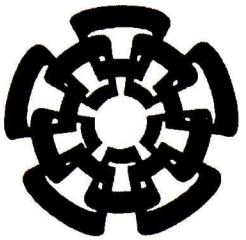
BC-647
Don. 2011

xx (178936.1)



Centro de Investigación y de Estudios Avanzados
del Instituto Politécnico Nacional
Unidad Guadalajara

Traducción de redes de Petri a ecuaciones booleanas para el diseño algebraico de controladores lógicos

Tesis que presenta:
Miriam Díaz Rodríguez

para obtener el grado de:
Maestro en Ciencias

en la especialidad de:
Ingeniería Eléctrica

Director de Tesis
Dr. Luis Ernesto López Mellado

CINVESTAV del IPN Unidad Guadalajara, Guadalajara, Jalisco, Diciembre de 2010.

Traducción de redes de Petri a ecuaciones booleanas para el diseño algebraico de controladores lógicos

Tesis de Maestría en Ciencias
Ingeniería Eléctrica

Por:
Miriam Díaz Rodríguez
Ingeniero en Computación
Universidad de Guadalajara 2003-2007

Director de Tesis
Dr. Luis Ernesto López Mellado

CINVESTAV del IPN Unidad Guadalajara, Diciembre de 2010.

Dedicatoria

Este trabajo está dedicado a mis papás, mi hermana y Alan.
¡Gracias por todo!

Agradecimientos

Agradezco a mi asesor el Dr. Ernesto López Mellado por guiarme y apoyarme durante el desarrollo de esta tesis.

A los doctores Félix Ramos, Raúl González, Mario Siller y Antonio Ramírez por la formación que me dieron durante mi estancia en el Cinvestav.

A mi familia por su apoyo y amor incondicional.

A Roberto por estar a mi lado.

A Abraham, Ana Paula, Carlos, Karina, Oscar y Victor por la valiosa ayuda y amistad que me han brindando.

A Corina, Dan, Erick, Janeth, Martha, Molus y Octavio por todos los buenos recuerdos que compartimos.

Al Consejo Nacional de Ciencia y Tecnología por el apoyo económico que me proporcionó para realizar esta maestría.

Y por último a todos mis amigos y compañeros, que hicieron de esta experiencia algo inolvidable.

Traducción de redes de Petri a sistemas de ecuaciones booleanas para el diseño algebraico de controladores lógicos.

Resumen

Esta tesis aborda el diseño confiable de controladores lógicos a partir de especificaciones para lograr un alto nivel de abstracción; en particular se adopta un método de síntesis algebraica de controladores que recibe como entrada un conjunto de ecuaciones booleanas. El trabajo presentado se enfoca en la traducción automatizada de modelos en redes de Petri interpretadas a conjuntos de ecuaciones booleanas en la forma en que son requeridas por el método de síntesis conocido. Se propone un algoritmo eficiente de traducción y se desarrolla un módulo de software que implementa dicho algoritmo; el módulo es incorporado a una herramienta de edición y análisis de redes de Petri donde es probado con diversos casos de estudio.

Translation from Petri nets into boolean equations systems for the algebraic design of logic controllers

Abstract

This thesis addresses the dependable design of logic controllers from specifications given at a high level of abstraction; in particular an algebraic synthesis method of controllers that processes a set of Boolean equations is adopted. The presented work focuses on the automated translation of interpreted Petri net models into sets of Boolean equations as it is required by the synthesis method. An efficient translation algorithm is proposed and a software module embedding such an algorithm is developed; the module is implemented within an existing tool for Petri nets edition and analysis, where it is tested on several cases of study.

Índice general

Introducción

Contexto

En la actualidad los sistemas de producción discretos han crecido en diversos aspectos tales como su dimensión física, y la variedad y complejidad de las tareas que realizan. Consecuentemente la coordinación de esas tareas es también compleja y el diseño de la entidad de coordinación, llamada controlador, debe llevarse a cabo de manera eficiente y confiable a fin de evitar problemas durante la operación del sistema controlado. El reto que enfrentan las técnicas de diseño de controladores confiables es el manejo adecuado de las tareas que incluyen operaciones que se realizan concurrentemente, que pueden sincronizarse y que compiten por recursos. Estas técnicas involucran procedimientos sofisticados derivados de métodos formales; su aplicación requiere frecuentemente ayuda computacional, con el fin de evitar errores de diseño aumentando así la confiabilidad de esta etapa.

El problema abordado

La presente tesis se inscribe en este contexto y se enfoca en la etapa de la especificación de los controladores; se propone una metodología que facilita la especificación de entrada a un método de síntesis de controladores lógicos. Esta entrada es expresada mediante redes de Petri (RP) interpretadas [Med98] dadas las ventajas bien conocidas por su capacidad de representar secuencias, paralelismo, manejo de recursos, sincronizaciones, etc [Mel97]. La propuesta consiste en un algoritmo de traducción de redes de Petri a un conjunto de ecuaciones booleanas que son procesadas por un método establecido para la síntesis algebraica de controladores lógicos [HRL08b].

Los trabajos relacionados

Existen trabajos en los que se relacionan las redes de Petri y la creación de controladores. Dimitrova y Batchkova en [DDB07] presentan el uso de RP con señales interpretadas (SIPN)

para la creación de controladores; en este trabajo el supervisor es modelado en una SIPN, su verificación se lleva a cabo utilizando una técnica de verificación de modelos (model checking) y árboles de decisión binaria; para esto, la RP se transforma en una máquina de estados finitos que es procesada por el software NuSMV.

También las redes de Petri son ampliamente utilizadas en verificación. En [CEPA+02] se presenta el uso de una nueva estructura: una extensión de los árboles de decisión binarios(BDD) llamadas Data Decision Diagrams (DDD) los cuales son aplicados en herramientas de verificación de diversas clases de redes de Petri.

En [PRCB94] se relaciona las RP con el álgebra booleana como se describe a continuación. Se modela el comportamiento de la RP segura asignando una función booleana a cada posible evolución del marcado; para esto se define una función característica (characteristic function) que representa una imagen del marcado asociando a cada lugar de la RP un valor 0 ó 1. Para asociar todos los posibles nodos en el grafo de alcanzabilidad utiliza la función de "encoding function" en la que se agrupa el conjunto de funciones características que representa la RP para un marcado inicial en específico. La finalidad de representar de esta manera la RP es construir una imagen computacional para analizar propiedades de seguridad, vivacidad y persistencia. En este enfoque se presenta el problema de la explosión de estados al obtener todos los marcados posibles y procesarlos. En [RCP95] se aplica este trabajo para la verificación de circuitos asíncronos, donde como especificación se tiene una RP; posteriormente es representada como función booleana y verificada usando la técnica symbolic model checking.

Existen trabajos que traducen redes de Petri interpretadas (*RPI*) a lenguaje ensamblador. En [ASPLLS08] se logra simular sistemas físicos. Para esto el sistema es modelado con RP y después traducido para ser implementado en un microcontrolador.

En [JLR01] se propone una técnica para traducir RP no binarias a diagramas de escalera para desarrollar un controlador lógico programable (PLC). Los diagramas generados modelan controladores, incluyendo en su especificación contadores y temporizadores.

En [BLKM06] y [Sot08] se presentan métodos para traducir RP a PLC. Métodos para traducir diagramas de escalera a PLC existen en [BLKM06] y [PZ04]. También en [BLKM06] RP de alto nivel con tiempo y SFC son traducidos en PLC. En[LSE07] se traducen SFC a PLC. En [GHY03] RP con señales son implementadas en PLC.

Enfoque

Este trabajo se centra en la generación de un conjunto de ecuaciones booleanas. La propuesta presentada en [Kam99] traduce una máquina de estados en un conjunto de ecuaciones booleanas que pueden ser implementadas de manera directa en un PLC.

En nuestro enfoque la obtención de ecuaciones booleanas es con el fin de proporcionar una entrada a un método de síntesis de controladores lógicos establecido por Hietter en su tesis doctoral [HRL08b]. Esta entrada constituye una especificación tanto del sistema como de las restricciones que se desea imponer mediante el controlador. En los artículos [HRL08b], [HRL08a],

[HJML08] y [Rou10] se presenta el método de síntesis utilizando diversos formalismos que nos muestran cómo podemos utilizar diversos medios de formalización para armar la especificación completa del sistema.

El trabajo presentado se enfoca en la traducción automatizada de modelos en RP interpretadas al conjunto de ecuaciones booleanas en la forma en que son requeridas por el método de síntesis antes mencionado. Se propone un algoritmo eficiente de traducción que evita la expansión del grafo de alcanzabilidad, recurso utilizado por [PRCB94]. Basado en la propuesta se desarrolla un módulo de software que implementa el algoritmo; el módulo es incorporado a una herramienta de edición y análisis de redes de Petri donde es probado con diversos casos de estudio.

Estructura de la tesis

La tesis está organizada como sigue. En el capítulo 1 se expone el método de síntesis algebraica. En el capítulo 2 se presenta el método de traducción de redes de Petri a sistemas de ecuaciones booleanas. El capítulo 3 describe la herramienta desarrollada que se basa en el método de traducción. El capítulo 4 presenta dos de los casos de estudio con los que se probó el software desarrollado. En las conclusiones se enuncian las principales características de la propuesta y su implementación, así como las posibles extensiones al presente trabajo.

Capítulo 1

Síntesis Algebraica

Resumen

Se presenta el método de síntesis algebraica de Hietter, el cual tiene como entrada especificaciones de diversos formalismos, que después de pasar por un proceso, provee las leyes de control para las incógnitas del sistema, las cuales son utilizadas para el desarrollo de controladores lógicos.

1.1. Introducción

Es importante garantizar que los sistemas cubran con los requerimientos, ya que si alguno no es considerado se corre el riesgo de que se presente un comportamiento riesgoso por parte del sistema, es decir, que el sistema falle o que alguna actividad crítica no sea realizada correctamente. Es por eso que se han propuesto diversos mecanismos y teorías para afrontar este reto, como son la verificación formal, Model-Cheking, Polinomios de Gunnarson, entre otros. Uno de estos enfoques consiste en deducir las leyes de control, este método es descrito en el presente capítulo.
En este capítulo se exponen los conceptos preliminares necesarios para comprender el método de síntesis propuesto por Hietter[Hie09], el cual permite obtener leyes de control a partir de una especificación que ha sido traducida a ecuaciones booleanas.

1.2. Álgebra booleana

Los dispositivos de dos estados se pueden encontrar en sistemas como televisión, radio, etc; que tienen como funciones prender o apagar, donde se puede considerar los estados como "verdadero" o "falso"

En 1854 George Boole publica su trabajo titulado "An Investigation of the laws of thought", en el cual se presenta un sistema de lógica matemática, que en nuestros días se conoce como lógica booleana [Gri98].

El trabajo presentado por C.E. Shannon en 1938 ve como el álgebra booleana se podía usar para analizar circuitos eléctricos [Joh05].

Como consecuencia Shannon desarrolló el álgebra de las funciones de conmutación y mostró la forma en la que su estructura se relacionaba con las ideas establecidas por Boole.

Dentro del álgebra booleana, las funciones booleanas juegan un papel muy importante, estas se definen en la sección 1.3.

1.3. Funciones booleanas

Las funciones booleanas se definen de la siguiente manera [Gri98]:

Definición 1.1. Sea $\mathbb{B}$ ={0,1}.Las operaciones de suma, producto y complemento para los elementos de $\mathbb{B}$, se definen como:

Suma	$0+0 = 0; 0+1 = 1+0 = 1+1 = 1$
Producto	$0\cdot0 = 1\cdot0 = 0\cdot1 = 0 ; 1\cdot1 = 1.$
Complemento.	$\overline{0} = 1; \overline{1} = 0.$

Definición 1.2. Una variable x se dice booleana si x sólo toma valores de $\mathbb{B}$. En consecuencia, $x + x = x$ y $x^2 = x\cdot x = xx = x$ para cualquier variable booleana x.

Si x, y, son variables booleanas, entonces
$x + y = 0$ si y sólo si $x = y = 0$; $xy = 1$ si y sólo si $x = y = 1$.

Si $n \in \mathbb{Z}^+$, entonces $\mathbb{B}^n = \{(b_1, b_2, ..., b_n) | b_i \in \{0, 1\}, 1 \leq i \leq n\}$.

Definición 1.3. Una función $f : \mathbb{B}^n \rightarrow \mathbb{B}$ es una función de conmutación o booleana de n variables. Las n variables se enfatizan si escribimos $f(x_1, x_2, ..., x_n)$, donde cada x_i es una variable booleana con $1 \leq i \leq n$.

Ejemplo 1.4. Sea $f : \mathbb{B}^2 \rightarrow \mathbb{B}$, donde $f(x, y) = x \cdot y$. Por lo tanto, la función estará determinada con las 4 posibles combinaciones que pueden darse al asignarle valores a las variables x y y. Esto se muestra en la tabla 1.1.

Tabla 1.1: Función booleana $f(x, y) = x \cdot y$

x	y	$f(x, y) = x \cdot y$
0	0	0
0	1	0
1	0	0
1	1	1

Definición 1.5. Para $n \in \mathbb{Z}^+$, sean f y $g : \mathbb{B}^n \rightarrow \mathbb{B}$ dos funciones booleanas con sus respectivas n variables booleanas$x_1, x_2, ..., x_n$. f y g serán iguales si para cada una de las 2^n valuaciones de las asignaciones posibles para la función f es la misma que para la función g.

Ejemplo 1.6. Se tienen las funciones $f(x, y) = x \cdot (x + y)$ y $g(x, y) = x \cdot (y + \overline{y})$ son iguales ya que las valuaciones para f son las mismas que para g, esto se comprueba al comparar las valuaciones en las tablas 1.2 y 1.3.

Tabla 1.2: Función booleana $f(x, y) = x \cdot (x + y)$

x	y	$f(x, y) = x \cdot (x + y)$
0	0	0
0	1	0
1	0	1
1	1	1

Hasta ahora se ha definido qué es una función booleana (1.3) y qué son las funciones iguales (1.5), ahora se definirán las operaciones de complemento, producto y suma.

Tabla 1.3: Función booleana $g(x,y) = x\cdot(y+\overline{y})$

x	y	$g(x,y) = x\cdot(y+\overline{y})$
0	0	0
0	1	0
1	0	1
1	1	1

Definición 1.7. Si $f : \mathbb{B}^n \to \mathbb{B}$, entonces el complemento de f, se denota con $\overline{f}$, y la definimos de la siguiente manera :

$$\overline{f}(b_1, b_2, ..., b_n) = \overline{f(b_1, b_2, ..., b_n)}.$$

Definición 1.8. Si $f : \mathbb{B}^n \to \mathbb{B}$, definimos $f+g$, $f\cdot g : \mathbb{B}^n \to \mathbb{B}$ la suma y producto de f, g, respectivamente, como :

$$(f+g)(b_1, b_2, ..., b_n) = f(b_1, b_2, ..., b_n) + g(b_1, b_2, ..., b_n),$$
$$(f\cdot g)(b_1, b_2, ..., b_n) = f(b_1, b_2, ..., b_n)\cdot g(b_1, b_2, ..., b_n).$$

Con las definiciones previas, es posible obtener las leyes que se muestran en la Tabla 1.4

Tabla 1.4: Leyes en funciones Booleanas

1. Ley del doble complemento	$\overline{\overline{f}} = f$
2. Leyes DeMorgan	$\overline{f+g} = \overline{f}\cdot\overline{g}$ y $\overline{f\cdot g} = \overline{f}+\overline{g}$
3. Leyes conmutativas	$f+g = g+f$ y $f\cdot g = g\cdot f$
4. Leyes asociativas	$f+(g+h) = (f+g)+h$ y $f\cdot(g\cdot h) = (f\cdot g)\cdot h$
5. Leyes distributivas	$f+(g\cdot h) = (f+g)\cdot(f+h)$ y $f\cdot(g+h) = (f\cdot g)+(f\cdot h)$
6. Leyes de idempotencia	$f+f = f$ Y $f\cdot f = f$
7. Leyes de identidad	$f+0 = f$ y $f\cdot 1 = f$
8. Leyes de los inversos	$f+\overline{f} = 1$ y $f\cdot\overline{f} = 0$
9. Leyes de dominación	$f+1 = 1$ y $f\cdot 0 = 0$
10. Leyes de absorción	$f+(f\cdot g) = f$ y $f\cdot(f+g) = f$

1.4. Estructura de un álgebra booleana

A continuación se presenta la definición de la estructura de un álgebra booleana. Esta definición es importante ya que muestra la notación que se usa, además de las propiedades que resultan útiles para el metódo de síntesis.

Definición 1.9. Sea $\mathbb{B}$ un conjunto no vacío que contiene dos elementos especiales 0 (el cero o elemento neutro) y 1 (el uno o elemento unidad), sobre el cual se definen las operaciones

Tabla 1.5: Condiciones necesarias en un álgebra booleana

$x+y=y+x$	$x\cdot y=y\cdot x$
$x(y+z)=x\cdot y+x\cdot z$	$x+(y\cdot+z)=(x+y)\cdot(x+z)$
$x+0=x$	$x\cdot 0=0$
$x+\overline{x}=1$	$x\cdot 1=x1=x$
$0\neq 1$	$x\cdot\overline{x}=x\overline{x}=0$

binarias cerradas "+", "·" y una operación unaria "$\overline{\ }$". Entonces $(\mathbb{B},+,\ \overline{\ },0,1)$ es un álgebra booleana si se cumplen las condiciones de la tabla 1.5 para todos $x,y,z\in\mathbb{B}$.

Algunos ejemplos de álgebras booleanas son:

- Teoría de conjuntos
- La lógica proposicional
- El conjunto de las funciones booleanas

Las condiciones de la definición 1.9 dan pie a más propiedades que se muestran en la siguiente sección.

1.5. Propiedades de una Álgebra booleana

Definición 1.10. Sea $\mathbb{B}$ una álgebra booleana y $x\in\mathbb{B}$, la Ley de idempotencia nos da las siguientes propiedades:

$x+x=x$

$x\cdot x=x$

Se da el principio de dualidad, a partir de la definición anterior se tiene que si en las expresiones cambiamos los símbolos + por · también se cumplen las leyes.

En la Tabla 1.6 se muestran las leyes que se obtienen de las álgebras booleanas.

En las secciones 1.6 y 1.7 se describen las relaciones de equivalencia y orden parcial, en las cuales se expresan las especificaciones de los sistemas lógicos que se ocupan en la fase de formalización en el método de síntesis.

1.6. Relación de equivalencia

Si R es una relación de X en Y, se define

$$R(x)=\{y\in Y : xRy\} \tag{1.1}$$

Tabla 1.6: Leyes en un álgebra booleana

1. Ley de dominancia	$x + 1 = 1$ $x \cdot 0 = 0$
2. Leyes absorción	$x \cdot (x + y) = x$ $x + x \cdot y = x$
3. Leyes de cancelación	$[xy = xz \text{ y } \overline{x}y = \overline{x}z] \Rightarrow y = z$ $\overline{x} + y = \overline{x} + z] \Rightarrow y = z$
4. Leyes asociativas	$x(yz) = (xy)z$ $x + (y + z) = (x + y) + z$
5. Unicidad de los inversos	$[x + y = 1 \text{ y } xy = 0] \Rightarrow y = \overline{x}$
6. Ley del doble complemento	$\overline{\overline{x}} = x$
7. Ley de DeMorgan	$\overline{xy} = \overline{x} + \overline{y}$ $\overline{x + y} = \overline{x} \cdot \overline{y}$
	$\overline{0} = 1$ $\overline{1} = 0$
	$x \cdot \overline{y} = 0$ si y sólo si $xy = x$ $x + \overline{y} = 1$ si y sólo si $x + y = x$

Sea X una colección de objetos.Una relación R en X es una relación de equivalencia si es reflexiva, simétrica y transitiva [Res98]. A continuación se da un listado de equivalencias para $x = y$:

$x = y$ $\qquad$ $x \cdot y + \overline{x} \cdot \overline{y} = 1$

$\overline{x} = \overline{y}$ $\qquad$ $x \cdot \overline{y} = 0$

$x \cdot \overline{y} + \overline{x} \cdot y = 0$ $\qquad$ $\overline{x} \cdot y = 0$

1.7. Relación de orden parcial

Sea X una colección de objetos. Una relación R en X es una relación de orden parcial (ó simplemente relación de orden) si es reflexiva, antisimétrica y transitiva [Res98]. Si R es una relación de orden parcial en X y xRy y con $x \neq y$, se dira que " x precede a y ", "x es un antecesor de y" o "y es un sucesor de x". Esta relación se expresa con el signo $\leq$.

Al trasladar este concepto al álgebra booleana se tiene la definición 1.11.

Definición 1.11. Si $x, y \in \mathbb{B}$, definimos $x \leq y$ si $xy = x$. Esta relación será de orden parcial y además tendremos que en un álgebra booleana, $0 \leq x$ y $x \leq 1$[Gri98]. Con esta relación tenemos las siguientes propiedades en el teorema 6 por Hietter[Hie09]:
Listado de equivalencias para $x \leq y$:

$x \leq y$	$x\overline{y} = 0$	$x + y = x$
$xy = x$	$\overline{y} \leq \overline{x}$	$\overline{x} + y = 1$
$\overline{x} + \overline{y} = \overline{x}$	$\overline{x} \cdot \overline{y} = \overline{y}$	

1.8. Método de síntesis algebraica

Los controladores son necesarios en sistemas embebidos, de transportes, de producción, plantas de energía, etc. El reto de estos es lograr la confiabilidad de controles lógicos.

El método de síntesis propuesto por Hietter en su tesis doctoral[Hie09] y en [HRL08a], aborda el diseño de controladores lógicos. También presenta el método de síntesis utilizando una solución parcial basada en modelos de estados [HRL08b] y redes de Petri [HJML08].

Definición 1.12. Confiabilidad (o del inglés, dependability).El comportamiento del sistema debe cumplir todos los requerimientos y estar libre de errores, además de que no debe presentar comportamientos riesgosos cuando el sistema esté en ejecución.

Este trabajo se limita a los sistemas binarios, como los que corresponden al álgebra booleana, cuyo comportamiento puede representarse con señales de dos estados, por ejemplo: encendido y apagado. Lograr que estos sistemas cumplan con el concepto de *confiabilidad* ha sido abordado por diversos métodos formales, entre ellos Model -checking, prueba de teoremas, teoría del control, métodos de síntesis.

El método de Hietter que pertenece a los métodos de síntesis consiste en cuatro pasos o fases: la formalización, la prueba de coherencia, la solución del sistema de ecuaciones y finalmente la selección de una solución. Estas fases se muestran en la Fig. 1.1. En la fase de formalización se tiene como entrada el conjunto de especificaciones expresadas en uno o varios formalismos como las redes de Petri, autómatas, etc. el cual es transformado en un sistema de ecuaciones booleanas, siendo ésta la salida de esta etapa. En la siguiente fase se hace un análisis de la salida para comprobar que exista al menos una solución; si no hay solución debemos regresar a la etapa anterior a resolver los conflictos generados por nuestras especificaciones; si por el contrario, existe una solución, se pasa a la tercera fase: encontrar las soluciones. En esta parte se obtiene la solución de manera paramétrica que engloba el conjunto de soluciones que satisfacen la ecuación, permitiendo el inicio de la cuarta fase donde se selecciona una de estas soluciones, concluyendo el proceso.

En las siguentes subsecciones se describe las cuatro fases del método, además se presenta un ejemplo.

1.8.1. Formalización

En esta fase se usan las relaciones de equivalencia 1.6 y de inclusión o de orden parcial 1.7 para definir las especificaciones del sistema y lograr crear el sistema de ecuaciones booleanas. Podemos obtener las especificaciones del lenguaje natural, de diagramas, de modelos parciales, etc.

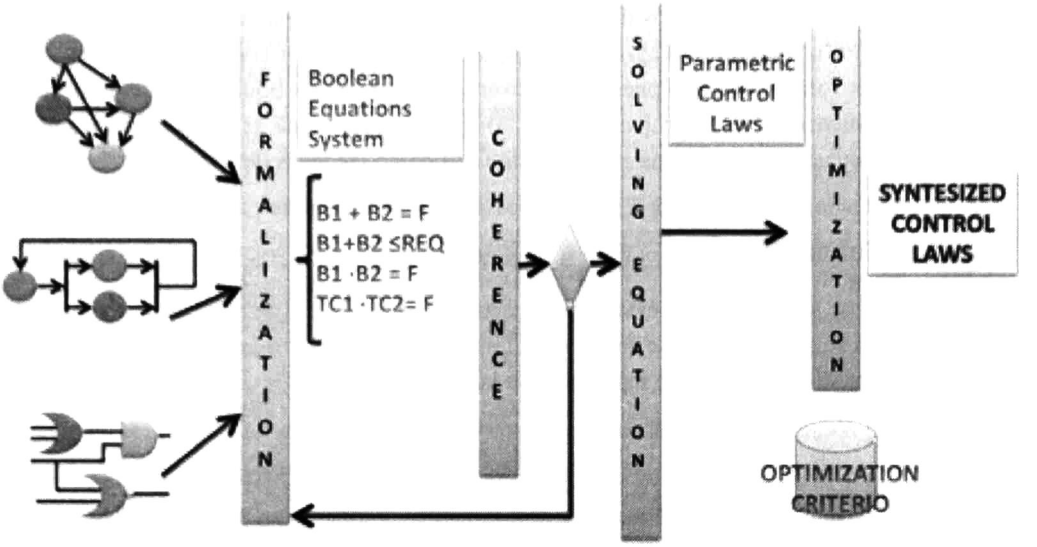


Figura 1.1: Fases del proceso de síntesis algebraica [HRL08a]

En la Tabla 1.7 se muestra cómo se puede expresar en álgebra booleana especificaciones en Lenguaje Natural, abarcando condiciones de suficiencia, necesidad, acciones que no pueden pasar y acciones que pasan siempre.

Tabla 1.7: Ejemplos de formalización del lenguaje natural.

Tipo	Estructura de la frase en lenguaje natural	Formalización
Condición de suficiencia	Si A es cierto, B es también cierto.	$A \leq B$
	Cuando A esta presente, B esta presente.	$A \leq B$
	Es suficiente tener A para tener B.	$A \leq B$
Condición de necesidad	Es necesario tener A para tener B.	$B \leq A$
	Es obligatorio tener A para tener B.	$B \leq A$
	A tiene obligado tener B.	$B \leq A$
Situación prohibitoria	Está prohibido tener A y B al mismo tiempo.	$A \cdot B = 0$
	·A y B nunca puede estar al mismo tiempo activas.	$A \cdot B = 0$
Situación invariante	Siempre tenemos A o B.	$A + B = 1$
Situación excluyente	Siempre esta activa A o B, pero no al mismo tiempo.	$A \cdot \overline{B} + \overline{A} \cdot B = 1$

En el capítulo 2 se muestra cómo usar en la fase de formalización las redes de Petri, para poderlas utilizar como especificaciones que podamos usar en el proceso de síntesis algebraica.

También se puede usar otros diagramas, como los requerimientos por funciones state function chart (SFC), autómatas, diagramas de escalera, diagramas por compuertas, etc.

Cualquier formalismo que pueda ser convertido a ecuaciones booleanas puede ser utilizado como especificación, solución parcial, etc. Al final de esta etapa se tiene como resultado un sistema de ecuaciones booleanas cuya solución nos darán las leyes de control.

1.8.2. Coherencia

En esta fase se verifica que el sistema de ecuaciones resultante de la fase anterior 1.8.1 tenga al menos una solución. Esta fase se basa en el marco teórico desarrollado por Hietter [HRL08b].

El sistema de ecuaciones se representa de la siguiente forma

$$R(Y_1, ..., Yn, B_1, ..., B_m) = F \tag{1.2}$$

En ella se representan los requerimientos como una composición de las funciones booleanas B_i (conocidas) y Y_j (incógnitas).

Es necesario distinguir entre las funciones B_i y Y_i, puesto que la función utiliza las incógnitas para efectuar las diversas combinaciones.

Sea $f(x_1, x_2, ..., x_n)$ una función lógica arbitraria que se expande de acuerdo a la expansión de Shannon [Sas99] :

$$f(x_1, x_2, ..., x_n) = \overline{x_1} f(0, x_2, ..., x_n) + x_1 f(1, x_2, x_3, ..., x_n). \tag{1.3}$$

Siguiendo la ecuación (1.2) con $n = 1$, aplicando la descomposición de Shannon (1.3) se obtiene la siguiente ecuación.

$$R(Y_1) = E_1 \cdot \overline{Y_1} + E_2 \cdot Y_1. \tag{1.4}$$

donde E_1 y E_2 son funciones booleanas de $\mathbb{B}$.

Al sustituir en la ecuación (1.4) los valores que nos da el asignarles T y F a la variable Y_1 en R resulta:

$$R(F) \cdot \overline{Y} + R(T) \cdot Y = F. \tag{1.5}$$

Al generalizar la ecuación (1.5) para múltiples variables desconocidas, tenemos que para $R(Y_1, .., Yp)$ se obtiene la siguiente ecuación:

$$R(Y_1, .., Y_P) = \sum_{\omega \in \Omega_p} (R(w) \cdot \prod_{i=1}^{p} (w_i \cdot Y_i + \overline{w_i} \cdot \overline{Y_i})) \tag{1.6}$$

donde Ω_p es el conjunto de 2^p tuplas cuyos componentes son T y F; w es un elemento de este conjunto donde w_i es el i^{th} elemento en la relación R.

Teorema 1.13. *Sea la ecuación $R(Y) = F$ expresada en un álgebra booleana $(\mathbb{B}, +, \cdot, \bar{}, 0, 1)$, donde $R(Y)$ es la expresión general para una sola variable desconocida Y. $R(Y) = F$ tiene solución si y sólo si [HRL08b]:*

$$R(F) \cdot R(T) = F. \tag{1.7}$$

Generalizando para múltiples variables el Teorema 1.13 se tiene el siguiente teorema:

Teorema 1.14. *$R(Y_1, Y_2, ..., Y_p) = F$ tiene solución si y sólo si:*

$$\prod_{\omega \in \Omega_p} R(w) = F. \tag{1.8}$$

Con el Teorema 1.14 se comprueba la existencia de al menos una solución para el sistema de ecuaciones que representa $R(Y_1, Y_2, ..., Y_p)$.

Ejemplo 1.15. Para un caso con 3 variables desconocidas $R(Y_1, Y_2, Y_3)$:

$$\begin{aligned}\prod_{\omega \in \Omega_p} R(w) =& R(F,F,F) + R(F,F,T) + R(F,T,F) + \\ & R(F,T,T) + R(T,F,F) + R(T,F,T) + R(T,T,F) + R(T,T,T).\end{aligned} \tag{1.9}$$

1.8.3. Solución del sistema de ecuaciones

Como se ve en la Fig. 1.1, si el sistema tiene solución entonces sigue la fase de búsqueda del conjunto de soluciones [HRL08a]:

Para el caso de una variable la siguiente ecuación expresa la solución:

$$Y = R(F) + \overline{R(T)} \cdot P \tag{1.10}$$

donde Y es la variable desconocida y P es una constante que pertenece a $\mathbb{B}$, la cual representa el conjunto de soluciones.

Para el caso de n variables desconocidas se encuentran las soluciones con las siguientes ecuaciones [HRL08a]:

$$\begin{aligned} Y_1 &= \textstyle\prod_{\omega \in \Omega_{p-1}} R(F, \omega_1, ..., \omega_{p-1}) + \overline{\textstyle\prod_{\omega \in \Omega_{p-1}} R(T, \omega_1, ..., \omega_{p-1})} \cdot P_1, \\ &\vdots \\ Y_i &= \textstyle\prod_{\omega \in \Omega_{p-i}} R(Y_1, ..., Y_{i-1}; F, \omega_1, ..., \omega_{p-i}) + \overline{\textstyle\prod_{\omega \in \Omega_{p-i}} R(Y_1, ..., Y_{i-1}, T, \omega_1, ..., \omega_{p-i})} \cdot P_i, \\ &\vdots \\ Y_p &= R(Y_1, ..., Y_{p-1}, F) + \overline{R(Y_1, ..., Y_{p-1}, T)} \cdot P_p. \end{aligned} \tag{1.11}$$

Con estas ecuaciones de manera incremental es posible resolver cada una de las variables desconocidas.

1.8.4. Escoger la solución

Como se aprecia en la fórmula 1.11, existen los parámetros P_i, para los cuales podremos escoger valores en el conjunto de las 2^{2^n} variables para el conjunto.

En esta fase se escoge un valor específico para cada parámetro, de manera que el diseñador tiene la posibilidad de completar las especificaciones dadas.

1.8.5. Ejemplo de síntesis algebraica

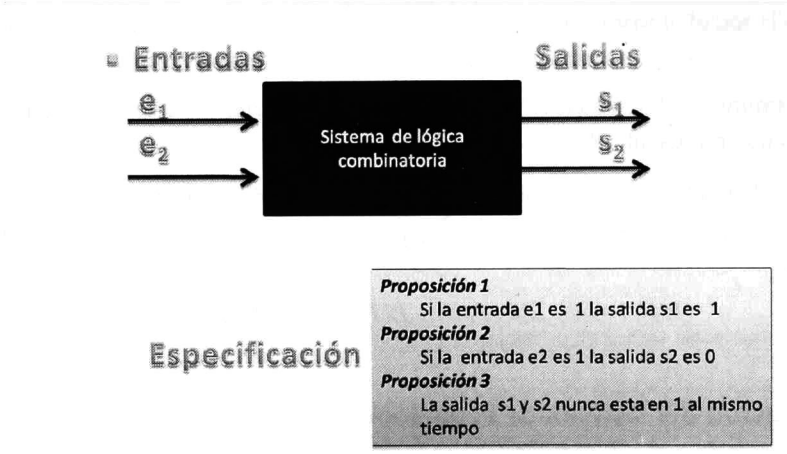


Figura 1.2: Sistema combinatorio [HRL08a]

A continuación se desarrolla el ejemplo 1.16 siguiendo el método completo de síntesis algebraica. Se apreciará en las subsecciones siguientes las operaciones matemáticas desarrolladas en las cuatro fases.

Ejemplo 1.16. En la Fig. 1.2 [Hie09]se muestra un sistema combinatorio, donde las entradas conocidas son e_1 y e_2 y las variables s_1 y s_2 son desconocidas. Es para dichas varaibles que se buscan las leyes de control correspondientes para que cubran con los requerimientos dados en la especificación.

Fase 1. Formalización

En esta fase se recopilan todos los requerimientos presentes en los diversos formalismos y se transforman a ecuaciones booleanas, para utilizar las leyes expuestas en la Tabla 1.6 y los listados de propiedades expuestos en las secciones 1.6 y 1.7. Así se logra traducir los requerimientos y crear un sistema de ecuaciones. En la Fig. 1.3 se muestran las especificaciones del ejemplo 1.16 las cuales del lenguaje natural son expresadas como relaciones de equivalencia y

Proposición	Descripción	Formalización
1	Si la entrada e_1 es 1 la salida s_1 es 1	$E_1 \leq S_1$
2	Si la entrada e_2 es 1 la salida s_2 es 0	$E_2 \leq \overline{S_2}$
3	La salida s_1 y s_2 nunca esta en 1 al mismo tiempo	$S_1 \cdot S_2 = F$

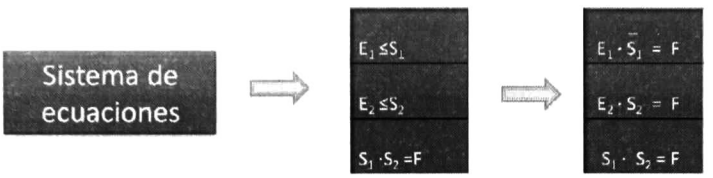


Figura 1.3: Fase 1 del método de síntesis

de orden parcial que después son transformadas en equivalencias a falso, es decir, se utiliza la equivalencia $x \cdot \overline{y} + \overline{x} \cdot y = 0$ entre otras leyes del listado 1.6 .

El resultado de esta fase se observa en la Fig. 1.3. Trabajando el sistema de ecuaciones, la función booleana expresada en una única función es :

$$E_1 \cdot \overline{S_1} + E_2 \cdot S_2 + S_1 \cdot S_2 = F \tag{1.12}$$

que es el resultado de unir con el operador "+" las tres ecuaciones que fueron obtenidas. Con este resultado se trabaja en las siguientes fases del método.

Fase 2. Coherencia

La relación que representa el Ejemplo 1.16 es representado siguiendo la ecuación (1.2)

$$R(S_1, S_2, E_1, E_2) = F \tag{1.13}$$

donde E_1y E_2 son las funciones booleanas conocidas, y S_1 y S_2 las funciones que el método encuentra.

Ahora, viendo la relación $R(S_1, S_2)$, se obtiene lo siguiente usando la ecuación (1.6)

$$\begin{aligned} R(Y_1, .., Y_P) = \sum_{\omega \in \Omega_p} (R(w) \cdot \prod_{i=1}^{p} (w_i \cdot Y_i + \overline{w_i} \cdot \overline{Y_i})) = \\ R(T, T, E_1, E_2) \cdot (T \cdot S_1 + F \cdot \overline{S_1}) \cdot (T \cdot S_2 + F \cdot \overline{S_2}) + \\ R(F, T, E_1, E_2) \cdot (F \cdot S_1 + T \cdot \overline{S_1}) \cdot (T \cdot S_2 + F \cdot \overline{S_2}) + \\ R(T, F, E_1, E_2) \cdot (T \cdot S_1 + F \cdot \overline{S_1}) \cdot (F \cdot S_2 + T \cdot \overline{S_2}) + \\ R(F, F, E_1, E_2) \cdot (F \cdot S_1 + T \cdot \overline{S_1}) \cdot (F \cdot S_2 + T \cdot \overline{S_2}) \end{aligned}$$

$$= R(T,T,E_1,E_2)\cdot(S_1+F)\cdot(S_2+F) + R(F,T,E_1,E_2)\cdot(F+\overline{S_1})\cdot(S_2+F)$$
$$R(T,F,E_1,E_2)\cdot(S_1+F)\cdot(F+\overline{S_2}) + R(F,F,E_1,E_2)\cdot(F+\overline{S_1})\cdot(F+\overline{S_2})$$

$$\begin{aligned} =& R(T,T,E_1,E_2)S_1\cdot S_2 + R(F,T,E_1,E_2)\cdot\overline{S_1}\cdot S_2 + \\ & R(T,F,E_1,E_2)\cdot S_1\cdot\overline{S_2} + R(F,F,E_1,E_2)\cdot\overline{S_1}\cdot\overline{S_2} \end{aligned} \tag{1.14}$$

Con esta generalización en la ecuación (1.14), se obtiene la base para encontrar la solución. Ahora se sustituyen los valores que puede tener la función para el ejemplo considerando el espacio de Ω_2 :

$\Omega_2 = (T,T),(T,F),(F,T),(F,F)$

$$R(T,T,E_1,E_2) = E_1\cdot\overline{T} + E_2\cdot T + T\cdot T = E_1 + E_2 + T = T, \tag{1.15}$$
$$R(T,F,E_1,E_2) = E_1\cdot\overline{T} + E_2\cdot F + T\cdot F = F + F + F = F, \tag{1.16}$$
$$R(F,T,E_1,E_2) = E_1\cdot\overline{F} + E_2\cdot T + F\cdot T = E_1 + E_2 + F = E_1 + E_2, \tag{1.17}$$
$$R(F,F,E_1,E_2) = E_1\cdot\overline{F} + E_2\cdot F + F\cdot F = E_1 + F + F = E_1. \tag{1.18}$$

Siguiendo la ecuación (1.8), y sustituyendo los valores en ésta previamente obtenidos en las ecuaciones (1.15), (1.16), (1.17) y (1.18) aplicamos el Teorema 1.14 para determinar si tiene solución :

$$\begin{aligned} \prod_{\omega\in\Omega_2} R(w) &= R(T,T,E_1,E_2)\cdot R(T,F,E_1,E_2)\cdot R(F,T,E_1,E_2)\cdot R(F,F,E_1,E_2) \\ &= T\cdot F\cdot(E_1+E_2)\cdot E_1 = F \end{aligned} \tag{1.19}$$

Como se muestra en la ecuación (1.19) se cumple con la ecuación, lo que indica que existe al menos una solución.

Fase 3. Solución

Las soluciones se obtienen a partir de las ecuaciones mostradas en (1.11).

Para obtener la solución de la variable desconocida S_1:

Con $\Omega_1 = \{T,F\}$ y $p = 2$

$$\begin{aligned} S_1 &= \prod_{\omega\in\Omega_{2-1}} R(F,\omega_1,E_1,E_2) + \overline{\prod_{\omega\in\Omega_{2-1}} R(T,\omega_1,E_1,E_2)}\cdot P_1 \\ &= R(F,T,E_1,E_2)\cdot R(F,F,E_1,E_2) + \overline{R(T,T,E_1,E_2)\cdot R(T,F,E_1,E_2)} = E_1 + P_1 \end{aligned} \tag{1.20}$$

$$\begin{aligned} S_2 &= R(S_1,F,E_1,E_2) + \overline{R(S_1,T,E_1,E_2)}\cdot P_2 \\ &= R(E_1+P_1,F,E_1,E_2) + \overline{R(E_1+P_1,T,E_1,E_2)}\cdot P_2 = \overline{E_1}\cdot\overline{E_2}\cdot\overline{P_1}\cdot P_2 \end{aligned} \tag{1.21}$$

Los resultados que se obtienen en las ecuaciones (1.20) y (1.21), para S_1 y S_2 respectivamente, son las funciones paramétricas con las que se cuenta para escoger la solución adecuada en la siguiente fase.

Fase 4. Optimización

Con la solución de forma paramétrica de S_1 y S_2 se asignan los valores a las variables P_i que se encontraron en las soluciones para así poder obtener la solución. En esta parte el diseñador decide cuáles son los criterios que utilizará para escoger la solución, ya sea alguna heurística, o asigna alguna prioridad a algún estado.

1.9. Conclusiones

En este capítulo se presenta el método de síntesis propuesto por Hietter, empezando por la base algebraica y finalizando con un ejemplo. Es importante conocer las capacidades de expresión del álgebra booleana, para así poder aplicarla adecuadamente al momento de hacer la especificación del sistema, así como su traducción.

Capítulo 2

Traducción de redes de Petri a sistemas de ecuaciones booleanas

---

Resumen

En este capítulo se presenta el algoritmo para la obtención del sistema de ecuaciones booleanas que representa el comportamiento de la red de Petri Interpretada. Además se incluye un caso de estudio donde se aplica el algoritmo.

2.1. Introducción

En la actualidad las redes de Petri son una herramienta poderosa para el modelado de sistemas, ya que poseen ciertas características que le dan ventajas con respecto a otros formalismos. Desde su creación en 1962, han sido utilizadas en varias áreas del conocimiento.

Este capítulo aborda el problema de la **traducción de redes de Petri a sistemas de ecuaciones booleanas**, para ser utilizadas como especificaciones de entrada en un método de síntesis algebraica.

2.2. Redes de Petri

En esta sección se presentan las definiciones formales de redes de Petri ordinarias y la extensión que se usará en esta tesis: las redes de Petri interpretadas (RPI). La notación utilizada es tomada de [TRM03].

Definición 2.1. Una **estructura de Red de Petri** (RP) es un grafo bipartita representado por una cuádrupla $G = (P, T, I, O)$ donde $P = \{p_1, p_2, ..., p_n\}$ es un número finito de vértices llamados lugares, $T = \{t_1, t_2, ..., t_n\}$ es un conjunto finito de vértices llamados transiciones, $I : PxT \rightarrow \{0, 1\}$ es una función que representa el peso de los arcos que van de los lugares a las transiciones y $O : PxT \rightarrow \{0, 1\}$ es una función que representa el peso de los arcos que van de las transiciones a los lugares.

La matriz de incidencia de G es $C = [c_{ij}]$, donde $c_{ij} = O(p_i, t_j) - I(p_i, t_j)$. La función de marcado $M : P \rightarrow \mathbb{Z}^+$ es un mapeo de cada lugar a un entero positivo que representa el número de marcas que tiene un lugar.

Definición 2.2. Una **Red Petri** es el par $N = (G, M_0)$, donde G es una estructura de Red de Petri y M_0 es un un marcado inicial.

Con la definición 2.2 es posible representar el sistema en algún nivel de abstracción. Para las reglas de evolución se definen: la regla de disparo y la ecuación de estado.

Para la **regla de disparo**, sea t_j una transición que estará habilitada en el marcado M_k si y sólo si $\forall p_i \in P, M_k(p_i) \geq I(p_i, t_j)$. Si lo anterior se cumple es posible proceder a disparar la transición. El nuevo marcado M_{k+1} resultante es dado por la **ecuación de estado** (2.1)

$$M_{k+1} = M_k + Cv_k \tag{2.1}$$

donde $v_k(i) = 0, i \neq j, v_k(j) = 1$. Además el conjunto de alcanzabilidad de la red de Petri que contiene todos los posibles marcados partiendo del estado inicial M_0, se escribe como $R(G, M_0)$.

Definición 2.3. Una **Red Petri Interpretada** (RPI) es la séxtupla $Q = (N, \sum, \Phi, \lambda, D, \varphi)$, donde:

$N = (G, M_0)$ es una RP;
$\sum = \{\sigma_1, \sigma_2, ..., \sigma_r\}$ es el alfabeto de entrada de la red, donde σ_i es un símbolo de entrada.

$\Phi = \{\phi_1, \phi_2, ..., \phi_s\}$ es el alfabeto de salida asociado a los lugares, donde ϕ_i, es un símbolo de salida;
$\lambda : T \rightarrow \sum \bigcup \{\varepsilon\}$ es la función de etiquetado a las transisiones con las siguientes restricciones: $\forall t_j, t_k \in T$, $j \neq k$ si $I(p_i, t_j) = I(p_i, t_k) \neq 0$ y ambos $\lambda(t_j), \lambda(t_k) \neq \varepsilon$ entonces $\lambda(t_j) \neq \lambda(t_k)$. En este caso, ε representa un evento interno.

$D : T \rightarrow \varsigma$ es una función "feed-foward", donde $\varsigma = \tau U\{\varepsilon\}$ y $\tau \subseteq T$. Cuando una transición t_j es disparada entonces su nombre t_j o el símbolo ε son dados como una salida de la red de Petri interpretada. En este caso, ε representa una señal de salida nula.

$\varphi : R(G, M_0) \rightarrow \{\phi\}^q$ es una función de salida, donde $R(G, M_0)$ es el conjunto de alcanzabilidad definido para la Red de Petri y q es el número de salidas disponibles asociadas a los lugares.

Se usa la definición expuesta en [TSB02] para φ:

Donde la función $\varphi : [\mathbb{Z}^+]^n \rightarrow [\mathbb{Z}^+]^q$ puede ser representada como una matriz $\varphi = [\varphi_{ij}]$ de dimensiones $q \times n$, donde el i-ésimo vector fila φ_i de φ es la transpuesta del vector elemental e_j si el marcado de j-ésimo lugar es disponible a la salida.

Una limitante importante es utilizar redes de Petri seguras. Esto porque se manipulan sistemas booleanos y existe una relación en cuanto a los estados que manejan ambas. Existen trabajos relacionados que hablan de la traducción de redes de Petri seguras a redes acotadas como en [BW00]. A continuación se presentan definiciones importantes para trabajar con redes de Petri seguras:

Definición 2.4. Sea $R(G, M_0)$ el conjunto de marcados alcanzables desde el marcado m. Una red de Petri $N = (G, M_0)$ es acotada si el conjunto $R(G, M_0)$ es finito.[PRCB94]

Definición 2.5. Una red de Petri $N = (G, M_0)$ es **k-acotada** si para cada $m \in R(G, M_0)$ y para cada lugar $p \in P$, $m(p) \leq k$.

Definición 2.6. Una red de Petri es **segura** (del inglés, *safe*) si es 1-acotada.

En la siguiente sección se presenta el proceso de traducción de redes de Petri interpretadas binarias a sistemas de ecuaciones booleanas.

2.3. Planteamiento del problema

Se tiene un conjunto de especificaciones con los cuales se pretende representar tanto el sistema como el comportamiento que se espera. Existen diversos formalismos para representar el sistema. Este trabajo se enfoca a las RP, así que el objetivo, como se muestra en la Fig.2.1 será convertir una RP a un conjunto de expresiones booleanas que representen el mismo comportamiento especificado. Considerando que la tendencia en los sistemas es crecer y ser cada

vez más complejos, es importante enfrentar este problema, de modo que automatizar el proceso es una ventaja importante.

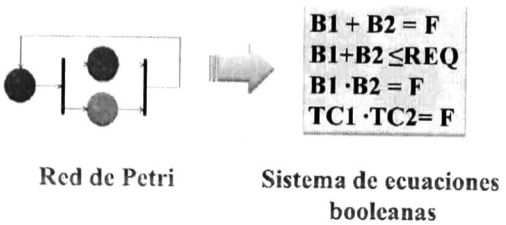


Figura 2.1: Planteamiento del problema

Existen en la literatura dos vertientes que nos permiten obtener la traducción que deseamos realizar: la primera consiste en obtener el grafo de alcanzabilidad [PRCB94], mientras que la segunda se basa en las propiedades intrínsecas de la RP (estructura, salidas, conflictos y marcado inicial)[HJML08].

El trabajo de esta tesis está basado en el segundo enfoque En las siguiente sección explicaremos a detalle cómo se lleva à cabo la traducción a partir de las propiedades de la RP.

2.3.1. Estructura de la red de Petri

Para representar la estructura de la red en ecuaciones booleanas se considera por separado cada lugar que la conforma. De esta manera cada lugar generará una ecuación; el conjunto de estas ecuaciones es la traducción de la estructura de la RP.

Para efecto de traducir càda lugar se deben considerar los dos siguientes casos. El primero supone que el marcado de la RP en el instante de tiempo anterior habilita alguna de las transiciones del Pre de dicho lugar, además de que dicha transición es disparada. En el segundo caso se asume que el lugar a analizar se encuentra marcado en el instante de tiempo anterior y que ninguna transición del Post de dicho lugar se ha disparado.

Considerando los casos expuestos anteriormente, para obtener la ecuación del lugar p_i del fragmento de la red de Petri de la Fig. 2.2 en el instante de tiempo k se considera:

- En el instante de tiempo $(k-1)$ está marcado el lugar p_j y se dispara la transición t_a o bien marcados los lugares p_k y p_l y se dispara la transición t_b, es decir, $p_j(k-1)\cdot t_a + p_k(k-1)\cdot p_l(k-1)\cdot t_b$.

- En el instante de tiempo $(k-1)$ se encuentra marcado el lugar p_i y ninguna de las transiciones t_c, t_d, t_e ni t_f es disparada, es decir, $\overline{t_c}\cdot\overline{t_d}\cdot\overline{t_e}\cdot\overline{t_f}\cdot p_i(k-1)$.

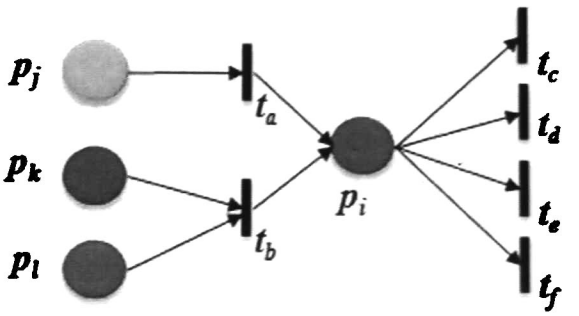


Figura 2.2: Fragmento de una PN

Integrando la representación de los dos comportamientos se tiene:

$$p_i(k) = p_j(k-1)\cdot t_a + p_k(k-1)\cdot p_l(k-1)\cdot t_b + \overline{t_c}\cdot\overline{t_d}\cdot\overline{t_e}\cdot\overline{t_f}\cdot p_i(k-1). \tag{2.2}$$

Con este razonamiento se puede enunciar una forma general de las ecuaciones que representan el comportamiento expresado por la estructura de la RP.

Sea $p_i \in P$ un lugar de la RP para $i = 0, ..., |P|-1$, $\{t_{i_0}, ..., t_{i_n}\} \in \bullet p_i$ el Pre del lugar p_i donde $n = |\bullet p_i| - 1$, $\{t'_{i_0}, ..., t'_{i_m}\} \in p_i\bullet$ el Post del lugar p_i donde $m = |p_i \bullet| - 1$ y $\{p_{s_0}, ..., p_{s_y}\} \in \bullet t_{i_s}$ es el Pre de t_{i_s} donde $y = |\bullet t_{i_s}| - 1$ para todo $t_{i_s} \in \bullet p_i$. Así se obtiene:

$$p_i = \sum_{s=0}^{n}((\prod_{r=0}^{y} p_{s_r}(k-1))\cdot t_{i_s}) + ((\prod_{j=0}^{m}\overline{t'_{i_j}})\cdot p_i(k-1). \tag{2.3}$$

De esta manera la RP queda representada por el sistema de $|P|$ ecuaciones expresadas en 2.3.

2.3.2. Salidas del sistema

Una de las características que posee una las RP interpretadas es que se le puede asociar a cada lugar un símbolo del alfabeto de salida Φ. Por lo tanto su representación en ecuaciones booleanas resulta intuitiva. Tomando como ejemplo la Fig. 2.3, se observa que existen tres lugares p_i, p_j y p_k. Las salidas ϕ_1 y ϕ_2 son asociadas a los lugares de la RP, obteniendo las siguientes las ecuaciones resultantes:

- $p_j = \phi_1$
- $p_k = \phi_2$

Generalizando el conjunto de ecuaciones se puede obtener a partir de cada uno de los elementos de la matriz φ, esto es, si $[\varphi_{ij}] = 1$ entonces se asocia el elemento del alfabeto de salida ϕ_i al lugar p_j, en forma matemática se tiene:

$$\text{Si } [\varphi_{ij}] = 1 \text{ entonces } p_j = \phi_i. \tag{2.4}$$

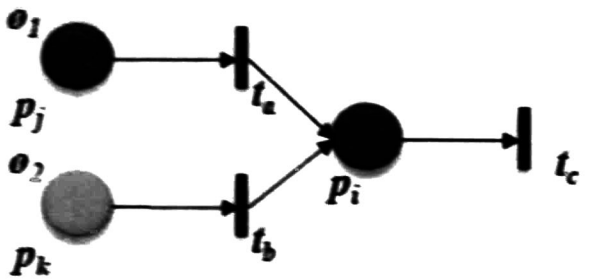


Figura 2.3: Fragmento de una PN

2.3.3. Marcado Inicial

Además de la estructura, es importante considerar el marcado inicial de una RP, ya que puede afectar varias de sus propiedades tales como vivacidad, acotamiento, etc. Para su representación se le puede considerar como la inicialización de las variables introducidas en la ecuación 2.3. Dado que se manejan RP seguras, entonces a cada uno de los lugares se les asocian un valor de 1 ó 0 (verdadero o falso). Si se considera la Fig. 2.4 se observa que los lugares p_i y p_k están marcados, por lo tanto, se les asigna un valor verdadero mientras que a p_j y p_l al no estar marcados se les asignará un valor de falso. De esta manera las ecuaciones resultantes para este ejemplo son:

$p_i(0) = 1$ $p_j(0) = 0$ $p_k(0) = 1$ $p_l(0) = 0$

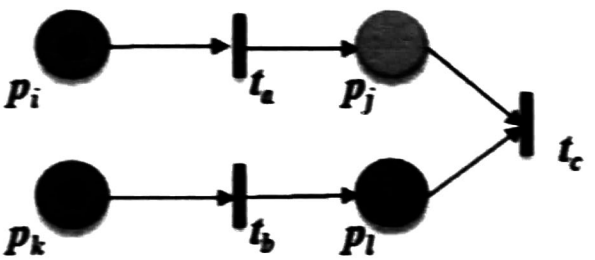


Figura 2.4: Fragmento de una PN

Así para cualquier RP con un marcado inicial M_0 las ecuaciones resultantes serán para cada $p_i \in P$:

$$p_i(0) = \begin{cases} 0 & \text{si } M_0(p_i) = 0 \\ 1 & \text{si } M_0(p_i) = 1. \end{cases} \tag{2.5}$$

En adelante, $p_i(0)$ será denotada como p_i, de esta forma para el ejemplo anterior las ecuaciones resultantes seran:

$p_i = 1$ $p_j = 0$ $p_k = 1$ $p_l = 0$

2.3.4. Conflictos

Al trabajar con RP seguras, se debe garantizar que si la marca de un lugar habilita dos o más transiciones entonces solo debe dispararse una de ellas; por ejemplo en la Fig. 2.5 para el lugar p_i con las transiciones t_a, t_b, t_c y t_d que pertenecen al conjunto $p_i\bullet$, se debe asegurar que ninguna de las posibles combinaciones entre éstas se disparen simultáneamente a fin de evitar un comportamiento no deseado.

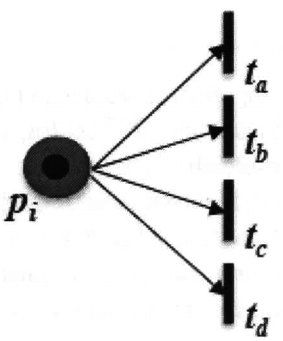


Figura 2.5: Fragmento de una PN

Para lograr lo anterior este comportamiento basta especificar la exclusión del disparo por parejas de transiciones, es decir:

$t_a \cdot t_b = 0$ $t_b \cdot t_c = 0$ $t_c \cdot t_d = 0$
$t_a \cdot t_c = 0$ $t_b \cdot t_d = 0$ $t_a \cdot t_d = 0$

Así de manera general se puede expresar los conflictos mediante ecuaciones como sigue:
Para cada $p_i \in P$, sea $\{t_{i_0}, ..., t_{i_n}\} \in p_i\bullet$ y $|p_i \bullet| \geq 2$ entonces:

$$(\sum_{j=0}^{n} \sum_{k=j+1}^{n} t_{i_j} \cdot t_{i_k}) = 0. \tag{2.6}$$

La ecuación (2.6) se aplica a cada lugar para encontrar todas las ecuaciones que nos representen esta característica en la RP.

En la siguiente sección se presenta un algoritmo que procesa la RP y obtiene los cuatro tipos de ecuaciones que se propusieron en estas subsecciones.

2.4. Algoritmo de traducción

A continuación se presenta la descripción del algoritmo para la traducción de redes de Petri a sistemas de ecuaciones booleanas.

2.4.1. Estrategia general

Paso 1. Se obtienen los datos de entrada (matriz de incidencia, símbolos de salida y marcado inicial) para el procedimientos.

Paso 2. De la matriz de incidencia se toman los datos requeridos por la ecuación 2.3 para generar las ecuaciones de este tipo para cada lugar.

Paso 3. Se crean las ecuaciones que relacionan los lugares con las salidas siguiendo la ecuación 2.4.

Paso 4. Con la ecuación 2.5 se crean las ecuaciones referentes al marcado inicial.

Paso 5. Las características de determinismo son desarrolladas con la ecuación 2.6.

Paso 6. El conjunto de las ecuaciones generados en los pasos 2-5 generan la salida de este algoritmo.

En la siguiente subsección se presenta a detalle el procedimiento de traducción siguiendo la estrategia mencionada.

2.4.2. Procedimiento de traducción

Algoritmo. Traducción de RP

Datos de entrada

A //Matriz de incidencia
Φ //Alfabeto de salida
M_0 //Marcado inicial
φ//Función de salida

Datos de Salida

SystemEquations sistemaBooleano //Objeto que contiene el conjunto de ecuaciones

---

//Inicialización
Caracter *ecuacion*1 ←"", *ecuacion*2 ←"", *ecuacion*3 ←"", *ecuacion*4 ←""
BooleanSystem sistemaBooleano //Objeto que contendrá el conjunto de ecuaciones
Entero $s \leftarrow 0$, $r \leftarrow 0$, $j \leftarrow 0$, $k \leftarrow 0$, $i \leftarrow 0$
//Proceso
Desde $i = 0$ hasta $|P| - 1$ //Ciclo para todos los lugares de la RP
Desde $s = 0$ hasta $|\bullet p_i| - 1$ //Donde $t_{i_s} \in \bullet p_i$
Desde $r = 0$ hasta $|\bullet t_{i_s}| - 1$ //Donde $p_{s_r} \in \bullet t_{i_s}$
//Procedimiento *contatenar*(c_1, var_1) pone
//enseguida de la cadena c_1 la cadena var_1
*ecuacion*1 ← concatenar(*ecuacion*1," $p_{s_r}(k-1)\cdot$")
Fin_Desde
*ecuacion*1 ← concatenar (*ecuacion*1," $t_{i_s}+$")
Fin_Desde
Desde $j = 0$ hasta $|p_i \bullet| - 1$ // Donde $t'_{i_j} \in p_i\bullet$
*ecuacion*1 ← concatenar(*ecuacion*1," $\overline{t'_{i_j}}$ ")
Desde $k = 0$ hasta $|p_i \bullet| - 1$ //Donde $t'_{i_k} \in p_i\bullet$
*ecuacion*2 ← $t'_{i_j} \cdot t'_{i_k} = 0$
//Procedimiento *agregarEcuacion*(s_1, e_1) agrega al objeto s_1 la ecuación e_1
agregarEcuacion(sistemaBooleano,ecuacion2)
Fin_Desde
Fin_Desde
*ecuacion*1 ← concatenar(*ecuacion*1," $p_i(k-1)$")
Desde $j = 0$ hasta $|\Phi|$

```
            Si [φ_ji] = 1
                ecuacion3 ← φ_j = p_i
                agregarEcuacion(sistemaBooleano, ecuacion3)
            Fin_Si
        Fin_Desde
        Si M_0(p_i) = 0
            ecuacion4 ← p_i(0) = 0
            agregarEcuacion(sistemaBooleano, ecuacion4)
        Sino
            ecuacion4 ← p_i(0) = 1
            agregarEcuacion(sistemaBooleano, ecuacion4)
        Fin_Si
    Fin_Desde
    Regresa sistemaBooleano
```

El algoritmo introducido hace un análisis de cada lugar de la red de Petri, por lo tanto de puede afirmar que la complejidad de este procedimiento es polinomial.

2.5. Ejemplo

A continuación se presenta un ejemplo donde un modelo en RP será traducido a un sistema de ecuaciones booleanas.

Ejemplo 2.7. Se aborda el caso de 2 procesos sincronizados (Fig. 2.6). En éste se tiene la siguiente secuencia:
Dos vehículos inicialmente colocados como se muestra en la Fig. 2.6, comienzan simultáneamente a correr a la derecha, cuando se oprime el botón M. Cuando la posición (b o d) sea alcanzada, el vagón correspondiente regresa inmediatamente a su posición inicial (a o c). Cuando la posición inicial es detectada, se detiene el vagón. Un nuevo ciclo comienza cuando los 2 vagones son detenidos.

El modelo de la Fig. 2.7 se muestra el modelado con redes de Petri interpretadas del ejemplo Fig. 2.6 donde:

- $P = \{p_1, p_2, p_3, p_4, p_5, p_6\}$
- $T = \{t_1, t_2, t_3, t_4, t_5\}$
- $\sum = \{a, b, c, d, M\}$

  $\Phi = \{R_1, R_2, L_1, L_2\}$

- $\varphi = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$.

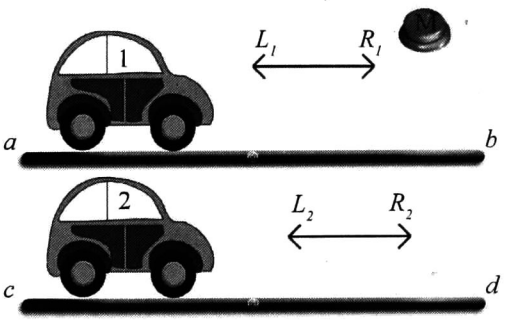


Figura 2.6: Ejemplo de sincronización de 2 procesos

La red de Petri que modela el Ejemplo 2.7 que se muestra en la Fig.2.7 tiene las características que le permiten ser traducidas por las reglas propuestas en la sección 2.3: que sea RPI y binaria.

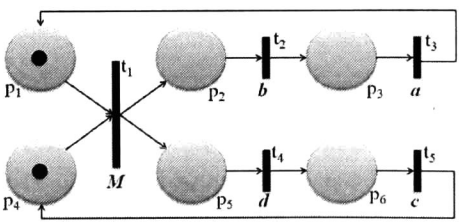


Figura 2.7: Modelado con una Red de Petri del Ejemplo2.7

2.5.1. Estructura de la red de Petri

Para obtener la ecuación estructural de la red de Petri Fig.2.7 se consideran todos los lugares de la red, según lo especificado en la ecuación 2.3. Para el lugar p_1, sean $\{t_3\} \in \bullet p_1$, $\{t_1\} \in p_1 \bullet$ y $\{p_3\} \in \bullet t_3, n = 1, y = 1$, y $m = 1$

$$p_1 = \sum_{s=0}^{1}((\prod_{r=0}^{1} p_3(k-1))\cdot t_3) + ((\prod_{j=0}^{1} \overline{t_1})\cdot p_1 = p_3(k-1)\cdot t_3 + \overline{t_1}\cdot p_1 \tag{2.7}$$

Para los lugares restantes se sigue el mismo procedimiento:

$p_2 = p_1(k-1)\cdot p_4(k-1)\cdot t_1 + \overline{t_2}\cdot p_2$
$p_3 = p_2(k-1)\cdot t_2 + \overline{t_3}\cdot p_3$
$p_4 = p_6(k-1)\cdot t_5 + \overline{t_1}\cdot p_4$
$p_5 = p_1(k-1)\cdot p_4(k-1)\cdot t_1 + \overline{t_4}\cdot p_5$
$p_6 = p_5(k-1)\cdot t_4 + \overline{t_5}\cdot p_6$

2.5.2. Salidas del sistema

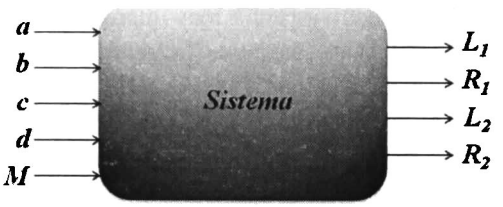


Figura 2.8: Entradas y salidas del sistema Fig. 2.6

Para cada uno de los elementos de la matriz φ,si $[\varphi_{ij}] = 1$ se asocia el elemento del alfabeto de salida ϕ_i al lugar p_j dando como resultado las siguientes ecuaciones:

$$p_2 = R_1 \quad p_3 = L_1 \quad p_5 = R_2 \quad p_6 = L_2$$

2.5.3. Marcado Inicial

Sea $M_0 = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}^T$. el marcado inicial en la RP de la Fig. 2.7.

Con el marcado inicial M_0 de la red de Petri, para cada $p_i \in P$ se generan la siguientes seis ecuaciones:

$$p_1(0) = 1 \quad p_2(0) = 0 \quad p_3(0) = 0$$
$$p_4(0) = 1 \quad p_5(0) = 0 \quad p_6(0) = 0$$

En realidad, para representar el lugar p_i en el instante cero, se denota como $p_i(0)$, osea el estado inicial; por lo que será la inicialización de las variables, de modo que las siguientes ecuaciones substituyen a las anteriores:

$$p_1 = 1 \quad p_2 = 0 \quad p_3 = 0$$
$$p_4 = 1 \quad p_5 = 0 \quad p_6 = 0$$

2.5.4. Conflictos

Se puede observar que el ejemplo considerado es un grafo marcado, por lo consiguiente, $\forall p_i \in P, |p_i \bullet| \leq 1$ por lo tanto no se genera ninguna ecuación para resolver conflictos.

De esta manera, las ecuaciones que se obtuvieron para este ejemplo conformarán el conjunto de ecuaciones que se deberá resolver para encontrar las variables desconocidas.

2.6. Conclusiones

En este capítulo se presentó una metodología para traducir RP a sistemas de ecuaciones booleanas. En ésta se incluyen cuatro aspectos muy importantes para las redes de Petri binarias: la estructura, el marcado inicial, la asociación de lugares a salidas del sistema y el determinismo que debe existir para que se respete el modelado brindado por las mismas.

Capítulo 3

Desarrollo de un traductor de redes de Petri

Resumen

En este capítulo se presenta el desarrollo del traductor, describiendo una parte del proceso de ingeniería de software. Se ilustra la utilización del módulo desarrollado a través de un caso de estudio.

3.1. Herramientas de análisis

Existen diversas herramientas orientadas a redes de Petri, con éstas se pretende aprovechar los beneficios que dan los métodos formales para la captura y análisis de sistemas de eventos discretos. Las herramientas con las que se cuenta se pueden clasificar en tres tipos: **editores, simuladores y analizadores**.[Ara10]

Editores: permiten editar redes de Petri. Ejemplos:
CoopBuilder, FLOWer, Lola, Maria, MISS-RdP, Snoppy, SPADES-V1, SPADES.

Simuladores: permiten simular el sistema, esto es, visualizar la evolución de la red. Ejemplos: F-net, ARP, HiQPN-Tol, SPADES, SPADES-V2.

Analizadores: permiten analizar la red de Petri, con diversos procedimientos de análisis. Ejemplos: Maria, Great-SPN, ExSpect, PIPE2, Predator, JFern, SPADES-V2.

La variedad de editores que existen dan como resultado muchas opciones para el trabajo con Redes de Petri. El editor que se eligió para desarrollar el traductor es el SPADES-V2 (Specification and Analysis of DiscreteEvent Systems versión 2) ya que cumple con las características de ser un editor, simulador y analizador; además permite al usuario agregar nuevos procedimientos de análisis en la herramienta, sin que afecte al sistema y sin necesidad de compilar código previamente existente, por lo que la modificabilidad se da de manera natural. Por lo anterior la automatización del proceso de traducción de redes de Petri a sistemas de ecuaciones booleanas se implementó en un módulo de la herramienta mencionada.

El SPADES-V2 es una herramienta desarrollada en el Cinvestav Guadalajara que nos permite trabajar con redes de Petri. Posee las siguientes características: es un Editor Gráfico, un simulador, está desarrollado en el lenguaje JAVA, su documentación fue desarrollada en UML, cuenta con una arquitectura siguiendo el Modelo-Vista-Controlador(MVC) y posee la facildad de agregación de procedimentos de análisis.

3.2. Requerimientos del módulo de traducción

El objetivo del módulo es proporcionar la funcionalidad de traducción de redes de Petri a un sistema de ecuaciones booleanas. Por las facilidades presentadas por el SPADES-V2, se escogió este software como la herramienta base de desarrollo. Se creará un procedimiento de análisis para extender este software.

Para el descubrimiento de los requerimientos se consideró el análisis de la herramienta a usar, el objetivo planteado, y la funcionalidad deseada por el usuario [Som05]. En la Fig. 3.2 se presentan los requerimientos del sistema, así como sus dependencias. Los principales requerimientos conciernen a la interfaz, la funcionalidad, la compatibilidad, entre otros y se describen a

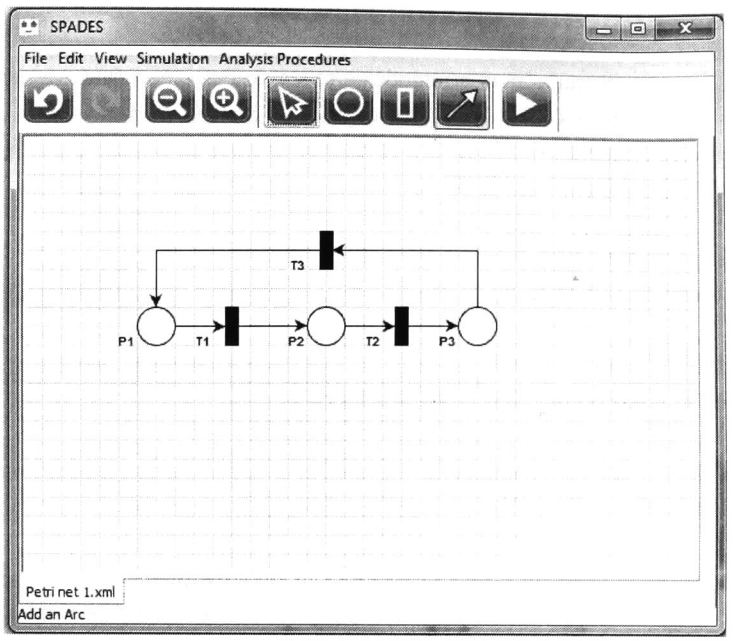


Figura 3.1: Interfaz SPADES-V2

continuación:

- El sistema provee el módulo de traducción de RP a SEB (*Sistemas de Ecuaciones Booleanas*). De manera general, el usuario tendrá la función de introducir una RP y el sistema dará como salida el conjunto de ecuaciones.

- El sistema será un módulo del SPADES V-2. Dicho módulo será desarrollado en el software mencionado, por lo cual se deberán considerar las restricciones del mismo para la agregación de procedimientos de análisis. La interfaz de este módulo contará con el despliegue del conjunto de ecuaciones.

- El sistema debe implementar el algoritmo de traducción detallado en la sección 2.4. La entrada del método se tomará de la interfaz proporcionada por el SPADES V-2 y la salida será presentada en la interfaz del módulo.

- El sistema permitirá la traducción de las redes que se encuentran en la interfaz del SPADES V-2 las cuales son las entradas para el algoritmo de traducción.

- El sistema proveerá la facilidad de agregar a la salida datos que son externos a la traducción y que la pueden completar. La interfaz deberá contemplar campos para las variables, las hipótesis, especificaciones adicionales, prioridades e información.

El sistema en su interfaz de variables permitirá la modificación para asignarle el tipo de variables que se maneja, ya sea que pertenezca cada variable a una definición, que sea conocida o una incógnita.

- El sistema proveerá una interfaz para mostrar los resultados. En ésta se presentarán tanto el conjunto de ecuaciones booleanas resultantes del procesamiento del algoritmo, como la separación en las clasificaciones asignadas a las mismas, esto para cubrir con la compatibilidad con el módulo BESS (Boolean Equation System Solver). Este módulo desarrollado en el Laboratorio Universitario de Investigación en Producción Automatizada (LURPA) obtiene las soluciones para el conjunto de ecuaciones booleanas.

- El sistema proveerá un repositorio para presentar la salida del método a un archivo compatible con la estructura especificada por el software BESS.

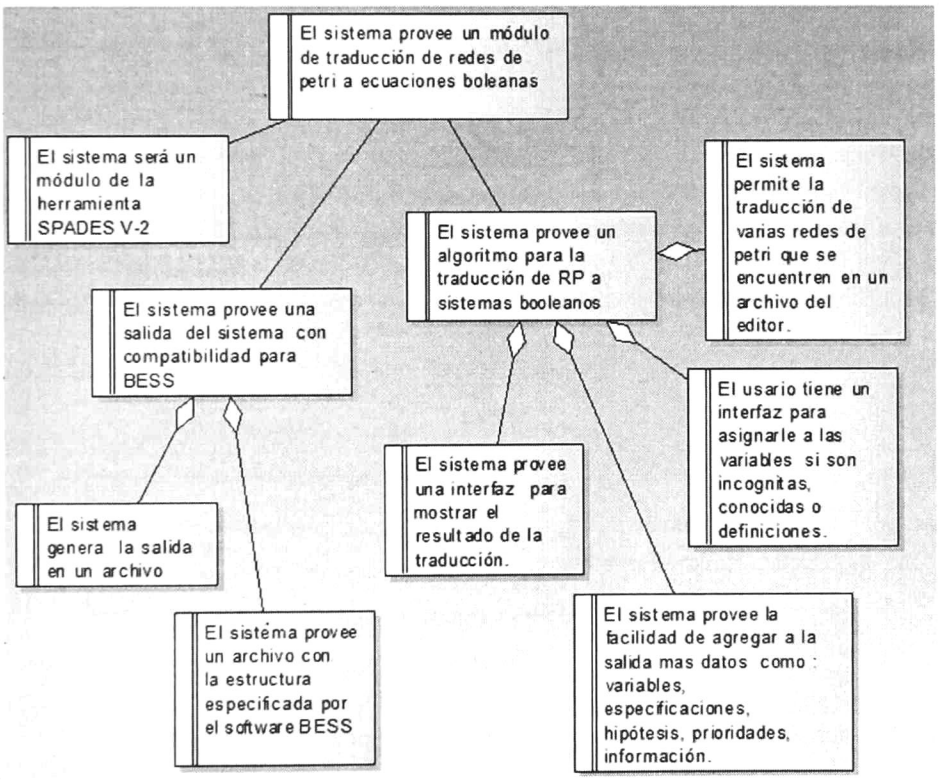


Figura 3.2: Diagrama de requerimientos

3.3. Modelo arquitectónico del Software

Tomando en cuenta los requerimientos se consideró que para cumplirlos se crearía un subsistema que funcione como parte del SPADES V-2. El modelo arquitectónico creado se puede observar en la Fig.3.3; en éste se muestran las interacciones de los bloques detectados así como se muestran los principales elementos a desarrollar y a organizar.

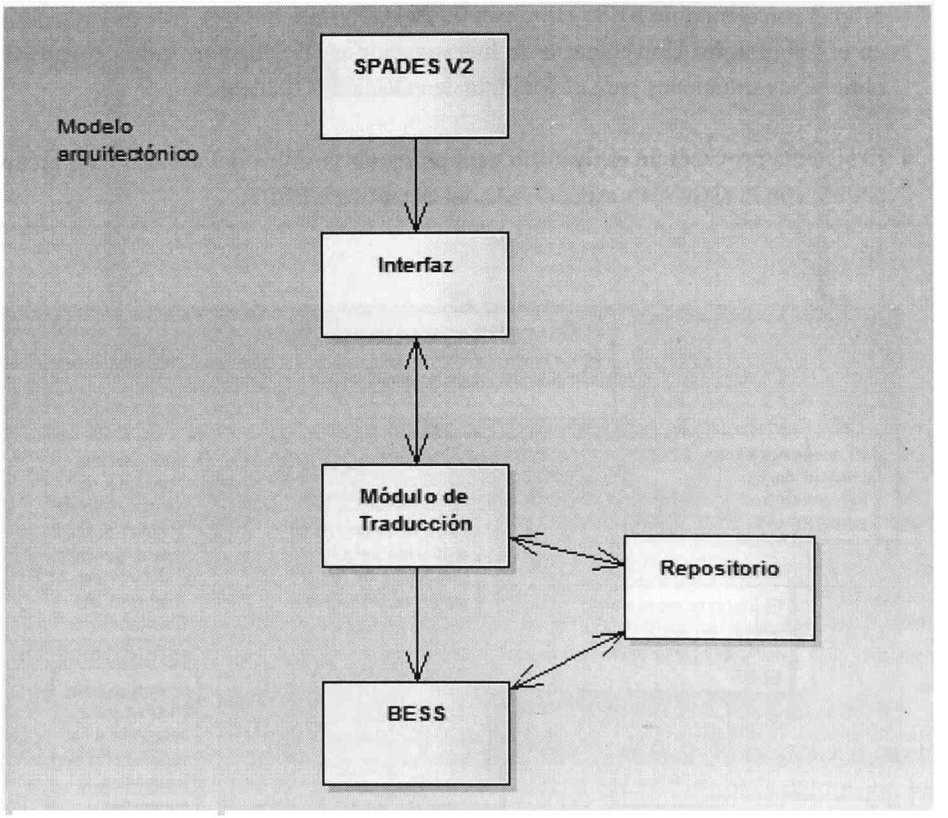


Figura 3.3: Modelo arquitectónico del subsistema

El modelo está conformado por los siguientes componentes principales:

- El SPADES-V2 funciona como la interfaz gráfica para hacer la captura de los datos que necesita nuestro módulo de traducción. En este caso, es necesario para ingresar las redes de Petri que posteriormente serán utilizadas.

- El Módulo de Traducción se encargará de tomar la información proporcionada por la interfaz gráfica descrita anteriormente, y de aplicar el algoritmo de traducción para dar como resultado el conjunto de ecuaciones booleanas que describen la red. Además permitirá introducir más información del sistema, en caso de que se necesiten más especificaciones que las traducidas de las redes de Petri.

- El conector entre el editor gráfico y el módulo principal para la traducción es la Interfaz entre ambos.

- El BESS es el sistema que procesa el sistema de ecuaciones.

- El Repositorio consistirá en el almacenamiento del conjunto de ecuaciones booleanas en el formato utilizado por BESS.

3.4. Implementación del Software

Los subsistemas descritos dan pie al desarrollo del diagrama de componentes presentado en la Fig. 3.4, el cual está formado por tres componentes. Si efectuamos la relación de los componentes con el modelo arquitectónico se observará que el SPADES-V2 es el componente sobre el cual funcionará el nuevo módulo de traducción. La interfaz entre estos dos elementos está provista por el SPADES-V2. El módulo de traducción provee la funcionalidad de transformar RP a sistemas de ecuaciones booleanos asi como la funcionalidad del repositorio, el cual funge como la interfaz entre el BESS y el módulo.
Detallando el componente del traductor donde se implementará toda la funcionalidad se tiene

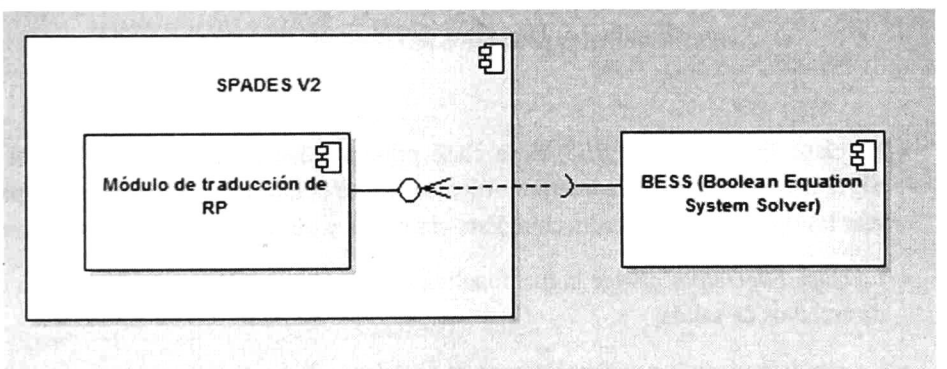


Figura 3.4: Diagrama de componentes del traductor

el diagrama de clases expuesto en la Fig. 3.5. A continuación se hace una breve descripción de las clases que conforman el módulo y la funcionalidad o estructura que provee cada una.

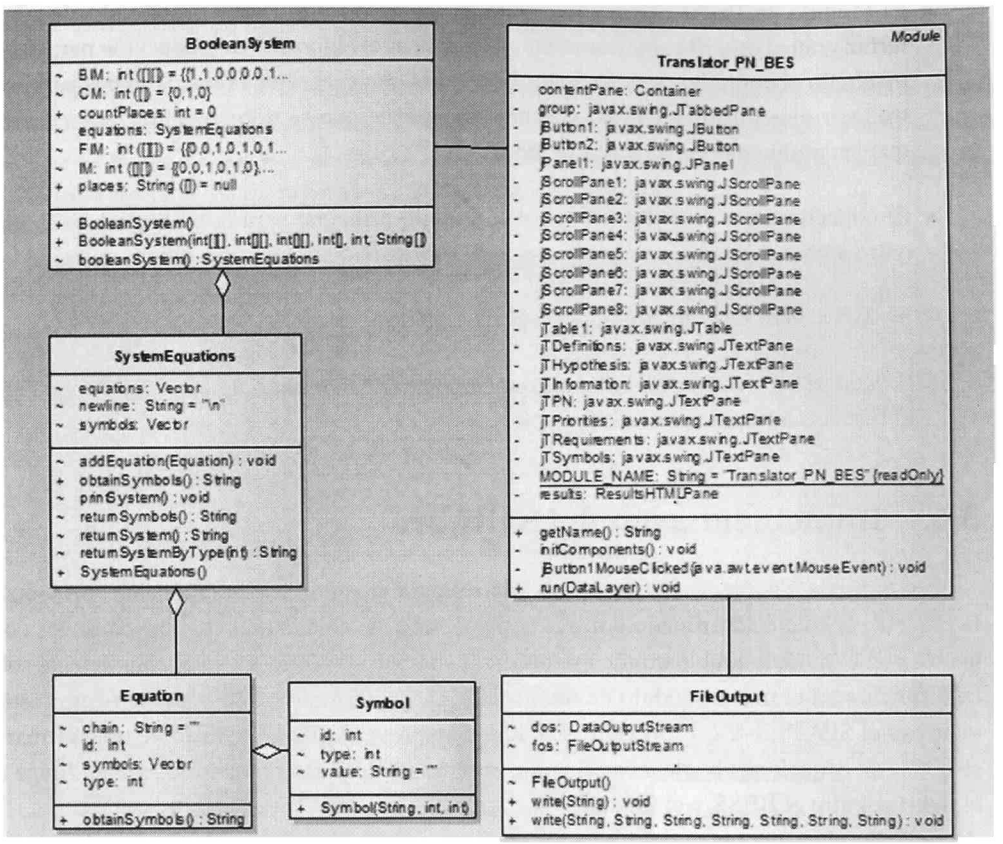


Figura 3.5: Diagrama de clases del traductor

- La clase *Translator_PN_BES*, es la clase principal que interactúa tanto con el sistema SPADES-V2 como con la interfaz. Además provee la interfaz gráfica necesaria para efectuar las operaciones de traducción. Esta clase crea y utiliza la clase *BooleanSystem*.
- La clase *FileOutput* provee la funcionalidad requerida por el repositorio, para la creación de archivos de salida.

  La clase *BooleanSystem* implementa el algoritmo de traducción obteniendo en ésta el conjunto de ecuaciones con la información obtenida en la red de Petri.
- La clase *SystemEquations* es la clase que representa el vector del conjunto de ecuaciones.
- La clase *Equation* representa la ecuación individual.

  La clase *Symbol* representa los símbolos presentes en cada ecuación.

3.5. Integración en el Software SPADES-V2

En los requerimientos se estableció que el traductor sería un módulo del SPADES-V2. Dicha funcionalidad se provee utilizando una estructura predefinida, por lo que el paquete Translator_PN_BES queda incorporado como un procedimiento de análisis en el SPADES. En la Fig.

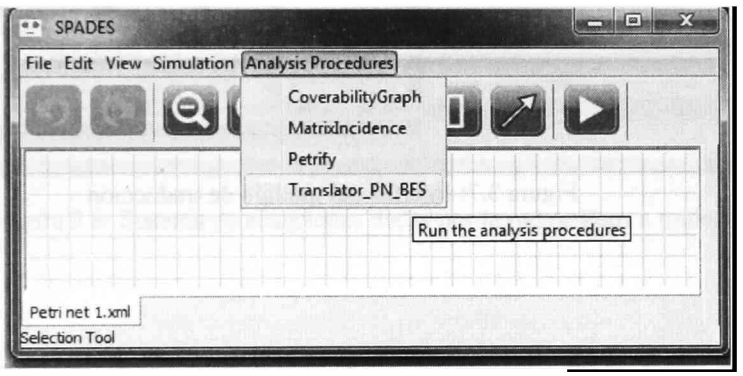


Figura 3.6: Integración del procedimiento de análisis

3.6 se presenta cómo el método en el menú es agregado, y así se vuelve parte de las herramientas del editor. Una vez seleccionado el método, se trabaja con la información que hay en él y comienza el proceso de traducción. En la Fig. 3.7 se presenta este proceso. Observamos la red de Petri a la izquierda, y en la parte derecha la interfaz que se obtiene al hacer la traducción. A continuación se presentan las interfaces con las que se puede finalizar el proceso.

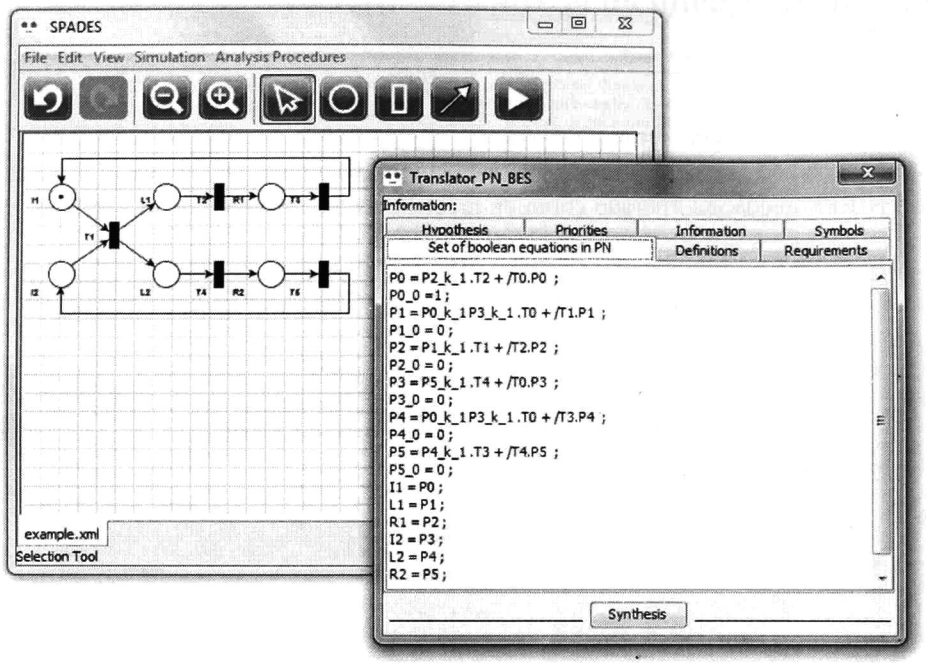


Figura 3.7: Interfaz del módulo de traducción

3.6. Interfaces

En esta sección se presentan las diversas interfaces que se tienen para el módulo desarrollado. Este módulo realiza la transformación automática de las RP en sistemas de ecuaciones boleanas. Las interfaces para este módulo están descritas en las Fig. 3.8, 3.9, 3.10, 3.11, 3.12, 3.13, 3.14 y 3.15.

3.6.1. Traducción

En la Fig. 3.8 se presenta una red de Petri que es traducida por el módulo. Las ecuaciones resultantes son repartidas en las diversas interfaces, ya que cada ecuación puede ser una hipótesis, una definición o un requerimiento.

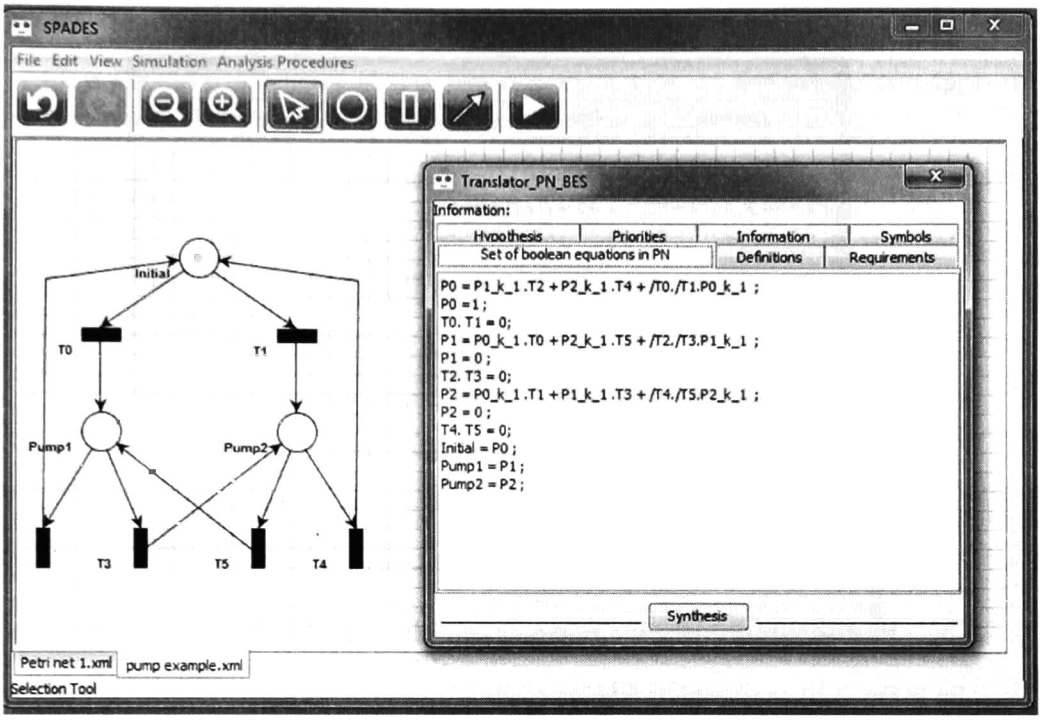


Figura 3.8: Sistema de ecuaciones dadas por la red de Petri a traducir.

3.6.2. Definiciones

En la Fig. 3.9 se muestran las definiciones. Las ecuaciones que obtenemos al representar los lugares pertenecen a esta categoría. Si se desean agregar más la siguiente expresión regular nos muestra cómo añadirlas.

$$(nombre = Expresion;)^* \tag{3.1}$$

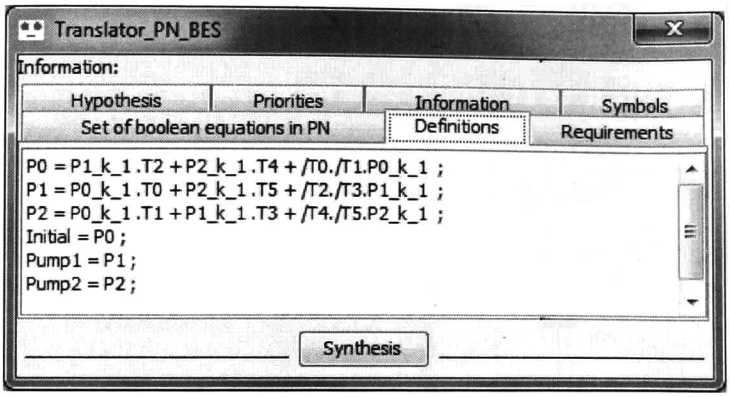


Figura 3.9: Definiciones en la especificación

3.6.3. Requerimientos para la especificación del sistema

En la Fig. 3.10 se muestra la interfaz donde se muestran las ecuaciones que se obtuvieron que resuelven el problema de los conflictos en la RP. Si el usuario desea agregar más, tiene que ingresarlas en el siguiente formato:

$$(nombreRequerimiento : Relacion;)^* \tag{3.2}$$

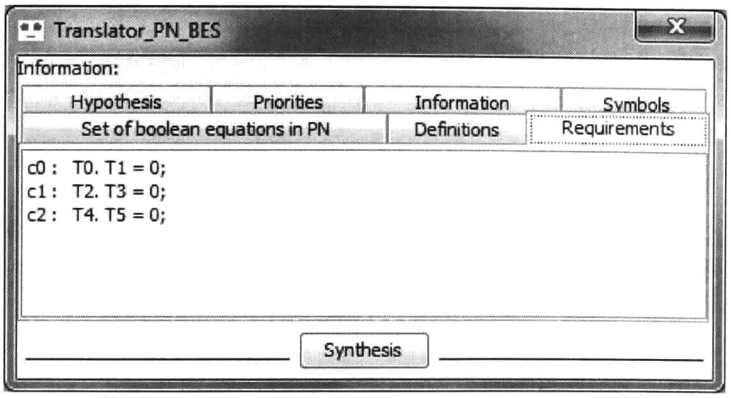


Figura 3.10: Requerimientos en la especificación

3.6.4. Hipótesis

En la Fig. 3.11 se tiene la interfaz para la especificación, en ésta van las ecuaciones del marcado inicial. Si se desean agregar más:

$$(nombreHipotesis : Relacion;)^* \tag{3.3}$$

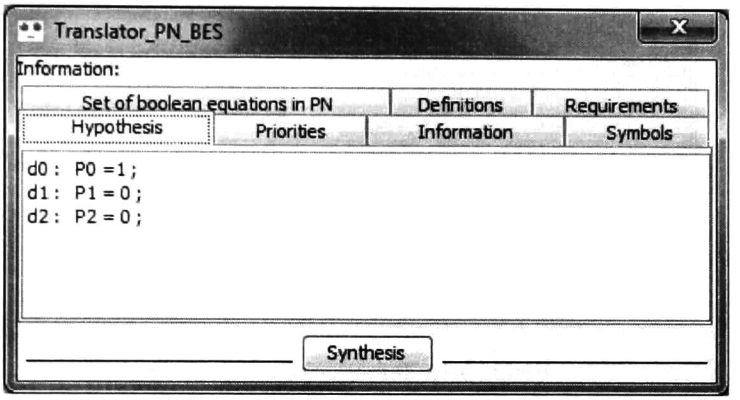


Figura 3.11: Hipótesis en la especificación

3.6.5. Prioridades

El software BESS funciona con BDDs, por lo cual el orden de variables es fundamental para la resolución. Para agregar prioridades en esta interfaz mostrada en la Fig. 3.12 se sigue el siguiente formato:

$$(variable >> variable;)^* \tag{3.4}$$

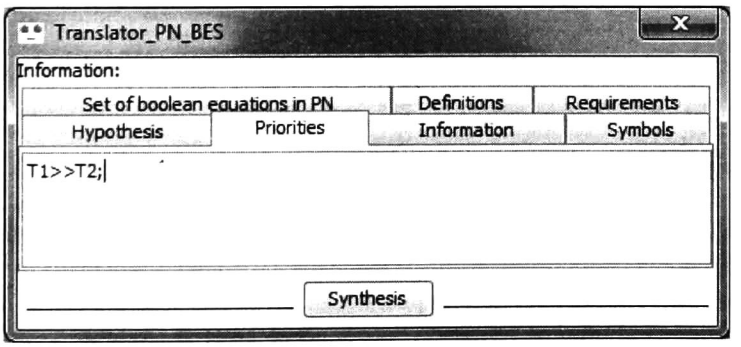


Figura 3.12: Prioridades en las variables del sistema.

3.6.6. Información

Esta interfaz permite agregar información general del sistema en el archivo que se generará, con el fin de brindarle al usuario la oportunidad de especificar el archivo con el que se está trabajando. Ésta se muestra en la Fig. 3.13.

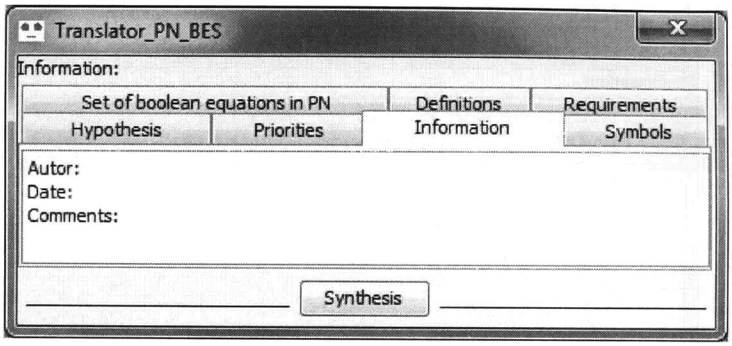


Figura 3.13: Información del problema.

3.6.7. Variables

En la Fig. 3.14 se muestra esta interfaz. En ella se decide si son incógnitas, conocidas o definiciones las variables utilizadas en la traducción. Para completar el proceso se requiere que se especifique qué tipo de variables son, sustituyendo la palabra *complete* con *Known*, *Defined*, o *Unknown*.

Si se desean agregar más variables, se declaran con la siguiente notación:

$$(variable : [Known \mid Defined \mid Unknown];)^* \tag{3.5}$$

3.7. Ejemplo

Aplicando al Ejemplo 2.7 el proceso automatizado de traducción, se observa que los resultados brindados por el módulo son los mismos que los que se muestran en el ejemplo desarrollado manualmente. En la sección 2.5 esto se puede apreciar en la Fig. 3.7.

Si se presiona el botón de Synthesis, se creará un archivo con toda la información que se encuentra en la interfaz. Para utilizar el BESS, solo arrastramos el archivo que fue generado partiendo de la ruta donde tenemos el SPADES-V2 en \structure \modules \Translator_PN_BES \Solver \Bess en el ejecutable bess.exe como se muestra en la Fig. 3.15.

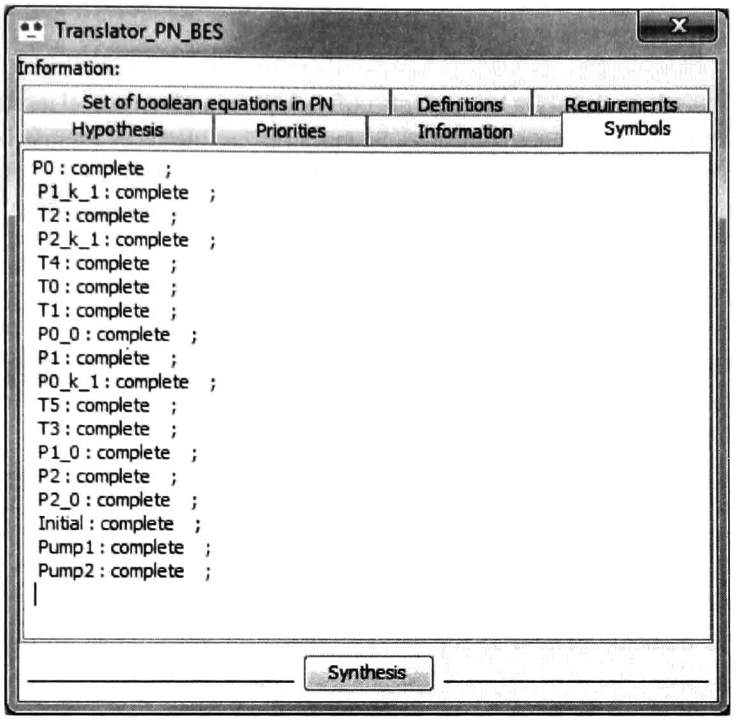


Figura 3.14: Variables del sistema.

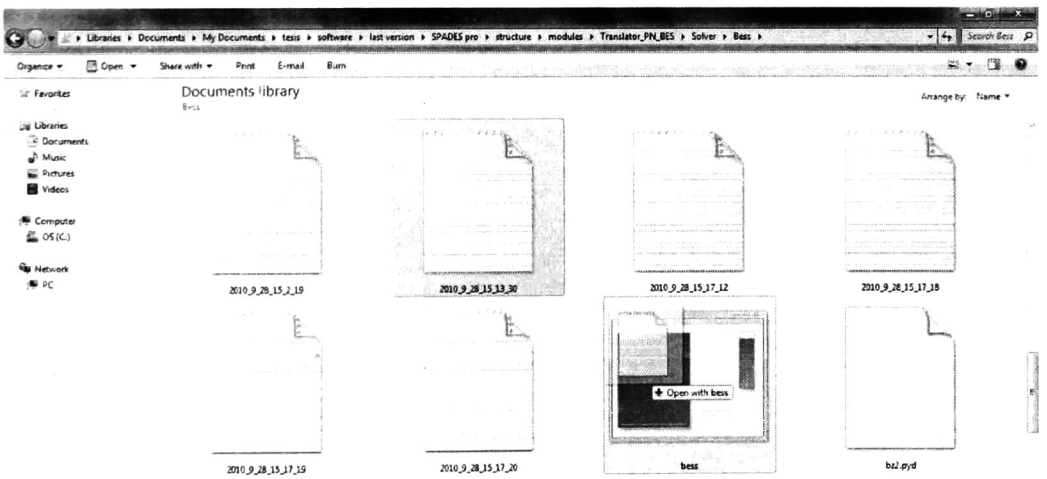


Figura 3.15: Ejemplo de cómo utilizar el archivo generado en BESS

3.8. Conclusiones

En este capítulo se detalla la herramienta utilizada para implementar el módulo de traducción de RP a sistemas de ecuaciones booleanas. Se muestran las interfaces además de la sintaxis para ser usada en cada una de ellas. Retomamos el ejemplo que se trabajó en el capítulo anterior y se efectuó la traducción con el nuevo módulo obteniendo las mismas ecuaciones.

Capítulo 4

Casos de estudio

Resumen

Se presentan dos de los casos de estudio con los que se probó el módulo de traducción de RP a sistemas de ecuaciones booleanos.

4.1. Introducción

En este capítulo se presentan los resultados obtenidos aplicados a dos casos de estudio. El primer caso en la subsección (4.2) es tomado del artículo [HJML08], en el cual se utiliza redes de Petri. El segundo caso de estudio presenta un protocolo de producción simple (4.3.1).

4.2. Caso de estudio 1

El artículo [HJML08] presenta un sistema que usa una red de Petri como solución parcial y aplica el método de síntesis. En este caso de estudio se muestra que el algoritmo de traducción propuesto en el capítulo 2 cubre los requerimientos, y el software descrito en el capítulo 3 es utilizado para la traduccion asi como para generar el archivo compatible con la herramienta BESS y presentar una solución al problema.

4.2.1. Descripción del caso de estudio

El sistema consiste en dos máquinas que son parte de un proceso de producción. En éste sólo una máquina se encuentra en funcionamiento. La Fig. 4.1 muestra la cadena de producción, y la Fig. 4.2 muestra las variables de entrada y salida que maneja el controlador que se ocupará de coordinar la producción en las 2 máquinas. Dicho controlador cuenta con cuatro entradas y dos salidas que son descritas a continuación:

- Entradas:

  1. **Dem**: pieza de trabajo pedido por la cadena de mando.
  2. **Pm1**: detección de falla en la máquina 1.
  3. **Pm2**: detección de falla en la máquina 2.
  4. **Pc**: falla global.

  Salidas

  1. **Mach1**: Se dirige el flujo de trabajo a la maquina 1.
  2. **Mach2**: Se dirige el flujo de trabajo a la máquina 2.

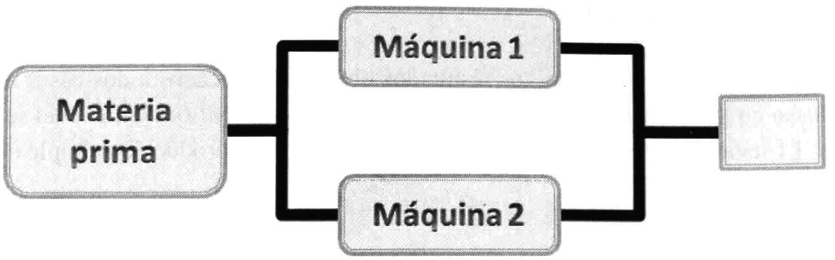


Figura 4.1: Cadena de producción[HJML08]

Figura 4.2: Entradas/Salidas del controlador [HJML08]

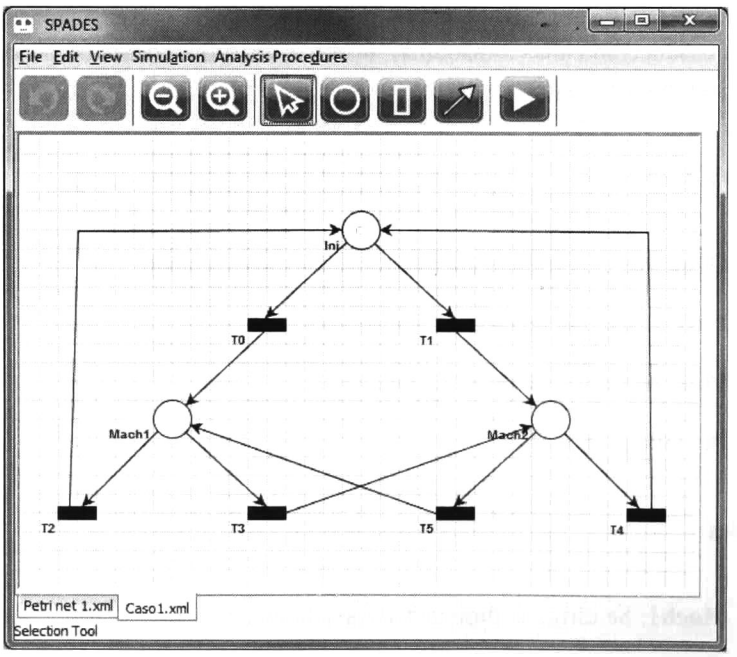


Figura 4.3: Solución Parcial

Una solución parcial que se propone a este sistema es expresada con la RP interpretada de la Fig. 4.3 donde las etiquetas de los lugares son los símbolos ϕ asociados a cada lugar. Por ejemplo el símbolo $\phi_1 = Mach1$, $\phi_2 = Mach2$, etc.

4.2.2. Traducción

Con la solución parcial de la Fig. 4.3 y la descripción de la subsección 4.2.1 realizamos la traducción en el módulo de traducción del SPADES V-2. El resultado se muestra en la Fig. 4.4 siendo el conjunto de ecuaciones la traducción resultante de la solución parcial. En las figuras 4.5, 4.6 y 4.7 se muestra cómo el sistema dividió las ecuaciones en el tipo de entrada al que pertenece ya sea requerimiento, hipótesis o definición.

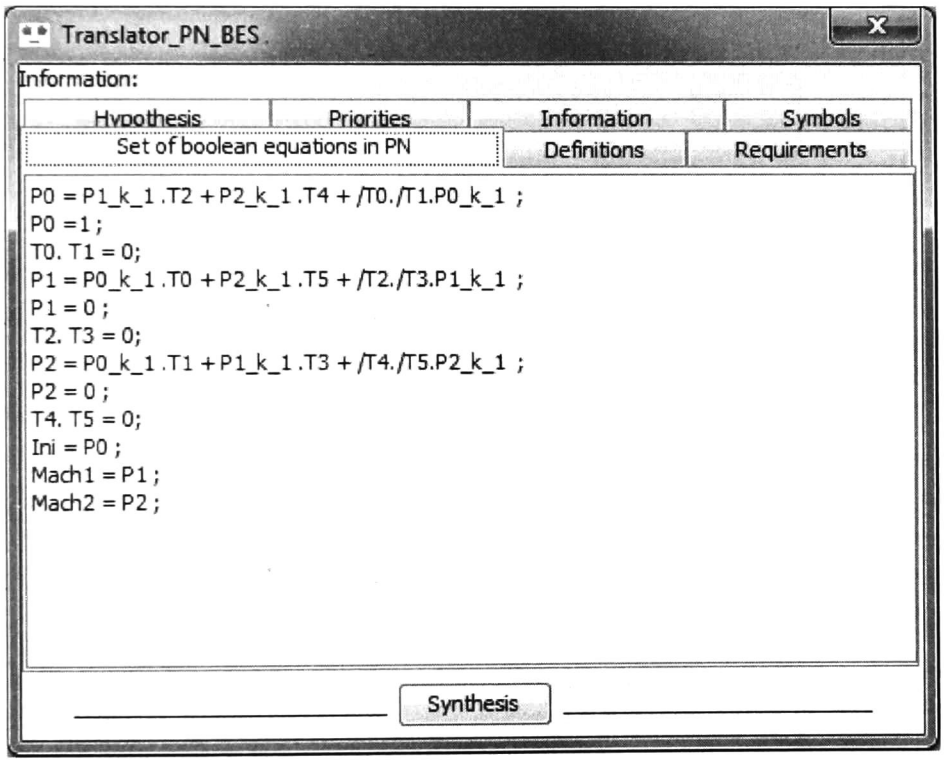


Figura 4.4: Salida de la traduccion de la RP de la Fig. 4.3

Al comparar los resultados obtenidos con los del artículo [HJML08], se corrobora que el módulo regresa la misma traducción. Ahora falta completar la especificación con los requerimientos funcionales del sistema para crear el controlador de éste.

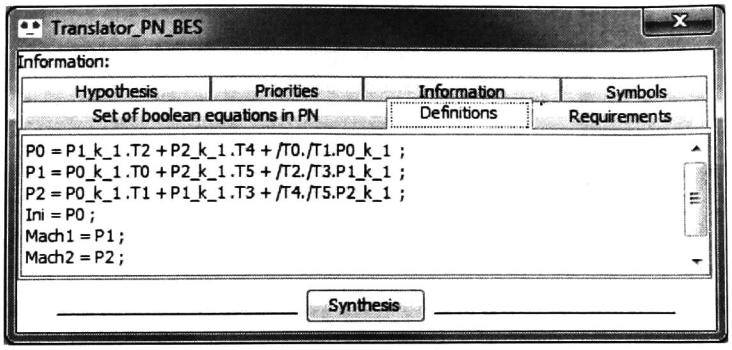


Figura 4.5: Definiciones resultantes de la traducción.

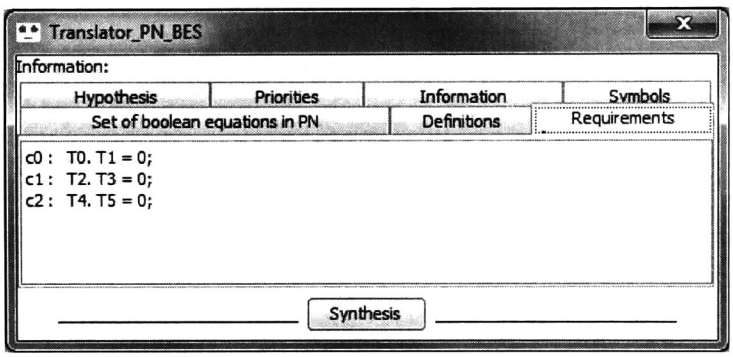


Figura 4.6: Requerimientos resultantes de la traducción.

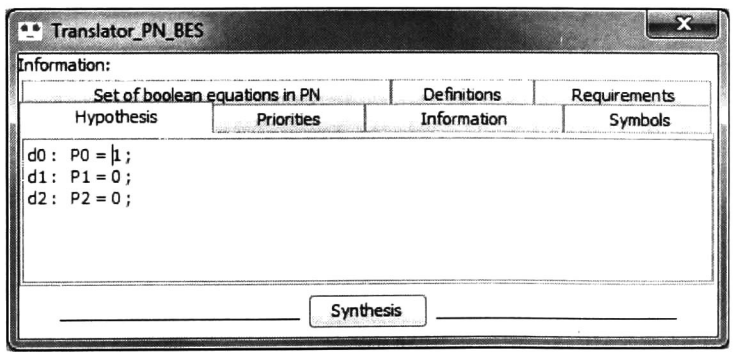


Figura 4.7: Hipótesis resultantes de la traducción.

4.2.3. Síntesis

Para el proceso de síntesis se utiliza la entrada que se encuentra en el texto enmarcado. En éste se presentan las entradas que se piden en la interfaz del software BESS. El este archivo de texto fue generado por el modulo spades V-2 y cuenta con la traducción de la solución parcial mas un conjunto de ecuaciones agregadas para completar la especificación y así obtener el control en general.

```
<PROBLEME>
<VARIABLES>
T0: Inconnue;
T1: Inconnue;
T2: Inconnue;
T3: Inconnue;
T4: Inconnue;
T5: Inconnue;
Pm1: Donnee;
Pm2: Donnee;
Pc: Donnee;
Dem: Donnee;
P0_k_1: Donnee;
P1_k_1: Donnee;
P2_k_1: Donnee;
P0: Declaration;
P1: Declaration;
P2: Declaration;
initial: Declaration;
Mach1: Declaration;.
Mach2: Declaration;
</VARIABLES>

<DECLARATIONS>
P0 = P1_k_1. T2 + P2_k_1. T4 + /T0./T1.P0_k_1;
P1 = P0_k_1. T0 + P2_k_1. T5 + /T2./T3.P1_k_1;
P2 = P0_k_1. T1 + P1_k_1. T3 + /T4./T5.P2_k_1;
initial = P0;
Mach1 = P1;
Mach2 = P2;
</DECLARATIONS>
```

```
<ASSERTIONS>

c0: T0. T1 = 0;
c1: T2. T3 = 0;
c2: T4. T5 = 0;

s1: Mach1.Mach2 = 0;
s2: Mach1+ Mach2 =<Dem;
s3: Mach1.Pm1 = 0;
s4: Mach2.Pm2 = 0;
s5: (Mach1+Mach2 ). Pc= 0;
s6: /(Pc+ (Pm1.Pm2)). Dem =<Mach1+ Mach2;
s7: /Pm1./Pm2.(T3+T5) =0;
s8: P0+P1+P2=1;
s9: P0 =</P1./P2;
s10: P1 =</P0./P2;
s11: P2 =</P1./P0;

</ASSERTIONS>

<HYPOTHESES>

d0: P0=1;
d1: P1= 0;
d2: P2 = 0;
d9: P0_k_1 =</P1_k_1. /P2_k_1;
d10: P1_k_1 =</P0_k_1. /P2_k_1;
d11: P2_k_1 =</P1_k_1. /P0_k_1;

</HYPOTHESES>
</PROBLEME>
```

4.2.4. Resultado

La salida del sistema proporcionado por el sistema **BESS** es:

```
Operations duration:
- Gathering equations: 23 ms
- Dimension of the BDD that represents the equation: 198
- Dimension of the BDD that represents the assumptions: 29
- Solving: 49 ms
<SOLUTIONS>
* T0 = /Pm1.Pm2./Pc.Dem./P1_k_1./P2_k_1+p_T0.(P1_k_1+P2_k_1+/Pm1./Pc.Dem)
* T1 = (Pm1./Pm2./Pc.Dem./P1_k_1./P2_k_1+/Pm2./Pc.Dem./P1_k_1./P2_k_1./p_T0)+
p_T1.(P1_k_1./p_T0+P2_k_1./p_T0)
* T2 = (Pc.P1_k_1+/Dem.P1_k_1+Pm1.Pm2.P1_k_1)+p_T2.(/P1_k_1)
* T3 = Pm1./Pm2./Pc.Dem.P1_k_1+p_T3.(Pm1./P1_k_1./p_T2+Pm2./P1_k_1./p_T2)
* T4 = (Pc.P2_k_1+/Dem.P2_k_1+Pm1.Pm2.P2_k_1)+p_T4.(/P2_k_1)
* T5 = /Pm1.Pm2./Pc.Dem.P2_k_1+p_T5.(Pm1./P2_k_1./p_T4+Pm2./P2_k_1./p_T4)
</SOLUTIONS>
```

4.3. Caso de estudio 2

El segundo caso de estudio consiste en una cadena de producción simple. Se tiene un proceso a efectuar, y se decide su destino: si es defectuoso el producto se aparta de los que fueron exitosos y si fue terminado exitosamente, está listo para ser llevado al almacén como se observa en la Fig. 4.8.

4.3.1. Descripción del caso de estudio

- Entradas:

  1. **Petición**: pieza de trabajo pedido por la cadena de mando.
  2. **Error**: el proceso falló y el producto es defectuoso.

- Salidas:

  1. **Inicio**: Se inicia el proceso de producción.
  2. **Proceso**: El producto se elabora.
  3. **Fallo**: El producto falló en su elaboración y debe ser mandado a los productos defectuosos.
  4. **Almacén**: El producto está listo para su empaque.

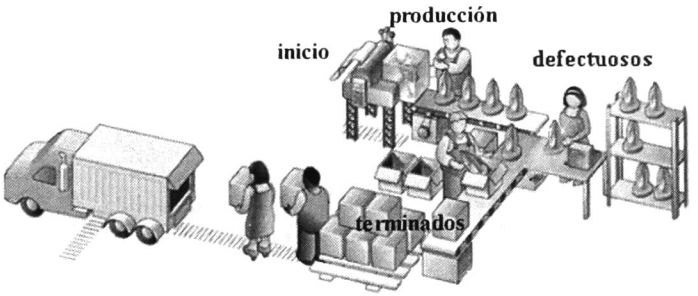


Figura 4.8: Cadena de producción

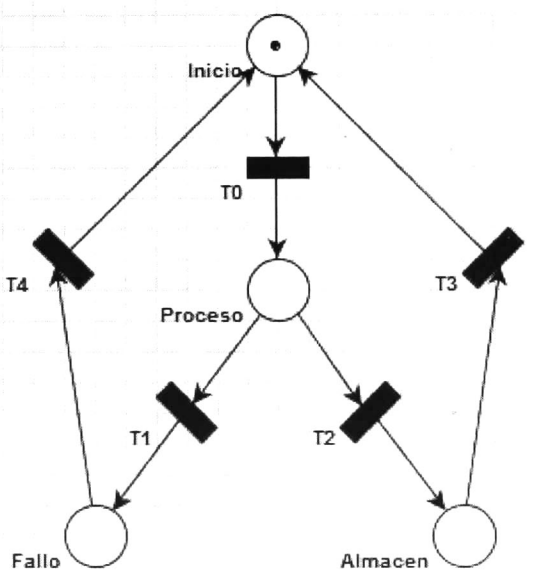


Figura 4.9: Solución Parcial

Una solución parcial que se propone a este sistema, es expresada con la red de Petri interpretada en la Fig. 4.9

4.3.2. Traducción

Con la solución parcial de las subsección 4.3 realizamos la traducción en el módulo de traducción del SPADES V-2. El resultado se muestra en la Fig. 4.10

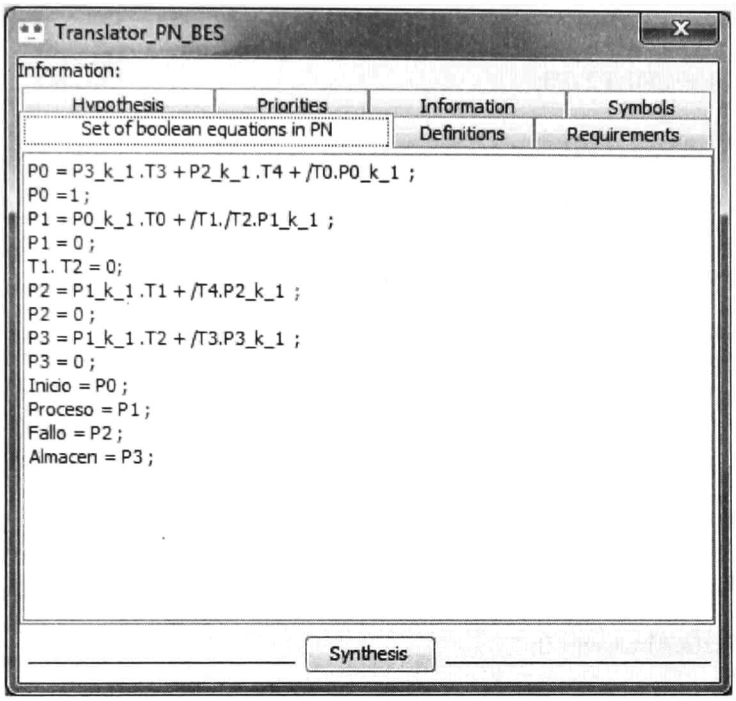


Figura 4.10: Traducción de la solución parcial a un sistema de ecuaciones booleanas

Con la información obtenida de la traducción de la solución parcial, se pasa a un proceso donde se agregan más especificaciones, para modelar completamente el comportamiento del sistema considerando las nuevas variables.

4.3.3. Síntesis

Para completar la traducción se agregaron requerimientos que se describen a continuación. Para asegurar que sea un trabajo secuencial se agregaron las siguientes cuatro ecuaciones:

- c1: T0. (T2+T1) = 0;

  c2: T1. T3 = 0;

- c3: T2. T4 = 0;

- c4: T0. (T3 +T4)= 0;

Para asegurar que el comportamiento de la red es binaria:

c8: P0+P1+P2+P3=1;

- c9: P0 =</P1./P2./P3;
- c10: P1 =</P0./P2./P3;
- c11: P2 =</P1./P0./P3;
- c12: P3=</P1./P0./P2;

Asociamos la presencia de sensores a las salidas del sistema;

c5: Proceso =<Peticion;

c6: Fallo=<Error;

- c7: Almacen=</ Error;

Condiciones de seguridad en el sistema:

- c13: Fallo.Almacen= 0;
- c14: Error.Almacen= 0;
- c15: T0 =<P0_k_1;
- c18: T3 =<P3_k_1;
- c19: T4 =<P2_k_1;

A las hipótesis se agregaran las siguientes ecuaciones:

- d8: P0_k_1+P1_k_1+P2_k_1+P3_k_1=1;
- dc9: P0_k_1 =</P1_k_1./P2_k_1./P3_k_1;
- d10: P1_k_1 =</P0_k_1./P2_k_1./P3_k_1;
- d11: P2_k_1 =</P1_k_1./P0_k_1./P3_k_1;
- d12: P3_k_1 =</P1_k_1./P0_k_1./P2_k_1;

Para el proceso de síntesis se utiliza la siguiente entrada:

```
<PROBLEME>
<VARIABLES>

P0_k_1: Donnee;
P1_k_1: Donnee;
P2_k_1: Donnee;
P3_k_1: Donnee;

T0: Inconnue;
T1: Inconnue;
T2: Inconnue;
T3: Inconnue;
T4: Inconnue;

P0: Declaration;
P1: Declaration;
P2: Declaration;
P3: Declaration;

Inicio: Declaration;
Proceso: Declaration;
Fallo: Declaration;
Almacen: Declaration;
Peticion: Donnee;
Prendida: Donnee;
Error: Donnee;
</VARIABLES>

<DECLARATIONS>
P0 = P3_k_1. T3 + P2_k_1. T4 + /T0.P0_k_1;
P1 = P0_k_1. T0 + /T1./T2.P1_k_1;
P2 = P1_k_1. T1 + /T4.P2_k_1;
P3 = P1_k_1. T2 + /T3.P3_k_1;
Inicio = P0;
Proceso = P1;
Fallo = P2;
Almacen = P3;
</DECLARATIONS>
```

```
<ASSERTIONS>
c0: T1. T2 = 0;

c1: T0. (T2+T1) = 0;
c2: T1. T3 = 0;
c3: T2. T4 = 0;
c4: T0. (T3 +T4)= 0;

c8: P0+P1+P2+P3=1;
c9: P0 =</P1./P2./P3;
c10: P1 =</P0./P2./P3;
c11: P2 =</P1./P0./P3;
c12: P3=</P1./P0./P2;

c5: Proceso =<Peticion;
c6: Fallo=<Error;
c7: Almacen=</ Error;

c13: Fallo.Almacen= 0;
c14: Error.Almacen= 0;
c15: T0 =<P0_k_1;
c18: T3 =<P3_k_1;
c19: T4 =<P2_k_1;

</ASSERTIONS>
<HYPOTHESES>
d0: P0 =1;
d1: P1= 0;
d2: P2 = 0;
d3: P3= 0;

d8: P0_k_1+P1_k_1+P2_k_1+P3_k_1=1;
dc9: P0_k_1 =</P1_k_1./P2_k_1./P3_k_1;
d10: P1_k_1 =</P0_k_1./P2_k_1./P3_k_1;
d11: P2_k_1 =</P1_k_1./P0_k_1./P3_k_1;
d12: P3_k_1 =</P1_k_1./P0_k_1./P2_k_1;
</HYPOTHESES>
</PROBLEME>
```

4.3.4. Resultados

Los resultados obtenidos del procesamiento de la entrada de la subsección 4.3.3 son presentados a continuación:

```
Operations duration:
- Gathering equations: 9 ms
- Dimension of the BDD that represents the equation: 44
- Dimension of the BDD that represents the assumptions: 26
- Solving: 18 ms

<SOLUTIONS>

* T0 = p_T0.(/P2_k_1./P3_k_1.Peticion)

* T1 = p_T1.(P2_k_1+/P3_k_1./Peticion+P3_k_1./Error+/P3_k_1./p_T0)

* T2 = p_T2.(P3_k_1.Error+P3_k_1./p_T1+/P2_k_1./Peticion./p_T1+
P2_k_1.Error./p_T1+/P2_k_1./p_T0./p_T1)

* T3 = P3_k_1.Error+p_T3.(P3_k_1./p_T1)

* T4 = P2_k_1./Error+p_T4.(P2_k_1.p_T1+P2_k_1./p_T2)

</SOLUTIONS>
```

4.4. Conclusiones

En este capítulo se abordaron 2 casos de estudio, en estos se aplicó el algoritmo presentado en el capítulo 2 utilizando su implementación en el módulo descrito en el capítulo 3. Para ambos casos de estudio se obtuvieron las primeras 3 fases del método de síntesis propuesto por Hietter.

Conclusiones

En esta tesis se ha abordado el problema de la síntesis de controladores lógicos a partir de especificaciones de alto nivel; en particular se trató con la ayuda computarizada para capturar estas especificaciones. Partiendo de un método de síntesis algebraica de controladores [Hie09], cuya entrada es un conjunto de ecuaciones booleanas, se propuso una técnica para automatizar la obtención de estas ecuaciones a partir de descripciones expresadas con redes de Petri interpretadas. Esta técnica de traducción es sistematizada por un algoritmo eficiente cuya complejidad es polinomial en el número de lugares de la red de Petri; esto constituye una ventaja evidente respecto a una técnica alternativa [PRCB94] la cual se basa en el tratamiento del grafo de alcanzabilidad del modelo a traducir. Un módulo de software fue desarrollado en base al algoritmo propuesto cumpliendo con dos requerimientos de compatibilidad:

- La salida del módulo de traducción es la entrada al software BESS [Rou10], el cual resuelve un sistema de ecuaciones con n incógnitas siguiendo el método de Hietter.
- El módulo recibe como entrada las estructuras proporcionadas por el editor gráfico de SPADES V-2 [Ara10] quedando implementado como un módulo más de esta herramienta.

Las pruebas realizadas del módulo desarrollado sobre diferentes casos de estudio han permitido por un lado, verificar su funcionamiento y por otro lado, constatar la utilidad de la traducción automatizada: ahorro en tiempo de diseño y eliminación de los errores de traducción. El trabajo presentado puede ser complementado con otras técnicas de traducción para otros formalismos, siendo el objetivo la obtención de ecuaciones booleanas, que constituyen la entrada al método de síntesis de controladores.

Referencias

[Ara10] A. K. Ríos Araujo. Una herramienta para el análisis de modelos en redes de petri. Master's thesis, Cinvestav Unidad Guadalajara, 2010.

[ASPLLS08] L. Aguirre-Salas, J. Pelayo-Lopez, E. Ortega-De Luna, and A. Santoyo Sánchez. Traducción de redes de petri interpretadas a lenguaje ensamblador para emulación de sistemas de eventos discretos. *6th. International Conference on Electrical and Electronics Engineering Research*, pages 399 – 405, 2008.

[BLKM06] S. Bogdan, F.L. Lewis, Z. Kovacic, and J. Mireles. *Manufacturing Systems Control Design Advances in Industrial Control.* Springer, 2006.

[BW00] E. Best and H. Wimmel. Reducing k -safe petri nets to pomset-equivalent 1-safe petri nets. *Application and Theory of Petri Nets 2000 Lecture Notes in Computer Science*, 1825/2000:63–82, 2000.

[CEPA+02] J.-M. Couvreur, E. Encrenaz, E. Paviot-Adet, D. Poitrenaud, and P.-A. Wacrenier. Data decision diagrams for petri net analysis. *APPLICATION AND THEORY OF PETRI NETS 2002 Lecture Notes in Computer Science*, 2360/2002:129–158, 2002.

[DDB07] G. Grey D. Dimitrova and I. Batchkova. Sequential control at the supervisory level of batch plant usin signal interpreted petri nets. In *International Conference AUTOMATICS AND INFORMATICS'07*, 2007.

[GHY03] E. Ioan Gergely, G. Husi, and S. Yildirim. Plc programs design using signal interpreted petri networks. *24th International Conference on Applications and Theory of Petri Nets, ICATPN 2003*, 2003.

[Gri98] R. P. Grimaldi. *Matemáticas discreta y combinatoria Una introducción con aplicaciones.* Pearson Prentice Hall, 1998.

[Hie09] Y. Hietter. *Synthèse algébrique de la loi de commande d'un système à évènements discrets logique.* PhD thesis, NORMALE SUPÉRIEURE DE CACHAN, Laboratoire Universitaire de Recherche en Production Automatisée, Mai 2009.

[HJML08] Y. Hietter, J.-M.Roussel, and J.-J. Lesage. Calcul des conditions de transition d'un réseau de petri par synthèse algébrique. In *Conférence Internationale Francophone d'Automatique, CIFA 2008, Bucarest, Roumanie*, Sep 2008.

[HRL08a] Y. Hietter, J.-M. Roussel, and J.-J. Lesage. Algebraic synthesis of dependable logic controllers. In *17th IFAC World Congress*, 2008.

[HRL08b] Y. Hietter, J.-M. Roussel, and J.-J. Lesage. Algebraic synthesis of transition conditions of a state model. *9th International Workshop on Discrete Event Systems, Göteborg, Sweden*, pages 187–192, May 2008.

[JLR01] I. Jiménez, E. López, and A. Ramírez. Synthesis of ladder diagrams from petri nets controller models. *IEEE International Symposium on Intelligent Control México City, Mexico.*, pages pp.225–230, Septiembre 2001.

[Joh05] R. Johnsonbaugh. *Matemáticas discretas.* Pearson Prentice Hall, 2005.

[Kam99] E.W. Kamen. Generation of boolean logic equations for discrete logic control. *7th IEEE International Conference on Emerging Technologies and Factory Automation*, 2:1541 – 1545, 1999.

[LSE07] S. Lohmann, O. Stursberg, and S. Engell. Comparison of event-triggered and cycle/driven models for verifying sfc programs. *Procedings of the 2007 American Control conference Marriott Marquis Hotel at Times Square*, July 2007.

[Med98] M.E. Meda. Identification using interpreted petri nets. *International Symposium of Robotics and Automation, Saltillo Coahuila, México*, pages 353–357, Diciembre 1998.

[Mel97] E. López Mellado. *Introducción a las redes de Petri.* Facultad de Ciencias Fisico-Matemáticas, Universidad Autónoma de Nuevo León, Octubre 1997.

[PRCB94] E. Pastor, O. Roig, J. Cortadella, and R. M. Badia. Petri net analysis using boolean manipulation. *Lecture Notes in Computer Science, Application and Theory of Petri Nets 1994*, 815/1994:416–435, 1994.

[PZ04] S. S. Peng and M. C. Zhou. Ladder diagram and petri-net-based discrete-event control design methods. *IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS PART C: APPLICATIONS AND REVIEWS*, 34(4), November 2004.

[RCP95] O. Roig, J. Cortadella, and E. Pastor. Verification of asynchronous circuits by bdd-based model checking of petri nets. *APPLICATION AND THEORY OF PETRI NETS 1995 Lecture Notes in Computer Science*, Volume 935/1995:374–391, 1995.

[Res98] G. Restrepo. *Los fundamentos de la matemática*. Editorial Universidad del Valle, 1998.

[Rou10] J.-M. Roussel. Etudes de cas traitées par synthèse algébrique. Technical report, LURPA,ENS Cachan, 25 janvier 2010.

[Sas99] T. Sasao. *Switching theory for logic synthesis*. Kluwer Acad. Publishers, 1999.

[Som05] I. Sommerville. *Ingeniería del Software*. Pearson Addison Wesley, séptima edición edition, 2005.

[Sot08] L. D. Murillo Soto. Redes de petri: Modelado e implementación de algoritmos para autómatas programables. *Tecnología en Marcha*, 21(4):102–125, Octubre-Diciembre 2008.

[TRM03] A. Ramirez Trevino, I. Rivera Rangel, and E. López Mellado. Observability of discrete event systems modeled by interpreted petri nets. *IEEE Transactions on Robotics and Automation*, 19 Issue:4:557 – 565, Aug. 2003.

[TSB02] A. Ramirez Trevino, L. Aguirre Salas, and O. Begovich. Diseño de observadores asintóticos para sistemas de eventos discretos basado en el modelo de error de estimación. *Computación y Sistemas*, ISNN 1405-5546:50–57, 2002.

CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS AVANZADOS DEL I.P.N.

UNIDAD GUADALAJARA

"2010, Año de la Patria, Bicentenario del Inicio de la Independencia y Centenario del Inicio de la Revolución"

El Jurado designado por la Unidad Guadalajara del Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional aprobó la tesis

Traducción de redes de Petri a ecuaciones booleanas para el diseño algebraico de controladores lógicos

del (la) C.

Miriam DÍAZ RODRÍGUEZ

el día 13 de Diciembre de 2010.

Dr. Luis Ernesto López Mellado
Investigador CINVESTAV 3B
CINVESTAV Unidad Guadalajara

Dr. Félix Francisco Ramos Corchado
Investigador CINVESTAV 3A
CINVESTAV Unidad Guadalajara

Dr. Antonio Ramírez Treviño
Investigador CINVESTAV 3A
CINVESTAV Unidad Guadalajara