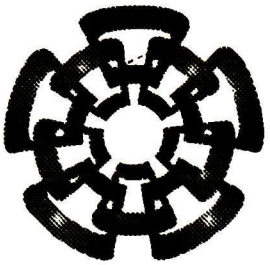
DIS-16168



CINVESTAV - IPN

Centro de Investigación y de Estudios Avanzados del IPN
Unidad Guadalajara

Laboratorio de Ingeniería Eléctrica y Ciencias de la Computación - Guadalajara

Sistema de Pruebas para Modelado y Control de Sistemas de Eventos Discretos

Tesis que presenta
Rosa María Córdova Canela

Para obtener el grado de
Maestro en Ciencias

En la especialidad de
Ingeniería Eléctrica

Guadalajara, Jal., Noviembre de 1998

Sistema de Pruebas para Modelado y Control de Sistemas de Eventos Discretos

Tesis de Maestría en Ciencias
Ingeniería Eléctrica

Por:

Rosa María Córdova Canela

Ingeniero en Comunicaciones y Electrónica
Centro Universitario de Ciencias Exactas e Ingeniería,
Universidad de Guadalajara 1988-1993

Becario del CONACYT, expediente no. 72343

Director de Tesis
Dr. Antonio Ramírez Treviño

CINVESTAV del IPN Unidad Guadalajara, Noviembre de 1998

Agradecimientos

A Dios por darme salud y fortaleza.

A mi Papá Fernando Córdova y Mamá Rosa Ma. Canela por su confianza, fé y amor que siempre han depositado en mí.

A mis hermanos Fernando y Quetzalli por su incondicional cariño y apoyo.

A mi Abuelita Concha y Abuelito Ramón, que siempre me inculcaron el amor por la vida y al estudio.

A mi Novio Roberto por su amor, comprensión, ayuda y ternura.

A mi Asesor de Tesis el Dr. Antonio Ramírez Treviño por su ayuda, sencillez, amistad y amor a la docencia.

A la Dra. Ofelia Begovich y Dr. Luis Ernesto Mellado por el tiempo prestado a la revisión de esta tesis.

A todo el plantel del CINVESTAV-GDL: Administrativo, Docente y Estudiantil, por su ayuda, amistad y gran calidad humana.

A Elvia Ruth, José Antonio, Fabio, Bernardo M., Bernardo H., Juan Carlos, Carlos A., Carlos M. por todos los bellos y duros momentos que pasamos en el transcurso de la maestría.

A todas las persona que de una manera u otra contribuyeron en la culminación de esta etapa de mi vida.

Al CONACYT y CINVESTAV-MEXICO.

Índice General

Simbología más utilizada

Σ	Alfabeto
σ	Conjunto de cadenas
w	Cadena de símbolos
$\overrightarrow{\sigma}$	Vector de Parik
$\mathcal{L}$	Lenguaje
$\mathcal{P}$	Conjunto Potencia
$\|T\|$	Cardinalidad de un Conjunto
M_0	Marcado inicial en la Red de Petri
$\mathbb{R}$	Conjunto de números Reales
$R(\mathcal{N}, M_0)$	Grafo de Alcanzabilidad de la Red de Petri
λ	Función de etiquetado para los actuadores
φ	Función de etiquetado para los sensores
ε, ϵ	Cadena vacía
$\|\|$	Operador Composición Concurrente
AF	Autómata Finito
RP	Red de Petri
RPI	Red de Petri Interpretada
RPT	Red de Petri Temporizada
GRPI	Red de Petri Generadora de Lenguaje
CPO	Conjunto Parcialmente Ordenado
Frames	Estructuras de los Objetos
FIFO	Lista, primero en entrar primero en salir
GERT	Evaluación Gráfica y Técnica de Revisión
LIFO	Lista, último en entrar primero en salir
OL	Optimalidad Local
PERT/ CMP	Evaluación de Proyectos y Revisión de Programas/ Método de la Ruta Crítica
POO	Programación Orientada a Objetos
SCD	Capa de diseño en KAPPA-PC para Control Supervisorio
SED	Sistema de Eventos Discretos
SED's	Sistemas de Eventos Discretos
SLL	Subsistema Lugar-Lugar
SMF	Sistema de Manufactura Flexible

Capítulo 1

INTRODUCCION

En la vida cotidiana, en el desarrollo de las industrias o empresas, un conjunto determinado de actividades contribuyen al cumplimiento de sus metas u objetivos, por ejemplo, en un banco se encuentran cajeros, clientes, asesores, gerentes, etc, los cuales intervienen en el desarrollo de una *serie de actividades*, entre ellas: llenar depósitos, retiros de efectivo, manejo de cuentas, etc. Se puede apreciar que en en el banco, existen ciertas actividades que demandan los clientes, las cuales deben ser realizadas por los cajeros. Por ejemplo, si en un momento dado, hay dos cajeros y tres clientes, la atención a los clientes podría ser de la siguiente manera, un cliente estaría pagando un servicio, otro cobrando un cheque y el tercero estaría en espera, puesto que no hay cajeros disponibles. En la Figura 1.1, se muestra una gráfica del comportamiento del banco. El banco[Banks 84] pertenece a la clase de los Sistemas de Eventos Discretos (SED), donde los valores $x_1 - x_5$ son los diferentes estados que puede tener el banco (o sistema), el cambio de un estado a otro se produce por las siguientes actividades {pago, retiro, desocupado} los cuales son llamados eventos.

Se puede ver que los SED's están descritos por los *estados* que son una colección de variables numerables a las cuales se les asigna algún valor, por los *eventos* que son ocurrencias instantáneas que permiten cambiar el estado del sistema y que además son diferenciables y numerables entre sí, y por los *recursos* que son los medios necesarios para llevar a cabo alguna actividad. En el caso de la descripción general de un banco, los estados son el número de cajeros ocupados, el número de clientes esperando en la fila o que están siendo atendidos, el número de clientes que llegan al banco, etc; los *eventos* son el inicio o finalización de la atención al cliente por los cajeros, la llegada de un cliente al banco, etc.; algunos recursos son alguna computadora, impresora, etc.; se puede ver en la Figura 1.1 que el cambio de estados no depende del tiempo, sino de la ocurrencia de los eventos, en la Figura 1.2 se muestra el comportamiento del banco en donde el cambio de estados de los clientes que esperan en la fila depende del tiempo de atención por los cajeros.

Un Sistema de Eventos Discretos [López Mellado 97] se define a continuación:

Definición 1.1 *Los Sistemas de Eventos Discretos (SED) son aquellos que están compuestos por elementos que manejan entidades discretas, es decir numerables y diferenciables entre sí. Su funcionamiento está caracterizado por una sucesión finita o infinita de estados estables delimitados por eventos que ocurren, generalmente, de manera asíncrona.*

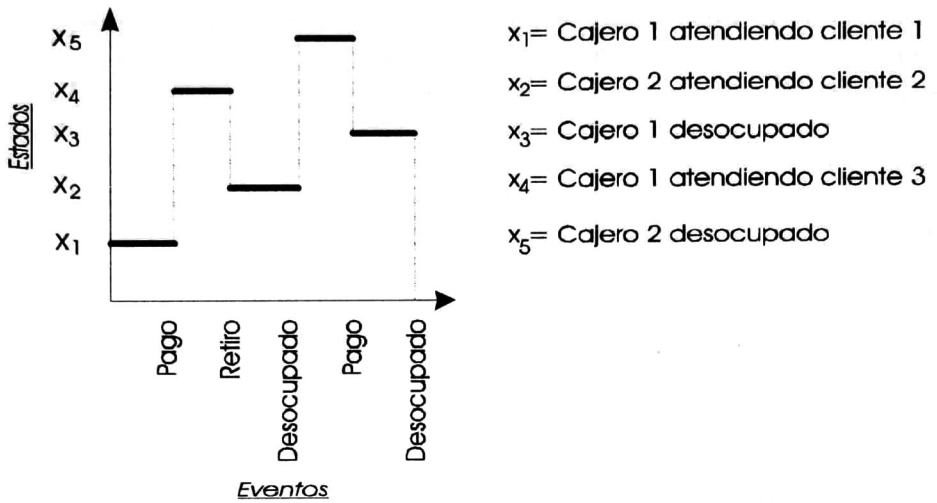


Figura 1.1: Evolución de los Estados de un Banco

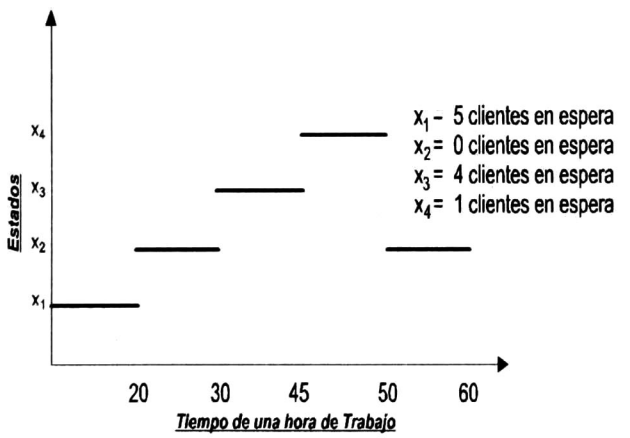


Figura 1.2: Evolución de estados del Banco contra Tiempo.

1.1 Importancia de los sistemas de eventos discretos en la industria

Los sistemas de eventos discretos en la sociedad actual se pueden encontrar en la mayoría de las industrias, ejemplo de ellos son: sistemas de manufactura flexible (inyección de plásticos, elaboración de componentes electrónicos, discos ópticos,etc), sistemas de tráfico (aviones, trenes,etc), sistemas de computación, protocolos de comunicación, etc. Últimamente, debido a la creciente competencia en el mercado de consumo, se ha incrementado la necesidad de ofrecer a los clientes, mejores productos, y por lo tanto, la necesidad de optimizar todas las actividades que se desarrollan en las industrias, con el fin de disminuir costos y poder ofrecer sus productos en el menor tiempo, con un precio competitivo y con la mayor calidad posible.

Las herramientas formales para representar SED's, permiten que el análisis, simulación y síntesis de dichos sistemas se faciliten, porque estas contribuyen en la prevención y disminución del porcentaje de error al aplicar las nuevas soluciones sobre el sistema real; economizando tiempo, dinero y esfuerzo.

1.2 Modelos de sistemas de eventos discretos

En la actualidad existen diferentes formalismos o herramientas para modelar SEDs los cuales pueden clasificarse de acuerdo a los elementos que representan, tales como: actividades, estados, tiempo asociado a los eventos, la finitud en el número de estados, etc. Enseguida se presentan dos clasificaciones para los modelos para SED y las herramientas formales de modelado para cada tipo.

Los modelos de SED pueden ser clasificados según [Giua 92] como:

- Modelos de Eventos Discretos Lógicos o sin temporizar, en los cuales no se toman en cuenta los tiempos de ocurrencia de los eventos y solamente se considera el orden en el cual ocurren.
- Modelos de Eventos Discretos Temporizados, donde se toma en cuenta los tiempos en que ocurren los eventos. Estos a su vez se pueden clasificar:
  - No Estocásticos: si el tiempo del evento es conocido a priori.
  - Estocásticos: si el tiempo asociado al evento no es conocido a priori debido a retardos variables u ocurrencias aleatorias del evento.

En la Figura 1.3 se muestra un cuadro referente a la clasificación anteriormente descrita, ejemplificando algunas herramientas de modelado para cada caso en particular.

En la Figura 1.4 se muestra otra posible clasificación de las herramientas de modelado cuando se modelan las actividades y los estados, los cuales son:

- Modelos de Eventos Discretos que representan Actividades. En los cuales el modelo describe la acción que se realiza entre dos estados.
- Modelos de Eventos Discretos que representan Estados. En los cuales el modelo describe solamente las acciones que desempeña cada componente del sistema para cada estado.

TEMPORIZADAS		*SIN TEMPORIZAR*
ESTOCASTICAS	*NO ESTOCASTICAS*	
REDES DE PETRI ESTOC. REDES DE COLAS CADENAS DE MARKOV GMSP/SIMULACION	REDES DE PETRI TEMP. FRAMES Y REGLAS	MAQUINAS DE TURING AUTOMATAS DE APILAMIENTO FRAMES Y REGLAS AUTOMATAS FINITOS TABLAS DE ESTADO REDES DE PETRI

Figura 1.3: Clasificación de herramientas de modelado para SED

	Representa Estados	*Representa Actividades*
FINITOS	REDES DE PETRI GRAFOS REDUCIDOS TABLAS DE ESTADO AUTOMATAS FINITOS	REDES DE PETRI GERT/PERT
NO FINITOS	REDES DE PETRI MAQUINAS DE TURING AUTOMATAS DE APILAMIENTO FRAMES Y REGLAS	REDES DE PETRI FRAMES Y REGLAS

Figura 1.4: Tipos de herramientas de modelado para SED según estados y actividades

1.3 Diferentes herramientas formales para representar sistemas de eventos discretos

De las herramientas mostradas con anterioridad en las Figuras 1.4 y 1.3, se describe brevemente algunas de ellas de acuerdo con la siguiente clasificación: basadas en estados, basadas en actividades e híbridas. A continuación se resumen las principales características de esta clasificación.

- **Técnicas basadas en Estados**. La representación de sistemas discretos en representación de estados tiene dos técnicas: tablas de estados y máquina de estados o grafo reducido.

  - *Tablas de estado.* La representación tabular representa toda la información del sistema en una tabla. Por lo general, la tabla tiene $2^n + m$ columnas, donde n representa las entradas al sistema y m las salidas del sistema. Conforme aumenta el número de variables en la entrada del sistema se origina un rápido crecimiento en las columnas de la tabla. Es decir, se incrementa el número de columnas y estados exponencialmente, por lo cual el manejo de la tabla se ve limitada.
  - *Grafo reducido.* De cierta manera, la representación del sistema apoyado en grafo reducido mejora la representación tabular en cuanto a claridad y compacidad, puesto que: permite conocer si un estado se alcanza, checando si éste se presenta en el grafo, permite verificar propiedades, presenta el inconveniente de una explosión combinatoria si el número de estados es muy grande. Por lo tanto, se limita a sistemas poco complejos en los que la cantidad de situaciones de paralelismo no es relevante.[1]

- **Técnicas basadas en Actividades**. Entre estas técnicas se encuentran los PERT y los GERT.

  - *PERT.* Las técnicas PERT/CMP representan las actividades a través de los arcos, todas las actividades se deben realizar y por lo tanto su utilización se limita a sólo una clase restringida de problemas, que se caracterizan por que no pueden representar ciclos y decisiones. Este método permite análisis para determinar tiempos de ejecución.
  - *GERT.* Para compensar las limitaciones del PERT surgen otro tipo de redes llamadas GERT igualmente representan las actividades en los arcos y con las ventajas de que se pueden representar ciclos, decisiones y sincronizaciones, permitiendo así, una capacidad de modelado mayor. Se pueden realizar análisis de tiempos de finalización, no permite análisis de propiedades y no hay metodologías de modelado.

- **Técnicas Híbridas**. En estas técnicas están las Redes de Petri que permiten modelar, analizar, simular y controlar sistemas discretos:

[1] El grafo reducido y la tabla de estado son descripciones de Autómatas Finitos (AF).

- *Redes de Petri.* Las *RP* son ampliamente utilizadas, debido a las características surgidas de su naturaleza gráfica y su soporte matemático, además de la claridad y la facilidad en la representación de comportamientos complejos que incluyan secuencias, concurrencia, paralelismo, sincronizaciones, intercambio de información y elección de alternativas.

  En un principio, las redes de Petri fueron utilizadas en protocolos de comunicación y rápidamente fueron aplicadas en otras áreas tales como los sistemas de manufactura, informática, robótica, inteligencia artificial, computación y en la automatización.

  En este formalismo se han desarrollado técnicas de análisis cualitativo que permiten detectar entro otros problemas, bloqueos y acumulación de material o de información, sin especificar la duración de las actividades representadas en el modelo. En general, las redes de Petri han demostrado ser una herramienta de gran utilidad en el modelado, análisis, simulación y control de diferentes tipos de sistemas, entre los que se encuentran principalmente los Sistemas de Manufactura Flexible.

- *Técnicas basadas en Inteligencia Artificial.* Dentro del campo de la Inteligencia Artificial (IA) existen variados esquemas para la representación del conocimiento, entre éstos se pueden mencionar las redes semánticas, los sistemas de producción, la lógica proposicional, los frames y reglas, etc.

  * *Sistemas de Producción*: En los sistemas de producción la base de conocimientos es llamada memoria de trabajo, y tienen una memoria separada para las reglas($premisa_1 \wedge premisa_2 ... \longrightarrow acción_1 \wedge acción_2 ...$), el sistema calcula el subconjunto de reglas donde las premisas satisfagan el contenido actual de la memoria trabajo, luego decide cuales reglas deben ser ejecutadas y aplica las acciones correspondientes sobre el sistema. Ejemplos: OPS-5, CLIPS, SOAR.
  * *Redes semánticas*: Este tipo de técnica, propone a los objetos como nodos en un grafo, todos ordenados de acuerdo a una determinada jerarquía donde la unión entre los nodos representa una relación binaria. Ejemplos de éste son: SNEPS, NETL, Conceptual Graphs.
  * *Sistemas de Programación Lógica.* En estos sistemas se elaboran algoritmos que a través de una serie de sentencias lógicas se genera información para el control del proceso de inferencia. Ejemplo: PROLOG.
  * *Frames y Reglas.* Este tipo de esquema genera una estructura sobre la cual se representan nuevos datos en términos de conceptos adquiridos a través de experiencias previas. La desventaja de utilizar éste tipo de aproximación es que no posee una herramienta formal para obtener propiedades de los modelos.

En el presente trabajo las herramientas de modelado para representar SED que se analizan y comparan son las Redes de Petri y los Autómatas Finitos, por lo que enseguida se muestra el modelo de un sistema de llenado de un tanque representado con RP y AF.

Ejemplo 1.1 *En la figura1.5 se presenta un sistema de llenado de un tanque, consta de una válvula que puede tomar los valores de abierto y cerrado, dos sensores: nivel alto y nivel bajo*

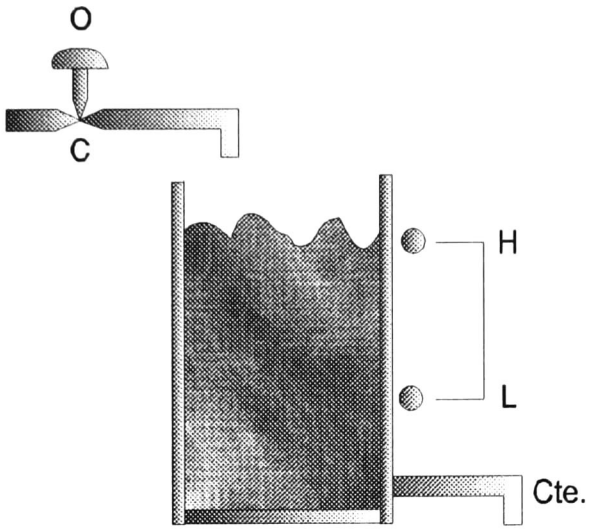


Figura 1.5: Sistema de Llenado de un Tanque

y una salida de flujo constante. Se presentan los modelos que representan el comportamiento del tanque utilizando a los Autómatas finitos y las Redes de Petri.

1.4 Objetivos y Metas de este trabajo

El objetivo central de este trabajo es el de estudiar dos de las herramientas para describir y analizar SED, como son las RP y los AF. En especial, se estudiarán los problemas de modelado y control de SED.

Como metas de este trabajo se realizará una comparación de las bondades de ambas herramientas, concluyendo qué ventajas se tiene en cuanto a la complejidad de los algoritmos de análisis y el tipo de sistemas que modelan, en particular se hará lo siguiente:

- Comparar las RP y los AF cuando son utilizados para *modelar* SED. A partir de este trabajo se logró obtener el siguiente resultado: Una metología de modelado para RPI.
- Comparar las RP y los AF cuando son usados para *controlar* SED. A partir de este trabajo se obtuvo el siguiente resultado: utilizando un nuevo concepto de controlabilidad en RPI se generó un Algoritmo de Búsqueda por anchura para conocer la controlabilidad.
- Además, se construirá un simulador gráfico para Control Supervisorio en SMF, utilizando un software denominado Kappa-PC, tomando como herramienta matemática de apoyo a las RP.

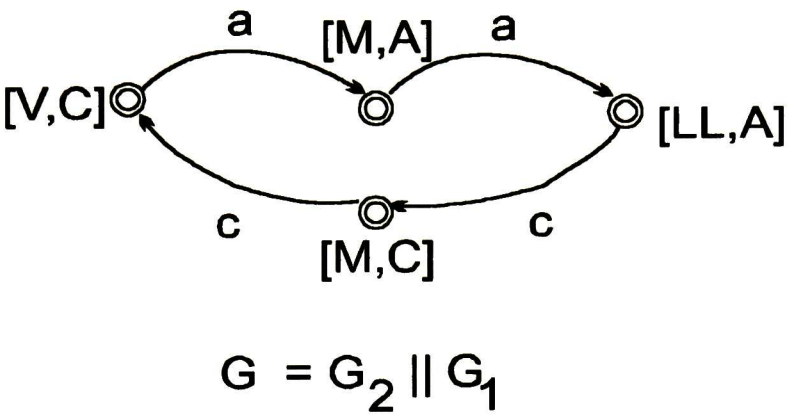


Figura 1.6: Modelo del Tanque mediante Grafo Reducido

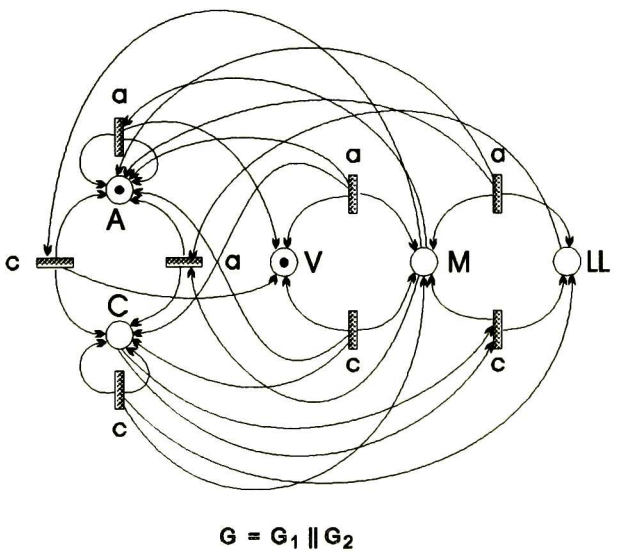


Figura 1.7: Modelo del Tanque con RP

1.5 Organización del trabajo

El siguiente trabajo se organiza de la siguiente manera, el Capítulo dos presenta a los Lenguajes Formales, Autómatas Finitos y las Redes de Petri. El Capítulo tres muestra algunas técnicas de modelado de SED usando AF y RP, se propone una técnica alternativa de modelado usando RP. En el Capítulo cuatro se presentan las técnicas de control Supervisorio usando RP y AF, además se propone un algoritmo para conocer la controlabilidad del sistema en RPI. En el Capítulo cinco se presentan algunos resultados sobre optimalidad local. El Capítulo seis muestra un simulador de SMF al cual se le agrega una capa de software que implementa al control Supervisor. Finalmente las conclusiones y trabajo futuro son presentadas.

Capítulo 2

Herramientas de modelado y análisis

2.1 Introducción

De entre las diferentes herramientas que existen para modelar sistemas de eventos discretos, utilizaremos dos de las más comunes que son los lenguajes regulares y las redes de Petri. Los lenguajes regulares son equivalentes a autómatas finitos y muchas de las propiedades se obtienen utilizando estos últimos. Por otra parte las RP nos permiten modelar lenguajes regulares, algunos libres de contexto y otros sensitivos a contexto. En primera instancia los autómatas finitos son un subconjunto de las RP y con estudiar sólo a éstas sería suficiente, pero resulta que algunos problemas son resueltos con una mejor eficiencia en RP o viceversa. Por tanto este capítulo presenta ambas herramientas y en capítulos posteriores se hace una comparación de el uso de ellas en el área de SED.

2.2 Autómatas finitos y lenguajes formales

Los autómatas finitos representan los estados que alcanza un sistema y las secuencias de eventos que ocurren para alcanzar dichos estados. Por su parte, los lenguajes formales capturan las secuencias de eventos que ocurren en el sistema. Tomando las secuencias de eventos etiquetados que se realizan en un autómata finito, se aprecia que éstas representan un lenguaje formal.

A continuación se presentan las definiciones de los autómatas finitos y los lenguajes regulares.

2.2.1 Definiciones básicas para Lenguajes Regulares

Definición 2.1 *Un alfabeto Σ es un conjunto finito de elementos llamados símbolos.*

Los símbolos pueden ser letras, números o cualquier etiqueta que permita distinguir entre sí los elementos del alfabeto.

Definición 2.2 *Una secuencia, palabra o cadena σ es la yuxtaposición de símbolos de un alfabeto.*

Si $\Sigma = \{a, b, c\}$ entonces una secuencia, utilizando estos símbolos, es $\sigma_1 = bcabbcaaa$. Nótese que la secuencia de símbolos puede ser infinita o vacía (una secuencia sin símbolos). La secuencia vacía se denota por ε o por σ_0.

Definición 2.3 *Un lenguaje $\mathcal{L}$ es un conjunto de cadenas de símbolos tomados de algún alfabeto Σ.*

Si Σ es un alfabeto, entonces Σ^* denota a todas las cadenas de símbolos de Σ.

Definición 2.4 *Sea $\Sigma, \mathcal{L}$ conjuntos finitos de Símbolos, y $\mathcal{L}^1$ y $\mathcal{L}^2$ conjuntos de cadenas de Σ^* La concatenación de $\mathcal{L}^1$ y $\mathcal{L}^2$, es denotada como $\mathcal{L}^1\mathcal{L}^2$, y es el conjunto $\{xz|$ x está en $\mathcal{L}^1$ y z está en $\mathcal{L}^2\}$. Es decir, las cadenas de $\mathcal{L}^1\mathcal{L}^2$ están formadas mediante la selección de una cadena $\mathcal{L}^1$ seguida de una cadena $\mathcal{L}^2$ en todas las combinaciones posibles. Defínase $\mathcal{L}^0 = \{\varepsilon\}$ y $\mathcal{L}^i = \mathcal{L}\mathcal{L}^{i-1}$ para $i \geq 1$. La cerradura de Kleene (o simplemente cerradura) de $\mathcal{L}$, denotada como $\mathcal{L}^*$, es el conjunto*

$$\mathcal{L}^* = \bigcup_{i=0}^{\infty} \mathcal{L}^i$$

y la cerradura positiva de $\mathcal{L}$, denotada por $\mathcal{L}^+$, es el conjunto

$$\mathcal{L}^+ = \bigcup_{i=1}^{\infty} \mathcal{L}^i$$

Es decir, $\mathcal{L}^*$ denota palabras construidas mediante la concatenación de cualquier número de palabras de $\mathcal{L}$. $\mathcal{L}^+$ es lo mismo que $\mathcal{L}^*$, sólo que ahora se excluye el caso de cero palabras, cuya "concatenación" se define como ε.

Nótese que no se hará ninguna distinción entre Σ como un alfabeto y Σ como un lenguaje de cadenas de longitud 1.

Los lenguajes regulares son construídos usando expresiones regulares, las cuales se describen a continuación.

Definición 2.5 *Para un alfabeto Σ una expresión regular se define recursivamente como se menciona a continuación.*

La cadena vacía es una expresión regular

Un símbolo de Σ es una expresión regular

Si r_1 y r_2 son expresiones regulares, entonces la unión, la concatencación y la cerradura de Kleene son expresiones regulares ($r_1 \cup r_2$, $r_1 r_2$,r^*)

Teorema 2.1 *Un lenguaje $\mathcal{L}$ es regular si todas sus secuencias se pueden representar por expresiones regulares.*

Antes de continuar es importante mostrar las relaciones de equivalencia entre los lenguajes regulares y los AF, por lo tanto, es necesario introducir la definición de Autómatas Finitos los cuales son otra forma de representar a los lenguajes regulares.

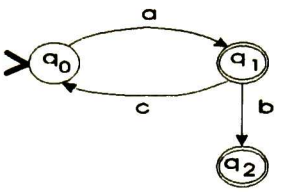


Figura 2.1: Autómata Finito

2.2.2 Autómatas Finitos

Definición 2.6 *Un autómata finito es la quintupla* $G=(Q, \Sigma, \delta, q_o, F)$ *donde:*

- $Q = \{q_0 ... q_n\}$ *Es el conjunto finito de elementos llamados estados*
- Σ *Es un alfabeto*
- q_0 *Es el estado inicial*
- $F \subseteq Q$ *Es el conjunto de estados finales*
- $\delta : Q \times \Sigma \rightarrow Q$ *Es una función parcial de transición de estado.*

Ejemplo 2.1 *Se tiene la siguiente descripción de un AF, el cual tiene tres estados: el estado inicial y dos estados finales,*

$Q = \{q_0, q_1, q_2\}$
$\Sigma = \{a, b, c\}$
q_0 *es el estado inicial*
$F = \{q_1, q_2\}$
$\delta(q_0, a) = q_1, \qquad \delta(q_1, c) = q_0, \qquad \delta(q_1, b) = q_2$
el autómata se puede representar por su diagrama de transición (grafo etiquetado), como se observa en la figura 2.1.

La función de transición indica cómo se alcanza un estado cuando el símbolo asociado a un arco ocurre. En el ejemplo se tiene $\delta(q_0, a) = q_1$, *esta notación se puede extender a palabras, en este caso caso* $\delta(q_0, ab) = q_2$.

Entonces, el lenguaje $\mathcal{L} \subseteq \Sigma^*$ reconocido por un autómata finito es:

$$\mathcal{L} := \{s \in \Sigma^* | \delta(q_o, s) \in F\}$$

y se dice que $\mathcal{L}$ es reconocido o aceptado por el autómata finito.

2.2.3 Equivalencia entre autómatas finitos y lenguajes regulares

Teorema 2.2 *(Teorema 2.4 [Hopcroft 79])Si un lenguaje* $\mathcal{L}$ *es reconocido por un AF entonces* $\mathcal{L}$ *es regular.*

Teorema 2.3 *(Teorema 2.3 [Hopcroft 79])Si un Lenguaje es regular entonces es reconocido por un AF.*

Estos dos teoremas son muy importantes, ya que cuando se tiene la equivalencia entre AF y LR, las dos perspectivas facilitan y complementan el análisis de las propiedades de los lenguajes regulares o AF. Enseguida se presentan las propiedades que poseen los lenguajes regulares.

2.2.4 Propiedades de los lenguajes regulares

Unión, Concatenación y Cerradura de Kleene

Teorema 2.4 *(Teorema 3.1 [Hopcroft 79]) Los conjuntos regulares son cerrados con respecto a la unión, concatenación y cerradura de Kleene.*

Complemento

Si $\mathcal{L} \subseteq \Sigma^*$ es un lenguaje regular, entonces $\Sigma^* - \mathcal{L}$ es un lenguaje regular.

Intersección

Si $\mathcal{L}_1 \subseteq \Sigma^*$ y $\mathcal{L}_2 \subseteq \Sigma^*$ lenguajes regulares, entonces $\mathcal{L}_1 \bigcap \mathcal{L}_2$ también es regular.

La demostración se puede tomar de $\mathcal{L}_1 \bigcap \mathcal{L}_2 = \overline{\overline{\mathcal{L}_1} \bigcup \overline{\mathcal{L}_2}}$, en donde la barra denota el complemento con respecto a un alfabeto que incluya a los alfabetos de $\mathcal{L}_1, \mathcal{L}_2$. La cerradura de la intersección se concluye de acuerdo a la cerradura de la unión y del complemento.

Proyección

Sea $\Sigma_1, ..., \Sigma_n$ alfabetos y $\Sigma = \Sigma_1 \bigcup ... \bigcup \Sigma_n$. Dada una cadena $w \in \Sigma^*$, la proyección (o restricción) de w sobre Σ_i es la cadena $w \uparrow_i$ obtenida con w borrando todos los símbolos no pertenecientes a Σ. Formalmente el operador proyección es un mapeo de Σ^* a Σ_i^*. $(P_i : \Sigma^* \rightarrow \Sigma_i^*)$ y se define como:

Definición 2.7 *Sea* $\mathcal{L}_1 \subseteq \Sigma^*$, $\mathcal{L}_2 \subseteq \Sigma^*, \Sigma = \Sigma \bigcup \Sigma$ *y* $\Sigma \bigcap \Sigma \neq \oslash$, *definimos el operador proyección, como:*

$P_i : \Sigma^* \rightarrow \Sigma_i^*$ $(i = 1, 2)$

donde se cumple que:

- $P_i(\varepsilon) = \varepsilon$
- $P_i(\sigma) = \begin{cases} \varepsilon \ si \ \sigma \notin \Sigma_i \\ \sigma \ si \ \sigma \in \Sigma_i \end{cases}$
- $P_i(s\sigma) = P_i(s)P_i(\sigma)$ *si* $s \in \Sigma^*, \sigma \in \Sigma$

Definición 2.8 *Sea* $\mathcal{L}_1 \subseteq \Sigma_1^*$, $\mathcal{L}_2 \subseteq \Sigma_2^*, \Sigma = \Sigma_1 \bigcup \Sigma_2$ *y* $\Sigma_1 \bigcap \Sigma_2 \neq \oslash$, *definimos el operador proyección inversa, como:*

$P_i^{-1} : \Sigma_i^* \rightarrow \Sigma^*$ $(i = 1, 2)$

Donde, dada una cadena $w \in \Sigma_i^*$, *la proyección inversa de* w *sobre* Σ^* *es un conjunto* w *de cadenas* $q_i (i = 1, 2, \ldots, n)$ *tales que* $P(q_i) = w$.

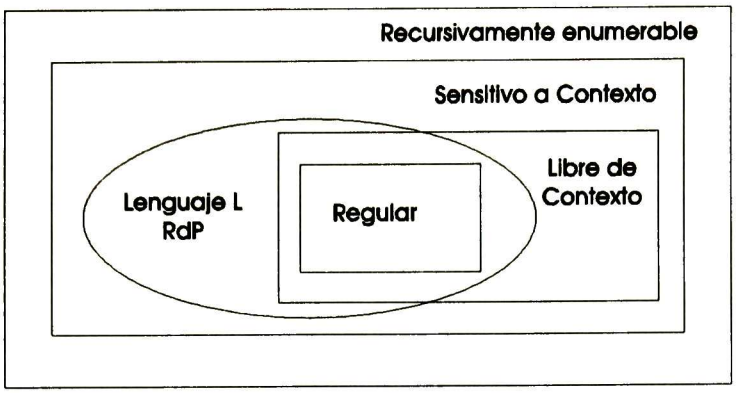


Figura 2.2: Relación entre los distintos Lenguajes Formales.

Enseguida se presenta el operador producto síncrono que será utilizado en capítulos posteriores.

2.2.5 Producto síncrono

Esta operación se trata por separado porque es de suma importancia para el modelado de SED, ya que establece cómo es el comportamiento de dos o más SED cuando actuan en conjunto. Se mencionan los casos siguientes: cuando los alfabetos son disjuntos, iguales y cuando no son iguales.

Producto Síncrono

Definición 2.9 *[Wonham 96]Sea* $\mathcal{L}_1 \subseteq \Sigma^*$, $\mathcal{L}_2 \subseteq \Sigma^*$ *se define el Producto Síncrono de* $\mathcal{L}_1||\mathcal{L}_2 \subseteq \Sigma^*$ *como:*

$\mathcal{L}_1||\mathcal{L}_2 = P_1^{-1}\mathcal{L}_1 \bigcap P_2^{-1}\mathcal{L}_2$

donde $s \in \mathcal{L}_1||\mathcal{L}_2$ *si* $P_1(s) \in \mathcal{L}_1$ *y* $P_2(s) \in \mathcal{L}_2$.

Producto Shuffle

Un caso especial del Producto Síncrono es el Producto Shuffle, en donde $\Sigma_1 \bigcap \Sigma_2 = \emptyset$, es decir, los alfabetos no tienen ningun símbolo común y se calcula igual que el producto síncrono.

Producto Meet

Otro caso especial, es el Producto Meet, donde $\Sigma_1 = \Sigma_2$, es decir, los alfabetos contienen los mismos símbolos. Se calcula como una intersección de lenguajes. $\mathcal{L} = \mathcal{L}_1 \bigcap \mathcal{L}_2$.

A parte de los lenguajes regulares, existen otros tipos de lenguaje que modelan el comportamiento de otros tipos de sistemas, en la figura 2.2 se muestra las clases tradicionales de lenguajes que son regular, libre de contexto, sensitivo a contexto, y lenguajes tipo-0, correspondientes a máquinas de estado finito, autómatas de apilamiento(pushdown), autómatas lineales acotados, y máquinas de Turing respectivamente.

2.3 Redes de Petri

Las Redes de Petri son una herramienta para representar SED, donde los lugares representan las tareas o estados del sistema y las transiciones las actividades ó eventos del sistema, con la particularidad de que se unen por medio de los arcos, sólo lugares con transiciones o viceversa. Si las transiciones se etiquetan, la(s) secuencia(s) de transiciones etiquetadas generan Lenguajes regulares, algunos Lenguajes Sensitivos a Contexto y algunos Libres de Contexto.

En seguida se define la estructura de una Red de Petri, algunas de sus propiedades y se muestran una clasificación para RP ordinarias. También se definen las RP Temporizadas y un Generador de Redes de Petri. Para mayor información acerca las diferentes clases de Lenguajes que pueden generar las Redes de Petri ver [Peterson 81].

2.3.1 Definiciones

Definición 2.10 *Una Red de Petri [Giua 92], [Silva 85] es una estructura*

$$\mathcal{N} = (P, T, \rho, \nu, M) \tag{2.1}$$

donde:

- $P = \{p_1, \ldots, p_n\}$ *es un conjunto de elementos llamados lugares representados por círculos,* $/P/ = m$*;*
- $T = \{t_1, \ldots t_n\}$ *es un conjunto de elementos llamados transiciones representadas por barras,* $/T/ = n$*;*
- $\rho : P \times T \rightarrow \mathbb{N} \bigcup \{0\}$ *es la función de pre-incidencia que especifica los arcos dirigidos de los lugares a las transiciones.*
- $\nu : P \times T \rightarrow \mathbb{N} \bigcup \{0\}$ *es la función de post-incidencia que especifica los arcos dirigidos de las transiciones a los lugares.*

  Se cumple $P \cap T = \emptyset$ *y* $P \cup T \neq \emptyset$.

  $M : P \rightarrow \mathbb{N} \bigcup \{0\}$, *es una función de Marcado.*

Otra manera de representar a la RP es por $\mathcal{N} = (P, T, F)$ donde P y T tienen el mismo significado que la representación anterior, y F representa los arcos de la red, tanto los de entrada como los de salida. Es decir, un arco $(p_i, t_j) \in F \Longleftrightarrow \rho(p_i, t_j) = 1$; o $\nu(p_j, t_i) = 1$.

Su representación matricial es a través de dos matrices. Sea $|P| = n$ (número de lugares de la red) y $|T| = m$ (número de transiciones de la red)se denominan:

$$\begin{array}{rcll} C^- & = & [c^-_{i\,j}]_{n \times m} & \text{Matriz de pre-incidencia, donde } c^-_{i\,j} = \rho(p_i, t_j) \\ C^+ & = & [c^+_{i\,j}]_{n \times m} & \text{Matriz de post-incidencia, donde } c^+_{i\,j} = \nu(p_j, t_i) \end{array}$$

Una red es pura si ésta no tiene autolazos, i.e., si $\forall t \in T\ [\rho(p,t) \wedge \nu(p,t) = 0]$ para cada lugar p. Si una red es pura las funciones de incidencia pueden ser representadas por una matriz, la *matriz de incidencia* de la red, definida como:

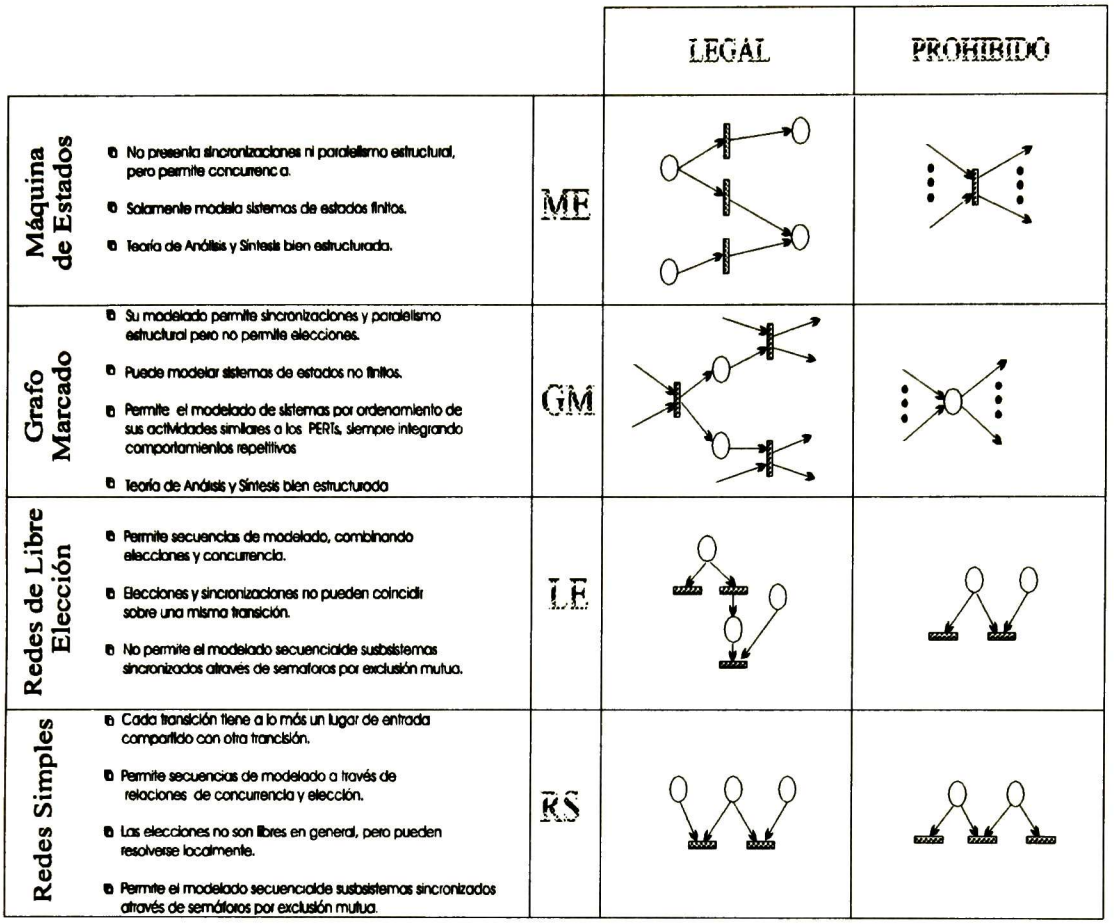


Figura 2.3: Clasificación de RP Ordinarias

$$C(p,t) = C^+(p,t) - C^-(p,t)$$

Los pre-conjuntos y post-conjuntos de una transición t son respectivamente:

$$^\bullet t = \{p \in P | \rho(p,t) \rangle 0\}$$
$$t^\bullet = \{p \in P | \nu(p,t) \rangle 0\}$$

Los pre-conjuntos y post-conjuntos de un lugar p son respectivamente:

$$^\bullet p = \{t \in T | \nu(p,t) \rangle 0\}$$
$$p^\bullet = \{t \in T | \rho(p,t) \rangle 0\}$$

En la Figura 2.3 se muestra una clasificación de las Redes de Petri Ordinarias y debido a que en el presente trabajo se consideran algunas de ellas, enseguida se mencionan sus definiciones formales.

Definición 2.11 *Un Grafo de Estados [Silva 85] (GE) o Máquina de Estados (ME) es una RP tal que:*

$$\forall t \in T \qquad |{}^\bullet t| = 1 \quad y \quad |t^\bullet| = 1$$

es decir, tal que en ella toda transición tiene un lugar de entrada y uno de salida.

Definición 2.12 *Un Grafo Marcado [Silva 85] (GM) o Grafo de Sincronización es una red de Petri tal que:*

$$\forall p \in P \qquad |{}^\bullet p| = 1 \quad y \quad |p^\bullet| = 1$$

Definición 2.13 *Una Red de Libre Elección (RLE) [Silva 85] es una red tal que:*
$\forall p \in P$, *si* $|p^\bullet| > 1$, *entonces* $\forall t_k \in p^\bullet, |{}^\bullet t_k| = 1$
Es decir, si dos transiciones t_i *y* t_j *tienen un lugar de entrada* p *en común, entonces* p *es el único lugar de entrada de* t_i *y de* t_j

Definición 2.14 *Una Red de Petri Simple [Silva 85] es una RP en la que toda transición tiene como máximo un lugar de entrada compartido con otras transiciones.*

Antes de continuar con las propiedades y dinámica de una Red de Petri, se definirá la RP como un Generador de Lenguaje.

Definición 2.15 *Una red de Petri etiquetada (o Red de Petri generadora de Lenguaje) es una 4-tupla* $GRP = (\mathcal{N}, \ell, M_o, F)$ *donde [Peterson 81]:*

- $\mathcal{N} = (P, T, \rho, \nu, M)$ es una estructura de la Red de Petri;
- $\ell : T \rightarrow \Sigma \bigcup \{\varepsilon\}$ es una *función de etiquetado* que asigna a cada transición una etiqueta desde el alfabeto de eventos Σ o asigna una cadena vacía como una etiqueta;
- M es un conjunto de marcados;
- F es un conjunto finito de marcados finales.

Dinámica de la RP

Una transición $t_i \in T$ está habiltada para un marcado M si y sólo si

$$\forall p_j \in P : M(p_j) \geq C^-(p_j, t_i) \tag{2.2}$$

Una transición habilitada puede ser disparada, y su disparo cambia el marcado de M a M' denotado por $M\,[t\rangle\,M'$, donde

$$M'(p) = M(p) - C^-(p_j, t_i) + C^+(p_j, t_i), \forall p \in P$$

Una secuencia de transiciones $\sigma = t_1 t_2 \ldots t_n$ es una secuencia de disparos en $(\mathcal{N}, M_0)$ si y sólo si existe una secuencia de marcados tal que $M_0\,[t_1\rangle\,M_1\,[t_2\rangle\,M_2 \ldots [t_n\rangle\,M_n$, y se dice que M_n es *alcanzable* desde M_0. Extendiendo la notación, quedaría como $M_0\,[\sigma\rangle\,M_n$. El conjunto de marcados alcanzable a partir de M_0 es:

$$M(\mathcal{N}, M_0) = \{M | (\exists \sigma \in \mathcal{L}(\mathcal{N}, M_0)) \wedge M_0\,[\sigma\rangle\,M\}$$

Definición 2.16 *El grafo de Alcanzabilidad asociado a la RP marcada* $(\mathcal{N}, M_0)$ *es un grafo* $R(\mathcal{N}, M_0)$ *en el que cada nodo representa un marcado alcanzable a partir de* M_0 *y cada arco el disparo de una transición. Existe un arco, etiquetado* t_m*, que va desde el nodo que representa* M_i *al que representa* M_j *sii al disparar* t_m *a partir de* M_i *se alcanza* M*, es decir,* $M : M_i [t_m\rangle M_j$ *[Silva 85].*

Una red de Petri es *fuertemente conexa [Giua 92]* si su *grafo de alcanzabilidad* lo es.

Definición 2.17 *Un Lenguaje* $\mathcal{L}$ *es el conjunto de secuencias de disparo* $\mathcal{L}(\mathcal{N}, M_0)$ *(también llamado lenguaje de la red) expresado por:*

$$\mathcal{L} = \{\sigma \,|M_0 [\sigma\rangle \, M \quad \forall M \in R(\mathcal{N}, M_0)\}$$

Sea M el marcado alcanzable desde M_0 disparando una secuencia de transiciones σ. Entonces la siguiente ecuación de estados se satisface: $M = M_0 + C \cdot \vec{\sigma}$, donde $\vec{\sigma} : T \rightarrow \mathbb{N}$ es un vector de enteros no negativos, llamado el *vector contador de disparos.* $\vec{\sigma}\,(t)$ representa el número de veces que la transición t aparece en σ.

El conjunto de marcados M tal que exista un vector $\vec{\sigma}$ que satisfaga la ecuación de estado es llamado el *conjunto potencialmente alcanzable* y es denotado por $PR(\mathcal{N}, M_0)$. Hay que notar que $PR(\mathcal{N}, M_0) \supseteq R(\mathcal{N}, M_0)$.

La *subred de disparo* dada por un vector contador de disparos $\vec{\sigma}$ consiste de todas las transiciones $t \ni \vec{\sigma}\,(t) \geq 0$, y de sus lugares de entrada y salida.

Definición 2.18 *Un lugar p de una red marcada* $\langle \mathcal{N}, M_0 \rangle$ *es implícito [Silva 85] si* $L(\mathcal{N}, M_0) = L(\mathcal{N}', M_0')$*, donde* $\langle \mathcal{N}', M_0' \rangle$ *es la red de Petri marcada que se obtiene removiendo el lugar p y todos sus arcos de entrada/salida. Un lugar p de una red* $\mathcal{N}$ *es estructuralmente implícito si existe un marcado inicial* M_0 *tal que p es implícito en* $\langle \mathcal{N}, M_0 \rangle$*.*

Definición 2.19 *Una red marcada* $\langle \mathcal{N}, M_0 \rangle$ *es sana si y solo si* $M(p) \leq 1 \;\; \forall p \in P \bigwedge \forall M \in R(\langle \mathcal{N}, M_0 \rangle)$

Todos los vectores X tal que $C \cdot X = 0$, $X \geq 0$ son llamados *T-semiflujos*; todos los vectores Y tal que $Y^T \cdot C = 0$, $Y \geq 0$ son llamados *P-semiflujos.*

El soporte de un t-semiflujo X se representa por $\|X\|$; se define como:

$$\|X\| = \{t_i \mid X(t_i) \neq 0\}$$

El vector característico del soporte es: $\left\|\vec{X}\right\| = [x_i]$

$$x_i = \begin{cases} 1 \text{ si } \; t_i \in \|X\| \\ 0 \text{ otro caso} \end{cases}$$

De manera similar se define el soporte de un p-semiflujo representado por $\|Y\|$.

Un p-semiflujo (t-semiflujo) $Y_i(X_i)$ es de soporte mínimo si y sólo si no existe otro p-semiflujo (t-semiflujo) $Y_i'(X_i')$ tal que $\|Y_i'\| \subset \|Y_i\|$ ($\|X_i'\| \subset \|X_i\|$). Un p-semiflujo (t-semiflujo) es *canónico* si el máximo común divisor de sus componentes es 1. Un p-semiflujo (t-semiflujo) es mínimo si y sólo si es canónico y de soporte mínimo.

Definición 2.20 ***T-Componente.*** *Sea $\mathcal{N}=(P,T,\rho,\nu)$ una red de Petri. Un grafo marcado fuertemente conexo $\mathcal{N}'=(P',T',\rho',\nu') \subseteq \mathcal{N}$ es una t-componente de $\mathcal{N}$ si y sólo si $P' =^{\bullet} T' = T'^{\bullet}$ En especial, una t-componente es mínima si $T' = \|X\|$, donde X es un t-semiflujo mínimo de $\mathcal{N}$*

Definición 2.21 ***P-Componente.*** *Sea $\mathcal{N}=(P,T,\rho,\nu)$ una red de Petri. Una máquina de estados fuertemente conexa $\mathcal{N}'=(P',T',\rho',\nu') \subseteq \mathcal{N}$ es una p-componente de $\mathcal{N}$ si y sólo si $T' =^{\bullet} P' = P'^{\bullet}$ En especial, una p-componente es mínima si $P' = \|Y\|$, donde Y es un p-semiflujo mínimo de $\mathcal{N}$.*

2.3.2 Propiedades Dinámicas

Definición 2.22 *Una transición t es viva para un marcado inicial dado M_0 sii existe una secuencia de disparos a partir de cualquier marcado M, sucesor de M_0, que comprenda a t:*

$$\forall M \in R(\mathcal{N}, M_0)$$

Definición 2.23 *Un lugar p es $k-$limitado para M_0 sii existe un número entero k tal que $M(p) \leq k$ para cualquier marcado $M \in R(\mathcal{N}, M_0)$. Se denomina límite del lugar p al menor entero k que verifica la desigualdad anterior.*

Una red marcada $\langle \mathcal{N}, M_0 \rangle$ es:

- **Limitada o k-acotada**, si existe un entero no negativo k tal que $M(p) \leq k < \infty$ para todo lugar p y para todo marcado $M \in R(\mathcal{N}, M_0)$, ésto quiere decir extrapolando a la realidad, que una máquina va a tener un número finito de piezas dentro de ella, o también que un almacén tendrá un mínimo o máximo de piezas, esto tiene que ver con la estabilidad del sistema;

- **Viva** para M_0 si y sólo si todas sus transiciones son vivas para M_0. Hablando de un sistema de manufactura flexible, se refiere a que ninguna máquina permanece ociosa.

  La red marcada $\langle \mathcal{N}, M_0 \rangle$ es *libre de bloqueos* si y sólo si $\exists M \in (\langle \mathcal{N}, M_0 \rangle) : \exists t \in T$ tal que M habilita t. En un sistema, se refiere a que la pieza será terminada y no se quedará estancada en alguna parte del proceso.

- **Reversible o Cíclica**, si el marcado inicial M_0 es alcanzable desde cualquier marcado alcanzable $M \in R(\mathcal{N}, M_0)$.

2.3.3 Propiedades Estructurales

- **Conservativa**, si existe un vector de enteros positivos Y tal que $Y^T \ M_0 = Y^T \ M$ para todo marcado $M \in R(\mathcal{N}, M_0)$, esto tiene que ver con la estabilidad del sistema;

- **Repetitiva**, si y sólo si existe un vector de enteros positivos X tal que $M_0 \cdot X = M \cdot X$ para todo marcado $M \in R(\mathcal{N}, M_0)$.

- Una red $\mathcal{N}$ es **estructuralmente k-acotada** si y sólo si $\forall M_0$ todas las redes marcadas $\langle \mathcal{N}, M \rangle$ son k-acotadas.

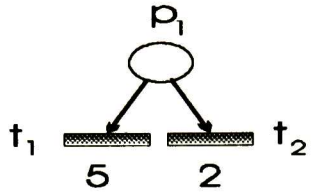


Figura 2.4: Transiciones en conflicto.

- Una red $\mathcal{N}$ es **estructuralmente viva** si y sólo si $\exists M_0$ tal que la red marcada $\langle \mathcal{N}, M_0 \rangle$ es viva.
- Una red $\mathcal{N}$ tiene un **conflicto estructural** cuando un lugar posee más de una transición de salida y todas ellas estan habilitadas y solamente una se puede disparar, tal como se muestra en la Figura 2.4.

2.3.4 Redes de Petri Temporizadas.

En este tipo de red de Petri se introduce el tiempo. Esta clase de modelos se utiliza normalmente para la evaluación de prestaciones (performance).

Definición de RPT.

Definición 2.24 *Una RP temporizada (RPT) es la tercia* $RPT = \langle \mathcal{N}, M_0, D \rangle$, *donde* $\mathcal{N}$ *es una RP,* M_0 *es su marcado inicia y* $D : T \rightarrow \mathbb{R} \geq 0$ *[Ramírez 93a].* $D(t_i) = d_i$ *se llama duración de la transición* t_i *e indica el tiempo que tarda la transición* t_i *en ser disparada.*

Nótese que es preferible trabajar con el modelo que asocia el tiempo a las transiciones que con el modelo que asocia tiempos a los lugares [Ramírez 93a] por ser más natural, puesto que las transiciones expresan las actividades del sistema. En el modelo, una marca puede tener cualquier de los siguientes dos atributos: *disponibles* y *no disponibles.*

Reglas de evolución en una RPT.

Una transición puede dispararse si está habilitada; una transición está habilitada si $M(p_i) \geq \rho(p_i, t_i) \forall p_i \in {}^\bullet t_i$, donde $M(p_i)$ son marcas *disponibles*; el disparo de una transición t_i congela $\rho(p, t_i)$ marcas de cada lugar de entrada p (o sea, no desmarca los lugares, pero si las etiqueta como *no disponibles*, de forma tal, que ninguna otra transición las puede utilizar en su disparo), pero no es sino hasta d_i unidades de tiempo más tarde cuando deposita $Post(p, t_i)$ marcas disponibles en el lugar de salida respectivo p. El marcado M_0 contiene sólo marcas diponibles.

Los tiempos que están asociados a las transiciones son *deterministas.*

Definición 2.25 *Sea* $\mathcal{N} = \langle P, T, F \rangle$ *una red de Petri, la* ***velocidad de disparos***[1] *de una transición* $t_i \in T$ *es el número de disparos de* t_i *por unidad de tiempo cuando el tiempo tiende a infinito. El* ***tiempo de ciclo***[2] *de* t_i *es el inverso de su velocidad de disparo.*

[1] throughput, según denominación inglesa

[2] También llamado tiempo medio de disparos entre una transición

El **tiempo de ciclo** de una **red** con respecto a un t-semiflujo disparable X_i (de números enteros), es el tiempo que tarda en dispararse X_i una sola vez.

Definición 2.26 *Un **schedule factible** S se define por una secuencia de pares ordenados (i_m, τ_m) , donde $\sigma = t_{i1}t_{i2}\ldots t_{im}\ldots$ es una secuencia de disparos realizable en la RP considerada compatible con las ratios de visita de las transiciones $(\overrightarrow{\sigma} = r \cdot v_k, r \in \mathbb{N})$ t_{im} y τ_k es el tiempo de inicio de disparo de la transición t_{im} en σ*

Definición 2.27 *Un **schedule bueno** con respecto a una cota "c" es un schedule que se calcula en tiempo polinomial en $|P| + |T|$ y el tiempo de ciclo de la red que resulta de aplicar dicho schedule es menor que la cota "c"*

Definición 2.28 *El **Schedule óptimo** es un schedule factible cuya ejecución permite maximizar la velocidad de disparo de las transiciones del sistema.*

2.3.5 Redes de Petri Interpretadas

En este trabajo se presentará una definición diferente a la presentada en [Silva 85], puesto que ésta apoyará la construcción de la metodología de aproximación para modelado en RPI presentada en el Capítulo tres.

Antes de redefinir a las RPI es necesario presentar las siguiente definiciones:

Definición 2.29 *Un actuador es una entidad física sobre la cual se ejecutan acciones determinadas para hacer que el SED o elementos de éste realicen cierta operación u tarea (dependiendo a lo que esté asociado el actuador).*

Definición 2.30 *Un sensor como su nombre lo indica es un dispositivo que es sensible al estado del sistema. Los sensores miden el estado del SED.*

Definición 2.31 *Una entrada o señal de los actuadores, es una señal de valor único que hace que el actuador cambie de un valor a otro.*

Definición 2.32 *Una salida o una señal de los sensores es el estado del SED indicado por los sensores (Nótese que una señal de salida es un vector).*

A partir de las definiciones anteriores se puede representar las entradas y salidas utilizando los símbolos de los alfabetos que a continuación se definen.

Definición 2.33 *Sea $\Sigma_{entrada} = \{a_i | a_i$ es una señal de los actuadores$\}$ el conjunto de todas las señales de los actuadores.*

Definición 2.34 *Sea $\Sigma_{salida} = \{s_i | s_i$ es una señal de los sensores$\}$ el conjunto de todas las señales de los sensores*

- Aquí, a los elementos de un alfabeto se les llama símbolos. Por lo que en este caso en particular es equivalente decir símbolo o señal, ya que los símbolos de los alfabetos son las señales de los actuadores y de los sensores.

- $\Sigma^*_{entrada}$ (Σ^*_{salida}) es el conjunto de todas las palabras de entrada (salida) que se pueden formar con los símbolos de $\Sigma_{entrada}$(Σ_{salida}).
- w (s) es una palabra de entrada (salida) y es una sucesión finita de símbolos que pertenecen al alfabeto de entrada $\Sigma_{entrada}$ (salida Σ_{salida}).
- $L_{entrada}$ (L_{salida}) es un lenguaje de entrada (salida) formada por palabras w (s) y puede ser cualquier subconjunto de $\Sigma_{entrada}$ (Σ_{salida}).

Definición 2.35 *Una RP Interpretada es la cuádrupla* $RPI(\mathcal{N}, M_0, \lambda, \varphi)$ *donde:*

- $\mathcal{N} = (L, T, \rho, \nu)$ *es una RP definida anteriormente.*
- M_0 *es el marcado inicial*
- $\lambda : T \rightarrow \Sigma \cup \{\epsilon\}$ *es la función de etiquetación que representa las transiciones que pueden ser habilitadas o deshabilitadas por las señales de los actuadores*
- $\varphi : M \rightarrow \Sigma \cup \{\epsilon\}$ *es una función que representa cuáles sensores se acitvan cuando se alcanza cierto marcado.*

Definición 2.36 *Una transición t es controlable ssi* $\lambda(t) \neq \epsilon$, *es decir que* $\lambda(t) = a \in \Sigma_{entrada}$.

Definición 2.37 *Una transición t es incontrolable ssi* $\lambda(t) = \epsilon$.

Definición 2.38 *Un lugar p es medible ssi* $\varphi(p) \neq \epsilon$, *es decir que* $\varphi(p) = s \in \Sigma_{salida}$

Definición 2.39 *Un lugar p es no medible ssi* $\varphi(p) = \epsilon$

Evolución de una RPI

1. El disparo de una transición controlable t ($\lambda(t) = a, a \in \Sigma_{entrada}$) se realiza siempre que esté habilitada y que la señal a esté presente.
2. El disparo de un transición incontrolable t ($\lambda(t) = a,$) se puede realizar siempre que esté habilitada.
3. El disparo de una transición t controlable ó incontrolable alcanza un marcado que hace que se generen las señales de los sensores asociados con los lugares medibles que fueron marcados. Como en el presente trabajo se limita al uso de las RPI binarias (salvo se indique lo contrario), la función φ presentada en la definición 2.35 se puede expresar como $\varphi : L \times \Sigma_{salida} \cup \{\epsilon\}$. Así, un símbolo de salida $\varphi(L) = [e_i]$ se calcula como:

$$e_i = \begin{cases} \varphi(p_i) & \text{si } M(p_i) = 1 \\ \epsilon & \text{si } M(p_i) = 0 \end{cases}$$

En la figura 2.5 se presenta el esquema de RPI presentado en [Silva 85] y el que se usa en esta tesis. En otros trabajos se extenderá el modelo.

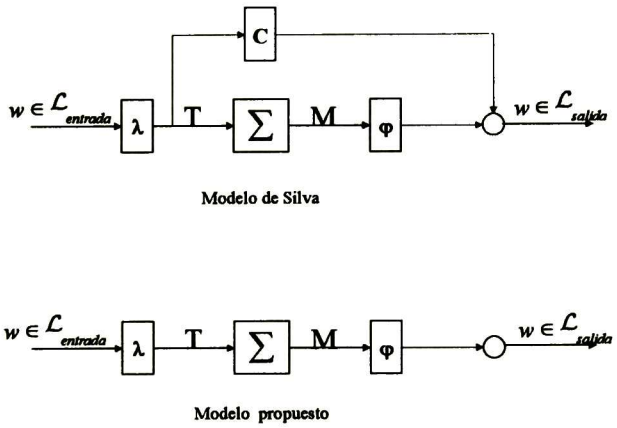


Figura 2.5: Modelos de RPI's

Capítulo 3

MODELADO DE SISTEMAS DE EVENTOS DISCRETOS

3.1 Introducción

En este Capítulo se presentarán dos enfoques para modelar SED, se mostrará la metodología de modelado propuesta por Wonham para AF, y la metodología propuesta por Giua para modelar con RP. Se muestran las características de las metodologías. Finalmente se propone una metodología basada en RP que es más adecuada que las anteriores para modelar SED, ya que los modelos obtenidos son sencillos de interpretar y relacionar con el sistema real, puesto que cada parte del sistema real puede ser diferenciado en el modelo.

3.2 Modelado en Autómatas Finitos

En el campo de los AF, el modelado se realiza bajo dos enfoques diferentes. El primer enfoque [Wonham 96] se basa en modelar cada subsistema de la planta con un AF. Si la planta tiene k subsistemas, entonces se tendrán $G_1 \ldots G_k$ modelos en AF para cada subsistema de la planta. Además esta metodología propone tener una o varias especificaciones por cada subsistema, así pues se tendrán $H_1 \ldots H_k$ modelos de especificación. Para conseguir el modelo de la planta total E se hace el producto síncrono entre los modelos G_i, i.e., $E = \|_{i=1}^{k} G_i$ Para conseguir el modelo total (con la especificación), al modelo E se le incorpora la especificación H_1 mediante el producto síncrono $E_1 = E||H_1$, en seguida se incorporan la especificación H_2 realizando $E_2 = E_1||H_2$ y así hasta conseguir E_k que será el modelado de toda la planta junto con sus especificaciones.

El segundo enfoque calcula la planta E de la misma forma que el anterior, pero crea primero un modelo de especificaciones, a cada H_i le agrega autolazos para incorporar todos los eventos que aparecen en E y no aparecen en H_i, después se crea el modelo de especificación $H = \bigcap_{i=1}^{k} H_i$, finalmente el modelo de la planta se obtiene por $E_k = H \bigcap E$.

El segundo enfoque es el más utilizado, por la familiaridad que se tiene en el uso del operador intersección; además se cuenta también con un programa denominado tct [Wonham 96] con el que se puede calcular el producto síncrono, contiene incluso otras funciones de ma-

nipulación de lenguajes y operaciones que ayudan a sintetizar controladores. Ejemplos de esta metodología se puede encontrar en [Balemi 93] donde se presenta el modelado de un Multiprocesador rápido térmico y se obtiene un modelo monolítico, en [Charbonnier 94] se puede encontrar el modelado para un sistema de manufactura automatizado del AIP[1] y se obtiene un modelo monolítico, en [Lauzon 96] con éste mismo enfoque se modela un sistema de manufactura flexible controlado por una PC y un PLC.

La metodología del primer enfoque es la propuesta por Wonham, la cual se detalla formalmente enseguida.

Si el sistema tiene k subsistemas, entonces se tendrán $G_1 \ldots G_k$ modelos, como se dijo anteriormente $\mathcal{L}(G_i)$ es el lenguaje que genera el subsistema G_i, y $\mathcal{L}_m(G_i)$ es el lenguaje formado por aquellas palabras de $\mathcal{L}(G_i)$ que llevan del estado inicial a algún estado final. El producto síncrono es asociativo, es decir, $E = G_1||G_2|| \ldots G_k = (((G_1||G_2)||G_3) \ldots G_k)$, por tanto, se muestra como obtener el producto síncrono entre dos subsistemas y el lector puede aplicar este algoritmo sucesivamente hasta agotar los G_k modelos. El producto Síncrono estará representado por $G' = G_1||G_2$, donde el sistema resultante genera los lenguajes $\mathcal{L}(G') = \mathcal{L}(G_1)||\mathcal{L}(G_2)$[2] y $\mathcal{L}_m(G') = \mathcal{L}_m(G_1)||\mathcal{L}_m(G_2)$[3]. El producto Síncrono de $G' = G_1||G_2$ se puede calcular de la siguiente forma:

$G' = (Q', \Sigma, \delta', q_0', F')$, donde:

- $Q' = Q_1 \times Q_2$
- $\delta' = \delta_1 \times \delta_2$

  Si $\Sigma_1 \bigcap \Sigma_2 = \emptyset$ entonces $(\delta_1 \times \delta_2)((q_i, q_j), \sigma) := [\delta_1(q_i, \sigma), \delta_2(q_j, \sigma)]$, $\sigma \in \Sigma_1 \backslash \Sigma_2 (\sigma \in \Sigma_2 \backslash \Sigma_1)$

  Si $\Sigma_1 \bigcap \Sigma_2 \neq \emptyset$ entonces δ se define como en el paso anterior y se incluye la siguiente restricción $(\delta_1 \times \delta_2)((q_1, q_2), \sigma) := [\delta_1(q_1, \sigma), \delta_2(q_2, \sigma)]$ Siempre que $\delta_1(q_1, \sigma)!$ y $\delta_2(q_2, \sigma)!$
- $q_0' = [q_{0,1}, q_{0,2}]$
- $F' = F_1 \times F_2$

3.2.1 Ejemplo de Modelado

Tomando el ejemplo del sistema de llenado de un tanque del Capítulo 1. En la Figura 3.1, se muestra el esquema del sistema del tanque que tiene dos sensores uno de nivel alto y otro de nivel bajo, una válvula de entrada todo-nada y una salida de flujo cte.

En la Figura 3.1, en el inciso a) se presenta el modelado del tanque con los tres estados que éste puede tomar: Vacío, Medio y Lleno, en el inciso b) se propone un modelado para la válvula, se incluyen autolazos en los estados de Abierto y Cerrado, que representan la acción de que la válvula permanece constantemente en cualquiera de los dos estados, en el sistema físico, ésto no ocurre, simplemente se abre o se cierra la válvula y en ese estado se queda. En el inciso c) se muestra la composición concurrente o producto síncrono de $G = G_1||G_2$ a partir del cual se obtienen todas las trayectorias posibles de ejecución por el sistema. Donde las funciones de transiciones parciales para el tanque quedan de la siguiente forma:

[1] Atelier Interétablissement Productique, Dauphiné-Savoie, France.

[2] Lenguaje o Comportamiento Cerrado

[3] Lenguaje o Comportamiento Marcado.

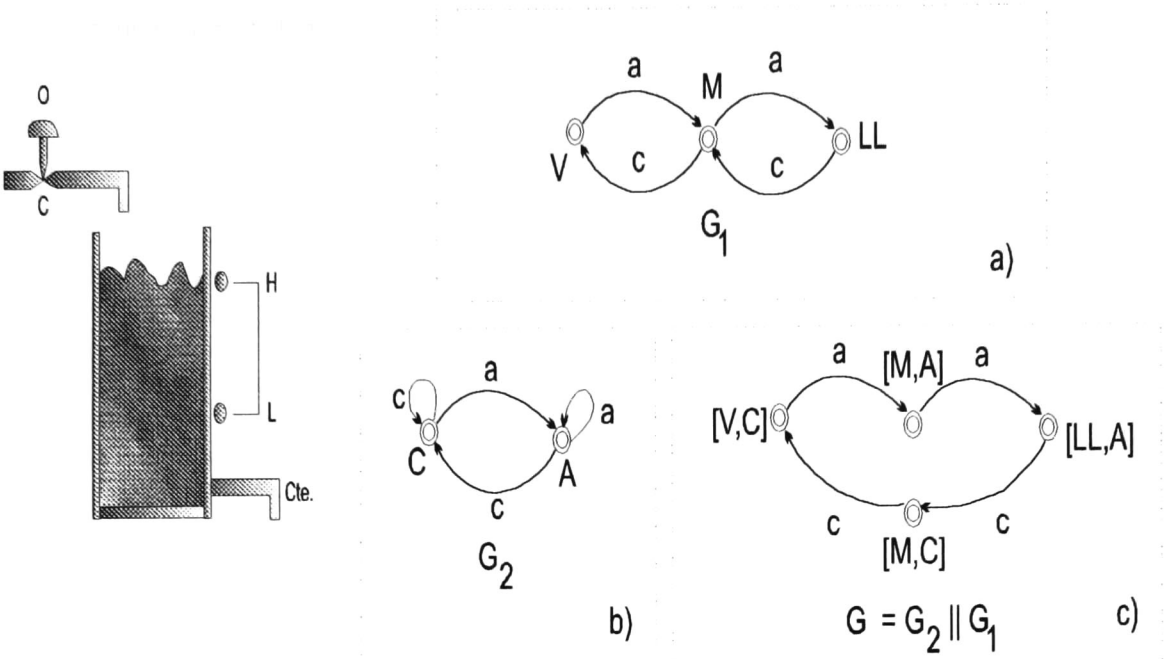


Figura 3.1: Modelo en Autómatas Finitos

$(\delta_1 \times \delta_2)((V,C),a) := [M,A]$
$(\delta_3 \times \delta_4)((M,C),c) := [V,C]$
$(\delta_5 \times \delta_6)((M,A),a) := [LL,A]$
$(\delta_7 \times \delta_8)((LL,A),c) := [M,C]$

3.3 Modelado en Redes de Petri

Varias metodologías han sido utilizadas para el modelado en Redes de Petri sobre sistemas de manufactura, sistemas de computo, protocolos de comunicación, entre otros. Debido a que el trabajo se enfocará a los sistemas de manufactura, enseguida se mencionarán algunas metodologías que se han utilizado para modelar este tipo de sistemas: metodología de lugares de control, metodología de síntesis descendente[4], metodología de síntesis ascendente[5] y metodología de síntesis con técnicas híbridas[6].

En el capítulo III del libro de [Proth 93] se mencionan los principales artículos que proponen metodologías de modelado. A continuación se resume dicho capítulo.

La metodología de Lugares de Control, por Holloway y Krogh es aplicado a los sistemas de manufactura; el sistema debe dividirse en pequeños subsistemas, sobre los cuales se introducen transiciones de entrada para la recepción de partes, transiciones de salida para partes maquinadas y lugares de control (representados por dos círculos concéntricos, y con la re-

[4]En la literatura Inglesa Top-Down.
[5]En la literatura Inglesa Bottom-Up.
[6]En la literatura Inglesa Hybrid synthesis.

stricción de contener una sola marca o ninguna), éstos lugares dependen de señales exógenas provenientes de una parte del sistema la cual se le denomina Sistema de Salida de Toma de Decisiones (O-DMS[7]) donde su marcado inicial depende de la secuencia de operaciones que se desea siga el sistema. Si el marcado, la ubicación de los lugares de control y el modelado de los subsistemas es adecuado, las Redes de Petri obtenidas conservan buenas propiedades de limitación, vivacidad y conservatividad. Proth extiende éste trabajo, proponiendo modelar cada máquina por separado y después crear trayectorias que sigan la secuencia de maquinado por cada pieza, en el orden que deseen ser obtenidas, Proth presenta modelados de almacenes FIFO, Máquinas multi-operaciones, Máquinas de una sóla operación, Máquinas de ensamble, sistemas en Jop-Shop, entre otros. Por lo general crea modelos grandes, sin embargo, se tiene una interpretación física sencilla.

Las técnicas ascendentes obtienen el modelo del sistema, a partir de la creación de pequeños modulos y posteriormente su integración.

Agerwala y Choes-Amphai han sido los pioneros trabajando sobre técnicas ascendentes en sistemas de manufactura, la metodología sugerida es llamada 1-camino de fusión, donde a través de la realización de redes simples o estructuras básicas que describen el sistema, éstas son integradas fusionando lugares comunes entre ellas, con la restricción de que la fusión entre ellos forme p-invariantes. Las redes de petri obtenidas conservan la propiedad de limitación. Narahari y Viswanadhan han hecho trabajos similares orientados a sistemas de manufactura flexible.

Beck, Krogh y Beck proponen otra metodología basado en técnicas ascendentes llamada Trayectorias Elementales Simples compartidas para sintetizar RP vivas y limitadas. Se forman un grupo de circuitos elementales y fusionan lugares y transiciones entre ellos mismos, donde la vivacidad y limitación depende de que el marcado inicial comprenda una marca para cada p-invariante que se tenga.

Las técnicas descendentes proponen elaborar un modelo compacto del sistema al cual se le va agregando más detalle. En éstos métodos comúnmente se pueden encontrar dos esquemas: expansión de lugares o expansión de transiciones.

Valette propone una metodología de refinamiento por transiciones, donde se puede remplazar una transición si cumple con las condiciones de un bloque bien formado[8]. La metodología de Valette puede ser considerado una generalización de los resultados de Bruno y Altman que desarrollaron una teoría de control de redes asíncronas para el modelado de estructuras de control de sistemas digitales, donde se obtienen redes libres de bloqueo. Para síntesis y reducción de grafos marcados han trabajado Johnsonbaugh y Murata, Murata, Koh y Murata, los cuales a través de métodos Serie, paralelo, circuito único, reducción AND-OR y técnicas de expansión encuentran grafos marcados vivos y limitados.

Susuki y Murata presentan una metodología de refinamiento por lugares, donde las propiedades de limitación, vivacidad y ciclicidad se conservan después de cada paso de refinamiento.

La Síntesis de modelado Híbrida es una combinación entre los métodos ascendentes y descendentes, cuyos modelos se restringen a la obtención de Redes de Petri Ordinarias. Zhou, Zhou y DiCesare presentan trabajos formales sobre ésta técnica aplicados en el área de Manufactura, su proceso de diseño se divide en: a) Usar como primer nivel de descripción para la RP el método descendente por refinamiento de lugares y/o transiciones, b) enseguida

[7] Output-Decision Making System

[8] Conocido en inglés como *well-formed block*

un método ascendente donde los lugares de recursos son añadidos. Si el o los recursos son compartidos por diferentes procesos, la síntesis de la RP se complica y no puede ser fácilmente extendida para un caso general.

En otro trabajo, [Giua 92] propone una metodología de modelado para un sistema G utilizando el operador composición concurrente. Esta metodología es relativamente reciente, su construcción es sencilla y genera todas las secuencias posibles a realizar por el SED, al igual que las demás mencionadas, sin embargo, se eligió esta metodología para hacer comparaciones entre las otras dos técnicas presentadas en el trabajo, puesto que Giua enfoca su metodología en el lenguaje en RP, posteriormente se mencionan algunos resultados de Control Supervisorio en SED's modelados en RP, por ejemplo, la existencia de un supervisor. Enseguida se presentará una descripción de dicha metodología.

Sean $G_1 = (N_1, \ell_1, M_{0,1}, F_1)$ y $G_2 = (N_2, \ell_2, M_{0,2}, F_2)$ dos Generadores de Redes de Petri. Cuya Composición Concurrente, es denotada por $G = G_1||G_2$, y el sistema resultante es $G = (N, \ell, M_0, F)$ que genera los lenguajes $L_{L}{}^{9}(G) = L_L(G_1)||L_L(G_2)$ y $L_P{}^{10}(G) = L_P(G_1)||L_P(G_2)$[Tesis Doctoral Giua]

La estructura de G puede ser determinada como sigue. Sea P_i, T_i y $\sum_i$ $(i = 1, 2)$ el conjunto de lugares, conjunto de transiciones y el alfabeto de G_i respectivamente.

- El conjunto de lugares P de N es la unión de el conjunto de los lugares de N_1 y N_2, es decir, $P = P_1 \cup P_2$.
- El conjunto de Transiciones T de N y las correspondientes etiquetas son calculadas como sigue:
  - Sea $b \in \sum_1 \setminus \sum_2 (b \in \sum_2 \setminus \sum_1)$ la etiqueta de una transición $t \in T_1 (t \in T_2)$ Entonces etiqueta una transición b en T con el mismo multiconjunto de entrada y de salida de t.
  - Sea $b \in \sum_1 \cap \sum_2$ etiquetando m_1 transiciones en T_1 y m_2 transiciones en T_2. Entonces $m_1 \times m_2$ transiciones en T serán etiquetadas b . La entrada (salida) de cada una de estas transiciones es la suma de la entrada (salida) de los multiconjuntos de una transición en T_1 y la de una transición en T_2.
  - $M_0 = [M_{0,1}^T M_{0,2}^T]^T$
  - F es el producto cartesiano de F_1 y F_2, es decir, $F = \{[M_1^T M_2^T]^T | M_1 \in F_1, M_2 \in F_2\}$.

3.3.1 Ejemplo de Modelado

Se sigue con el modelado del tanque del ejemplo anterior.

En la Figura 3.2, el subsistema G_1 es la RP que representa la válvula , el subsistema G_2 representa el modelo del comportamiento interno del tanque y el sistema G representa la composición concurrente $G = G_1||G_2$. Esta composición concurrente a partir del marcado mostrado en la Red, genera el lenguaje que puede realizarse en el sistema físico, todos los estados que se presentan en el grafo de alcanzabilidad son realizables físicamente por el Sistema de llenado de Tanque mostrado en la Figura 3.2.

[10] Lenguaje tipo P (cerrado)

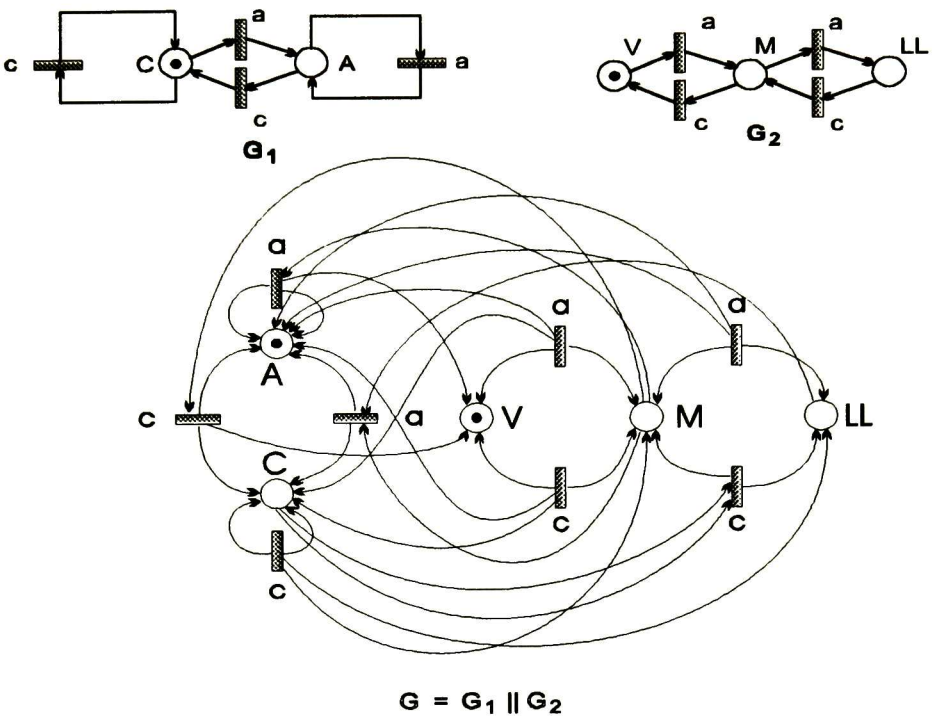


Figura 3.2: Modelo en Redes de Petri

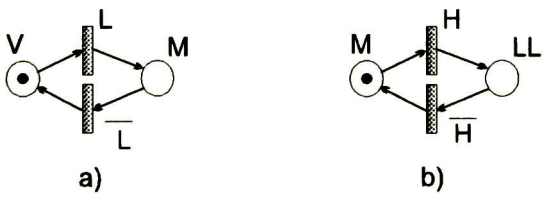


Figura 3.3: Modelos de Sensor: a) bajo nivel b) alto nivel

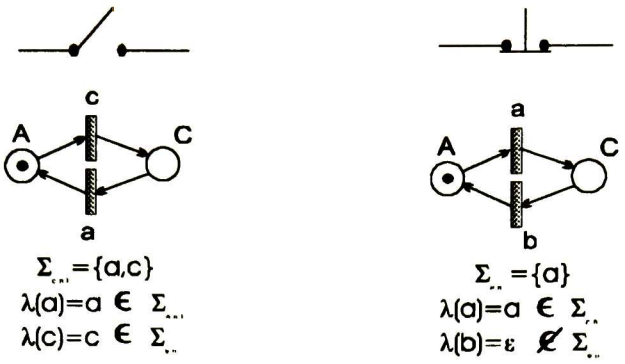


Figura 3.4: Modelos de Actuadores

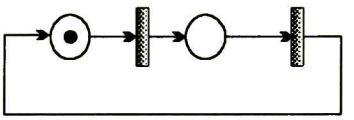


Figura 3.5: RP con Evolución Secuencial

3.4 Modelado en Redes de Petri propuesto

Debido a que un SED se puede ver como un sistema compuesto de tres partes: Sensores, Actuadores y Modelo Interno, la metodología propuesta modela cada parte del sistema por separado y luego los une.

Los Actuadores modelados en Redes de Petri pueden presentar dos tipos de eventos: los controlables e incontrolables, en la Figura 3.4, se muestra dos ejemplos de ellos, donde los eventos controlables e incontrolables son etiquetados a partir de la función λ tal como se definión en el Capítulo 2.

Los Actuadores se clasifican de acuerdo al modelo en Redes de Petri donde cada señal a_i que genera el actuador $Ac_{2\alpha}$ se representa por una transición t_i, dicha transición se etiqueta con la señal a_i. El lugar de salida de t_i será el estado estable que alcanza el actuador cuando se le aplica a_i. La clasificación se realiza:

- Con evoluciones Secuenciales, en Redes de Petri tendríamos, como se muestra en la Figura 3.5:

  Ejemplo 3.1 *Un ejemplo de éste tipo de estructura se encuentra al modelar una válvula, como se muestra en la Figura 3.6.*

- Con evoluciones de Elección, en RP se muestra en la Figura 3.7:

  Ejemplo 3.2 *En la Figura 3.8 se representa a través del tipo de RP con evol. de elección el comportamiento de un interruptor de un polo y tres tiros.*

Figura 3.6: Modelo de un Actuador con Evolución Secuencial

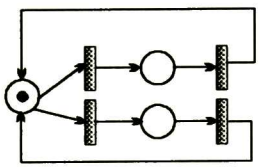


Figura 3.7: RP de Elección.

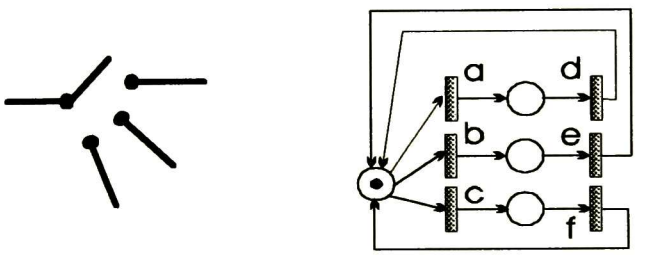


Figura 3.8: Modelo de un Actuador con Evolución de Elección

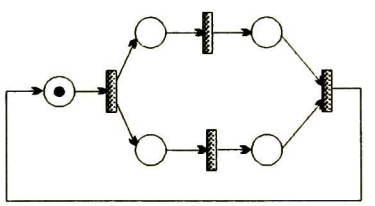


Figura 3.9: RP con Evolución Paralela

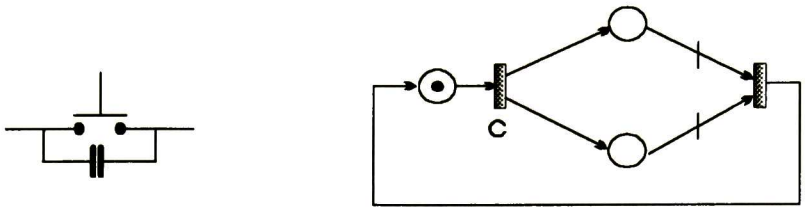


Figura 3.10: Modelo de un Actuador con Evolución Paralela

- Con evoluciones paralelas, se muestra en la Figura 3.9:

Ejemplo 3.3 *Un interruptor puede representar claramente este tipo de comportamiento modelado por la RP como se muestra en la Figura 3.10.*

El modelo Interno puede ser cualquier SED, por ejemplo, almacenes tipo LIFO (Figura 3.11), almacenes tipo FIFO (Figura 3.12), de manufactura flexible (robot, bandas transportadoras, pulidoras, taladros, etc), de protocolo de comunicación, etc. Todos estos sistemas son modelados de acuerdo a cualquiera de las metodologías mencionadas en la sección 3.3 de RP, sin embargo, es necesario para que el modelo se adapte a la representación del modelado propuesto que dicho modelo interno se discretice en estados, esto significa que a cada estado e_i se le represente por un lugar p_i(en este caso, que tenga asociado el estado de cada sensor en el sistema). Si del estado e_i se llega al estado e_j, entonces se pone una transición t_{ij} de p_i a p_j. A t_{ij} se le asigna el *símbolo* (no etiqueta) de los estados de los actuadores que hacen que t_{ij} se realice.

Si t_{ij} no depende de ningún estado de los actuadores, entonces t_{ij} no recibe ningún símbolo.

Enseguida se presentan algunos SED de los cuales ya se obtuvo el modelo en RP, en [Proth 93] se pueden encontrar modelados de SED tales como: sistema de banda-buffer, sistema de transporte, almacén LIFO, almacén FIFO, sistema de máquina de mono-operación, sistema de máquinas de multi-operación, sistema de máquina de ensamblado, sistema de máquinas trabajando en paralelo, Job-Shop, sistema de máquinas de ensamble con dispositivos de reparación, sistema industrial para la inserción de un pistón en una máquina de un vehículo de motor y simulador de un sistema de Manufactura Flexible.

En [López Mellado 97] se pueden encontrar modelados de almacén LIFO, almacén FIFO, problema de los filósofos comensales, sistema de carga y descarga, procesos químicos en tanda, sistema de tráfico de trenes, sistemas tráfico entre 2 y entre 3 vehículos. En [Desrochers 95] se encuentran una variedad de modelos de sistemas de manufactura tales como: buffer, líneas de transferencia: con dos máquinas y un buffer, con tres máquinas y dos buffers, celda de ensamble de un pistón y estación de trabajo de maquinado. Si el lector desea obtener el modelo de algún sistema que no se encuentra en la literatura mencionada, se puede utilizar cualquiera de las metodologías para el diseño del sistema.

En la Figura 3.13, se presenta el modelo del sistema del tanque obtenido a partir de la metodología de modelado propuesto , donde el lenguaje $L(G)$ genera todas la secuencias realizables en el sistema físico.

De acuerdo al ejemplo anterior se pueden enumerar algunas reglas para la unión entre los elementos del sistema:

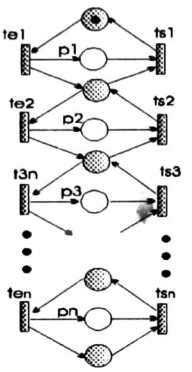


Figura 3.11: Almacén tipo LIFO

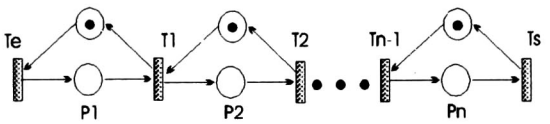


Figura 3.12: Almacenes tipo FIFO

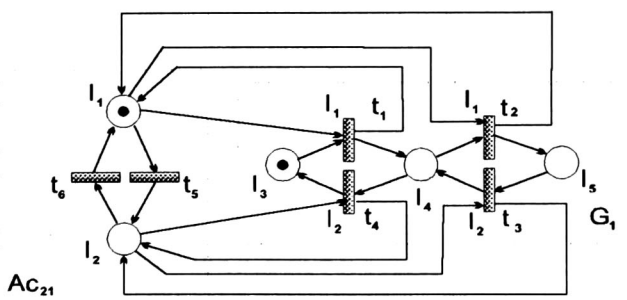


Figura 3.13: Modelo del Sistema con la aproximación propuesta

REGLAS PARA LA UNION DE ELEMENTOS EN LA APROXIMACION PROPUESTA

1. Se procede a especificar las características del modelo del comportamiento interno y los actuadores del sistema.

- El modelo del comportamiento interno se obtiene de la siguiente forma:

  - A cada lugar se le asigna una etiqueta que representa la señal discreta que envía el sensor cuando alcanza dicho estado. De acuerdo a la función φ queda que $\varphi(p) = s \in \Sigma_{salida}$ (medibles) o $\varphi(p) = \varepsilon$ (no medibles), en el ejemplo, $\Sigma_{salida} = \{V, M, LL\}$ y por lo tanto la función quedaría:
    $\varphi(l_3) = V \in \Sigma_{salida}$ Tanque Vacío.
    $\varphi(l_4) = M \in \Sigma_{salida}$ Tanque con nivel Medio.
    $\varphi(l_5) = LL \in \Sigma_{salida}$ Tanque con nivel Lleno.
  - Como se mencionó anteriormente, a cada transición t_{ij} del modelo interno, se le asigna el símbolo de los estados de los actuadores que hacen que t_{ij} se realice., en el caso del ejemplo, $T_1 = \{t_1, t_2, t_3, t_4\}$ se le asigna a t_1 y t_2 el símbolo l_1 y a t_3 y t_4 el símbolo l_2.

- El modelo de cada actuador en el sistema queda como sigue:

  - Cada transición se etiqueta según la función $\lambda(t) = s \in \Sigma_{entrada}$ o $\lambda(t) = \varepsilon$, en el ejemplo, $\Sigma_{entrada} = \{a, c\}$ por lo tanto queda:
    $\lambda(t_6) = a \in \Sigma_{entrada}$ es la transición que indica abriendo válvula.
    $\lambda(t_5) = c \in \Sigma_{entrada}$ es la transición que indica cerrando válvula.
  - Cada lugar se le asigna una etiqueta que represente el estado estable que toma el actuador cuando se dispara la transición asociada a la señal a_i, en el ejemplo, $P_{21} = \{l_1, l_2\}$ donde las etiquetas representan:
    l_1 estado de la válvula = abierta.
    l_2 estado de la válvula = cerrada.

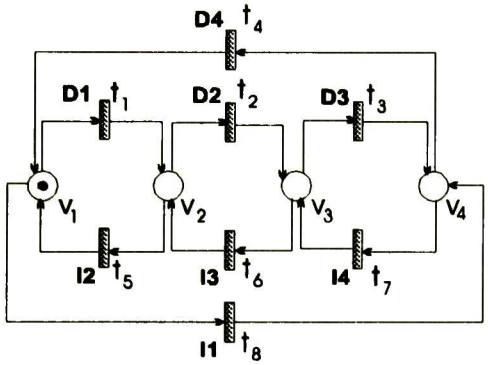
2. Una vez asignados símbolos y etiquetas se procede a:

- Realizar la Unión del conjunto de los lugares de los actuadores P_α y los lugares del modelo interno del sistema P, es decir, $P = P_{2\alpha} \cup P_1$, en el ejemplo anterior quedaría $P_{2\alpha} \cup P_1 = \{l_1, l_2, l_3, l_4, l_5\}$.

- Realizar la Unión del conjunto de transiciones de los actuadores $T_{2\alpha}$ y las transiciones del modelo interno del sistema T_1, es decir, $T = T_{2\alpha} \cup T_1$. En el ejemplo anterior quedaría $T_{21} \cup T_1 = \{t_5, t_6, t_1, t_2, t_3, t_4\}$.

- El $M_{0,2\alpha}$ de los Actuadores y el $M_{0,1}$ del modelo interno queda como $M_0 = [M_{0,1} M_{0,2\alpha}]^T$ En el ejemplo anterior quedaría:

$$M_0 = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 \\ l_3 & l_4 & l_5 & l_1 & l_2 \end{bmatrix}^T$$



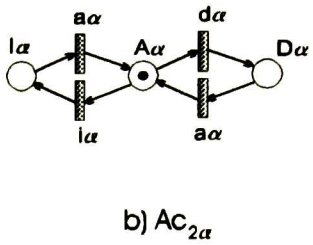


Figura 3.14: a) G_1 Comportamiento Interno b) Actuadores $Ac_{2\alpha}(\alpha = 1, 2, 3, 4)$

- El $F_{2\alpha}$ de los Actuadores y el F_1 del modelo interno queda como $F = [F_1 F_{2\alpha}]^T$ En el ejemplo anterior, se podría proponer el siguiente marcado Final $F = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 \end{bmatrix}^T$
- Los marcados del modelo interno se pueden representar como una función de marcado a partir del modelo de los sensores del sistema.

3. Por último se generan los autolazos que caracterizan el modelado propuesto, el cual consiste en unir los lugares de los actuadores con las transiciones del modelo del comportamiento interno, es decir:

- Si una transición t_{ij} del modelo interno del sistema tiene asignado un símbolo de los estados de los actuadores entonces la transición t_{ij} se une con el estado del actuador que hace que t_{ij} se realice, formando un autolazo.

  En el ejemplo del tanque, t_1 tiene asociada el símbolo l_1 por lo tanto el autolazo se genera entre esta transición y la etiqueta del actuador con el mismo símbolo, tal como se muestra en la Figura 3.13.

Ejemplo 3.4 *Se desea modelar un sistema formado por un tren, 4 tramos de vías y cuatro actuadores.*

En la Figura 3.14, se muestra el modelado interno de los cuatro tramos de vias donde estos pueden ser energizados individualmente hacia la derecha o hacia la izquierda, la marca en la vía asociada con el lugar V1(es la señal del sensor) indica la presencia del Tren, también se ilustra un modelo general del actuador, donde todos los eventos son controlables. Existen dos posibles comportamientos para una vía marcada, por ejemplo, cuando el tren está en V1, puede avanzar hacia la derecha de V1 a V2 ó hacia la izquierda de V1 a V4, por lo tanto,.a la transición t_1 se le asocia el símbolo D2 y a la transición t_5 se le asocia el símbolo I2, y así sucesivamente para todas las transiciones del modelo interno.

En la Figura 3.15, se presenta el modelo del sistema, que se obtuvo según la metodología propuesta, trazando arcos de cada una de las transiciones del modelo interno(asociadas con los símbolos) con los estados de los actuadores en los que coinciden la etiqueta con el símbolo de la transición.

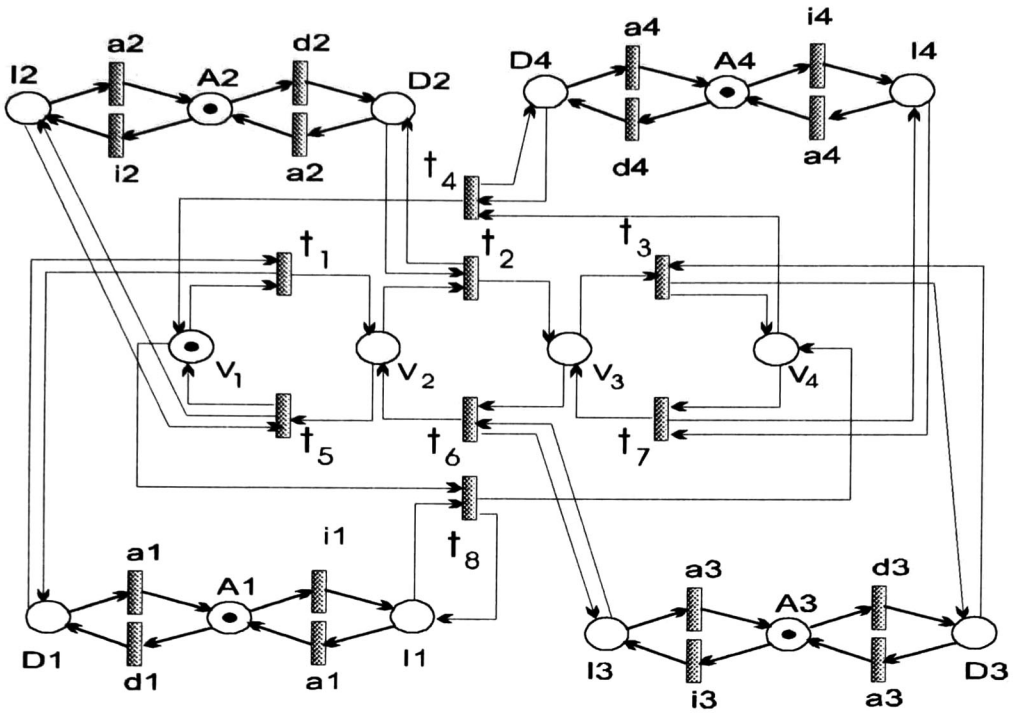


Figura 3.15: Modelado de Tren con 4 vías.

3.4.1 Cuadro Comparativo de las metodologías propuestas

A partir de las metodologías de modelado anteriormente descritas, se presenta un cuadro comparativo de las ventajas y desventajas que cada una presenta para modelar a los SED's.

	Poder de Abstracción	Potencia de Modelado	Complejidad Espacial	Facilidad de Interpretación	Potencial de Representación
Metodología de Modelado Wonham	*Cada subsistema k aporta n estados, los Q estados - del sistema global se - calculan: $Q = \prod_{i=1}^{k} n_i$	* Lenguajes Regulares	* No puede representar Automátas de Estados Infinitos.	* Se fusionan los estados y por lo tanto, su interpretación física se abstrae.	* Puede representar elecciones, pero no puede representar sincronizaciones, exclusión mutua, paralelismo.
Metodología de Modelado Giua	* El modelo que resulta a partir de los subsistemas k, queda ex-- presado por: $P = \bigcup_{i=1}^{k} p_i$ $T \ll \prod_{i=1}^{k} T_i$	* Lenguajes Regulares * Algunos Lenguajes Libre de Contexto * Algunos Lenguajes Sensitivos de Contexto	* Puede representar algunos Automátas de Estados Infinitos.	* El modelo del sistema crece, debido a que aumenta el no. de transiciones. Es difícil identificar cada elemento físico del sistema, sin -- embargo, describe con claridad el comporta-- miento del sistema.	* Puede representar elecciones, sincronizaciones, exclusión mutua, paralelismo.
Metodología de Modelado Propuesto	* La unión de los subsistemas k generan un modelo : $P = \bigcup_{i=1}^{k} p_i$ $T = \bigcup_{i=1}^{k} T_i$	* Lenguajes Regulares * Algunos Lenguajes Libre de Contexto * Algunos Lenguajes Sensitivos de Contexto	* Puede representar algunos Automátas de Estados Infinitos.	* Su interpretación física es sencilla, puesto que debido a las características del modelado cada k subsistema - queda intacto en el -- modelo gral. del sistema.	* Puede representar elecciones, sincronizaciones, exclusión mutua, paralelismo.

Figura 3.16: Sinopsis de Metodologías de Modelado para AF y RP

Capítulo 4

Control supervisorio

4.1 Introducción

Los SED requieren ser controlados para lograr que el sistema siga un lenguaje de especificación, de tal forma que este realice sólo las actividades necesarias para los objetivos y metas de producción en el caso de SMF, para hacer eficiente la transmisión de datos en el caso de los protocolos de comunicación, etc. La teoría de control realizada por Wonham y Ramadge, utiliza los Lenguajes Formales para modelar los SED no controlados y el comportamiento de especificación o deseado. El problema básico del control supervisorio es modificar el comportamiento en lazo abierto de un SED eliminando secuencias de eventos en el comportamiento del sistema, es decir, restringir el comportamiento del sistema de tal forma que estos estén contenidos en un comportamiento especificado o deseado, llamado Lenguaje de especificación. Esto se logra restringiendo la ejecución de los eventos del generador de eventos discretos sincronizándolos con otro sistema, que se le denomina Supervisor.

El sistema controlado se describe como un generador de lenguaje formal, mientras que el Controlador o Supervisor, es construido para ser un *reconocedor* de un lenguaje especificado que incorpora el comportamiento del sistema cuando el controlador se conecta en el lazo de retroalimentación del sistema.

En las siguientes secciones se presentan quienes han trabajado sobre la teoría de control supervisorio aplicado en SED's y algunos resultados sobre control supervisorio aplicado en SED utilizando AF y RP, finalmente se presenta definición de controlabilidad en RPI y la condición que debe cumplir para que la Red sea controlable.

4.2 Teoría de Control Supervisorio en SED's

[Wonham 87] ha trabajado en la elaboración de una teoría del Control Supervisorio utilizando AF, basado en el concepto de un supervisor que es es un agente capaz de deshabilitar los eventos controlables de un SED en respuesta a las secuencias de los eventos que son generados, este Supervisor debe restringir el comportamiento del SED a un Lenguaje o secuencia de operaciones que el sistema debe seguir. Si no se puede obtener el supervisor deseado, puede optarse por obtener un modelo más pequeño, que contemple las más secuencias posibles del supervisor, es decir, obtener un sublenguaje supremo controlable [Wonham 87a]. También [Lafortune 90] ha trabajado en la obtención de un operador superlenguaje mín-

imo controlable cerrado, presenta algunas propiedades que éste presenta y lo utiliza para resolver el problema de encontrar un Supervisor no bloqueado. [Lin 88] han trabajado sobre el control Descentralizado y su coordinación en observaciones parciales sobre SED, es decir, el SED modelado por un AF lo dividen en subsistemas y a cada parte se le diseña un control supervisorio local y luego hacen un control supervisorio global que coordine las acciones entre todos los supervisorios locales. [Tadmor 89] tienen como tema principal de su estudio la integración automática de la estructura local y las restricciones operacionales de cada sistema por separado, para después combinarlos y formar sistemas globales. [Usio 90] muestra la equivalencia entre supervisores de estados finitos determinísticos y no determinísticos. Además de dar la condición necesaria y suficiente para la existencia de un supervisor que puede ser representado por un autómata de estados finitos. [Lin 90] extiende un trabajo anterior sobre control descentralizado con observaciones parciales, aquí se presenta las condiciones para la existencia de un supervisor coordinador, ilustran los resultados en un modelo de manufactura flexible. [Charbonnier 94] presentan un sistema de manufactura del Atelier Interétablissemente de Productique[1] donde aplican los conceptos de control en autómatas para realizar un control supervisorio sobre el sistema a partir de la modularización de tareas, es decir, la división de las tareas globales del supervisor en dos o más subtareas. [Balemi 93] presentan una aplicación de la teoría del control supervisorio sobre una celda de manufactura de un multiprocesador rápido térmico.

En Redes de Petri, varios investigadores han trabajado en la teoría de control. Algunos de ellos, por ejemplo, [Holloway 90] presentan un control lógico en lazo cerrado para la clase de RP Grafos marcados cíclicos controlados para resolver el problema de llegar a estados prohibidos, proponiendo un método computacional, los resultados los aplican sobre un vehículo autoguiado en coordinación con un sistema de manufactura. [Sreenivas 92] consideran una clase de control supervisorio que requiere supervisores de estados infinitos e introducen las RP con arco inhibidor para modelar los supervisores. [Holloway 92] presentan resultados en Grafos marcados cíclicos para resolver el problema sobre vivacidad en de un sistema en lazo cerrado dando condiciones suficientes para una política permisible de control retroalimentado. [Giua 92] trata el problema de encontrar las condiciones necesarias y suficientes para la existencia de un supervisor en RP, siempre y cuando el lenguaje del sistema y el lenguaje de especificación sean ambos Lenguajes en RP.

4.3 Control Supervisorio en Autómatas Finitos

En primera instancia se presenta la estructura formal de un DES como un generador en AF, este ya fue definido en el Capítulo 2:

$$G = (Q, \Sigma, \delta, q_0, F)$$

donde:
$\Sigma = \Sigma_u \cup \Sigma_c$
Cada subconjunto de eventos $\Gamma = \{\gamma \in \mathcal{P}(\Sigma) \mid \gamma \supseteq \Sigma_u\}$ es un patrón de control.

[1] AIP, Saupliné Davoie, France

4.3.1 Definición de Control Supervisorio

Un Control Supervisorio para G es un mapeo $S : \mathcal{L}(G) \longrightarrow \Gamma$. El par (G, S) escribirá como S/G, lo cual significa "G bajo la supervisión de S"

El Lenguaje Cerrado de S/G es definido como el Lenguaje $\mathcal{L}(S/G) \subseteq \mathcal{L}(G)$ descrito como sigue

i) $\varepsilon \in \mathcal{L}(S/G)$

ii) Si $s \in \mathcal{L}(S/G)$, $\sigma \in S(s)$, y $s\sigma \in \mathcal{L}(G)$ entonces $s\sigma \in \mathcal{L}(S/G)$.

iii) Ninguna otra cadena pertenece a $\mathcal{L}(S/G)$.

Siempre se tendrá que $\{\varepsilon\} \subseteq \mathcal{L}(S/G) \subseteq \mathcal{L}(G)$, claramente $\mathcal{L}(S/G)$ es no vacío y cerrado.
El lenguaje Marcado para S/G es

$$\mathcal{L}_m(S/G) = \mathcal{L}(S/G) \cap \mathcal{L}_m(G)$$

Por lo tanto, siempre se tiene $\emptyset \subseteq \mathcal{L}_m(S/G) \subseteq \mathcal{L}_m(G)$.
Se dice que S es no bloqueado (Para G) si

$$\overline{\mathcal{L}_m}(S/G) = \mathcal{L}(S/G)$$

El objetivo es caracterizar estos lenguajes para que califiquen como lenguajes marcados bajo algún supervisor S. Un lenguaje $K \subseteq \Sigma^*$ se dice controlable (con respecto a $\mathcal{L}(G)$ y Σ_u) si:

$$\overline{K}\Sigma_u \cap \mathcal{L}(G) \subseteq \overline{K}$$

Significa que es posible restringir $\mathcal{L}(G)$ dentro de K.
Un lenguaje $K \subseteq \mathcal{L}_m(G)$ se dice ser $\mathcal{L}_m(G) - cerrado$ si:

$$K = \overline{K} \cap \mathcal{L}_m(G)$$

Esto significa que cualquier cadena marcada de G que es el prefijo de alguna cadena de K es también una cadena de K.

4.3.2 Existencia de un Control Supervisorio en AF

Enseguida se presentan unas condiciones necesarias y suficientes para la existencia de un Control Supervisorio no Bloqueado (CSN) y un Control Supervisorio Marcado-no Bloqueado(CSMN).

Teorema 4.1 *Sea $K \subseteq \mathcal{L}_m(G)$, $K \neq \emptyset$. Existe un Control Supervisorio no bloqueado S para G tal que $\mathcal{L}_m(S/G) = K$ si y solo si*

a) K es controlable con respecto a G, y

b) K es $\mathcal{L}_m(G) - cerrado$.

Teorema 4.2 *Sea $K \subseteq \mathcal{L}_m(G)$, $K \neq \emptyset$. Existe un CSMN S para (K, G) tal que $\mathcal{L}_m(S/G)=K$*
si y sólo si K es controlable con respecto a G.

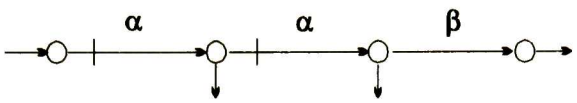


Figura 4.1: G

4.3.3 Sublenguaje Supremo Controlable en AF

Ahora, introduciremos el concepto de Sublenguaje supremo Controlable, con la finalidad de mostrar la obtención de un Supervisor que contemple la mayoría de las secuencias especificadas. Sea $D \subseteq \sum^*$, D será el Lenguaje de Especificación o deseado para el control Supervisorio de G. El conjunto de todos los sublenguajes de D que son controlables con respecto de G:

$$\mathcal{C}(D) = \{K \subseteq D | K \quad \text{es controlable con respecto de } G\}$$

Como un subconjunto de sublenguajes de D, $\mathcal{C}(D)$ es un Conjunto Parcialmente Ordenado (CPO) con respecto a la inclusión.

Proposición 4.1 *$\mathcal{C}(D)$ es no vacío y cerrado bajo uniones arbitrarias. En particular, $\mathcal{C}(D)$ contiene un (único) elemento supremo [el cual se denotará por el $\sup \mathcal{C}(D)$].*
$\sup \mathcal{C}(D) = \cup\{K | K \in \mathcal{C}(D)\}$

De ahora en adelante, sea $D, L \subseteq \Sigma^*$. Se dirá que D es $\mathcal{L} - marcado$ si $D \supseteq \overline{D} \cap L$.

Proposición 4.2 *Sea $D \subseteq \Sigma^*$ $\mathcal{L}_m(G) - marcado$. Entonces $\sup \mathcal{C}(D \cap L_m(G))$ es $\mathcal{L}_m(G) - cerrado$*

Teorema 4.3 *Sea $D \subseteq \Sigma^*$ $\mathcal{L}_m(G) - marcado$, y sea $K = \sup \mathcal{C}(D \cap L_m(G))$. Si $K \neq \emptyset$, existe un CSN S para G tal que $\mathcal{L}_m(S/G) = K$.*

Teorema 4.4 *Sea $D \subseteq \Sigma^*$ y $K = \sup \mathcal{C}(D \cap L_m(G))$. Si $K \neq \emptyset$ existe un CSMN S para G tal que $\mathcal{L}_m(S/G) = K$.*

Ejemplo 4.1 *Sea G un SED controlado mostrado en la Figura 4.1 donde:*

$$\Sigma = \{\alpha, \beta\}, \qquad \Sigma_c = \{\alpha\}, \qquad \Sigma_u = \{\beta\},$$
$$\mathcal{L}(G) = \{\varepsilon, \alpha, \alpha^2, \alpha^2\beta\},$$
$$\mathcal{L}_m(G) = \{\alpha, \alpha^2, \alpha^2\beta\}$$

Se pretende obtener un Supervisor que cumpla con el siguiente Lenguaje de especificación $D = \{\alpha, \beta, \alpha^2\}$. Entonces:

$$\overline{D} = \{\varepsilon, \alpha, \alpha^2, \beta\},$$
$$\overline{D} \cap \mathcal{L}_m(G) = \{\alpha, \alpha^2\},$$

Ahora al obtiener el supremo, queda:

$$K = \sup \mathcal{C}(D \cap L_m(G)) = \{\alpha\},$$
$$\overline{K} = \overline{\{\alpha\}} = \{\varepsilon, \alpha\}, \qquad K = \overline{\{\alpha\}} \cap \mathcal{L}_m(G) = \{\alpha\}$$

Desde éstos resultados se puede ver que D es $L_m(G)$ − *marcado*, y concluir que el $\sup \mathcal{C}(D \cap L_m(G))$ es $L_m(G)$ − *cerrado* como se presenta en el teorema 4.3. Para el Control Supervisorio se podría tomar $S(\varepsilon) = \{\alpha, \beta\}$, $S(\alpha) = \{\beta\}$, y$S(s) = \{\beta\}$ en otro caso. Entonces

$$\mathcal{L}(S/G) = \{\varepsilon, \alpha\}, \qquad \mathcal{L}_m(S/G) := \mathcal{L}(S/G) \cap \mathcal{L}_m(G) = \{\alpha\}$$

donde claramente se ve que S es no-bloqueado para G, como se esperaba.

Ejemplo 4.2 *En el modelo del llenado del tanque representado por AF en el capítulo tres, se tiene que el lenguaje generado por el sistema es* $\mathcal{L}(G) = \overline{(aacc)^*} + \overline{(aca(ac)^*c)^*} + \overline{a(ac)^*}$ *y el lenguaje marcado es* $\mathcal{L}_m(G) = (aacc)^* + (aca(ac)^*c)^* + a(ac)^*$*; donde se tiene que probar si existe un supervisor* S *que sea* $\mathcal{L}_m(G)$ − *cerrado y controlable.*

- *Si* $K = (aacc)^*$ *entonces:*

  1. $K = \overline{K} \cap \mathcal{L}_m(G)$
     $\overline{K} = \overline{(aacc)^*}$
     $K = (aacc)^*$ *es cerrado.*
  2. $\overline{K}\Sigma_u \cap \mathcal{L}(G) \subseteq \overline{K}$
     $\overline{K}\Sigma_u = \overline{(aacc)^*}$
     $\overline{(aacc)^*} = \overline{(aacc)^*}$ *es controlable.*

- *Si* $K = a(ac)^*$ *entonces:*

  1. $\overline{K} = \overline{a(ac)^*}$
     $K = \overline{a(ac)^*} \cap \overline{a(ac)^*}$
     $K = \overline{a(ac)^*}$ *es cerrado.*
  2. $\overline{K}\Sigma_u = \overline{a(ac)^*}$
     $\overline{a(ac)^*} \subseteq \overline{a(ac)^*}$ *es controlable*

En ambos casos, existe un supervisor porque el Lenguaje Marcado es cerrado y el sistema es controlable con respecto al Lenguaje de G.

4.4 Control Supervisorio en Redes de Petri

Enseguida se presenta la estructura formal de un DES como un generador en RP, este ya fue definido en el Capítulo 2, como:

$$GRP = (\mathcal{N}, \ell, M_o, F)$$

donde:

$\ell : T \rightarrow \Sigma \cup \{\varepsilon\}$ es una *función de etiquetado* que asigna a cada transición una etiqueta desde el alfabeto de eventos $\Sigma = \Sigma_u \cup \Sigma_c$ o asigna una cadena vacía como una etiqueta

Cada subconjunto de eventos $\Gamma = \{\gamma \in \mathcal{P}(\Sigma) \,|\, \gamma \supseteq \Sigma_u\}$ es un patrón de control.

4.4.1 Control Supervisorio

Para poder representar un sistema en RP, el sistema y el supervisor deben ser generadores de Redes de Petri, GRP y S, respectivamente, y entonces es posible construir un modelo en RP del sistema en lazo cerrado bajo el control de S/GRP utilizando el operador intersección sobre los lenguajes [Peterson 81]. Si GRP y S son generadores determinísticos (como siempre se asume) S/GRP también resulta determinístico.

4.4.2 Existencia del Supervisor en RP

A continuación se presentan las condiciones necesarias y suficientes para la existencia de un Supervisor en RP [Giua 92], también se muestra en un ejemplo posterior que la existencia del supervisor depende no solamente de que pueda ser representado como un generador en RP, sino que también cumpla con ser un lenguaje controlable y cerrado.

Teorema 4.5 *Sea GRP una RP no bloqueada y $\mathcal{L} \subseteq \mathcal{L}(GRP)$ un lenguaje no vacío. Existe un Supervisor en RP S tal que $\mathcal{L}(S/GRP) = \mathcal{L}$ si y solo si $\mathcal{L}$ es un Lenguaje en RP tipo-P determinístico controlable, i.e., $\mathcal{L} \in \mathcal{P}_d{}^2 \cap \mathcal{C}(GRP)$.*

Teorema 4.6 *Sea GRP una RP no bloqueada y $\mathcal{L} \subseteq \mathcal{L}_m(GRP)$ un lenguaje no vacío. Existe un Supervisor en RP S tal que $\mathcal{L}_m(S/GRP) = \mathcal{L}$ si y solo si $\mathcal{L} \in \mathcal{L}_{DP} \cap \mathcal{C}(GRP)$ y $\mathcal{L}$ es $\mathcal{L}_m(GRP)$ – cerrado.*

Ejemplo 4.3 *El lenguaje generado por la metodología de modelado propuesto en el capítulo tres del sistema de llenado de un tanque es* $\mathcal{L}(GRP) = \overline{(t_1t_2t_5t_3t_4t_6)^* + (t_5t_6)^* + (t_1t_5t_4t_6)^* + t_1(t_5t_6)^* + t_1t_2(t_5t_6)^*}$ *y* $\mathcal{L}_m(GRP) = (t_1t_2t_5t_3t_4t_6)^* + (t_5t_6)^* + (t_1t_5t_4t_6)^* + t_1(t_5t_6)^* + t_1t_2(t_5t_6)^*$, *si*

$F = \{[\ 1\ \ 0\ \ 0\ \ 1\ \ 0\ ], [\ 0\ \ 1\ \ 0\ \ 1\ \ 0\ ], [\ 0\ \ 0\ \ 1\ \ 1\ \ 0\ ]\}$, *donde* $\ell_u = \{t_1, t_2, t_3, t_4\}$ *y* $\ell_c = \{t_5, t_6\}$.

- *Si* $K = (t_1t_2t_5t_3t_4t_6)^*$ *entonces:*

  1. $\overline{K} = \overline{(t_1t_2t_5t_3t_4t_6)^*}$
     $K = \overline{(t_1t_2t_5t_3t_4t_6)^*} \cap (t_1t_2t_5t_3t_4t_6)^* = (t_1t_2t_5t_3t_4t_6)^*$ *es cerrado.*
  2. $\overline{K}\Sigma_u = \{\underline{t_1}, \underline{t_2}, \underline{t_3}.\underline{t_4}, t_1\underline{t_1}, t_1\underline{t_2}, \ldots, t_1t_2\underline{t_1}, t_1t_2\underline{t_2}, \ldots, t_1t_2t_5\underline{t_1}, \ldots$
     $t_1t_2t_5t_3\underline{t_1}, \ldots, t_1t_2t_5t_3t_4\underline{t_1}, \ldots, t_1t_2t_5t_3t_4t_6\underline{t_1}, \ldots\}$
     $\overline{K}\Sigma_u \cap \mathcal{L}(GRP) \subseteq \overline{K}$
     $\overline{K}\Sigma_u \cap \mathcal{L}(GRP) = \{\underline{t_1}, t_1\underline{t_2}, t_1t_2t_5\underline{t_3}, t_1t_2t_5t_3\underline{t_4}, t_1t_2t_5t_3t_4t_6\underline{t_1}, \ldots\} \subseteq \overline{K}$ *es controlable.*
     Como el lenguaje de la red, es el mismo que el del comportamiento del sistema, en éste caso el sistema es con respecto al Lenguaje cerrado y controlable.

- *Si* $K = t_1(t_2t_5t_3t_6)^*$ *entonces:*

  1. $\overline{K} = \overline{t_1(t_2t_5t_3t_6)^*}$
     $K = \overline{t_1(t_2t_5t_3t_6)^*} \cap t_1(t_2t_5t_3t_6)^* = t_1(t_2t_5t_3t_6)^*$ *es cerrado.*

2. $\overline{K\Sigma_u} = \{t_1, \underline{t_2}, \underline{t_3}.\underline{t_4}, t_1\underline{t_1}, t_1\underline{t_2}, \ldots, t_1t_2\underline{t_1}, t_1t_2\underline{t_2}, \ldots, t_1t_2t_5\underline{t_1}, \ldots$
$t_1t_2t_5t_3\underline{t_1}, \ldots, t_1t_2t_5t_3t_6\underline{t_1}, \ldots, t_1t_2t_5t_3t_6t_2\underline{t_1},$
$\ldots, t_1t_2t_5t_3t_6t_2t_5\underline{t_1}, \ldots\}$

$\overline{K} = \{\varepsilon, t_1, t_1t_2, t_1t_2t_5, t_1t_2t_5t_3, t_1t_2t_5t_3t_6, t_1t_2t_5t_3t_6t_1, \ldots\}$

$\overline{K\Sigma_u} \cap \mathcal{L}(GRP) = \{t_1, t_1t_2, t_1t_2t_5t_3, t_1t_2t_5t_3t_4, t_1t_2t_5t_3t_6t_2,$
$t_1t_2t_5t_3t_6t_2t_5t_3, t_1t_2t_5t_3t_6t_2t_5t_3t_4, \ldots\}$ *no es controlable.*

No existe un Supervisor para este Lenguaje, puesto que en la condición de controlabilidad, los prefijos del lenguaje del supervisor no son comparables con la intersección de los prefijos y las transiciones incontrolables con el lenguaje que genera el sistema.

Para el modelo del llenado del Tanque a partir de la metodología de Giua, el lenguaje generado por el sistema para el marcado inicial dado es igual al descrito en el ejemplo 4.2 para AF.

4.5 Controlabilidad en RPI

En el concepto de controlabilidad introducido por Wonham se pide que el sistema sea capaz de generar un conjunto de palabras dada por una especificación. Ningún prefijo debe habilitar un evento incontrolable si no está contemplado en la especificación. Desafortunadamente, esta aproximación de controlabilidad resulta insuficiente si se pide que el sistema alcance un estado. Es decir, si uno quiere llevar un sistema de un estado inicial a un estado final, usando la definición de Wonham el sistema puede resultar incontrolable, aún cuando exista una secuencia que permita que el sistema alcance el estado final.

Por otro lado, el concepto de alcanzabilidad de las Redes de Petri también resulta insuficiente. El que exista una secuencia que lleve el sistema al estado final, no implica que esta secuencia no habilita eventos incontrolables.

Este tipo de problemas aparecen cuando se requiere llevar un SED a un estado final o a una especificación dada (como lo hace Wonham) donde una secuencia transitoria es necesaria para llevar primero al sistema a un estado que permita empezar a realiar la especificación. Tal es el caso cuando en un sistema de manufactura flexible se introducen nuevas secuencias de producción cuando aún están otras secuencias realizándose.

Para tratar este caso, se propone la siguiente definición de controlabilidad, la cual es una mezcla de controlabilidad y alcanzabilidad. En este caso se dice que un SED es controlable ssi existe una secuencia σ que a partir del marcado inicial lleve a un marcado final predeterminado. Si alguna de las transiciones de σ es incontrolable, entonces a partir del marcado alcanzado con el disparo de esa transición debe existir una secuencia (sufijo de σ) que lleve al marcado final. Formalmente se tiene la siguiente definición.

Definición 4.1 (Meda 98) *Sea GRPI (como se definió en Capítulo 2), F el conjunto de estados finales y $\sum_u$ los eventos incontrolables, el SED descrito por la GRPI se dice que es controlable si se cumple lo siguiente:*

1. *Ningún marcado final habilita transiciones incontrolables, y* $\forall M_f \in F$

$\exists \sigma = \sigma_a\sigma_b \mid M_0[\sigma_a > M_x[\sigma_b > M_f \qquad M_x \in R(\mathcal{N}, M_0)$

2. $\forall M_x$ *de la condición anterior*

a) $\nexists s \in \Sigma_u \mid M_x[s >$ *(no se habilita un evento incontrolable) ó*

b) $\forall s \in \Sigma_u \mid M_x[s > M_y \rightarrow \exists \sigma' = \sigma'_a\sigma'_b \mid M_y[\sigma'_a > M_z[\sigma'_b > M_i$ *y* M_z *cumple con 2 (la incontrolable lleva un estado que todavía permite llegar al estado final).*

Donde:
M_0 es el marcado inicial
Σ_u es el conjunto de transiciones incontrolables
M_x, M_y, M_z, etc son marcados alcanzables marcados intermedios entre M_0 y M_f alcanzables con algunos prefijos de σ.

El significado de esta definición es que un SED es controlable si existe una secuencia que puede particionarse en $\sigma = \sigma_a\sigma_b$, tal que σ_a lleva a un estado intermedio M_x a partir del estado inicial M_0, y a partir de M_x con la secuencia σ_b se llega a $M_i \in F$.

La segunda condición dice que para todo marcado intermedio M_x no debe existir un evento incontrolable $s \in \Sigma_u$; o en el caso de que este evento incontrolable ocurra, entonces a partir de M_x con el evento incontrolable se llegará al marcado M_y para el cual deberá existir una secuencia $\sigma' = \sigma'_a\sigma'_b$ tal que M_y seguido de σ'_a llegue a un marcado M_z y a partir de éste con la secuencia σ'_b llegue a un marcado final M_i.

La definición previa permite saber si un estado es alcanzable por medio de secuencias controlables. El siguiente algoritmo permite saber si un sistema es controlable.

Este problema se puede programar como el siguiente algoritmo, notar que en el peor de los casos es NP-Completo.

4.5.1 Algoritmo para resolver la incontrolabilidad o controlabilidad de un Sistema

Enseguida se propone un algoritmo de búsqueda [Dym 91], [Russell 95], para resolver la controlabilidad o no controlabilidad en un Sistema de Eventos Discretos.

1. *Si algún marcado final habilita transiciones incontrolables, entonces el sistema es incontrolable.*

2. Sea:

C	Matriz de incidencia de la red.
M_0	Marcado inicial en la red.
M_x	Marcados intermedios
M_f	Conjunto de marcados finales.
$\vec{\sigma}$	Vector característico asociado a la secuencia de disparos.
$\vec{\sigma}_a, \vec{\sigma}'_a$ prefijos	Vectores característicos, de la secuencia que lleva al estado final y sus
t_u	Transiciones incontrolables.

3. Probar si $\exists\ \vec{\sigma}$

   sujeto a $M_f = M_0 + C \cdot \vec{\sigma}$

   Si no tiene solución salir del algoritmo, el sistema es incontrolable

4. Probar si $\exists\ \vec{\sigma}$

   sujeto a $M_f = M_0 + C \cdot \vec{\sigma}$

   donde no existe un $M_x = M_0 + C \cdot \vec{\sigma}_a$ tal que

   $$M_x \geqslant C^-(\cdot t_u)$$
   $$\vec{\sigma}_a \leq \vec{\sigma}$$

   Si existe salir del algoritmo y concluir que el sistema es controlable.

5. Si no existe crear una estructura de árbol donde M_0 es la raíz del árbol, y $M_{hoja} = \{M_0\}$:

   (a) Para cada $M_i \in M_{hoja}$ hacer lo siguiente:

   Eliminar M_i de M_{hoja}

   Encontrar todos los sucesores de M_i que no hayan sido visitados anteriormente, llamar a este conjunto M_S

   $\forall M_l \in M_S$

   Probar si $\exists\ \vec{\sigma}$

   sujeto a $M_f = M_l + C \cdot \vec{\sigma}$ y no se revisitan estados.

   Si no tiene solución y $M_i[t_c > M_l$, t_c transición controlable eliminar M_l de M_s.

   Si no tiene solución y $M_i[t_u > M_l$, t_u transición incontrolable eliminar del árbol M_i. Viajando en el árbol desde M_0 a M_i, detenerse en un marcado M_k antes de la primer transición incontrolable que aparezca, eliminar desde M_k hasta M_i del árbol. Todos los nodos fuente y sus ramas (a excepción del raíz) también deben ser eliminados del árbol. Los marcados que han sido eliminados también deben retirarse de M_{hoja} y M_s, si es que existen.

   (a) $\forall M_l \in M_S$

   si $\exists\ \vec{\sigma}$

   sujeto a $M_f = M_l + C \cdot \vec{\sigma}$

   donde no existe un $M_x = M_l + C \cdot \vec{\sigma}_a$ tal que

   $M_x \geqslant C^-(\cdot t_u)$

   $\vec{\sigma}_a \leq \vec{\sigma}$

   Si tiene solución para algún M_l, entonces salir del algoritmo y concluir que el sistema es controlable.

   Incluir M_s en M_{hoja}

   Si M_{hoja} es vacío, *salir del algoritmo y concluir que el sistema es incontrolable*, de lo contrario, ir al paso a).

Como se observa este algoritmo permite probar si una GRPI acotada es controlable o no. El algoritmo en sí resulta NP completo en el peor de los casos, pero gracias a las herurísticas en él introducidas, el tiempo de cómputo resulta en la mayoría de los casos pequeño.

Capítulo 5

Optimalidad Local

5.1 Introducción

En un SED existen muchas secuencias que llevan al estado final. No todas estas secuencias son iguales en cuanto su desempeño. Por ejemplo, existen secuencias que resultan muy rápidas en su ejecución (tiempo mínimo), otras que resultan baratas (menor costo de producción). El problema de optimización [Ramírez 93a] consta de tres fases, en la primera se determina el conjunto de secuencias a optimizar, en la segunda se determina cómo medir el desempeño de cada secuencia y por último se genera un algoritmo para encontrar el mejor elemento del conjunto de secuencias. Formalmente el problema se describe a continuación.

1. Conjuntos y secuencias: Determinar los siguientes conjuntos:

   $W = \{(x, y, z) | x \in R, y \in T, z \in \mathbb{R}^+\}$; donde R es el conjunto de los recursos, T es el conjunto de las tareas. El número real z representa el tiempo de inicio de la tarea y.

   Un schedule $s = w_1 w_2 \ldots w_k$ es una secuencia de elementos $w_i \in W$ y $w = \{s | s$ es un schedule factible$\}$.

2. Función Objetivo: Encontrar la función f que describa el criterio de optimización, $f : w \rightarrow \mathbb{R}$.

3. Optimización: Encontrar, al menos, un elemento $s_i^* \in w$ tal que:

   $f(s_i^*) = s_i \in w \min f(s_i)$ (caso de minimización)

   $f(s_i^*) = s_i \in w \max f(s_i)$ (caso de minimización)

La última fase de este problema es la que se conoce como el diseño del scheduler o algoritmo de optimización discreta. Existen diferentes algoritmos de optimización, todos ellos se aplican en casos específicos, incluso los que utilizan inteligencia artificial [Ramírez 93a]. Una clasificación de estos algoritmos es la siguiente:

- Métodos basados en investigación de operaciones. Dentro de éstos métodos algunos son: PERT-CPM, método Húngaro.
- Métodos constructivos. Algunos de ellos son: SPT (shortest processing time), EDD (earliest due date), razón de Smith, Regla de Johnson.

- Métodos basados en inteligencia artificial. Algunos métodos son: Reglas de despache, aproximación jerárquica, métodos informados de búsqueda, scheduling oportunista, sistemas expertos.

A continuación se mencionan algunas de las personas que han trabajado en la elaboración de algoritmos de optimización aplicados a Sistemas de Eventos Discretos. [Passino 89] ha trabajado sobre el control óptimo en SED construyendo un control óptimo utilizando el algoritmo A^*; [Laftit 92] trabajaron en la elaboración de un criterio de optimización en tiempo basado en los p-invariantes de un grafo de eventos[1] fuertemente conexo, y propone un algoritmo de Ajuste heurístico y otro algoritmo de optimización convexo, los cuales son aplicados en un sistema de Job-Shop y en otro de Kanban.

Puesto que el cálculo del tiempo mínimo de ciclo en una red Libre Elección viva y sana es un problema NP-Completo[Ramírez 93a], ahi mismo se propone una clasificación en RP de Libre elección, donde para cada clase se propone un algoritmo heurístico de optimización en tiempo, para que se puedan encontrar schedules buenos. En la misma, se define y estudia la propiedad de "*Optimalidad Local*" (OL) en RP; proponiendo las características que estos subsistemas deben reunir.

La OL permite construir algoritmos de optimización muy rápidos. En sí se utilizan los algoritmos ya existentes, pero particionando adecuadamente el SED de manera que se resuelvan muchos problemas pequeños en vez de uno muy grande. Por ejemplo, si un algoritmo de optimización tarda 2^X nanosegundos en resolver un problema, donde X es el número de datos, un problema de 80 datos será resuelto en más de 10 millones de años, mientras que ocho problemas de 10 datos cada uno será resuelto en menos de 5 minutos.

En este trabajo se tratará la optimalidad local, la cual tiene como principio obtener la optimización global de un sistema, cuando se optimiza cada subsistema por separado, es decir, la optimización local divide el sistema en pequeños subsistemas, luego para cada subsistema obtiene una trayectoria óptima y finalmente se optimiza todas las secuencias que quedaron de una forma global. En un problema donde la complejidad es mayor que la lineal, la frase de divide y vencerás es útil, porque la optimización puede ser resuelta en mucho menor tiempo por partes que toda completa. Es decir, si $\tau(X)$ es el tiempo que tarda en realizarse la optimización global en un sistema y $\tau(x_i)$ es el tiempo que tarda en resolverse la optimización del subsistema x_i; con $\bigcup_{i=0}^{m} x_i = X$ y $x_i \cap x_j = \emptyset \ \ \forall x_i \neq x_j$

Entonces

$$\tau(X) \geq \sum_i \tau(x_i)$$

Es decir

$$k^{|X|} \geq \sum_{i=0}^{m} k^{|x_i|}$$

Sin embargo, optimizar localmente no siempre genera la secuencia óptima global, pero existen ciertos casos en los cuales la optimización local puede utilizarse y de esta manera

[1] event graph

ahorrar tiempo de cálculo. En este capítulo la optimización en un SED se basa en encontrar una trayectoria de mínimo tiempo.

En el presente trabajo se presentan unas definiciones sobre los pares inevitables y un teorema que establece bajo qué condiciones una RP presenta la propiedad de "*Optimalidad Local*".

5.2 Optimalidad Local

5.2.1 Definiciones

En seguida se definen algunos conceptos necesarios para dar la condicion suficiente para el teorema de Optimalidad Local propuesto que se apoya en el concepto de pares inevitables.

Definición 5.1 *Si un sistema puede ser dividido en subsistemas y las prestaciones óptimas de los subsistemas garantizan las prestaciones óptimas de todo el sistema, entonces el sistema posee la propiedad de* ***Optimalidad Local*** *(OL) con respecto a esa división [Ramírez 93].*

Definición 5.2 *Una subred de* $\mathcal{N} = (P, T, \rho, \nu)$ *es una red* $\mathcal{N}' = (P', T', \rho', \nu')$ *tal que* $P' \subseteq P$ *y* $T' \subseteq T$. ρ' *y* ν' *son restricciones de* ρ *y* ν *sobre* $P' \times T'$ *[Silva 85].*

Debido a que en la literatura de Optimalidad Local a una subred se le llama corte, de aquí en adelante a la subred también se le denomina Corte C_j.

Definición 5.3 *Sea* $R(\mathcal{N}, M_0)$ *el grafo de alcanzabilidad de la Red de Petri* $(\mathcal{N}, M_0)$. $[M_K, M_L]$ ***forman un par "inevitable"*** *en* $R(\mathcal{N}, M_0)$ *si todas las ramas que salen de* M_K *(Marcado de Entrada, mec) llegan a* M_L *(Marcado de Salida, msc) o todas las ramas que llegan a* M_L *salen de* M_K.

Definición 5.4 *Sea* $\mathcal{N}$ *una RP y* $\mathcal{N}'$ *un corte de* $\mathcal{N}$ *, entonces* $\mathcal{N}'$ *genera un Subgrafo de Alcanzabilidad del grafo de alcanzabilidad de* $\mathcal{N}$ *el cual se constuye como se describe a continuación:*

El nodo E_i *del Grafo de Alcanzabilidad pertenece al Subgrafo de Alcanzabilidad* SA_j *si en* E_i *al menos un lugar del corte* C_j *está marcado; el arco* $a_{lk} \in SA_j$ *si* E_i *y* $E_j \in SA_j$ *y existe un arco de* E_i *a* E_j *en el grafo de alcanzabilidad.*

En la Figura 5.2 se muestra una red de Petri $RP = (\mathcal{N}, M_0)$, y se presenta un posible corte C_j, su Grafo de Alcanzabilidad $R(\mathcal{N}, M_0)$ se presenta en la Figura 5.3a), en el inciso b) de la misma figura se muestra el subgrafo de alcanzabilidad del corte C_j.

5.3 Teorema de Optimalidad Local en RP

Este teorema relaciona el concepto de pares inevitables anteriormente definidos con la propiedad de Optimalidad Local en RP.

Teorema 5.1 *Sea una Red de Petri* $RP = (\mathcal{N}, M_0)$ *y un conjunto de cortes* $C = \{C_1, C_2, \ldots C_k\}$ *de la RP donde* $C_i \cap C_j = \emptyset \ \forall C_i, C_j \in C$. *Si el grafo de alcanzabilidad de cada* C_i *forma un par inevitable en* $R(\mathcal{N}, M_0)$ *entonces la RP tiene la propiedad de Optimalidad Local con respecto al conjunto de Cortes* C.

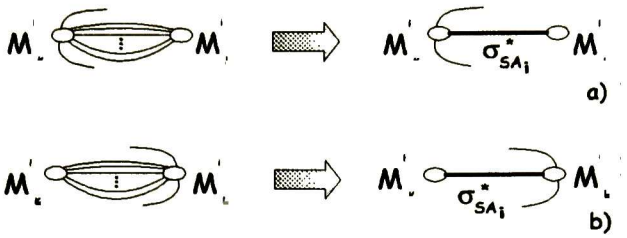


Figura 5.1: Casos de Trayectorias Optimas.

Prueba 5.1 *Cada corte C_i genera un subgrafo de alcanzabilidad $SA_i = R(\mathcal{N}', M_0')$ que forma un par inevitable $[M_K^i, M_L^i]$. Donde cada corte cumple con la definición de par inevitable puesto que cada trayectoria que forma parte de algún corte C_i no tiene intersección con ninguna trayectoria de algún otro corte C_j exceptuando el lugar de entrada y el lugar de salida en cada corte que pertenezca a C, como enseguida se verá.*

Para cada par inevitable se puede encontrar la rama de M_K^i a M_L^i óptima $\sigma^*_{GR_i}$ (que sea de mínimo tiempo).

Entonces se puede construir un $GR_i \subset R(\mathcal{N}, M_0)$ donde en GR_i se ha sustituido el par inevitable $[M_K^i, M_L^i]$ por un rama de tiempo mínimo $\sigma^*_{GR_i}$.

Claramente se ve que todas las secuencias en GR_i también pertenecen a $R(\mathcal{N}, M_0)$.

Además se puede ver que una secuencia óptima σ^*_R en GR_i es una secuencia óptima en $R(\mathcal{N}, M_0)$, por el siguiente razonamiento.

Si $\exists$ σ^* en $R(\mathcal{N}, M_0)$ óptima y no está en GR_i se puede tener que:

1. *Si $\tau(\sigma^*) = \tau(\sigma^*_R)$, entonces como σ^*_R también pertenece a $R(\mathcal{N}, M_0)$, σ^*_R es una secuencia óptima en $R(\mathcal{N}, M_0)$ y optimizando los cortes por separado se tiene que σ^*_R es un óptimo global, es decir, la RP tiene la propiedad de OL con respecto a esos cortes.*

2. *$\tau(\sigma^*) > \tau(\sigma^*_R)$, este caso no se puede dar porque σ^*_R también está en $R(\mathcal{N}, M_0)$ y habría sido encontrada como la secuencia óptima cumpliéndose con la OL.*

3. *$\tau(\sigma^*) < \tau(\sigma^*_R)$.*

   a) *Si σ^* y σ^*_R no tienen elementos comunes, entonces σ^* también pertenecería a GR_i y σ^*_R no sería el óptimo. Se ve entonces que la optimización local permite obtener el óptimo global.*

   b) *Ahora si éstas tienen elementos comunes, en general se tiene que $\sigma^* = \sec a | \sec b | \sec c$ en general y $\sigma^*_R = \sec a | \sigma^*_{GR_i} | \sec c$ ($\sec a$ y $\sec c$ pueden ser secuencias vacías y $\sec b$ es una secuencia del mismo corte por ser par inevitable).*

$\rightarrow \tau(\sec b) < \tau(\sigma^*_{GR_i})$ *dice que $\sec b$ es mejor que $\sigma^*_{GR_i}$, lo cual no puede ser porque $\sigma^*_{GR_i}$ ya es óptima.*

Por tanto σ^*_R sería una secuencia óptima, permitiendo que optimizaciones locales lleven a un óptimo global, es decir, se tiene la propiedad de OL.

Como para cualquier caso, los pares inevitables permiten la OL, la demostración se sigue.

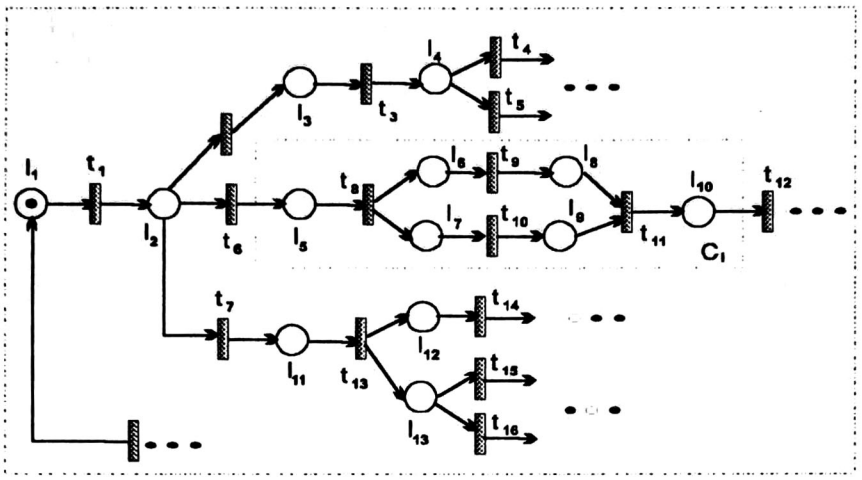


Figura 5.2: RP

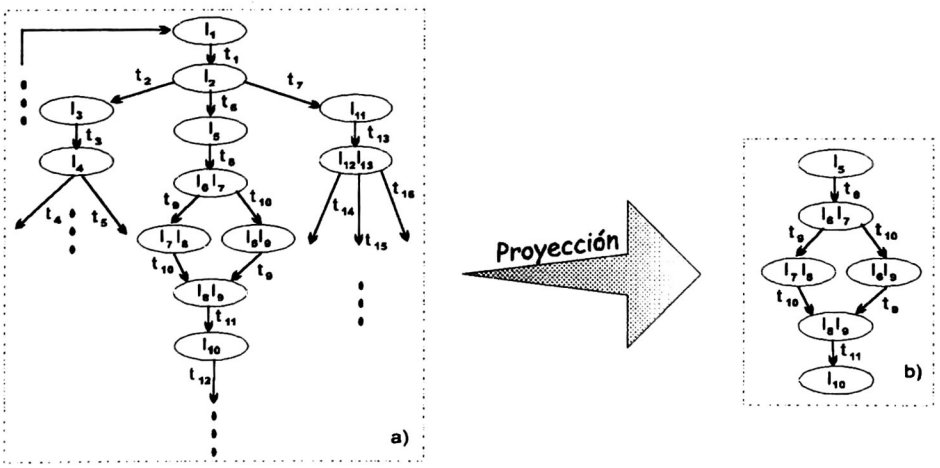


Figura 5.3: Grafo de Alcanzabilidad de la RP y una Proyección.

5.4 Relaciones entre los recientes Resultados y los existentes sobre Optimalidad Local

En la [Ramírez 93a] se introduce el concepto de Optimalidad Local (OL) en RP y se dan ciertas condiciones de suficiencia caracterizadas por las propiedades estructurales de las Redes.

En especial propone dos tipos de subsistemas: los subsistemas Lugar-Lugar y los subsistemas Transición-Transición y las RP que presentan la optimalidad local en base a estos subsistemas. Las RP que poseen la OL estudiadas en ese trabajo son un caso perticular de las estudiadas aquí. En seguida se presentan algunas definiciones de subsistemas y después se muestra que las RP presentadas en [Ramírez 93a] son un caso particular del teorema presentado aquí.

Definición 5.5 *Subsistema Lugar-Lugar (SLL). Sea* $RP = (\mathcal{N}, M_0)$ *un sistema. Un corte* $\mathcal{N}' = (P', T', C'^{-}, C'^{+})$ *de* $\mathcal{N}$ *(o sea* $P' \subseteq P$, $T' \subseteq T$, $C'^{-} = C|_{P' \times T'}$, $C'^{+} = C|_{P' \times T'}$*) con un* M'_0 *que es la proyección del marcado inicial de* $\mathcal{N}$ *sobre* $\mathcal{N}'$ *es un SLL si y sólo si:*

- ${}^\bullet t$ *y* $t^\bullet \in P'; \forall t \in T'$
- $\exists | mec \in P'$ *tal que* $mec \subseteq T \setminus T'$ *y* $mec \neq \emptyset$.
- $\exists | msc \in P'$ *tal que* $msc \subseteq T \setminus T'$ *y* $msc \neq \emptyset$.
- *Si* $C_i \in P' \cup T'$ *entonces debe existir un camino en* $C'^{-} \cup C'^{+}$ *dirigido de mec a* C_i *y otro camino de* C_i *a msc.*
- $\forall t \in T' \ \ \exists p \in {}^\bullet t, Y \geq 0$ *tal que* $Y^T C \geq 0, Y(p) = Y(mec) = Y(msc) > 0$.

Donde el subsistema Lugar Lugar es un corte C_i que cumple con las condiciones para un $C_i \in C$ dadas por el Teorema 5.1.

En la Figura 5.4 a)se muestra un corte SLL que forma un par inevitable. En general los subsistemas son denotados $\mathrm{Smec_i msc_j}$, donde $\mathrm{mec_i}$ es el elemento (lugar) de entrada al subsistema y $\mathrm{msc_j}$ es el elemento (lugar) de salida del subsistema. Cuando $\mathrm{mec_i}$ y/o $\mathrm{msc_j}$ son transiciones entonces se le denominará subsistemas $\mathrm{ST_{mec_i} T_{msc_j}}$, entonces aplicando una técnica de expansión se puede cambiar a un subsistema SLL. En la figura 5.4 b) se aplica la transformación de un subsistema $\mathrm{ST_{mec_i} msc_j}$ a un subsistema $\mathrm{Smec_i msc_j}$.

En [Ramírez 93a] se presentan las siguientes propiedades en máquinas de estados y grafos marcados, que si cumplen con ser un corte SLL, poseen la propiedad de optimalidad local.

Propiedad 5.1 *Si se obtiene una máquina de estados sana* $\Sigma = \langle \mathcal{N}, M_0 \rangle$ *cuando cada SLL es reducido a un lugar entonces* Σ *tiene la propiedad de Optimalidad Local.*

Propiedad 5.2 *Si una red marcada, viva y sana* $\Sigma = \langle \mathcal{N}, M \rangle$ *tiene solamente subsistemas que tienen un único t-semiflujo con todos sus elementos iguales a uno y se cumple que, cuando reducidos los SLL de la red resulta en un grafo marcado, entonces la red marcada* $\Sigma = \langle \mathcal{N}, M \rangle$ *posee la propiedad de Optimalidad Local.*

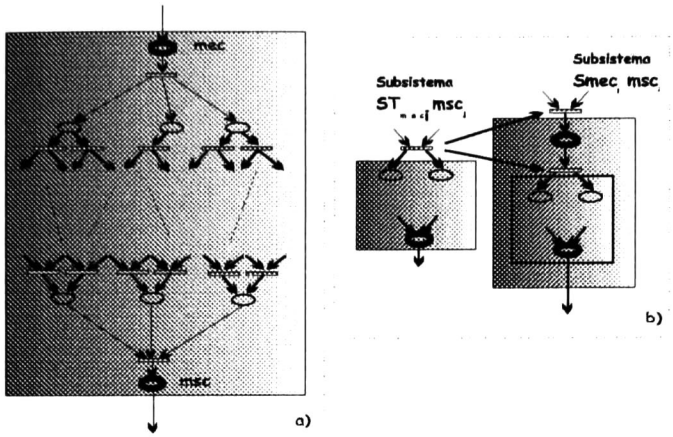


Figura 5.4: Subsistema SLL

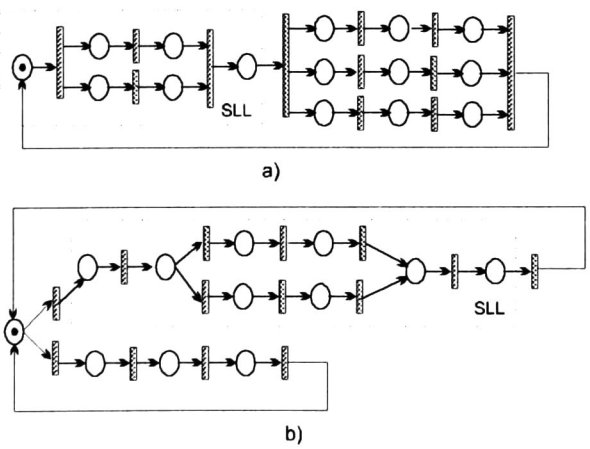


Figura 5.5: a) Grafo Marcado b) Máquina de Estado

Estas propiedades son dos casos particulares de los pares inevitables, ya que como se observa, un corte SLL todas las ramas que salen del estado en que se marca el lugar de entrada del corte llegan al lugar de salida del corte, ver Figura 5.5. Para trabajo futuro queda generalizar la definición de pares inevitables y hacer las caracterizaciones correspondientes para los grafos de alcanzabilidad y las Redes de Petri para la nueva definición.

Capítulo 6

Herramienta de aplicación para Control Supervisorio.

Esta capítulo describe de manera resumida la organización y funcionamiento de un simulador de control supervisorio para sistemas de Manufactura Flexible.

6.1 Generalidades

A partir del simulador gráfico para SMF que fué creado por [López 97], se agrega una capa de software donde se implementa un Control Supervisorio diseñado previamente. Este simulador implementa la(s) secuencia(s) de control supervisorio que son las especificaciones que el SMF debe de cumplir, dicha secuencia se obtiene a partir de las propiedades que poseen las RdP, el supervisor que representa el Lenguaje de especificación o deseado, debe cumplir que sea controlable, cerrado y además que pueda ser representado por un generador de *RdP*.

En la Figura 6.1 se muestra que el lenguaje de Kappa-PC está formado por la POO y los Frames y Reglas. Las características que presenta la POO, tales como: la abstracción de Datos (clases), herencia, Polimorfismo y encapsulamiento, hacen de Kappa una herramienta poderosa en el diseño para Sistemas de Eventos Discretos y particularmente en los SMF. Los Frames y Reglas con sus muy particulares propiedades y elementos como: objetos, métodos, mensajes y motor de inferencias, integrados a Kappa, permiten la simulación, ya sea paso a paso ó automáticamente, de la evolución del SMF y del Control Supervisorio que en este

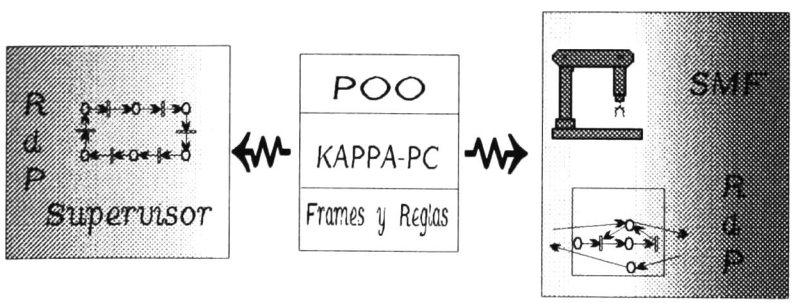


Figura 6.1: Kappa-PC

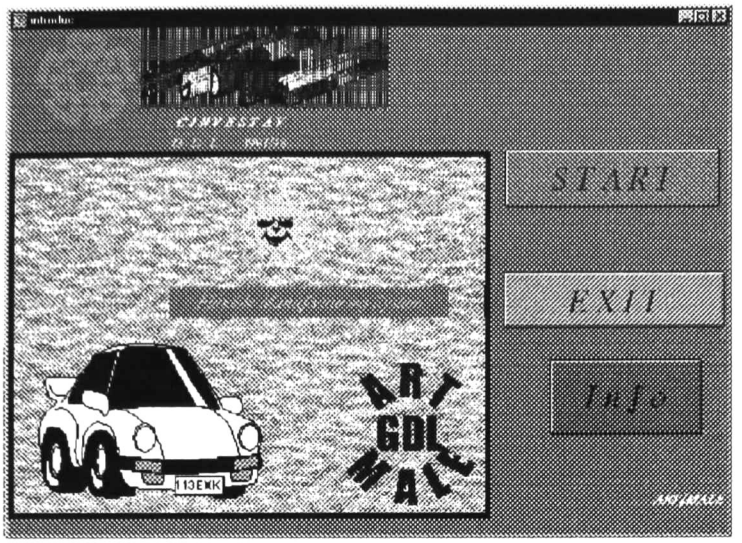


Figura 6.2: Pantalla de Inicio de Kappa-PC

trabajo se presenta, además permite prevenir si la secuencia de operaciones no tendrá algún tipo de problema, por ejemplo: que existan máquinas ociosas en el sistema, secuencias de operaciones que se bloqueen, etc.

6.1.1 Instalación e Inicio de GDLS

Para correr esta aplicación:

1. Instalar Kappa-PC y en la ventana principal del programa, seleccionar File → Open, abrir el archivo C:\KAPPA\SMF_US\GDLS.

2. Enseguida aparecerá la pantalla principal que ofrece las siguientes opciones, tal como se muestra en la Figura 6.2:

   Info: Contiene información acerca de GDLS.

   Start: Iniciar GDLS.

   Exit: Salir de la aplicación.

   La selección se realiza pulsando el botón izquierdo del mouse en cualquiera de las opciones mostradas.

6.1.2 Construcción del Control Supervisorio sobre el SMF.

Al seleccionar el ícono de **Start**, inmediatamente aparece la pantalla de Diseño de Sistema de Manufactura Flexible (**SMF**) y la de Diseño de Control Supervisorio (**SCD**) presentada en la Figura 6.3, en SMF el Sistema de Manufactura puede ser construido siguiendo los pasos presentados en la [López 97]. Solo se mencionará algunos íconos de FMS, que fueron agregados:

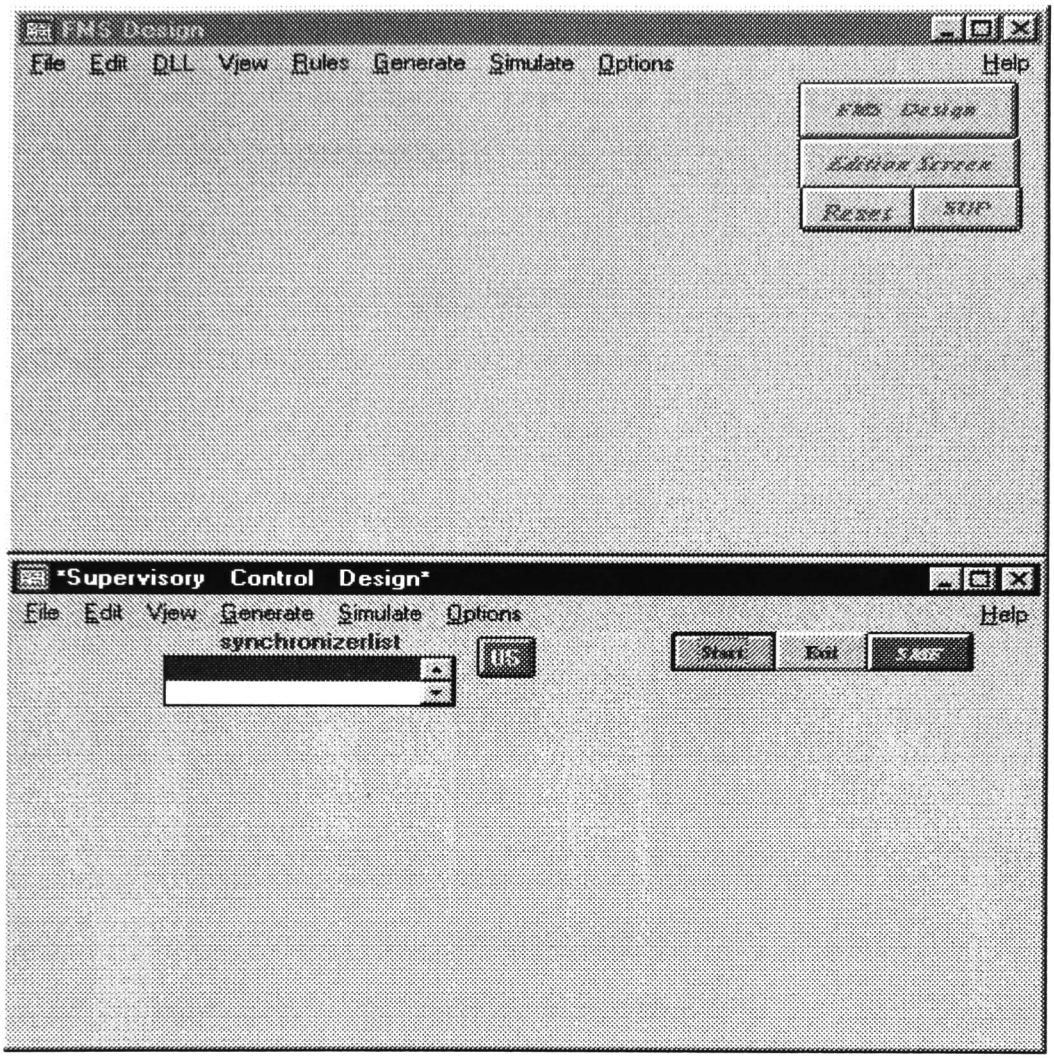


Figura 6.3: Pantalla de SMF y SC

- **SUP**: Cambia a la pantalla de SCD.
- **Reset**: Limpia las pantallas de SMF y SCD.

Enseguida se presenta una breve descripción y el procedimiento necesario en el software de la capa de Control Supervisorio.

Una vez terminado el diseño del sistema en FMS, pulsar el botón de SUP y aparecerá la pantalla mostrada en la Figura 6.4, las funciones mostradas en el menú son las generales para Kappa-PC, el sistema puede ser diseñado en ambas pantallas en este modo, sin embargo, para correrlo es necesario cambiar a los respectivos menús de SMF y Supervisory Control(**SC**).

Para la creación de objetos en SC oprimir botón derecho del ratón y se presentarán tres opciones: Lugares, Sincronizaciones Horizontales y Sincronizaciones Verticales.

Se tiene los siguientes íconos:

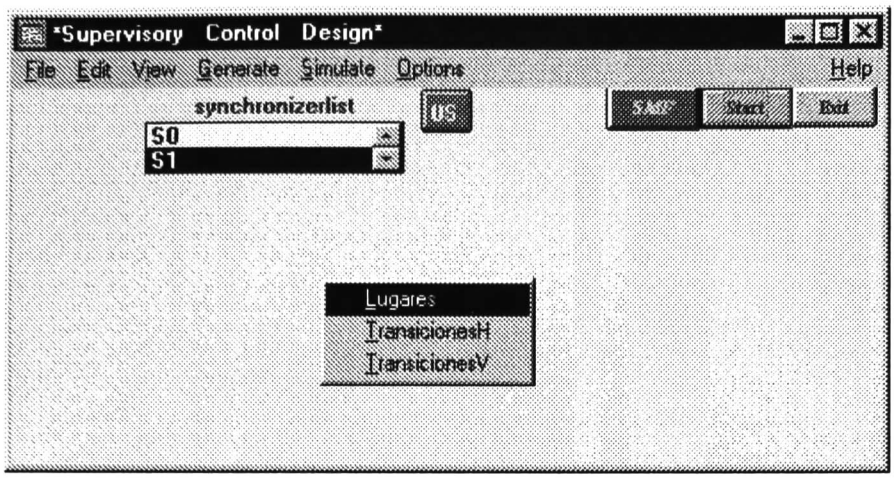


Figura 6.4: Construcción del Control Supervisorio

- **SMF**: Conmuta a la pantalla de SMF.
- **Syncronizerlist**: Muestra una lista para seleccionar Sincronizaciones del SMF.
- **US**: Debido a que la lista de selección de Sincronizaciones del SMF no se actualiza automáticamente.este botón las actualiza, tal como se muestra en la figura 6.4.

Para la construcción de la RdP en la pantalla de control supervisorio se realizan los siguientes pasos:

- Generar el lugar como se muestra en la Figura 6.5.
- Para la generación de la(s) sincronizacion(es) es necesario:

1. Pulsar el botón **US**.
2. Seleccionar con el mouse sobre el cuadro de diálogo de Syncronizerlist la sincronización de SMF que se relaciona con la transición de control Supervisorio en RdP
3. Pulsar el botón derecho del ratón sobre la pantalla para seleccionar la sincronización con la posición deseada y entonces en la pantalla aparecerá un mensaje indicando que el sincronizador seleccionado quedó asociado con la sincronización seleccionada en SCD, esto se muestra en la Figura 6.5.
4. Y se siguen los pasos del 2 al 3 para todas las sincronizaciones que deseen ser generadas.

Para la unión (arco) entre el lugar y la transición se hace lo siguiente:

1. Pulsar el botón derecho del ratón sobre la transición y aparecerá un menu con opciones, de la cual se elige input u output, en la Figura 6.6 se muestra la opción de "input"
2. Pulsar el botón derecho del ratón sobre el lugar al cual desee ser unido y elegir la opción connect, tal como se muestra en la Figura 6.7.

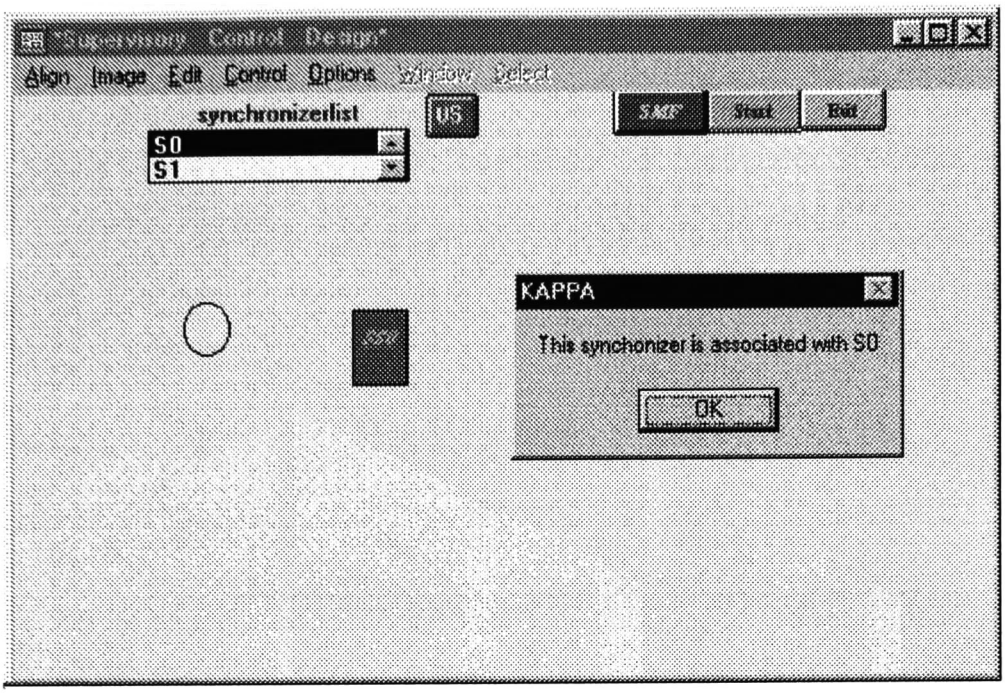


Figura 6.5: Relación de la sincronización de control con la sincronización en el SMF.

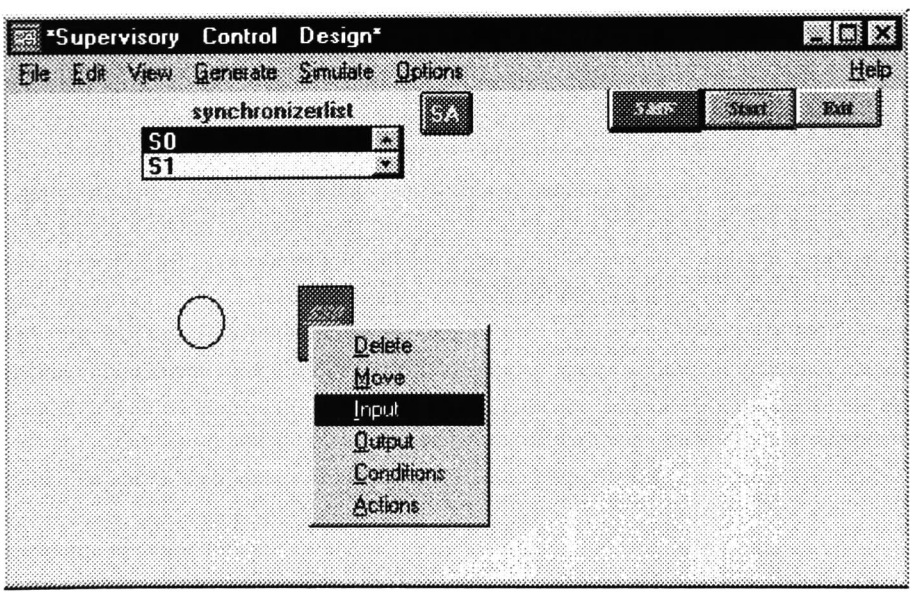


Figura 6.6: Procedimiento de Unión para un arco de entrada en la sincronización.

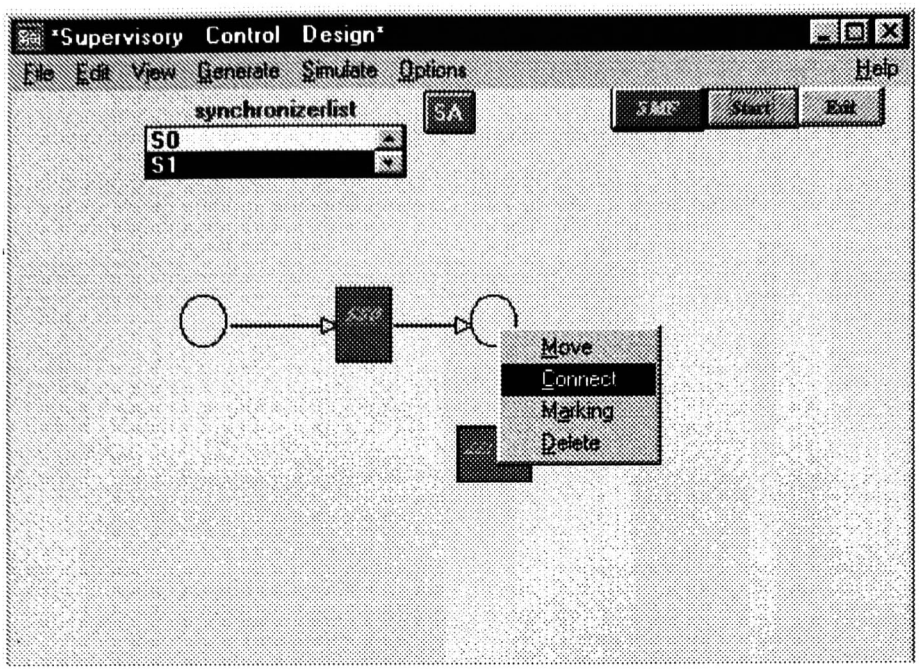


Figura 6.7: Procedimiento de Unión para un arco de entrada en el lugar.

3. Y seguir pasos 1 y 2 para todos los arcos que se necesiten generar.

En la Figura 6.8, se muestra un ejemplo de un SMF que consta de un almacén de entrada con una capacidad para almacenar 3 piezas, un robot con capacidad para trasladar 1 pieza, un taladro con capacidad para 1 pieza y un almacén de salida con una capacidad de almacenamiento para 3 piezas. La secuencia de control que debe seguir es la de almacenar como máximo 3 piezas en el almacén de entrada, que el robot tome la pieza y la coloque en el taladro, el robot no debe tomar otra pieza del almacén de entrada sino hasta que el taladro pase la pieza procesada, el robot la tome y la coloque en el almacén de salida y hasta entonces pueda tomar la siguiente pieza del almacén de entrada. Dicha secuencia se muestra en la pantalla inferior mostrada en la Figura 6.8. Pueden presentarse otro tipo de situaciones no deseadas sobre el SMF sino se implementa un lenguaje de especificación o deseado, por ejemplo, puede presentarse el caso en el que debido a que la capacidad del almacén de entrada es de tres piezas el robot tome la primer pieza y la coloque en el taladro, mientras que esta pieza es procesada en el taladro, puede que el robot tome las dos piezas restantes en el almacén de entrada y las coloque en el almacén de salida y posteriormente tome la pieza del taladro una vez que esta ya ha sido procesada y la coloque en el almacén de salida, entonces se presenta un bloqueo en la secuencia de operaciones por la que debe pasar cada pieza en el sistema.

En la Figura 6.9 se muestra un cuadro de diálogo que aparece al seleccionar la opción Marking, donde se incluyen las etiquetas designadas para las piezas a procesar, y en cuanto a la opción de capacidad, conforme las piezas van saliendo del buffer, automáticamente se actualiza la capacidad que se le asigna.

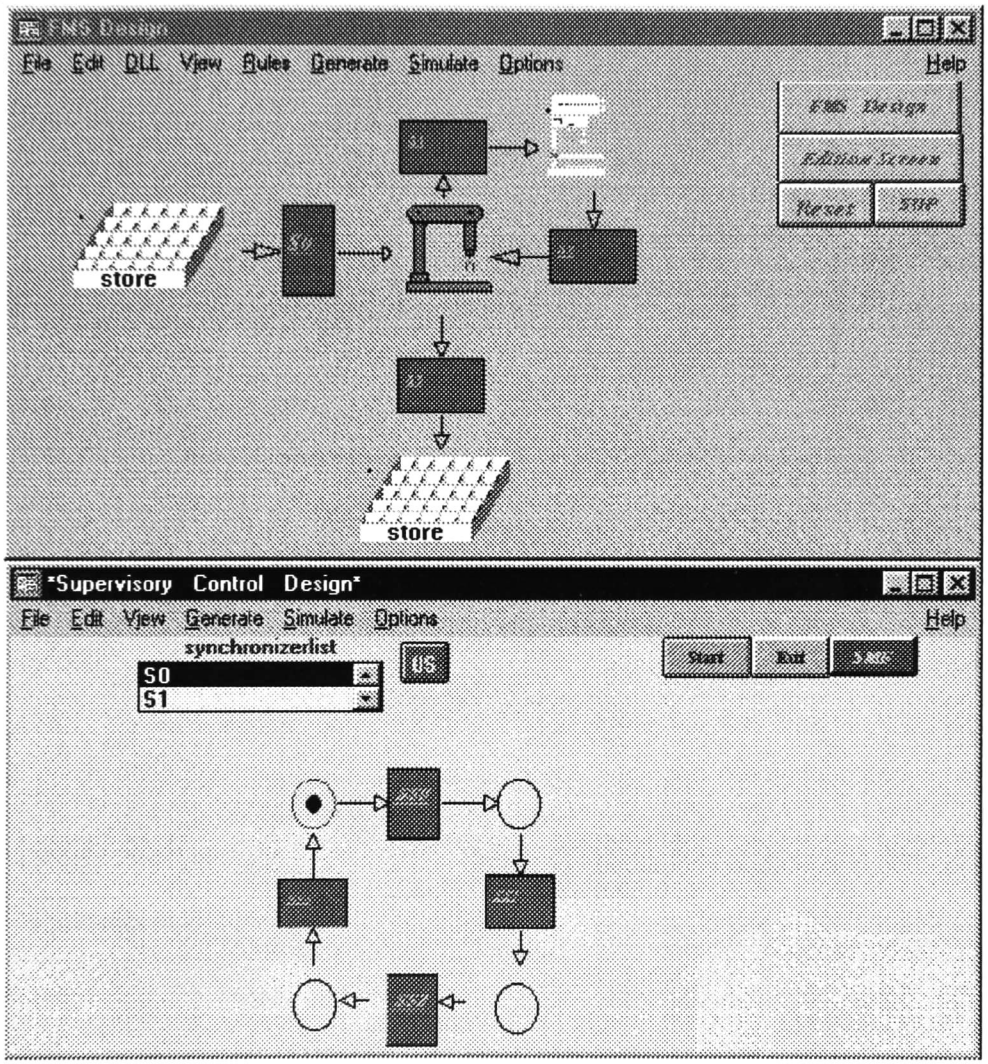


Figura 6.8: Ejemplo de SMF y control supervisorio.

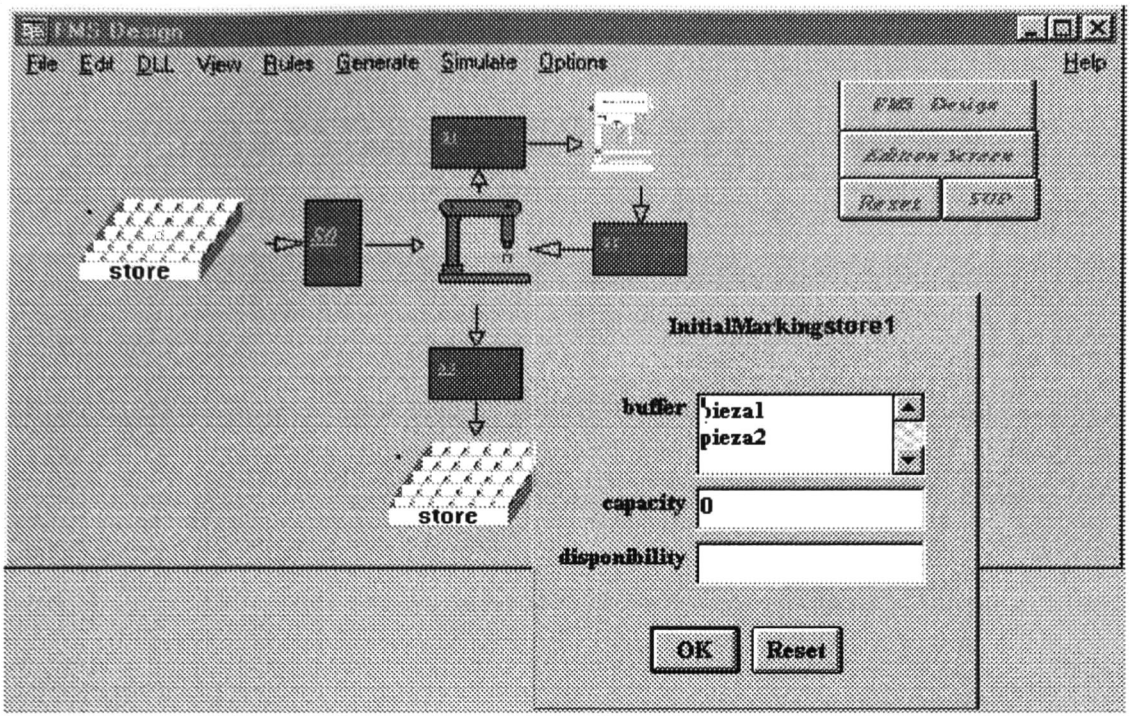


Figura 6.9: Marcado Inicial en el almacén de entrada.

Una vez cargada toda la información de marcado en todos los objetos de SMF, se procede a generar las Reglas, se selecciona del menú la opción Generate→RSMFSC, como se muestra en la Figura 6.10.

Enseguida para que el sistema evolucione, se selecciona del menú la opción Simulate, y se muestra en la Figura 6.11 el marcado actual para el Robot, donde también se aprecia la evolución en la RdP para el Supervisor, y así sucesivamente se sigue pulsando la opción Simulate, donde en las Figuras 6.12, 6.13 y 6.14, se muestra la evolución del SMF a través del menú de marcados para cada máquina del SMF y la RdP va mostrando el marcado de la secuencia permisible para el SMF.

Este programa tiene más bondades de las que aquí se presentaron, además existen funciones y elementos que se pueden agregar y/o mejorar

El objetivo del GDLS es el de ser un programa auxiliar para la simulación de secuencias de control supervisorio sobre los Sistemas de Manufactura Flexible.

Para cualquier sugerencia, duda o aclaración dirigirse a:

- Antonio Ramírez Treviño

  Cinvestav-Unidad Guadalajara

  E-Mail: art@gdl.cinvestav.mx

- M. Antonio López E.

  E-Mail: tono@gdl.cinvestav.mx

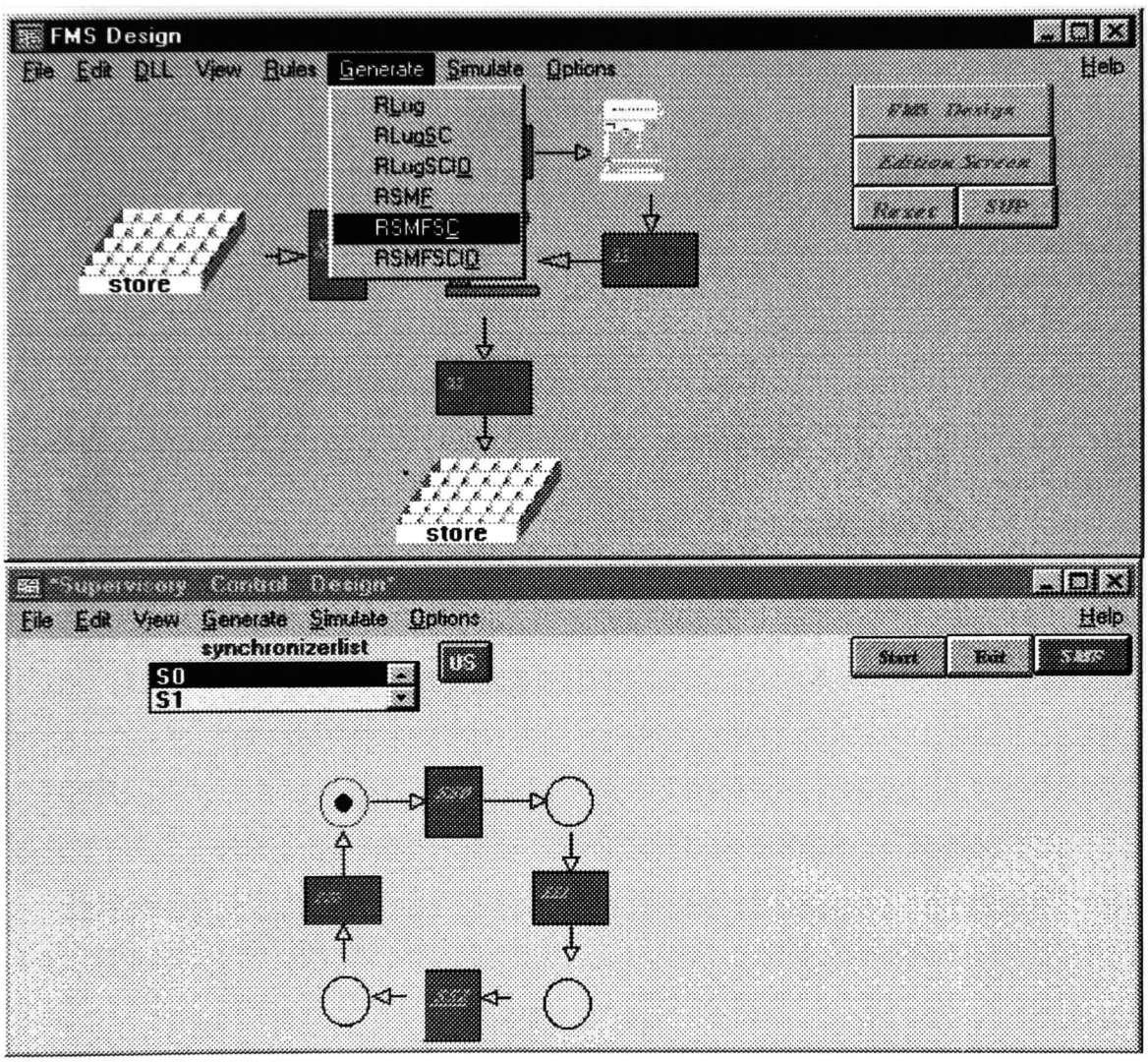


Figura 6.10: Generación de Reglas para el SMF incluyendo el Supervisor.

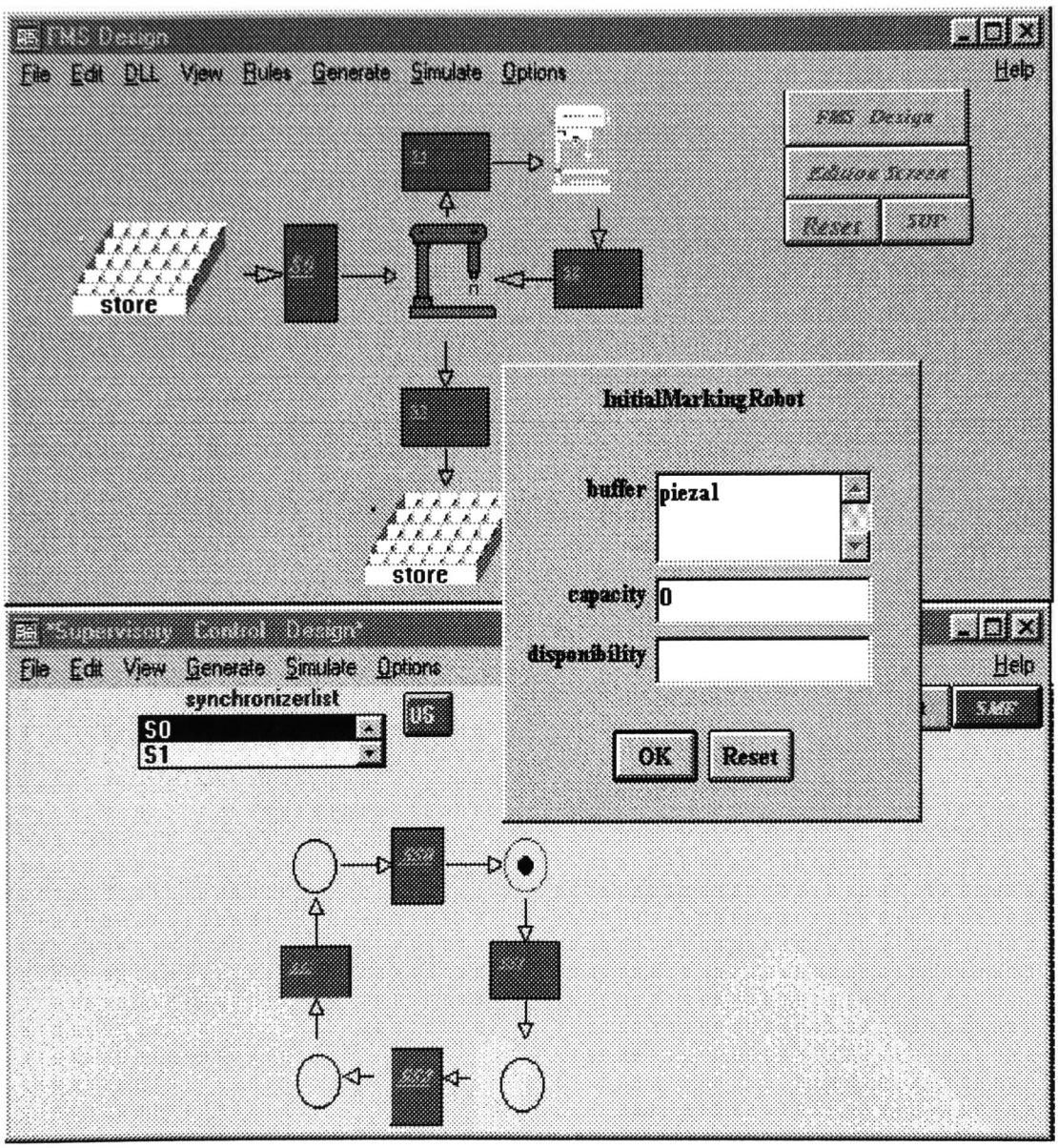


Figura 6.11: Fase Uno de Simulación de Evolución del SMF

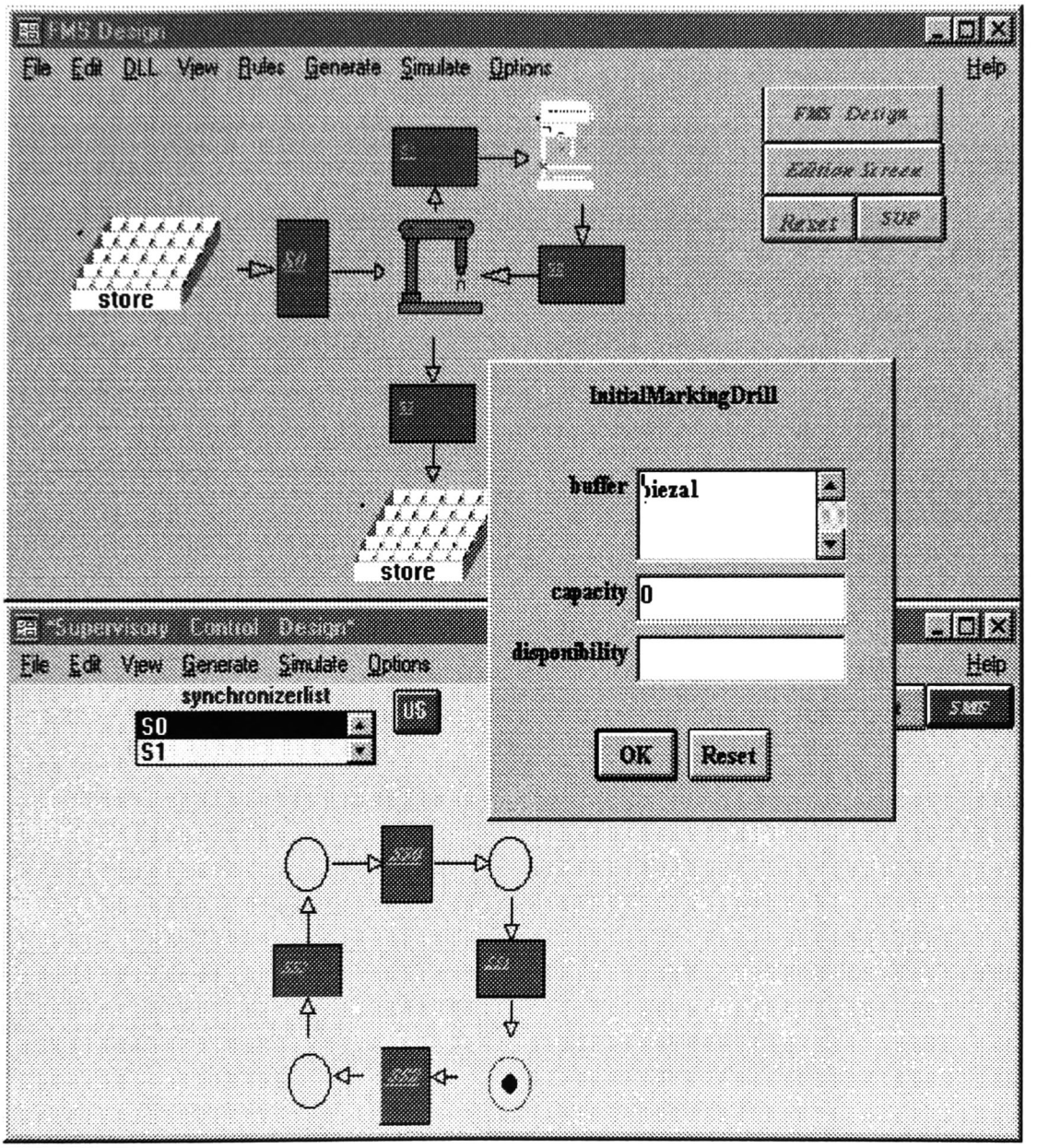


Figura 6.12: Fase Dos de Simulación de Evolución del SMF

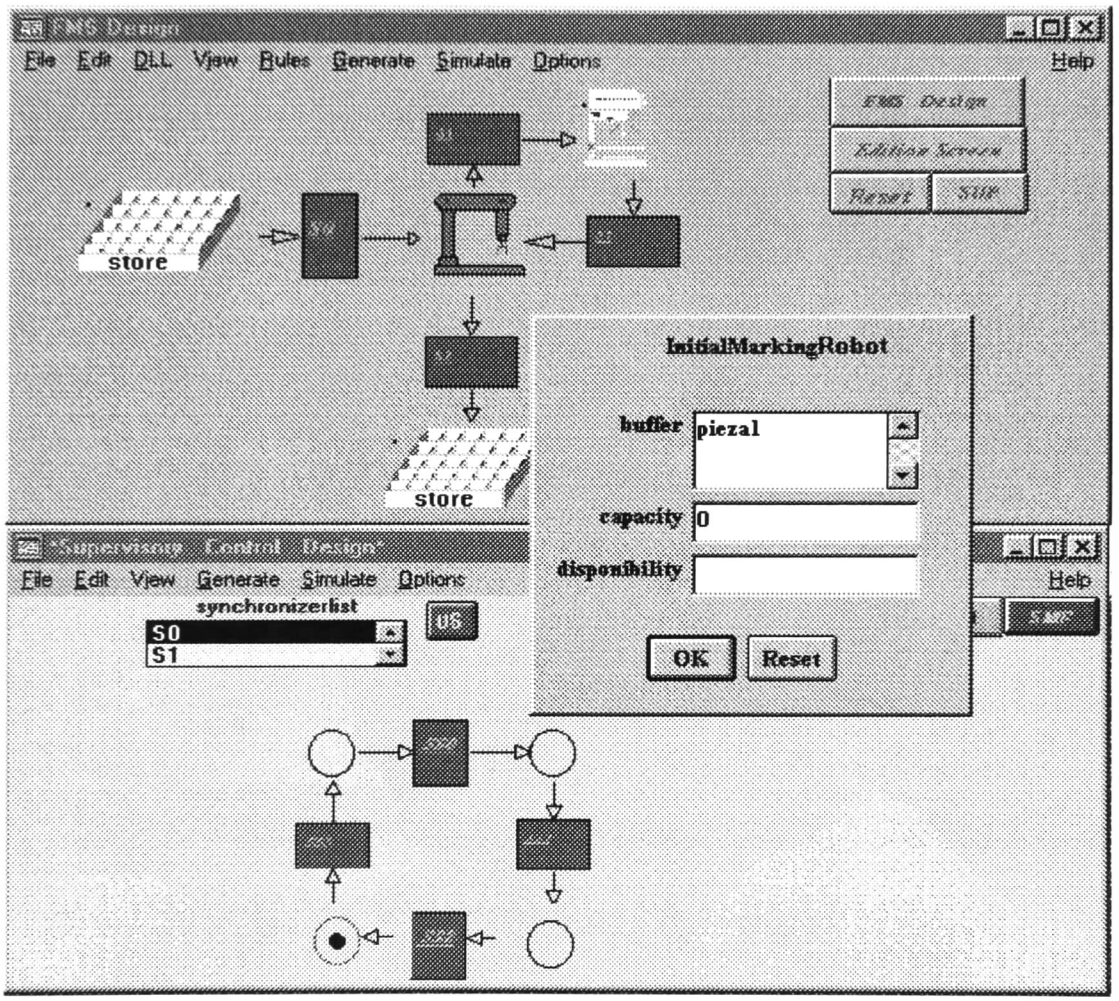


Figura 6.13: Fase Tres de Simulación de Evolución del SMF

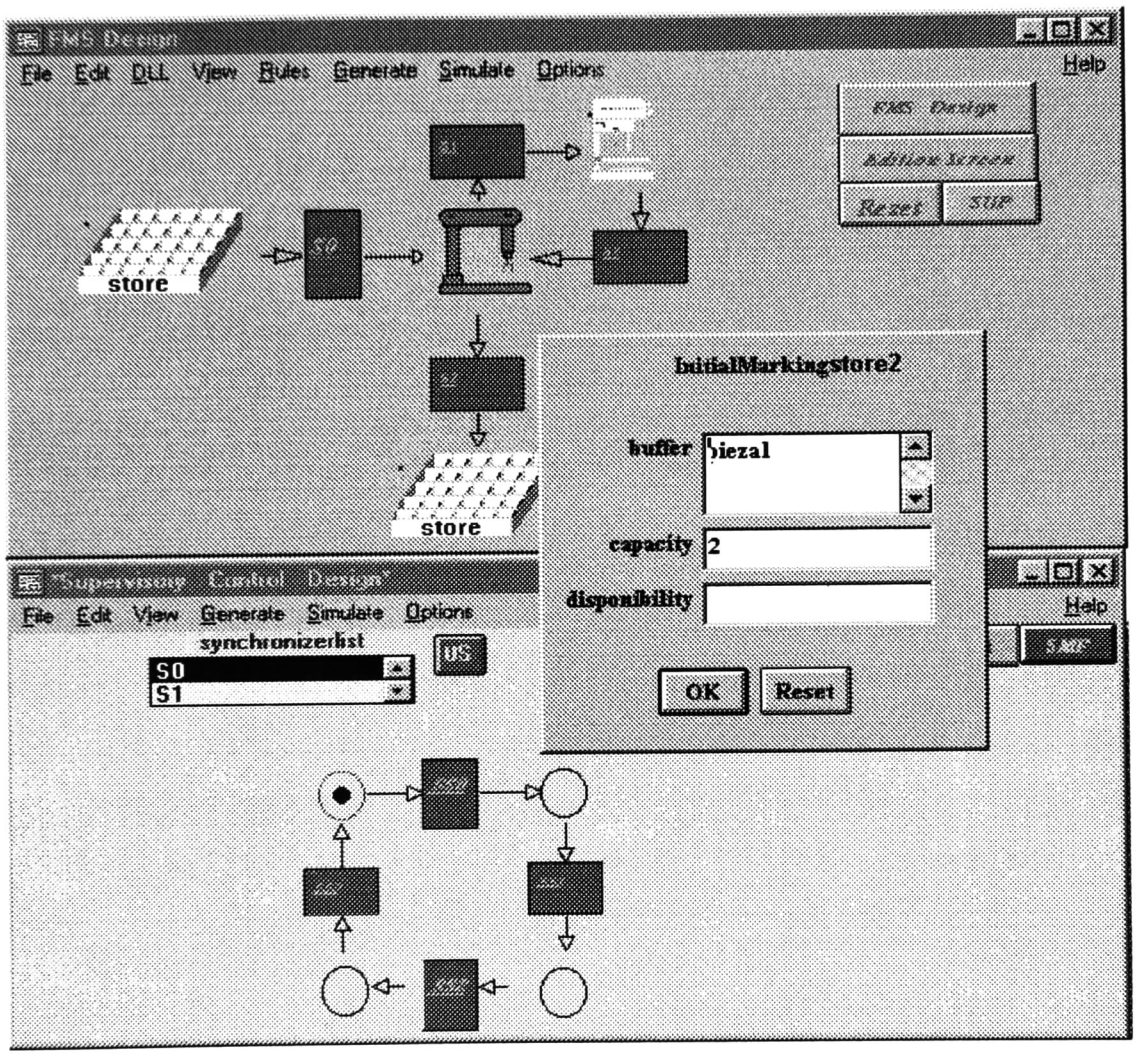


Figura 6.14: Fase Cuatro de Simulación de Evolución del SMF

- Rosa Ma. Córdova C.
  E-Mail: rossy@gdl.cinvestav.mx

Capítulo 7

Conclusiones y Trabajo Futuro

7.1 Conclusiones

- Los AF son la herramienta más recomendable y práctica si no incluyen sincronizaciones, exclusiones mutuas y paralelismos. Las RP son una herramienta que poseen un potencial de representación mayor que los AF, ya que permiten representar sincronizaciones, exclusiones mutuas y paralelismos. Gracias a las propiedades dinámicas y estructurales de las RP en algunos sistemas se puede hacer un análisis sencillo y claro del comportamiento que presentan.

- La metodología de modelado en RPI disminuye la complejidad del modelo en RP, y con respecto a los AF, permite la fácil interpretación y relación entre cada modelo del sistema con el respectivo elemento en el sistema real.

- La nueva definición de controlabilidad permite controlar sistemas cuando se pretende alcanzar un estado o mantener al sistema visitando un conjunto de estados, independientemente de la condición inicial del sistema.

- Se estudió la optimización de sistemas de eventos discretos modelados con RP. En particular se estudió la optimización de una RP por secciones a partir de la definición de pares inevitables sobre el grafo de alcanzabilidad que genera la RP, obteniéndose un teorema que permite saber cuando una RP puede ser optimizada localmente. Este teorema está basado en el grafo de alcanzabilidad, pero se mostró que la topología de algunas clases de redes permiten garantizar la OL, permitiendo su optimización de manera rápida.

- Al simulador de sistemas de manufactura flexible presentado en [López 97], se le agregó una capa de Software. Esta capa permite realizar control supervisor sobre los SMF. Además se agregó a este software las llamadas a puertos necesarias para comunicación con el medio ambiente. Desafortunadamente, este software no es adecuado para añadirle otra capa más, para realizar control óptimo.

7.2 Trabajo Futuro

- Determinar la clase de lenguajes no regulares generada por las RP que es cerrada bajo el operador supremo controlable.
- Obtener condiciones necesarias y suficientes para que un sistema modelado en RP posea la propiedad de optimalidad local.
- Implementar el simulador/controlador en ADA o C++. Se debe especificar claramente los objetos y sus funcionalidades, así como un motor de inferencias que permita incluir toma de decisiones basadas en heurísticas en la etapa de control óptimo.

Bibliografía

[Balemi 93] "Supervisory Control of a Rapid Thermal Multiprocessor", S. Balemi, G. J. Hoffmann, P. Gyugyi, H. Wong-Toi, and G. F. Franklin, Life Fellow, IEEE "Transactions on Automatic Control", Vol. 38, No. 7, July 1993

[Banks 84] "Discrete-Event System Simulation", Jerry Banks & John A.Carson, II, Prentice-Hall International series in Industrial and Systems Engineering, 1984

[Charbonnier 94] "The Supervisory Control of the Automated Manufacturing System of the AIP" Bertil A. Brandin and Francois E. Charbonnier, IEEE, 1994

[Desrochers 95] "Applications of Petri Nets in Manufacturing Systems", Alan A. Desrochers and Robert Y. Al-Jaar, IEEE Press Marketing, 1995

[Dym 91] "Knowledge Based Systems in Engineering", Clive L. Dym & Raymond E. Levitt, McGraw-Hill International Editions, International Edition 1991

[Giua 92] "Doctoral Tesis Giua", "Petri Nets as Discrete Event Models for Supervisory Control", Alessandro Giua, Rensselaer Polytechnic Institute, Computer and Systems Engineering, Troy, New York, August 1992

[Holloway 90] "Synthesis of Feedback Control Logic for a Class of Controlled Petri Nets", Lawrence E. Holloway and Bruce H. Krogh, IEEE Transactions on Automatic Control, Vol. 35, No. 5, May 1990

[Holloway 92] "On Closed-Loop Liveness of Discrete-Event Systems Under Maximally Permissive Control", Lawrence E. Holloway and Bruce H. Krogh, IEEE Transactions on Automatic Control, Vol. 37, No. 5, May 1992

[Hopcroft 79] "Introducción a la teoría de Autómatas, Lenguajes y Computación", John E.Hopcroft y Jeffrey D. Ullman, Traducción: Homero Flores Samaniego, Compañía Editorial Continental, S.A. de C.V. México, Marzo 1979

[Lafortune 90] "The Infimal Closed Controllable Superlanguage and Its Application in Supervisory Control" Stéphane Lafortune and Enke Chen, IEEE Transactions on Automatic Control, Vol. 35, No. 4, April 1990

[Laftit 92] "Optimization of invariant Criteria for Event Graph" Said Laftit, X. Xie and Jean-Marie Proth, IEEE Transactions on Automatic Control, Vol. 37, No.5, May 1992

[Lauzon 96] "Application of Discrete-Event-System Theory to flexible Manufacturing", S.C. Lauzon, A.K.L. Ma, J.K. Mills, and B. Benhabib, IEEE Control Systems, Vol.16, Febrero 1996, Pg. 41-48

[Lin 88] "Contollability and Observability in the State-Feedback Control of Discrete-Event Systems", Yong Li and W. M. Wonham, Proceedings of the 27th Conference on Decision and Control, Austin (Texas), December 1988

[Lin 90] "Decentralized Control and Coordination of Discrete-Event Systems with Partial Observation" Feng Lin and W. Murray Wonham, IEEE Transactions on Automatic Control, Vol. 35, No. 12, December 1990

[López 97] "Modelado de un Sistema de Manufactura Flexible" Marco Antonio López Enríquez, Departamento de Ingeniería Eléctrica, Sección de Control Automático, CINVESTAV, México, D. F., Mayo 1997

[López Mellado 97] "Introducción a las Redes de Petri", Ernesto López Mellado,Facultad de Ciencias Físico-Matemáticas, Octubre 1997

[Meda 98] "Identificación de Sistemas Dinámicos de Eventos Discretos Utilizando Redes de Petri" María Elena Meda Campaña, CINVESTAV, Guadalajara, México, Octubre 1998

[Passino 89] "On the Optimal Control of Discrete Event Systems" K. M. Passino and P. J. Antsaklis, Proceedings of the 28th Conference on Decision and Control, Tampa (Florida), December 1989

[Peterson 81] "Petri Net Theory and the Modeling of Systems" J. L. Peterson, Prentice Hall, Englewood Cliffs, New Jersey, 1981

[Proth 93] "Practice of Petri Nets in Manufacturing", F. DiCesare, G. Harhalakis, J. M. Proth, M. Silva, F. B. Vernadat, CHAPMAN & HALL, First Edition 1993. [Peterson 81] J.L. Peterson, "Petri Net Theory and the Modeling of Systems" Prentice-Hall, 1981

[Ramírez 90] "Tesis de Maestría Ramírez","Modelado de Tareas y Especificación de Controladores en Células Robotizadas" Antonio Ramírez Treviño, Departamento de Ingeniería Eléctrica, Sección de Control Automático, CINVESTAV. México, D.F., Agosto 1990

[Ramírez 93] "On Optimal Scheduling in DEDS", Antonio Ramírez, J. Campos y Manuel Silva, Proceedings of the 1993 IEEE International Conference on Robotics and Automation, pag. 821-826, Atlanta (USA), May 1993

[Ramírez 93a] "Tesis Doctoral Ramírez" "Scheduling en Redes de Petri", Antonio Ramírez Treviño, Departamento de Ingeniería Eléctrica e Informática, Centro Politécnico Superior de Ingenieros, Universidad de Zaragoza

[Russell 95] "Artificial Intelligence a Modern Approach", Stuart Russell & Peter Norvig, Prentice Hall Series in Artificial Intelligence, 1995

[Silva 85] "Las Redes de Petri: en la Automática y la Informática", Manuel Silva, Editorial AC, libros científicos y técnicos Madrid, 1985

[Sreenivas 92] "On Petri Net Models of Infinite State Supervisors", R. S. Sreenivas and B. H. Krogh, IEEE Transactions on automatic Control, Vol. 37, No. 2, February 1992

[Tadmor 89] "Control of Large Discrete Event Systems: Constructive Algorithms" Gilead Tadmor and Oded Maimon, IEEE Transactions on Automatic Control, Vol. 34, No. 11, November 1989

[Usio 90] "On the Existence of Finite State Supervisors in Discrete-Event Systems", Toshimitsu Usio, Proceedins of the 29th Conference on Decision and Control, Honolulu (Hawaii), December 1990

[Wonham 87] "Supervisory Control of a Class of Discrete Event Process" P. J. Ramadge and W. M. Wonham, Society for Industrial Applied Mathematics, Control and Optimization, Vol. 25, No. 1, January 1987

[Wonham 87a] "On the Supremal Controllable Sublanguage of a Given Language" W. M. Wonham and P. J. Ramadge, Society for Industrial Applied Mathematics, Control and Optimization, Vol. 25, No. 3, May 1987

[Wonham 88] "A Control Theory for Discrete-Event Systems", W.M. Wonham, Advanced Computing Concepts and Techniques in Control Engineering, M.J. Denham and A. J. Laub (eds.),Springer-Verlag, pag. 129-169, 1988.

[Wonham 96] "Notes on Control of Discrete-Event Systems", ECE 1636F/1637S 1996-1997, W.M. Wonham, Systems Control Group, Dept. of Electrical & Computer Engineering, University of Toronto.

CENTRO DE INVESTIGACION Y DE ESTUDIOS AVANZADOS DEL IPN
UNIDAD GUADALAJARA

El Jurado designado por el Laboratorio de Ingeniería Eléctrica y Ciencias de la Computación del Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional, aprobó la tesis: **Sistema de Pruebas para Modelado y Control de Sistemas de Eventos Discretos** el día 19 de Noviembre de 1998.

El jurado:

Dra. Ofelia Begovich Mendoza
Profesora Investigador 3A
CINVESTAV DEL IPN
Guadalajara

Dr. Luis Ernesto López Mellado
Profesor Investigador 3A
CINVESTAV DEL IPN
Guadalajara

Dr. Antonio Ramírez Treviño
Profesor Investigador 2A
CINVESTAV DEL IPN
Guadalajara

PROL. LOPEZ MATEOS SUR 590 APARTADO POSTAL 31-438 GUADALAJARA, JAL. 45090 TEL. (3) 684-1580 FAX (3) 684-1708