especificación continua para la determinación de un control generador de eventos, donde los aspectos de tiempo óptimo son introducidos como una opción en la última aproximación.

Las redes de Petri han sido utilizadas para modelar sistemas híbridos. En [17] se hace una presentación tanto intuitiva como formal de las redes de Petri híbridas, es decir, de redes de Petri conteniendo una parte discreta y una continua, cuya aplicación permite el modelado de sistemas con eventos discretos y dinámicas de tiempo continuas. Para sistemas híbridos se ha modelado la parte discreta por una red de Petri mientras el aspecto continuo está descrito como un conjunto de ecuaciones algebraicas diferenciales. Este modelo puede ser visto como una extensión de un autómata híbrido [21]. Las redes de Petri de alto nivel están caracterizadas por el uso de *tokens* individuales estructurados, este modelo proveé de una representación gráfica simple de sistemas híbridos tomando ventaja de la estructura de las redes de Petri para ver una descripción compacta de los sistemas.

Las redes de Petri híbridas que modelan sistemas híbridos, están compuestas de una red de Petri discreta (sección *1.1*) y una red de Petri continua (sección *5.1*), que interactúan cada una de ellas a través de señales de condición evento. La cual toma las ventajas de la estructura de las redes de Petri y las características modulares de los sistemas condición-evento.

En la presente tesis, solo nos concierne el desarrollo y presentación de nuevos modelos para sistemas híbridos, sin realizar análisis (esto es causado por el poco trabajo desarrollado en la literatura), para lo cual es necesario un trabajo de mayor profundidad.

Capítulo 3

Representación de conocimientos

3.1 Introducción

Las Redes de Petri (RP) tienen la capacidad de representar para su análisis el fenómeno de concurrencia y sicronización, como las evoluciones concurrentes, donde varios procesos que evolucionan simultáneamente son parcialmente independientes. Además, la aproximación con RP puede ser combinada con otras técnicas y teorías tales como programación orientada a objetos, teoría difusa, redes neuronales, etc. Estas RP modificadas son utilizadas extensamente en computación, sistemas de manufactura, robótica, sistemas basados en conocimiento, control de procesos, así como otro tipo de aplicaciones en ingeniería.

Por muchos años, uno de los tópicos en la investigación de la inteligencia artificial (IA) ha sido el desarrollo de notaciones suficientemente precisas para la representación del conocimieto. Para hacer el conocimiento del mundo real conveniente para el procesado en computadoras, muchos métodos de representación del conocimiento han sido desarrollados, como las reglas de producción [3], reglas de producción difusa [12], [37], redes semánticas [10], referenciales [10], redes de transición afirmadas [18], redes de Petri [39], [41], [47], guiones, gráficas conceptuales [15], [45], [46], proposiciones [54], etc.

De tales combinaciones surge la propuesta de utilizar la teoría difusa, esto con el fin de obtener modelos difusos con redes de Petri, las cuales tienen una cualidad inherente en la representación de la lógica en forma intuitiva y visual, es de suponerse que en cuestiones como la lógica difusa se podria establecer capacidades semejantes. En este capítulo, un

modelo de red de Petri difusa nombrado RPD se da aconocer para representar las reglas de producción difusa, esto permite la representación estructurada del conocimiento y tiene un procedimiento sistemático para apoyar el razonamiento difuso.

Dichas Redes de Petri Difusas (RPD), han resultado capaces de representar y razonar las relaciones de la lógica difusa como se muestra en los artículos [14], [35], [42], [51], [2], [7]. Pero existen dos importantes debilidades en los métodos propuestos por estos autores:

- La complejidad del sistema de conocimiento causa un enorme modelo de Red de Petri Difusa [14].
- El sistema de conocimiento es actualizado o modificado frecuentemente. Un modelo apropiado para ello tendria que ser adaptable [20].

Los sistemas de conocimiento pueden ser considerados como sistemas dinámicos, los modelos de conocimiento deben tener la habilidad de ajustarse de acuerdo a los cambios de los sistemas. Asi pues, la falta del mecanismo (aprendizaje) de ajuste en las RPD no puede hacer frente a los cambios potenciales de los sistemas actuales. En algunos artículos se han propuesto redes de Petri difusas ajustables, como en [35], donde se presenta un algoritmo para ajustar los umbrales de la RPD, pero el ajuste de los pesos fue realizados mediante pruebas. El artículo [48] propone un modelo de RPD generalizada (RPDG) la cual puede ser transformada en redes neuronales con neuronas lógicas OR/AND [20], así los parámetros de las correspondientes redes neuronales pueden ser aprendidos (entrenamiento). De hecho, el aprendizaje del conocimiento en [48] fue bajo un marco de trabajo de redes neuronales. Las Redes de Petri Difusas Adaptables (RPDA) [33] tienen también la habilidad de aprender de una red neuronal, pero no necesitan ser transformadas a redes neuronales. Sin embargo el algoritmo de aprendizaje está basado en una regla especial de disparo de la transición (ver sección *3. 3)*, es necesario conocer los factores de certeza de cada proposición consecuente en el sistema. En el capítulo, se propone un mecanismo de aprendizaje para la RPDA. El algoritmo de retropropagación es desarrollado para el aprendizaje de conocimiento bajo condiciones generalizadas.

3.2 Red de Petri Difusa

3.2.1 Representación del Conocimiento.

En muchas situaciones, puede ser difícil el capturar datos en forma precisa. Una forma para representar propiamente el conocimiento del mundo real, bajo la teoria difusa son las reglas de producción difusas [12], [37].

Una regla de producción difusa es una regla la cual describe la relación entre dos proposiciones lingüísticas. Sea R un conjunto de reglas de producción difusas $R = \{R_1, R_2, ..., R_n\}$. La formulación general de la i-ésima regla de producción difusa está dada como sigue:

$$R_i : \text{IF } d_j \text{ THEN } d_k(CF = \mu_i) \tag{3.1}$$

donde; d_j and d_k son proposiciones las cuales tiene asociadas un valor denominado grado de verdad [12]. Si la porción antecedente o la porción consecuente de una regla de producción difusa contiene conectores y u o, entonces es llamada una regla de producción difusa compuesta. El grado de verdad de cada proposición es un valor real entre cero y uno. Un ejemplo de escalas de verdad y su correspondiente intervalo numérico está dado en la Tabla *3.1*:

Tabla *3.1.*

Escalas de Verdad	Intervalos Numéricos
siempre verdadero	[1.00, ∞)
extremadamente verdadero	[0.95, 0.99]
muy verdadero	[0.80, 0.94]
considerablemente verdadero	[0.65, 0.79]
moderadamente verdadero	[0.45, 0.64]
más o menos verdadero	[0.30, 0.44]
minoritariamente verdadero	[0.10, 0.29]
mínimamente verdadero	[0.01, 0.09]
no verdadero	(-∞, 0.00]

μ_i es el valor de el factor de certeza (CF), $\mu_i \in [0,1]$. Representa la fuerza de credibilidad en la regla. Entre más grande el valor más se cree en la regla, y $1 \leq i \leq n$.

Por ejemplo, la siguiente es una regla de producción:

$$R_i: \ IF \text{ está caliente } THEN \text{ la humedad es baja } (CF = 0.90). \tag{3.2}$$

De acuerdo con [12] y [37], se define un caso particular para evaluar las reglas; Sea λ un valor umbral, donde $\lambda \in [0,1]$, si el grado de verdad de la proposición d_j de la regla presentada en ec.3.1 es y_j, donde $y_j \in [0,1]$, entonces

Caso 3.1 *Si $y_j \geq \lambda$, entonces (la ecc.3.1) será disparada. Obteniendo el grado de verdad de la proposición d_k de (la ecc.3.1) como $y_j \cdot \mu_i$.*

Caso 3.2 *Si $y_j < \lambda$, entonces (la ecc.3.1) no se dispara.*

En el ejemplo previo, si el grado de verdad de la proposición *está caliente* es 0.90 y el valor umbral $\lambda = 0.20$, entonces la regla R_1 puede ser disparada, y el grado de verdad de la proposición *la humedad es baja* es $0.90 \cdot 0.90 = 0.81$. Esto indica que la posibilidad de humedad baja es 0.81.

Podemos utilizar una gráfica de red de Petri difusa (RPD) para representar las reglas de producción. Una red de Petri difusa difiere de una red tradicional de Petri principalmente en que los lugares tienen marcado de *tokens* asociados con un valor de verdad entre cero y uno, y cada transición está asociada con un valor de factor de certeza entre cero y uno. El concepto de red de Petri difusa está derivado de las redes de Petri de [39].

Definición 3.1 *[12] Una estructura generalizada de red de Petri difusa puede ser definida como una 8-tupla:*

$$RPD = (P, T, D, I, O, f, \alpha, \beta), \tag{3.3}$$

donde $P = \{p_1, p_2, ..., p_n\}$ es un conjunto finito de lugares, $T = \{t_1, t_2, ..., t_m\}$ es un conjunto finito de transiciones, $D = \{d_1, d_2, ..., d_n\}$ es un conjunto finito de proposiciones, $P \cap T \cap D = \varnothing$, $|P| = |D|$, $I : T \rightarrow P^{\infty}$ es la función de entrada, un mapeo de transiciones a bolsas de

lugares, $O : T \rightarrow P^{\infty}$ *es la función de salida, un mapeo de transiciones a bolsas de lugares,* $f : T \rightarrow [0, 1]$ *es una función de asociación, un mapeo de transiciones a valores reales entre cero y uno,* $\alpha : P \rightarrow [0, 1]$ *es una función de asociación, un mapeo de lugares a valores reales entre cero y uno,* $\beta : P \rightarrow D$ *es una función de asociación, un mapeo biyectivo de lugares a proposiciones.*

Sea A un conjunto de arcos direccionados. Si $p_j \in I(t_i)$, entonces existe un arco direccionado a_{ji}, $a_{ji} \in A$, del lugar p_j a la transición t_i. Si $p_k \in O(t_i)$, entonces existe un arco direccionado a_{ik}, $a_{ik} \in A$, de la transición t_i al lugar p_k. Si $f(t_i) = \mu_i$, $\mu_i \in [0, 1]$, entonces la transición t_i se dice que esta asociada con un valor real μ_i. Si $\beta(p_i) = d_i$, $d_i \in D$, entonces el lugar p_i se dice estar asociado con la proposición d_i.

Una red de Petri difusa con algunos lugares conteniendo *tokens* es llamada una red de Petri difusa marcada. En una red de Petri difusa marcada el *token* en un lugar p_i está representado por un punto etiquetado $\cdot^{\alpha(p_i)}$.

El valor del *token* en un lugar p_i, $p_i \in P$, está denotado por $\alpha(p_i)$, donde $\alpha(p_i) \in [0, 1]$. Si $\alpha(p_i) = y_i$, $y_i \in [0, 1]$, y $\beta(p_i) = d_i$, entonces indica que el grado de verdad de la proposición d_i es y_i.

Utilizando una red de Petri difusa, la regla de producción difusa

$$R_i : \text{IF } d_j \text{ THEN } d_k(CF = \mu_i)$$

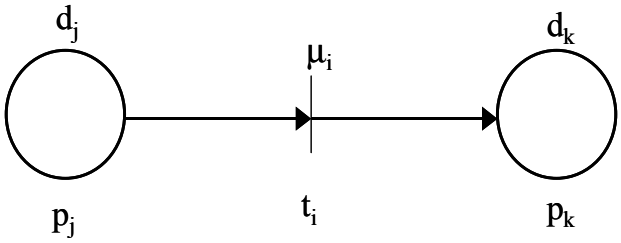

Figura 3-1: Red de Petri Difusa

puede ser modelada como se muestra en la Figura 3-1 y en 3.4.

$$\begin{aligned} FPN &= \{P, Y, D, I, O, f, \alpha, \beta\}, \\ P &= \{p_1, p_2\}, \\ T &= \{t_1\}, \\ D &= \{\text{está caliente, la humedad es baja}\}, \\ I(t_1) &= \{p_1\}, O(t_1) = \{p_2\}, f(t_1) = 0.90, \\ \alpha(p_1) &= 0.90, \alpha(p_2) = 0, \\ \beta(p_1) &= \text{está caliente}, \\ \beta(p_2) &= \text{la humedad es baja}. \end{aligned} \tag{3.4}$$

Una transición t_i está habilitada para disparar, si para todo $p_j \in I(t_i)$, $\alpha(p_j) \geq \lambda$, donde λ es un valor umbral y $\lambda \in [0, 1]$. De acuerdo con [37], el valor del *token* en un lugar de salida de t_i es calculado como se muestra en la Figura 3-2.

Las reglas de producción difusas (ver 3.1) en disparo pueden ser consideradas como transiciones en disparo. De acuerdo con [34], la regla de producción difusa compuesta puede ser distinguida en los siguientes tipos de reglas[1]:

Tipo 1: IF d_{j1} y d_{j2} y ... y d_{jn} THEN $d_k(CF = \mu_i)$. Este tipo de regla es modelado por una red de Petri difusa marcada como se muestra en la Figura 3-3.

[1]Hay que hacer la observación, de que las últimas dos reglas se pueden reducir a las dos primeras.

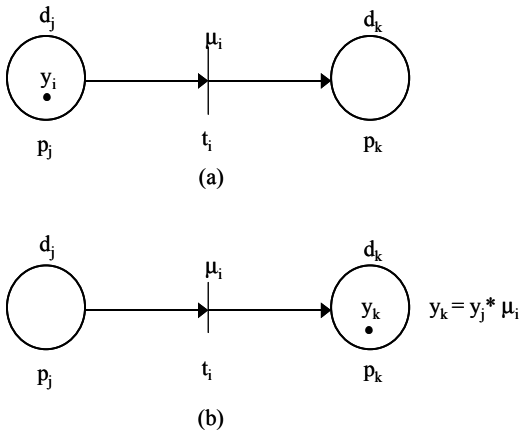

Figura 3-2: Disparo de una red de Petri difusa. (a) Antes de disparar la transición t_i. (b) Despues de disparar la transición t_i.

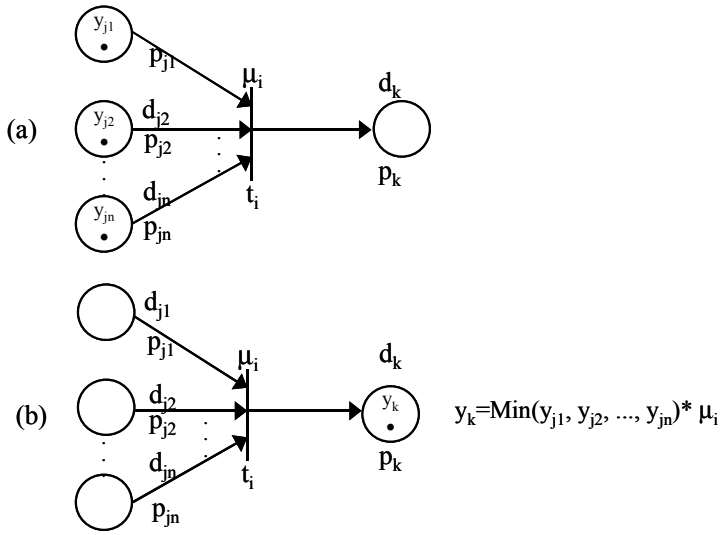

Figura 3-3: Red de Petri difusa marcada del tipo 1. (a) Antes de disparar la transición t_i. (b) Despues de disparar t_i.

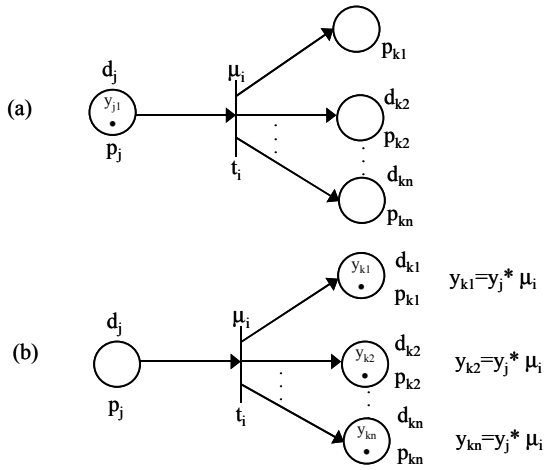

Figura 3-4: Red de Petri difusa marcada del tipo 2. (a) Antes de disparar la transición t_i. (b) Despues de disparar t_i.

Tipo 2: IF d_j THEN d_{k1} y d_{k2} y ... y d_{kn} $(CF = \mu_i)$. Este tipo de regla es modelado por una red de Petri difusa marcada como se muestra en la Figura 3-4.

Tipo 3: IF d_{j1} o d_{j2} o ... o d_{jn} THEN $d_k(CF = \mu_i)$. Este tipo de regla es modelado por una red de Petri difusa marcada como se muestra en la Figura 3-5.

Tipo 4: IF d_j THEN d_{k1} o d_{k2} o ... o d_{kn} $(CF = \mu_i)$. Este tipo de regla puede ser modelado por una red de Petri difusa al igual que los anteriores ejemplos. Las reglas de este tipo son inconvenientes para el control deductivo por que no hacen implicaciones especificas. Este tipo de reglas se mencionaron de forma ilustrativa, ya que no son parte del tema central de la tesis, en lo siguiente no se discutirá este tipo de reglas.

3.3 Red de Petri Difusa Adaptable

Como se mencionó en la introducción, en la representación del conocimiento se han utilizado las redes de Petri difusas (sección *3.2*), pero en el modelado del sistema de conocimiento, se requiere la actualización del conocimiento, por lo que diversos autores han propuesto diversos tipos de modelos. En este caso se presentan las redes de Petri difusas adaptables [33], las cuales introducen el concepto *adaptable* a las redes de Petri difusas.

Red de Petri difusa adaptable (RPDA)

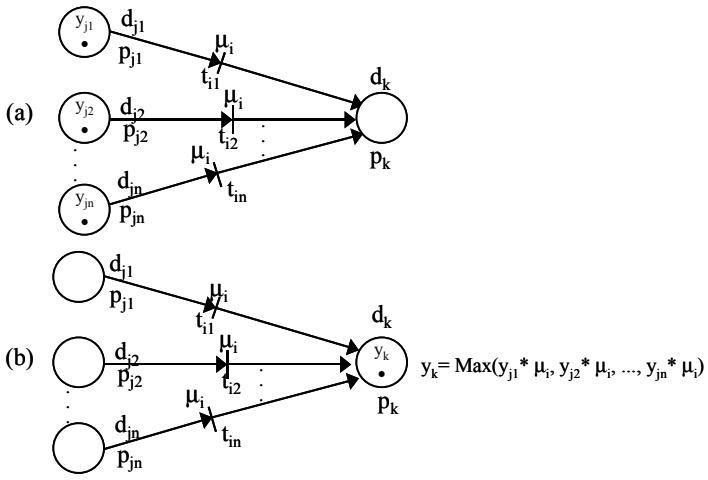

Figura 3-5: Red de Petri difusa marcada del tipo 3. (a) Antes de disparar la transición t_i. (b) Despues de disparar t_i.

Definición 3.2 *[33] Una red de Petri difusa adaptable es una 9-tupla*

$$RPDA = (P, T, D, I, O, \alpha, \beta, Th, W)$$

donde $P, T, D, I, O, \alpha, \beta$, *están definidas al igual que 3.3.* $Th : T \rightarrow [0,1]$ *es la función que asigna un valor umbral* λ_i *desde cero a uno a la transición* t_i, $Th = \{\lambda_1, \lambda_2, ..., \lambda_m\}$. $W = W_I \cup W_O$, $W_I : I \rightarrow [0,1]$ *y* $W_O : O \rightarrow [0,1]$, *son los conjuntos de pesos de entrada y los pesos de salida los cuales asignan valores de pesos a todos los arcos de una red.*

Representación gráfica de una red de Petri difusa adaptable

Las RPDA son el modelo de una clase particular de reglas de producción difusas, estas reglas se les conoce como reglas de producción difusas pesadas (su definición está dada en [33]). En [33], se presentan tres tipos de reglas:

Tipo 1: Una regla de producción difusa simple

R: IF a THEN $c, Th(t) = \lambda, W_O(t, p_j) = \mu, W_I(p_i, t) = w$.

Tipo 2: Una regla conjunta compuesta

R: IF a_1 y a_2 y ... y a_n THEN $c, Th(t) = \lambda, W_O(t, p_j) = \mu, W_I(p_i, t) = w_i, i = 1, ..., n$.

Tipo 3: Una regla disjunta compuesta

R: IF a_1 o a_2 o ... o a_n THEN $c, Th(t_i) = \lambda_i, W_O(t_i, p_j) = \mu_i, W_I(p_i, t_i) = w_i, i = 1, ..., n$.

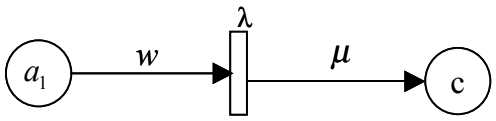

Figura 3-6: Red de Petri difusa adaptable del tipo 1.

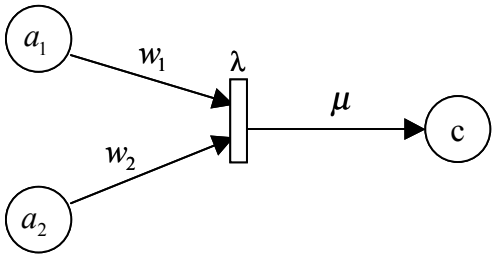

Figura 3-7: Red de Petri difusa adaptable del tipo 2.

Sus gráficas correspondientes se muestran en las Figuras 3-6, 3-7 y 3-8, respectivamente.

Razonamiento difuso utilizando RPDA

Primero, damos algunas definiciones básicas las cuales son útiles para explicar la regla de disparo de la transición de cada RPDA.

Definición 3.3 *(Lugar fuente, lugar pozo) un lugar p es llamado un lugar fuente si no tiene transiciones de entrada. Es llamada un lugar pozo si no tiene transiciones de salida.*

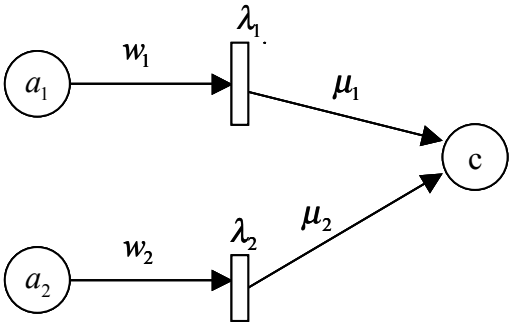

Figura 3-8: Red de Petri difusa adaptable del tipo 3.

Un lugar fuente corresponde a una proposición de precondición en la regla de producción difusa pesada, y un lugar pozole corresponde a la consecuencia. Por ejemplo, en la Figura 3-9, P_1, P_2, P_3 son lugares fuente, P_6 es un lugar pozo.

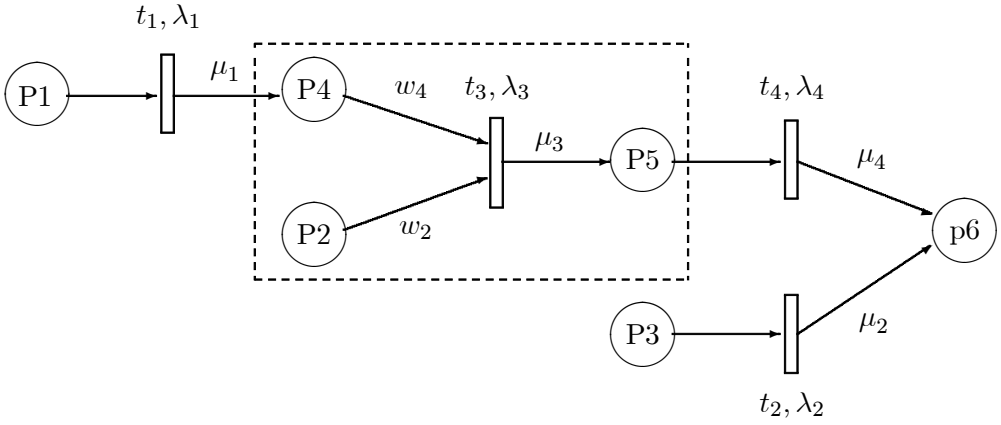

Figura 3-9: RPDA del ejemplo 3.1.

Definición 3.4 *(Ruta) dado un lugar p, una cadena de transiciones $t_1 t_2 \cdots t_n$ es llamada una ruta para p si p puede tener un token a través del disparo de esta cadena de transiciones en secuencia desde un grupo de lugares fuente. Si una cadena de transiciones dispara en secuencia, llamamos a la correspondiente ruta activa.*

Para un lugar p, es posible que existan mas de una ruta. Por ejemplo, en la Figura 3-9, $t_1 t_3 t_4$ es una ruta para P_6, t_2 es otra ruta para ella. Sea $I(t) = \{p_{i1}, p_{i2}, \cdots, p_{in}\}$, w_{I1}, w_{I2}, $\cdots$, w_{In} los correspondientes pesos de entrada a estos lugares, λ_1, λ_2, $\cdots$, λ_n los umbrales. Sea $O(t) = \{p_{o1}, p_{o2}, \cdots, p_{om}\}$, y w_{o1}, w_{o2}, $\cdots$, w_{om} los correspondientes pesos de salida de estos lugares.

Dividimos el conjunto de lugares P en tres partes $P = P_{UI} \cup P_{int} \cup P_O$, donde P es el conjunto de lugares de la RPDA; $P_{UI} = \{p \in P \mid {}^{\cdot}p = \varnothing\}$, $p \in P_{UI}$ es llamado un lugar de entrada usuario; $P_{int} = \{p \in P \mid {}^{\cdot}p \neq \varnothing \ \ and \ \ p^{\cdot} \neq \varnothing\}$, $p \in P_{int}$ es llamado un lugar interior; $P_O = \{p^{\cdot} = \varnothing\}$, $p \in P_O$ es llamado un lugar de salida. Donde $\varnothing$ es un conjunto vacío.

Definición 3.5 *La marcación de un lugar $m(p)$ está definida como el factor de certeza del token dentro.*

Definición 3.6 *$\forall t \in T$, t está habilitada si $\forall p_{Ij} \in I(t)$, $m(p_{Ij}) > 0$, $j = 1, 2, \cdots, n$.*

Definición 3.7 *Cuando t está habilitada, produce un nuevo factor de certeza $CF(t)$*

$$CF(t) = \begin{cases} \sum_j m(p_{Ij}) \cdot w_{Ij}, & \sum_j m(p_{Ij}) \cdot w_{Ij} > Th(t) \\ 0, & \sum_j m(p_{Ij}) \cdot w_{Ij} < Th(t) \end{cases}$$

Podemos utilizar una función continua $CF(t)(x)$ para aproximar $CF(t)$

$$CF(t)(x) = x \cdot F(x)$$

donde

$$x = \sum_j m(p_{Ij}) \cdot w_{Ij}$$

$F(x)$ es una función sigmoide la cual aproxima el umbral de t,

$$F(x) = 1/\left(1 + e^{-b(x-Th(t))}\right)$$

donde b es un instante. Si b es lo suficiente grande, cuando $x > Th(t)$, $e^{-b(x-Th(t))} \approx 0$, entonces $F(x) \approx 1$, y cuando $x < Th(t)$, $e^{-b(x-Th(t))} \to \infty$, entonces $F(x) \approx 0$.

Proposición 3.1 *Si $x > Th(t)$, la transición t dispara, al mismo tiempo, la transmición del token toma lugar, entonces el valor del factor de certeza cambia de acuerdo a los siguientes casos:*

1) Si un lugar p_{ok} tiene solamente una transición de entrada t,un nuevo *token* con factor de certeza $w_{ok} \cdot CF(t)$ es colocado en cada lugar de salida p_{ok}, $k = 1, 2, \cdots, m$, y todos los *tokens* dentro p_{Ij}, $j = 1, 2, \cdots, n$ son removidos.

2) Si un lugar p_{ok} tiene mas de una transición de entrada, y mas de una de ellas dispara, esto es, mas de una ruta estaban activas al mismo tiempo, entonces el nuevo factor de certeza

de p_{ok} es decidido por el centro de gravedad de las transiciones disparadas

$$m(p_{ok}) = \frac{\sum_j [w_{oj} \cdot CF(t_j)]}{\sum_j w_{oj}},$$

donde t_j dispara, $t_j \in {}^{\cdot}p_{ok}$.

De acuerdo a las definiciones anteriores, una transición t está habilitada si todos sus lugares de entrada tienen *tokens*, si el factor de certeza es mayor que su umbral , entonces t dispara, asi una RPDA puede ser implementada. De este modo, a traves de transiciones disparadas, los factores de certeza pueden ser razonados desde un conjunto de proposiciones antecedentes conocido a un conjunto de proposiciones consecuentes paso a paso.

Definición 3.8 *Sea* $T_{inicial} = \{t \in T \mid {}^{\cdot}t \cap P_{UI} \neq \varnothing \;\; y \;\; {}^{\cdot}t \cap P_{int} = \varnothing\}$, $t \in T_{inicial}$ *es llamada una transición habilitada inicialmente.*

Definición 3.9 *Sea* $T_{actual} = \{t \in T_{inicial} \mid \forall p_i \in {}^{\cdot}t, m(p_i) > 0, \, y \, CF(t) > Th(t)\}$, $t \in T_{actual}$ *es llamada una transición habilitada actual.*

Algoritmo de Razonamiento Difuso

Resumiendo, de las definiciones se ha desarrollado un algoritmo que calcula los valores de certeza de un conjunto de proposiciones consecuentes, debido a la ejecución del conjunto de proposiciones antecedentes.

INPUT: los factores de certeza de un conjunto de proposiciones antecedentes (corresponde a P_{UI} en la RPDA)

OUTPUT: los factores de certeza de un conjunto de proposiciones consecuencia (corresponde a $P_{int} \cup P_O$ en la RPDA)

Paso 1. Construya el conjunto de plazas de entrada usuario P_{UI}.

Paso 2. Construya el conjunto de transiciones habilitadas inicialmente $T_{inicial}$.

Paso 3. Encuentre las transiciones habilitadas actuales T_{actual} de acuerdo a la *Definición 3.5.*

Paso 4. Calcular nuevos factores de certeza producidos por el disparo de transiciones de acuerdo a la *Definición 3.6.*

Paso 5. Hacer transmición de token de acuerdo a la *Proposición 3.1.*

Paso 6. Hacer $T = T - T_{actual}$, $P = P - \dot{T}_{actual}$.

Paso 7. Hasta que $T_{actual} = \emptyset$, ir al *Paso 3* y repetir.

3.3.1 Aprendizaje y entrenamiendo de la RPDA

Con el fin de dar la capacidad de aprendizaje (como una red neuronal), en el artículo[32] fue desarrollado un algoritmo de aprendizaje de pesos bajo las siguientes condiciones:

1. Se necesita conocer los factores de certeza de todas las plazas de salidad.
2. Solamente una capa de pesos puede ser aprendida.
3. Para las reglas del tipo 3, si existe más de un disparo de transición, se debe conocer cual transición de entrada es el *token* contribuyente del lugar de salida.
4. En el caso 2 de la proposición, conocemos los pesos de salida.

Estas condiciones son restrictivas, dado que esta información puede no estar disponible en sistemas expertos reales. En esta sección llevaremos estas condiciones a casos mas generales. La idea principal es que todas las capas de pesos puedan ser actualizadas mediante el algoritmo de retropropagación, si los factores de certeza de todos los lugares pozo están dados.

Algoritmo de Retropropagación

Para el desarrollo del algoritmo de retropropagación se hacen las siguientes suposiciones

- El modelo RPDA de un sistema experto ha sido desarrollado.
- En el modelo RPDA, Th y W_O son conocidos.
- Un conjunto de factores de certeza P_{UI} y P_O está dado.

Aqui tomamos la regla de producción difusa pesada de tipo 2 como una ilustración para mostrar el proceso de aprendizaje de conocimiento utilizado en la RPDA. La regla de tipo 2 puede ser transladada en una RPDA. Esta estructura RPDA puede ser transladada a una estructura de red neuronal, definiendo a G como

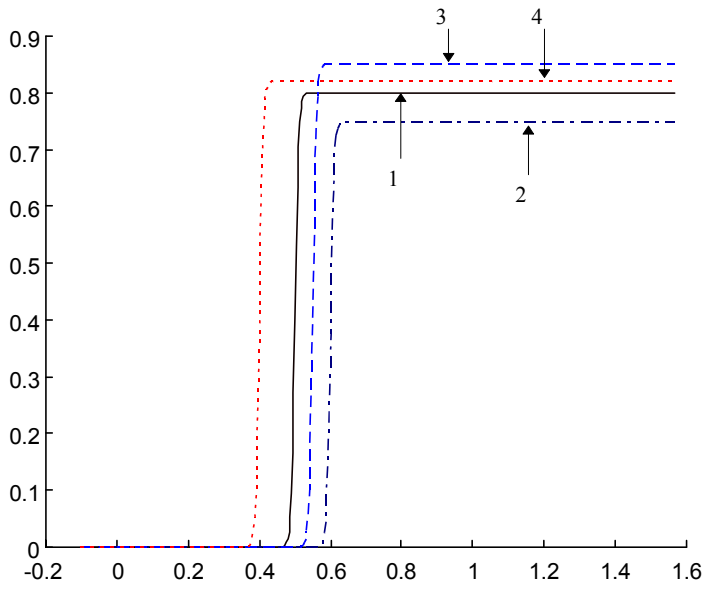

Figura 3-10: Funciones sigmiodes del Ejemplo 3.1.

$$G(x) := f(x) \cdot x, \tag{3.5}$$

con $x = W^T\Lambda = \sum_{i=1}^{n} \alpha_i w_i$; $f(x) = \frac{\mu}{1+e^{-b(x-\lambda)}}$ es una función sigmoide, b es una constante la cual se ajusta a $f(x)$, W es el vector de pesos $W := [w_1, w_2 \cdots w_n]^T$, α es el vector de salida de la capa previa, $\Lambda := [\alpha_1, \alpha_2 \cdots \alpha_n]^T$. La función $f(x)$ continua puede aproximar un umbral si μ, b y λ son valores apropiados seleccionables, ver No. 1 en la Figura 3-10 donde $\mu = 0.8$, $b = 200$ y $\lambda = 0.5$.

Para un lugar p, existen algunas rutas de aprendizaje las cuales pertenecen a un conjunto de lugares fuente. Los pesos en estas rutas pueden ser entrenados de acuerdo el algoritmo de retropropagación desarrollado en esta sección. A lo largo de la ruta seleccionada Ω, el proceso de propagación postalimentado (una capa oculta), es dada una información de entrada U y los pesos fijos w_i, la salida $O(\Omega)$ puede exprezarse

$$O(\Omega) = G^{(n)}[W^{(n)}G^{(n-1)}(W^{(n-1)}G^{(n-2)} \cdots W^{(2)}G^{(1)}\left(W^{(1)}U\right))].$$

donde $G^{(k)}$ $(k = 1 \cdots n)$ es la función activa de la k-ésima capa, $W^{(k)}$ $(k = 1 \cdots n)$ es el peso de la k-ésima capa.

Si la información real es O^*, el vector de error de salida es

$$e_n = O(\Omega) - O^*.$$

Mientras no procesemos los *tokens* en la capa de salida, esta capa puede ser seleccionada con la regla del centro de gravedad (ver *Proposición 3.1*), esto es,

$$G^{(n)}(x) = \frac{x}{\sum_j w_{oj}}. \tag{3.6}$$

El algoritmo de aprendizaje es el mismo que el de retropropagación de las redes neuronales multicapa:

- Los pesos en la capa de salida se actualizan como

$$W^{(n)}(k+1) = W^{(n)}(k) - \gamma_i \dot{G}^{(n)} e_n \Lambda^{(n)},$$

  donde $\Lambda^{(n)}$ es la entrada a la $n-$ésima capa, $\gamma_i > 0$ es la ganancia adaptiva, $W(k)$ es el peso al tiempo k.

- Los errores en las capas ocultas y la capa de entrada se propagan hacia atrás como

$$e_i = e_{i+1} \dot{G}^{(i)} W^{(i)} \tag{3.7}$$

  los pesos son actualizados como

$$\begin{aligned} W^{(n-1)}(k+1) &= W^{(n-1)}(k) - \gamma_{n-1} \dot{G}^{(n-1)} e_{n-1}\Lambda^{(n-1)} \\ &\vdots \\ W^{(2)}(k+1) &= W^{(2)}(k) - \gamma_2 \dot{G}^{(2)} e_2 \Lambda^{(2)} \\ W^{(1)}(k+1) &= W^{(1)}(k) - \gamma_1 \dot{G}^{(1)} e_1 \Lambda^{(1)} \end{aligned} \tag{3.8}$$

donde $\dot{G}$ es la derivada de la función no lineal G [**?**],

$$\dot{G} := \frac{d}{dx}\left[\frac{\mu \cdot x}{1+e^{-b(x-\lambda)}}\right] = \frac{\mu x b e^{-b(x-\lambda)}}{\left(1+e^{-b(x-\lambda)}\right)^2} + \frac{\mu}{1+e^{-b(x-\lambda)}}. \tag{3.9}$$

Algoritmo de Aprendizaje Retropropagación

Finalmente el algoritmo de aprendizaje de la RPDA se puede resumir en

Paso 1. Seleccionar un conjunto de pesos iniciales.

Paso 2. Para cada conjunto de datos de entrada, encontrar todas las rutas activas, y marcarlas $\Omega_1, \Omega_2, \cdots, \Omega_k$.

Paso 3. Seguir cada ruta activa, de acuerdo al algoritmo de razonamiento, calcular la correspondiente salida.

Paso 4. Conjuntar la diferencia entre la salida ideal y la salida calculada como el error e, seleccionar γ_n usar (3.8) para ajustar los pesos en estas rutas.

3.4 Simulación

En esta sección, seleccionamos dos ejemplos típicos para mostrar los resultados de las secciones anteriores.

Ejemplo 3.1 *P1, P2, P3, P4, P5 y P6 son proposiciones relacionadas de un sistema experto* Γ_1. *Entre ellas existen las siguientes reglas de producción difusas pesadas.*

R1: si P1 THEN P4 (λ_1, μ_1)

R2: si P2 AND P4 THEN P5 (w_2, w_4, λ_3, μ_3)

R3: si P3 OR P5 THEN P6 (λ_2, λ_4, μ_2, μ_4)

Este ejemplo incluye los tres tipos de reglas, en la cual R1 es una simple regla de producción difusa pesada, R2 es una regla conjunta compuesta, y R3 es una regla disjunta compuesta. Queremos mostrar el racionamiento difuso y el algoritmo de aprendizaje de pesos.

Primero, basado en el principio de translación, mapeamos Γ_1 en una RPDA como sigue (ver Figura 3-9),

$$AFPN_1 = \{P, T, D, I, O, \alpha, \beta, Th, W\},$$

donde $P = \{p_1,\ p_2,\ p_3,\ p_4,\ p_5,\ p_6\}$, $T = \{t_1,\ t_2,\ t_3,\ t_4\}$, $D = \{P1, P2, P3, P4, P5, P6\}$, $Th = \{\lambda_1,\ \lambda_2,\ \lambda_3,\ \lambda_4\}$, $W_I = \{w_2,\ w_4\}$, $W_O = \{\mu_1,\ \mu_2,\ \mu_3,\ \mu_4\}$.

Tenemos tres proposiciones de entrada (P1, P2 y P3) y tres proposiciones consecuencia (P4, P5 y P6). Los datos están dados como

$$\begin{aligned} \mu_1 = 0.80, \quad \mu_2 = 0.75, \quad \mu_3 = 0.85, \quad \mu_4 = 0.82 \\ \lambda_1 = 0.50, \quad \lambda_2 = 0.60, \quad \lambda_3 = 0.55, \quad \lambda_4 = 0.40 \\ w_4 = 0.63, \quad w_2 = 0.37, \end{aligned}$$

los cuales fueron obtenidos por simulación y hacen de la función sigmoide una aproximación a un escalón (umbral), ver la Figura 3-10.

Utilizamos 4 funciones sigmoide

$$F_i(x) := \frac{\mu_i}{1 + e^{-b_i(x-\lambda_i)}}, \quad i = 1, 2, 3, 4$$

para aproximar los 4 umbrales $\lambda_1, \lambda_2, \lambda_3, \lambda_4$, las pendientes b_i son seleccionadas como 200 (ver Figura 3-10). Para la transición t_3 (ver Figura 3-9), el argumento de la función $G(x)$ es

$$x = \alpha(P4)w_4 + \alpha(P2)w_2.$$

Usando el algoritmo de razonamiento difuso, un conjunto de datos de salida (factores de certeza de proposiciones consecuencia) pueden ser calculados de acuerdo a los datos de entrada (factores de certeza de proposiciones antecedente). La Tabla 1 da el resultado de la RPDA.

Tabla *3.2*

Group No.	α(P1)	α(P2)	α(P3)	α(P4)	α(P6)	α(P6)
1	0.2190	0.0470	0.6789	0	0	0.3243
2	0.6793	0.9347	0.3835	0.5434	0.5850	0.3055
3	0.5194	0.8310	0.0346	0.4072	0.4517	0.2359
4	0.0535	0.5297	0.6711	0	0	0.3206
5	0.0077	0.3834	0.0668	0	0	0
6	0.4175	0.6868	0.5890	0	0	0.0279
7	0.9304	0.8462	0.5269	0.7443	0.6647	0.3472
8	0.0920	0.6539	0.4160	0	0	0
9	0.7012	0.9103	0.7622	0.5610	0.5867	0.6705
10	0.2625	0.0475	0.7361	0	0	0.3516

Uno puede ver que algunos datos son 0. Esto significa que los umbrales correspondientes no fueron pasados. Por ejemplo en el Grupo 1, α(P1) = 0.2190, el umbral λ_1 es 0.50. Mientras que $\alpha(\text{P1}) < \lambda_1$, la transición t_1 no puede disparar, así, de acuerdo a R1 el factor de certeza de salida α(P4) es 0. El uso de una función sigmoide para aproximar un umbral significa que el cero exacto es imposible de obtener, pero si el coeficiente de la pendiente es suficientemente pequeño, la función sigmoide aproxima al umbral con buena exactitud.

Si los pesos son deconocidos, la técnica de redes neuronales puede utilizarse para estimar los pesos. La parte de aprendizaje de la RPDA (ver el bloque resaltado en la Figura 3-9) puede ser formada como una red neuronal estandar de una sola capa, ver Figura 3-11. Proponiendo que los pesos ideales sean

$$W_2 = 0.37, W_4 = 0.63.$$

La función sigmoide es

$$F_3(x) := \frac{0.85}{1 + e^{-200(x-0.55)}} \tag{3.10}$$

Si las entradas α(P1) y α(P2) están dadas aleatoriamente desde 1 a 0, podemos obtener la salida real α(P5) de acuerdo al sistema experto Γ_1. Dada una condición inicial para w_2

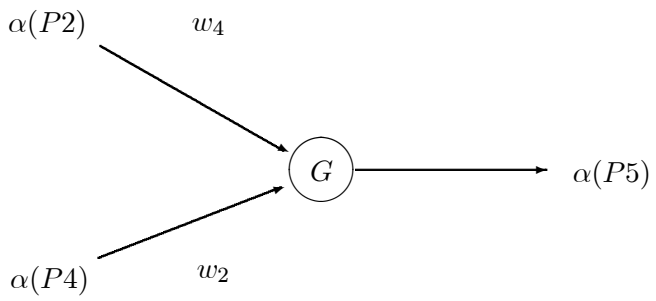

Figura 3-11: La translación de la red neuronal de la parte de aprendizaje del Ejemplo 3.1.

y, w_4 ponemos las mismas entradas a la red neuronal. El error entre la salida de la red neuronal ($\alpha^{'}$(P5)) y la del sistema experto $\Gamma_1(\alpha(\text{P5}))$ puede ser usado para modificar los pesos, podemos utilizar la siguiente ley de aprendizaje

$$\begin{aligned} W(k+1) = W(k) + \dot{G}\, \delta e(k) \Lambda(P(k)) \quad \delta > 0 \\ e(k) := \alpha(P5)(k) - \alpha^{'}(P5)(k), \end{aligned} \tag{3.11}$$

la cual es el gradiente descendiente, ya que la translación es una red neuronal de una sola capa, no hay necesidad de utilizar el algoritmo de retropropagación, aqui δ es la taza de aprendizaje, una pequeña δ puede asegurar la estabilidad del proceso de aprendizaje. Seleccionamos $\delta = 0.07$, $W(k) = [w_2(k), w_4(k)]$, $\Lambda(P(k)) = [\alpha(P2(k)), \alpha(P4(k))]$, y

$$G(x) = \frac{0.85x}{1 + e^{-200(x-0.55)}}.$$

Después del proceso de entrenamiento los pesos convergen a valores reales, esto se observa en la gráfica obtenida de la simulación donde w_2 y, w_4 oscilan hasta estabilizarce en $k > 400$, Figura 3-12.

En este ejemplo solamente hay una capa de aprendizaje. El ejemplo 2 mostrará un caso más complicado donde dos capas de aprendizaje (perceptrón multicapa) es utilizado.

Ejemplo 3.2 *P1, P2, P3, P4, P5, P6, P7, P8, P9, P10, P11 y P12 son proposiciones relacionadas y un sistema experto* Γ_2. *Existen las siguientes reglas de producción difusas pesadas.*

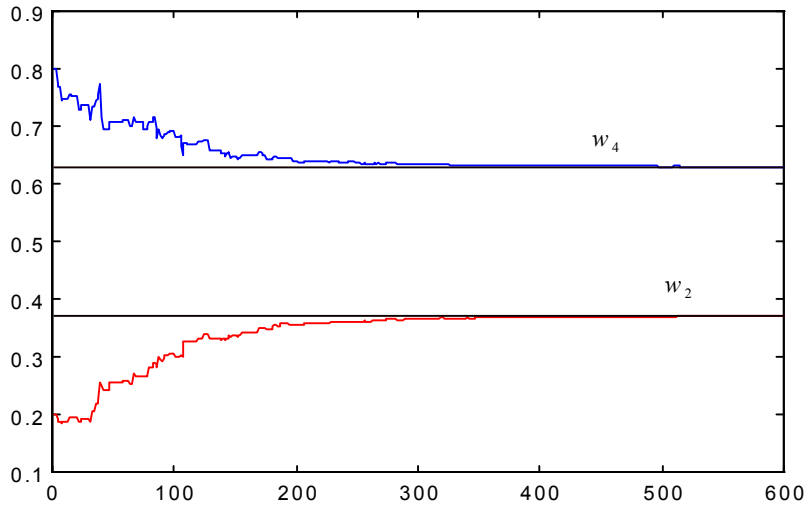

Figura 3-12: Resultados de una capa de aprendizaje del Ejemplo 3.1.

R1: si P1 AND P2 AND P3 THEN P8 (w_1, w_2, w_3, λ_1, μ_1)

R2: si P4 AND P5 THEN P9 (w_4, w_5, λ_2, μ_2)

R3: si P6 AND P8 THEN P10 (w_6, w_8, λ_5, μ_5)

R4: si P7 OR P9 THEN P11 (λ_3, λ_4, μ_3, μ_{42})

R5: si P9 OR P10 THEN P12 (λ_4, λ_6, μ_{42}, μ_6)

Basado en el principio de translación, mapeamos Γ_2 en una RPDA (ver Figura 3-13).

$$AFPN_2 = \{P, T, D, I, O, \alpha, \beta, Th, W\},$$

donde

$P = \{p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_9, p_{10}, p_{11}, p_{12}\}$,

$T = \{t_1, t_2, t_3, t_4, t_5, t_6\}$,

$D = \{$P1, P2, P3, P4, P5, P6, P7, P8, P9, P10, P11, P12$\}$,

$Th = \{\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5, \lambda_6\}$,

$W_I = \{w_1, w_2, w_3, w_4, w_5, w_6, w_8\}$,

$W_O = \{\mu_1, \mu_2, \mu_3, \mu_{41}, \mu_{42}, \mu_5, \mu_6\}$.

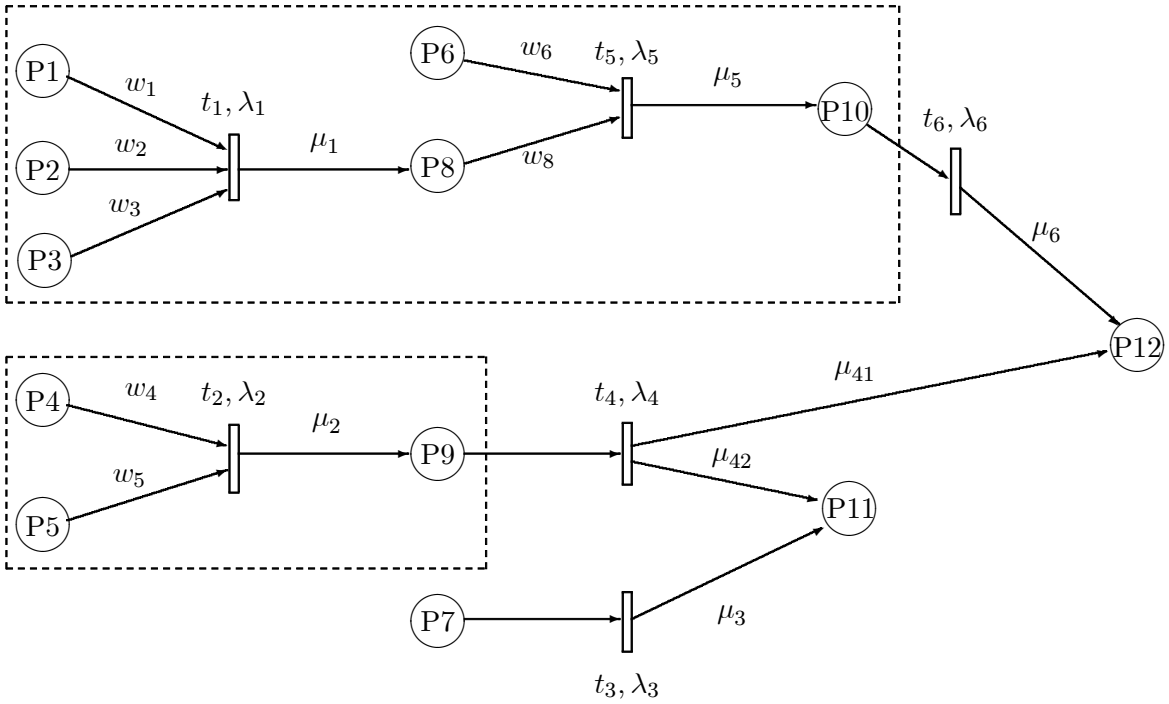

Figura 3-13: RPDA del ejemplo 3.2.

Así, el modelo de RPDA para este sistema experto puede ser representado como en la Figura 3-13, los dos bloques resaltados son las partes de aprendizaje. Este modelo de RPDA puede ser transferido en una red neuronal como en la Figura 3-14.

Mientras los pesos de μ_{41}, μ_{42}, μ_3 y μ_6 son conocidos, podemos simplificar esta red neuronal compleja en dos redes neuronales simples: NN1 y NN2. Aquí la sub red NN1 es de una sola capa, la sub red NN2 es multicapa. Las redes neuronales correspondientes a G_5 son fijas.

Podemos entrenar las 2 redes independientemente. El error original de aprendizaje es e_{12}. porque la función de salida es seleccionada como (3.6)

$$G^{(4)}(x) = \frac{x}{\mu_{41} + \mu_6}, \quad x = \mu_{41}\alpha(P9) + \mu_6\alpha(P10).$$

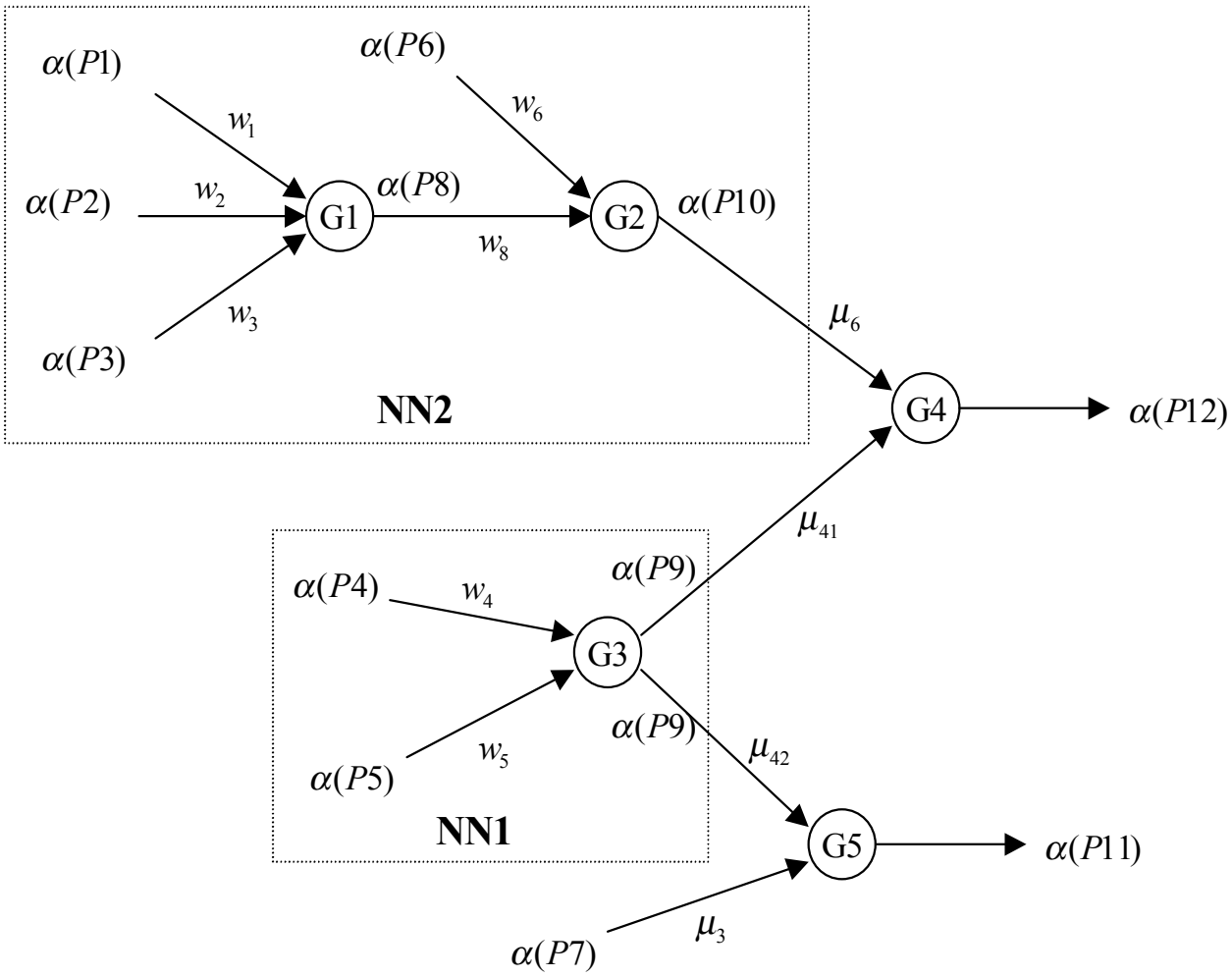

Figura 3-14: La translación a red neuronal de la RPDA de la figura 3-13.

- En el caso 1 de la Definición 7, si solamente t_4 dispara, entonces:

$$e_9 = \mu_{41} e_{12}, \quad e_{10} = 0$$

si solamente t_6 dispara, entonces:

$$e_{10} = \mu_6 e_{12}, \quad e_9 = 0$$

- En el caso 2 de la definición 7, cuando t_4 y t_6 disparan al mismo tiempo, de acuerdo a la regla del error de retropropagación (3.7)

$$e_9 = e_{12} \times \frac{1}{\mu_{41}+\mu_6} \times \mu_{41} = \frac{\mu_{41}}{\mu_{41}+\mu_6} e_{12}$$
$$e_{10} = e_{12} \times \frac{1}{\mu_{41}+\mu_6} \times \mu_6 = \frac{\mu_6}{\mu_{41}+\mu_6} e_{12}$$

Los algoritmos de aprendizaje para redes neuronales de una sola capa son los mismos que en el ejemplo 3.1. La ley de adaptación para el perceptrón multicapa es como en la ecuación(3.8).

Estudiemos la segunda parte de aprendizaje, esto es, NN2 en la Figura 3-14. Proponemos que los pesos ideales son

$$W_1 = 0.53, \quad W_2 = 0.17, \quad W_3 = 0.9, \quad W_8 = 0.37, \quad W_6 = 0.68.$$

Un conjunto de datos acerca de la parte de aprendizaje de la RPDA

$$\lambda_1 = 0.5, \quad \lambda_5 = 0.4, \quad \mu_1 = 0.9, \quad \mu_5 = 0.8, \quad b_1 = b_2 = 20.$$

Dado un conjunto de valores iniciales de los pesos

$$w_1(1) = 0.8, \quad w_2(1) = 0.2, \quad w_3(1) = 0.5, \quad w_8(1) = 0.7, \quad w_6(1) = 0.1,$$

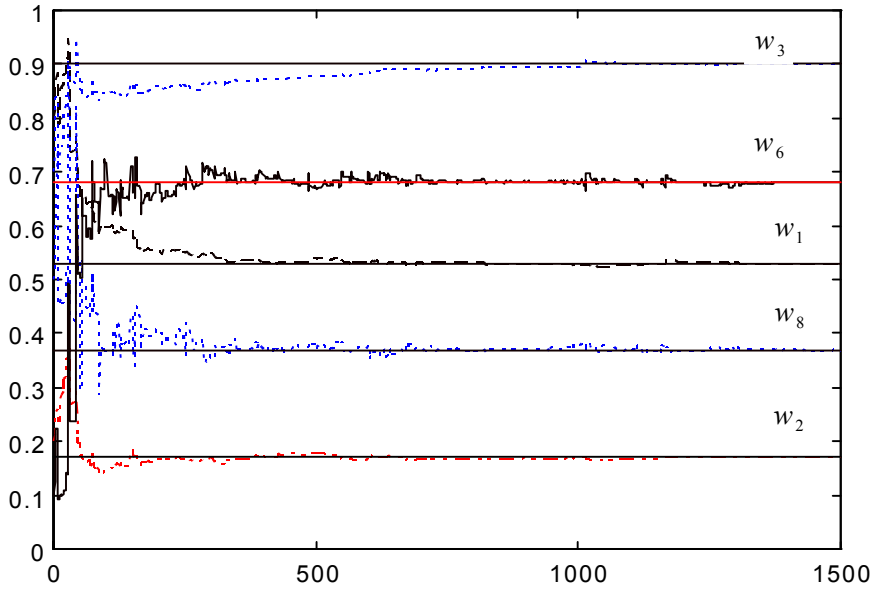

Figura 3-15: Resultados del aprendizaje del PMC del Ejemplo 3.2.

y la taza de aprendizaje $\delta = 0.5$. Los resultados del aprendizaje en línea del PMC se muestran en la Figura 3-15.

De estos dos ejemplos, podemos ver que el algoritmo de razonamiento difuso y el algoritmo de retropropagación son muy efectivos si no conocemos los pesos de la RPDA. Después de un proceso de entrenamiento, podemos tener un excelente mapeo entrada salida del sistema de conocimiento.

3.5 Conclusión

Este capítulo presentó una red de Petri difusa modificada: La Red de Petri Difusa Adaptiva (RPDA) la cual tiene la habilidad de aprendizaje como una red neuronal. De este hecho, el modelo aventaja a los demás modelos citados en la literatura, en que puede hacer frente a los cambios dados en un sistema experto. Por el contrario su debilidad redica en que requiere de ciertos conocimientos a priori para funcionar (caso del algortitmo de retropropagación). Siendo la idea mostrada en este capítulo una nueva manera formal de resolver el problema

del aprendizaje de conocimiento en sistemas expertos, surge el cuestionamiento, al dotar a una red de Petri de la habilidad de aprendizaje ¿no sería mejor utilizar una red neuronal? dado que ya es un campo ampliamente estudiado. ¿Por que no incluir un arreglo neuronal en la red de Petri? o más general ¿se podria dar una combinación de manera tal que tengamos una red híbrida, compuesta de una red de Petri y una red neuronal?

En el capítulo siguiente se hace la propuesta novedosa de tal arreglo, para construir un modelo que pueda representar un sistema híbrido.

Capítulo 4

Modelado de Sistemas Híbridos mediante Redes de Petri Difusas y Redes Neuronales

4.1 Introducción

Uno de los esfuerzos mas recientes y mas intensos en la teoría de control se centra en el manejo de sistemas dinámicos que incluyen tanto al proceso continuo como a su mecanismo supervisorio. La visíon que se adopta en la tesis sobre los sistemas *híbridos,* es considerarlos como la combinación de plantas continuas y plantas de eventos-discretos. Los sistemas híbridos han sido intensamente estudiados en los últimos años pora establecer sus fundamentos matemáticos [26] y su diseño ingenieril [25]. Recientes resultados muestran que las técnicas de identificación con redes neuronales han sido muy efectivas para identificar una extensa clase de plantas continuas cuando no se tiene la información del modelo, o está incompleta [22]. Por otro lado, las redes de Petri (RP) son extensamente utilizadas para sistemas de eventos-discretos. Las redes de Petri han sido desarrolladas para representar y analizar en forma sencilla los sistemas de procesamiento de información que son caracterizados por ser concurrentes, asíncronos, distribuidos, paralelos, nodeterminísticos, y/o estocásticos [16]. Además, la aproximación con RP puede ser facilmente combinada con

otras técnicas y teorías; programación, teoría difusa, redes neuronales, etc. Estas RP modificadas son ampliamente empleadas en computación, manufactura, robótica, sistemas basados en conocimiento, control de procesos, asi como otros tipos de aplicaciones en ingeniería.

Las RP poseen la cualidad de representar la lógica clásica de una manera intuitiva y visual. Muchos resultados prueban que las redes de Petri difusas (RPD) son apropiadas para la representación y el razonamiento de relaciones de implicación de lógica borrosa [11], [50], [14], [35], [42], [51], [33]. Es por estas razones, que en el modelado de sistemas híbridos se ha recurrido al uso de modelos basados en RP (Capítulo *2,* Sección *2.3*). Dividiendo al sistema híbrido en dos niveles: nivel lógico (supervisor) y el nivel de proceso continuo, la parte lógica puede ser modelada mediante la red de Petri difusa y la parte continua podría ser modelada por redes neuronales. Estas son nuestras motivaciones para este capítulo: combinar redes de Petri y redes neuronales para modelar sistemas híbridos.

4.2 Modelado de la dinámica continua con Redes Neuronales

Considere un proceso dinámico descrito por

$$\dot{\mathbf{x}}_t = f(\mathbf{x}_t, \mathbf{u}_t), \tag{4.1}$$

donde $x_t \in \Re^n$ es el estado, $u_t \in \Re^n$ es el vector de entrada. $f : \Re^n \times \Re^n \rightarrow \Re^n$ es localmente Lipschitz. Permitámonos considerar la siguiente red neuronal dinámica para identificar el proceso (4.1)

$$\dot{\widehat{\mathbf{x}}}_t = A\mathbf{x}_t + W_{1,t}\sigma(\widehat{\mathbf{x}}_t) + W_{2,t}\phi(\widehat{\mathbf{x}}_t)\mathbf{u}_t, \tag{4.2}$$

aqui $\widehat{\mathbf{x}}_t \in \Re^n$ es el estado de la red neuronal, $A \in \Re^{n\times n}$ es una matriz estable. $W_{1,t} \in \Re^{n\times n}$ y $W_{2,t} \in \Re^{n\times n}$ son las matrices de pesos. El campo vectorial $\sigma(\mathbf{x}_t) : \Re^n \rightarrow \Re^n$ se asume que tiene elementos monotónicamente crecientes. La función $\phi(\cdot)$ es la transformación de $\Re^n$ a

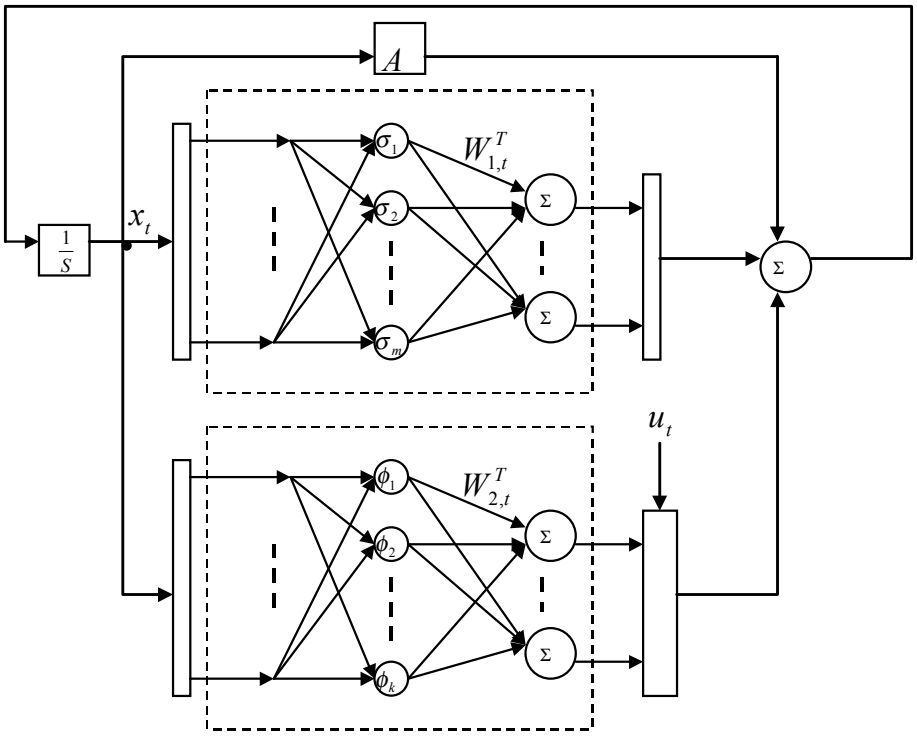

Figura 4-1: Estructura de la red neuronal dinámica.

$\Re^{n\times n}$. La presentación de los elementos $\sigma_i(\cdot)$ y $\phi_{ij}(.)$ es como funciones sigmoide

$$\sigma_i(\widehat{x}_{i,t}) = \frac{a_i}{(1+e^{-b_i\widehat{x}_{i,t}})} - c_i.$$

La estructura de las redes neuronales dinámicas (4.2) se muestran en la Figura 4-1. Este tipo de redes neuronales dinámicas al tener retroalimentación tienen poderosas capacidades de representación, pero tienen el inconveniente cuendo se utilizan redes multicapa, no se puede garantizar la estabilidad del error de identificación. Esto ha sido discutido en [52], [28] y [53], en los cuales se establecen leyes de aprendizaje que garantizan la estabilidad del error.

Basado en [53], una red neuronal dinámica (4.2) no puede seguir al sistema no lineal (4.1) exactamente, representando al sistema no lineal como

$$\dot{\mathbf{x}}_t = A\mathbf{x}_t + W_1^*\sigma(\mathbf{x}_t) + W_2^*\phi(\mathbf{x}_t)\pi(\mathbf{u}_t) - \widetilde{f}_t, \tag{4.3}$$

donde W_1^* y W_2^* son matrices óptimas las cuales pueden minimizar el error de modelado $\widetilde{f}_t$.

Están acotadas por

$$W_1^* \Lambda_1^{-1} W_1^{*T} \leq \overline{W}_1, \quad W_2^* \Lambda_2^{-1} W_2^{*T} \leq \overline{W}_2. \tag{4.4}$$

Como $\sigma(\cdot)$ y $\phi(\cdot)$ han sido seleccionadas como funciones sigmoide, satisfacen la condición de Lipschitz,

$$\widetilde{\sigma}_t^T \Lambda_1 \widetilde{\sigma}_t \leq \Delta_t^T \Lambda_\sigma \Delta_t, \quad \widetilde{\phi}_t^T \Lambda_2 \widetilde{\phi}_t \leq \overline{u} \Delta_t^T \Lambda_\phi \Delta_t, \tag{4.5}$$

donde $\widetilde{\sigma}_t := \sigma(\widehat{\mathbf{x}}_t) - \sigma(\mathbf{x}_t)$, $\widetilde{\phi}_t := \phi(\widehat{\mathbf{x}}_t)\mathbf{u}_t - \phi(\mathbf{x}_t)\mathbf{u}_t$, Λ_1, Λ_2, Λ_σ y Λ_ϕ son matrices definidas positivas. Se define el error de identificación como

$$\Delta_t := \widehat{\mathbf{x}}_t - \mathbf{x}_t,$$

su dinámica se obtiene de (4.2) y (4.3)

$$\dot{\Delta}_t = A\Delta_t + \widetilde{W}_{1,t}\sigma(\widehat{\mathbf{x}}_t) + \widetilde{W}_{2,t}\phi(\widehat{\mathbf{x}}_t)\mathbf{u}_t + W_1^*\widetilde{\sigma}_t + W_2^*\widetilde{\phi}_t + \widetilde{f}_t, \tag{4.6}$$

donde $\widetilde{W}_{1,t} := W_{1,t} - W_1^*$, $\widetilde{W}_{2,t} := W_{2,t} - W_2^*$.

El siguiente Teorema establece el procedimiento de aprendizaje de un identificador neuronal.

Teorema 4.1 *Si el error de modelado $\widetilde{f}_t$ es acotado ($\widetilde{f}_t^T \Lambda_f^{-1} \widetilde{f}_t \leq \overline{\eta}$) y los pesos $W_{1,t}$ y $W_{2,t}$ de la red neuronal dinámica definida por (4.2) son actualizados como*

$$\dot{W}_{1,t} = -K_1 \sigma(\widehat{\mathbf{x}}_t)\Delta_t^T, \quad \dot{W}_{2,t} = -K_2 \phi(\widehat{\mathbf{x}}_t)\mathbf{u}_t \Delta_t^T, \tag{4.7}$$

donde K_1 y K_2 son matrices positivas, entonces el error de identificación es acotado y satisface

$$\limsup_{T\to\infty} \frac{1}{T} \int_0^T \|\Delta_t\|_{Q_0}^2 \, dt \leq \overline{\eta}, \quad Q_0 = Q_0^T > 0, \tag{4.8}$$

donde Q_0 es unamatriz definida positiva yla norma está dada por $\|\Delta_t\|^2_{Q_0} = \Delta_t^T Q_0 \Delta_t$.

Prueba. Seleccionar una función de Lyapunov S_t como

$$S_t = \Delta_t^T P \Delta_t + tr\left\{\widetilde{W}_{1,t}^T k_1^{-1} \widetilde{W}_{1,t}\right\} + tr\left\{\widetilde{W}_{2,t}^T k_2^{-1} \widetilde{W}_{2,t}\right\},$$

donde P, k_1, $k_2 \in \Re^{n\times n}$ son matrices definidas positivas, $tr\{\cdot\}$ es la *traza* y se define como la suma de todos los elementos de la diagonal de una matriz. De acuerdo con (4.2), su derivada es

$$\begin{aligned}\dot{S}_t =& \Delta_t^T \left(PA + A^T P\right) \Delta_t + 2\Delta_t^T P \widetilde{f}_t + 2\Delta_t^T P \left[W_1^* \widetilde{\sigma}_t + W_2^* \widetilde{\phi}_t\right] \\ &+2\Delta_t^T P \widetilde{W}_{1,t} \sigma(\widehat{\mathbf{x}}_t) + 2\Delta_t^T P \widetilde{W}_{2,t} \phi(\widehat{\mathbf{x}}_t) \mathbf{u}_t + 2tr\left\{\dot{\widetilde{W}}_{1,t}^T k_1^{-1} \widetilde{W}_{1,t}\right\} + 2tr\left\{\dot{\widetilde{W}}_{2,t}^T k_2^{-1} \widetilde{W}_{2,t}\right\}.\end{aligned} \tag{4.9}$$

Utilizando la desigualdad matricial

$$X^T Y + \left(X^T Y\right)^T \leq X^T \Lambda^{-1} X + Y^T \Lambda Y \tag{4.10}$$

donde $X, Y, \Lambda \in \Re^{n\times k}$ son matrices, además $\Lambda = \Lambda^T > 0$ (matriz simétrica definida positiva), de (4.5) podemos estimar el escalar $2\Delta_t^T P W_1^* \widetilde{\sigma}_t$ como

$$2\Delta_t^T P W_1^* \widetilde{\sigma}_t \leq \Delta_t^T P W_1^* \Lambda_1^{-1} W_1^{*T} P \Delta_t + \widetilde{\sigma}_t^T \Lambda_1 \widetilde{\sigma}_t \leq \Delta_t^T \left(P \overline{W}_1 P + D_\sigma\right) \Delta_t. \tag{4.11}$$

Tenemos similarmente que

$$2\Delta_t^T P W_2^0 \widetilde{\phi}_t \leq \Delta_t^T \left(P \overline{W}_2 P + \overline{u} D_\phi\right) \Delta_t.$$

Adicionando y sustrayendo $\Delta_t^T Q_0 \Delta_t$ en (4.9) obtenemos

$$\begin{aligned}\dot{S}_t \leq& \Delta_t^T \left[PA + A^T P + P\left(\overline{W}_1 + \overline{W}_2\right) P + \left(D_\sigma + \overline{u} D_\phi + Q_0\right)\right] \Delta_t - \Delta_t^T Q_0 \Delta_t + 2\Delta_t^T P \widetilde{f}_t \\ &+2tr\left\{\dot{\widetilde{W}}_{1,t}^T k_1^{-1} \widetilde{W}_{1,t}\right\} + 2\Delta_t^T P \widetilde{W}_{1,t} \sigma(\widehat{\mathbf{x}}_t) + 2tr\left\{\dot{\widetilde{W}}_{2,t}^T k_2^{-1} \widetilde{W}_{2,t}\right\} + 2\Delta_t^T P \widetilde{W}_{2,t} \phi(\widehat{\mathbf{x}}_t) \mathbf{u}_t.\end{aligned}$$

Definamos $R := \overline{W}_1 + \overline{W}_2$, $Q := D_\sigma + \overline{u}D_\phi + Q_0$. Mientras A es una matriz estable, sabemos que es posible seleccionar una matriz definida positiva Q_0 tal que la ecuación matricial de Riccati

$$A^T P + PA + PRP + Q = 0, \tag{4.12}$$

tenga una solución positiva $P = P^T > 0$. Si definimos $K_1 := k_1 P$, $K_2 := k_2 P$, la ley de actualización (4.7) y la ecuación matricial de Riccati (4.12) implican

$$\dot{S}_t \leq -\Delta_t^T Q_0 \Delta_t + 2\Delta_t^T P \widetilde{f}_t. \tag{4.13}$$

En vista de la desigualdad matricial (4.10),

$$2\Delta_t^T P \widetilde{f}_t \leq \Delta_t^T P \Lambda_f P \Delta_t + \widetilde{f}_t^T \Lambda_f^{-1} \widetilde{f}_t.$$

Asi la ecuación (4.13) se vuelve

$$\begin{aligned} \dot{S}_t &\leq -\Delta_t^T Q_0 \Delta_t + 2\Delta_t^T P \widetilde{f}_t \\ &\leq -\lambda_{\min}(Q_0) \|\Delta_t\|^2 + \Delta_t^T P \Lambda_f P \Delta_t + \widetilde{f}_t^T \Lambda_f^{-1} \widetilde{f}_t = -\alpha \|\Delta_t\| + \beta \left\| \widetilde{f}_t \right\|, \end{aligned}$$

donde $\alpha := [\lambda_{\min}(Q_0) - \lambda_{\max}(P\Lambda_f P)] \|\Delta_t\|$, $\beta := \lambda_{\max}\left(\Lambda_f^{-1}\right) \left\| \widetilde{f}_t \right\|$. Podemos seleccionar una matriz definida positiva Λ_f tal que

$$\lambda_{\max}(P\Lambda_f P) \leq \lambda_{\min}(Q_0). \tag{4.14}$$

Asi α y β son funciones $\mathcal{K}_\infty$, S_t es una función de Lyapunov de Entrada-de-Estado Estable (EEE) [31]. Utilizando el Teorema 1 de [31], la dinámica del error de identificación (4.6) es de entrada de estado estable, entonces tenemos que la entrada es el error de modelado, el estado es el error de identificación, por lo que sabemos que Δ_t está acotada [31]. Ahora solo falta definir cual es la cota.

Si definimos $R := \overline{W}_1 + \overline{W}_2 + \Lambda_f$, $Q := D_\sigma + \overline{u}D_\phi + Q_0$. (4.12) también tiene solución

positiva, (4.13) se vuelve

$$\dot{S}_t \leq -\Delta_t^T Q_0 \Delta_t + + \widetilde{f}_t^T \Lambda_f^{-1} \widetilde{f}_t, \tag{4.15}$$

integrando (4.15) de 0 a T resulta

$$S_T - S_0 \leq -\int_0^T \left(\Delta_t^T Q_0 \Delta_t\right) dt + \overline{\eta} T,$$

entonces

$$\int_0^T \Delta_t^T Q_0 \Delta_t dt \leq S_0 - S_T + \overline{\eta} T \leq V_0 + \overline{\eta} T,$$

(4.8) se establece. ■

4.3 Modelado del sistema híbrido

La estructura de un sistema híbrido se muestra en la Figura 4-2, donde se observa la separación del sistema en dos componentes; el supervisor y el controlador lógico son la parte evento discreto, mientras que el proceso es de naturaleza continua. El modelo híbrido consta de dos bloques: La red de Petri se encarga de supervisar y controlar las acciones, esto es, representa la toma de decisiones ante la actividad del proceso continuo, como por ejemplo, un sensor mide el nivel del líquido en un tanque (supervisión) y manda la señal para que cierre la válvula de llenado (acción de control), la red neuronal representa el proceso continuo de llenado indicando a la red de Petri cuando ejecutar estas acciones.

Definición 4.1 *La estructura de la red híbrida se define como una 9-tupla*

$$H = \left(P, T, I, O, RN_\sigma, f', \alpha', \beta'\right),$$

donde I, O, tienen la misma definición que una red de Petri, $P = P_P \cup P_N$, P_P es un subconjunto de lugares, P_N es un subconjunto de lugares asociados con las redes neuronales, $P_P \cap P_N = \emptyset$, $T = T_P \cup T_N, T_P \cap T_N = \emptyset$, T_P es un subconjunto de transiciones, $T_N = \{t_i | t_i \in$

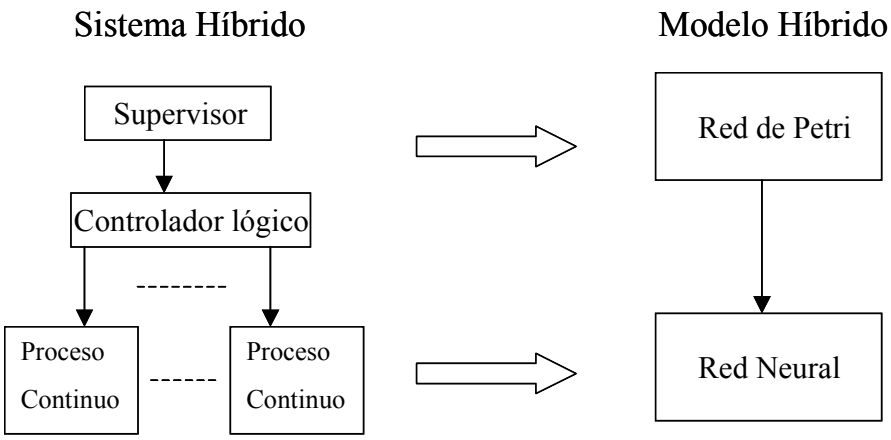

Figura 4-2: Estructura para el modelado híbrido.

$\{{}^{\cdot}t \cup t^{\cdot}\} \times P_N\}$ *es un subconjunto de transiciones de entrada y salida de lugares asociados con redes nuronales,* RN_σ *es la red neuronal,* $\sigma \in M = \{1, 2, ..., m\}$ *es el conjunto de procesos continuos, las redes neuronales dinámicas son*

$$\dot{\widehat{\mathbf{x}}}_t = A_\sigma \widehat{\mathbf{x}}_t + W^\sigma_{1,t} \sigma_\sigma(\widehat{\mathbf{x}}_t) + W^\sigma_{2,t} \phi_\sigma(\widehat{\mathbf{x}}_t) \mathbf{u}_t, \tag{4.16}$$

pueden ser reescritas como el siguiente sistema dinámico

$$\dot{\mathbf{x}} = f_\sigma(\mathbf{x}, \mathbf{u}), \quad \mathbf{x}(0) = \mathbf{x}_0, \tag{4.17}$$

$f' : T_N \rightarrow \mathrm{Re}$ *es una función de asociación, un mapeo de transiciones a valores reales,* $\alpha' : P_P \rightarrow \mathrm{Re}$ *es una función de asociación, un mapeo de lugares a valores reales y* $\beta' : P_N \rightarrow RN_\sigma$ *es una función de asociación continua, un mapeo biyectivo de lugares a la red neuronal.*

La representación gráfica se muestra en la Figura 4-3.La ejecución de H se realiza de manera semejante a una RPD, pero difiere en que la red neuronal (representa la parte continua) al estar asociada con los lugares en H, esta se encuentra sujeta al tiempo de evolución del proceso, por lo que, si se tiene una transición de salida que dependa del lugar en que se encuentre la red neuronal la ejecución se detiene hasta que termine el proceso

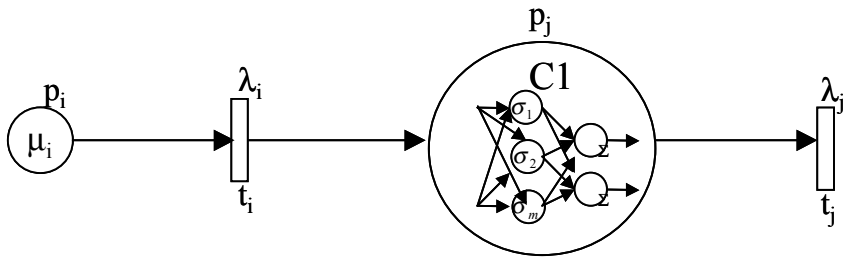

Figura 4-3: Gráfica de la red híbrida.

continuo (esto significa que en el lugar su *token* entregue un valor necesario para habilitar a la transición).

Sea $p_i \in I(t_i)$, $p_i \in P_P$, $p_j \in O(t_i), p_j \in P_N$ y $t_i \in T_N$, el valor del *token* en el lugar p_i esta denotado por $\alpha'(\ p_i) = \mu_i$, $\mu_i \in$ Re, la transición t_i tiene un valor asociado $f'(t_i) = \lambda_i, \lambda_i \in$ Re, si $\alpha'(\ p_i) \sim f'(t_i)$ (donde $\sim$ es un operador de relación, $<, \leq, =, \geq, >$) entonces t_i está habilitada y dispara, ahora el valor del *token* en p_j es $\beta'(\ p_j) = \alpha'(\ p_i)$ la red neuronal RN_σ comienza a trabajar y $\beta'(p_j) = x(0)$.

Sea $\beta'(p_j) = x(t)$ al ir adiquiriendo los valores entregados por la red el lugar p_j podrá habilitar a una transición t_j al cumplirse $\beta'(\ p_j) \sim f'(t_j)$ entonces tendrá lugar el disparo. Esto resume el comportamiento de un lugar asociado con la red neuronal tanto para la entrada como a la salida de su respectiva transición.

4.4 Simulación

Acontinuación se dan dos ejemplos que ilustran como se lleva a cabo el diseño del modelo híbrido. En el priemr ejemplo de manera particular se utiliza una red de Petri común y se dan los resultados de una simulación llevada acabo. Mientras que en el segundo ejemplo se diseña el modelo utilizando una red de Petri difusa, en este caso solo se plantea el modelo.

Ejemplo 4.1 *Considerar el sistema de dos tanques representado en la Figura 4-4. Este es un sistema híbrido el cual puede ser modelado por un autómata*[1] *[24]. Pero el autómata no*

[1] *Un autómata se define como:* $\mathcal{A} = \{\Phi, \alpha, \beta, F(\cdot,\cdot), H(\cdot,\cdot)\}$, *donde* Φ *es un conjunto de estados internos,*

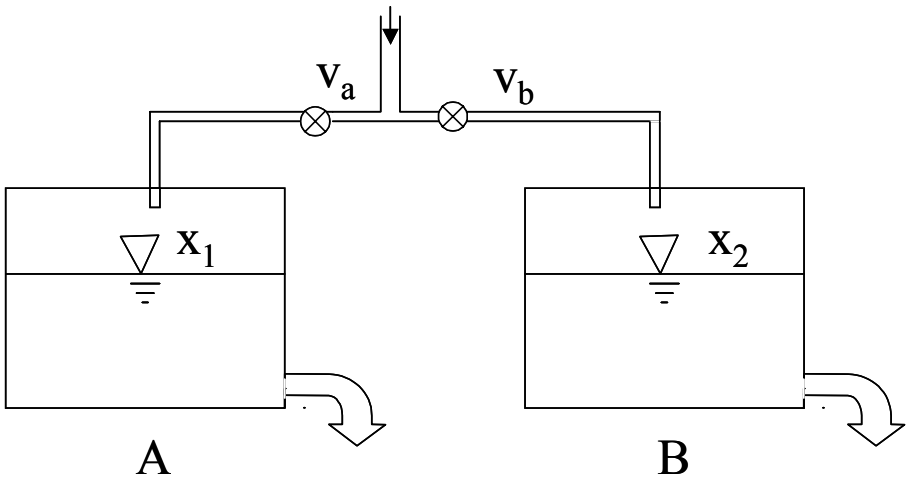

Figura 4-4: Sistema de dos tanques.

puede modelar la parte lógica. En el método propuesto en este capítulo, podemos modelar ambos, el interruptor lógico y los dos tanques de agua. Dividiendo al sistema es dos partes, en la discreta,tenemos dos válvulas controladoras V_a y V_b, los niveles en los tanques son X_1 y X_2. Los interruptores lógicos son: si el nivel de agua en el tanque A es $X_1 > 12$, entonces vaciamos el agua dentro de el tanque B. Si el nivel de agua en el tanque B es $X_2 > 10$, entonces vaciamos el agua dentro del tanque.

La parte continua puede ser representada por un par de ecuaciones lineales

$$\dot{X_1} = -0.2X_1 + 0.026V_a,$$
$$\dot{X_2} = -0.3X_2 + 0.03V_b,$$

donde $V_a = 0.89u$, $V_b = 0.7u$.

En cuya representacion de red de Petri el token del lugar p_1 indica abrir V_a, esto habilita la transición t_1 hasta que $X_1 > 12$ (disparando), entonces el token disparado va para p_2 (abriendo V_b) habilitando la transición t_2 hasta que $X_2 > 10$. El interruptor lógico puede ser representado por una red de Petri como se ve en la Figura 4-5. En este ejemplo no

β es un conjunto de acciones de entrada, α es un conjunto de salida, $F(\cdot,\cdot) : \Phi \times \beta \rightarrow \Phi$ es una función que mapea el estado actual y la entrada actual en el siguiente estado y $H(\cdot,\cdot) : \Phi \times \beta \rightarrow \alpha$ es una función que mapea el estado actualy la entrada a la salida actual [38].

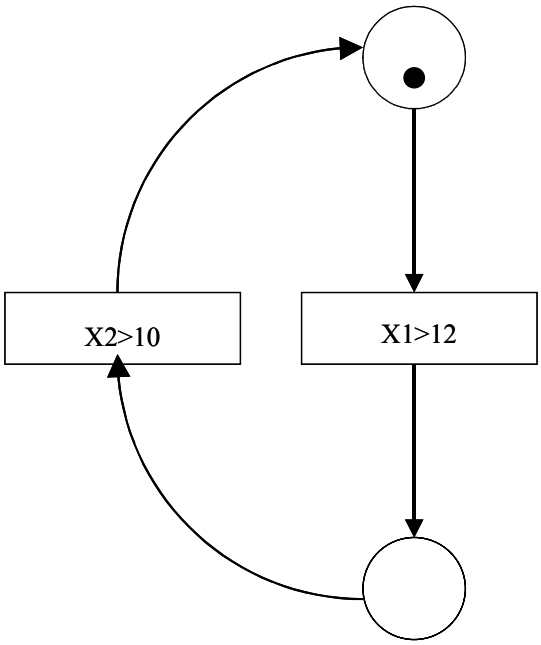

Figura 4-5: Modelo lógico del sistema de dos tanques.

empleamos una estructura RPD, pero si utilizamos dos lugares con RN_σ.

Nuetra estructura híbridad es

$$H = (P, T, I, O, RN_\sigma, f', \beta'),$$

donde $f'(t_1) = 12, f'(t_2) = 10, \beta'(p_1) = X_1$, $\beta'(p_2) = X_2$, las relaciones quedan establecidas por $\beta'(p_1) > f'(t_1), \beta'(p_2) > f2'(t_2)$. Ahora podemos usar las dos redes neuronales dinámicas (4.16) para modelar los dos tanques $M = \{1, 2\}$.

$$\begin{aligned} \text{Tanque } A: \quad & \dot{\widehat{X}}_{1,t} = A_1 \widehat{X}_{1,t} + W_{1,t}\sigma(\widehat{X}_{1,t}) + W_{2,t}\phi(\widehat{X}_{1,t})V_a \\ \text{Tanque } B: \quad & \dot{\widehat{X}}_{2,t} = A_2 \widehat{X}_{2,t} + V_{1,t}\sigma(\widehat{X}_{2,t}) + V_{2,t}\phi(\widehat{X}_{2,t})V_b. \end{aligned}$$

Su gráfica se presenta en la Figura 4-6. Este ejemplo ilustra la idea de dividir al sistema híbrido en dos partes, para la construcción de su modelo, en el siguiente ejemplo se muetra un sistema, en el que el modelo ya incluye una estructura RPD.

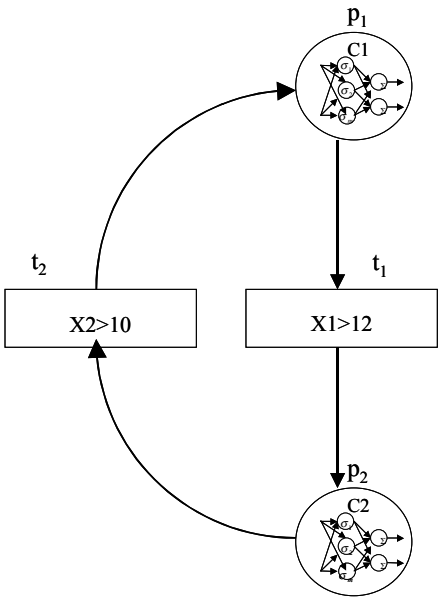

Figura 4-6: Modelo híbrido de los dos tanques.

De este ejemplo, se realizó una simulación de la combinación de la RPD y la RN, utilizando los datos del sistema

$$\dot{X_1} = -0.2X_1 + 0.026V_a$$
$$\dot{X_2} = -0.3X_2 + 0.03V_b,$$

estos datos nos garantizan que los tanques no tiene problemas, es decir, el gasto de salidad del líquido no es mayor al de entrada (lo cual repercutiría en que nunca se alcance el nivel de llenado requerido).

Se procedió a simular mediante la utilización de *Matlab Simulink* (para la parte continua del sistema). Mientras que el comportamiento lógico del sistema se utilizó un simulador de redes de Petri en *Visual Basic*. La red de Petri se emplea como un interruptor temporal (en *Simulink* se empleó una función para este propósito) para las redes neuronales, ya que la parte continua depende del tiempo en que se alcanza el nivel requerido (en *Visual Basic* sólo se programó el tiempo para el disparo de la transición obtenido de la red neuronal).

Laas gráficas del comportamiento de los dos tanques se muestran en las Figuras 4-7 y 4-8, se puede observar como aumentan y descienden los niveles X_1 y X_2, con respecto al

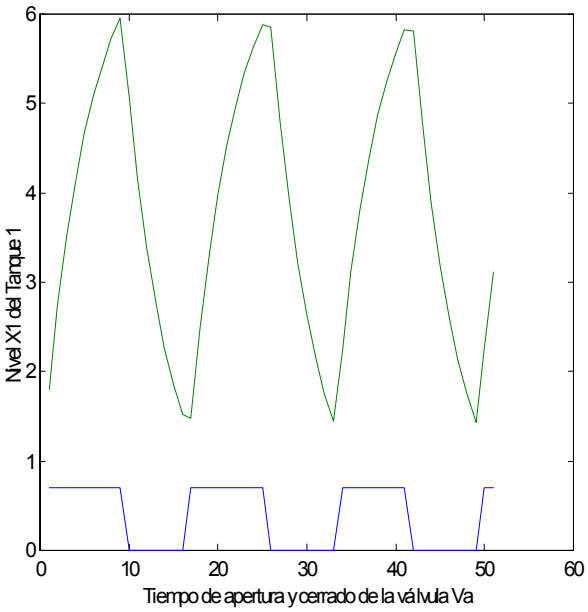

Figura 4-7: Comportamiento del Tanque 1.

cierre y abertura de las válvulas.

Los resultados de la simulaciones del comportamiento con las redes neuronales de los niveles X_1 y X_2, se dan en las Figuras 4-9 y 4-10 respectivamente. Las gráficas se comparan con el comportamiento del sistema real, para mostrar el desempeño de la red y obtenemos el tiempo que se tarda en realizar el disparo de las transiciones del modelo lógico de los tanque, esto es, el tiempo que transcurre para la apertura y cerrado del las válvulas. La simulación en *Visual Basic* puede verse en la Figura 4-11.

En el caso particular de las redes de Petri difusas la estructura híbrida propuesta puede representar cada proposición en la RPD, donde la consecuencia esta correspondida por una red neuronal (sobre los distintos tipos de proposiciones, ver capítulo *3* sección *3.2*). Incluso también se puede representar las reglas de producción difusas pesadas (sección *3.3*) como ejemplo la regla:

Tipo 2: Una Regla Conjuntiva Compuesta

R: IF a_1 AND a_2 AND $\cdots$ AND a_n THEN process c_2, $Th(t) = \lambda$, $W_O(t, p_j) = \mu$, $W_I(p_i, t) = w_i$, $i = 1, \cdots, n$.

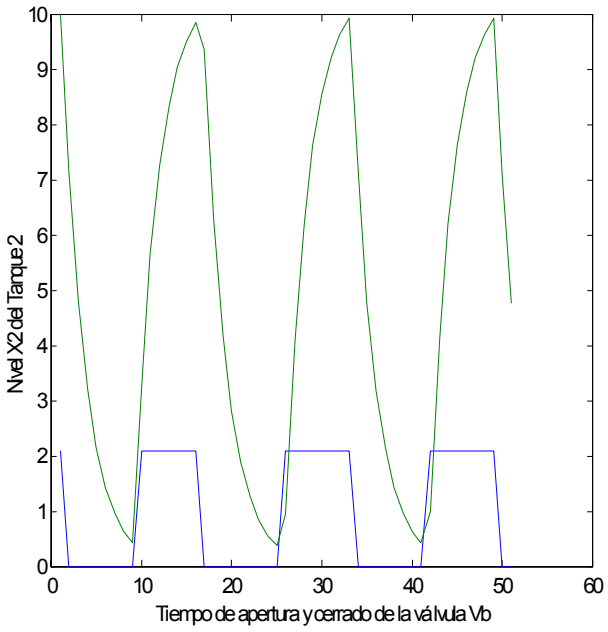

Figura 4-8: Comportamiento del Tanque 2.

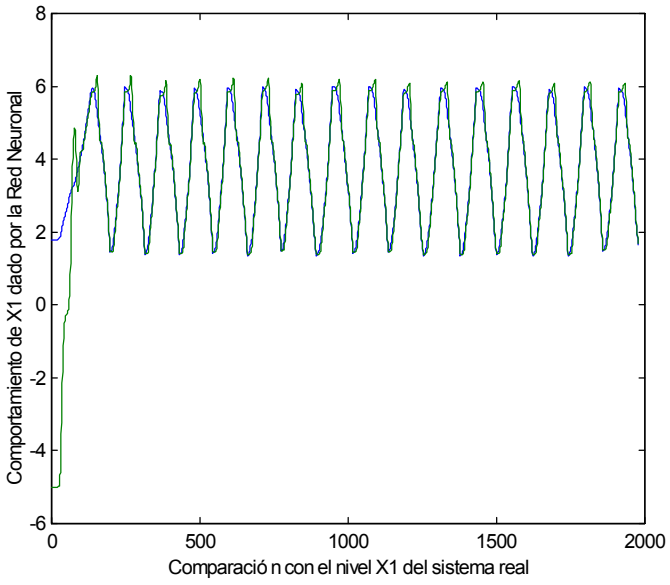

Figura 4-9: Comportamiento de la red neuronal del Tanque 1.

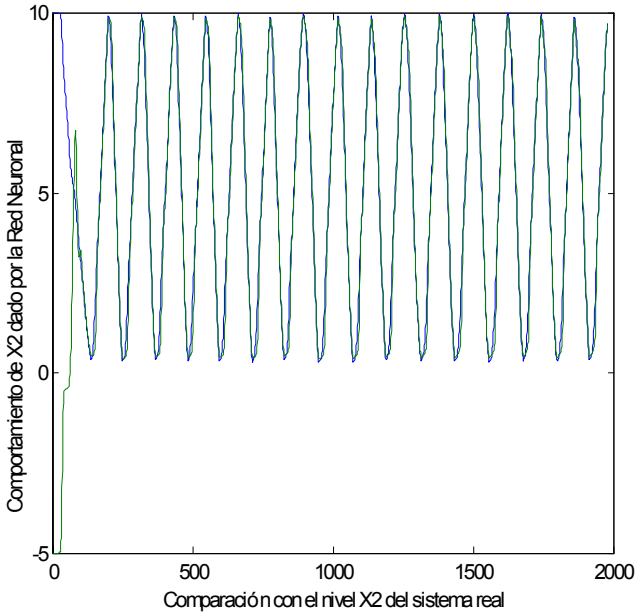

Figura 4-10: Comportamiento de la red neuronal del Tanque 2.

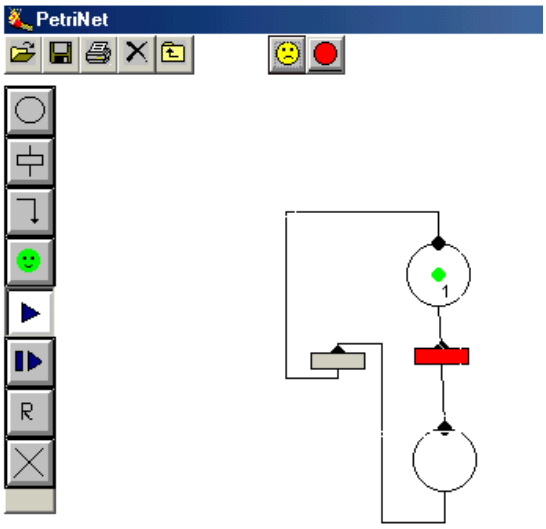

Figura 4-11: Simulación del comportamiento lógico de los dos tanques.

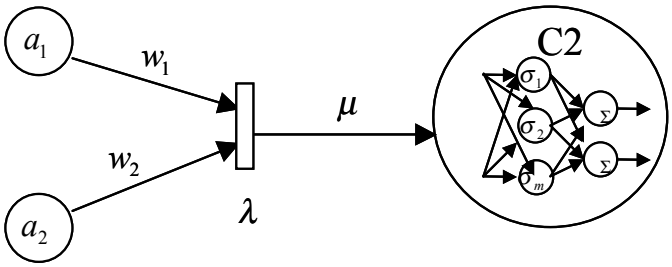

Figura 4-12: Modelo híbrido de la regla del tipo 2.

Estaá dada como se muestra en la gráfica de la Figura 4-12.El segundo ejemplo muestra, como se podria aprovechar estas reglas para modelar un sistema, realizado de manera esquemática.

Ejemplo 4.2 *Una típica celda de llenado se muestra esquemáticamente en la Figura 4-13. La celda consiste de dos reactores y un tanque de mezclado. El reactor* $P4$ *es controlado por el comando* $P1$

$$IF\ P1\ THEN\ P4(\lambda_1,\mu_1),$$

es decir, si se verifica $P1$*, se llena* $P4$*. El reactor* $P5$ *es controlado por el comando* $P2$ *y también el reactor* $P4$ *se detiene*

$$IF\ P2\ AND\ P4\ THEN\ P5(w_2, w_4, \lambda_3, \mu_3),$$

al llenarse $P4$ *y cumplirse* $P2$*, se llena* $P5$*. El tanque de mezclado* $P6$ *es inicializado si el reactor* $P5$ *se detiene, o se tiene otra entrada de otra celda de llenado* $P3$

$$IF\ P3\ OR\ P5\ THEN\ P6(\lambda_2, \mu_2, \lambda_4, \mu_4)$$

Primero, mapeamos el sistema lógico en una RPD como sigue

$$RPD = \{P, T, D, I, O, \alpha, \beta, Th, W\}$$

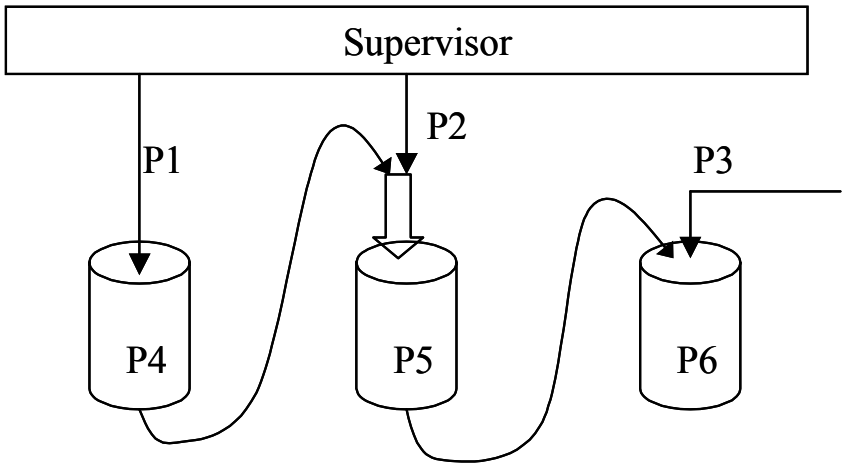

Figura 4-13: Esquema de una celda del proceso de llenado.

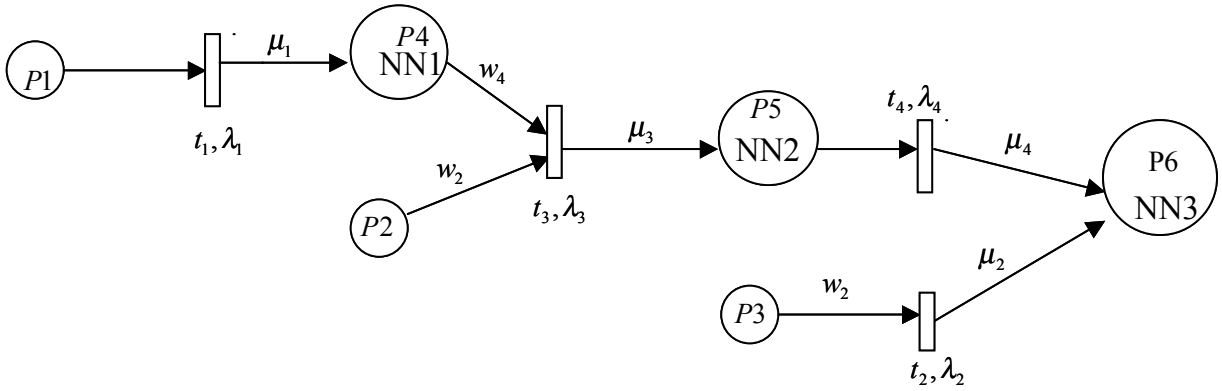

Figura 4-14: Modelo híbrido de una celda de llenado.

donde $P = \{p_1,\ p_2,\ p_3,\ p_4,\ p_5,\ p_6\}$, $T = \{t_1,\ t_2,\ t_3,\ t_4\}$, $D = \{P1, P2, P3, P4, P5, P6\}$, $Th = \{\lambda_1,\ \lambda_2,\ \lambda_3,\ \lambda_4\}$, $W_I = \{w_2,\ w_4\}$, $W_O = \{\mu_1,\ \mu_2,\ \mu_3,\ \mu_4\}$.

Y los dos reactores junto con el tanque de mezclado son modelados por tres redes neuronales dinámicas RN1-RN3.

Entonces nuestra estructura esta dada por

$$H = (P, T, D, I, O, RN_\sigma, f', \alpha', \beta').$$

Su gráfica se muestra en la Figura 4-14. Podemos ver que la combinación de la red de Petri y las redes neuronales pueden modelar sistemas híbridos de base lógica.

4.5 Conclusión

En resumen, este capítulo presenta un nuevo método para el modelado de sistemas híbridos con redes de Petri y rede neuronales. Así nace la estuctura H la cual se divide en dos partes; una red de Petri (la cual puede ser también una red difusa) y redes neuronales. Esto en base a la visión de considerar a los sistemas híbridos formados de dos partes: el sistema de conocimiento y el sistema continuo. De hecho la construcción del modelo híbrido parte de estos preceptos:

1. Separamos el comportamiento lógico del sistema para representarlo con una red de Petri.

2. El modelo lógico establece en que lugares de la red se encuentran las partes continuas, para representarlas con las redes neuronales.

3. Por último se establecen los mecanismos de ejecución de la red, es decir, la mecánica de disparo de las transiciones y la asignación de los valores para los *tokens.*

En el capítulo pasado se dotaban a las redes de Petri difusas de la habilidad de aprender, utilizando una estructura semejante a una red neuronal, de esta idea nació la propuesta de utilizar redes neuronales, esto garantiza el respaldo que se tiene de toda la teoría neuronal desarrollada en años pasados y no tener que desarrollar una nueva para las redes de Petri difusas adaptables (aunque de todas formas el análisis de convergencia se hace a través de la conversión de la RPDA a una red neuronal).

Por último hay que resaltar que en la definición de H en el caso de utilizar una estructura difusa se debe profundizar en como se da la relación de las proposiciones D en los lugares P_N y como se haría el análisis para esta red, en otras palabras, quedan abiertas estas cuestiones para futuros trabajos. En el siguiente capítulos se hace otra propuesta de red híbrida mas enfocada a la representación de reglas difusas utilizando redes de Petri continuas.

Capítulo 5

Modelado de sistemas híbridos mediante Redes de Petri Continuas

5.1 Introducción

En al literatura se ha encontrado una estructura que puede representar a un sistema continuo mediante una red de Petri, esta estructura se define como red de Petri continua (RPC), la red de Petri continua es una extensión de una red de Petri donde los *tokens* son números reales, por lo que se tiene un marcado continuo y un disparo de transiciones también continuo. Al tener estas características es necesario establecer definiciones para las reglas de ejecucion de una red de este tipo. En particular [13] se presenta un modelo para sistemas híbridos utilizando la RPC, sin embargo los modelos construidos utilizan redes de Petri normales. Como los temas tratados en la tesis, han sido referentes a redes de Petri difusas, surge el planteamiento de una nueva estructura que utilice estas redes junto con las RPC. En este capítulo se desarrolla un modelo con estas características. Con este modelo se plantea una relación entre proposiciones lingüísticas con lugares de la red de Petri continua, lo cual abre camino a una respuesta del planteamiento realizado al final del capitulo pasado.

5.2 Red de Petri Continua

Antes de dar la definición formal de una red de Petri continua se debe tener en claro la siguiente idea: En una red de Petri común, el disparo de la transición se hace en un instante de duración cero, pero en la RPC se utiliza la noción de flujo o gasto (q), el cual dictamina la forma en que se hace la transmisión del *token,* es decir, se tiene un disparo continuo de valores.

Definición 5.1 *La estructura de una red de Petri continua está dada por una 5-tupla*

$$RPC = (P_c, T_c, q, W_c, x_0),$$

donde P_c y T_c son respectivamente, un conjunto finito de lugares continuos y un conjunto finito de transiciones continuas, $q : T_c \rightarrow \{X \rightarrow \mathrm{Re}^+\}$ es una función de flujo, donde $X \subseteq (\mathrm{Re}^+)^{n_c}$ es un conjunto de marcaciones continuas, n_c es el número de lugares continuos, $W_c : A \rightarrow \{X \rightarrow \mathrm{Re}^+\}$ es un conjunto de pesos que se asignan a los arcos de la red, y $x_0 \in X$ es el marcado continuo inicial.

Gráficamente, los lugares y transiciones continuas son representados por círculos dobles y barras rectangulares, respectivamente. El flujo se señala junto a su correspondiente transición y los pesos continuos se escriben junto a sus correspondientes arcos (se omiten si su valor es de 1). Una red de Petri continua se muestra en la Figura 5-1 (a).

Es necesario añadir otra característica a la RPC para poder construir un modelo híbrido, para lo cual se definen las transiciones de salida discretas adicionales, estas transiciones son disparadas cuando un estado continuo alcanza un umbral dado.

Definición 5.2 *Una red de Petri continua con transiciones de salida discretas está descrita como*

$$RPC_{out} = (RPC, T_{out}, F_{out}, Op_{out}),$$

donde, T_{out} es un conjunto de transiciones de salida discreta, $T_{out} \cap T_c = \emptyset$, $F_{out} \subseteq P_c \times T_{out}$ es un conjunto de arcos direccionados conectando lugares continuos con las transiciones de

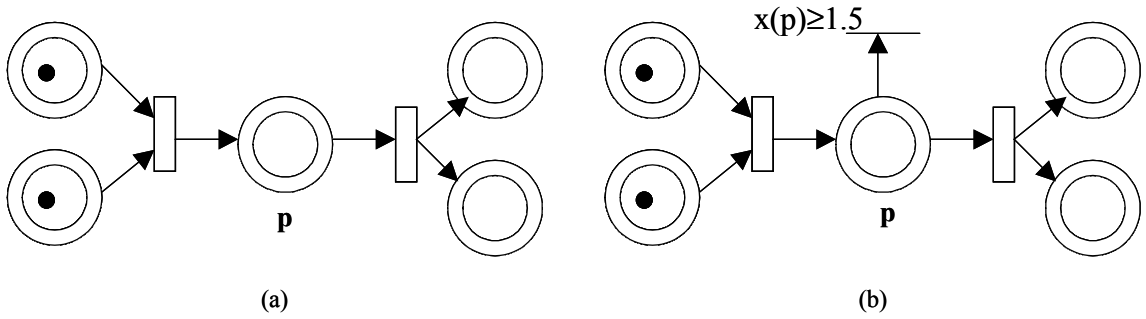

Figura 5-1: (a) Red de Petri Continua. (b) Red de Petri continua con transición de salida discreta.

salida, Op_{out} *es un conjunto de operadores de relación asociados con las transiciones de salida* $Op_{out} := \{<, \leq, =, \geq, >\}$. *Para cada* $t \in T_{out}$, $Op_{out}(t)$ *es un operador de relación definido en la marcación de los lugares de entrada de* t *(esta marcación parcial está denotada por* $x|^{\cdot}t$*).*

Un lugar $p \in P_c$ es llamado un lugar de entrada de t si $(p,t) \in F_{out}$. Una transición de salida discreta t dispara en RPC_{out} tan pronto como $Op_{out}(t)(x|^{\cdot}t)$ se vuelve verdadero. El disparo de la transición de salida discreta no cambia el estado (la marcación) de la red continua. Una transición de salida discreta dispara sólamente al instante que el operador de relación se vuelve verdadero. Por ejemplo, para la red de Petri continua con una transición de salida discreta mostrado en la Figura 5-1 (b), si inicialmente $x(p) = 1.0$ y $x(p)$ es creciente, entonces la transición t disparará tan pronto como $x(p)$ alcance 1.5. De otra forma, si inicialmente $x(p) = 2.0$, $x(p)$ no disparará incluso si con el operador de relación $x(p) \geq 1.5$ es verdadero.

5.3 Modelado de sistemas híbridos via Redes de Petri Híbridas.

Una red de Petri difusa y una red de Petri continua (refiriendonos por simplicidad, a la red de Petri continua con transición de salida discreta como red de Petri continua), pueden estar

interconectadas a través de condiciones y eventos para formar una nueva red, llamada red de Petri híbrida (RPH).

Antes de definir la red de Petri híbrida, se debe hacer la observación, en una RPC_{out} se utiliza Op_{out} relacionados con las tranciciones discretas de la red de Petri. En nuestro caso al tratarse de una red difusa utilizaremos un subconjunto de las proposiciones D,esto es, los eventos E.

Definición 5.3 *Si todas las condiciónes y eventos de una red de Petri difusa y una red de Petri continua están acopladas, la resultante RPH puede ser descrita:*

$$RPH = (RPD, RPC_{out}, D, C, \gamma, \epsilon),$$

donde $RPD = (P, T, D_P, I, O, f, \alpha, \beta)$ *(ver capítulo* 3*),* $RPC_{out} = (RPC, T_{out}, F_{out}, E)$*,* $D = D_P \cup E$*,* $D_P \cap D_{T_{out}} = \emptyset$*,* D_P *es un cojnunto de proposiciones relacionadas con los lugares de la RPD,* E *es un conjunto de eventos, los los cuales son generados por las transiciones de salida discreta, esto es, un evento se genera cuando se cumple con la relación de operación de la transición de salida discreta* T_{out}*,* $E \subseteq (T_{out} \times T)$*,* C *es un conjunto de condiciones, las cuales son usadas para controlar las transiciones continuas de la red, regulan el flujo* q *de las tranciciones a los lugares continuos,* $C \subseteq (P \times T_c)$*,* $\gamma : (P \times D) \rightarrow q$ *es una función de asociación, un mapeo de lugares asociados con proposiciones a valores de flujo,* $\epsilon : P_c \rightarrow (T_{out} \times T)$ *es una función de relación, un mapeo del valor del token continuo para las los eventos de las transiciones de salida discreta.*

Su representación gráfica se muestra en la Figura5-2. Las conecciones de los eventos y las condiciones son representadas por un arco direccionado del lugar a la transición t_i con un punto sólido al final, y un arco direccionado del lugar p_j a la transición continua con un punto sólido al final, respectivamente.

La ejecución de RPH se reliza en principio como una RPD, pero difiere de esta en que la red continua está asociada con los lugares discretos para el disparo de su transición continua. El control de la transición de la red difusa $t_i \in T_{out}$, se encuentra sujeta al tiempo en que el valor del *token continuo* llegue al valor necesitado para que la proposición (la cual tiene asignada un valor difuso) sea verdadera y dispare la transición.

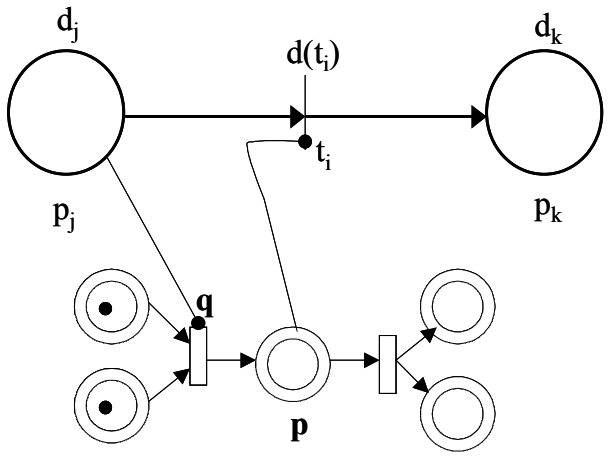

Figura 5-2: Gráfica de una red de Petri híbrida.

Recordatorio de la red de Petri difusa:

Sea $p_j \in I(t_i)$ y $p_k \in O(t_i)$, el valor del *token* en el lugar p_i esta denotado por $\alpha(\, p_j) = y_j$, $y_j \in [0,1]$, la transición t_i tiene un valor asociado $f(t_i) = \mu_i, \mu_i \in [0,1]$, si $\beta(p_j) = d_j$, $d_j \in D$, entonces el lugar p_j se dice estar asociado con la proposición d_j.

Para le red de Petri híbrida:

El valor del gasto está denotado por $\gamma(p_j, d_j) = q(t_{c_i})$, es decir, el valor de la función de gasto depende del valor del *token* en p_j, el cual está asociado con la proposición d_j. De esta forma se *condiciona* al gasto en la transición. El gasto regula el flujo del marcado continuo X_i en el lugar continuo p_{c_i} al llegar a un determinado valor, este se compara dentro de la función de relación, $\epsilon(p_{c_i}) = \{X_i \sim \mu_i\}$, si este valor es el requerido por el evento $d(t_i)$, al ser verdadera (o falsa según sea el caso) se habilita a la transición t_i para disparar.

Se deben señalar algunos puntos:

- El valor del gasto es dado por la condición relacionada a las proposiciones de los lugares. Por ejemplo; en un sistema, *un tanque está medio lleno,* por lo tanto hay que llenarlo a una razón de 6 lt/seg.

- Los eventos tienen asignados valores difusos para su operación (al igual que los valores asignados para los lugares).

- Estamos utilizando en la RPH a los valores difusos $\mu_i \in [0,1]$ como umbrales para el disparo de las transiciones, por lo que el valor del marcadoX_i se normaliza a x_i para utilizarlo en la proposición ya que la debe hacerse verdadera o falsa según sea el caso. Por ejemplo; *un tanque no debe llenarse a tope pero si que rebase la mitad,* los valores podría ser *0.5, 1.0,* si el tanque tiene capacidad de *15*lts hay que normalizarlo para utilizarlo, en este caso nuestro evento estaría dado en forma de pregunta lingüística *¿está el tanque lleno*?

5.4 Ejemplos de redes de Petri híbridas

En esta sección presentamos dos ejemplos de modelos obtenidos con base en esta estructura híbrida. En el primer ejemplo se obtiene el modelo de una regla sencilla, mientras que en el segundo ejemplo, se presenta el modelo de un sistema completo.

Ejemplo 5.1 *De los ejemplos mencionados en los puntos anteriores, considerese la regla de producción difusa*

$$R_i: \; IF \text{ está medio lleno } THEN \text{ llénalo, pero no a tope.}$$

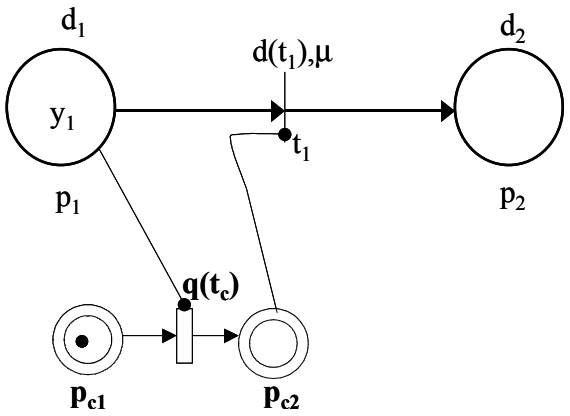

Figura 5-3: Red de Petri híbrida de la regla de producción difusa.

Puede ser modelada con la siguiente estructura

$$
\begin{aligned}
RPH &= \{RPD, RPC_{out}, D, C, E, \gamma, \epsilon\},\\
P &= \{p_1, p_2\},\\
P_c &= \{p_{c1}, p_{c2}\},\\
T &= T_{out} = \{t_1\},\\
T_c &= \{t_c\},\\
D &= \{\text{está medio lleno, ¿está lleno?, está lleno pero no a tope}\},\\
E &= \{\text{¿está lleno?}{=}x \sim \mu\},\\
C &= \text{llenarse rápido},\\
I(t_1) &= \{p_1\}, O(t_1) = \{p_2\}, f(t_1) = \mu,\\
\alpha(p_1) &= y_1, \alpha(p_2) = 0,\\
\gamma(p_1, d_1) &= q(t_c), \epsilon(p_{c2}) = X \sim \mu,\\
\beta(p_1) &= \text{está medio lleno},\\
\beta(p_2) &= \text{lleno, pero no a tope},\\
X &= \text{la cantidad de líquido en el tanque, } x_0 = \text{la cantidad inicial.}
\end{aligned}
$$

Su gráfica se presenta en la Figura 5-3 El modelo trabaja de la siguiente manera:

Tenemos la regla R_i, la cual está compuesta de tres proposiciones, en la primera proposicón, *está medio lleno.* Se le asigna una valor determinado a la proposición dado por $\alpha(p_1) = y_1$, con el cual vamos a condicionar a la transición continua mediante la asignación de un valor para el flujo $\gamma(p_1, d_1) = q(t_c)$, por ejemplo, que se llene de manera *rápida, normal, lenta, etc.* La transición discreta se habilitará, tan pronto en el lugar p_{c2} el valor del *token* continuo, llegue al valor necesario para que el evento $\epsilon(p_{c2}) = X \sim \mu$, sea verdadero, dando paso al disparo de la transición.

Como puede observarse el modelo representa la regla sin problemas, sin embargo este es un caso particular, por lo que se debe tener en cuenta los tipos de reglas generales que existen (capítulo *3*).

En el siguiente ejemplo, tratamos con un sistema compuesto de dos reglas, esto con el fin de establecer una metodología para la construcción del modelo, la cual puede ser aplicada a sistemas más complejos.

Ejemplo 5.2 *Un aire acondicionado se controla en base a las reglas:*

1. Si hace calor entonces enfria.

2.Si hace frio entonces calienta.

Para construir el modelo hay identificar las proposiciones, las condiciones y los eventos. Una vez realizado esto, establecemos el número de lugares, transiciones, tokens, tanto para la red dicreta como para la red continua. Al final se asignan los valores respectivos de cada red, a los lugares, transiciones, tokens, etc. La gráfica.de la red se diseña una vez que se hayan realizado los anteriores pasos y se verifica que funcione de acuerdo al sistema modelado.

En nuestron ejemplo, la red de Petri híbrida obtenida es

$$
\begin{aligned}
RPH &= \{RPD, RPC_{out}, D, C, E, \gamma, \epsilon\}, \\
P &= \{p_1, p_2\}, \\
P_c &= \{p_{c1}, p_{c2}\}, \\
T &= T_{out} = \{t_1, t_2\}, \\
T_c &= \{t_{c1}, t_{c2}\}, \\
D &= \{\text{hace calor, ¿está frio?, hace frio, ¿está caliente?}\}, \\
E &= \{\text{¿está frio?}{=}x \sim \mu_1, \text{¿está caliente?}{=}x \sim \mu_2\}, \\
C &= \{\text{calentar, enfriar}\}, \\
I(t_1) &= \{p_1\}, O(t_1) = \{p_2\}, I(t_2) = \{p_2\}, O(t_2) = \{p_1\}, \\
, f(t_1) &= \mu_1, f(t_2) = \mu_2, \\
\alpha(p_1) &= y_1, \alpha(p_2) = 0, \\
\gamma(p_1, d_1) &= q(t_{c1}), \gamma(p_2, d_2) = q(t_{c2}), \\
\epsilon(p_{c1}) &= X \sim \mu_1, \epsilon(p2) = X \sim \mu_2, \\
\beta(p_1) &= \text{hace calor,} \\
\beta(p_2) &= \text{hace frio,} \\
X &= \text{la temperatura ambiente, } x_0 = \text{la temperatura inicial.}
\end{aligned}
$$

y su gráfica muestra en la Figura 5-4.El modelo trabaja de la siguiente forma:

Dado el estado inicial (marcado en p_1), *hace calor* se da la condición de que se baje la tempertaura, el evento ¿está frio? se activa cuando se alcanza la temperatura deseada. Al cumplirse con la condición llegamos al siguiente estado que es, *hace frio*, la siguiente transición no se activa ya que nos encontramos con un valor de tempertatura que no debe activar la condición. El caso contrario sería cuando estuviesemos inicialmente en *hace frio*, con lo cual tendríamos el proceso reverso (la marcación inicial estaría dada en el lugar p_2).

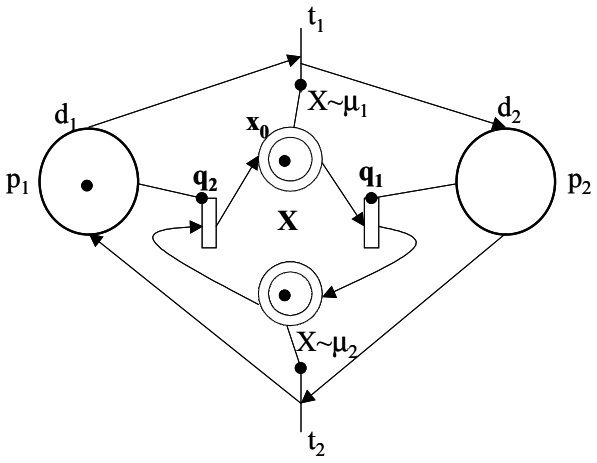

Figura 5-4: Gráfica de la red de Petri híbrida del ejemplo *5.2.*

5.5 Conclusión

Este capítulo presenta un nuevo método para el modelado de sistemas híbridos. Con base a la unión de una red de Petri difusa con una red de Petri continua (las propuestas de redes de Petri comunes con redes continuas ya tienen un importante respaldo en la literatura). Al contrario del modelo híbrido obtenido con una red neuronal, donde se tienen lugares asociados con las redes neuronales (representadas gráficamente dentro de los lugares), en esta estructura se visualiza la separación existente entre la RPD y la RPC, las cuales se unen por sus respectivas transiciones (ver definición *5.3*). La construcción del modelo puede llegar a ser compleja, dada la cantidad de procesos continuos que contenga el sistema a modelar. No se ha realizado una comparación formal del arreglo obtenido con redes neuronales, ya que para hacer una comparación eficaz del desempeño de ambos modelos es necesario tomar en cuenta el análisis de estas redes de Petri híbridas. Pero el modelo obtenido ha servido para establecer la relación necesaria entre las proposiciones y la parte continua (RPC) de la red híbrida. De manera semejante se podría establecer esta relación en las redes de Petri con redes neuronales. Lo cual queda para trabajos futuros, en los cuales se puede elaborar un programa que lleve a cabo una simulación formal de un sistema real, bajo el modelo propuesto de red de Petri híbrida RPH.

Capítulo 6

Control de Sistemas Híbridos mediante Redes de Petri Híbridas

6.1 Introducción

Los sistemas dinámicos de eventos discretos (DEDS) son sistemas dinámicos que son básicamente asíncronos y que evolucionan de acuerdo con la ocurrencia de los eventos. La teoría de control supervisado para los DEDS, fue introducida primero por Ramadge y Wonham [40]-[49], está basada en modelos de automatas y lenguaje formal. En estos modelos, el tiempo cronológico no está involucrado, esto es, solo se concentran en el ordenamiento de los eventos, suponiendo que genera eventos espontáneamente. Su comportamiento puede ser descrito por secuencias de eventos y formar un lenguage sobre el alfabeto de los mismos (teniendo en cuenta que el alfabeto es un conjunto de eventos de salida [40]). Dado un proceso, el objetivo de la teoría [40]-[49] es el diseñar un supervisor de manera que el *proceso es supervisado*, es decir, el proceso acoplado con el supervisor, se comporta de acuerdo a varias restricciones. Podemos sintetizar un supervisor tal que el comportamiento del proceso supervisado resultante cumpla con tres condiciones: 1) no contradiga algunas especificaciones del comportamiento; 2) es permisible con las especificaciones; y 3)no tiene bloqueos. La teoría puede ser extendida para cubrir supervisión modular, decentralizada y jerárquica.

En la teoría Ramadge-Wonham (RW), se analiza al proceso con la supoción de que

evoluciona espontáneamente. Sin embargo, un proceso requiere de la adición de un agente de control externo que forza algunos eventos para que ocurran. Dentro de estas *aproximaciones de evento forzado,* la supervisión no se ajusta a su función original y los conceptos de control y supervisión se mezclan. Esto genera varios límites para la síntetis del control. En particular, uno no puede usar una aproximación modular que es consistente con la prueba de controlabilidad. Esto nos obliga a construir un único agente de control supervisado que observe todos los eventos que vienen desde el proceso, resultando así en una pérdida de consistencia.

Este capítulo presenta el concepto de control supervisado de [8], [9]. Este nuevo concepto nos permite separar claramente el control y la supervisión. Dentro de esta aproximación, el controlador puede forzar a algunos eventos a ocurrir en el proceso. Para un observador externo, desde el punto de vista del supervisor parece que el proceso está acoplado con el controlador, denominándolo como *el proceso extendido*, el cual genera eventos espontáneamente. La supervisión está asi confinada a prohibir algunos de los eventos a ser generados en el proceso extendido. El marco jerárquico introducido por el control supervisado clarifica la entrada y la salida de el sistema, nos habilita para formalizar y sistematizar la transición desde la síntesis a la implementación de el control. Con el señalamiento, de que se hace la propuesta bajo el uso general de los modelos presentados en los capítulos anteriores.

6.2 Control supervisado.

Los sistemas dinámicos de evento discreto (DEDS) son sistemas que evolucionan de acuerdo con la ocurrencia de eventos. Para un análisis cualitativo, el tiempo en que ocurren los eventos no está involucrado. El comportamiento de una DEDS puede ser descrito como un conjunto de secuencias sobre el alfabeto de eventos y define un lenguaje formal [21].

6.2.1 Modelo lógico para DEDS

Una DEDS puede ser modelada por una *gráfica de transición de estados.* En la aproximación RW, se supone que uno observa desde afuera la *simultaneidad de algunos eventos con algunas transiciones.* No sabemos si los eventos son causas o consecuencias de las transiciones. Se

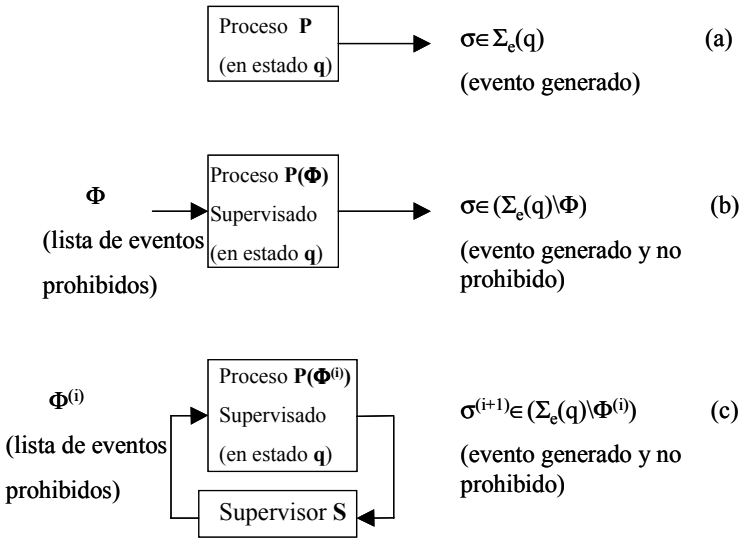

Figura 6-1: Control supervisado. (a) Proceso P en aislamiento. (b) Proceso P con comportamiento restringido. (c) Esquema del control supervisado.

dice que son *generados* por el proceso.

Definición 6.1 *La gráfica de transición de un proceso P se puede definir como:*

$$P = (Q, \Sigma, \delta, q_0),$$

donde: Q es el conjunto de estados q; Σ es el alfabeto de eventos; $\delta : Q \times \Sigma \rightarrow Q$ es una función parcial de transición; q_0 es el estado inicial (hay que señalar, que está definición viene dada de la teoría de autómatas, en el caso de la tesis, nos sirven sus fundamentos básicos para su uso con RP).

El comportamiento de P en aislamiento puede verse en la Figura 6-1 (a).Para cada actual estado q, P evoluciona cuando se genera un evento en $\Sigma_e(q)$, donde $\Sigma_e(q)$ es el conjunto de todos los *eventos elegibles* $\sigma \in \Sigma$ tal que $\delta(q, \sigma)$ está definida.

6.2.2 Concepto de control supervisado y controlabilidad

El proceso acoplado P con un supervisor y reciviendo, en la entrada, una *lista de eventos prohibidos* Φ será denotada $P(\Phi)$ y será referencia para un *proceso supervisado.* Si $P(\Phi)$

está en el estado q entonces se comporta como se muestra en Figura 6-1 (b): $P(\Phi)$ puede solamente evolucionar por la generación de un evento de $\Sigma_e(q)\backslash\Phi$. Esto significa que un evento σ es elegible en $P(\Phi)$ si y solo si σ es elegible en P, $\sigma \in \Sigma_e(q)$ y σ no está prohibido por el control supervisorio, $\sigma \notin \Phi$. Note que un proceso supervisado puede ser definido en una manera dual especificando, en la entrada, la lista de eventos autorizados.

Damos la siguiente partición: $\Sigma = \Sigma_c \cup \Sigma_u$, donde Σ_c y Σ_u son respectivamente los conjuntos de *eventos controlables* e *incontrolables*. Como los eventos incontrolables pueden no ser inhabilitados a través del control supervisado, se requiere siempre para Φ que $\Sigma_u \cap \Phi = \emptyset$. Formalmente, un supervisor S puede ser definido como una máquina de Moore (autómata [21]) mediante una sextupla $S = (R, \Sigma, \xi, r_0, 2^{\Sigma_c}, \Theta)$ donde R es el conjunto de estados; Σ es el conjunto finito de entradas; $\xi : R \times \Sigma \rightarrow R$ es la función de transición; r_0 es el estado inicial; 2^{Σ_c} es el conjunto finito de salidas; y $\Theta : R \rightarrow 2^{\Sigma_c}$ es la función de salida. El comportamiento de un proceso P acoplado con S se ilustra en la Figura 6-1 (c). En cualquier tiempo lógico i, S proveé a P con $\Phi^{(i)}$. El proceso supervisado $P(\Phi^{(i)})$ puede entonces generar el $(i+1)$-ésimo evento $\sigma^{(i+1)}$, el cual es un evento en $\Sigma_e(q)\backslash\Phi^{(i)}$. La ocurrencia de $\sigma^{(i+1)}$ puede conducir S dentro de un nuevo estado. Desde este estado, $\Phi^{(i+1)}$ [con la posibilidad $\Phi^{(i+1)} \neq \Phi^{(i)}$] es suministrado a P, y asi continua.

Dado un proceso P, y una especificación K, la existencia de un supervisor S tal que el comportamiento de P acoplado con S respecto de K, yace en el concepto de controlabilidad dado en [49].

En esta tesis la intención es utilizar en lugar de una máquina de Moore, los arreglos propuestos en los capítulos pasados bajo el esquema dado por la teoría RW.

6.3 Control supervisado mediante Redes de Petri

La Figura 6-2 presenta el *concepto de control supervisado.* Un proceso acoplado con el controlador lógico se muestra en la Figura 6-2 (a). El proceso a ser controlado es visto como un DEDS por el controlador lógico que es en si mismo un DEDS. Las entradas al controlador lógico son las salidas de el proceso a ser controlado y viceversa. Básicamente, las entradas y salidas de el controlador lógico, llamadas X y Z, son variables Booleanas. Por lo que,

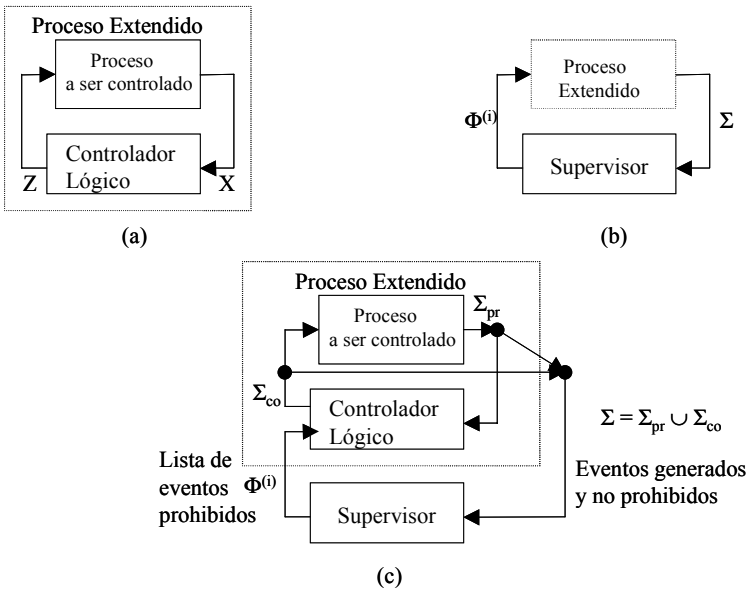

Figura 6-2: Control supervisado. (a) El proceso extendido. (b) El esquema del supervisor. (c) El esquema del control supervisado.

esto no es una restricción; uno puede siempre codificar el estado de una DEDS mediante variables Booleanas y además cualquier evento puede ser asociado con la cota superior o la cota inferior de una variable Booleana. Se sigue que uno puede tener eventos para X y Z. Para un observador externo, el *proceso extendido*, es el proceso acoplado con el controlador lógico, que puede ser visto como un dispositivo que evoluciona espontáneamente, es decir, sin forzamiento externo. Si uno quiere algunas especificaciones de comportamiento para ser respetadas por el proceso extendido, un supervisor puede ser acoplado con el proceso extendido como se muestra en la Figura 6-2 (b). Las entradas a el supervisor son las salidas de el proceso extendido. El supervisor basa su observación en el alfabeto Σ producido en el proceso extendido. En cualquier tiempo lógico i, el supervisor proveé al proceso extendido con una *lista de eventos prohibidos,* $\Phi^{(i)}$. Esto significa que el proceso extendido puede solamente generar eventos en $\Sigma \backslash \Phi^{(i)}$.

El *esquema de control supervisado,* se muestra en la Figura 6-2 (c). En este esquema jerárquico, los conceptos de control y supervisión están claramente separados. El proceso puede ser visto como un dispositivo que evoluciona mediante la producción de eventos en Σ_{pr}. El controlador lógico puede también ser visto como un DEDS que controla el proceso

mediante la *producción de eventos* en Σ_{co}. Para un observador externo, el *proceso extendido* puede ser visto como un dispositivo que evoluciona mediante la generación de eventos en $\Sigma = \Sigma_{pr} \cup \Sigma_{co}$.

Observación 1: En el esquema de la Figura 6-2 (c), la *supervisión adecua su función original,* es decir, la tarea supervisoria está confinada a *prohibir que ocurran algunos de los eventos.* Se tiene que la *tarea del control es independiente desde la tarea supervisoria.* Por lo tanto, si uno modifica las especificaciones de comportamiento que deben ser aplicadas a el proceso extendido, el sistema de control no necesita ser reconstruido.

Se considera que los eventos producidos por el proceso son eventos incontrolables. De hecho estos son información que proviene de los sensores. Esto implica que $\Sigma_{pr} \subseteq \Sigma_u$. También, se considera que si se necesita, uno puede siempre deshabilitar el controlador lógico para producir eventos en Σ_{co}, es decir, órdenes para los actuadores del proceso. Por lo que, para algunas especificaciones de comportamiento, todos los eventos en Σ_{co} no se necesitan que sean eventos controlables, entonces en general, Σ_c puede ser definido como un subconjunto de Σ_{co} (y $\Sigma_u = \Sigma \backslash \Sigma_c$). Note que en la Figura 6-2 (c), las salidas de el supervisor son algunas de las entradas de el controlador lógico, de este modo se habilita al supervisor para cancelar, a cualquier tiempo, algunos de los eventos controlables a ser producidos por el controlador lógico.

Observación 2: Desde un punto de vista teórico, cualquier restricción dada por el supervisor puede ser añadida al controlador es decir, el caso límite el cual es la unión de el controlador y el supervisor es un controlador, mientras la propiedad recíproca no es verdadera.

Uno considera que existe un control único y un supervisor único. Por lo tanto, el controlador y el supervisor pueden estar compuestos de varias partes.

El diseño de un sistema de control supervisado está basado en los siguientes dos pasos: 1) el diseño de el controlador (si todavia no existe) y 2) el diseño de el supervisor que restringe el comportamiento de el proceso extendido. Note que que el primer paso necesariamente precede al segundo: el diseño de el supervisor depende de el proceso extendido. Esto representa una mayor diferencia con respecto a [1].

6.3.1 Variables y eventos booleanos

Consideremos una variable Booleana A, suponiendo que conocemos el rango de tiempo [0, T). Entonces el tiempo cuando los eventos $\uparrow A$ (cota superior) y $\downarrow A$ (cota inferior) ocurren son conocidos. Recíprocamente, si todas las ocurrencias de $\uparrow A$ y $\downarrow A$ entre 0 y T se conocen y el valor de A es conocido en $t = 0$, entonces el valor de A es conocido para cualquier tiempo t en el rango [0, T). Se tiene que uno puede seleccionar para modelar las entradas (X en la Figura 6-2 (a)) y las salidas (Z en la Figura 6-2 (a)) de un DEDS ya sea variables Booleanas o eventos o ambos. Los modelos híbridos permiten el uso de variables Booleanas y eventos para entradas y salidas. También hay que señalar que cuando el tiempo de una variable Booleana A es constante tiene una duración finita, los eventos $\uparrow A$ y $\downarrow A$ no tienen duración.

Notación: Usamos una letra mayúscula para una variable Booleana, ejemplo, B y $B\prime$ para su complemento. Un evento será representado por una letra minúscula, por ejemplo $b = \uparrow B$ y $b' = \downarrow B = \uparrow B'$. Por simplicidad, un dispositivo (sensor o actuador) será representado por el mismo símbolo como su correspondiente variable Booleana.

A continuación presentamos un ejemplo de como se utilizan a los eventos y las variables Booleanas.

Ejemplo 6.1 *Llenado del Tanque: Un sistema de dos tanques está representado en la Figura 6-3. Ambos tanques son usados en de manera similar, por esta razón solo detallaremos el funcionamiento del primer tanque. El tanque 1 está vacío cuando el nivel es $B_1 \leq 0$, está lleno cuando el nivel $H_1 \geq 1$. En el estado inicial ambos tanques están vacíos. Tan pronto como un botón M es presionado (evento m), ambos tanques son llenados por la abertura de las válvulas, $V_1 = V_2 = 1$ significa que la válvulas están abiertas. Cuando el tanque 1 está lleno, se detiene el llenado (se cierra la válvula V_1) y el contenido comienza a salir (se abre la válvula W_1). Cuando el tanque está vacío, la válvula W se cierra. El llenado puede solamente comenzar cuando ambos tanques estén vacíos, el siguiente tiempo en que M será presionado.*

El modelo en la Figura 6-4 (a) representa estas condiciones. Los lugares etiquetados del 1 al 6 son pasos. En el tiempo inicial, los pasos en el conjunto {1,4} están activos

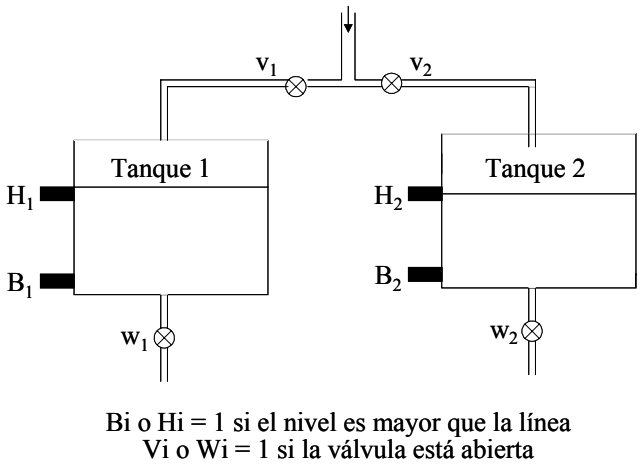

Figura 6-3: Ejemplo 2: Llenado de tanques.

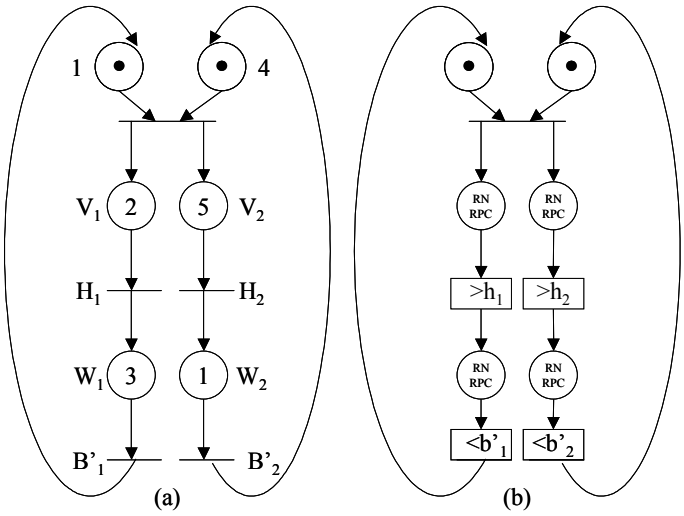

Figura 6-4: Dos modelos correspondientes a las misma especificaciones. (a) Las entradas son eventos y variables Booleanas, las salidas son variables Booleanas (Red de Petri). (b) Todas las entradas y salidas son eventos (Red Híbrida).

(representados por los lugares 1 y 4). Entonces la transición (1) que sigue estos pasos puede ser disparada tan pronto como el evento m, asociado con (1), ocurre. Después de este disparo, los pasos que están activos son el 2 y el 5. Cuando el paso 2 está activo, la salida $V_1 = 1$ (representada por V_1 escrito en un rectángulo asociado con el paso 2) y la transición (2) puede ser disparada si $H_1 = 1$.

Los pasos 1, 2 y 3 corresponden a los estados del tanque 1 (vacío, durante el llenado, y durante el vaciado respectivamente) y los pasos 4, 5, y 6 corresponden a los estados del tanque 2. Uno puede observar que los tiempos cuando las transiciones son disparadas dependen de las entradas al controlador (X en la Figura 6-2 (a)), a decir verdad, el evento m asociado con la transición (1) y las variables Booleanas H_1, H_2, B_1', y B_2', asociadas con las transiciones (2), (4), (3), y (5), respectivamente, corresponden a la información que viene desde el proceso. El evento (por ejemplo, m) o condición (por ejemplo, H_1) asociada con las transiciones son llamados *receptividades.* Las *acciones* V_1, V_2, W_1 y W_2 asociadas con algunos pasos de el modelo corresponden a las salidas del controlador (Z en la Figura 6-2 (a)). En este ejemplo, todas las acciones son *acciones nivel*, es decir, variables Booleanas. También, es posible tener *acciones impulso* correspodientes a los eventos (cuendo se utilizan eventos en lugar de variables Booleanas). En este caso, la acción impulso asociada con un paso i es ejecutado exactamente cuando el paso i pasa del estado inactivo al estado activo si es una acción incondicional (el caso donde es condicional se ilustrará en la Sección *6.4*).

6.3.2 Uso de eventos en un modelo híbrido

Como ejemplo, vamos a mostrar que todas las receptividades y todas las acciones pueden ser eventos en un modelo híbrido, ver Figura 6-4 (a).

1) *Eventos en lugar de condiciones para receptividades:* Cuando el Paso 2 se vuelve activo al disparo de (1), el tanque 1 está vacío por que el Paso 1 se alcanza cuando este tanque está vaciado. Se tiene que $H_1 = 0$ a este tiempo. Ahora el tiempo cuando la receptividad H_1, asociada con la transición (2), toma el valor uno, es exactamente el tiempo cuando el evento $h_1 =\uparrow H_1$ ocurre. Por lo tanto, la receptividad H_1 (condición) puede ser reemplazada por h_1 (evento). Este razonamiento es el mismo para las receptividades asociadas con las

Figura 6-5: Un sistema de manufactura.

transiciones (3), (4) y (5).

2) *Acciones impulso (eventos) en lugar de acciones nivel (variables Booleanas):* La acción nivel V_1 asociada con el Paso 2 es tal que $V_1 = 1$ exactamente cuando el Paso 2 está activo. Esto significa que $v_1 =\uparrow V_1$ ocurre cuando el Paso 2 se vuelve activo y que $v_1' =\downarrow V_1$ ocurre cuando el Paso 2 se vuelve inactivo, es decir, cuando el Paso 3 se vuelve activo. Entonces, la acción nivel V_1 en la Figura 6-4 (a) puede ser reemplazada por un lugar (con una red neuronal) o un lugar continuo (RN_σ o RPC) como se ve en la Figura 6-4 (b): v_1 está asociada con el Paso 2 y v_1' asociada con el Paso 3. Una transformación similar puede ser desempeñada por las acciones nivel V_2, W_1 y W_2.

Observación 3: En el ejemplo de la Figura 6-4, ha sido obtenido un modelo híbrido, donde todas las entradas y salidas son eventos. Esto no es siempre el caso y la transformación mencionada no proveé un algoritmo general. Solamente enfatizamos el hecho de que, si los estados iniciales de las variables Booleanas en la entrada de el controlador lógico son conocidas, entonces un modelo híbrido puede ser siempre obtenido. En este modelo todas las entradas y salidas son eventos. En general, puede ser necesario ir hacia atrás en las especificaciones a manera de diseñar el modelo requerido.

6.3.3 Especificaciones para el control

Ejemplo 6.2 *Consideremos un sistema de manufactura (Figura 6-5) compuesto de dos máquinas idénticas M_1 y M_2, trabajando independientemente., como se muestra en la Figura 6-6. El inicio del trabajo, el evento w_i se supone que sea simultáneo con la recogida de una parte. Así pues, el transporte en M_i está modelado, desde el estado ocupado (correspondiente al trabajo por terminar y a la parte por empezar M_i), el comienzo del transporte, el evento t_i lleva al estado transporte. Entonces el final del transporte (evento f_i) lleva la máquina al*

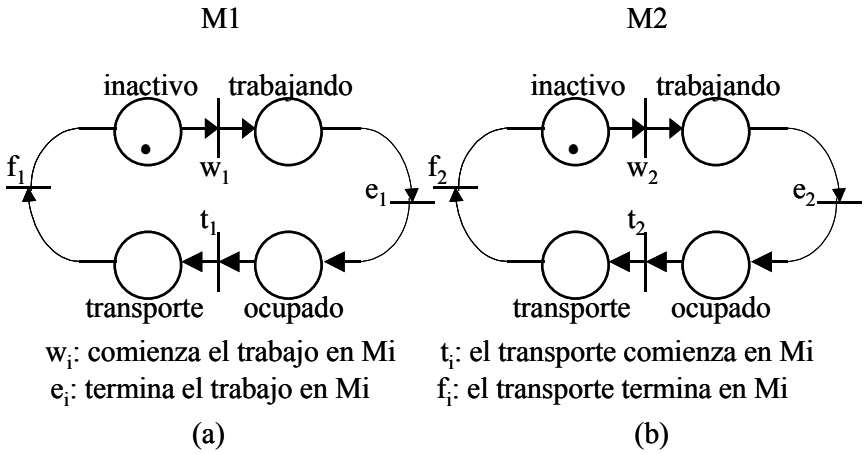

Figura 6-6: Modelos del sistema de manufactura. (a) Modelo de M_1. (b) Modelo de M_2.

estado inactivo donde un nuevo ciclo puede empezar.

Especificación 1: Las dos máquinas se consideran que trabajan en serie (una detrás de otra) y el comportamiento del sistema debe respetar la presencia de un *buffer* que está colocado entre M_1 y M_2. Esto significa que: 1) una parte visitará primero M_1 entonces M_2; 2) M_1 puede o no puede poner una parte en el *buffe*r si el último está lleno; y 3) M_2 no puede comenzar un ciclo, si no puede tomar una parte del *buffer* (cuando el *buffer* está vacio). Se considera que el *buffe*r tiene una capacidad de uno y que está vacío en el estado inicial.

Especificaciones para el control: Imponemos a cada máquina M_i que trabaje como sigue: En su estado inicial, M_i está inactiva. M_i debe empezar a trabajar inmediatamente después de la toma de una pieza. Cuando el final del trabajo ocurre entonces la transferencia de la pieza debe comenzar. Un nuevo ciclo podrá comenzar tan pronto como la transferencia termine.

Con estas especificaciones, el comportamiento de las máquinas M_i es similar a aquellas de la Figura 6-6. Estas especificaciones del control pueden ser modeladas por la red de Petri de la Figura 6-7 donde w_i, e_i, t_i y $f_i (i = 1, 2)$ tiene el mismo significado que en la Figura 6-6. Consideremos el modelo del control para M_1 de la Figura 6-7 (a): Los pasos 1 (M_1 está trabajando) y 2 (la transferencia comienza a procesarse hacia M_1) corresponden a los estados de M_1. En el estado inicial, el paso 1 está activo y la acción impulso w_1 es ejecutada.

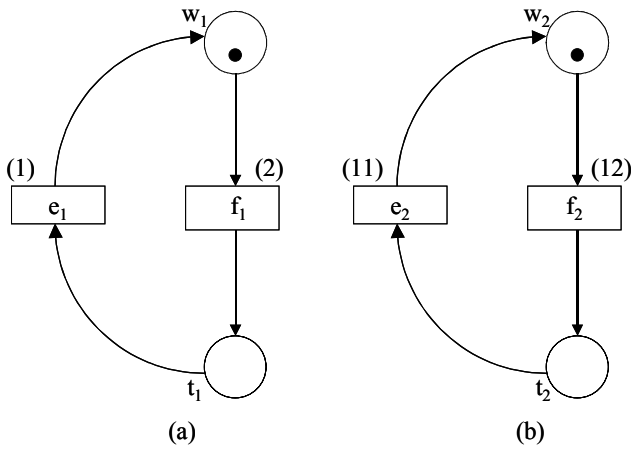

Figura 6-7: Especificaciones del control para el sistema de manufactura. (a) Modelo híbrido para M_1. (b) Modelo híbrido para M_2.

Tan pronto como el evento e_1 ocurre, t_1 es ejecutado. Entonces la ocurrencia de f_1 lleva al modelo de la Figura 6-7 (a) a su estado inicial. Uno puede observar en la Figura 6-7 que todas las entradas y todas las salidas de el controlador lógico son eventos. De acuerdo a la Figura 6-2 (c), el conjunto de eventos $\Sigma_{pr} = \{e_1, f_1, e_2, f_2\}$ y $\Sigma_{co} = \{w_1, t_1, w_2, t_2\}$ son producidos por el proceso actual mediante el controlador lógico.

Del modelo de la Figura 6-7, un árbol del controlador lógico puede ser obtenido. Esto está ilustrado en la Figura 6-8 (a). Cuando el sistema comienza a andar, los pasos 1 y 11 en la Figura 6-7 se vuelven activos (estado {1, 11}) y las acciones impulso w_1 y w_2 son ejecutadas inmediatamente. Denotemos $w = \{w_1, w_2\}$ al evento que es la ocurrencia simultánea de w_1 y w_2. Así, un arco entrante etiquetado por w está asociado con el estado inicial {1, 11} en la Figura 6-8 (a). Recordemos que la etiqueta de un arco entre dos estados no es necesariamente la causa de la transición, le corresponde al evento el cual es *simultáneo* con la transición. Similarmente, el arco etiquetado w significa que el evento w ocurre en la inicialización.

En nuestro ejemplo, e_1 y e_2 no pueden ser simultáneos desde el estado {1, 11}. Suponemos que e_1 ocurre primero, entonces tan pronto como e_1 ocurre, el paso 2 de la Figura 6-7 es activado (estado {2, 11}) y la acción impulso t_1 es ejecutada inmediatamente. Por estas razones debemos sugerir la siguiente hipótesis para casos generales en los cuales deben existir eventos que no son simultáneos.

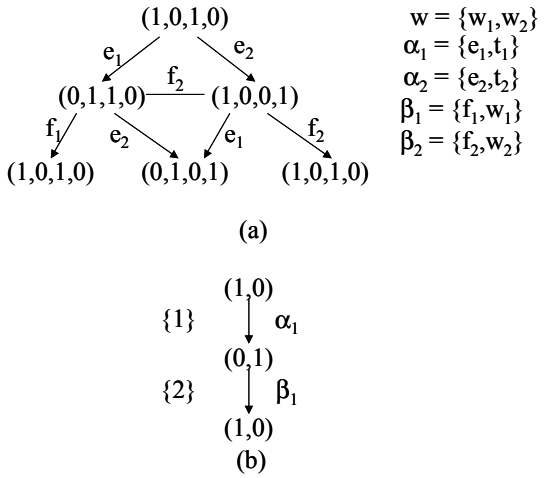

Figura 6-8: Árbol de estados del proceso extendido. (a) Árbol global. (b) Árbol para M_1.

Hipótesis 1: Dos eventos externos, eventos de Σ_{pr}, no pueden ocurrir simultáneamente.

Para un observador externo, los eventos e_1 y t_1 ocurren simultáneamente. Esto es representado por el evento $\alpha_1 = \{e_1, t_1\}$ (α_1 representa el evento que es la ocurrencia simultánea de e_1 y t_1) asociado con la transición {1, 11}→{2, 11} de la Figura 6-8 (a).

Observación 4: Si un controlador está compuesto de varios modelos, uno puede considerar el obtener un modelo híbrido del controlador desde los modelos híbridos de diferentes sistemas.

El árbol para M_1, obtenido de la Figura 6-7 (a), es mostrado en la Figura 6-8 (b). En este árbol, los estados {1} y {2}corresponden a los Pasos 1 y 2 del modelo (Figura 6-7 (a)). Recuerdese que tan pronto como el Paso 1 de la Figura 6-7 (a) se vuelve activo (la maquina se vuelve inactiva), w_1 es ejecutado (el ciclo de trabajo empieza). Por lo tanto, M_1 permanece un tiempo en el estado *inactivo.* Comparado con el modelo de M_1 de la Figura 6-6 (a), uno puede notar que los estados *inactivo* y *trabajando* están fusionados en el modelo de la Figura 6-7 (a). La fusión de estos dos estados corresponden al estado {1} de la Figura 6-8 (b). Similarmente, la fusión de los estados *ocupado* y *transporte* corresponden al estado {2} por que, cuando la máquina está aislada, una parte terminada es inmediatamente transportada. La simultaneidad de algunas entradas con algunas salidas es solamente debido al modelo híbrido.

Observación 5: La producción no bloqueada puede ser direccionada mediante la especificación de un conjunto de pasos marcados en el modelo híbrido.

En nuestro ejemplo, se tiene que los Pasos 1 y 11 de la Figura 6-7 están marcados, entonces el estado {1, 11} en la Figura 6-8 sería un estado marcado.

Note que en la Figura 6-7 el control se efectua cuando un evento de Σ_{pr}, la entrada del controlador lógico ocurre. Si se tiene que el evento σ de Σ_{pr} ocurre: las transiciones asociadas con σ y por las cuales los pasos primeros están activos, son entonces disparadas. El árbol de la Figura 6-8 (a) puede ser directamente obtenido desde el modelo híbrido de el controlador de la Figura 6-7 mediante la aplicación del siguiente algoritmo.

Algoritmo 1:

Paso 1) Sea r_0 el estado inicial, el conjunto de pasos iniciales del modelo híbrido. Inicializamos el modelo con el estado r_0. El arco de entrada de r_0 es etiquetado por el evento que modela la ocurrencia simultanea de las acciones impulso asociadas con los pasos iniciales de el modelo híbrido. Denotemos L como la lista de estados a ser tratados, y conjuntamos $L = \{r_0\}$.

Paso 2) Considerar un estado r_k de L, el conjunto de pasos activos en el modelo híbrido. Definir todos los estados r_j que pueden ser alcanzados desde r_k. Para cada estado r_j: 1) si r_j no existe en el modelo entonces adicionar r_j al modelo y adicionar r_j a L; 2) adicionar un arco $r_k \rightarrow r_j$ asociado con el evento que corresponde al cambio desde r_k a r_j.

Paso 3) Remover r_k desde L. Si l no está vacío, regresar al Paso 2.

Note que no hay hipótesis hecha sobre la estructura del modelo híbrido. Por lo que, solamente nos enfocaremos en modelos que tienen eventos como entradas y salidas. En este caso, suponemos que un evento σ de Σ_{pr} causa el cambio desde el estado r_k a el estado r_j, entonces la transición $r_k \rightarrow r_j$ tiene que ser etiquetada por el evento $\tau = \{\sigma, a_1, a_2, ..., a_n\}$ donde $a_i (i = 1, ...n)$ es una acción impulso asociada con un paso que está activada desde r_k a r_j (Paso 2).

En general , una acción tiene como un resultado el que pueda ser observada por el controlador lógico. En nuestro ejemplo, la ocurrencia de e_i, asociada con la transición (1) en la Figura 6-7, es observada por el controlador lógico como una consecuencia de w_1, de manera que comience la transferencia (evento t_1). En el modelo de la Figura 6-7, los eventos

e_i y f_i son, respectivamente, las consecuencia de las acciones w_i y t_i asociadas con los pasos precedentes. Entonces podemos sugerir las siguientes hipótesis que serán consideradas para próximos resultados.

Hipótesis 2: El evento asociado con una receptividad es la consecuencia de la ejecución de las acciones que están asociadas con los primeros pasos desde la transición.

Note que e_1 puede ser producido por el proceso si y solo si la acción nivel w_1 ha sido ejecutada, es decir, cuando el Paso 1 de la Figura 6-7 está activo. De hecho, resulta que de la Hipótesis 2, la ocurrencia de un evento es una condición necesaria y suficiente si el controlador lógico puede tomarlo en cuenta, dicho de otra manera, existe una transición del controlador que puede ser disparada cuando este evento ocurre. Por consiguiente, el controlador lógico contiene suficiente información para describir el comportamiento de el proceso extendido. El árbol del controlador lógico de la Figura 6-8 (a) es equivalente a el modelo del proceso extendido, esto es, M_1 y M_2 acoplados con el controlador lógico de la Figura 6-7.

Observación 6: La Hipótesis 2 no permite obtener el modelo del proceso extendido desde el conocimiento de el sistema de control, solamente. En la práctica, se vé que la Hipótesis no es restrictiva. Si la Hipótesis 2 no se mantiene, es necesario ir hacia atrás de las especificaciones para obtener un modelo del proceso extendido.

6.3.4 Especificaciones para la supervisión

Los modelos híbridos también puede ser utilizado para las especificaciones de la supervisión. En nuestro ejemplo, consideramos las especificaciones descritas en la *especificación 1* (Sección *6.2.3*). Las cuales pueden ser modeladas como en la Figura 6-9. Este modelo proveé dos salidas Booleanas: F_{w2} y F_{t1}, es decir, prohibiendo los eventos w_2 y t_1, respectivamente. En el estado inicial, $\{21\}$, la *acción nivel* F_{w2} a lo largo del Paso 21 está activa. Esto significa que desde el estado $\{21\}$ que corresponde al *buffer* en estado vacío, el comienzo del trabajo de M_2 (el evento w_2) es prohibido por el supervisor. Cuando el evento f_1 ocurra entonces el estado $\{22\}$ es alcanzado y el *buffer* está lleno. En este estado, $F_{t1} = 1$ significa que el supervisor prohibe el comienzo de la transferencia en M_1.

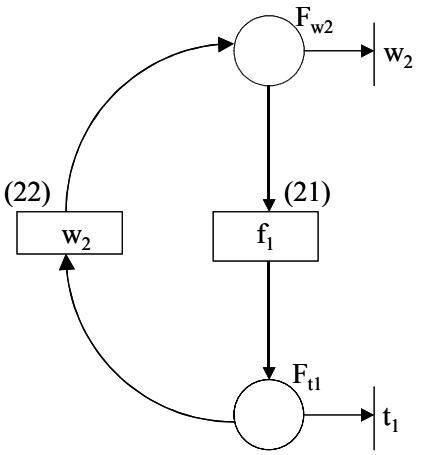

Figura 6-9: Especificaciones para el supervisor.

Observación 7: De acuerdo a la Figura 6-2 (c), las entradas a el modelo de la supervisión corresponden a las salidas de el proceso extendido, $\Sigma = \Sigma_{pr} \cup \Sigma_{co}$. En general, las salidas de el modelo son variables Booleanas, F_σ donde $\sigma \in \Sigma_{pr} \cup \Sigma_{co}$ significando que el evento σ es prohibido cuando $F_\sigma = 1$. Consideremos el caso donde $\sigma \in \Sigma_{pr}$ lo que necesariamente implica que $\sigma \in \Sigma_u$. En este caso, aunque el supervisor no puede prohibir la ocurrencia de σ, $F_\sigma = 1$ significa que la ocurrencia de σ puede tal vez no ocurrir. Note que esto es consistente con la definición de una especificación (Sección *6.2.2*).

6.4 Ejemplo de control de sistemas híbridos

En nuestro ejemplo *6.2*, las especificaciones del supervisor pueden ser forzadas por el acople del modelo híbrido de la Figura 6-9 con el proceso extendido (Figura 6-7). Recordemos que este modelo del supervisor tiene que prohibir la ocurrencia de los *eventos controlables potenciales* t_1 y w_2. Por consiguiente, estos eventos tienen que ser controlables y conjuntamos: $\Sigma_c = \{t_1, w_2\}$ y $\Sigma_u = \{w_1, t_2, e_1, f_1, e_2, f_2\}$.

Observación 8: Si bien w_1 y t_2 son *eventos controlables potencialmente,* nunca necesitaremos prohibirlos por el supervisor. Entonces los dejaremos en el conjunto Σ_u de eventos incontrolables. (Por lo cual, uno puede considerar a todos los eventos de Σ_{co} como eventos controlables.) Si la ruta tiene que ser invertida, es decir, una parte debe visitar M_2 entonces

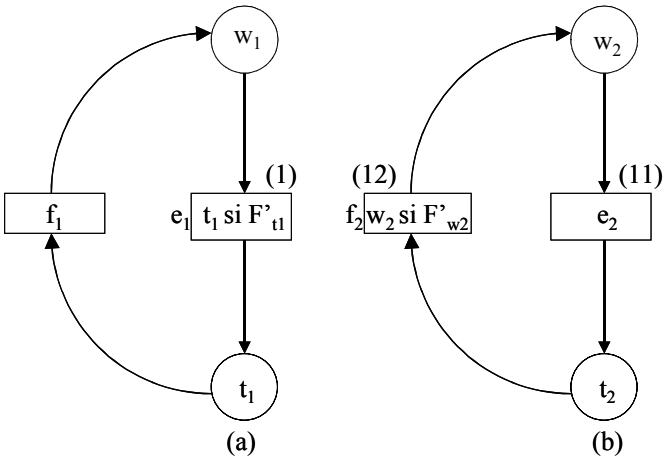

Figura 6-10: Proceso extendido bajo supervisión. (a) Modelo híbrido de M_1. (b) Modelo híbrido de M_2.

M_1, entonces w_1 y t_2 podrían ser eventos controlables.

Uno tiene que modificar el controlador de forma que esté disponible para prohibir algunas de sus salidas (de hecho, uno no puede prohibir la ocurrencia de t_1 o w_2 de la Figura 6-7). Nos referiremos al sistema que cumple con las condiciones mencionadas, como el *proceso extendido bajo supervisión.*

6.4.1 El proceso extendido bajo supervisión

En nuestro ejemplo, el modelo del *proceso extendido bajo supervisión,* obtenido de la Figura 6-7, es mostrado en la Figura 6-10. Consideremos las especificaciones del control para M_1 (Figura 6-10 (a)). La acción impulso t_1, asociada con el paso 2 (Figura 6-7 (a)), ha sido reemplazada por la acción impulso condicional $[t_1 \text{ si } F'_{t1}]$ de la Figura 6-10 (a), donde F'_{t1} es la variable Booleana la cual es el complemento de F_{t1}. Dos casos son posibles e ilustrados en la Figura 6-11. Denotemos como X_2 el valor Booleano que es igual a uno exactamente cuando el Paso 2 de la Figura 6-10 (a) está activo. En la Figura 6-11 (a), cuando el valor de X_2 cambia de cero a uno, esto es, el Paso 2 se vuelve activo, el valor de F_{t1} es igual a cero, $F'_{t1} = 1$, y la acción impulso t_1 es ejecutada. El caso donde $F_{t1} = 1$ cuando el Paso 2 se vuelve activo está ilustrado en la Figura 6-11 (b): t_1 será ejecutada exactamente la próxima vez que F_{t1} tome el valor de cero. Se tiene que, si la condición es verdadera, una

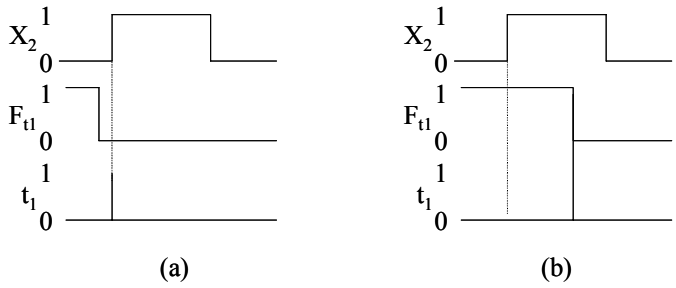

Figura 6-11: Ejecución de una acción impulso condicional.

acción impulso condicional asociada con un paso i es ejecutada tan pronto como el paso i se vuelve activo, la acción es ejecutada en la primera instancia de la condición. Igualmente para M_2, uno reemplaza la acción impulso w_2 asociada con el Paso 11 de la Figura 6-7 (b) por la acción impulso condicional [w_2 si F'_{w2}] de la Figura 6-10 (b).

Un modelo híbrido del proceso extendido bajo supervisión puede ser obtenido desde el proceso extendido mediante el remplazamiento de cada acción impulso σ, con $\sigma \in \Sigma_c$, mediante la acción impulso condicional: [σ si F'_σ].

6.4.2 Obtención del modelo del proceso extendido bajo supervisión

Uno puede obtener un modelo del proceso extendido bajo supervisión por la transformacón del modelo híbrido. En nuestro ejemplo, el modelo de la Figura 6-10 puede ser transformado de acuerdo al modelo de la Figura 6-12. Esto actualmente corresponde a la división de dos estados representandos como *ocupado* y *transporte*, mientras la máquina no está aislada. Esto está basado en la siguiente regla.

Regla de Construcción 1: Para cada paso k asociado con una acción impulso condicional (ejemplo [t_1 si F'_{t1}]) asociada con el Paso 2 de la Figura 6-10 (a): 1) adicionamos una transición (k') , en nuestro caso (2') de la Figura 6-12, desde el paso k, asociado con la condición de la acción (F'_{t1}); 2) adicionamos un Paso k' (el paso 2') desde la transición (k'); y 3) removemos la acción de impulso condicional del Paso k ([t_1 si F'_{t1}] del Paso 2) y adicionamos su correspondiente acción impulso incondicional al Paso k' ([t_1] al Paso 2').

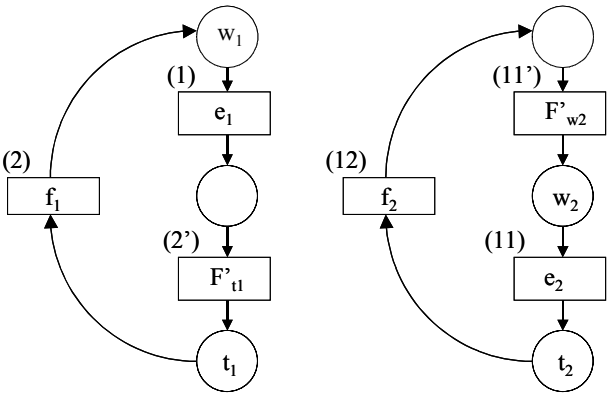

Figura 6-12: Modelos equivalentes a los de la Fig.6-10.

La mencionada regla se establece solamente para modelos que tienen al menos una acción impulso condicional asociada con un paso. Así pues, esta regla puede ser generalizada. Suponiendo que la Hipótesis 2 se mantiene entonces tenemos las siguiente propiedad.

Propiedad 1: Sea G'un modelo obtenido de un modelo G por la regla de construcción 1. Entonces G' tiene el mismo comportamiento entrada/salida como G.

Se tiene que el modelo de la Figura 6-12 es equivalente desde el punto de vista de control y supervisión a el modelo de la Figura 6-10. Los árboles obtenidos desde los modelos de la Figura 6-12 son mostrados en la Figura 6-13 (a) y (b). Consideremos M_1 [Figura 6-13 (a)]: al tiempo inicial, el Paso 1 de la Figura 6-12 (a) es activado y w_1 es ejecutado inmediatamente. Esto es representado en la Figura 6-13 (a) por el arco entrante etiquetado por el evento w_1 asociado con el estado $\{1\}$. Cuandos e_1 ocurra, el Paso 2 de la Figura 6-12 (a) es activado. En este tiempo, si F_{t1} es igual a cero ($F'_{t1} = 1$), el Paso 2 es inmediatamente desactivado, el paso 2' es activado, y t_1 es ejecutado (el estado donde el Paso 2 es activo se dice que es *inestable).* Para un observador externo, los eventos e_1 y t_1 ocurren simultaneamente. Esto es representado en la Figura 6-13 (a) por la transición $\{1\}\rightarrow \{2'\}$ etiquetada por el evento $\alpha'_1 = [\{e_1, t_1\} \cdot F'_{t1}]$ (α'_1 representa la ocurrencia simultánea de e_1 y t_1 cuando sea $F_{t1} = 0$). Desde un punto de vista teórico, esto es consistente (con la teoría RW) mientras el producto de un evento por una variable Booleana es un evento. De lo contrario, si $F_{t1} = 1$ cuando el evento e_1 ocurra, la transición (2') puede ser no disparada (el estado donde el Paso 2 es activo y se dice estar estable). El evento $\alpha''_1 = [\{e_1\} \cdot F_{t1}]$ que lleva al estado $\{1\}$ al estado

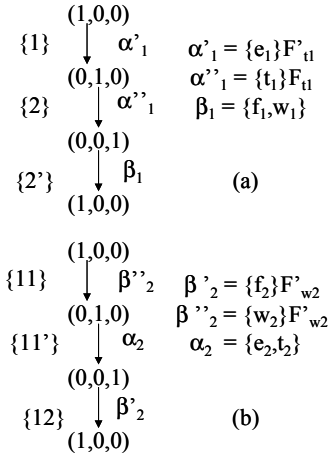

Figura 6-13: Árboles de el proceso extendido bajo supervisión, obtenidos de la Fig.6-12. (a) Árbol para M_1. (b) Árbol para M_2.

$\{2\}$ en la Figura 6-13 (a), modela la ocurrencia de e_1 cuando ocurra que $F_{t1} = 1$, y asi sucesivamente.

Uno obtiene el árbol para M_2 en la Figura 6-13 (b) del modelo de la Figura 6-12 (b). Los árboles de la Figura 6-13 están construidos en el conjunto de eventos $\Sigma' = \{w_1, \alpha'_1, \alpha''_1, \alpha'''_1, \beta_1, \alpha_2, \beta'_2, \beta''_2, \beta'''_2\}$ donde cada evento en Σ' es la ocurrencia simultánea de algunos eventos de Σ (posiblemente condicionados por algunas variables Booleanas F_σ, con $\sigma \in \Sigma_c$). Los eventos β_1 y α_2 mantienen el mismo significado que aquellos de la Figura 6-8 (a). Los eventos α_1 y β_2 que contienen los eventos controlables t_1 y w_2, respectivamente, en la Figura 6-8 (a), son reemplazados por $\alpha'_1, \alpha''_1, \alpha'''_1$ y por $\beta'_2, \beta''_2, \beta'''_2$ en la Figura 6-13.

Observación 9: Los árboles de la Figura 6-13 pueden ser obtenidos desde los modelos de la Figura 6-12 por la aplicación del algoritmo 1 (esto consiste en la búsqueda de todos los *estados estables* que pueden ser alcanzados en el modelo, observando los valores que pueden tomar las variables Booleanas F_{t1} y F_{w2}).

Uno puede obtener un modelo global del proceso extendido bajo supervisión. Esto puede ser realizado simplemente por el cómputo del producto de los árboles de la Figura 6-13 (a) y (b), obteniendo al árbol de la Figura 6-14, donde cada estado corresponde a el estado de la Figura 6-13 (a), el estado de M_1, y a el estado de la Figura 6-13 (b), el estado de M_2. Sobre todo, etiquetamos al arco entrante de el estado inicial, $\{1, 11\}$, por el evento que

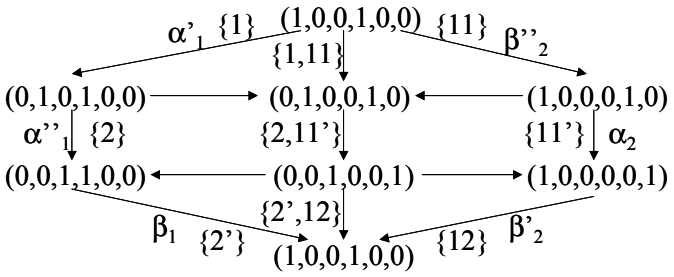

Figura 6-14: Árbol global de el proceso extendido bajo supervisión.

corresponde a la ocurrencia simultánea de los eventos asociados con el arco entrante de los estados iniciales de la Figura 6-13 (a) y en (b). w_1 es asociada con el arco entrante de {1, 11} en la Figura 6-14. Esto significa que la ocurrencia de w_1 ocurre en el estado inicial. Por lo tanto, la ocurrencia de w_1 precede la ocurrencia de otros eventos.

Los eventos α_1' y α_1''' (Figura 6-13) contienen al evento controlable t_1 y son condicionados con la variable Booleana F_{t1}'. Esto implica que es posible prevenir a cualquier tiempo la ocurrencia de α_1' y α_1'''. De hecho, si F_{t1} es igual a 1, t_1 es prohibido, α_1' y α_1''' no pueden ocurrir. A manera de corolario, un evento de Σ' que no contiene cualquier evento controlable de Σ es incontrolable. Entonces la siguiente propiedad se cumple.

Propiedad 2: Cualquier evento en Σ' que contiene un evento controlable de Σ, es decir, que es condicionado por cualquier variable Booleana $F_\sigma', \sigma \in \Sigma$, es controlable.

En nuestro ejemplo, $\Sigma_c' = \{\alpha_1', \alpha_1''', \beta_2', \beta_2'''\}$ y $\Sigma_u' = \{w_1, \alpha_1'', \beta_1, \alpha_2, \beta_2''\}$ son conjuntos de eventos controlables e incontrolables sobre Σ', respectivamente.

6.4.3 Implementación del control supervisado

Como se ilustra en la Figura 6-15, el control supervisado de nuestro sistema de manufactura puede ser logrado simplemente por la implementación conjunta del árbol de control de la Figura 6-8 y el modelo de supervisión de la Figura 6-9.

Observación 10:

1. Dado un conjunto de estados marcados en el modelo del proceso extendido, es posible verificar no tener bloqueos. Esta propiedad puede ser probada utilizando las técnicas

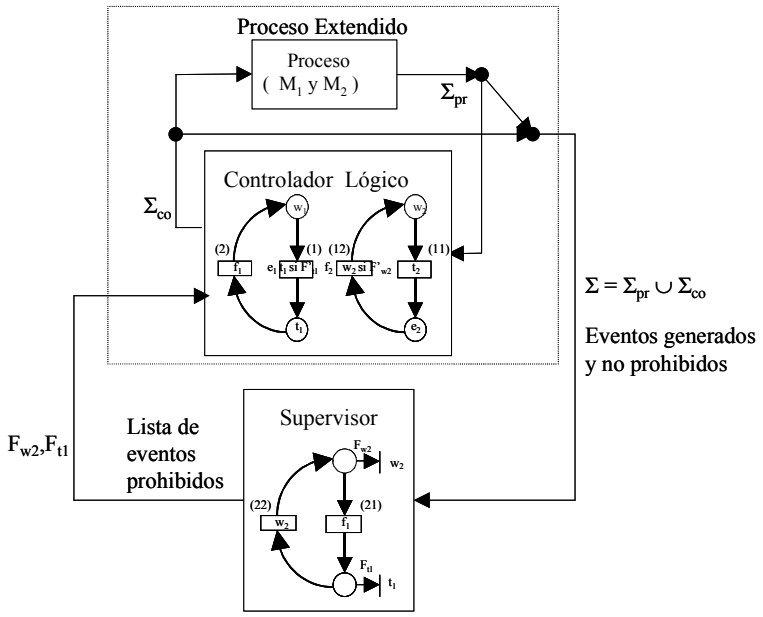

Figura 6-15: Esquema del control supervisado de el ejemplo del sistema de manufactura.

de [40], asi se garantiza que para cualquier estado no marcado que pueda ser alcanzado por el proceso extendido acoplado con el supervisor, existe una secuencia no prohibida de eventos que lleva al proceso a cualquier estado, marcado. En nuestro ejemplo, los Pasos 1 y 11 de la Figura 6-8 están marcados, es decir, el estado {1, 11} en la Figura 6-14 es un estado marcado, entonces el comportamiento de lazo cerrado de el sistema de manufactura se puede demostrar ser *no bloqueado.*

2. En este capítulo, la prueba de controlabilidad requiere el cambio a modelos híbridos. Uno de nuestros objetivos es ser habilitable el hacer pruebas y el sintetizar supervisores directamente desde modelos híbridos de manera de evitar la explosión de estado que se presenta en modelos autómatas.

6.5 Conclusión

En este capítulo se definie el *concepto de control supervisado.* Con la ayuda de un ejemplo de sistema de manufactura, donde la separación del control y el supervisor no incrementan la consistencia del problema de control. El uso del modelo híbrido hace la implementación

de un sistema de control supervisado sencilla. También, el modelo híbrido y los algoritmos dados no llevan a asegurar que la prueba de controlabilidad también se mantiene en los modelos implementados. Hay que señalar que este capítulo se basó en el articulo [9], en el cual la construcción del control supervisado se basa en autómatas, e incluso la prueba de controlablilidad queda sujeta a la teoria RW dada en [49]. En el ejemplo presentado en el capítulo, observese que de las especificaciones de comportamiento, uno puede obtener el controlador de la Figura 6-10,dado el controlador de la Figura 6-7 y el modelo del supervisor de la Figura 6-9. Las Figuras 6-8-6-12 y los correspondientes comentarios han sido presentados para mostrar la consistencia de esta aproximación con la teoría RW. El concepto de control supervisado brinda un buen marco de trabajo para posteriores investigaciones, en las cuales se debe de particularizar las pruebas de controlabilidad a los casos presentados en esta tesis, así como la implementación del control supervisado en un sistema real.

Capítulo 7

Conclusiones y trabajos futuros

En el capítulo *3* se presentó una red de Petri difusa modificada: La Red de Petri Difusa Adaptiva (RPDA) la cual tiene la habilidad de aprendizaje como una red neuronal. De este hecho, el modelo aventaja a los demás modelos citados en la literatura, en que puede hacer frente a los cambios dados en un sistema experto. Por el contrario su debilidad redica en que requiere de ciertos conocimientos a priori para funcionar (caso del algortitmo de retropropagación). Siendo la idea mostrada en este capítulo una nueva manera formal de resolver el problema del aprendizaje de conocimiento en sistemas expertos.

En el capítulo *4* se presenta un nuevo método para el modelado de sistemas híbridos mediante redes de Petri y rede neuronales. Así nace la estuctura H la cual se divide en dos partes; una red de Petri y la otra compuesta de redes neuronales. Esto en base a la visión de considerar a los sistemas híbridos formados de dos partes: el sistema de conocimiento y el sistema continuo. De hecho la construcción del modelo híbrido parte de estos preceptos:

1. Separamos el comportamiento lógico del sistema para representarlo con una red de Petri difusa.

2. El modelo lógico establece en que lugares de la red se encuentran las partes continuas, para representarlas con las redes neuronales.

3. Por último se establecen los mecanismos de ejecución de la red, es decir, la mecánica de disparo de las transiciones y la asignación de los valores para los *tokens.*

En el capítulo *3* se dotaban a las redes de Petri difusas de la habilidad de aprender, utilizando una estructura semejante a una red neuronal, de esta idea nació la propuesta de utilizar redes neuronales, esto garantiza el respaldo que se tiene de toda la teoría neuronal desarrollada en años pasados y no tener que desarrollar una nueva para las redes de Petri difusas adaptables (aunque de todas formas el análisis de convergencia se hace a través de la conversión de la RPDA a una red neuronal).

Por último hay que resaltar que en la definición de H en el caso de utilizar una estructura difusa se debe profundizar en como se da la relación de las proposiciones D en los lugares P_N y como se haría el análisis para esta red, en otras palabras, quedan abiertas estas cuestiones para futuros trabajos.

El capítulo *5* presentó un nuevo método para el modelado de sistemas híbridos. En base a la unión de una red de Petri difusa con una red de Petri continua (las propuestas de redes de Petri comunes con redes continuas ya tienen un importante respaldo en la literatura). Al contrario del modelo híbrido obtenido con una red neuronal, donde se tenian lugares asociados con las redes neuronales (representadas gráficamente dentro de los lugares), en esta estructura se visualiza la separación existente entre la RPD y la RPC, las cuales se unen por sus respectivas transiciones (ver definición *5.3*). La construcción del modelo puede llegar a ser compleja, dada la cantidad de procesos continuos que contenga el sistema a modelar. No se ha realizado una comparación formal con el arreglo obtenido con redes neuronales, ya que para hacer una comparación eficaz del desempeño de ambos modelos es necesario tomar en cuenta el análisis de estas redes de Petri híbridas.

En el capítulo *6* se definió el *concepto de control supervisado.* Con la ayuda de un ejemplo de sistema de manufactura, donde la separación del control y el supervisor no incrementan la consistencia del problema de control. El uso del modelo híbrido hace la implementación de un sistema de control supervisado sencilla. También, el modelo híbrido y los algoritmos dados no llevan a asegurar que la prueba de controlabilidad también se mantiene en los modelos implementados. Hay que señalar que este capítulo se basó en el articulo [9], en el cual la construcción del control supervisado se basa en autómatas, e incluso la prueba de controlablilidad queda sujeta a la teoria RW dada en [49]. En el ejemplo presentado en el capítulo, observese que de las especificaciones de comportamiento, uno puede obtener el con-

trolador de la Figura 6-10,dado el controlador de la Figura 6-7 y el modelo del supervisor de la Figura 6-9. Las Figuras 6-8-6-12 y los correspondientes comentarios han sido presentados para mostrar la consistencia de esta aproximación con la teoría RW. El concepto de control supervisado brinda un buen marco de trabajo para posteriores investigaciones, en las cuales se debe de particularizar las pruebas de controlabilidad a los casos presentados en esta tesis, así como la implementación del control supervisado en un sistema real.

En resumen, la necesidad de poder representar el comportamiento de sistemas híbridos mediante la construcción de modelos hibridos, ha dado lugar al surgimiento de distintas propuestas sobre los mismos. En el caso de esta tesis los modelos presentados en este trabajo son trabajos originales como en el caso de [44] y [43], mientras que otros están basados en [33], [11]. Esto da una panorámica del estado en que se encuentra el estudio de los sistemas híbridos y las distintas maneras de representación para el desarrollo de modelos mas factibles para su análisis.

El uso de redes neuronales en combinación con redes de Petri ha demostrados ser una de las mejores propuestas debido a que el modelo desarrollado representa de manera confiable el comportamiento de un sistema híbrido. Hay que hacer notar que en los casos estudiados en esta tesis no se tomó en cuenta sistemas en los que el comportamiento de la parte continua fuera no lineal, quedando abierto el tema para futuros trabajos, en los cuales se tendría que incorporar redes neuronales mas complejas para representar el comportamiento no lineal del sistema ó presentar una nueva definición de red de Petri continua la cual pueda representar este comportamiento con distintos arreglos.

Las ventajas en utilizar una rede de Petri difusa adaptable ó una red de Petri híbrida (aqui englobamos las redes combindadas con redes neuronales y redes de Petri continuas), realmente depende del interés en que se quiera enfocar el trabajo. En esta tesis hay que resaltar que el análisis del comportamiento de las redes (la alcanzabilidad de estados) implica un desarrollo teórico de mayor profundidad y además quedó fuera de los objetivos centrales de la tesis. En el marco de este trabajo fue necesario el establecer un esquema teórico de como se implementaría un control supervisado utilizando los modelos desarrollados. Esto fue presentado con un esquema general sin profundizar en casos particulares ya que el pricipal problema encontrado fue el de la poca literatura que aborde el tema, ya que los marcos en

los que se utliza un control supervisado para sistemas híbridos se utilizan en su mayoria autómatas tanto para la construcción como para la prueba de controlabilidad. Por último a manera personal concluyo que descubrí en las redes de Petri una gran felxibilidad, ya que se pueden definir distintas estructuras según las necesidades de los autores. Incluso hay herramientas para modelar que han sido construidas en base a las redes de Petri, tal es el caso del *Grafcet*. Claro que de toda la teoría desarrollada un pequeño porcentaje llega a ser aplicable en la vida cotidiana.

Bibliografía

[1] S. Balemi, G. J. Hoffman, P. Gyugyi, H. Wong-Toi, and G. F. Franklin, "Surpevisory control of a rapid thermal multiprocessor," *IEEE Trans. Automat. Contr.,* vol. 38, July 1993.

[2] A.J. Bugarn and S. Barro, Fuzzy reasoning supported by Petri nets, IEEE Trans. Fuzzy Systems, pp.135-150, 2(2), 1994.

[3] B. G. Buchanan and E. H. Shortliffe, *Rule-Based Expert Systems: The MYCIN Experiments of The Standford Heuristic programming Projects.* Reading, MA: Addison-Wesley, 1984.

[4] Branicky, M. S. (1998). Multiple Lyapunov functions and other analysis tools for switched and hybrid systems. *IEEE Transactions on Automatic Control,* 43, 475-482.

[5] Branicky, M. S., Borkar, V. S., & Mitter, S. K. (1998). A unified framework for hybrid control: model and optimal control theory. *IEEE Transactions on Automatic Control,* 43, 31-45.

[6] Dogruel, M., Ozguner, U., & Drakunov, S. (1996). Sliding-mode control in discrete-state and hybrid systems. *IEEE Transactions on Automatic Control,* 41, 414-417.

[7] T. Cao and A.C. Sanderson, Representation and analysis of uncertainty using fuzzy Petri nets, *J. of Intelligent and Fuzzy Systems*, Vol. 3, 1995, pp. 3-19.

[8] F. Charbonnier, "Commande supervisée des systèmes à evénements discrets," *doctoral dissertation*, Institut National Polytechnique de Grenoble, France, Jan. 1996.

[9] F. Charbonnier, H. Alla and R. David, "The supervised control of discrete-event dynamic systems: A new approach," in 34*th Conf. Decision Contr.,* New Orleans, Dec. 1995.

[10] C. L. Chang, *Introduction to Artificial Intelligence Techniques*. Austin, TX: JMA Press, 1985.

[11] S. Chen, J. Ke, and J. Chang, Knowledge representation using fuzzy Petri nets, *IEEE Trans. Knowledge and Data Engineering*, Vol.2, No.3, 311-319, 1990.

[12] S. M. Chen, "A new approach to handling fuzzy decision-making problems," *IEEE Trans. Syst., Man, Cybern.,* vol. SMC-18, no. 6, pp. 1012-1016, Nov./Dec. 1988.

[13] H. Chen, and H. M. Hanisch, "Analysis of Hybrid Systems Based on Hybrid Net Condition/Event System Model," in *Proc. 3th Int. Conf.on Automation of Mixed Processes*, Reims France, vol. 11, 2001, pp. 163-185.

[14] S. Chen, J. Ke, and J. Chang, Knowledge representation using fuzzy Petri nets, *IEEE Trans. Knowledge and Data Engineering,* pp.311-319, 2(3), 1990.

[15] P. N. Creasy, "An information systems view of conceptual graphs," in *Proc. Int. Comput. Symp.*, Taiwan, vol. 2, 1988, pp. 833-838.

[16] R. David and H.alla, Petri Nets for Modeling of Dynamic Systems-A Survey, *Automatica*, Vol.30, No.2, 175-202, 1994.

[17] R. David and H. Alla, *Petri Nets and Grafcet*, Prentice Hall Int. 1992.

[18] A. Giordana and L. Saitta, "Modeling production rules by means of predicate transition network," *Inform. Sci.*, vol. 35, pp. 1-41, 1985.

[19] A. Gölli and P. Varaiya. "Hybrid Dynamical Syatems," in *Proc. of the 28th CDC*, pp. 2708-2712, Tampa, Fl, Dec. 1989.

[20] K. Hirota and W. Pedrycz, OR/AND neuron in modeling fuzzy set connectives, *IEEE Trans. Fuzzy Systems*, pp.151-161, 2(2), 1994.

[21] J. E. Hopcroft and J. D. Ullman, *Introduction to Automata Theory, Languages, and Computation.* Reading, MA: Addison-Wesley, 1979.

[22] K.J.Hunt, D.Sbarbaro, R.Zbikowski and P.J.Gawthrop, Neural Networks for Control Systems- A Survey, *Automatica*, Vol. 28, 1083-1112, 1992.

[23] S. Haykin, *Neural Networks-A comprehensive Foundation*, Macmillan College Publ. Co., New York, 1994.

[24] K.H.Johansson, J.Lygeros, S.Sastry and M.Egersted, Simulation of Zeno Hybrid Automata, 38th *IEEE Conf. on Decision and Control*, Phoenix, 3538-3543, 1999.

[25] T.J.Koo and S.Sastry, Output Tracking Control Design of a Helicopter Model Based on Approximate Linearization, *IEEE Conf. on Decision and Control*, Florida, 3635-3640, 1998.

[26] J.Lygeros, C.Tomlin and S.Sastry, Controllers for reachability Specifications for Hybrid Systems, *Automatica,* 3, 1999.

[27] W. S. McCullock and W. Pitss, A logical calculus of the ideasimmanent in nervous activity, *Bulletin of Mathematical Biophysics*, vol. 5, 115-133.

[28] G.A.Rovithakis and M.A.Christodoulou, Adaptive Control of Unknown Plants Using Dynamical Neural Networks, *IEEE Trans. on Syst., Man and Cybern.*, Vol. 24, 400-412, 1994.

[29] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain", *Psychological Review*, vol. 65, 1958.

[30] F. Rosenblatt, "*Principles of Neurodynamics*", Spartan Books, Washington D. C., 1962.

[31] E.D.Sontag and Y.Wang, On Characterization of the Input-to-State Stability Property, *System & Control Letters*, Vol.24, 351-359, 1995.

[32] Xiaoou Li and Felipe Lara Rosano, A new fuzzy Petri net model for knowledge representation and reasoning, *the IASTED international Conference on Modeling and Simulation*, Philadelphia, USA, May, 5-8, pp. 322-325, 1999.

[33] Xiaoou Li, Wen Yu, Felipe Lara Rosano, Dynamic Knowledge Inference and Learning Under Adaptive Fuzzy Petri Net Framework, *IEEE Trans. On Systems, Man, and Cybernetics-Part C*, vol.30, no.4, 442-450, 2000.

[34] C. G. Looney and A. R. Alfize, "Logical control via Boolean rule matrix transformations," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-17, no. 6, pp. 1077-1082, Nov./Dec. 1987.

[35] C.G. Looney, *Fuzzy Petri nets and applications, Fuzzy Reasoning in Information, Decision and Control Systems*, edited by Spyros G. Tzafestas and Anastasios N. Venetsanopoulos, Kluwer Academic Publisher, pp 511-527, 1994.

[36] Malmborg, J., Bernhardsson, J. M., & Aström, K. J. (1996). A stabilizing switching scheme for multicontroller systems. *Proceedings IFAC 13th World Congress,* San Francisco, CA (pp. 229-234).

[37] C. V. Negoita, *Expert Systems and Fuzzy Systems.* Massachusetts: Benjamin/Cummings, 1985.

[38] Kumpati Naredra, Learning Automata An Introduction. Pretince Hall 1989.

[39] J. L. Peterson, Petri Nets, Theory and The Modeling of Systems. Englewood Cliffs, NJ: Prentice-Hall, 1981.

[40] J. G. Ramadge and W. M. Wonham "Supervisory control of a class of discrete-event processes," *SIAM J., Contr. Optimization,* vol. 25, pp. 206-230, Jan. 1987.

[41] S. Ribaric, "Knowledge representation scheme based on Petri net theory," *Int. Pattern Recognition Artif. Intell.*, vol. 2, pp. 691-700, 1988.

[42] H. Scarpelli, F. Gomide, and R.R.Yager, A reasoning algorithm for high-level fuzzy Petri nets, *IEEE Trans. Fuzzy Systems*, pp. 282-293, 4(3), 1996.

[43] Sergio M. Perez, Xiaoou Li, Wen Yu, , Adaptive Fuzzy Petri Nets for Supervisory Hybrid Systems Modeling, *15th IFAC World Congress*, Barcelona Spain, 2002.

[44] Sergio M. PerezWen Yu, Xiaoou Li, Modeling Hybrid Systems via Fuzzy Petri Nets and Neural Networks, *8th IEEE International Conference on Emerging Technologies and Factory Automation*, Riviera, French, 2001.

[45] J. F. Sowa, "*Conceptual graph for a database interface*," IBM J. Res. Develop., vol. 20, pp. 336-357, 1976.

[46] J. F. Sowa, *Conceptual Structures: Information Processing in Mind and Machines.* Reading, MA: Addison-Wesley, 1984.

[47] D. Tabak, "Petri net representation of decision models," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-15, no. 6, pp. 812-818, Nov./Dec. 1985.

[48] W. Pedrycz and F. Gomide, A generalized fuzzy Petri net model, *IEEE Trans. Fuzzy Systems,* pp. 295-301, 2(4), 1994.

[49] W. M. Wonham and J. G. Ramadge, "On the supremal controllable sublanguage of a given language," *Siam J. Contr. Optimization,* vol. 25, pp. 637-659, May 1987.

[50] D.S. Yeung and E.C.C. Tsang, A multilevel weighted fuzzy reasoning algorithm for expert systems, *IEEE Trans. SMC-Part A: Systems and Humans*, Vol.28, No.2, 149-158, 1998.

[51] D.S. Yeung and E.C.C. Tsang, Fuzzy knowledge representation and reasoning using Petri nets, *Expert System Application*, Vol.7, pp. 281-290, 1994.

[52] Wen Yu, Alexander S. Poznyak, Xiaoou Li, Multilayer Dynamic Neural Networks for Nonlinear System On-line Identification, *International Journal of Control*, Vol.74, No.18, 1858-1864,2001

[53] Wen Yu and Alexander S.Poznyak, Indirect Adaptive Control via Parallel Dynamic Neural Networks, *IEE Proceedings - Control Theory and Applications*, Vol.146, No.1, 25-30, 1999.

[54] —, "*A theory of commonsense knowledge*," in Aspects of Vagueness, H. J. Skala, S. Termini, and E. Trillas, Eds. Dortrecht: Reidel, 1984, pp. 257-295.