xx(110517.1)

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

Síntesis de Controladores de Procedimientos para una Clase de Especificaciones de Seguridad utilizando Técnicas de Cálculo Incremental Explícito

Tesis que presenta
Juan José Venegas Moreno

Para obtener el grado de
Maestro en Ciencias

En la especialidad de
Ingeniería Eléctrica

Guadalajara, jal. Diciembre de 2002

Síntesis de Controladores de Procedimientos para una Clase de Especificaciones de Seguridad utilizando Técnicas de Cálculo Incremental Explícito

Tesis de Maestría en Ciencias
en Ingeniería Eléctrica

por:

Juan José Venegas Moreno

Ingeniero en Electrónica
Instituto Tecnológico de Ciudad Guzmán
1993 – 1998

Becado por CONACYT, Expediente No. **143962**

Director de Tesis:

Dr. Arturo del Sagrado Corazón Sánchez Carmona

CINVESTAV del IPN Unidad Guadalajara, Diciembre de 2002

Agradezco a todas las personas que me brindaron su apoyo y ayuda durante la realización de esta tesis. En especial a mi asesor de tesis Arturo del Sagrado Corazón Sánchez Carmona por su interés y colaboración. A mi familia y a mis compañeros de grupo.

Al CONACYT y al CINVESTAV por haberme otorgado el apoyo económico con el cual pude continuar mis estudios.

Índice general

Índice de cuadros

Índice de figuras

Capítulo 1

Introducción

1.1. Contenido del capítulo

En este capítulo se describen brevemente los sistemas de eventos discretos (SED), su existencia y caracterización, así como la problemática que motiva al desarrollo de nuevos métodos de síntesis de controladores de SEDs (Sección 1.2). Enseguida se muestra el estado del arte en el uso de herramientas de cálculo explícito y cálculo simbólico (Sección 1.3). Posteriormente se presentan las limitantes encontradas mediante el diagnostico hecho por Parra *et al.* (1999) sobre el método propuesto por Sánchez *et al.* (1999) enumeradas como; 1) Demanda de recursos computacionales, 2) Uso de especificaciones sin análisis previo, 3) Verificación del control vs. especificaciones. Conociendo las limitantes y retomando el método de síntesis de Sánchez *et al.* (1999) mejorado (Michel (2002)), se presenta el objetivo de este trabajo (Sección 1.4).

1.2. Ámbito del problema y motivación

Un sistema de eventos discretos (SED) es aquel en el cual la evolución de su estado depende de la ocurrencia de eventos asíncronos discretos en el tiempo (Cassandras and Lafortune, 1999). Algunos SED comunes son sistemas de manufactura, sistemas de tráfico, protocolos de comunicación y sistemas de monitoreo.

En este trabajo estamos interesados en estudiar la síntesis de una clase de controladores para SEDs aplicados a plantas de proceso (e.g. farmacéuticas, de alimentos). La operación de estas plantas comprende una gran cantidad de procedimientos secuenciales (e.g. paros, arranques, procedimientos de emergencia y operación). En plantas automatizadas estas actividades son ejecutadas por dispositivos de control con capacidades lógicas y numéricas, como controladores lógicos programables (PLCs) y sistemas de control distribuido (DCSs). Estos dispositivos de control han tenido una gran aceptación industrial, ya que se ha demostrado que contribuyen de manera importante en la mejora de los procesos productivos.

En la última década se han desarrollado diferentes métodos formales para la síntesis, verificación y validación de SEDs, los cuales se están incorporando en la práctica con resultados exitosos. Estos ayudan a garantizar que la lógica de control y su implementación realicen las especificaciones y estén libres de errores que comprometan la seguridad de la operación del sistema.

Existen muchas tecnologías para el control de SEDs como control supervisor, control lógico, control basado en redes, control óptimo, etc. También existen muchos formalismos para describir o modelar el comportamiento de los SEDs (e.g. autómatas, redes de Petri, ecuaciones booleanas, diagramas de tiempo). En este trabajo trataremos con la síntesis de una clase de controladores de eventos discretos identificados como *controladores de procedimientos* (CP), para un tipo de especificaciones, propuesto por Sánchez *et al.* (1999).

Sánchez *et al.* (1999) estudió la síntesis de controladores lógicos para la ejecución de procedimientos de operación que satisfacen propiedades de seguridad en donde se declaran: i) estados prohibidos, ii) secuencias de acciones que se ejecutan o se deben evitar como respuesta a un evento que ocurre en el proceso o un estado que se alcanza durante la operación. A las cuales en este trabajo nos referimos como *especificaciones de seguridad.* Basados en técnicas de especificación y control para sistemas a eventos discretos secuenciales modelados como *máquinas de estados finitos* (MEF) (ver definición 2.2.1) que generan trayectorias finitas (Ramadge and Wonham, 1987b), Sánchez *et al.* (1999) propusieron un tipo de controlador que recibe el nombre de

controlador de procedimientos. Este mecanismo emite de manera secuencial únicamente una acción de control en respuesta a un conjunto de señales producidas por el proceso.

La base teórica para el estudio de estos sistemas se identifica como Teoría de Control de Procedimientos (TCP). TCP garantiza el comportamiento controlable de un sistema dado y la existencia de un controlador de forma tal que se satisfagan mínimamente un conjunto de especificaciones de seguridad. Para la síntesis de estos controladores se propuso un método iterativo (Sanchez *et al.*, 1999) que incluye:

- El modelado del proceso a lazo abierto y especificaciones de seguridad que capturan el comportamiento deseado a lazo cerrado (especificaciones) utilizando máquinas de estados finitos (Sanchez, 1996).

- Procedimientos de síntesis del controlador.

El método se probó con casos a nivel industrial dando buenos resultados (Alsop *et al.*, 1996; Sanchez *et al.*, 2001).

En lo que respecta al cálculo con SEDs basados en Autómatas existen 2 vertientes que se discuten en la siguiente sección:

1. Cálculo explícito: consiste en obtener los resultados a través de operaciones explícitas.

2. Cálculo simbólico: consiste en obtener los resultados a través de la ejecución simbólica de operaciones.

1.3. Estado del arte

Numerosos métodos y técnicas para la síntesis de controladores de SEDs han aparecido en la literatura. A continuación se comenta la bibliografía que se conoce de los métodos de síntesis implementados bajo la noción de control supervisor y control de procedimientos.

1.3.1. Control supervisor

Ramadge y Wonham (1987a) plantean un marco de trabajo para la síntesis de controladores de SEDs y la implementación mediante técnicas de cálculo explícito, llamado teoría del control supervisor (TCS). El objetivo es diseñar un controlador que ejecute patrones de control obedeciendo a varias restricciones (tal controlador es llamado supervisor), a partir de un modelo abstracto del proceso a controlar y un lenguaje de especificación. Para ilustrar la utilidad de este marco teórico, se muestra solo la solución de ejemplos simples, es decir, no muestran ejemplos prácticos de aplicación real.

La implementación de los algoritmos porpuestos en TCS se basa en operadores de punto fijo. Estos se encuentran en el software de dominio público conocido como TCT.

Hoffmann y Wong-Toi (1992) realizaron una implementación mas eficiente de los algoritmos propuestos por Ramadge y Wonham (1987a) y demostraron su uso en una aplicación real. Esta implementación está basada en una estructura de datos conocida como Diagramas de Decisión Binaria (DDBs) (Bryan, 1986). Los DDBs permiten implementar la representación de funciones booleanas en una forma efectiva y compacta. La implementación consiste en representar tanto la planta como las especificaciones como funciones booleanas. Para mostrar la importancia práctica así como la factibilidad del método de síntesis, se sintetiza un controlador para una aplicación real con un tamaño de proceso 4.2 (10^8) estados. Los cálculos se realizaron en una estación DEC 5000/133, con los siguientes resultados; 1,235 segundos en tiempo de CPU, 7.8 MB de memoria y 7932 estados del tamaño del controlador.

De igual manera Balemi (1992), realizó la implementación de los algoritmos propuestos por Ramadge y Wonham (1987a) y presentó la síntesis para un sistema de proceso real (multiprocesador térmico), con un tamaño de modelo de 2.3 (10^6) estados. Los cálculos se realizaron en una estación DEC 3100, con los siguientes resultados; 160 segundos en tiempo de CPU, y 4.4 MB de memoria.

1.3.2. Control de procedimientos

Sánchez *et al.* (1999) plantea un marco de trabajo para la síntesis de controladores lógicos de SEDs, basado en técnicas de especificación y control para sistemas de eventos discretos secuenciales modelados como máquinas de estados finitos (MEF) que generan trayectorias finitas (Ramadge and Wonham, 1987b). Este abarca técnicas para el modelado discreto del comportamiento del proceso en lazo abierto y especificaciones de operación, así como para la síntesis de un mecanismo de control en retroalimentación llamado controlador de procedimientos. Con base en este formalismo matemático, Sánchez *et al.* (1999) también presenta un método de síntesis.

La aplicabilidad de este marco de trabajo a sistemas de complejidad industrial se muestra mediante la síntesis de un CP, y la implementación de éste a código de control para una planta piloto multiproceso de operación automática.

El método de síntesis que plantea Sánchez *et al.* (1999) requiere de la construcción de un modelo del proceso como MEF, lo cual computacionalmente resulta muy costoso.

Mouillé (1997) plantea un método de síntesis de CP mediante cálculo explícito. La principal característica es que no requiere la construcción completa del modelo del proceso. Este construye la máxima superestructura (conjunto de controladores de procedimientos del proceso) de manera incremental, facilitando el cálculo del controlador deseado y mejorando la capacidad de síntesis, con respecto a Sánchez *et al.* (1999). Para ilustrar el funcionamiento del método, se muestra la síntesis de control para tres casos de estudio. El sistema de mayor tamaño utilizado consta de 186,624 estados. Los cálculos se realizaron en una estación de trabajo SUN Ultra, con los siguientes resultados: 30 segundos en tiempo de CPU, 6.5 MB de memoria y 30 estados del tamaño del controlador. Sin embargo, al parecer este método tiene errores en el planteamiento de sus algoritmos, dado que no es capaz de sintetizar el controlador que satisface los requerimientos establecidos en el ejemplo motivacional utilizado en Sánchez *et al.* (1999).

1.3.3. Control de SEDS en general

Asarin *et al.* (1994) plantea algoritmos para la síntesis de controladores discretos, en general, utilizando herramientas de cálculo simbólico. La eficiencia de la implementación simbólica, se debe principalmente a la representación de la estructura de estado–transición del sistema como un conjunto de ecuaciones booleanas.

El problema de síntesis se plantea como un problema en teoría de juegos, en donde la solución del problema se define como el encontrar una estrategia ganadora. No se muestran resultados que permitan comparar la capacidad de síntesis.

Tsitsiklis (1989) demuestra que la complejidad de los algoritmos que resuelven el problema del control supervisor mediante el uso de autómatas, en el mejor de los casos será polinomial. Para dicha demostración, plantea que un SED es muy similar a una cadena de Markov discreta en el tiempo, excepto que en la estructura de estado-transición de un SED no se suponen probabilidades.

1.4. Definición del problema y objetivo de la tesis

Como se mencionó en la introducción de este capítulo, Parra *et al.* (1999) realizó un diagnóstico al método de Sánchez *et al.* (1999), determinando las problemáticas siguientes:

1. Demanda de recursos computacionales. Los productos asíncronos entre MEFs y sincronización de trayectorias (o producto síncrono entre MEFs) se utilizan de manera extensiva. Estos algoritmos son de complejidad polinomial en el número de estados y debido a que el tamaño de las MEFs tanto del modelo como de las especificaciones puede crecer con facilidad, el tiempo de cómputo y requerimientos de espacio de memoria pueden ser muy grandes.

2. Uso de conjuntos de especificaciones sin análisis previo. Las especificaciones que declaran secuencias de comportamiento se modelan como un conjunto de MEFs, cada una describiendo parte del comportamiento deseado. El que este conjunto de especificaciones capture correctamente el comportamiento deseado

se deja como responsabilidad de la persona que realiza la síntesis del controlador. Esto trae como consecuencia que con frecuencia existan especificaciones duplicadas. También puede ocurrir que las especificaciones establezcan comportamiento contradictorio. En el primer caso se aumenta innecesariamente el tiempo y consumo de recursos de cómputo, mientras que las especificaciones contradictorias provocan que el comportamiento posible bajo control no sea de interés práctico y en muchas ocasiones se obtenga el conjunto vacío.

3. Verificación del controlador vs. especificaciones. Con frecuencia el comportamiento a lazo cerrado no incluye todas las trayectorias declaradas por las especificaciones ya que éstas pueden ser no controlables. La identificación de las especificaciones no satisfechas se realiza de manera manual una vez que se ha sintetizado el controlador.

El método originalmente propuesto por Sánchez *et al.* (1999) tiene como características principales el modelar las especificaciones de seguridad dinámicas como MEFs y realizar todos los cálculos requeridos con implementaciones explícitas.

Parra *et al.* (1999) presenta una mejora a este método proponiendo un tipo de especificaciones mas compacta con el fin de trabajar con modelos mas pequeños, atacando de esta manera la primera de las 3 limitantes diagnosticadas. Esta propuesta consiste en modelar las especificaciones como secuencias de asignaciones en vez de MEFs como se propuso en el método original. A estas secuencias de asignación nos referiremos como *semitrayectorias* y se definen de manera formal en el capítulo 2.

Michel (2002) llevo a cabo la implementación de esta propuesta de modelos de especificaciones mas compactas retomando el método original (Sanchez *et al.*, 1999) con el objetivo de atacar las limitantes identificadas. El método se muestra en la figura 1.1 y constituye los pasos siguientes:

1. Cálculo del modelo del proceso a lazo abierto.

   a) Producto asíncrono de los componentes elementales.

   b) Eliminación de estados prohibidos para el proceso de la máquina resultante del paso 1*a*.

   c) Producto síncrono entre el comportamiento causal y la máquina resultante del paso 1*b*.

2. Modelado de especificaciones del comportamiento deseado a lazo cerrado.

   a) Traducción de estados prohibidos para el controlador a estados de MEFs.

   b) Traducción de especificaciones dinámicas para el controlador a MEFs.

3. Síntesis del controlador. Este paso se divide en dos partes:

   a) Cálculo de la máxima superestructura de controladores.

      1) Eliminación de estados prohibidos para el controlador al modelo resultante del paso 1.

      2) Producto síncrono entre especificaciones dinámicas para el controlador y la máquina resultante del paso 3*a*1.

      3) Aplicación del algoritmo de la máxima superestructura al resultado del paso 3*a*2.

      4) Asear la máquina resultante del paso 3*a*3.

   b) Síntesis del controlador.

      1) Aplicación del algoritmo para el cálculo del controlador a la máquina resultante del paso 3*a*.

      2) Asear la máquina resultante del paso 3*b*1.

En este trabajo retomamos el método de Sánchez *et al.* (1999) mejorado (Michel (2002)), con el fin de reducir el gasto sobre la primera limitante diagnosticada. Con este fin se propone implementar un algoritmo de síntesis utilizando *cálculo incremental explícito*. Consiste en *crear* y *analizar* de manera secuencial y en profundidad los estados posibles del controlador, uno por uno.

Primeramente se crea el estado físicamente posible. Luego se establece si éste es candidato a formar parte del controlador. La síntesis se hace considerando el tipo de especificaciones de seguridad dadas en Sánchez *et al.* (1999), las cuales se modelan mediante el concepto de semitrayectorias propuesto por Parra *et al.* (1999).

Cuando se hace la síntesis de controladores, es práctica común la construcción de un modelo de proceso, lo cual computacionalmente resulta ser muy costoso. El algoritmo que en esta tesis se propone no requiere de la construcción de un modelo. Este calcula un controlador de procedimientos sobre la marcha, utilizando una estrategia recursiva en profundidad.

Con la implementación del algoritmo que aquí se propone, obtenemos la evolución del método de Michel (2002) como se ilustra en la figura 1.2 y constituye los pasos siguientes:

1. Modelado.

   a) Traducción de modelos de componentes elementales a MEFs.

   b) Traducción de especificaciones a MEFs.

2. Análisis de consistencia, duplicidad y controlabilidad de especificaciones.

3. Síntesis del controlador.

   a) Aplicación del algoritmo para el cálculo directo del controlador después del paso 1*a* y 1*b*.

Como se puede observar es notable la evolución del método de Michel (2002), ya que este nuevo planteamiento no requiere de la construcción de un modelo del proceso. Esto se refleja en una exitosa mejora sobre la primera limitante diagnosticada.

1.5. Contenido de la tesis

Este trabajo está organizado en seis capítulos. Este primer capítulo presenta el ámbito sobre el cual estamos trabajando así como el estado del arte. Se describen los problemas diagnosticados sobre el método de síntesis de Sánchez *et al.* (1999), con la finalidad de mostrar y encaminar el objetivo de esta investigación.

El capítulo 2 contiene los conceptos básicos del marco teórico de modelado de sistemas sobre el cual trabajaremos, así como la formalización del concepto de semitrayectoria, definido en Michel (2002). También muestra los conceptos básicos del tipo de controlador en el cual estamos interesados.

El capítulo 3 se dedica a la exposición del algoritmo que se propone en esta tesis para ser implementado en el método de síntesis de controladores de procedimientos (CP) propuesto por Sánchez *et al.* (1999), mejorado por Parra *et al.* (1999) y por último mejorado por Michel (2002). El primero sentó las bases para el modelado y cálculo de controladores de procedimientos utilizando MEFs. El segundo mejoró este método mediante la propuesta de modelado de las especificaciones utilizando semitrayectorias. Y el tercero a su vez mejoró el método propuesto por Parra *et al.* (1999), agregando el análisis de especificaciones y la verificación de satisfacción de éstas en el controlador.

Nuestro trabajo consistió en mejorar el método propuesto por Michel (2002) implementando un algoritmo de síntesis más eficiente.

El capítulo 4 ilustra el uso y funcionamiento del método y el algoritmo implementado, tomando como ejemplo un proceso industrial hipotético. También muestra los resultados de los experimentos realizados para evaluar el desempeño del algoritmo. Las variables de interés que se midieron son: tiempo de síntesis, memoria y número de ejecuciones recursivas, en función del tamaño de proceso. Los métodos de cálculo explícito actuales son capaces de resolver sistemas de proceso con espacios de estados en el orden de 10^4. Con el algoritmo incremental que en este trabajo de tesis se propone se logró la resolución de sistemas con espacios de estados en el orden de 10^7

Finalmente se presenta un esquema comparativo entre los resultados obtenidos con este algoritmo incremental y los alcanzados mediante el método Michel (2002), así como las curvas que describen la complejidad del algoritmo para cada una de las variables de interés mencionadas.

En el capítulo 5 se presentan tanto las características del nuevo algoritmo, como la descripción del trabajo futuro que cubrirá las desventajas hasta ahora identificadas.

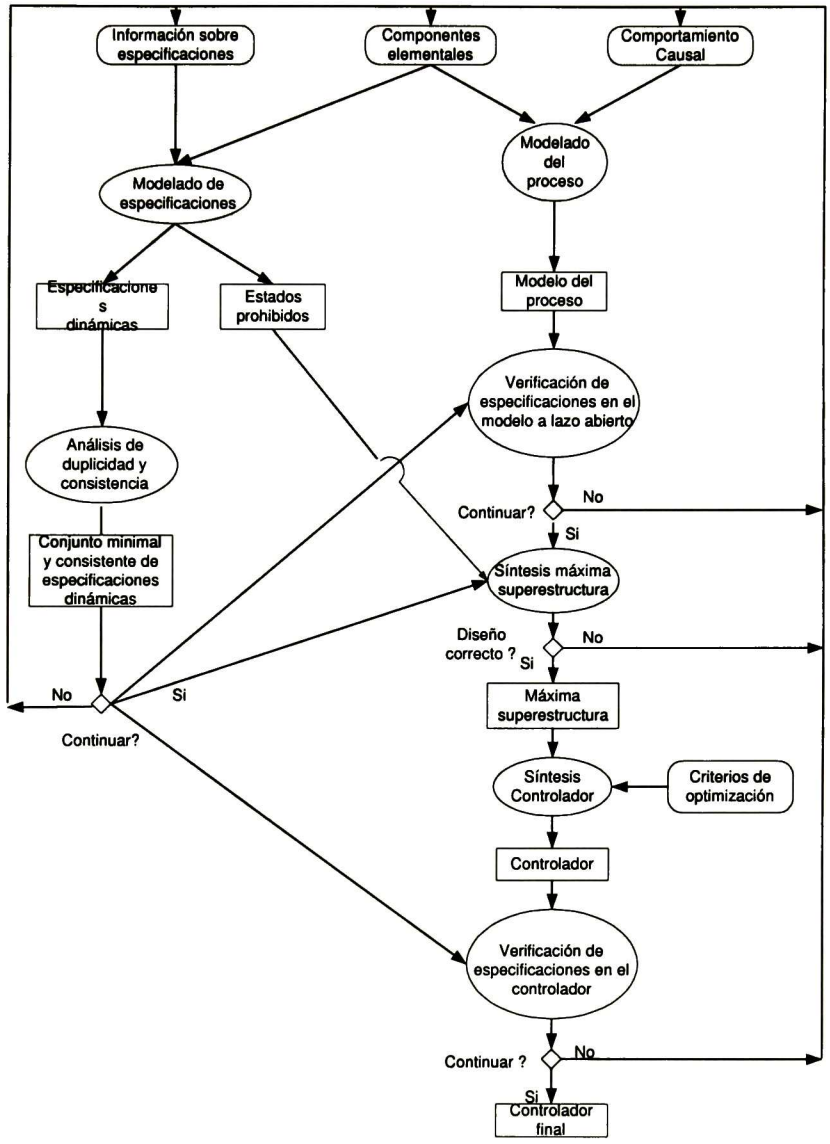

Figura 1.1: Método de síntesis antes de la implementación del algoritmo incremental

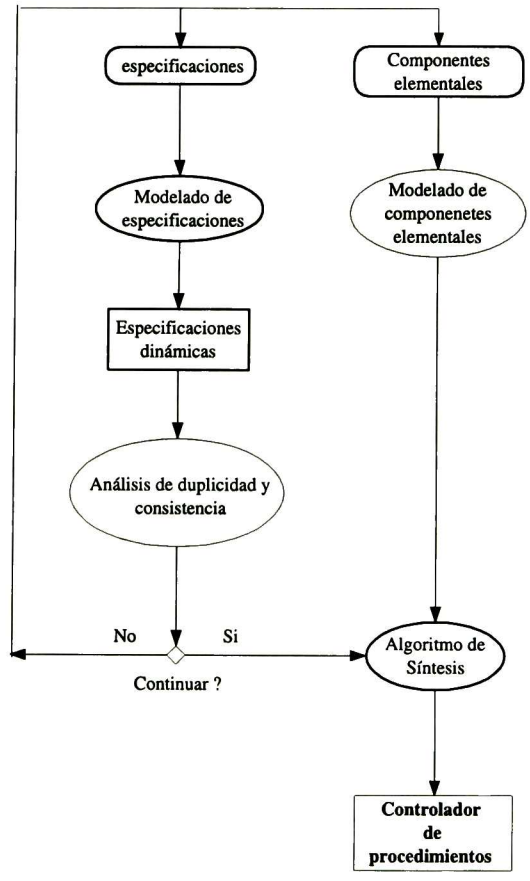

Figura 1.2: Método de síntesis con algoritmo incremental

Capítulo 2

Conceptos básicos

2.1. Contenido del capítulo

En este capítulo se presenta las máquinas de estados finitos (MEFs) (Ramadge and Wonham, 1987b) como herramienta para el modelado de SEDs y algunas propiedades relacionadas. Esta herramienta se utiliza para modelar los componentes elementales del proceso (botones, válvulas, sensores etc.) y las especificaciones dinámicas como MEFs (Sección 2.2). Posteriormente se introduce el concepto formal de semitrayectoria, tomado de Michel (2002) (Sección 2.3). Enseguida se muestran los tipos de especificaciones (especificaciones de seguridad) a considerar para el proceso de síntesis, así como la representación formal de éstas como semitrayectoria (Sección 2.4). Por último se presentan las bases involucradas en la teoría del tipo de control que utilizaremos en este trabajo (i.e. el controlador de procedimientos) y las propiedades relacionadas (Sección 2.5).

2.2. Marco de modelado

Como herramienta formal para el modelado de SEDs, en este trabajo hacemos uso de las máquinas de estados finitos (MEF). Una máquina de estados finitos o autómata finito es un modelo matemático de un sistema, con entradas y salidas discretas.

Definición 2.2.1. Una **máquina de estados finitos** $P = \{Q, V, \Sigma, \delta, \zeta, q_0, Q_m\}$

(MEF) (Sanchez *et al.*, 1999) donde cada estado es etiquetado, es una estructura estado-transición con un estado inicial definido, estados marcados y relación entre estados y transiciones dada por una función parcial. Donde:

Q es el conjunto de estados.

V es el conjunto de variables de estado. $V = \{v_1, v_2, ..., v_n\}$, donde n es el número de variables de estado.

Σ es el conjunto de transiciones.

δ es la función estado–transición, definida como una función parcial $\delta : \Sigma \times Q \longrightarrow Q$.
ζ es la función transición–variable de estado, definida como una función parcial $\zeta : \Sigma \times V \longrightarrow V$

q_0 es el estado inicial.

Q_m Conjunto de estados marcados.

Una variable de estado (v_j), $j = 1 \ldots n$, describe el estado de un actor dentro del proceso (e.g. la posición de una válvula solenoide, el estado de una bomba) y es caracterizado por un dominio de posibles valores D_j (e.g. $\{abierto, cerrado\}$). Se asume que un sistema es descrito totalmente por el conjunto de variables V.

Definición 2.2.2. Una **asignación** al conjunto de variables del sistema V, es dada mediante la función s: $V \longrightarrow D$, donde D es la unión de los dominios respectivos de las variables.

A cada estado q se le asocia por una n-ada ordenada $(s(v_1), s(v_2), ..., s(v_n))$ por medio de la función β: $Q \longrightarrow D$, donde $s(v_i)$ es el valor asignado por s a v_i para cada $i = 0, 1, ..., n$. Por ejemplo, considere un sistema compuesto de una válvula solenoide.

La variable de estado de este componente describe la "posición de la válvula" y su dominio es $\{abierto, cerrado\}$. Por lo tanto, el modelo de este sistema tiene en total dos estados: (abierto) y (cerrado). Así, un modelo de dos válvulas solenoides considerando todas las posibles combinaciones está compuesto de cuatro estados con dos variables de estado cada uno. Esto es: (abierto, abierto), (abierto, cerrado), (cerrado, abierto) y (cerrado, cerrado).

En adición a los valores del dominio, un símbolo más puede ser asignado a una variable de estado: ∞_j, que simboliza todos los valores posibles que puede tomar la variable de estado v_j. Este símbolo es introducido para el manejo de información incompleta en el modelado del proceso o durante la fase de especificación.

Las transiciones son instantáneas que relacionan un estado fuente con un estado destino y cambia sólo el valor de una variable de estado. Σ es el conjunto de las transiciones del proceso y Σ^* es el conjunto de todas las trayectorias finitas del proceso compuesto de transiciones en Σ. La función parcial $\delta : \Sigma \times Q \longrightarrow Q$ define la relación entre estados y transiciones, mientras que la función parcial $\zeta : \Sigma \times V \longrightarrow V$ hace lo mismo para variables de estado y transiciones.

El estado inicial q_0 indica el estado a partir del cual inicia la operación del sistema. El conjunto de estados marcados Q_m distingue los estados de importancia especial para el sistema, por ejemplo, aquellos donde el funcionamiento puede ser detenido o una tarea ha sido completada pueden ser considerados como marcados.

Los *componentes elementales* describen el comportamiento de cada uno de los actores que intervienen en el funcionamiento de un sistema (e.g. válvulas, dispositivos de medición, interruptores) y son modelados como MEFs. Sólo una variable de estado es asociada con cada componente elemental (e.g. "posición de la válvula", "estado del nivel"). Una transición siempre es asociada con un cambio en el valor de la variable de estado (e.g. si el valor actual de la variable de estado del componente que representa la "posición de la válvula" es "abierto" y ocurre la transición "cerrar válvula", en el próximo estado del modelo el valor de la variable de estado será "cerrado"). Así, en un modelo más complejo generado a partir de varios componentes elementales, la función parcial $\zeta : \Sigma \times V \longrightarrow V$ define cuál variable de estado es cambiada bajo la ejecución de una transición dada.

Definición 2.2.3. La asignación $s(v_i)$ **cubre** a la asignación $s'(v_i)$ ($s'(v_i)$ refina a $s(v_i)$) si y solo si $s(v_i) = \infty$ y $s'(v_i)$ toma un valor del dominio pero diferente a ∞.

La asignación s **cubre** a la asignación s' (s' **refina a** s) si y solo si existe al menos un $j \in \{1, 2, \ldots, n\}$ tal que $s(v_j) = \infty$, $s'(v_j) \in d_j / \infty$ y $s(v_k) = s'(v_k)$ para k $\neq$ j, donde n es el número de variables de estado y d_j es el dominio asociado a la variable de estado v_j.

Definición 2.2.4. Dos estados q y q' son **iguales** si y solo si $\beta(q) = \beta(q')$ y $s(v_i) \neq \infty$, $s'(v_i) \neq \infty$ para todo $i \in \{1 \ldots n\}$.

Definición 2.2.5. Dos estados q y q' son **equivalentes** si $\beta(q)$ y $\beta(q)'$ son iguales, o si $\beta(q)$ cubre o refina a $\beta(q)'$

Definición 2.2.6 (Lenguaje). Un lenguaje definido sobre un conjunto de transiciones Σ es un conjunto de cadenas s de longitud finita formadas por transiciones en Σ.

Un SED es un generador de un lenguaje formal, definido sobre un alfabeto Σ. Los lenguajes asociados a un SED $G = \{Q, V, \Sigma, \delta, q_0, Q_m\}$ son *lenguaje generado* y *lenguaje marcado.*

Definición 2.2.7 (Lenguaje generado). El lenguaje generado de G es aquel que representa la posible evolución del sistema, y se denota por $L(G)$ donde:

$$L(G) := \{s \in \Sigma | \delta(q_0, s)!)$$

Definición 2.2.8 (Lenguaje marcado). El lenguaje marcado de G, es aquel que representa la evolución correspondiente a la realización de ciertas tareas. Se denota por $L_m(G)$ donde:

$$L_m(G) := \{s | \delta(q_0, s) \in Q_m\}.$$

2.2.1. Propiedades importantes de una MEF candidato a controlador

Definición 2.2.9 (Alcanzabilidad). Una MEF es alcanzable si del estado inicial es posible alcanzar todos los estados de la MEF. Es decir

$$\forall q \in Q, \exists s \in \Sigma^* \mid \delta(q_0, s) = q$$

Definición 2.2.10 (Coalcanzabilidad). Una MEF es coalcanzable si de cada estado es posible llegar a un estado marcado. Es decir

$$\forall q \in Q, \exists s \in \Sigma^* \mid \delta(q, s) \in Q_m$$

Definición 2.2.11 (No bloqueo). Una MEF es no bloqueante si y solo si cada estado alcanzable es coalcanzable, i.e. $\overline{L_m}(G) = L(G)$.

2.2.2. Modelado de componentes elementales como MEF

Un sistema de proceso está constituido por componentes elementales, como son; válvulas, botones, sensores, etc. cada uno de los cuales es codificado con un modelo en MEF. Es decir los modelos de componentes elementales son generados individualmente para cada una de las partes del proceso (botones, válvulas, sensores, etc.). Sólo una variable de estado es asociada con cada modelo. Por ejemplo: posición de la válvula, estado del sensor, etc. Una transición siempre es asociada con un cambio en el valor de la variable de estado. Por ejemplo, el cambio de la válvula de cerrado a abierto. La figura 2.1 muestra el modelo (MEF) de una válvula de dos posiciones (cerrado/abierto).

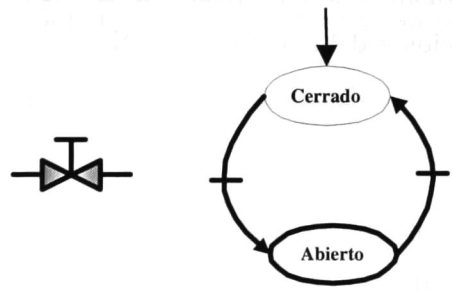

Figura 2.1: Modelo de una válvula de dos posiciones (cerrado-abierto)

2.3. Semitrayectorias

Como se mencionó en el capítulo 1, Michel (2002) formalizó el concepto de semitrayectoria que se proporciona a continuación y se utilizará para modelar los tipos de especificaciones de seguridad que se describen en la sección 2.4.

2.3.1. Sintaxis

El lenguaje de las expresiones adoptadas para construir semitrayectorias está constituido de las siguientes componentes:

- Variable de estado v_i. Representa el estatus de la componente elemental i de un sistema. Al igual que las variables de estado en las MEFs, estas variables toman valores de un dominio definido por la naturaleza del componente e incluye el símbolo ∞, que denota "valor no definido" Por ejemplo, una variable de estado que describe la posición de una válvula solenoide toma valores de un dominio compuesto por {abierta, cerrada, ∞}.

- Asignación s. Al igual que la asignación en una MEF, la asignación de una semitrayectoria es una función $s : V \rightarrow D$, donde V es el conjunto de variables del sistema , $V = \{v_1, v_2, \ldots, v_n\}$, D es la unión de los dominios respectivos de las variables y n es el número de variables que describen un estado del sistema. El estado q se representa mediante una n-ada ordenada $(s(v_1), s(v_2), \ldots, s(v_n))$, donde $s(v_i)$ es el valor asignado por s a v_i, para cada $i = 0, 1, \ldots, n$.

- Transición τ. Representa un evento que realiza el cambio de valor en una variable de estado. Toda transición τ debe pertenecer a Σ.

- Conectivos. $\rightarrow$, $\neg$ y $\wedge$.

- Símbolos. "(", ")" y ",".

2.3.2. Tipo de expresiones

Los tipos de expresiones definidas para el modelado de especificaciones como semitrayectorias son:

1. Transiciones a partir de un estado de operación. Se realiza la ejecución o la no ejecución de transiciones a partir de una asignación.

   a) Secuencias de transiciones a partir de una asignación:
   $(s(v_1), s(v_2), \ldots, s(v_n)) \rightarrow \tau_1 \rightarrow \tau_2 \rightarrow \ldots \rightarrow \tau_k$, donde k es el número total de transiciones involucradas en la semitrayectoria.

   b) Transiciones no ejecutables a partir de una asignación:
   $(s(v_1), s(v_2), \ldots, s(v_n)) \rightarrow \neg\tau_1 \wedge \neg\tau_2 \wedge \ldots \wedge \neg\tau_k$, donde k es el número total de transiciones involucradas en la semitrayectoria.

2. Trayectorias a partir de la ocurrencia de un evento. En vez de realizarse la ejecución de transiciones a partir de un estado, el mecanismo de inicio es la ocurrencia de una transición a partir de una asignación:
   $(s(v_1), s(v_2), \ldots, s(v_n)) \wedge \tau \rightarrow \tau_1 \rightarrow \tau_2 \rightarrow \ldots \rightarrow \tau_k$, donde k es el número total de transiciones involucradas en la semitrayectoria.

2.4. Especificaciones de seguridad

Informalmente una especificación de seguridad establece que algo malo no debe suceder en el proceso.

En nuestra experiencia con plantas de proceso, una parte considerable de los comportamientos descritos en procedimientos de operación utilizados como especificaciones para el diseño de controladores de eventos discretos, pueden describirse en términos de estados prohibidos y las siguientes tres situaciones:

1. Dado un estado de operación, se prohíbe la ejecución de un conjunto de comandos de control.

2. Dado un estado de operación, un conjunto de comandos de control es ejecutado secuencialmente.

3. Dado un estado de operación y la ocurrencia de un evento de proceso, un conjunto de comandos de control es ejecutado secuencialmente.

Tipo de especificacin	*Sintaxis*
1	[(q)] *implic* $\neg\tau_1$ *y* $\neg\tau_2$ ... *y* $\neg\tau_n$
2	[(q)] *implic* [τ_1] *implic* [τ_2] ... *implic* [τ_n]
3	[(q) *y* τ] *implic* [τ_1] *implic* [τ_2] ... *implic* [τ_n]

Cuadro 2.1: Sintaxis correspondiente a los 3 tipos de especificaciones.

La formalización de estos tres tipos de especificaciones de seguridad como semitrayectorias se definen a continuación.

Definición 2.4.1 (Especificación de seguridad tipo 1 como Semitrayectoria). Una semitrayectoria de tipo 1 es una semitrayectoria con dos asignaciones de la forma $\pi = s_0^{\tau_1} s_1$, donde s_0 y s_1 son asignaciones fijas, mientras que τ_1 puede ser cualquier transición tal que $\gamma(s_0, \tau_1) = s_1$ y τ_1 pertenece al conjunto $net(s_0)$. s_0 es la asignación inicial y $net()$ representa un conjunto de transiciones no deseadas o fisicamente no posibles a partir de s_0. Este tipo de semitrayectoria se escribirá como $\pi = s_0^{\vee\tau} s_1$

Definición 2.4.2 (Especificación de seguridad tipo 2 como Semitrayectoria). La semitrayectoria $\pi = s_0^{\tau_1} s_1^{\tau_2} \ldots^{\tau_m} s_m$ de longitud $m+1$ con $m > 0$ modela una especificación tipo 2. s_0 representa la *asignación inicial* de π.

Definición 2.4.3 (Especificación de seguridad tipo 3 como semitrayectoria). La semitrayectoria $\pi = s_0^{\tau_1} s_1^{\tau_2} \ldots^{\tau_m} s_m$ de longitud $m+1$ con $m > 0$ modela una especificación de tipo 3 si y solo si s_0 cubre s_1. $\gamma(s_0, \tau)$ se utiliza como la asignación inicial s_1 de donde el resto de π es ejecutado. Esta semitrayectoria es una manera compacta de representar la activación con una transición de una sucesión. Este tipo de semitrayectoria se escribirá como $\pi = s_0^{\wedge\tau_1} s_1^{\tau_2} s_2^{\tau_3} \ldots s_m$, donde la expresión $\wedge\tau_1$ indica el énfasis sobre la ejecución de eventos.

En la tabla 2.1 se muestra la sintaxis correspondiente a los 3 tipos de especificaciones de seguridad que utilizaremos para la captura del comportamiento deseado. El signo $\neg$ representa la no ejecución de la transición.

2.4.1. Consistencia entre especificaciones

En los sistemas que manejamos es común que el comportamiento que se desea especificar esté representado por más de una semitrayectoria, de manera que es posible

que entre estas surjan errores y duplicaciones que en ocasiones no es tan simple apreciar, ocasionando errores en el proceso de síntesis. Por lo tanto es deseable tener un conjunto de especificaciones libre de errores y duplicaciones.

Por ejemplo, si tomamos el caso de una especificación que declara la ejecución de una transición a partir de un estado dado, en tanto que otra especificación declara su no ejecución o la ejecución de una transición diferente, tendremos como consecuencia la obtención de controladores que no resultan de interés práctico. Por lo anterior es necesario crear conjuntos de especificaciones consistentes.

La duplicación de trayectorias entre dos o más especificaciones no trae consecuencias en el controlador resultante, pero se consumen recursos computacionales, por lo tanto es deseable contar con un conjunto minimal consistente de especificaciones.

Las definiciones que se darán a continuación fueron tomadas de Michel (2002) y nos ayudan a obtener un conjunto minimal consistente de especificaciones antes de iniciar la síntesis del controlador.

Definición 2.4.4. Dos especificaciones son **consistentes** si y solo si se cumple solo uno de los siguientes enunciados:

1. No comparten estados equivalentes a lo largo de sus secuencias.
2. Si comparten estados equivalentes que implican la ejecución de una transición, entonces esta debe ser la misma en dichos estados.
3. Si comparten estados equivalentes en el que uno de ellos implica la no ejecución de un conjunto de transiciones negadas y el otro implica la ejecución de una transición entonces dicha transición no debe aparecer en el conjunto de transiciones negadas.

Definición 2.4.5. Dos especificaciones son **duplicadas** si y solo si comparten estados equivalentes que:

1. Implican la ejecución de la misma trayectoria ó
2. Implican conjunciones de transiciones en donde por lo menos una de las transiciones de la conjunción implicada por uno de los estados está contenida en la conjunción del otro estado.

Definición 2.4.6. Un **conjunto consistente de especificaciones**, es aquel en el que todas las especificaciones son consistentes entre sí.

Definición 2.4.7. Un **conjunto minimal consistente de especificaciones**, es aquel en el que no existen especificaciones duplicadas y todas son consistentes entre sí.

2.4.2. Especificaciones de modelo y de control

Las especificaciones de seguridad se dividen en especificaciones de modelo o de proceso y especificaciones de control.

El marco formal de especificaciones de seguridad descrito en esta sección es utilizado para modelar tanto el comportamiento causal del proceso (aquel comportamiento definido por la naturaleza del proceso), como el comportamiento deseado a lazo cerrado (ejecución condicional de una secuencia de transiciones controlables).

Especificaciones que describen el comportamiento causal se identificarán con el término "especificaciones de modelo o de proceso", mientras aquellas que describen el comportamiento deseado a lazo cerrado se denominarán "especificaciones de control"

Por ejemplo; consideremos el llenado de un tanque a través de una válvula:

- Una especificación de modelo sería:

  El nivel del tanque no puede aumentar si la válvula de llenado está cerrada.

- Una especificación de control sería:

  Una vez que el tanque alcance un determinado nivel, entonces cerrar la válvula.

Nótese que las especificaciones de seguridad se componen por tres tipos, a cada una de las cuales corresponde un tipo de semitrayectoria. Estos tres tipos de especificaciones se utilizan tanto para especificaciones de modelo como para especificaciones de control, por lo tanto se identificará cada categoría con la siguiente nomenclatura.

1. Especificaciones de modelo

   - tipo 1 "1M"

- tipo 2 "2M"
- tipo 3 "3M"

2. Especificaciones de Control

   - tipo 1 "1C"
   - tipo 2 "2C"
   - tipo 3 "3C"

2.5. Control de procedimientos

Sánchez *et al.* (1999) introdujo el dispositivo de control que utilizaremos en este trabajo, llamado controlador de procedimientos. La base teórica del control de procedimientos se fundamenta en la teoría de control supervisor.

Definición 2.5.1 (CP). Un controlador de procedimientos modelado como MEF es $C = \{X, \Sigma, \gamma, x_0, X_m\}$, donde:

- X es el conjunto de estados.
- Σ es el conjunto de transiciones, el mismo que para el modelo del proceso.
- $\gamma : \Sigma \times X \rightarrow X$, es la función parcial de estado transición.
- x_0 es el estado inicial
- X_m es el conjunto de estados marcados

En particular, para cada $x \in X$, $\sigma \in \Sigma$ tal que $\gamma(\sigma, x)$ está definida, uno de los puntos siguientes es verdadero:

1. $\sigma \in \Sigma_u$
2. $\sigma \in \Sigma_c$ y para toda $(\forall \sigma' \neq \sigma) \in \Sigma$, $\gamma(\sigma', x)$ es indefinida.

Esto es, un controlador de procedimientos puede estar en cualquiera de las siguientes dos situaciones: 1)un estado en el que un conjunto de transiciones incontrolables ocurre o, 2) un estado en el que la ejecución de una controlable es forzada.

La MEF controlador de procedimientos se define de tal manera que no es necesario un mecanismo externo para decidir cual transición controlable ejecutar en un estado dado y el comportamiento en lazo cerrado es igual al comportamiento del controlador. Sánchez *et al.* (2001) argumentan que ésto facilita la implementación del controlador de procedimientos en código de control o lógica alambrada.

Se asume que un controlador de procedimientos actúa en sincronía con el proceso adelantándose a la ocurrencia de cualquier transición incontrolable que pueda ocurrir sobre el estado de proceso actual. Aunque esta suposición de adelanto puede parecer fuerte, en gran parte es un problema de modelado, que en la mayoría de los casos resulta ser real para el tipo de plantas de proceso consideradas en este trabajo (es decir, plantas de procesamiento con dinámica relativamente lenta y principalmente de naturaleza procesal). Partiendo de lo anterior, es posible afirmar que si una transición controlable tiene dinámica lenta, esta siempre puede ser dividida en dos transiciones: una controlable rápida, representando la decisión de ejecución, y una incontrolable, representando la respuesta lenta del sistema a esa decisión.

Dentro del marco de trabajo de control de procedimientos, dos conceptos son de particular relevancia:

- Controlabilidad.
- Completud.

Definición 2.5.2 (Controlabilidad). Dado una MEF de proceso P y un lenguaje $\mathrm{K} \subseteq L(P)$, K es controlable con respecto a P si y solo si $\forall s \subseteq \bar{\mathrm{K}}$ se cumple alguna de las siguientes condiciones:

1. $s\Sigma_u \in P \wedge s\Sigma_u \in \bar{\mathrm{K}}$
2. $\exists \sigma_c \in \Sigma_c \mid s\sigma_c \in \bar{\mathrm{K}} \wedge (s\Sigma_u \cap \bar{\mathrm{K}} = 0)$

Definición 2.5.3 (Completud). Una MEF M = {X,Σ,γ,x_0,X_m} es completa con respecto a un proceso dado P={Q,Σ, δ, q_0, Q_m} si para $s \in \Sigma^*$ y $\sigma_u \in \Sigma_u$, las condiciones $\gamma(s, x_0)!$ y $\delta(s\sigma_u, q_0)!$ implica que una de las siguientes aseveraciones se cumple:

1. $\gamma(s\sigma_u, x_0)!$ o bien
2. $\exists \sigma_c \in \Sigma_c \mid \gamma(s\sigma_c, x_0)! \wedge \gamma(s\Sigma_u, x_0)$ es indefinida

Esto es, cualquier extensión de s con una transición incontrolable realizable por el proceso, también puede ser generada por el controlador, o bien existe una transición controlable que realiza la extensión de dicha trayectoria.

La noción de controlabilidad define el conjunto de trayectorias en el que el proceso puede existir sin peligro, mientras que la completud declara un controlador capaz de reproducir todas las trayectorias que pueden surgir durante la operación en lazo cerrado.

Definición 2.5.4 (Estado controlable). Dado un proceso P, el estado $x \in X$ es controlable con respecto a P si es posible definir $\delta(x, \sigma)$ bajo alguna de las condiciones siguientes:

1. $\forall \sigma_u \in \Sigma_u \mid x\sigma_u \in P$; $\delta(x, \sigma_u)!$ o bien
2. $\exists \sigma_c \in \Sigma_c \mid x\sigma_c \in P \wedge \delta(q, \sigma_c)!$

Esto es, es posible definir el estado con todas las incontrolables posibles en el proceso o con alguna controlable.

Definición 2.5.5 (Especificaciones controlables). Basados en la definición de estado controlable. Decimos que una especificación es controlable si no contiene estados incontrolables.

Sánchez *et al.* (1999) presentaron las siguientes condiciones de existencia (y su demostración) de un controlador de procedimientos.

Teorema 1. *Dado un lenguaje de especificación K(M)* $\subseteq L(P)$ *y* $K_m = L_m(P) \cap K$ $(K_m \neq 0)$, *si M es alcanzable y coalcanzable, y* K_m *es cerrado y controlable entonces* $\exists$ $C = \{X, \Sigma, \gamma, q_0, X_m\}$ *no bloqueante y completo, tal que* $L(C/P) \subseteq \overline{K_m}$.

2.6. Conclusiones

Este capítulo presentó el marco teórico que sustenta el algoritmo que se describe en el capítulo siguiente.

Capítulo 3

Algoritmo de síntesis

En este capítulo se presenta a detalle el algoritmo de cálculo incremental explícito que se propone en este trabajo de tesis para sintetizar un controlador de procedimiento dado un proceso, a partir de un conjunto de especificaciones de seguridad consistentes y especificaciones controlables.

Primeramente, se comenta el principio de los algoritmos que implementan los métodos conocidos a diferencia del algoritmo que en este trabajo de tesis se propone (Sección 2.5.3). Enseguida se presenta el principio de operación del algoritmo que se propone (Sección 3.2). Posteriormente se describe la operación detallada de éste (Sección 3.3). Se continúa con la descripción del pseudocódigo que constituye la implementación del algoritmo (Sección 3.4). Enseguida se muestra una descripción a bloques de la implementación del algoritmo sobre el método Michel (2002) (Sección 3.5). Por último, se presentan las conclusiones de este capítulo.

3.1. Introducción

Cuando se hace la síntesis de controladores de eventos discretos utilizando cálculo explícito, es práctica común modelar tanto el proceso de eventos discretos como el conjunto de especificaciones utilizando MEFs, de tal manera que su producto síncrono es utilizado como la base para calcular una superestructura de estado-transición (por ejemplo, la MEF que acepta el lenguaje supremo controlable en la teoría de control

supervisor propuesta por Ramadge y Wonham (1987a) o la máxima superestructura controlable en el caso del controlador de procedimientos propuesto por Sánchez *et al.* (1999)). De ésta estructura se obtiene un mecanismo controlador, el cual es mínimamente restrictivo con respecto a las especificaciones. Como se discutió en el capitulo 1, este procedimiento de construcción tanto del modelo de proceso como de la máxima superestructura es muy costoso computacionalmente.

El algoritmo que aquí se presenta no requiere de la construcción de un modelo de proceso ni de la máxima superestructura. Este calcula un controlador de procedimientos sobre la marcha, utilizando una estrategia recursiva en profundidad.

3.2. Principio de funcionamiento

El algoritmo se caracteriza por la forma incremental en que va construyendo el controlador. Esto consiste en la construcción en profundidad de la MEF estado por estado. Un estado forma parte del controlador si es alcanzable y controlable. Además si es un estado final, este debe ser marcado.

El proceso incremental estado por estado se lleva a acabo por *evolución natural del proceso*, es decir, cada uno de ellos se genera en función del estado antecesor y una transición habilitada por el proceso. Antes de aplicar el crecimiento por evolución natural, primeramente verifica si existe alguna especificación para crecer. Si es así, entonces el crecimiento es por especificación; en caso contrario el crecimiento se dará por evolución natural del proceso.

Antes de iniciar la descripción detallada del algoritmo, es necesario describir lo que es un *estado duplicable.*

3.2.1. Estados duplicables

Dentro del proceso de síntesis en general, un estado puede ser definido ya sea por comportamiento deseado (semitrayectoria de especificación) o por la naturaleza del proceso (evolución natural del proceso). Es decir, puede ser definido por alguna de las siguientes situaciones:

1. Alguna semitrayectoria tipo 2 o 3.

2. Evolución natural del proceso.

Lo anterior provoca que en la máquina del CP se definan estados diferentes con asignaciones iguales (estados duplicables). Dichos estados se definen a continuación.

Definición 3.2.1 (Estados duplicables). Sea una semitrayectoria de tipo 2 o 3, $\pi = s_0^{\tau_1} s_1^{\tau_2} \ldots^{\tau_m} s_m$ donde $m \in N_o$.

Un estado $x \in X$ es duplicable si y solo si $\beta(x) = s$, donde $s \in \{s_1, s_2, ..., s_{m-1}\}$.

3.3. Descripción del funcionamiento

Sánchez *et al.* (1999) expone las condiciones de existencia de un controlador de procedimientos no bloqueante y completo. Pero el algoritmo que aquí se presenta no garantiza que el controlador que calcula sea no bloqueante. Este calcula un controlador de procedimientos con base en las especificaciones y evolución natural del proceso.

Como se puede ver en la figura 3.3, el algoritmo de síntesis trabaja con la representación en MEFs de los modelos de componentes elementales, estado inicial, estados marcados y especificaciones.

En el proceso de construcción incremental del controlador mediante el algoritmo, decimos que la MEF correspondiente a la semitrayectoria de especificación de tipo 1 se *aplica* al estado en cuestión, quitando de éste las transiciones comunes que indica la especificación, ya que ésta describe la no ejecución de una transición o transiciones. Para las MEFs correspondientes a las semitrayectorias de especificaciones tipo 2 y 3, decimos que se *agregan* al controlador, ya que estas describen la ejecución condicional de una secuencia de eventos controlados.

En forma previa al procedimiento de síntesis mediante el algoritmo, existe una etapa de verificación de consistencia y controlabilidad de especificaciones. El proceso de síntesis inicia si las especificaciones son consistentes y controlables. La etapa de verificación de consistencia fue implementada por Michel (2002). La etapa de verificación de controlabilidad previa es parte de la investigación de este trabajo de tesis.

Básicamente el algoritmo está compuesto por la función recursiva *Crece*. Una descripción a bloques se presenta en la figura 3.1. El proceso de síntesis inicia con la llamada a la función, utilizando el estado inicial como argumento.

De acuerdo con la descripción en la figura 3.1. El primer paso dentro de la función consiste en verificar si el estado en cuestión es un estado identificado como *estado problema* (estado incontrolable o no deseado). Una vez que se identifica un estado problema, éste es memorizado para comparar con cada estado nuevo que surge, evitando el análisis repetitivo. Si el estado en cuestión se identifica como estado problema, entonces regresa paso recursivo. Si no es así, se verifica la condición de *existencia y duplicidad.* Es decir, para cada estado nuevo que no es identificado como estado problema, se verifica si ya es parte del controlador o no y si es duplicable o no; si es parte del controlador se verifica si es duplicable, si es duplicable se verifica si ya fue duplicado, si ya fue duplicado se cierra la trayectoria con éste y regresa paso recursivo. En el caso que el estado no exista aún en el controlador o exista, sea duplicable y no haya sido duplicado, entonces se verifica si existe alguna especificación tipo 1 para el estado en cuestión. Si existe especificación tipo 1, entonces se aplica al estado removiendo de éste las transiciones no deseadas que indica la especificación y después se verifica si existe alguna especificación de tipo 2 para tal estado. Si no existe especificación de tipo 1, directamente se verifica si existe alguna especificación de tipo 2. Si existe especificación tipo 2, entonces se agrega al controlador la MEF de la semitrayectoria de especificación correspondiente y se llama a la función crece con el último estado de la semitrayectoria. Si no existe especificación de tipo 2 para el estado en cuestión, entonces se aplica la definición de controlador de procedimientos (primero se intenta definir el estado con todas las incontrolables habilitadas por el proceso, si no es posible, se define con una controlable habilitada por el proceso) y después se verifica si existe especificación tipo 3 para el estado y cada transición según sea el caso a definir para el estado (todas las incontrolables habilitadas por el proceso o una controlable). Si existe especificación tipo 3 para el estado y transición, entonces agrega al controlador la MEF correspondiente a la semitrayectoria de especificación y se llama a *crece* con el último estado de la semitrayectoria. Si no existe especificación de tipo 3, entonces se llama a la función crece con el estado siguiente (estado sucesor

al estado en cuestión, producto de la evolución natural del proceso).

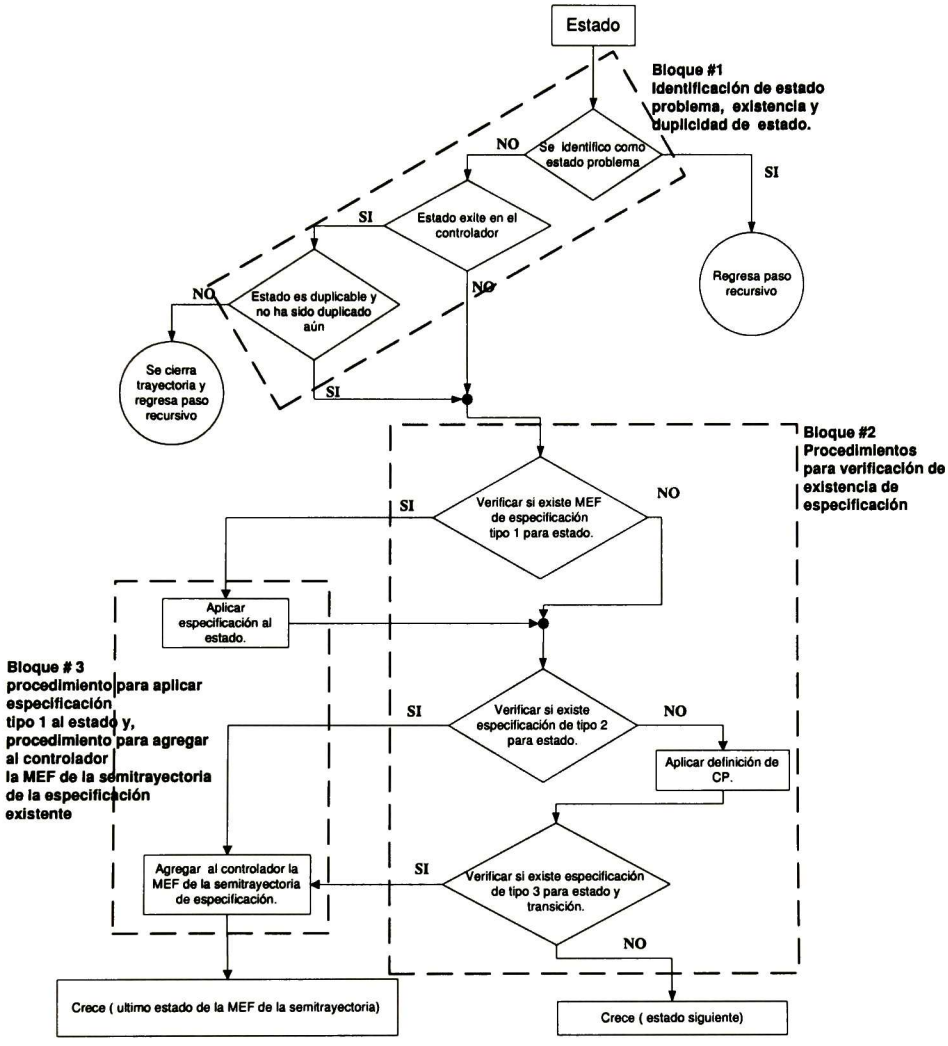

Figura 3.1: Función recursiva principal, Crece().

3.4. Pseudocódigo

De acuerdo con el funcionamiento del algoritmo descrito en 3.3, el pseudocódigo correspondiente a la función crece del algoritmo, comprende un procedimiento de

identificación de estado problema, existencia y duplicidad de estado. Tres procedimientos que verifican la existencia de especificación. Además un procedimiento para aplicar especificación tipo 1 al estado, y un procedimiento para agregar al controlador la MEF de la semitrayectoria de la especificación existente. En la figura 3.1 estos procedimientos se señalan como bloque 1, bloque 2 y bloque 3, en orden correspondiente.

A continuación se presenta el pseudocódigo correspondiente, dividiendo el algoritmo en dos bloques que comprenden:

1. Bloque #1; Verificación de estado problema, existencia y duplicidad de estado, y existencia de especificaciones tipo 1 y 2.

2. Bloque #2; Aplicar definición de controlador y verificar la existencia de especificación tipo 3.

En la figura 3.2 se señala con un recuadro cada uno de los bloques.

3.4.1. Pseudocódigo del bloque 1

Este procedimiento está compuesto por los siguientes puntos:

```
PARA CADA estado generado (estado nuevo creado por evolución
natural del proceso o último estado de la MEF de alguna
especificación tipo 2 o 3 agregada) HACER:

SI el estado no existe en el controlador y no ha sido analizado
aún o es duplicable y no ha sido duplicado aún, ENTONCES:

   PARA CADA especificación de tipo 1 HACER:

      SI existe especificación tipo 1 para el estado ENTONCES:

         Aplicar especificación tipo 1 al estado.
```

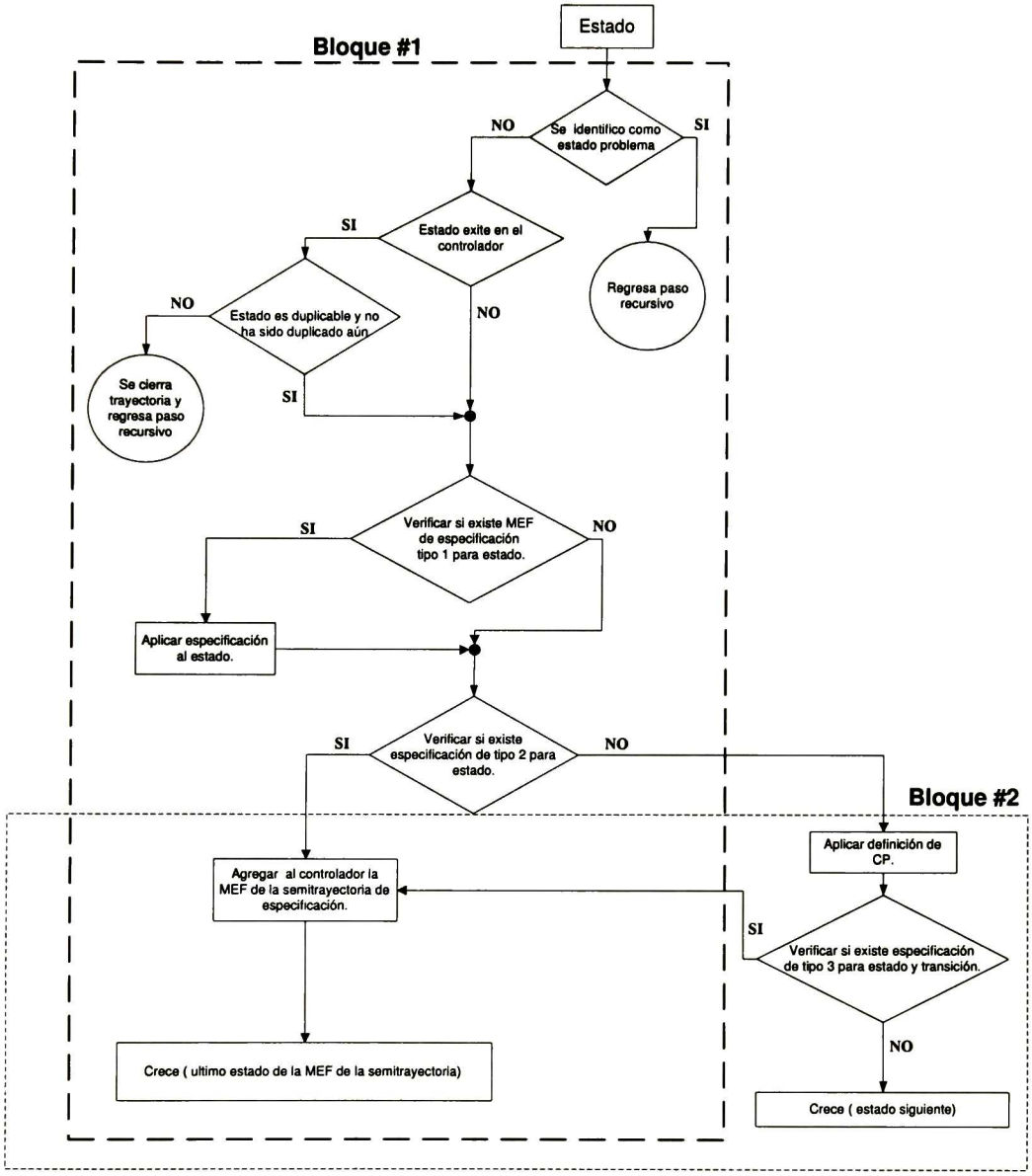

Figura 3.2: Bloques representativos del pseudocódigo de la función Crece.

```
PARA CADA especificación tipo 2 HACER:

    SI existe especificación tipo 2 para el estado ENTONCES:

        Agregar MEF de la semitrayectoria de especificación al
        controlador y llamar función crece con el último estado de
        la semitrayectoria de especificación.

    SI NO

        Pasar al bloque de definición de controlador de procedimientos y
        verificación de especificación tipo 3.
SI NO

    Regresa paso recursivo.
```

3.4.2. Pseudocódigo del bloque 2

En este bloque se define una trayectoria para el estado aplicando la definición de controlador. Esto es, el estado contiene todas las incontrolables o una controlable habilitadas por el proceso.

Primeramente se prueba si es posible definir con todas las incontrolables. Si esto no sucede, entonces se busca una controlable habilitada por el proceso. Cuando no se encuentra alguna de las anteriores especificaciones, éste debe ser un estado marcado para poder formar parte del controlador.

Ya sea con todas las incontrolables o una controlable, para el estado y cada transición, se verifica si existe una especificación de tipo 3, es decir, en este bloque se combina el pseudocódigo de la definición de controlador con el pseudocódigo del análisis de existencia de especificación tipo 3.

Este procedimiento está compuesto por los siguientes puntos:

```
SI estado contiene incontrolables que conllevan a estados
```

```
incontrolables ENTONCES:

   SI estado contiene controlables HACER:
       Tomar una controlable cualquiera.

       SI existe especificación tipo 3 para estado y transición HACER:
          AGREGAR especificación al controlador y llamar función crece
          utilizando el último estado de la semitrayectoria de la
          especificación como argumento.
   SI NO
       GENERAR estado siguiente correspondiente a la transición y llamar
       función crece utilizando el estado generado como argumento.
   SI NO
       SI estado es marcado HACER:
           AGREGAR estado a controlador y analizar estado siguiente (estado
           con transiciones pendientes por definir)
       SI NO
           Eliminar estado, guardarlo en la MEF de los estados problema y
           regresar paso recursivo.
SI NO para estado y cada transición HACER:
   SI existe especificación tipo 3 para estado y transición HACER:
       AGREGAR especificación al controlador y llamar función crece
       utilizando el último estado de la semitrayectoria de la
       especificación como argumento.
   SI NO
       GENERAR estado siguiente correspondiente a la transición y llamar
       función crece utilizando el estado generado como argumento.
```

3.5. Método de síntesis e implementación del algoritmo

El método de síntesis (Michel (2002)) con la implementación del algoritmo que aquí se propone, se define mediante el procedimiento a bloques que se muestra en la figura 3.3. Esto es, dado un proceso SED se obtiene primeramente la información necesaria para la síntesis: componentes elementales, especificaciones, estados marcados, y estado inicial. Cada información es codificada como una MEF individual. Estas MEFs individuales se utilizan como información de entrada al algoritmo para calcular un controlador de procedimientos.

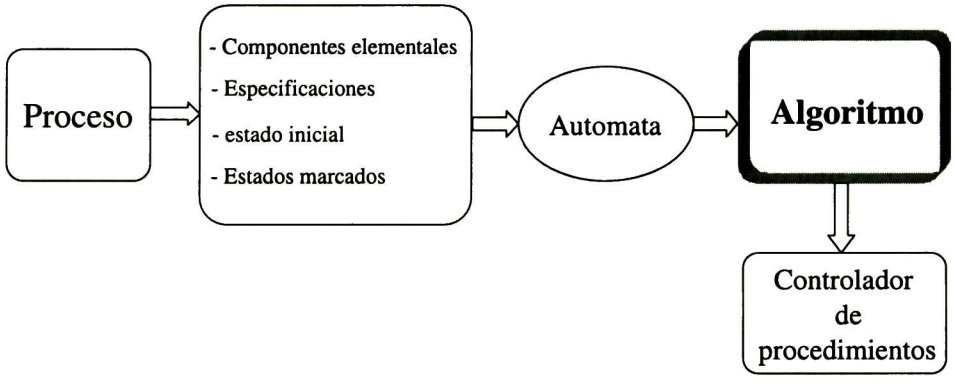

Figura 3.3: Representación a bloques del método de síntesis de controladores

3.6. Conclusiones

El algoritmo calcula un controlador de procedimientos C, codificado como una MEF, tomando solamente los modelos MEFs de especificaciones, componentes elementales, estados marcados y estado inicial, sin necesidad de construir un modelo del proceso.

Nótese que a diferencia de la inconsistencia entre especificaciones descrita en 2.4.1, en nuestro caso con el algoritmo incremental que aquí se propone, la inconsistencia se presenta solo entre especificaciones del mismo tipo, y entre tipo 1 con 2 y 3 de la

tabla 2.1. Por lo tanto, las especificaciones del mismo tipo son consistentes si y solo si se cumple solo uno de los siguientes enunciados:

1. No comparten *estados iniciales* equivalentes.

2. Si comparten *estados iniciales* equivalentes que implican la ejecución de una transición, entonces ésta debe ser la misma en dichos estados.

Las especificaciones de tipo 1 con 2 y 3 son consistentes si y solo si se cumple el siguiente enunciado:

1. Si comparten estados equivalentes, entonces no comparten transiciones.

Así en este trabajo de tesis las especificaciones de tipo 2 y 3, a diferencia de Michel (2002), son excluyentes en el sentido de inconsistencia, dado que el algoritmo que aquí se propone considera que tienen estados iniciales distintos, partiendo del hecho condicional. Es decir, las especificaciones de tipo 2 describen la ejecución de una transición, únicamente en función condicional de un estado, en tanto que las especificaciones de tipo 3 describen la ejecución de una transición, en función condicional de estado-transición-estado.

Si en el proceso de síntesis surge un estado incontrolable o no deseado, entonces el algoritmo resuelve esta situación con un bloqueo, constituido por una componente con un ciclo que no contiene un estado marcado.

Para ilustrar esta situación de bloqueo veamos la figura 3.4 como un controlador de procedimientos en construcción. Supongamos que el proceso de síntesis se ejecuta sobre el estado (V). Este estado tiene las transiciones incontrolables a, b, c, y las controlables e y f. Considerando que no existe especificación tipo 2 para el estado (V), el algoritmo trata de definir este estado con todas las incontrolables, pero la incontrolable c lleva a un estado no deseado, como se muestra en la figura 3.5, mismo que debe ser removido, implicando la no definición del estado en cuestión (V) con todas las incontrolables. En este momento el algoritmo toma una transición controlable (no lo hace de manera aleatoria, toma la primera de izquierda a derecha siempre). Como se muestra en la figura 3.6, esta transicion controlable lleva a un estado en el que la

incontrolable c, que es la que conduce a un estado problema, continua activa, y si no existe especificación de tipo 2 para el estado, tampoco se podrá resolver con todas las incontrolables. El algoritmo toma la controlable e' que lleva al estado antecesor, generando el bloqueo del controlador como se muestra en la figura 3.7. Como esta situación está bien definida, servirá para que el diseñador especifique la acción de control que evite la ocurrencia del estado problema. Para evitar este tipo de bloqueos, se decidió no incluir los estados no deseados como un tipo de especificación. Lo anterior lleva a concluir que si se desea anular un cierto estado, es necesario definir una acción de control que lo evite.

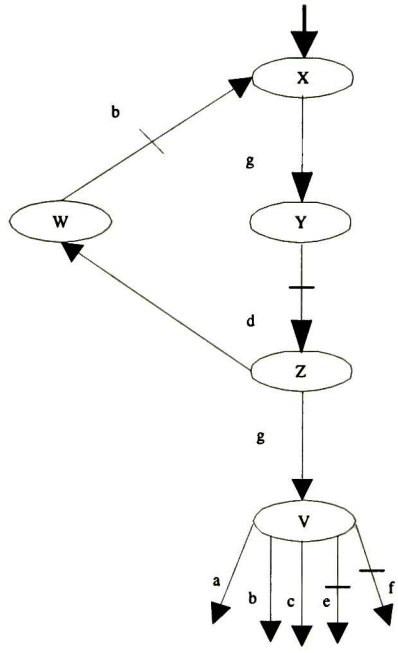

Figura 3.4: Bloqueo durante la síntesis incremental(1)

El presente capítulo ofreció un panorama sobre la construcción detallada del algoritmo de cálculo incremental. Para ilustrar el funcionamiento de éste, en el siguiente capítulo se mostrará la síntesis con detalle para un proceso hipotético.

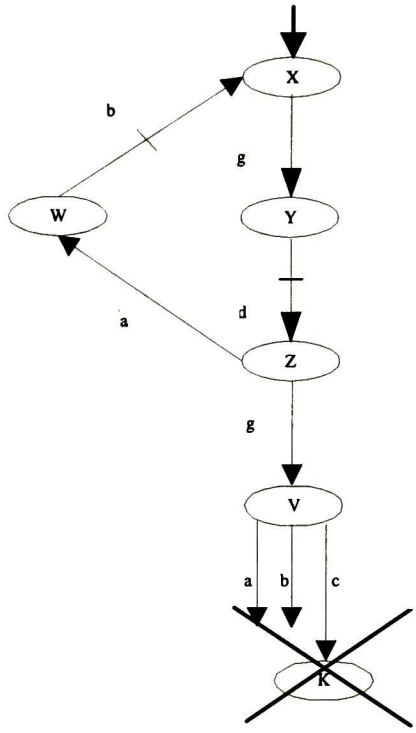

Figura 3.5: Bloqueo durante la síntesis incremental(2)

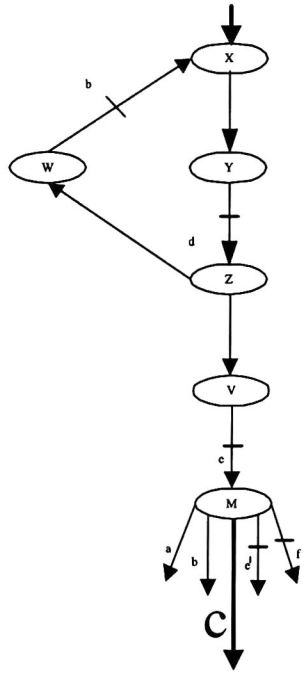

Figura 3.6: Bloqueo durante la síntesis incremental(3)

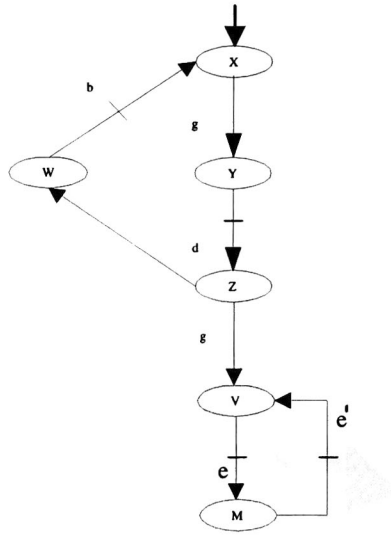

Figura 3.7: Bloqueo durante la síntesis incremental(4)

Capítulo 4

Implementación del algoritmo sobre el método Michel (2002) y prueba en un ejemplo hipotético

4.1. Contenido

Este capítulo ilustra el uso del algoritmo descrito en el capítulo 3, mismo que se implementa sobre el método de Michel (2002) para llevar a cabo la síntesis de un controlador de procedimientos utilizando un proceso hipotético.

Primeramente se comenta el objetivo de este capítulo y se define en qué consisten los ejemplos de prueba (Sección 4.2). Enseguida se muestra la descripción del proceso básico hipotético (proceso de menor tamaño en las pruebas)(Sección 4.3), que aparece en la figura 4.1. Posteriormente se describe paso a paso el desarrollo de la síntesis para el proceso básico (Sección 4.4). El tamaño del proceso es incrementado por bloques. Buscando ilustrar dicho incremento se muestra la síntesis para el proceso hipotético, primeramente con dos *bloques independientes* (Sección 4.6) y enseguida para dos *bloques dependientes* (Sección 4.7). Posteriormente se muestra una comparación gráfica de los resultados obtenidos de las variables de interés (tiempo de síntesis, tamaño del controlador, consumo de memoria, y recursividad) en la síntesis de los procesos de 1 a 7 bloques, donde se consideran bloques dependientes y bloques independientes

(Sección 4.8). Enseguida se muestra una comparación de los resultados obtenidos mediante la síntesis de los procesos de 1 a 4 bloques, utilizando el método Michel (2002) antes y después de la implementación del algoritmo que aquí se propone (Sección 4.9). Por último se comentan los resultados alcanzados con la implementación del nuevo algoritmo (Sección 4.10).

4.2. Bloques dependientes e independientes

Con el fin de obtener la respuesta de las variables de interés (tiempo de síntesis, tamaño del controlador, consumo de memoria, y recursividad) ante la explosión de estados, el tamaño del proceso es manipulado mediante el incremento por bloques a partir de un proceso básico constituido por un bloque. Cada bloque está formado por un actuador, una perturbación y un medidor.

Se hace la síntesis tanto para procesos con bloques dependientes como para procesos con bloques independientes. La diferencia entre unos y otros, radica en la relación de operación de los bloques en lazo cerrado. Lo anterior significa que un grupo de bloques operan de manera independiente cuando su ciclo de operación no depende en ningún momento de lo que ocurra en algún bloque distinto, en tanto que otro grupo de bloques operan de manera dependiente cuando su ciclo depende en algún momento de lo que ocurra en un bloque distinto. Así deducimos que que la relación dependiente o independiente se define mediante la descripción de las especificaciones, a partir de las cuales se describe la forma de operación deseada.

Las pruebas con bloques dependientes y bloques independientes se realizaron con el fin de probar si esta circunstancia tendría un efecto importante en el desempeño del algoritmo que el presente trabajo de tesis propone.

4.3. Descripción del ejemplo

Un SED, en general, está compuesto por elementos básicos como son: actuadores (e.g. válvulas, motores), perturbaciones (e.g. botones de emergencia), y medidores (e.g. sensores).

Con la finalidad de probar y mostrar el funcionamiento del algoritmo, se propone un proceso hipotético con una estructura general.

4.3.1. Descripción del proceso

El proceso básico hipotético se presenta en la figura 4.1. Este se compone de un actuador de dos posiciones (activado/desactivado), una perturbación (activada/desactivada) y un medidor de propiedad del proceso. Este último indica tres valores para la propiedad: bajo, normal y alto. Dicha propiedad puede incrementar su valor sólo si el actuador está activado, pero puede decrementarlo independientemente del estado del actuador. El elemento mencionado cambia de estado en función del elemento perturbación, lo cual significa que si la perturbación se activa, el actuador se activa. Si la perturbación se desactiva, entonces el actuador se desactiva. En el estado inicial el actuador y la perturbación se encuentran desactivados y el sensor en valor bajo.

A continuación se desea sintetizar un controlador de procedimientos para que el comportamiento en lazo cerrado del sistema cumpla con los siguientes requerimientos:

1. Si el actuador está activado y la medición cambia de valor bajo a normal, entonces el actuador se debe desactivar.

2. Si el actuador está desactivado, la medición baja y la perturbación se activa, entonces el actuador se debe activar.

3. Si el actuador y la perturbación están activados y la perturbación se desactiva, entonces el actuador se debe desactivar.

Se considera que la operación ha terminado con éxito cuando el proceso alcanza un estado en el que la medición es normal y el actuador y la perturbación están desactivados.

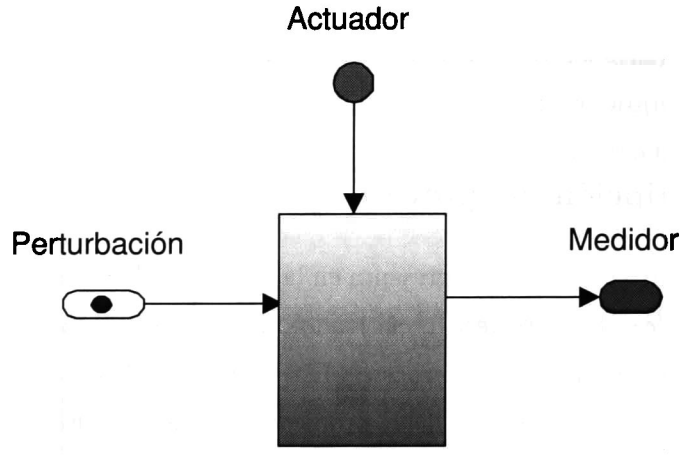

Figura 4.1: Proceso básico hipotético(un bloque).

4.4. Descripción del proceso de síntesis

4.4.1. Modelos de componentes elementales

Los modelos MEF correspondientes a cada componente se muestran en la figura 4.2. Estos fueron construidos considerando solo condiciones normales de operación.

Las transiciones controlables son identificadas mediante una línea transversal, y las incontrolables se identifican con ausencia de línea.

En la tabla 4.1 se muestra una interpretación física de las etiquetas de cada uno de los estados y transiciones de los modelos.

El estado del proceso se define mediante un vector ordenado, constituido por las variables de los componentes elementales que participan en el proceso. Por ejemplo Para el proceso básico hipotético (un bloque) descrito, el estado global del proceso se representará de manera particular mediante la siguiente distribución de variables:

[Perturbación, Actuador, Medidor]

Cabe mencionar que la distribución de variables no tiene efecto en el proceso de síntesis, pero si en la interpretación del vector de variables.

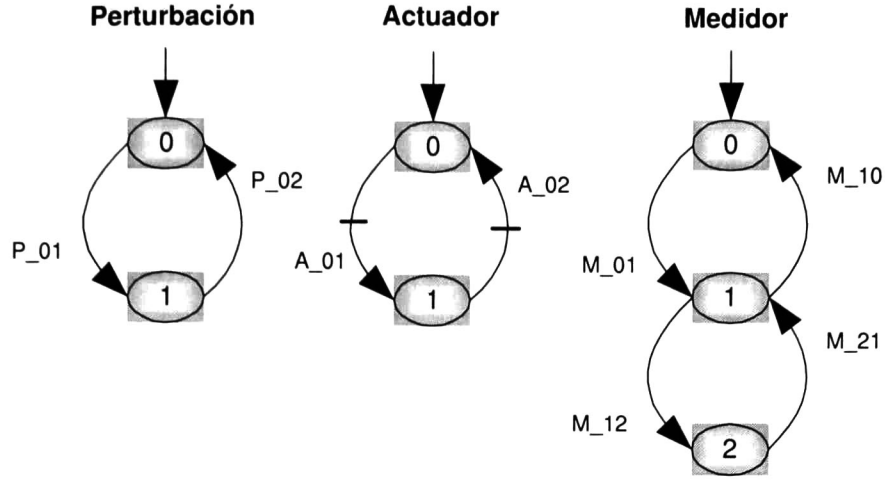

Figura 4.2: Modelos de componentes elementales del proceso básico(un bloque)

4.4.2. Modelos de especificaciones

El comportamiento deseado descrito en 4.3.1 es capturado mediante especificaciones de modelo y de control.

- Especificaciones de modelo (comportamiento causal)

1. Si el actuador no se encuentra activado, entonces el medidor no puede registrar un aumento en el valor de la medición. El aumento de la medición, corresponde a las etiquetas M_01 y M_12 del elemento de medición. Este comportamiento es capturado mediante la siguiente semitrayectoria de tipo 1.

$$(\infty,0,\infty) \text{ implic } [(\neg\, t = M\_01)\ y\ (\neg\, t = M\_12)]$$

- Especificaciones de Control

1. Ante el actuador activado y la medición normal, el actuador se debe desactivar.
2. Ante el actuador desactivado y la medición baja, si la perturbación cambia a activa, entonces el actuador debe, consecuentemente, activarse.

Componentes elementales	Valores de variables de estado	Transición	
		Etiqueta	Descripción
Medidor	0: Bajo 1: Normal 2: Alto	M_01 * M_12 * M_21 * M_10 *	Bajo a Normal Normal a Alto Alto a Normal Normal a Bajo
Actuador	0: Desactivado 1: Activado	A_01 A_10	Desactivado a Activado Activado a Desactivado
Perturbación	0: Desactivado 1: Activado	P_01 P_10	Desactivada a Activada Activada a Desactivada

Cuadro 4.1: Etiquetas de estados y transiciones

3. Ante el actuador activado y la perturbación activada, si la perturbación cambia a desactivada, entonces, el actuador se debe desactivar.

El primer requerimiento es capturado mediante la siguiente semitrayectoria de tipo 2C de la tabla 2.1:

$$[\ (\ \infty,\ 1,\ 1\ )\ ]\ \text{implic}\ [(\text{t=A\_10})]$$

El identificador ∞, indica cobertura (no importa) de la variable correspondiente.

El segundo y tercer requerimiento son capturados mediante las siguientes semitrayectorias de tipo 3C de la tabla 2.1.

$$[\ (\ 0{,}0{,}0\ )\ y\ (\text{t=P\_01})\ ]\ \textit{implic}\ [(\text{t=A\_01})]$$
$$[\ (\ 1,\ 1,\ \infty\ )\ y\ (\text{t=P\_10})\ ]\ \textit{implic}\ [(\text{t=A\_10})]$$

El estado inicial correspondiente es:

[(0, 0, 0)]

Y el estado marcado correspondiente es:

[(0, 0, 1)]

4.4.3. Descripción detallada de la síntesis y resultado

En primera instancia, previo al proceso de síntesis se verifica la consistencia y controlabilidad de especificaciones. De acuerdo con la definición de consistencia y controlabilidad descritos en 2.4.1 y 2.5.2 respectivamente, las especificaciones capturadas en este ejemplo son, precisamente, consistentes y controlables.

De acuerdo al diagrama de flujo de la figura 3.1, el procedimiento de síntesis arranca del estado inicial del proceso. En este ejemplo, el estado inicial es (0,0,0), el cual describe un estado de reposo de los elementos del proceso (actuador, perturbación, medidor). Se llama a la función crece con el estado inicial como argumento. El primer paso dentro de la función es verificar si es un estado que en pasos anteriores se registró como estado problema, lo cual se evalúa a falso.

El siguiente paso es verificar si ya existe en el controlador, lo cual también se evalúa falso. Entonces se verifica si existe alguna especificación tipo 1 para definir el conjunto de transiciones del estado. Después se verifica si el estado inicial es asignación inicial de alguna especificación tipo 2. Se puede ver en las especificaciones de tipo 2 obtenidas que el estado inicial no existe como asignación inicial.

Entonces, se aplica la definición de controlador (se toman una a una las transiciones incontrolables del estado y si no es posible definir con las incontrolables se toma una controlable), y se verifica la existencia del estado y la transición en la asignación inicial (estado y transición) de las especificaciones de tipo 3. En este caso el estado inicial solo tiene la transición incontrolable t = P_01 habilitada por el proceso.

Tomando el estado (0,0,0) y la transición t = P_01, se puede observar que es condición inicial de la especificación tipo 3C [(0,0,0) *y* (t=P_01)] *implic* [(t=A_01)]. Entonces se agrega en el controlador la MEF de la semitrayectoria correspondiente a la especificación. El resultado incremental es la estructura de la figura 4.3.

La ejecución recursiva del algoritmo se procesa con el último estado (1,1,0) de la semitrayectoria de la especificación. Pasando enseguida a hacer la llamada a la función crece con (1,1,0). Cabe mencionar que si surge un estado que ya existe en el controlador y no está definido como un estado duplicable (ver definición en 3.2.1), entonces el proceso incremental de la trayectoria en construcción termina con la unión

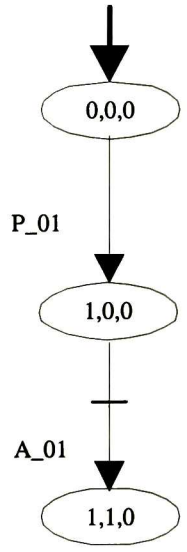

Figura 4.3: Controlador incremental(1)

de esta con el estado que ya existe en el controlador, y decimos que se cierra la trayectoria.

Todo estado suceptible de análisis, tiene como condicionante el verificar si en algún paso previo se detectó como un estado problema.

El estado (1,1,0) actual no se identifica como estado problema, entonces se verifica si existe en el controlador de modo que es posible cerrar la trayectoria. Que tampoco es el caso, por lo que se pasa a definir el conjunto de transiciones (si existe especificación tipo 1 se aplica para definir el conjunto de especificaciones) y después se constata si el estado existe como asignación inicial de especificación tipo 2. Dado que no se trata de una asignación inicial de tipo 2, el siguiente paso consiste en identificar si estado y transición es asignación inicial de alguna especificación tipo 3. El estado tiene dos transiciones incontrolables habilitadas por el proceso (t=P_10, t=M_01), es decir, existen dos trayectorias que requieren definición. Se toma el estado (1,1,0) y la transición incontrolable t = P_10, quedando por definir t=M_01. Se puede ver que estado y transición es asignación inicial de la especificación tipo 3C [(1, 1, ∞) *y* (t=P_10)] *implic* [(t=A_10)], la cual se agrega al controlador, obteniendo la

estructura de la figura 4.4.

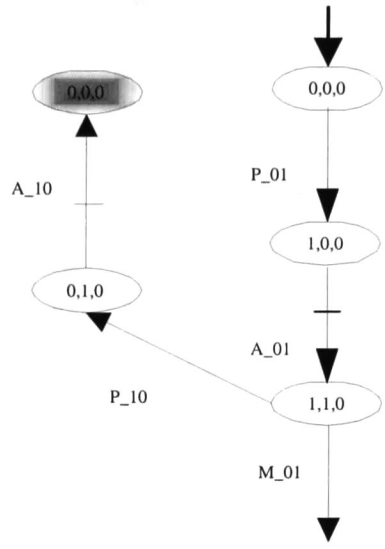

Figura 4.4: Controlador incremental(2)

El algoritmo se ejecuta para procesar el último estado (0,0,0) de la semitrayectoria de la especificación agregada. Entonces se llama a la función crece con (0,0,0). (0,0,0) no es un estado problema, pero se puede ver en la figura 4.4 (controlador actual) que ya existe en el controlador. Y como no se trata de un estado duplicable, entonces se cierra la trayectoria, obteniendo la estructura de la figura 4.5. El proceso recursivo retrocede hacia un estado antecesor con transiciones pendientes por definir. Entonces, el proceso recursivo se ejecuta con el estado (1,1,0), el cual tiene por definir la transición t=M_01. La ejecución del algoritmo con el estado (1,1,0) se encontraba en el punto de definición de estado y transición; entonces se toma el estado (1,1,0) y la transición t=M_01. Se puede ver que no es asignación inicial de tipo 3, por lo que se genera el estado nuevo sucesor (1,1,1) correspondiente a la transición t=M_01 y se hace la llamada a la función crece. Se verifica que (1,1,1) no sea un estado problema. Entonces se verifica si existe en el controlador de modo que es posible cerrar la trayectoria. Tampoco existe tal caso, entonces se obtiene el conjunto de transiciones

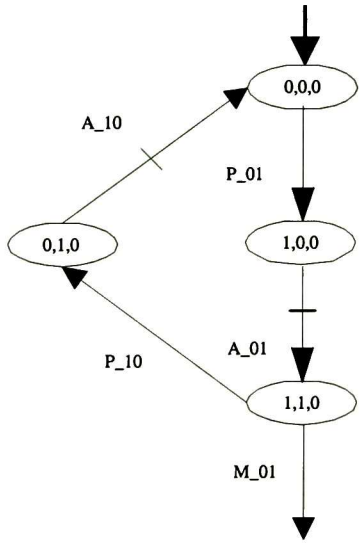

Figura 4.5: Controlador incremental(3)

habilitadas por el proceso, y después se verifica si existe como asignación inicial de especificación tipo 2. Dado que sí existe en [(∞, 1, 1)] implic [(t=A_10)], se agrega la MEF de la semitrayectoria correspondiente, obteniendo como resultado incremental la estructura de la figura 4.6.

El algoritmo se encuentra en el último estado (1,0,1) de la semitrayectoria, al igual que se analizaron los estados anteriores se puede ver que (1,0,1) no es un estado problema y no existe en el controlador de manera que se cierre la trayectoria. Entonces se obtienen las transiciones habilitadas por el proceso y se verifica si existe como asignación inicial de especificación de tipo 2. Como no existe, se aplica la definición de controlador y se verifica si estado y transición es asignación inicial de especificación tipo 3. El estado actual tiene las transiciones incontrolables t=M_10 y t=P_10 habilitadas por el proceso. Se busca si estado (1,0,1) y transición t=M_10 es asignación inicial de especificaciones tipo 3, quedando la transición t=P_10 por definir. Dado que no existe especificación para estado y transición, se genera el estado nuevo (1,0,0) correspondiente a la transición t=M_10. Se llama a la función crece con (1,0,0). Este no es un estado problema pero sí existe en el controlador, entonces se verifica si es

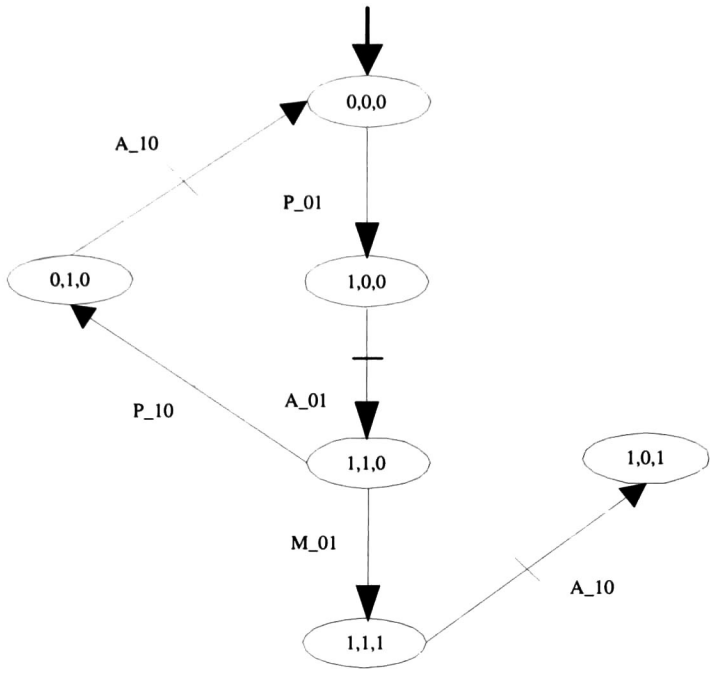

Figura 4.6: Controlador incremental(4)

duplicable y no ha sido duplicado aún. Dado que el estado (1,0,0) es duplicable, y en la figura 4.6 (controlador actual) se puede observar que no se ha duplicado aún. Se duplica obteniendo la estructura incremental de la figura 4.7.

Consecuentemente, se obtienen las transiciones habilitadas por el proceso, y se verifica si existe como asignación inicial en especificaciones de tipo 2. Como no existe tal, se aplica la definición de controlador. El estado (1,0,0) tiene solo la transición incontrolable t= P_10 habilitada por el proceso. Como estado y transición no es asignación inicial de especificación tipo 3, se genera el estado nuevo (0,0,0) correspondiente a la transición (t=P_10) como se muestra en la figura 4.8. Se llama a la función con (0,0,0). Este no es estado problema, pero ya existe en el controlador actual (figura 4.8) y no es duplicable. Por lo tanto, se cierra la trayectoria y se regresa del paso recursivo. La estructura resultante se muestra en la figura 4.9.

El proceso recursivo regresa al estado (1,0,1), ya que tiene la transición t=P_10 por definir. El análisis para (1,0,1) se encontraba en el punto de definición de estado

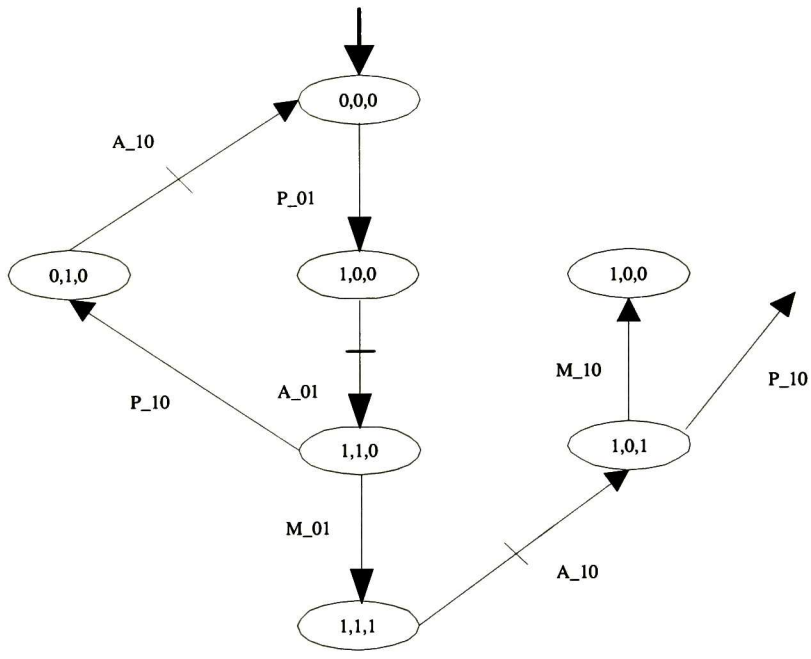

Figura 4.7: Controlador incremental(5)

y transición; entonces se toma el estado (1,0,1) y la transición t=P_10, que no es asignación inicial de tipo 3, por lo que se genera el estado nuevo (0,0,1) correspondiente a la transición (t=P_10), el resultado se ejemplifica en la estructura de la figura 4.10.

Como siguiente paso se llama a la función crece con (0,0,1). (0,0,1) no es un estado problema, tampoco es un estado que exista en el controlador, y no aparece como asignación inicial en especificación de tipo 2. Entonces, se aplica la definición de controlador y se verifica estado y transición. El estado tiene las transiciones incontrolables t=P_01 y t=M_10 habilitadas por el proceso. El estado (0,0,1) y la transición t=M_10 no existe como asignación inicial en especificación de tipo 3, entonces se genera el estado nuevo (0,0,0) correspondiente a la transición (t=M_10). (0,0,0) no aparece como estado problema, pero ya existe en el controlador actual (4.8) y no es duplicable. En este momento se cierra la trayectoria obteniendo la estructura de la figura 4.11, y regresa paso recursivo. El proceso recursivo regresa al estado (0,0,1) con la transición t = P_01, para definir continuar el análisis para estado y transición.

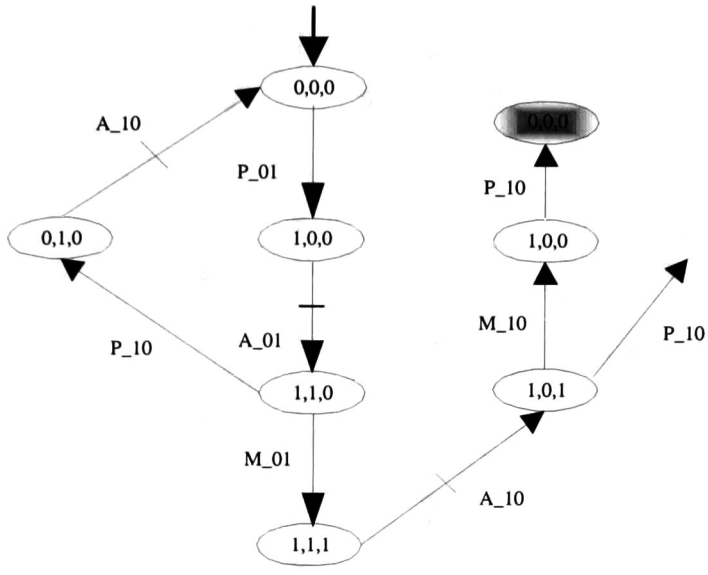

Figura 4.8: Controlador incremental(6)

Estado (0,0,1) y transición t = P_01 no es asignación inicial de especificación tipo 3, por lo que se genera el estado nuevo (1,0,1) correspondiente a la transición. (1,0,1) no es estado problema, pero ya existe en el controlador y no es duplicable, por lo que se cierra la trayectoria y el proceso recursivo regresa al estado inicial, debido a que no existen estados con trayectorias por definir.

En la figura 4.12 se muestra el controlador de procedimientos resultante. Este contiene las semitrayectorias de los requerimientos (trayectorias de color obscuro). Es decir, el controlador es capaz de conducir al proceso de manera que satisface los requerimientos.

4.5. Síntesis y crecimiento del proceso mediante el incremento por bloques

Se realizó la síntesis para modelos de proceso mediante el incremento de 1 a 6 bloques, tanto para procesos con bloques dependientes como independientes. Como

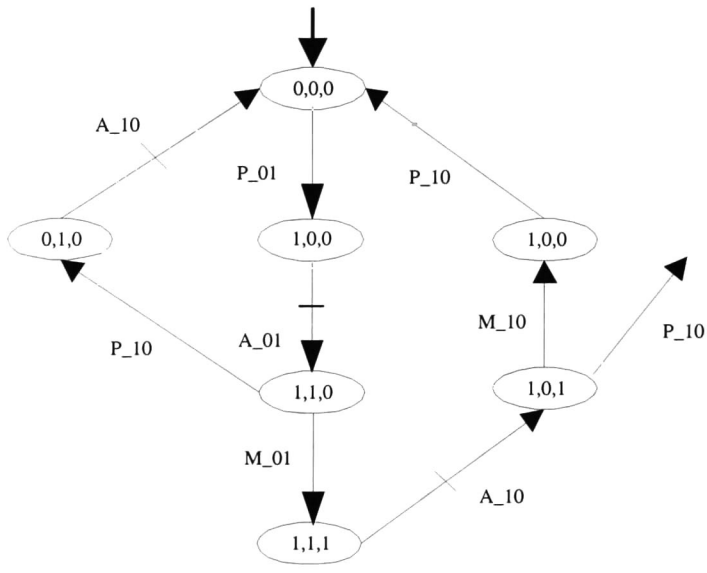

Figura 4.9: Controlador incremental(7)

ya se aclaró, la dependencia entre bloques radica únicamente en las especificaciones.

Buscando ilustrar el crecimiento del proceso por bloques, a continuación se presenta la síntesis para el proceso con dos de ellos. Se alude al caso tanto para bloques independientes, como para bloques dependientes.

4.6. Descripción del proceso con dos bloques independientes

La figura 4.13 muestra el diagrama del proceso correspondiente a dos bloques. Cada uno de ellos debe operar de manera independiente. Es decir, se desea sintetizar un controlador de procedimientos que permita conducir el proceso de modo que cada bloque funcione de manera independiente de acuerdo con las especificaciones dadas para el proceso básico (un bloque).

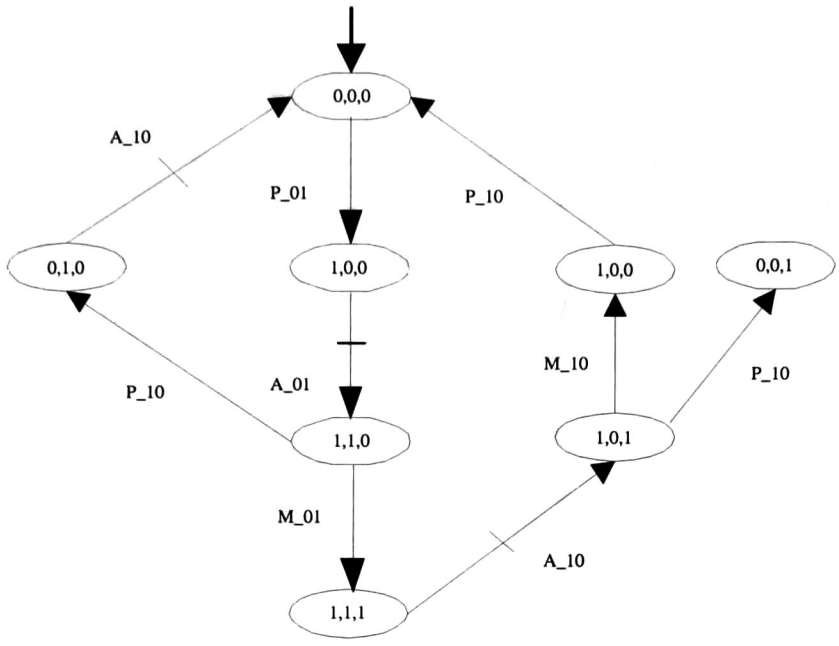

Figura 4.10: Controlador incremental(8)

4.6.1. Modelos de componentes elementales

Los modelos de componentes elementales por bloque, corresponden a los mostrados en la figura 4.2. De ahí se deduce que el proceso completo para dos bloques se constituye por seis modelos de este tipo. Las etiquetas para cada uno de los estados y transiciones por bloque se presentaron en la tabla 4.1.

Para definir el estado del proceso con dos bloques consideramos el vector ordenado para el proceso básico hipotético ([Perturbación, Actuador, Medidor]) descrito para un bloque. Así, para el proceso con dos bloques ordenaremos de manera particular el estado global del proceso como:

[Bloque1, Bloque2]

Obteniendo la siguiente distribución de variables:

[Perturbación1, Actuador1, Medidor1, Perturbación2, Actuador2, Medidor2]

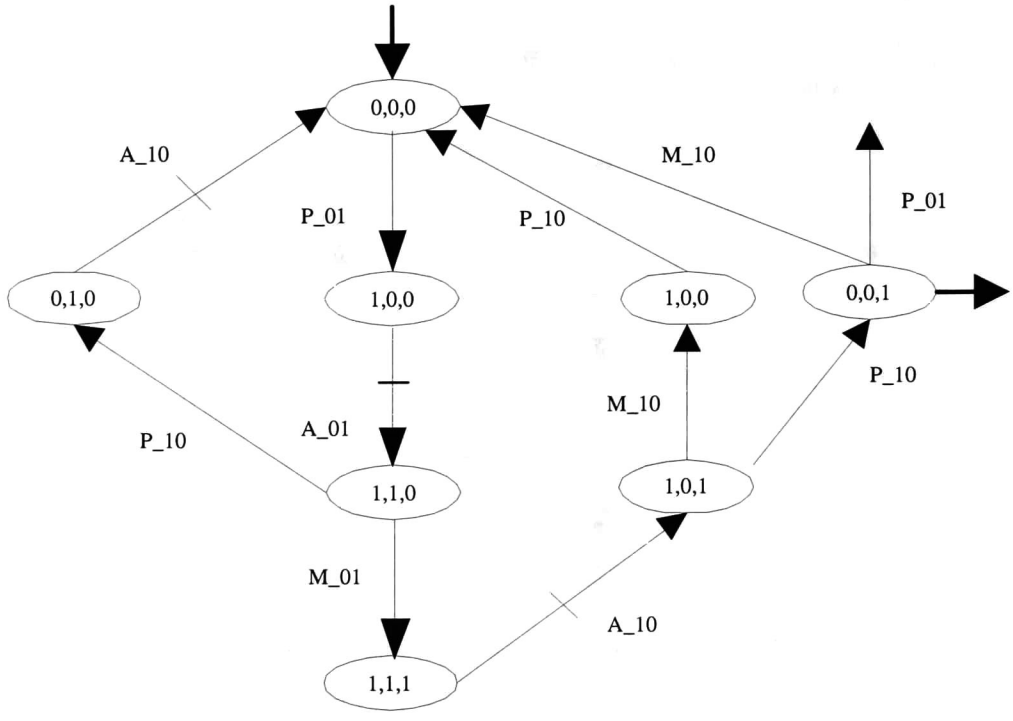

Figura 4.11: Controlador incremental(9)

4.6.2. Modelos de especificaciones

Las especificaciones de comportamiento a lazo abierto y lazo cerrado para cada uno de los bloques son las mismas que se dieron para el proceso básico (un bloque). De acuerdo con el estado global dado para dos bloques, corresponden las siguientes semitrayectorias.

1. Especificaciones de comportamiento a lazo abierto (comportamiento causal):

   [(∞,0,∞,∞,∞,∞)] *implic* [($\neg$ t=M1_01) *y* ($\neg$ t=M1_12)]

   [(∞,∞,∞,∞,0,∞)] *implic* [($\neg$ t=M2_01) *y* ($\neg$ t=M2_12)]

2. Especificaciones de comportamiento a lazo cerrado:

   [(∞,1,1,∞,∞,∞)] *implic* [(t=A1_10)]

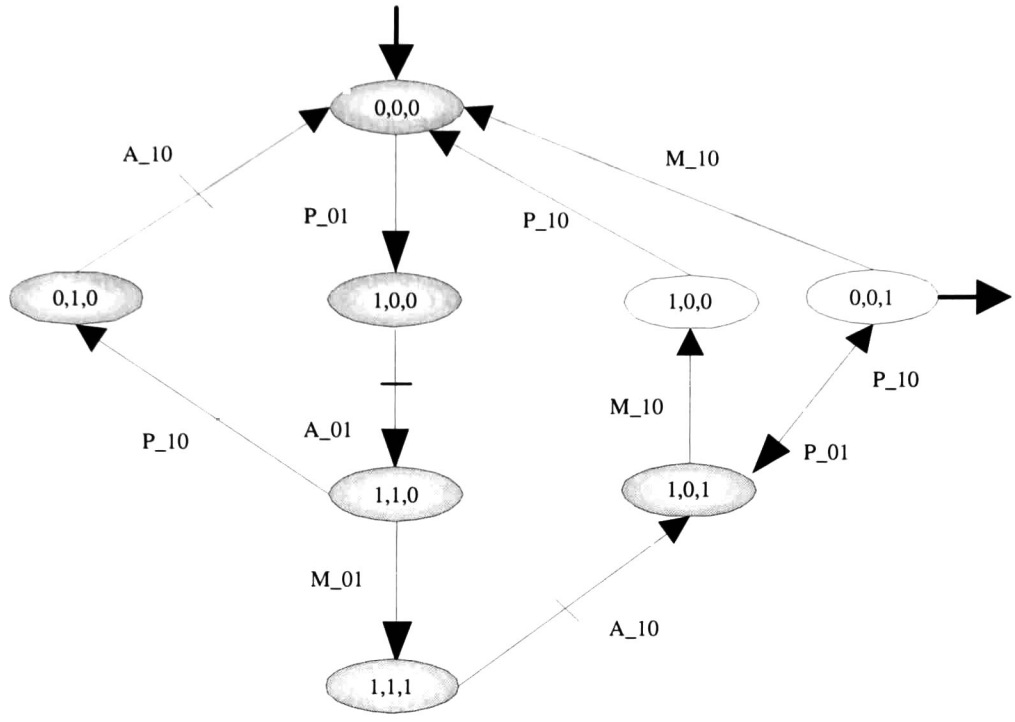

Figura 4.12: Controlador incremental(9)

[(∞,∞,∞,∞,1,1)] *implic* [(t=A2_10)]

[(0,0,0,∞,∞,∞) *y* (t=P1_01)] *implic* [(t=A1_01)]

[(∞,∞,∞,0,0,0) *y* (t=P2_01)] *implic* [(t=A2_01)]

[(1,1,∞,∞,∞,∞) *y* (t=P1_10)] *implic* [(t=A1_10)]

[(∞,∞,∞,1,1,∞) *y* (t=P2_10)] *implic* [(t=A2_10)]

4.6.3. Resultados del proceso con dos bloques independientes

El tamaño del proceso con dos bloques consta de 144 estados. La solución del cálculo del controlador se realizó en 2 segundos, obteniendo un tamaño de controlador de 55 estados, un consumo de memoria de 1,116 KB y una ejecución recursiva en profundidad del algoritmo de 25 recursiones.

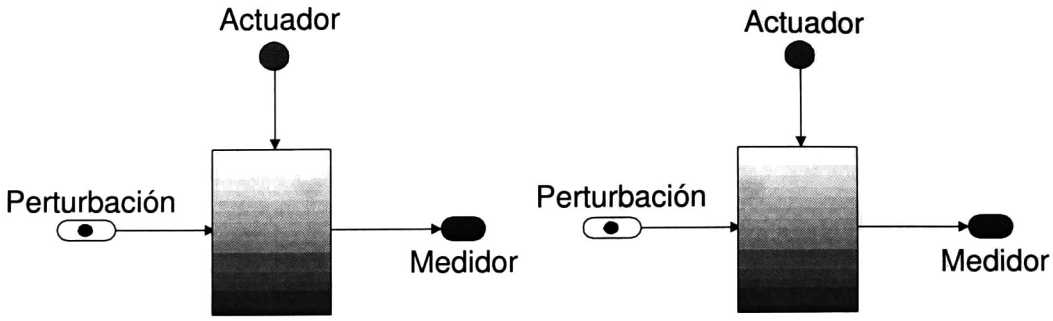

Figura 4.13: Proceso hipotético con dos bloques.

Estos resultados fueron obtenidos utilizando una computadora con las siguientes características:

- Pentium III 800 MHZ
- 632 MB RAM

4.7 Descripción del proceso con dos bloques dependientes

Como ya se comentó, el proceso dependiente e independiente no implica diferencia entre el proceso físico, así el proceso para dos bloques dependientes es el mismo que se utilizó para el proceso con bloques independientes.

Se quiere sintetizar un controlador para operar los bloques mediante los requerimientos establecidos para el proceso anterior con bloques independientes, agregando el siguiente requerimiento:

- Cada actuador se activa si ante el medidor correspondiente en bajo y el medidor del bloque antecesor en normal, la perturbación correspondiente se activa.

Este nuevo requerimiento establece una relación de funcionamiento secuencial dependiente entre los bloques consecutivos, es decir; los bloques operarán en serie.

4.7.1. Modelo de especificaciones

Las especificaciones de comportamiento a lazo abierto y lazo cerrado correspondientes al proceso con bloques dependientes, se formalizan mediante las siguientes semitrayectorias. (considerando la misma distribución de variables (bloque1, bloque2) que para el caso con bloques independientes):

1. Especificaciones de comportamiento a lazo abierto (comportamiento causal):

   [(∞,0,∞,∞,∞,∞)] *implic* [(¬ t=M1_01) *y* (¬ t=M1_12)]

   [(∞,∞,∞,∞,0,∞)] *implic* [(¬ t=M2_01) *y* (¬ t=M2_12)]

2. Especificaciones de comportamiento a lazo cerrado:

   [(∞,1,1,∞,∞,∞)] *implic* [(t=A1_10)]

   [(∞,∞,∞,∞,1,1)] *implic* [(t=A2_10)]

   [(0,0,0,∞,∞,∞) *y* (t=P1_01)] *implic* [(t=A1_01)]

   [(∞,∞,1,0,0,0) *y* (t=P2_01)] *implic* [(t=A2_01)]

Como se puede observar el conjunto de especificaciones obtenido hasta el momento es equivalente al del proceso con bloques independientes. Las semitrayectorias correspondientes al nuevo requerimiento son:

[(1,1,∞,∞,∞,∞) *y* (t=P1_10)] *implic* [(t=A1_10)]

[(∞,∞,∞,1,1,∞) *y* (t=P2_10)] *implic* [(t=A2_10)]

4.7.2. Resultados del proceso con dos bloques dependientes

El tamaño del proceso con dos bloques consta de 144 estados. La solución del cálculo del controlador se realizó en 1.6 segundos, obteniendo un tamaño de controlador de 52 estados, un consumo de memoria de 1,120 KB y una ejecución recursiva en profundidad del algoritmo de 24 recursiones.

4.8. Resultados ante el incremento por bloques (explosión de estados)

Se realizó la síntesis para los procesos de 1 a 7 bloques dependientes e independientes. Los resultados se muestran a continuación como gráficas en escala logarítmica. Las figuras 4.14, 4.16, 4.15 y, 4.17, respectivamente, muestran el comportamiento de las variables (todas en función del incremento del tamaño de proceso):

- Tiempo de síntesis.
- Tamaño del controlador de procedimientos.
- Memoria.
- Recursividad.

Estos resultados fueron obtenidos utilizando una computadora con las siguientes características:

- Pentium III 800 MHZ

  632 MB RAM

De la gráfica en la figura 4.14 se deduce que la complejidad temporal del algoritmo es exponencial con respecto al número de estados alcanzables del proceso, en virtud de que los resultados describen una línea recta sobre la escala logarítmica. De las gráficas 4.16 y 4.17, correspondientes a tamaño del controlador vs. proceso en lazo abierto y recursividad vs. proceso lazo abierto sucesivamente, se deduce que la complejidad del algoritmo es menor a la exponencial. Este resultado es consecuencia del comportamiento decreciente de éstas.

Por último, de la gráfica 4.15 correspondiente a la memoria utilizada, se deduce que la complejidad espacial del algoritmo con respecto al número de estados alcanzables es mayor a la exponencial, dado el comportamiento creciente de ésta.

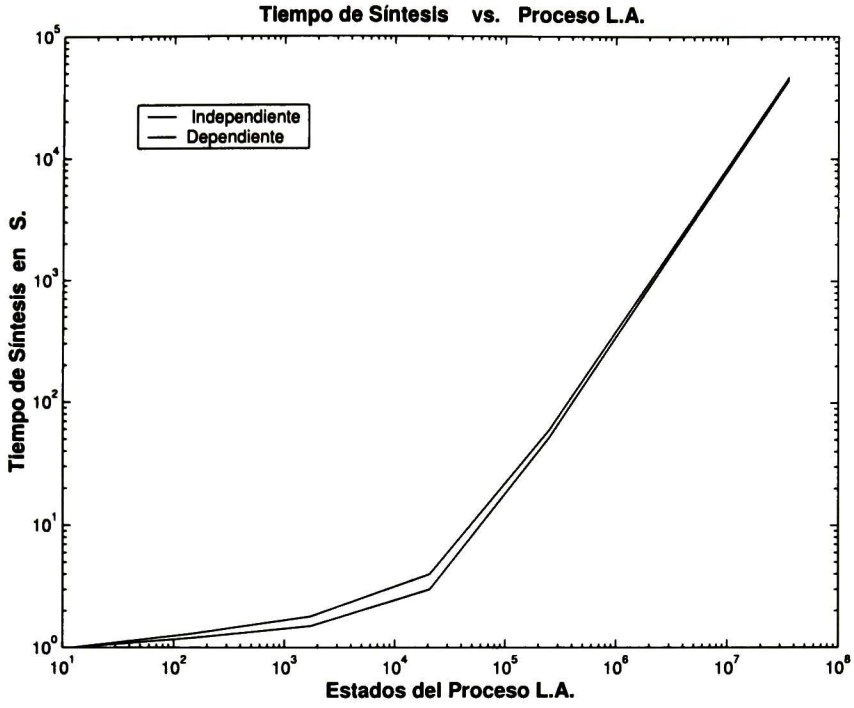

Figura 4.14: Resultados de Tiempo de síntesis vs. Proceso en lazo abierto

4.9. Comparación de resultados entre métodos

Con el fin de evaluar los resultados del método propuesto en Michel (2002) antes y después de la implementación del algoritmo desarrollado en este trabajo de tesis, en las tablas 4.2 y 4.3 se muestran los resultados de la síntesis tanto para procesos con bloques dependientes como para procesos con bloques independientes. Tales resultados se muestran hasta un cierto número de bloques, de acuerdo con la *capacidad de cálculo* (tamaño máximo de proceso que se puede resolver) del método correspondiente.

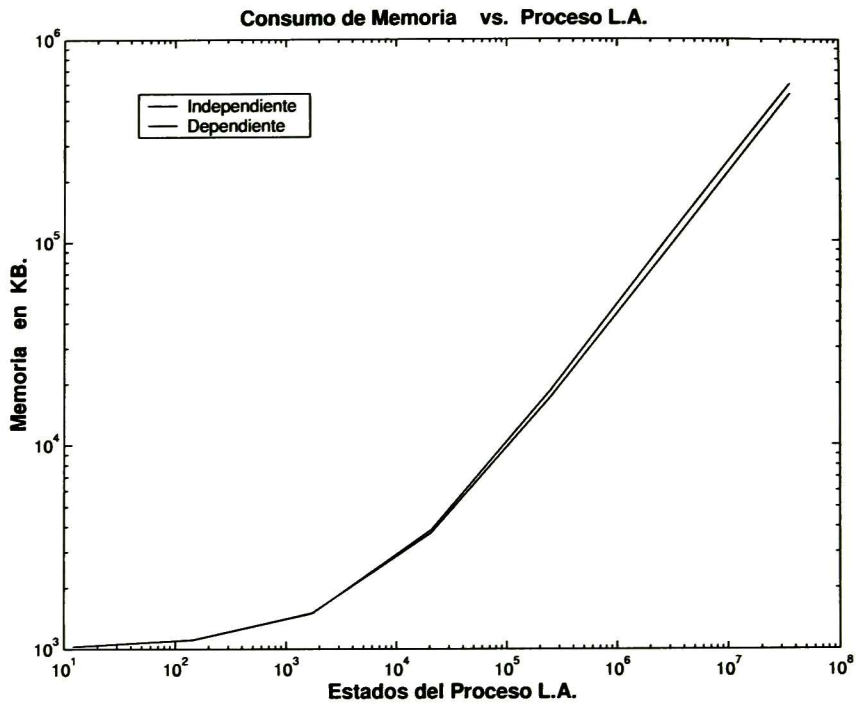

Figura 4.15: Resultados de Memoria vs. Proceso en lazo abierto

1	12	1	8	1,040
2	144	2	55	1,116
3	1,728	3	350	1,588
4	20,736	9	2,125	4,648
5	248,832	62	12,500	25,556
6	2,985,984	2,070	71,875	164,020
7	35,831,808	46,314	274,768	596,968
1	12	1	8	925
2	144	2	55	1,512
3	1,728	25	350	9,800
4	20,736	11,460	2,125	157,828

Cuadro 4.2: Tabla de resultados ante procesos con bloques independientes.

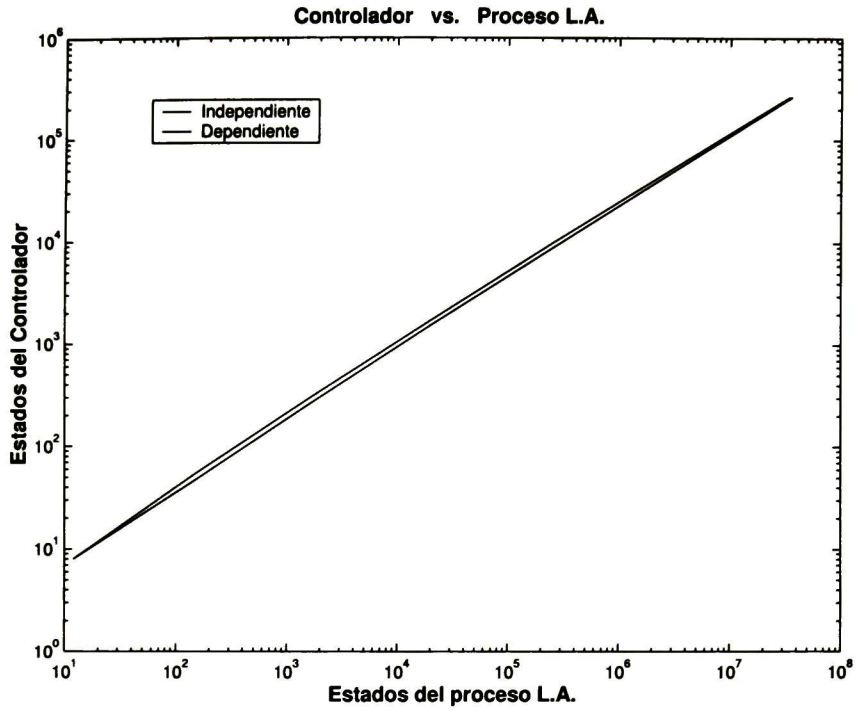

Figura 4.16: Resultados de Tamaño del controlador vs. Proceso en lazo abierto

1	12	1	8	1,028
2	144	1	52	1,108
3	1,728	4	320	1,500
4	20,736	6	1900	3,876
5	248,832	52	8,946	18,612
6	2,985,984	1,546	48,903	107,536
7	35,831,808	44,040	264,684	531,756
1	12	1	8	925
2	144	2	52	1,504
3	1,728	25	320	9,764
4	20,736	11,400	1,900	156,456

Cuadro 4.3: Tabla de resultados ante procesos con bloques dependientes.

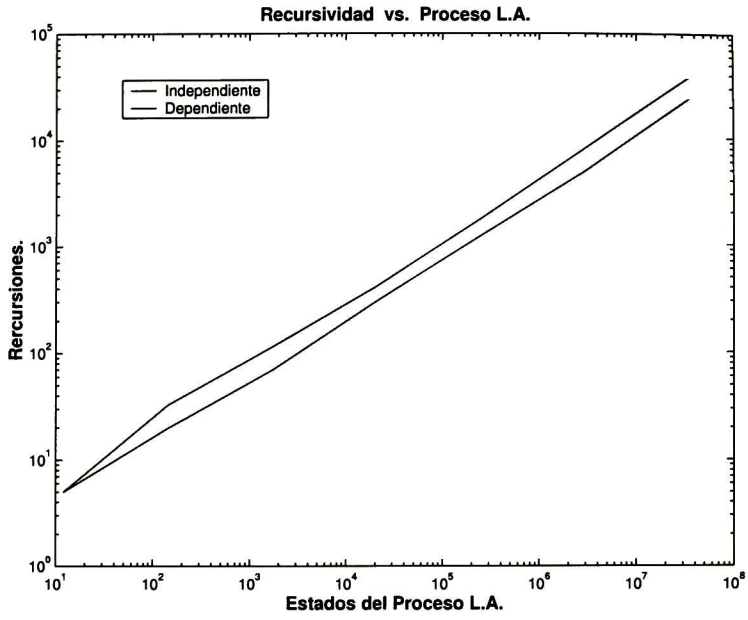

Figura 4.17: Resultados de Recursividad vs. Proceso en lazo abierto

4.10. Conclusiones

Los resultados obtenidos para procesos con bloques dependientes y procesos con bloques independientes describen curvas similares, es decir; la dependencia o independencia entre bloques no trae como consecuencia un efecto considerable en la complejidad del algoritmo.

Como se puede observar en los resultados de las tablas 4.2 y 4.3, el método de Michel (2002) tiene una capacidad de cálculo para tamaños de proceso de $2(10^4)$ estados. Con la implementación del algoritmo que se propone, se alcanzó una capacidad de cálculo para tamaños de proceso de $3(10^7)$ estados.

La capacidad de cálculo del algoritmo que aquí se propone está en función directa del tamaño del controlador resultante y no necesariamente del tamaño del proceso. Esta premisa se deduce de manera lógica y con base empirica, partiendo del siguiente razonamiento:

Conforme el algoritmo va construyendo de manera incremental el controlador, implementa procedimientos de búsqueda sobre éste para cada uno de los estados

en cuestión. A medida que el controlador crece, la búsqueda se vuelve más costosa computacionalmente hablando.

El algoritmo que se desarrolló tiene características importantes, del mismo modo que carece de ciertas propiedades que debería garantizar.

En el siguiente capítulo se presentan estas características y las propuestas para desarrollar un trabajo futuro que ayude a garantizar los resultados deseados.

Capítulo 5

Conclusiones y trabajo futuro

5.1. Contenido

En el presente capítulo se exponen los resultados alcanzados a través de esta investigación, asì como el trabajo futuro a realizar. Primeramente se describen los resultados alcanzados (Sección 5.2). Posteriormente se describe en que consiste el trabajo en perspectiva (Sección 5.3).

5.2. Resultados alcanzados

Como principal resultado alcanzado en este trabajo de investigación tenemos un nuevo algoritmo de síntesis de controladores de procedimientos con las siguientes características:

1. Cálculo directo del controlador sin necesidad de un modelo del proceso o de una máxima superestructura.
2. Capacidad de cálculo para sistemas con ordenes de $3(10^7)$ en el espacio de estados.
3. Procedimiento encargado del análisis de controlabilidad pre-síntesis, del conjunto de especificaciones introducido como comportamiento deseable bajo la acción del controlador.

4. Flexibilidad de consistencia entre especificaciones tipo 2 y 3. Es decir, las semi-trayectorias aquí descritas como tipo 2 y 3, revisten consistencia independiente para el algoritmo que se propone, debido a que sus condiciones iniciales se consideran excluyentes en el proceso incremental. Por ende la inconsistencia entre una especificación tipo 2 con una especificación tipo 3 no ocurre como en (Sánchez *et al.*, 2002).

   En determinadas situaciones se quiere ejecutar una acción de control diferente sobre un mismo estado dependiendo de la ocurrencia previa al mismo. Esto conduce a la lógica inferencia que sustenta un hecho:

   Para un mismo estado del proceso debe ser posible ejecutar una acción de control distinta dependiendo de las condiciones previas. Lo cual es posible gracias a la flexibilidad de especificaciones tipo 2 y 3 que presenta el algoritmo que aquí se propone.

5.3. Trabajo futuro

Como trabajo futuro, existen los siguientes puntos:

1. Garantizar que el algoritmo que aquí se propone calcule un controlador no bloqueante.

2. Aseo de la MEF del controlador resultante. Es decir, buscar y eliminar de la máquina del controlador calculado las posibles trayectorias no alcanzables. Las cuales si aparecen, son producto de la eliminación de un estado problema o no controlable.

   Así mismo la eliminación de trayectorias comunes que representarían código repetible en la implementación del código de control.

3. Mejorar la capacidad de cálculo del algoritmo. Esto es, implementar nuevas técnicas en el proceso de búsqueda del estado en cuestión sobre el controlador. Esto en virtud de que la lógica del proceso de síntesis mediante el algoritmo

incremental que aquí se propone y las pruebas realizadas nos lleva a deducir que la mayor parte del tiempo de síntesis se invierte en el proceso de búsqueda.

4. Cambio de compilador con el fin de no vernos limitados en el proceso recursivo.Es decir, buscar un compilador en el que la memoria de la pila (unidad en la cual se almacena momentaneamente la información del proceso recursivo) no esté limitada por el sistema.

   El algoritmo se programó utilizando la herramienta de programación C++ Builder, el cual permite asignar un máximo de 16 Mega Bytes de memoria en la pila, cantidad que resulto insuficiente para tratar procesos de mayor tamaño que los aquí mostrados.

5. Formalizar la complejidad del algoritmo. Esto es, aunque se hicieron pruebas numéricas para tratar de identificar la complejidad del algoritmo, es necesario definir esta mediante procedimientos formales existentes.

Referencias

A. Michel. Verificación de una clase de especificaciones de seguridad en controladores de procedimientos. Technical report, Cinvestav, 2002. Tesis de Maestría.

N. Alsop, L. Camillocci, A. Sanchez, and S. Macchietto. Synthesis of procedural controllers - Application to a batch plant. *Computers and Chemical Engineering*, 20(Suppl.):S1481–S1486, 1996.

E. Asarin, O. Maler, and A. Pnueli. Symbolic controller synthesis for discrete and timed systems. In J. Hartmanis G. Goos and J. Van Leeumen, editors, *Lecture Notes in Computer Science*, pages 1–20. Springer, 1994.

S. Balemi. Discrete–event systems control of a rapid thermal multiprocessor. In *Proc. 7th. IFAC/IFIP/IFORS/IMACS/ISPE Symp. Inf. Contr. Problems in Manufacturing Technology*, Toronto, Canada, June 24-26 1992.

R.E. Bryan. Graph-based algorithms for boolean function manipulation. Technical report, IEEE Transactions on Computer, 1986.

C. G. Cassandras and S. Lafortune. *Introduction to Discret Event Systems.* Kluwer Academic, 1999.

G. Hoffmann and H. Wong–Toi. Symbolic synthesis of supervisory controllers. In *Proceedings of the 1992 American Control Conference*, pages 2789–2793, June 24-26 1992.

P Mouillé. Synthesis of stable and optimal procedural controllers for chemical processes. Technical report, Imperial College, London, 1997.

L. Parra, N. Acosta, and A. Sánchez. Documentación del kernel de cálculo de spectool v1.0. Technical report, CINVESTAV, 1999.

P. J. Ramadge and W. M. Wonham. Modular feedback logic for discrete event systems. *SIAM Journal of Control and Optimization*, 25(5):1202–1218, 1987.

P. J. Ramadge and W. M. Wonham. Supervisory control of a class of discrete–event processes. *SIAM Journal of Control and Optimization*, 25(1):206–230, 1987.

A. Sanchez, G. Rotstein, N. Alsop, and S. Macchietto. Synthesis and implementation of procedural controllers for event–driven operations. *AIChE Journal*, 45(8):1753–1775, 1999.

A. Sanchez, G. E. Rotstein, N. Alsop, J. P. Bromberg, C. Gollain, S. Sorensen, S. Macchietto, and C. Jakeman. Improving the development of event-driven control systems in the batch processing industry. A case study. *ISA Trans.*, 2001. In press.

A. Sanchez. *Formal Specification and Synthesis of Procedural Controllers for Process Systems.* Lecture Notes on Control and Information Sciences, v. 212, Springer–Verlag, 1996.

A. Sánchez, R. E. Gonzalez, and A. Michel. Analysis of safety properties in the synthesis of discrete-event controllers. In *15th IFAC World Congress. Barcelona, Spain*, July 2002.

J. Tsitsiklis. On the control of discrete–event dynamical systems. *Mathematics of Control, Signals and Systems*, 2:95–107, 1989.

Centro de Investigación y de Estudios Avanzados del IPN

Unidad Guadalajara

El Jurado designado por la Unidad Guadalajara del Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional, aprobó la tesis: SÍNTESIS DE CONTROLADORES DE PROCEDIMIENTOS PARA UNA CLASE DE ESPECIFICACIONES DE SEGURIDAD UTILIZANDO TÉCNICAS DE CÁLCULO INCREMENTAL EXPLÍCITO del(a) C. Juan José VENEGAS MORENO el día 13 de Diciembre de 2002

DR. ARTURO DEL SAGRADO CORAZÓN SÁNCHEZ CARMONA
INVESTIGADOR CINVESTAV 3B
CINVESTAV GDL
GUADALAJARA

DR. JOSÉ JAVIER RUIZ LEÓN
INVESTIGADOR CINVESTAV 2C
CINVESTAV GDL
GUADALAJARA

DR. RAUL ERNESTO GONZÁLEZ TORRES
INVESTIGADOR CINVESTAV 2C
CINVESTAV GDL
GUADALAJARA