UNIVERSIDAD POLITÉCNICA DE VALENCIA

DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

TESIS DOCTORAL

Redes Reconfigurables. Modelización y Verificación

Presentada por:

Mª Luisa Llorens Agost

Dirigida por:

Dr. Javier Oliver Villarroya

Valencia, 2003

A mi abuela.
A mis padres.
A mi hermano.

Agradecimientos

Esta tesis ha sido el fruto de años de trabajo. No hubiera sido posible sin la ayuda de mi familia, amigos y compañeros. Por ello, quiero dar las gracias:

A mi director de Tesis, Javier Oliver, que fue quien me animó a comenzar los estudios de doctorado y quien ha hecho posible con su interés, su dedicación y su paciencia que esta tesis saliera adelante.

A mis compañeros del DSIC. En particular a Inma, por ser más que una compañera, por ser mi amiga. Y a Gabi, por apoyarme y por darme la solución a tantos problemas que, para mí, parecían no tenerla.

A mi familia, por estar siempre a mi lado, dándome su cariño, su apoyo y su comprensión. En especial a mis padres y a mi hermano, porque siempre han estado ahí cuando los he necesitado.

A mis amigos de siempre. Especialmente a Bea y a Lourdes, porque sé que siempre podré contar con ellas.

A todos... gracias.

Resumen

En esta Tesis Doctoral se aborda el problema de la *modelización* y la *verificación* de sistemas concurrentes sujetos a cambios dinámicos. El formalismo de base es el de las redes de Petri. En lo que concierne a la expresividad del modelo se busca un mecanismo que tenga en cuenta los cambios dinámicos estructurales de manera local, interna e incremental. Al mismo tiempo, las propiedades básicas de las redes de Petri (acotabilidad de lugares, alcanzabilidad, interbloqueo y vivacidad) deben continuar siendo decidibles para este modelo extendido. En general, lo que se gana normalmente en términos de expresividad se traduce en una pérdida en términos de propiedades decidibles. Hay que buscar, entonces, un equilibrio entre expresividad y computabilidad.

Las *gramáticas de grafos* y las *redes automodificantes de Valk* son las dos líneas de investigación origen de nuestro modelo general: los *sistemas de reescritura de redes*. Ambas líneas dan lugar a modelos que mejoran la expresividad de las redes de Petri para describir el cambio dinámico en sistemas concurrentes pero tienen el inconveniente de que casi todas las propiedades básicas decidibles de las redes de Petri se pierden. Por ello, para estos modelos extendidos no pueden construirse herramientas automáticas de verificación.

Los *sistemas de reescritura de redes* son una combinación de redes de Petri con sistemas de reescritura de grafos. Cada configuración del sistema es una red de Petri y un cambio de configuración es una regla de reescritura de grafos. La expresividad de los sistemas de reescritura de redes es la misma que la de la máquina de Turing, es decir, las propiedades básicas decidibles de las redes de Petri se pierden en estos sistemas, no siendo posible la verificación automática.

Las *redes reconfigurables* son una subclase de los sistemas de reescritura de redes equivalente formalmente a las redes de Petri, lo que asegura que todas las propiedades fundamentales de las redes de Petri siguen siendo decibibles para este modelo y, por tanto, es factible la verificación automática.

Resum

En aquesta Tesi Doctoral s' aborda el problema de la *modelització* i la *verificació* de sistemes concurrents subjectes a canvis dinàmics. El formalisme de base és el de les xarxes de Petri. Pel que fa a l' expressivitat del model es busca un mecanisme que tinga en compte els canvis dinàmics estructurals de manera local, interna i incremental. Al mateix temps, les propietats bàsiques de les xarxes de Petri (acotabilitat de llocs, assequibilitat, interbloqueig i vivacitat) deuen continuar sent decidibles per aquest model estès. En general, el que es guanya normalment en termes d' expressivitat es tradueix en una pèrdua en termes de propietats decidibles. Hi ha que buscar, aleshores, un equilibri entre expressivitat i computabilitat.

Les *gramàtiques de grafs* i les *xarxes automodificants de Valk* són les dues línies d' investigació origen del nostre model general: els *sistemes de reescritura de xarxes*. Ambdues línies donen lloc a models que milloren l' expressivitat de les xarxes de Petri per a descriure el canvi dinàmic en sistemes concurrents però tenen l' inconvenient de que quasi totes les propietats bàsiques decidibles de les xarxes de Petri es perden. Per això, per aquestos models estesos no poden construir-se ferramentes automàtiques de verificació.

Els *sistemes de reescritura de xarxes* són una combinació de xarxes de Petri amb sistemes de reescritura de grafs. Cada configuració del sistema és una xarxa de Petri i un canvi de configuració és una regla de reescritura de grafs. L' expressivitat dels sistemes de reescritura de xarxes és la mateixa que la de la màquina de Turing, és a dir, les propietats bàsiques decidibles de les xarxes de Petri es perden en aquestos sistemes, no sent possible la verificació automàtica.

Les *xarxes reconfigurables* són una subclasse dels sistemes de reescritura de xarxes equivalent formalment a les xarxes de Petri, la qual cosa assegura que totes les propietats fonamentals de les xarxes de Petri segueixen sent decibibles per aquest model i, per tant, és factible la verificació automàtica.

Abstract

In this Thesis we address the issues of *modelling* and *verification* of concurrent systems subject to dynamic changes using Petri net formalisms. As far as the expressiveness of the model is concerned a built-in and decentralized mechanism for handling structural changes is looked for. At the same time the basic decidable properties of Petri nets (place boundedness, reachability, deadlock and liveness) should remain decidable for the extended model. In general, the gain in terms of modelling power is usually paid by a loss of decidable properties. A trade-off needs to be found between expressiveness and computability.

The proposed general model of *net rewriting systems* arises from two different lines of research: *graph grammars* and *Valk's self-modifying nets*. Both proposals lead to models that enhance the expressiveness of the basic model of Petri nets so that it can support the description of concurrent systems subject to dynamic changes. However, most of the basic decidable properties of Petri nets are lost for these extended models. Therefore no automatic verification tools could be implemented in the context of these models.

Net rewriting systems combine Petri nets with graph rewriting systems. The idea is to describe a system configuration as a Petri net and a change of configuration as a graph rewriting rule. The class of net rewriting systems is Turing powerful, that is, the basic decidable properties of Petri nets are lost and thus automatic verification is not longer possible for this larger class.

Reconfigurable nets are a subclass of net rewriting systems formally equivalent to Petri nets. This equivalence ensures that all the fundamental properties of Petri nets are still decidable for reconfigurable nets and this model is thus amenable to automatic verification tools.

Tabla de contenidos

Agradecimientos v

Resumen vii

Resum ix

Abstract xi

Tabla de contenidos xiii

Índice de figuras xvii

Índice de tablas xxi

Índice de algoritmos xxiii

Prólogo 1

1. Redes de Petri 5

1.1. Conceptos Básicos 6

1.1.1. Subclases de Redes de Petri más habituales 9

1.2. Propiedades 10

1.2.1. Propiedades de Comportamiento 11

1.2.2. Propiedades Estructurales 13
1.3. Análisis de redes de Petri 14
1.3.1. Árbol de Cobertura (o Alcanzabilidad) 14
1.3.2. Ecuaciones Matriciales 17
1.3.3. Técnicas de reducción y síntesis 24
1.4. Decidibilidad y Complejidad 24

2. Líneas de Investigación 37
2.1. Redes Automodificantes 37
2.2. Gramáticas de Grafos 40

3. Revisión de Trabajos Relacionados 49
3.1. Las Redes Móviles 50
3.2. Las Redes Dinámicas 52
3.3. Las Redes Δ 54
3.4. El Formalismo POP 56
3.5. Los Autómatas Cooperativos 58

4. Redes Reconfigurables. Propuesta inicial 63
4.1. Cambios Dinámicos en Sistemas Workflow: Redes Reconfigurables 65
4.1.1. Sistemas Workflow 65
4.1.2. Redes Reconfigurables 66
4.1.3. Acotamiento de una Red Reconfigurable 74
4.1.4. Redes Reconfigurables como Redes Automodificantes . . 80
4.2. Redes Reconfigurables Reversibles 83
4.2.1. Redes de Petri Controladas por un Autómata 84
4.2.2. Redes Reconfigurables Reversibles 90

4.2.3. Redes Reconfigurables Reversibles y Redes de Petri Controladas por un Autómata 94

4.2.4. Redes Reconfigurables Reversibles y Redes de Petri Estratificadas 98

4.3. Algunos ejemplos 107

4.3.1. Envío de paquetes de información en una red de transmisión 107

4.3.2. Una fábrica de ensamblaje con diferentes modos de operación 109

4.3.3. Flujos en una red 113

5. Sistemas de Reescritura de Redes. Redes Reconfigurables 117

5.1. Sistema de Reescritura de Redes 118

5.2. Redes Reconfigurables 123

5.3. Simulación de Máquinas de Turing con Sistemas de Reescritura de Redes 129

5.3.1. Sistemas de Reescritura de Redes K-ordenadas 130

5.3.2. Simulación de una Máquina de Turing 135

5.4. Implementación de Redes Reconfigurables con Redes de Petri . . 140

5.5. Ejemplo 146

6. Verificación 153

6.1. Árbol de Cobertura de una Red Reconfigurable 155

6.2. Ecuación de Estados 157

7. Desarrollo de una herramienta software 161

7.1. Editor gráfico 164

7.1.1. PED. Un editor de redes de Petri 164

7.1.2. Diseño de la red reconfigurable en PED 165

7.2. Simulador 166
7.2.1. Simulador A 166
7.2.2. Simulador B 167
7.3. Analizador 167
7.3.1. Analizador A 167
7.3.2. Analizador B 168

8. Conclusiones y trabajos futuros **169**

Referencias **174**

Índice de figuras

1.1. Red de Petri (a) 7

1.2. Marcado de la red de Petri de la Figura 1.1 después del disparo de la transición t_1 9

1.3. Árbol de cobertura (alcanzabilidad) de la red de Petri de la Figura 1.1 16

1.4. Red de Petri (b) 17

1.5. Árbol de cobertura de la red de Petri de la Figura 1.4 17

1.6. Seis transformaciones que preservan vivacidad, seguridad y acotamiento [Mur89] 25

1.7. Test de cero 27

1.8. Arco inhibidor 27

1.9. Puesta a cero de un lugar p 27

1.10. Arco reset 28

1.11. Transferencia del contenido del lugar p al lugar q 28

1.12. Arco de transferencia 29

1.13. Arco doble 29

1.14. Clases de Decidibilidad 35

2.1. Implementación de arcos inhibidores, reset y de transferencia en redes automodificantes 40

2.2. Diagrama de Pegado 43

2.3. Derivación directa como diagrama de pegado doble 45

2.4. Un paso de reescritura de grafos en la aproximación de doble pegado 46

2.5. (a) Una transición de una red de Petri. (b) La correspondiente producción de grafos. 47

2.6. (a) Disparo de una transición de una red de Petri. (b) Diagrama de pegado doble correspondiente. 47

4.1. Cambio *synthetic cut-over* 70

4.2. Fragmento del grafo de marcado de la red reconfigurable de la Figura 4.1 71

4.3. Traducción de una red reconfigurable en una red automodificante equivalente 81

4.4. Desde una red impura a una red pura 87

4.5. Red Reconfigurable Reversible de la Figura 4.1 como una Red de Petri Controlada por un Autómata 95

4.6. Red de Petri estratificada equivalente a la red reconfigurable de la Figura 4.1 100

4.7. Red reconfigurable reversible como una composición en cascada de redes de Petri 103

4.8. Red de Petri estratificada equivalente a una red reconfigurable reversible simple 106

4.9. Red reconfigurable reversible marcada 108

4.10. Las 12 configuraciones de la red reconfigurable reversible de la Figura 4.9 109

4.11. RPCA de la red reconfigurable reversible de la Figura 4.9 110

4.12. Fábrica de ensamblaje con diferentes modos de operación 111

4.13. Cambio del modo I al modo II 113

4.14. Un flujo (en la parte derecha) del sistema workflow (en la parte izquierda) 114

4.15. Un conjunto generador de ciclos 114
4.16. Red asociada al conjunto generador de ciclos de la Figura 4.15 . 115
4.17. Comportamiento de un nodo de la red 115

5.1. Sistema de Reescritura de Redes modelizando un sistema de impresoras 120
5.2. Cambio de configuración debido a la regla de reescritura *triple* . 122
5.3. Sistema con dos líneas *half-duplex* entre dos productores/consumidores 126
5.4. Regla de reescritura R_1 127
5.5. Regla de reescritura R_2 127
5.6. Regla de reescritura R_3 128
5.7. Regla de reescritura R_4 128
5.8. Estado obtenido tras aplicar la regla R_1 al estado representado en la Figura 5.3 129
5.9. Fragmento del grafo de estados del sistema de la Figura 5.3 . . . 129
5.10. Representación del orden en una transición y un lugar 134
5.11. Implementación de una máquina de Turing con un sistema de reescritura de redes K-ordenadas 137
5.12. Parte de la red de Petri equivalente a la red reconfigurable del Ejemplo 166 145
5.13. Estado inicial de la red de transmisión 147
5.14. Las 8 reglas de reescritura de la red reconfigurable de la Figura 5.13 149
5.15. Las 12 configuraciones de la red reconfigurable de la Figura 5.13 150
5.16. Grafo de configuración de la red reconfigurable de la Figura 5.13 151
5.17. Nuevo estado alcanzado tras la aplicación de la regla R_8 al estado de la Figura 5.13 151
5.18. Red de Petri equivalente a la Figura 5.13 152

Índice de tablas

1.1. Propiedades Estructurales. Condiciones necesarias y suficientes . 21
1.2. Tabla Dufourd 33
1.3. Tabla Dufourd completa 34

Índice de algoritmos

1. Árbol de Cobertura de una Red de Petri 15
2. Derivación Directa 45
3. Árbol de Cobertura de una Red Reconfigurable Marcada 76
4. Árbol de Cobertura de una RPCA 90
5. Árbol de Cobertura de una RRR 99
6. Red de Petri equivalente a una Red Reconfigurable 142
7. Árbol de Cobertura de una Red Reconfigurable 156

Prólogo

Las *redes de Petri* [Mur89][Pet81] son un formalismo para modelizar, analizar, simular, controlar y evaluar el comportamiento de sistemas concurrentes y distribuidos. Han jugado un papel importante en el desarrollo de estos sistemas ya que, entre otras posibilidades, pueden considerarse como un lenguaje formal y gráfico para su modelización. Además, las redes de Petri tienen una sólida base matemática que permite el análisis cualitativo de las propiedades de tales sistemas como el interbloqueo y la acotabilidad. Sin embargo, el formalismo de las redes de Petri no ofrece una manera directa de representar características tales como cambios dinámicos de actividades, migración de tareas, modos de operación múltiples, etc., que son importantes en el diseño de sistemas concurrentes. Por ello, para modelizar sistemas concurrentes y distribuidos reales se utilizan extensiones de redes de Petri diseñadas para permitir la modelización de alguna de las características anteriores. Con estas extensiones, normalmente, lo que se gana en poder de modelado se paga en pérdida de propiedades decidibles. Lo que hay que conseguir es, por tanto, un equilibrio entre expresividad y computabilidad.

El objetivo fundamental que ha guiado el desarrollo de la Tesis Doctoral que aquí se presenta es la *modelización* y *verificación* de sistemas concurrentes sujetos a cambios dinámicos. Vamos a centrarnos en un modelo de red de Petri que va a permitir el cambio dinámico en la propia estructura de la red. Dinámico significa que un proceso puede reiniciarse en un momento dado sin que el sistema tenga que pararse, demorarse o abortar y las transacciones en progreso pueden continuar en medio del cambio estructural. Además, es importante que el mecanismo que maneje el cambio dinámico esté representado explícitamente en el modelo para que en cada etapa del desarrollo del producto, los diseñadores puedan experimentar el efecto de los cambios estructurales, por ejemplo, utilizando prototipos. Esto significa que los cambios estructurales se tienen en cuenta desde el inicio del proceso de diseño en lugar de ser tratados por un sistema global y externo, por ejemplo, algún mecanismo de manejo de excepciones, diseñado y añadido al modelo describiendo el comportamiento normal del sistema. Por tanto, en nuestro modelo se fomenta una descripción de los cambios *interna* e *incremental* sobre una descripción *externa* y *uniforme*, y un tratamiento de los cambios *local* sobre un tratamiento *global*, que suele ser la manera habitual de abordarlo.

El modelo que aquí se propone son los *sistemas de reescritura de redes*, que tienen su origen en dos líneas de investigación diferentes ambas relacionadas con el formalismo de las redes de Petri y cuyo propósito es mejorar la expresividad del modelo básico de las redes de Petri para soportar la descripción de sistemas concurrentes sujetos a cambios dinámicos. La primera línea de investigación estudia la manera de fusionar redes de Petri con Gramáticas de Grafos [Sch93][Cor95][Bal00], mientras que la segunda, representada en particular por las redes Automodificantes de R. Valk [Val78a, Val78b, Val81], estudia redes de Petri cuyas relaciones de flujo pueden cambiar en tiempo de ejecución. Ambas propuestas dan lugar a modelos expresivos que aportan indudables ventajas con respecto al resto de los modelos existentes. Sin embargo, la mayoría (si no todas) las propiedades básicas decidibles de las redes de Petri (acotabilidad de lugares, alcanzabilidad, interbloqueo y vivacidad) se pierden en estos modelos. Como consecuencia, no pueden construirse herramientas automáticas de verificación para dichos modelos extendidos. Las *redes reconfigurables*, una subclase de los sistemas de reescritura de redes, intentan combinar las características más significativas de las dos aproximaciones de manera que las propiedades fundamentales sean decidibles, con lo que en este modelo será factible la verificación automática.

Cabe resaltar que el origen del trabajo desarrollado en la presente Tesis surge como consecuencia de la colaboración en el proyecto *Paragraphe* del centro de investigación IRISA de Rennes (Francia), perteneciente al INRIA francés. En concreto, uno de sus miembros, el doctor Eric Badouel ha participado activamente en gran parte del proceso.

Esta disertación se estructura como sigue.

En el Capítulo 1 se presenta el formalismo de las redes de Petri (definiciones formales, clasificación, propiedades, técnicas de análisis y estudios sobre decidibilidad y complejidad).

En el Capítulo 2 se presentan las dos líneas de investigación origen del modelo de los sistemas de reescritura de redes y de las redes reconfigurables, las redes Automodificantes de R. Valk [Val78a, Val78b, Val81] y las Gramáticas de Grafos [Ehr87][Sch93][Cor95][Bal00].

En el Capítulo 3 se presentan otros modelos existentes en la literatura que se basan en la misma idea para describir sistemas concurrentes dinámicos: las *redes móviles* de Asperti y Busi [AB96], las *redes dinámicas* de Buscemi y Sassone [BS01], las *redes* Δ de Gradit y Vernadat [GVA99, GV99], el *formalismo POP* de Engelfriet, Leih y Rozenberg [ELR91] y los *autómatas cooperativos* de Badouel, Darondeau y Tokmakoff [BDT99].

En el Capítulo 4 se presentan los estudios previos sobre redes reconfigurables basados en parte de los trabajos realizados por Eric Badouel y Javier Oliver en [BO98a] y [BO99] y que fundamentan la presente Tesis.

En el Capítulo 5 se introduce la definición de Sistemas de Reescritura de Redes y la definición de red Reconfigurable como una subclase de sistema de reescritura de redes. Para las redes reconfigurables demostramos que su expresividad y la de las redes de Petri es equivalente, pero con las redes reconfigurables podemos modelizar más fácil y directamente sistemas que cambian su estructura dinámicamente. Así pues, todas las propiedades fundamentales de las redes de Petri siguen siendo decidibles para las redes reconfigurables. Para la clase de los sistemas de reescritura de redes demostramos que tienen el poder expresivo de la máquina de Turing. Por tanto, para esta clase no es posible encontrar herramientas de verificación automática, mientras que sí que lo es para la subclase de las redes reconfigurables.

En el Capítulo 6, partiendo de la equivalencia entre redes de Petri y redes reconfigurables y de que el tamaño de un sistema modelizado con una red reconfigurable es mucho menor que el tamaño del mismo sistema modelizado mediante una red de Petri, se presentan distintas técnicas de verificación de propiedades de las redes de Petri implementadas directamente en el modelo de las redes reconfigurables.

En el Capítulo 7 se muestra el desarrollo de una herramienta software de simulación y verificación para el modelo de las redes reconfigurables, que hace uso de las técnicas presentadas en el Capítulo 6 para el análisis de propiedades.

Por último, en el Capítulo 8 se presentan las conclusiones obtenidas tras la realización del presente trabajo de tesis y se exponen brevemente las líneas de trabajo futuras para la ampliación de ciertos aspectos interesantes que quedan fuera del alcance de la presente disertación.

Capítulo 1

Redes de Petri

Las redes de Petri, introducidas por Carl Adam Petri a principios de los años 60 [Pet62][Rei85], son una herramienta gráfica y matemática para el estudio de un gran número de sistemas: son uno de los formalismos más ampliamente aceptados para modelizar sistemas concurrentes y distribuidos. El éxito de las redes de Petri en los últimos 40 años puede medirse no sólo por las incontables situaciones en las que se pueden aplicar sino también por el desarrollo de aspectos teóricos, que van desde un análisis completo de distintos fenómenos en los modelos más simples a la definición de clases de redes con mayor expresividad (y más complejas). La principal aplicación de las redes de Petri es la *modelización* y el *análisis* de sistemas con componentes concurrentes que interactúan. Un modelo es una representación (en términos matemáticos) de las características más importantes del objeto o sistema de estudio. Manipulando esta representación, se pueden obtener nuevos conocimientos del sistema modelado sin ningún coste o peligro para el sistema real. Sin embargo, el modelado por sí solo sirve de poco. Es necesario analizar el sistema modelado. El sistema se modela primero como una red de Petri y después, este modelo se analiza. Este análisis nos lleva a una mejor comprensión del comportamiento del sistema modelado. Para realizar el análisis de las propiedades de una red de Petri se han desarrollado diferentes técnicas, automáticas o no, según los casos, que permiten la *verificación* de las propiedades que el sistema construido posea. De hecho, una línea de trabajo muy importante en los últimos años ha sido la construcción de herramientas automáticas de verificación para las diferentes extensiones que sobre el modelo de las redes de Petri se han propuesto.

Este capítulo está dedicado íntegramente a las redes de Petri, y constituye la base para el desarrollo del resto de capítulos. En la Sección 1.1 presentamos los conceptos básicos de las redes de Petri, incluyendo la definición de las subclases de redes de Petri más habituales. En la Sección 1.2 mostramos los distintos tipos de propiedades que se pueden estudiar con redes de Petri. En la Sección 1.3 exponemos los métodos existentes de análisis de redes de Petri. Por último, dedicamos la Sección 1.4 a presentar los tópicos sobre decidibilidad y complejidad de problemas de verificación de redes de Petri.

1.1. Conceptos Básicos

Definición 1 (Red de Petri [Mur89][Rei85]) *Una* red de Petri *es una tupla $N = (P, T, F)$ donde $P = \{p_1, p_2, \ldots, p_m\}$ es un conjunto finito de lugares, $T = \{t_1, t_2, \ldots, t_n\}$ es un conjunto finito de transiciones, tales que $P \cap T = \emptyset$ y $P \cup T \neq \emptyset$ y $F : (P \times T) \cup (T \times P) \rightarrow \{0, 1, 2, 3, \ldots\}$ es un conjunto de arcos ponderados (o relación de flujo).*

Definición 2 (Marcado) *Un* marcado *es una aplicación $M : P \rightarrow \{0, 1, 2, 3, \ldots\}$ que asigna a cada lugar un entero no negativo (*token*), de forma que decimos que un lugar p* está marcado *con k tokens si el marcado asigna al lugar p un entero k. Un marcado M se representa mediante un vector con tantas componentes como lugares tenga la red. El número de tokens en el lugar p será $M(p)$.*

Definición 3 (Red de Petri Marcada) *Una* red de Petri marcada *es un par (N, M) formado por una red de Petri N y un marcado M.*

A una *red de Petri* podemos asociarle un grafo dirigido con dos clases disjuntas de nodos, los *lugares* y las *transiciones*. Un círculo ◯ representa un lugar y una barra ▯ representa una transición. Los arcos dirigidos (flechas) conectan lugares y transiciones. Algunos arcos van desde un lugar a una transición y otros desde una transición a un lugar. Un arco dirigido desde un lugar p a una transición t define p como un *lugar de entrada* para t. Un *lugar de salida* se indica con un arco desde la transición al lugar. Una transición tiene un determinado número de *lugares de entrada* (o *precondiciones*) y de *lugares de salida* (o *postcondiciones*). Los arcos se etiquetan con sus pesos (enteros positivos). Si una de esas etiquetas se omite, significa que el arco tiene peso uno. Los tokens se representan como puntos negros • en los lugares.

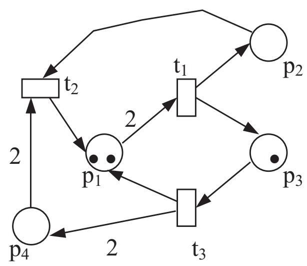


Figura 1.1: Red de Petri (a)

Definición 4 (Preset de una transición t) *El* preset *de una transición t es el conjunto de todos los lugares de entrada a la transición t,* $^\bullet t = \{\forall p \in P \mid F(p,t) \neq 0\}$.

Definición 5 (Postset de una transición t) *El* postset *de una transición t es el conjunto de todos los lugares de salida de la transición t,* $t^\bullet = \{\forall p \in P \mid F(t,p) \neq 0\}$.

Definición 6 (Preset de un lugar p) *El* preset *de un lugar p es el conjunto de todas las transiciones de entrada al lugar p,* $^\bullet p = \{\forall t \in T \mid F(t,p) \neq 0\}$.

Definición 7 (Postset de un lugar p) *El* postset *de un lugar p es el conjunto de todas las transiciones de salida del lugar p,* $p^\bullet = \{\forall t \in T \mid F(p,t) \neq 0\}$.

Ejemplo 8 *Para la red de Petri* (N, M) *de la Figura 1.1, los conjuntos preset y postset de transiciones y lugares son los siguientes:*

$$\begin{array}{llll}
^\bullet t_1 = \{p_1\} & t_1^\bullet = \{p_2, p_3\} & ^\bullet p_1 = \{t_2, t_3\} & p_1^\bullet = \{t_1\} \\
^\bullet t_2 = \{p_2, p_4\} & t_2^\bullet = \{p_1\} & ^\bullet p_2 = \{t_1\} & p_2^\bullet = \{t_2\} \\
^\bullet t_3 = \{p_3\} & t_3^\bullet = \{p_1, p_4\} & ^\bullet p_3 = \{t_1\} & p_3^\bullet = \{t_3\} \\
 & & ^\bullet p_4 = \{t_3\} & p_4^\bullet = \{t_2\}
\end{array}$$

El cambio en el marcado de la red es el que simula el comportamiento dinámico de un sistema modelado mediante una red de Petri. Una red de Petri se ejecuta por el *disparo* de transiciones. Una transición se *dispara* eliminando tokens de sus lugares de entrada y creando nuevos tokens distribuidos por sus lugares de salida.

Una transición puede dispararse si está *habilitada*. Una transición está habilitada si en cada uno de sus lugares de entrada hay, al menos, tantos tokens como el peso de los arcos que los conectan con la transición. Formalmente,

Definición 9 (Transición Habilitada) *Una transición t se dice* habilitada *si cada lugar de entrada p de t está marcado con al menos $F(p,t)$ tokens, es decir, con el entero que representa el peso del arco de p a t,*

$$\forall p \in {}^{\bullet}t \quad M(p) \geq F(p,t)$$

El *disparo* de una transición habilitada hace que se eliminen tantos tokens de sus lugares de entrada como indique el peso de los arcos que los conectan con la transición, y hace que en sus lugares de salida se depositen tantos tokens como indique el peso de los arcos que conectan dichos lugares con la transición.

Disparar una transición cambia el marcado M de una red de Petri a un nuevo marcado M'. Puesto que únicamente pueden dispararse las transiciones habilitadas, el número de tokens de cada lugar siempre será positivo cuando una transición se dispare. Si no hay tokens suficientes en alguno de los lugares de entrada de una transición, entonces la transición no estará habilitada y no se disparará.

Definición 10 (Disparo de una transición) *Una transición puede* dispararse *siempre que esté habilitada.*

El disparo *de una transición t habilitada en un marcado M elimina $F(p,t)$ tokens de cada lugar de entrada p de t, y añade $F(t,p)$ tokens a cada lugar de salida p de t, donde $F(t,p)$ es el peso del arco de t a p.*

El marcado M' resultado del disparo de una transición t habilitada en un marcado M, $M[t\rangle M'$, se define como sigue

$$\begin{array}{ll}
M'(p) = M(p) - F(p,t) + F(p,t) & \forall p \in ({}^{\bullet}t \cap t^{\bullet}) \\
M'(p) = M(p) - F(p,t) & \forall p \in {}^{\bullet}t \backslash t^{\bullet} \\
M'(p) = M(p) + F(t,p) & \forall p \in t^{\bullet} \backslash {}^{\bullet}t \\
M'(p) = M(p) & \textit{en otro caso}
\end{array}$$

Ejemplo 11 *Las transiciones habilitadas en la red de Petri de la Figura 1.1 son t_1 y t_3. Si se dispara la transición t_1, el marcado de la red $M = (2,0,1,0)$ cambia al marcado $M' = (0,1,2,0)$ como se muestra en la Figura 1.2.*

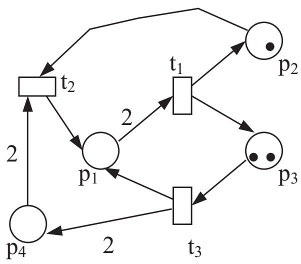


Figura 1.2: Marcado de la red de Petri de la Figura 1.1 después del disparo de la transición t_1

1.1.1. Subclases de Redes de Petri más habituales

Definición 12 (Red de Petri Pura) *Una red de Petri se llama* pura *si no tiene bucles, es decir,* $\forall t \in T\ \{^\bullet t\} \cap \{t^\bullet\} = \emptyset$*, los lugares de entrada de* t *no son a la vez lugares de salida de* t*.*

Definición 13 (Red de Petri Ordinaria) *Una red de Petri se dice* ordinaria *si el peso de todos sus arcos es* 1*.*

Definición 14 (Red de Petri Simple) *Una red de Petri* simple *es una red de Petri ordinaria tal que:* $p_1^\bullet \cap p_2^\bullet \neq \emptyset \Rightarrow (p_1^\bullet \subseteq p_2^\bullet) \vee (p_2^\bullet \subseteq p_1^\bullet)$*.*

Definición 15 (Grafo Marcado) *Un* grafo marcado *es una red de Petri ordinaria en la que cada lugar tiene exactamente una transición de entrada y una de salida, es decir,* $|^\bullet p| = |p^\bullet| = 1\ \forall p \in P$*.*

Definición 16 (Máquina de Estados) *Una* máquina de estados *es una red de Petri ordinaria en la que cada transición tiene exactamente un lugar de entrada y uno de salida, es decir,* $|^\bullet t| = |t^\bullet| = 1\ \forall t \in T$*.*

Definición 17 (Red de Libre Elección (Free-Choice Net)) *Una* red de libre elección *es una red de Petri ordinaria tal que un arco desde un lugar es el único arco saliente o es la única entrada a una transición, es decir,* $\forall t_j \in T$ *y* $p_i \in {}^\bullet t_j$*,* $^\bullet t_j = \{p_i\}$ *o* $p_i^\bullet = \{t_j\}$*.*

Definición 18 (Red de Petri Lugar/Transición (Red L/T)) *Una* red de Petri Lugar/Transición *es una 6-tupla* $N = (P, T, F, K, M, W)$ *tal que*

- (P, T, F) *es una red finita siendo* P *y* T *un conjunto de lugares y transiciones, respectivamente;*
- $K : P \rightarrow \mathbb{N} \cup \{\omega\}$ *da una capacidad (posiblemente ilimitada) a cada lugar;*
- $W : F \rightarrow \mathbb{N} \setminus \{0\}$ *asigna un peso a cada arco de la red;*
- $M : P \rightarrow \mathbb{N} \cup \{\omega\}$ *es el marcado inicial, respetando las capacidades, es decir,* $M(p) \leq K(p)$ *para todo* $p \in P$.

En la siguiente definición, damos la regla de disparo para redes L/T.

Definición 19 (Regla de Disparo para redes L/T) *Sea* N *una red Lugar/Transición,*

Una aplicación $M : P \rightarrow \mathbb{N} \cup \{\omega\}$ *se llama* marcado *de* N *sii* $M(p) \leq K(p)$ *para todo* $p \in P$.

Una transición $t \in T$ *está* habilitada *en un marcado* M *sii*

$$\forall p \in {}^{\bullet}t : M(p) \geq W(p,t) \text{ y } \forall p \in t^{\bullet} : M(p) \leq K(p) - W(t,p)$$

Una transición $t \in T$ *habilitada en* M *conduce a un marcado* M', $M[t\rangle M'$, *tal que para cada* $p \in P$

$$M'(p) = \begin{cases} M(p) - W(p,t) & \text{sii } p \in {}^{\bullet}t \setminus t^{\bullet} \\ M(p) + W(t,p) & \text{sii } p \in t^{\bullet} \setminus {}^{\bullet}t \\ M(p) - W(p,t) + W(t,p) & \text{sii } p \in {}^{\bullet}t \cap t^{\bullet} \\ M(p) & \text{en otro caso} \end{cases}$$

1.2. Propiedades

Con una red de Petri pueden estudiarse dos tipos de propiedades: las que dependen del marcado inicial (*propiedades de comportamiento*) y las que son independientes del marcado inicial (*propiedades estructurales*).

1.2.1. Propiedades de Comportamiento

Las principales propiedades de comportamiento son la alcanzabilidad, la acotabilidad, la vivacidad, la reversibilidad, la cobertura y la persistencia.

Definición 20 (Marcado Alcanzable) *Un marcado M_n se dice* alcanzable *desde un marcado M_0 si existe una secuencia de disparos que transforma M_0 en M_n. Una* secuencia de disparo *se denota por $\sigma = t_1 t_2 \cdots t_n$. En este caso, M_n es alcanzable desde M_0 mediante σ y se escribe $M_0[\sigma\rangle M_n$. El conjunto de todos los posibles marcados alcanzables desde M_0 en una red (N, M_0) se denota por $R(N, M_0)$ o simplemente por $R(M_0)$.*

Definición 21 (Problema de Alcanzabilidad) *El* problema de la alcanzabilidad *para las redes de Petri será el problema de encontrar si $M_n \in R(M_0)$ en una red dada (N, M_0).*

Definición 22 (Red de Petri Acotada) *Una red de Petri (N, M_0) se dice* k-acotada *o* acotada *si el número de tokens en cada lugar no es superior a un número finito k para cualquier marcado alcanzable desde M_0, es decir, $M(p) \leq k$ para todo lugar p y todo marcado $M \in R(M_0)$.*

Definición 23 (Red de Petri Segura) *Una red de Petri (N, M_0) se dice* segura *si está 1-acotada.*

Definición 24 (Interbloqueo) *Se dice que en una red de Petri ocurre un* interbloqueo *cuando se alcanza un marcado desde el que no se puede disparar ninguna transición.*

Definición 25 (Red de Petri Viva) *Una red de Petri (N, M_0) se dice que está* viva *(o equivalentemente se dice que M_0 es un* marcado vivo *para N) si, sea cual sea el marcado que se alcance desde M_0, existe una secuencia disparable que permite disparar cualquier transición de la red.*

La vivacidad garantiza, por tanto, la ausencia de interbloqueos. Esto implica que cualquier transición es eventualmente disparable en alguna secuencia de disparo. Esta es una propiedad muy fuerte y, a menudo, muy difícil de verificar. Por ello, se habla se grados de disparo denominados *niveles* [Mur89][Pet81]. Concretamente, cuando una transición t está *viva* en

- *Nivel 0*: Nunca puede ser disparada. Es una transición *muerta*.
- *Nivel 1*: Es potencialmente disparable, lo que significa que existe un marcado $M \in R(M_0)$ en el que la transición está habilitada.
- *Nivel 2*: Para cada entero finito positivo n existe una secuencia de disparo en la que t ocurre al menos n veces.
- *Nivel 3*: Existe una secuencia de disparo de longitud infinita en la que t ocurre infinitamente a menudo.
- *Nivel 4*: Está viva en el nivel 1 para cada marcado del conjunto de alcanzabilidad $R(M_0)$.

Así pues, una red de Petri está viva en el nivel i si cada transición está viva en el nivel i. Una red de Petri está viva si todas las transiciones están vivas. Nótese que una transición que esté viva en el nivel 4, lo está en los niveles 3, 2 y 1.

Definición 26 (Red de Petri Reversible) *Una red de Petri (N, M_0) se dice* reversible *si, para cualquier marcado $M \in R(M_0)$, M_0 es alcanzable desde M. Así pues, el marcado inicial es alcanzable desde todos los marcados alcanzables.*

Definición 27 (Estado Home) *Un marcado M' se dice que es un* estado home *si para todo marcado $M \in R(M_0)$, M' es alcanzable desde M.*

Definición 28 (Cobertura) *Un marcado M' en una red de Petri (N, M_0)* cubre *a un marcado M si $M'(p) \geq M(p)$ para cualquier lugar p de la red.*

Definición 29 (Problema de Cobertura) *El* problema de la cobertura *dada una red de Petri (N, M_0) y un marcado M consistirá en comprobar si existe un marcado alcanzable $M' \in R(M_0)$, tal que $M' \geq M$, (es decir, M' cubre a M).*

Definición 30 (Red de Petri Persistente) *Una red de Petri (N, M_0) se dice* persistente *si, para dos transiciones habilitadas cualesquiera, el disparo de una de ellas no deshabilita la otra.*

1.2.2. Propiedades Estructurales

Las propiedades estructurales son aquéllas que dependen de la estructura topológica de las redes de Petri. Son independientes del marcado inicial en el sentido de que dichas propiedades se cumplen para cualquier marcado inicial. Son propiedades estructurales la acotabilidad estructural, la vivacidad estructural, la controlabilidad, la conservatividad, la repetitividad y la consistencia.

Definición 31 (Red de Petri Acotada Estructuralmente) *Una red de Petri está* acotada estructuralmente *si está acotada para cualquier marcado inicial finito.*

Definición 32 (Lugar No Acotado Estructuralmente) *Un* lugar p *en una red de Petri se dice* no acotado estructuralmente *si existe un marcado* M_0 *y una secuencia de disparo* σ *desde* M_0 *tal que* p *no está acotado.*

Definición 33 (Red de Petri Estructuralmente Viva) *Una red de Petri está* estructuralmente viva *si existe algún marcado inicial para el que está viva.*

Definición 34 (Red de Petri Completamente Controlable) *Una red de Petri se dice* completamente controlable *si cualquier marcado es alcanzable desde cualquier otro marcado.*

Definición 35 (Red de Petri Estructuralmente Conservativa) *Una red de Petri es* estructuralmente conservativa *si, para cualquier marcado inicial* M_0 *y un marcado alcanzable* $M \in R(M_0)$*, existe un vector* x $(n \times 1)$ *tal que* $x_i \neq 0$*, para cualquier* $i = 1, \ldots, n$*, y* $x^T M = x^T M_0$.

Definición 36 (Red de Petri (parcialmente) Repetitiva) *Una red de Petri es* (parcialmente) repetitiva *si existe un marcado finito* M_0 *y una secuencia de disparo* σ *tal que (alguna) toda transición ocurre un número infinito de veces en* σ.

Definición 37 (Red de Petri (parcialmente) Consistente) *Una red de Petri es* (parcialmente) consistente *si existe un marcado finito* M_0 *y una secuencia de disparo* σ *cíclica (desde* M_0 *a* M_0*) tal que (alguna) toda transición ocurre al menos una vez en* σ.

1.3. Análisis de redes de Petri

Los métodos para analizar redes de Petri pueden clasificarse en tres grupos:

1. el árbol de cobertura (o alcanzabilidad),
2. las ecuaciones matriciales y
3. las técnicas de reducción y síntesis.

El primer método implica la enumeración de todos los marcados alcanzables de la red [Mur89][Fin90]. Es aplicable a toda clase de redes pero se limita a redes pequeñas debido a la explosión del espacio de estados. Por otra parte, las ecuaciones matriciales y las técnicas de reducción son más potentes pero en muchos casos sólo son aplicables a subclases especiales de redes de Petri o en situaciones especiales.

1.3.1. Árbol de Cobertura (o Alcanzabilidad)

Para enumerar todos los marcados alcanzables de una red se construye el *árbol de alcanzabilidad*. La construcción del árbol de alcanzabilidad consiste en tomar como raíz del árbol el marcado inicial de la red y disparar todas las transiciones habilitadas. Esto lleva a nuevos marcados que habilitan otras transiciones. Tomando cada uno de esos nuevos marcados como una "nueva raíz", se pueden generar recursivamente todos los marcados alcanzables. Es muy posible que el árbol crezca indefinidamente. Para reducir la talla del árbol y obtener un árbol finito, se utiliza un símbolo especial ω que representa un número de tokens arbitrariamente grande y que está sujeto a cuatro reglas aritméticas especiales, tales que para cualquier entero a:

$$a < \omega \qquad \omega \leq \omega \qquad \omega + a = \omega \qquad \omega - a = \omega$$

Utilizando esta notación y un algoritmo basado en el algoritmo presentado en [KM69] se obtiene el *árbol de cobertura*. Si el símbolo ω no aparece, el árbol se llama *árbol de alcanzabilidad*.

El árbol de cobertura para una red de Petri (N, M_0) se construye mediante el Algoritmo 1.

Algoritmo 1 Árbol de Cobertura de una Red de Petri

- Etiquetar el marcado "raíz" M_0 como "nuevo".
- Mientras existan marcados "nuevos" hacer:
  - Seleccionar un marcado "nuevo" M;
  - Si no hay transiciones habilitadas en M, entonces etiquetar M "muerto";
  - Si M es idéntico a algún marcado en el camino desde la raíz a M, etiquetar M como "viejo" e ir a otro marcado nuevo.
  - Para todas las transiciones habilitadas en M hacer:
    - Obtener el marcado M' resultado del disparo de una transición;
    - Si desde la raíz a M existe un M'' tal que $M'(p_i) \geq M''(p_i)$ para todo $i = 1, \ldots, n$ y $M' \neq M''$ entonces
      - Reemplazar $M'(p_i)$ por ω siempre que $M'(p_i) > M''(p_i)$;
      - Añadir M' al árbol etiquetando el arco con la transición apropiada;
      - Etiquetar M' como "nuevo";

Este método es dependiente del marcado inicial, por tanto, es útil para determinar propiedades de comportamiento. Algunas de las propiedades que pueden estudiarse con el árbol de cobertura T para una red de Petri (N, M_0) son las siguientes:

1. Una red (N, M_0) está *acotada* y, por tanto, $R(M_0)$ es finito si y solamente si ω no aparece en ningún nodo de T.
2. Una red (N, M_0) es *segura* si y solamente si en los nodos de T sólo aparecen 0's y 1's.
3. Una transición t está *muerta* si y solamente si no aparece como etiqueta de un arco en T.
4. Si M es alcanzable desde M_0 entonces existe un nodo etiquetado con M' tal que $M \leq M'$ (M' *cubre* a M).
5. Una red (N, M_0) es *conservativa* si y solamente si la suma ponderada de los tokens en cada nodo del árbol de alcanzabilidad es constante.

Así pues, el árbol de cobertura puede utilizarse para resolver los problemas de seguridad, acotamiento, conservatibilidad y cobertura. Desafortunadamente, en general, no puede utilizarse para resolver los problemas de alcanzabilidad o vivacidad o para determinar qué secuencias de disparo son posibles. Estos problemas están limitados por la existencia del símbolo ω que implica una pérdida de información. Para resolver dichos problemas, son necesarias otras aproximaciones.

Ejemplo 38 *La Figura 1.3 muestra el árbol de cobertura de la red de la Figura 1.1. En este árbol de cobertura, el símbolo ω no aparece en ningún nodo, podemos hablar, por tanto, de árbol de alcanzabilidad. El estudio del árbol nos permite concluir que la red está acotada y, por tanto, $R(M_0)$ es finito. Además, no hay transiciones muertas puesto que todas las transiciones de la red aparecen etiquetando algún arco del árbol.*

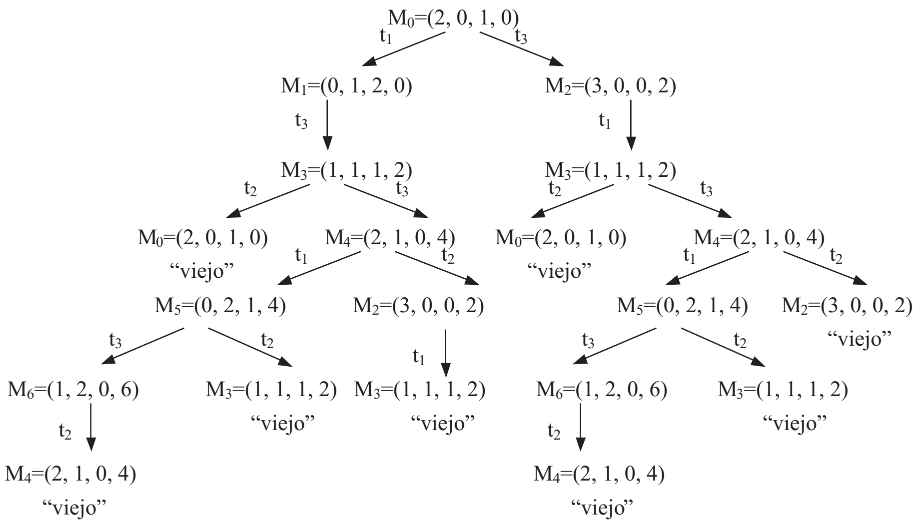


Figura 1.3: Árbol de cobertura (alcanzabilidad) de la red de Petri de la Figura 1.1

Ejemplo 39 *La Figura 1.5 muestra el árbol de cobertura de la Figura 1.4. El símbolo ω en dicho árbol indica que la red no está acotada, así como también qué lugares de la red no están acotados; en este caso, los lugares no acotados son p_2 y p_4. El hecho de que en el árbol aparezca un nodo etiquetado como "muerto" indica que la red no está viva.*

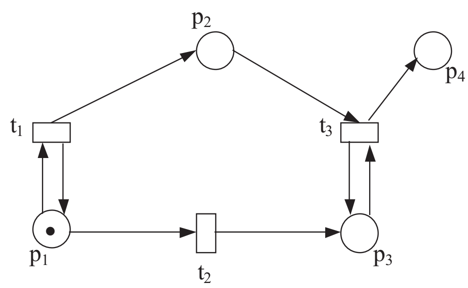


Figura 1.4: Red de Petri (b)

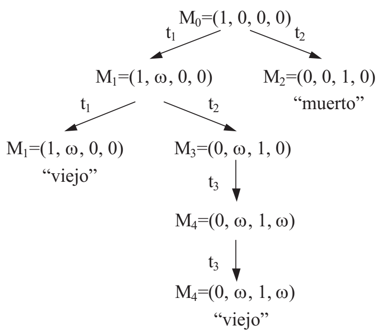


Figura 1.5: Árbol de cobertura de la red de Petri de la Figura 1.4

1.3.2. Ecuaciones Matriciales

En la segunda técnica de análisis se utiliza la *matriz de incidencia* [Mur89][Pet81] (matriz que describe cómo los nodos en una red de Petri están interconectados). Esta interconexión de los nodos no está relacionada con el marcado inicial, por ello, esta aproximación se utiliza para determinar las propiedades estructurales de las redes de Petri, aunque también puede utilizarse para determinar alguna propiedad de comportamiento.

Definición 40 (Matriz de Incidencia) *La* matriz de incidencia $A = [a_{ij}]$ *de una red de Petri con* n *transiciones y* m *lugares es una matriz* $(n \times m)$ *de enteros tal que*

$$a_{ij} = a_{ij}^{+} - a_{ij}^{-}$$

siendo $a_{ij}^{+} = F(i,j)$ *el peso del arco desde la transición* i *a su lugar de salida* j *y* $a_{ij}^{-} = F(j,i)$ *el peso del arco a la transición* i *desde su lugar de entrada* j*.*

Es fácil comprobar que a_{ij}^-, a_{ij}^+ y a_{ij} son, respectivamente, los tokens eliminados, añadidos y cambiados en el lugar j cuando la transición i se dispara una vez. La transición i estará habilitada en el marcado M si y solamente si

$$a_{ij}^- \leq M(j) \quad j = 1, 2, \ldots, m.$$

Definición 41 (Ecuación de Estados) *Al escribir ecuaciones de matrices, escribimos un marcado M_k como un vector columna ($m\times 1$). La j-ésima entrada de M_k denota el número de tokens en el lugar j inmediatamente después del k-ésimo disparo en alguna secuencia de disparo. El k-ésimo disparo o* vector de control *u_k es un vector columna ($n \times 1$) de ($n-1$) 0's y una entrada distinta de 0, un 1 en la i-ésima posición indica que la transición i se dispara en el k-ésimo disparo. Ya que la i-ésima fila de la matriz de incidencia A indica el cambio de marcado como resultado del disparo de la transición i, podemos escribir la siguiente* ecuación de estado *para una red de Petri:*

$$M_k = M_{k-1} + A^T u_k \quad k = 1, 2, \cdots .$$

Ejemplo 42 *La matriz de incidencia A de la red de Petri de la Figura 1.1 es la siguiente:*

$$A = \begin{bmatrix} -2 & 1 & 1 & 0 \\ 1 & -1 & 0 & -2 \\ 1 & 0 & -1 & 2 \end{bmatrix}$$

La ecuación de estados que se muestra a continuación representa el disparo de la transición t_3 en el marcado $M_0 = (2, 0, 1, 0)$ dando como resultado el marcado $M_1 = (3, 0, 0, 2)$.

$$\begin{bmatrix} 3 \\ 0 \\ 0 \\ 2 \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \\ 1 \\ 0 \end{bmatrix} + \begin{bmatrix} -2 & 1 & 1 \\ 1 & -1 & 0 \\ 1 & 0 & -1 \\ 0 & -2 & 2 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

Definición 43 (Condición Necesaria de Alcanzabilidad) *Supongamos que un marcado destino M_d es alcanzable desde M_0 a través de una secuencia de disparo $\{u_1, u_2, \ldots, u_d\}$. Escribiendo las ecuaciones de estado para $i = 1, 2, \ldots, d$ y sumándolas, obtenemos*

$$M_d = M_0 + A^T \sum_{k=1}^{d} u_k$$

que puede reescribirse como $A^T x = \Delta M$ *siendo* $\Delta M = M_d - M_0$ *y* $x = \sum_{k=1}^{d} u_k$. x *es un vector columna de enteros no negativos que se denomina* vector contador de disparos o vector de Parikh. *La i-ésima entrada de* x *denota el número de veces que una transición i debe dispararse para transformar* M_0 *en* M_d.

Si un marcado M_d *es alcanzable desde otro marcado* M_0*, entonces* $M_d = M_0 + A^T x$ *da como resultado una solución entera no negativa. Por otra parte, lo contrario no es necesariamente cierto; si* $M_d = M_0 + A^T x$ *da como resultado una solución entera no negativa,* M_d *puede o no ser alcanzable desde* M_0*. Por lo tanto, es una condición necesaria pero no suficiente para la alcanzabilidad.*

En términos matemáticos, un conjunto de ecuaciones lineales $A^T x = \Delta M$ tiene una solución x si y sólo si ΔM es ortogonal a cada solución y de su sistema homogéneo, $Ay = 0$.

Si r es el rango de A y dividimos A de la siguiente forma:

$$A = \begin{array}{c} \overset{m-r}{\longleftrightarrow} \quad \overset{r}{\longleftrightarrow} \\ \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \end{array} \begin{array}{l} \\ \updownarrow r \\ \updownarrow n-r \end{array} \tag{1.1}$$

donde A_{12} es una matriz cuadrada no singular de orden r. Puede encontrarse un conjunto de soluciones linealmente independientes para $Ay = 0$ como las $(m-r)$ columnas de la siguiente matriz $(m-r) \times m$:

$$B_f = \left[I_\mu : -A_{11}^T (A_{12}^T)^{-1} \right] \tag{1.2}$$

donde I_μ es la matriz identidad de orden $\mu = m - r$. Nótese que $AB_f^T = 0$. Ahora, la condición de que ΔM es ortogonal a cada solución para $Ay = 0$ es equivalente a la siguiente condición:

$$B_f \Delta M = 0 \tag{1.3}$$

Así que si M_d es alcanzable desde M_0, entonces el correspondiente vector contador de disparos x debe existir y debe cumplirse la condición 1.3.

Teorema 44 *Si* M_d *es alcanzable desde* M_0 *en una red de Petri* (N, M_0) *entonces* $B_f \Delta M = 0$ *siendo* $\Delta M = M_d - M_0$ *y* $B_f = \left[I_\mu : -A_{11}^T (A_{12}^T)^{-1} \right]$.

A partir del Teorema anterior se deduce la siguiente condición suficiente de no alcanzabilidad.

Corolario 45 *En una red de Petri* (N, M_0)*, un marcado* M_d *no es alcanzable desde* M_0 $(\neq M_d)$ *si su diferencia es una combinación lineal de los vectores columna de* B_f*, es decir,* $\Delta M = B_f^T z$*, donde* z *es un vector columna* $(\mu \times 1)$ *distinto de* 0.

Ejemplo 46 *El rango de la matriz de incidencia del Ejemplo 42 es 2 y puede dividirse según (1.1) en*

$$A_{11} = \begin{bmatrix} -2 & 1 \\ 1 & -1 \end{bmatrix} \text{ y } A_{12} = \begin{bmatrix} 1 & 0 \\ 0 & -2 \end{bmatrix}.$$

Según (1.2), la matriz B_f *será*

$$B_f = \begin{bmatrix} 1 & 0 & 2 & 1/2 \\ 0 & 1 & -1 & -1/2 \end{bmatrix}$$

Es sencillo comprobar que $B_f \Delta M = 0$ *se cumple para* $\Delta M = M_1 - M_0 = (\ 1\ \ 0\ \ -1\ \ 2\ )^T$.

Las propiedades estructurales pueden definirse en términos de la matriz de incidencia y sus ecuaciones o inecuaciones asociadas.

Teorema 47 *Si una red de Petri* N *con* m *lugares es completamente controlable entonces rango*$(A) = m$.

Teorema 48 *Una red de Petri está estructuralmente acotada si y sólo si existe un vector* y $(m \times 1)$ *de enteros positivos tal que* $Ay \leq 0$.

Corolario 49 *Un lugar* p *de una red de Petri* N *es no acotado estructuralmente si y sólo si existe un vector* x $(n \times 1)$ *de enteros no negativos tal que* $A^T x = \Delta M \gneqq 0$*, donde la* p*-ésima entrada de* $\Delta M > 0$ *(es decir,* $\Delta M(p) > 0$*).*

Teorema 50 *Una red de Petri* N *es (parcialmente) conservativa si y sólo si existe un vector* y $(m \times 1)$ *de enteros (no negativos) positivos tal que* $Ay = 0$*,* $y \neq 0$.

Teorema 51 *Una red de Petri N es (parcialmente) repetitiva si y sólo si existe un vector x $(n \times 1)$ de enteros (no negativos) positivos tal que $A^T x \geq 0$, $x \neq 0$.*

Teorema 52 *Una red de Petri N es (parcialmente) consistente si y sólo si existe un vector x $(n \times 1)$ de enteros (no negativos) positivos tal que $A^T x = 0$, $x \neq 0$.*

En [Mur89], Murata demuestra los teoremas anteriores y presenta la Tabla 1.1, que resume las condiciones necesarias y suficientes para algunas propiedades estructurales.

Propiedades	Condiciones Necesarias y Suficientes
Conservativa	$\exists y > 0,\ Ay = 0$
Parcialmente Conservativa	$\exists y \gneqq 0,\ Ay = 0$
Repetitiva	$\exists x > 0,\ A^T x \geq 0$
Parcialmente Repetitiva	$\exists x \gneqq 0,\ A^T x \geq 0$
Consistente	$\exists x > 0,\ A^T x = 0$
Parcialmente Consistente	$\exists x \gneqq 0,\ A^T x = 0$

$x \gneqq y$ significa $x \geq y$ y $x_i \neq y_i$ para algún i.

Tabla 1.1: Propiedades Estructurales. Condiciones necesarias y suficientes

Un invariante de un sistema es una afirmación que se cumple en cualquier estado alcanzable. En las redes es posible encontrar ciertos vectores, directamente a partir de la estructura de la red, que inducen invariantes.

Definición 53 (P-invariante) *Un* P-invariante *es un vector x $(n \times 1)$ de enteros no negativos que satisface la ecuación $x^T A = 0$.*

Combinando esta ecuación con $M_d = M_0 + A^T x$, tenemos que $x^T M = x^T M_0$, lo que implica que el número total de tokens iniciales en M_0, ponderado por el P-invariante, es constante (ver Definición 35).

Definición 54 (T-invariante) *Un* T-invariante *es un vector y $(m \times 1)$ de enteros no negativos que satisface la ecuación $Ay = 0$.*

Combinando esta ecuación con $M_d = M_0 + A^T x$, tenemos que $M_d = M_0$ con $y = x$, lo que implica que si el vector contador de disparos es idéntico a un T-invariante entonces el marcado final es igual al marcado inicial.

Teorema 55 *Un vector x de enteros no negativos es un P-invariante si y sólo si $x^T M = x^T M_0$ para algún marcado inicial M_0 y algún marcado $M \in R(M_0)$.*

Teorema 56 *Un vector y de enteros no negativos es un T-invariante si y sólo si existe un marcado M y una secuencia de disparo desde M hasta M cuyo vector contador de disparos es y.*

Definición 57 (Soporte de un Invariante) *El* soporte de un invariante *es el conjunto de lugares (transiciones) correspondientes a entradas distintas de cero en un P-invariante (T-invariante). Se denota por $< x >$ ($< y >$).*

Definición 58 (Soporte Minimal de un Invariante) *Un soporte de un invariante es* minimal *si y sólo si no contiene el soporte de otro invariante que no sea él mismo y el conjunto vacío.*

Definición 59 (Invariante Minimal) *Un* invariante minimal *es un invariante que no es combinación lineal de otros invariantes.*

Teorema 60 *Existen $(n-r)$ P-invariantes minimales y $(m-r)$ T-invariantes minimales siendo $r = rango(A)$.*

Para una mejor comprensión de las propiedades de las redes de Petri, se consideran dos tipos especiales de lugares: sifones y trampas.

Definición 61 (Sifón) *Un subconjunto no vacío de lugares S en una red ordinaria N es un* sifón *si ${}^\bullet S \subseteq S^\bullet$, es decir, toda transición que tenga un lugar de salida en S tiene un lugar de entrada en S.*

Un *sifón* tiene una propiedad de comportamiento tal que si en algún marcado no tiene tokens, permanecerá sin tokens en cualquier marcado sucesor.

Definición 62 (Trampa) *Un subconjunto no vacío de lugares Q en una red ordinaria N es una* trampa *si $Q^\bullet \subseteq {}^\bullet Q$, es decir, toda transición que tenga un lugar de entrada en Q tiene un lugar de salida en Q.*

Una *trampa* tiene una propiedad de comportamiento tal que si está marcada (tiene al menos 1 token), permanecerá marcada en cualquier marcado sucesor.

Teorema 63 *Si x es un P-invariante, entonces $< x >$ es una trampa y un sifón.*

Teorema 64 *Sean x e y dos invariantes del mismo tipo. Entonces, se cumple lo siguiente:*

1. *$ax + by$ es un invariante para a y b enteros no negativos.*
2. *Si $(x - y)$ no tiene elementos negativos, entonces es un invariante.*
3. *$< x + y >=< x > \cup < y >$.*

Teorema 65 *Sea el lugar p_i perteneciente al soporte de un P-invariante en una red de Petri pura con marcado inicial acotado m_0. Sean $I_1, \ldots, I_j$ los soportes minimales que contienen a p_i y $x_1, \ldots, x_j$ los P-invariantes minimales asociados a los soportes minimales. Entonces, para todo marcado alcanzable $m \in R(m_0)$, $m(p_i)$ está acotado superiormente por:*

$$m(p_i) \leq \min_{l=1,\ldots,j} [(m_0^T x_l)/(x_l(p_i))].$$

Este hecho es muy importante puesto que puede utilizarse para determinar el k-acotamiento de una red a partir de los P-invariantes. Nótese que si una red está cubierta por P-invariantes y m_0 está acotado, entonces la red está acotada.

Teorema 66 *Una red de Petri pura está viva y acotada si está cubierta por P-invariantes, todos los P-invariantes están marcados con tokens y ninguno de los sifones se vacía nunca de tokens.*

El siguiente es un procedimiento sistemático sencillo para aplicar el Teorema anterior:

1. Determinar todos los P-invariantes minimales y comprobar que todo lugar pertenece a algún P-invariante.
2. Utilizando el marcado inicial m, verificar que todos los P-invariantes están marcados.
3. Encontrar todos los sifones (minimales). Esto requiere una búsqueda exhaustiva.

4. Para todo sifón hacer: utilizando las ecuaciones de los P-invariantes, asignar cero tokens a todos los lugares del sifón. Si, para al menos un sifón, las ecuaciones resultantes son consistentes, entonces la red de Petri no está viva.

1.3.3. Técnicas de reducción y síntesis

Para facilitar el análisis de grandes sistemas, el sistema modelado se reduce a uno más simple que preserve las propiedades del sistema a analizar. A la inversa, las técnicas para transformar un modelo abstracto en un modelo más refinado de una forma jerárquica pueden utilizarse para síntesis. Existen muchas técnicas de transformación para las redes de Petri. Aquí presentamos las más sencillas que pueden utilizarse para analizar vivacidad, seguridad y acotamiento.

No es difícil comprobar que las siguientes seis operaciones preservan las propiedades de vivacidad, seguridad y acotamiento. Es decir, sean (N, M_0) y (N', M'_0) redes de Petri antes y después de una de las siguientes transformaciones. Entonces (N', M'_0) está viva, segura o acotada si y sólo si (N, M_0) está viva, segura o acotada, respectivamente.

1. *Fusión de lugares en serie* (ver Figura 1.6(a)).
2. *Fusión de transiciones en serie* (ver Figura 1.6(b)).
3. *Fusión de lugares en paralelo* (ver Figura 1.6(c)).
4. *Fusión de transiciones en paralelo* (ver Figura 1.6(d)).
5. *Eliminación de lugares en ciclo* (ver Figura 1.6(e)).
6. *Eliminación de transiciones en ciclo* (ver Figura 1.6(f)).

1.4. Decidibilidad y Complejidad

Un *problema de decisión* [Sud88] es un conjunto de cuestiones cuya respuesta es sí o no. Una *solución* para un problema de decisión es un procedimiento efectivo (completo, mecánico y determinista) que determina la respuesta a cada cuestión del conjunto. Un problema de decisión es *indecidible* si no existe un algoritmo que lo resuelva. Existen numerosos estudios sobre la decidibilidad y la

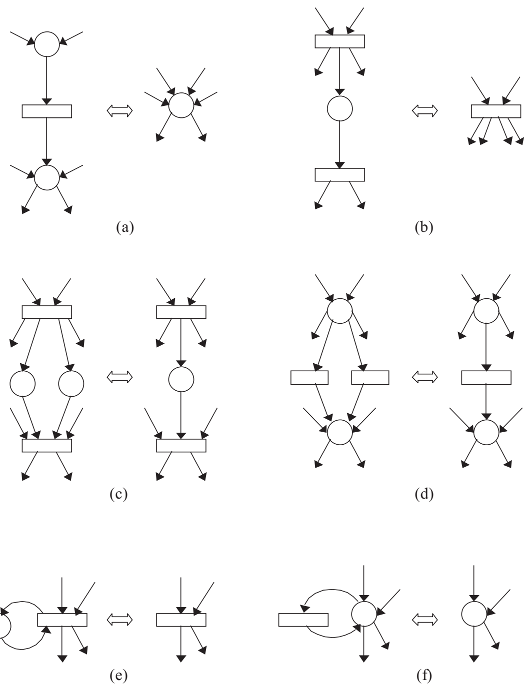


Figura 1.6: Seis transformaciones que preservan vivacidad, seguridad y acotamiento [Mur89]

complejidad del análisis de las propiedades de las redes de Petri [Pet81]. Se sabe que los problemas de acotabilidad, alcanzabilidad, interbloqueo y vivacidad son todos decidibles para las redes de Petri [KM69][Hac76a][AK77][May84][EN94]. Sin embargo, para algunas extensiones de las redes de Petri puede darse la posibilidad de que un problema no tenga solución, es decir, sea indecidible [DFS98]. También puede ocurrir que aunque la solución exista, su coste (tiempo y espacio de memoria necesarios) sea elevado [DF97]. La habilidad de las máquinas de Turing para devolver respuestas afirmativas o negativas las convierte en un formalismo matemático apropiado para construir soluciones a problemas de decisión. La *reducibilidad* [HU79][Pet81] entre problemas se usa habitualmente para demostrar que un problema es decidible o indecidible. Para demostrar decidibilidad en las propiedades de una red de Petri, es necesario reducir un problema de red de Petri a un problema con solución conocida. Para demostrar indecidibilidad, hay que reducir un problema que se sabe es indecidible a un problema de red de Petri.

Son muy pocos los estudios que existen sobre decidibilidad para extensiones de redes de Petri, comparados con la cantidad de artículos de investigación existentes sobre subclases de redes de Petri. En su tesis [Duf98] y también en [DFS98] y [DJS99], Catherine Dufourd estudia y compara bastantes extensiones de las redes de Petri en términos de decidibilidad. Para ello, define las *G-redes* que constituyen una extensión de las redes de Petri y que engloban a las *redes Automodificantes* de Valk [Val78a, Val78b]. Dufourd realiza el estudio de las siguientes propiedades: terminación, cobertura, acotamiento de una red, acotamiento de un lugar, acotamiento estructural, alcanzabilidad, interbloqueo, vivacidad y vivacidad de una transición, sobre nueve subclases de G-redes. Para cada clase determina qué propiedades son decidibles, lo que le permite agruparlas (o separarlas), obteniendo una jerarquía muy fina de decidibilidad.

Presentamos la clasificación que Dufourd muestra en [DFS98] y [Duf98]. Su motivación es encontrar extensiones de las redes de Petri que se puedan analizar, centrando su estudio sobre distintos tipos de arcos extendidos. En primer lugar, damos las definiciones de dichos arcos.

Definición 67 (Pre-arco) *Un arco desde el lugar p a la transición t (un arco de entrada a t) es un* pre-arco*.*

Definición 68 (Post-arco) *Un arco desde la transición t al lugar p (un arco de entrada a p) es un* post-arco*.*

Definición 69 (Arco clásico o del tipo red de Petri) *Un* arco *se denomina* clásico o del tipo red de Petri *si su etiqueta es un número natural: $F(x, y) \in \mathbb{N}$, $x, y \in P \cup T$.*

Definición 70 (Arco extendido) *Un* arco *se dice* extendido *si su etiqueta es un polinomio.*

Definición 71 (Arco Inhibidor) *Un* arco inhibidor *(o test de cero) desde un lugar p a una transición t indica que la transición sólo puede dispararse si no hay tokens en el lugar, es decir, una transición t a la que le llega un arco inhibidor desde un lugar p está habilitada si lo está en el sentido habitual y además hay cero tokens en el lugar p.*

$$\text{Si } F(p,t) = 2p \text{ entonces } t \text{ está habilitada sólo si } M(p) \geq 2M(p),$$
$$\text{es decir, si y sólo si } M(p) = 0.$$

La transición se dispara eliminando los tokens de todos sus lugares (normales) de entrada.

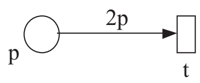


Figura 1.7: Test de cero

Un arco inhibidor se representa mediante un arco no orientado que acaba con un pequeño círculo.

Figura 1.8: Arco inhibidor

Definición 72 (Arco Reset) *Un* arco reset *(o puesta a cero) es un pre-arco tal que $F(p,t) = p$ (con $F(t,p) = 0$). El disparo de una transición t con un arco reset desde un lugar p vacía dicho lugar.*

$$\textit{Si } F(p,t) = p,\ F(t,p) = 0 \textit{ y } M[t\rangle M' \textit{ entonces } M'(p) = 0.$$

La puesta a cero de un lugar consiste en eliminar su propio contenido, es decir, en vaciar el lugar.

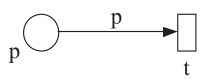


Figura 1.9: Puesta a cero de un lugar p

Un arco reset se representa mediante un arco no orientado provisto de un aspa.

Los arcos reset son pre-arcos "débiles" por tres razones: a) la puesta a cero siempre se puede realizar y nunca es un obstáculo para el disparo de una

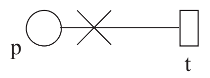


Figura 1.10: Arco reset

transición; b) es una acción local que concierne únicamente a un lugar, sin tener en cuenta ningún otro; y c) puede implementarse utilizando el polinomio identidad.

La acción de un arco reset es diferente de la de un arco inhibidor, aunque en ambos casos el contenido del lugar después del disparo de la transición es cero. El disparo de un arco reset siempre es posible. Por el contrario, el test de cero supone una restricción muy fuerte. El arco reset puede simularse con el test de cero pero lo contrario no es posible.

Definición 73 (Arco de Transferencia) *Un* arco de transferencia *traslada todos los tokens de un lugar p a un lugar q cuando la transición t se dispara, vaciando p.*

$$\begin{gathered} \textit{Si } F(p,t) = F(t,q) = p,\ F(t,p) = F(q,t) = 0 \textit{ y } M[t\rangle M' \\ \textit{entonces } M'(p) = 0 \textit{ y } M'(q) = M(q) + M(p). \end{gathered}$$

La transferencia traslada de golpe el contenido de un lugar a otro lugar. Esta acción se representa como la unión de un pre-arco y un post-arco. Para transferir el contenido de p a q, la transición t vacía p mediante un arco $F(p,t) = p$ y añade p tokens a q mediante un arco $F(t,q) = p$, si $F(t,p) = F(q,t) = 0$.

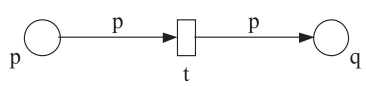


Figura 1.11: Transferencia del contenido del lugar p al lugar q

La puesta a cero de un lugar y su añadido a otro lugar se considera como un todo, como una sola acción que se representa gráficamente mediante un arco de transferencia. Un arco de transferencia se representa mediante dos flechas dobles.

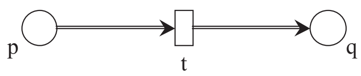


Figura 1.12: Arco de transferencia

Definición 74 (Arco Doble) *Un* arco doble *es un post-arco tal que* $F(t,p) = p$ *(con* $F(p,t) = 0$*).*

$$\textit{Si } F(t,p) = p,\ F(p,t) = 0 \textit{ y } M[t\rangle M' \textit{ entonces } M'(p) = 2M(p).$$

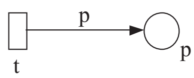


Figura 1.13: Arco doble

Las G-redes constituyen una extensión de las redes de Petri. Más exactamente, extienden a las redes Automodificantes de Valk [Val78a, Val78b] que, ellas mismas, son una extensión de las redes de Petri. Las redes Automodificantes y, por tanto, las G-redes, son lo suficientemente potentes para simular máquinas enumeradoras, lo que hace indecidibles a todas las propiedades no triviales. Sin embargo, muchas subclases de G-redes, más potentes que las redes de Petri, son (en parte) analizables. Dufourd [Duf98] define una jerarquía de 11 subclases de G-redes (incluyendo a las redes de Petri y a las redes Automodificantes) de las que 9 son más potentes que las redes de Petri. Damos, a continuación, la definición de G-red y sus subclases.

Definición 75 (Red Automodificante Generalizada (G-red)) *Una* red Automodificante Generalizada (G-red) *es una 4-tupla* $N = (P, T, F, m_0)$ *en la que*

- $P = \{p_1, \ldots, p_{|P|}\}$ *es un conjunto finito de lugares,*
- T *es un conjunto finito de transiciones* $(P \cap T = \emptyset)$*,*

- $F : (P \times T) \cup (T \times P) \to \mathbb{N}[P]$ *es una relación de flujo tal que*

$$\forall x, y \in P \cup T,\ F(x,y) \text{ es de la forma } \textstyle\sum_{j \in J} \lambda_j p_{i_j}^{n_j}$$

  siendo J un conjunto finito, $\lambda_j \in \mathbb{N}$, $n_j \in \mathbb{N}$ y $1 \leq i_j \leq |P|$, y

- $m_0 \in \mathbb{N}^{|P|}$ *es el marcado inicial.*

Para las *redes Automodificantes de Valk* [Val78a, Val78b], los polinomios son de grado 1, es decir, combinaciones lineales.

Definición 76 (Red Automodificante de Valk (AM-red)) *Una* red Automodificante *es una G-red $N = (P, T, F, m_0)$ tal que:*

$$F : (P \times T) \cup (T \times P) \to c + \sum_{i=1}^{|P|} \lambda_i p_i \text{ con } c,\ \lambda_i \in \mathbb{N} \text{ y } p_i \in P$$

Para las redes de Petri [Pet62], los polinomios son de grado 0, es decir, constantes.

Definición 77 (Red de Petri) *Una* red de Petri *es una G-red $N = (P, T, F, m_0)$ tal que:*

$$F : (P \times T) \cup (T \times P) \to \mathbb{N}$$

Definición 78 (Red G-Post) *Una* red G-Post *es una G-red en la que sólo los post-arcos son arcos extendidos o, lo que es lo mismo, los pre-arcos son arcos clásicos $\forall p \in P$, $\forall t \in T$, $F(p,t) \in \mathbb{N}$.*

Definición 79 (Red Post-AM) *Una* red Post-AM *es una AM-red tal que los pre-arcos son arcos clásicos: $\forall p \in P$, $\forall t \in T$, $F(p,t) \in \mathbb{N}$ y sólo los post-arcos son arcos extendidos.*

Definición 80 (Red G-Post+Reset) *Una* red G-Post+Reset *es una G-red en la que todos los pre-arcos son arcos reset o arcos clásicos: $\forall p \in P$, $\forall t \in T$, $F(p,t) = p$ o $F(p,t) \in \mathbb{N}$.*

Definición 81 (Red de Petri Reset) *Una* red de Petri Reset *es una red G-Post+Reset tal que todos los post-arcos son arcos clásicos:* $F(t, p) \in \mathbb{N}$*.*

Definición 82 (Red G-Post+Transfer) *Una* red G-Post+Transfer *es una red G-Post+Reset tal que para todo pre-arco reset* $F(p, t) = p$ *existe un post-arco* $F(t, p') = p$*.*

Definición 83 (Red de Petri Transfer) *Una* red de Petri Transfer *es una red G-Post+Transfer tal que todos los arcos son arcos clásicos o arcos de transferencia.*

Definición 84 (Red de Petri Doble) *Una* red de Petri Doble *es una red G-Post tal que todos los post-arcos son arcos dobles o arcos clásicos:* $\forall p \in P$*,* $t \in T$*,* $F(t, p) = p$ *o* $F(t, p) \in \mathbb{N}$*.*

Definición 85 (Red de Petri Normalizada) *Una* red de Petri normalizada *es una red de Petri cuyos arcos están etiquetados con* 0 *o* 1 *y cuyo marcado inicial es un vector* $\{0, 1\}^{|P|}$ *donde* P *es el conjunto de lugares de la red.*

A continuación, se revisan los principales resultados de complejidad y decidibilidad de las redes de Petri (véanse [Esp98] y [EN94] para un resumen más completo).

En 1978, Rackoff [Rac78] demuestra que si una red no está acotada entonces existe una secuencia de disparos de transiciones cuya longitud está acotada por una exponencial doble en función de la talla de la red. De ahí, Rackoff deduce un resultado de complejidad espacial para decidir el acotamiento de una red N (válido también para la cobertura) de $2^{c.|N| \log |N|}$ siendo $|N|$ la talla de la red y c una constante. La decisión del acotamiento estructural de una red se realiza, sin embargo, en tiempo polinomial.

Numerosos resultados de decidibilidad y, en particular, el de acotamiento, se deben a Karp y Miller [KM69]. Son ellos quienes introducen el concepto de árbol de cobertura (véase también Finkel [Fin93]), cuyo principio, como ya hemos visto, consiste en probar sistemáticamente todos los disparos de transiciones posibles a partir del marcado inicial. Se construye una aproximación que "cubre" al conjunto de estados accesibles que permite decidir muchos problemas como la cobertura o el acotamiento de un lugar. Lipton da una cota inferior de complejidad espacial para la cobertura y el acotamiento de $2^{\sqrt{|N|}}$ y Bouziane [Bou96] refina la cota para el acotamiento a $2^{|N|}$.

El problema de la alcanzabilidad en las redes de Petri es muy célebre por su dificultad de análisis. Este problema es EXPSPACE-duro [CLM76][Lip76]; Kosaraju [Kos82] y Mayr [May84] demuestran que es decidible, aunque los algoritmos propuestos son recursivos no primitivos. El interbloqueo y la vivacidad son recursivamente equivalentes a la alcanzabilidad [Hac76a] y, por tanto, son también decidibles.

Durante años, la cuestión de la complejidad del problema de alcanzabilidad estuvo abierta. En el 98, Zakariae Bouziane [Bou98] cerró esta cuestión demostrando que el problema de la accesibilidad es recursivo primitivo. Además, establece que la cota superior espacial es exponencial. Sin embargo, Petr Jančar demuestra en [Jan00] que el algoritmo presentado por Bouziane no es correcto y, por tanto, la complejidad del problema de alcanzabilidad para las redes de Petri sigue sin estar claro.

Para facilitar la modelización de sistemas y para enriquecer las descripciones, se proponen extensiones de las redes de Petri, entre otras, extensiones relativas a funciones que etiquetan los arcos. [FMP96] es un estudio general de este tipo de extensiones que considera funciones verificando "buenas propiedades" (como la monotonía).

Las redes de Petri con dos arcos inhibidores pueden simular a las máquinas de Turing [Hac76b], con lo que todas las propiedades no triviales son indecidibles. Por el contrario, para las redes de Petri con un solo arco inhibidor, la alcanzabilidad es decidible [Rei95].

En 1978, Valk [Val78a, Val78b, Val81] propuso dos extensiones de las redes de Petri. La extensión más débil, las redes Post-Automodificantes, permiten añadir a los lugares, no sólo constantes, sino también combinaciones lineales del marcado actual de la red. Para este modelo, la cobertura y el acotamiento de una red (y de un lugar) siguen siendo decidibles, pero la alcanzabilidad pasa a ser indecidible. La segunda extensión, las redes Automodificantes Fuertes [Val78b], es más potente. Permite a una transición sustraer, en lugar de añadir a los lugares, combinaciones lineales del marcado actual de la red. Se obtiene, de esta forma, un modelo para el que todas las propiedades no triviales son indecidibles, puesto que las redes Automodificantes Fuertes permiten implementar arcos inhibidores. Las redes Automodificantes se presentan más ampliamente en el Capítulo 2.

Las redes Reset son un caso particular de redes Automodificantes. En 1977, Araki y Kasami [AK77] demuestran que para la clase de redes de Petri con arcos reset, la alcanzabilidad es indecidible. La prueba consiste en simular una máquina enumeradora y es muy parecida a la prueba de indecidibilidad de la alcanzabilidad para las redes Post-Automodificantes. De hecho, dos arcos reset, post o de transferencia son suficientes para hacer indecidible la alcanzabilidad [DFS98].

En la Tabla 1.2 se resume la situación de los problemas de decisión de las G-redes y de sus subclases, al inicio de la Tesis Doctoral de Dufourd [Duf98]. Las propiedades que aparecen en la tabla son: Terminación (T), Cobertura (C), Acotamiento (A), Acotamiento de un lugar (A-L), Acotamiento Estructural (A-E), Interbloqueo (I), Alcanzabilidad (Alc), Vivacidad (V) y t-Vivacidad (t-V).

	T	C	A	A-L	A-E	I Alc	V t-V
G-Redes	I	I	I	I	I	I	I
AM-Redes	I	I	I	I	I	I	I
G-Post+Reset	?	?	?	?	?	I	?
Reset	?	?	?	?	?	I	?
G-Post+Transfer	?	?	?	?	?	I	?
Transfer	?	?	?	?	?	I	?
G-Post	?	?	?	?	?	I	?
Post-AM	D	D	D	D	?	I	?
Redes de Petri	D	D	D	D	D	D	D

Tabla 1.2: Tabla Dufourd

En su tesis [Duf98], Dufourd consigue completar la tabla, obteniendo los resultados de la Tabla 1.3.

El diagrama de la Figura 1.14 resume los resultados, agrupando para cada problema las subclases para las que es decidible. Las conclusiones obtenidas en [Duf98] y que se pueden extraer de dicho diagrama son las siguientes:

- Hay muchas clases que extienden las redes de Petri siendo parcialmente analizables. En concreto, la cobertura es decidible para las redes Reset+G-Post que constituyen una clase amplia de redes de Petri extendidas. Además, el acotamiento de una red es decidible para las redes

	T	C	A	A-L	A-E	I Alc	V t-V
G-Redes	I	I	I	I	I	I	I
AM-Redes	I	I	I	I	I	I	I
G-Post+Reset	D	D	I	I	I	I	I
Reset	D	D	I	I	I	I	I
G-Post+Transfer	D	D	D	I	I	I	I
Transfer	D	D	D	I	I	I	I
G-Post	D	D	D	D	D	I	I
Post-AM	D	D	D	D	D	I	I
Redes de Petri	D	D	D	D	D	D	D

Tabla 1.3: Tabla Dufourd completa

Transfer+G-Post y el acotamiento de un lugar es decidible para las redes G-Post.

- El acotamiento de una red es indecidible para las redes de Petri con arcos reset (tres arcos son suficientes para hacer el problema indecidible).
- Para las redes de Petri con arcos de transferencia, el acotamiento de una red es decidible, mientras que el acotamiento de un lugar es indecidible.
- El acotamiento de una red separa las redes Transfer de las redes Reset y el acotamiento de un lugar separa las redes G-Post de las redes Transfer.

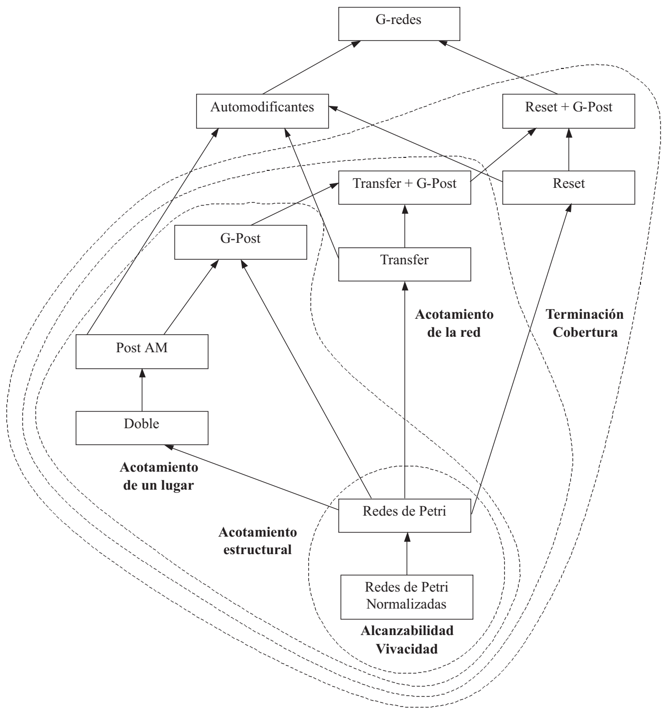


Figura 1.14: Clases de Decidibilidad

Capítulo 2

Líneas de Investigación

El propósito de este capítulo es presentar las dos líneas de investigación en las que nos hemos basado: las *redes automodificantes* [Val78a, Val78b, Val81] y las *gramáticas de grafos* [Ehr87][Sch93][Cor95][Bal00]. Ambas propuestas tienen relación con el formalismo de las redes de Petri y pretenden mejorar la expresividad del modelo básico de redes de Petri para que se puedan describir sistemas concurrentes en los que ocurren cambios dinámicos. En la Sección 2.1 presentamos las redes automodificantes de R. Valk [Val78a, Val78b, Val81], que son redes de Petri cuyas relaciones de flujo pueden cambiar en tiempo de ejecución. En la Sección 2.2 presentamos las gramáticas de grafos [Ehr87][Sch93][Cor95][Bal00], que extienden a las redes de Petri representando el estado de un sistema como un grafo y las transformaciones locales de estados como aplicaciones de producciones de grafos.

2.1. Redes Automodificantes

Las redes automodificantes de R. Valk [Val78a, Val78b, Val81] fueron un primer intento de extensión del modelo de redes de Petri con un mecanismo para el manejo de cambios. Además, son una generalización de las redes de Petri en la que las relaciones de flujo no son constantes sino que dependen del marcado actual que varía con la evolución del sistema.

Definición 86 (Red Automodificante) *Una* red automodificante *es una estructura $N = (P, T, F)$ en la que $P = \{p_1, \ldots, p_m\}$ es un conjunto finito y no vacío de lugares, $T = \{t_1, \ldots, t_n\}$ es un conjunto finito y no vacío de* *transiciones disjunto de* P *y* $F : (P \times T) \cup (T \times P) \rightarrow \mathbb{N}^{P_*}$ *es la relación de flujo, donde* $P_* = P \cup \{*\}$ *y* $* \notin P$.

Un vector $\varphi \in \mathbb{N}^{P_*}$ *puede representarse mediante una suma formal* $\varphi = \lambda_0 + \sum_{i=1}^{m} \lambda_i \cdot p_i$ *donde el coeficiente constante es la entrada correspondiente al lugar ficticio:* $\lambda_0 = \varphi(*)$ *y* $\lambda_i = \varphi(p_i)$.

Un marcado *de la red* N *es una función* $M : P \rightarrow \mathbb{N}$. *Si* $M \in \mathbb{N}^P$ *es un marcado y* $\varphi \in \mathbb{N}^{P_*}$, *entonces* $\varphi(M) = \lambda_0 + \sum_{i=1}^{m} \lambda_i \cdot M(p_i)$ *denota la evaluación de la función afín* φ *en el marcado* M.

$M[t\rangle M'$ *significa que la transición* t *está* habilitada *en el marcado* M *y lleva al marcado* M'. *Esta relación de transición viene dada por:*

$$M[t\rangle M' \Longleftrightarrow \forall p \in P \; M(p) \geq F(p,t)(M) \wedge M' = M - F(p,t)(M) + F(t,p)(M)$$

Una red automodificante marcada *es una red automodificante junto con un marcado inicial.*

Habitualmente, las relaciones de flujo dependen de un número limitado de lugares que llamamos *lugares de control*. Al contenido de estos lugares sólo le afectan ciertas transiciones llamadas *cambios de configuración* para diferenciarlas de las transiciones *ordinarias*.

Todo marcado M se asocia con una red de Petri $N_M = (P, T, F_M)$ que se obtiene evaluando las relaciones de flujo en el marcado M: $F_M(p,t) = F(p,t)(M)$ y $F_M(t,p) = F(t,p)(M)$. Esta red de Petri N_M es la *configuración* de la red automodificante N con marcado M. Siempre que no se modifiquen los lugares de control, la red automodificante se comporta como su configuración actual.

Las redes automodificantes se adaptan de manera adecuada a situaciones en las que coexisten varios modos de operación. Las nociones de lugares de control y configuración no son originarias del formalismo de las redes automodificantes, en cualquier ejemplo real pueden identificarse fácilmente. Si esto no ocurre, por ejemplo, si la mayoría de lugares pueden actuar como lugares de control y si la mayoría de transiciones los modifican, el comportamiento de la red no podrá predecirse.

En cualquier situación de modelado hay que identificar las posibles configuraciones del sistema, lo cual no significa que se hayan de enumerar todas las posibilidades ya que podrían ser infinitas. Una configuración se describe sólo *localmente*, de igual forma que el estado de un sistema descrito mediante una red de Petri viene dado por un conjunto de valores locales (los marcados). El espacio de estados de una red de Petri viene implícitamente dado por el conjunto de *marcados alcanzables*, que puede ser infinito, y aunque podemos decidir si un marcado es alcanzable, sabemos también que este procedimiento es, en general, bastante complejo. De manera similar, el conjunto de todas las configuraciones alcanzables de una red automodificante puede ser muy grande o incluso infinito.

La segunda etapa consiste en identificar cómo se llevan a cabo los cambios, es decir, cómo se va de una configuración a otra, que en el caso de las redes automodificantes tiene que ver con los lugares de control. La característica de estas redes que las hace interesantes en nuestra opinión es que la descripción de los cambios estructurales es *interna* (al formalismo) y *local*.

Como se indica en la Figura 2.1, las redes de Petri con *arcos inhibidores* (una transición t a la que le llega un arco inhibidor desde un lugar p está habilitada si lo está en el sentido habitual y además hay cero tokens en el lugar p) pueden implementarse como redes automodificantes. Por tanto, este formalismo tiene el poder computacional de la máquina de Turing y no se pueden obtener herramientas de verificación automáticas para toda la clase de redes automodificantes. Ya hemos visto en la Sección 1.4 que parece difícil encontrar subclases interesantes de redes automodificantes con propiedades decidibles (acotabilidad, vivacidad, alcanzabilidad, ...) excepto para las *redes Post-automodificantes* [Val78b] en las que todas las precondiciones son constantes ($\forall p \in P,\ t \in T,\ F(p,t) \in \mathbb{N}$). Además, Dufourd et al. en [DFS98] demuestran que la acotabilidad es también indecidible para las redes de Petri con *arcos reset* [AK77] (el disparo de una transición t con un arco reset desde un lugar p vacía el lugar p), que es un caso simple de automodificación (ver Figura 2.1). Otro caso simple de automodificación tiene que ver con los *arcos de transferencia* [Cia94] (se utilizan para transferir todos los tokens de un lugar p a un lugar p' cuando la transición t se dispara, vaciando p). En [DFS98], Dufourd et al. demuestran que la acotabilidad de un lugar es indecidible para redes de Petri con arcos de transferencia aunque la acotabilidad de la red sea decidible para estas redes. Sin embargo, hay que decir que la cobertura y la terminación, propiedades indecidibles para las redes automodificantes, son decidibles para las redes de Petri con arcos reset y para las redes de Petri con arcos de transferencia.

No obstante, existe un "gap" entre lo que es teóricamente expresable en términos de redes automodificantes y la forma en la que se pretende utili-

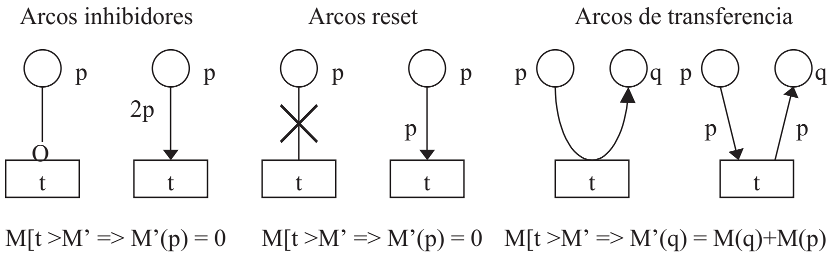


Figura 2.1: Implementación de arcos inhibidores, reset y de transferencia en redes automodificantes

zar este modelo para modelizar sistemas concurrentes adaptables a situaciones cambiantes. Aunque el uso de arcos reset y de transferencia resulta bastante natural en este contexto, es posible obtener un modelo más tratable con características similares. Pretendemos utilizar las redes automodificantes para describir un sistema formado por un conjunto de redes de Petri, llamadas sus configuraciones, junto con algún mecanismo (los llamados lugares de control y cambios de configuración en las redes automodificantes) que permita al sistema evolucionar de una configuración a otra bajo ciertas circunstancias. La idea de considerar las configuraciones y los cambios de configuración como "ciudadanos de primera clase" nos va a conducir al modelo de *redes reconfigurables*, una de las aportaciones de la presente tesis que, como veremos, constituye una alternativa que, a la vez que es más cercana a la modelización de los cambios dinámicos en sistemas concurrentes, constituye una clase de modelos para la que las propiedades básicas de las redes de Petri (acotabilidad de un lugar, alcanzabilidad, interbloqueo y vivacidad) siguen siendo decidibles. Es, por ello, un modelo más tratable tanto para modelización como para verificación.

2.2. Gramáticas de Grafos

Los sistemas concurrentes tienen una naturaleza bidimensional. Por una parte, distinguimos la estructura topológica del sistema, es decir, qué procesos existen y con qué otros procesos se comunican y, por otra parte, el comportamiento del sistema, es decir, el flujo de control dentro de los procesos y el flujo de comunicación entre ellos. Para describir sistemas dinámicos hay que tener en cuenta también qué procesos se crean y se destruyen y qué comunicaciones tienen lugar durante la ejecución.

Las redes de Petri son un formalismo ampliamente aceptado para especificar tanto la estructura de sistemas concurrentes/distribuidos como su comportamiento. Además, el estado de una red (la distribución de los tokens en el conjunto de lugares) tiene una naturaleza intrínsecamente distribuida y una transición modifica una parte local del estado, siendo posible el disparo concurrente de transiciones cuando se consumen conjuntos de tokens mutuamente disjuntos. Una situación de exclusión mutua se puede representar mediante dos transiciones compitiendo por un único token. Así pues, fenómenos como la exclusión mutua, la concurrencia, la composición secuencial y el indeterminismo pueden representarse de forma explícita.

Muchos investigadores afirman que las gramáticas de grafos son más expresivas que las redes de Petri para la especificación de sistemas concurrentes y distribuidos. Los grafos permiten describir sistemas concurrentes de forma intuitiva y poseen una base formal que permite especificar el comportamiento del sistema. Además, las reglas de reescritura de grafos permiten modelar los cambios dinámicos de la estructura topológiga de un sistema, mientras que la estructura topológica de las redes de Petri es estática. Sin embargo, en la literatura clásica, las gramáticas de grafos se han considerado en la mayoría de los casos como una generalización de las gramáticas de cadenas o de los sistemas de reescritura de términos para la reescritura de estructuras más complejas.

Las gramáticas de grafos surgen a partir de las gramáticas de N. Chomsky [HU79]. Utilizar gramáticas de grafos para especificar sistemas concurrentes significa dar un conjunto de reglas a partir de las cuales un objeto inicial dado (un grafo etiquetado), puede ser transformado. Por ello, los principales resultados relacionados con la concurrencia y el paralelismo de la teoría algebraica de gramáticas de grafos [Ehr87] tienen que ver con las nociones y resultados de los sistemas de reescritura de términos, tales como confluencia, propiedad de Church-Rosser, ortogonalidad de redexes, etc.

En la actualidad, existen muchas propuestas en la literatura que relacionan redes de Petri y gramáticas de grafos, y todas relacionan algunos conceptos básicos de los dos formalismos, como concurrencia de transiciones e independencia paralela de producciones. En [Sch93] y también en [Cor96], se puede encontrar un resumen. En la mayoría de las propuestas, las redes de Petri aparecen como una clase muy simple de gramáticas de grafos.

Una gramática de grafos permite describir un conjunto (posiblemente infinito) de grafos, es decir, los grafos que pueden obtenerse desde un grafo de partida por la aplicación sucesiva de producciones de grafos. Una producción de grafo es una regla del tipo $r : L \rightsquigarrow R$, que especifica que, bajo ciertas condiciones, cuando en un grafo G se detecta una ocurrencia (*match*) de la parte izquierda L, puede reemplazarse por la parte derecha R. La forma de las producciones de grafos, la noción de *match* y, en general, el mecanismo que establece cómo una producción puede aplicarse a un grafo y cuál es el grafo resultante, depende del formalismo específico de reescritura utilizado. Mostramos a continuación las definiciones básicas de la teoría algebraica de gramáticas de grafos conocida como *aproximación algebraica de doble pegado* (*double-pushout*) [Ehr87], en la que las nociones básicas de producción y derivación directa se definen en términos de construcciones y diagramas en una categoría adecuada.

Definición 87 (Grafo) *Dados dos alfabetos fijos* Ω_A *y* Ω_V *para las etiquetas de arcos y vértices, respectivamente, un* grafo *es una tupla* $G = (A, V, s, t, l_A, l_V)$*, siendo:*

- A *un conjunto finito de arcos,*
- V *un conjunto finito de vértices,*
- $s, t : A \to V$ *son las funciones origen y destino, y*
- $l_A : A \to \Omega_A$ *y* $l_V : V \to \Omega_V$ *son las funciones de etiquetado de arcos y vértices, respectivamente.*

Definición 88 (Grafo Discreto) *Un grafo* G *se denomina* discreto *si el conjunto de arcos* A *es vacío.*

Definición 89 (Subgrafo) *Un grafo* G' *es un* subgrafo *de* G *si* $A' \subseteq A$*,* $V' \subseteq V$ *y todas las funciones* s'*,* t'*,* $l_{A'}$*,* $l_{V'}$ *están restringidas a las correspondientes funciones de* G*.*

Definición 90 (Morfismo entre grafos) *Un* morfismo *entre grafos* $f : G \to G'$ *es un par* $f = (f_A : A \to A',\ f_V : V \to V')$ *de funciones que mantienen orígenes, destinos y etiquetas, es decir, tal que* $f_V \circ t = t' \circ f_A,\ f_V \circ s = s' \circ f_A,\ l'_V \circ f_V = l_V$ *y* $l'_A \circ f_A = l_A$*.*

Definición 91 (Isomorfismo) *Un morfismo* f *es un* isomorfismo *si tanto* f_A *como* f_V *son biyecciones.*

Definición 92 (Categoría de grafos) *Los grafos y morfismos entre grafos así definidos definen una categoría en el sentido de la teoría de categorías, llamada* categoría de grafos.

Definición 93 (Producción de Grafos) *Una* producción de grafo $(L \xleftarrow{l} K \xrightarrow{r} R)$ *es un par de morfismos inyectivos de grafos* $l : K \to L$ *y* $r : K \to R$*. Los grafos* L*,* K *y* R *se denominan* parte izquierda, interfaz *y* parte derecha *de la producción, respectivamente.*

Definición 94 (Diagrama de Pegado) *Dados dos morfismos entre grafos* $b : K \to B$ *y* $d : K \to D$ *siendo* b *inyectivo, el pegado de* B *y* D *a través de* K *viene dado por el grafo* G *descrito a continuación. Además, se obtienen los morfismos entre grafos* $c : D \to G$ *y* $g : B \to G$ *produciendo el diagrama de la Figura 2.2, llamado* diagrama de pegado*:*

$$\begin{tikzcd}
\mathbf{K} \arrow[r, "b"] \arrow[d, "d"'] \arrow[dr, phantom, "(1)"] & \mathbf{B} \arrow[d, "g"] \\
\mathbf{D} \arrow[r, "c"'] & \mathbf{G}
\end{tikzcd}$$

Figura 2.2: Diagrama de Pegado

1. $G = D + (B - b(K))$ *para los conjuntos de arcos y nodos, respectivamente, siendo:*

   - *la función origen de* G*, para todo* $a \in A$*,*

   $$s(a) = \begin{cases} s_D(a) & si\ a \in A_D \\ s_B(a) & si\ a \notin A_D \wedge s_B(a) \in V_{B-b(K)} \\ d(k) & si\ a \notin A_D \wedge s_B(a) = b(k)\ para\ k \in V_K \end{cases}$$

   - *la función destino de* G*, para todo* $a \in A$*,*

   $$t(a) = \begin{cases} t_D(a) & si\ a \in A_D \\ t_B(a) & si\ a \notin A_D \wedge t_B(a) \in V_{B-b(K)} \\ d(k) & si\ a \notin A_D \wedge t_B(a) = b(k)\ para\ k \in V_K \end{cases}$$

- *la función de etiquetado de arcos y vértices, para todo $x \in G$,*

$$l(x) = \begin{cases} l_D(x) & si\ x \in D \\ l_B(x) & en\ caso\ contrario \end{cases}$$

2. $c : D \to G$ *es la inclusión de D en G.*

3. $g : B \to G$ *se define para todos los items $x \in B$ como*

$$g(x) = \begin{cases} x & si\ x \in B - b(K) \\ d(k) & si\ x = b(k)\ para\ k \in K \end{cases}$$

Definición 95 (Condición de Pegado) *Dados $b : K \to B$ inyectivo y $g : B \to G$ podemos construir un grafo de contexto D produciendo el diagrama de pegado de la Figura 2.2 si y sólo si se satisface la siguiente* condición de pegado*:*

$$LÍMITE \subseteq PEGADO$$

donde $LÍMITE$ y $PEGADO$ son subgrafos de B definidos como sigue:

- $PEGADO = b(K)$
- $LÍMITE = PENDIENTE \cup IDENTIFICACIÓN$ *donde*
  $PENDIENTE = \{x \in V_B | \exists a \in (G - g(B))\ g(x) = s(a) \vee g(x) = t(a)\}$
  $IDENTIFICACIÓN = \{x \in B | \exists y \in B \quad x \neq y \ \wedge \ g(x) = g(y)\}$

Si g es inyectivo $IDENTIFICACIÓN = \emptyset$, definiéndose los puntos límite como aquellos puntos de B (subgrafo de G) que son origen o destino de los arcos de $G - B$. Los puntos de pegado de B vienen dados por K.

Definición 96 (Derivación directa) *Dado un grafo G, una producción $p : (L \xleftarrow{l} K \xrightarrow{r} R)$ y un morfismo de grafos $g : L \to G$, una* derivación directa *desde G hasta H utilizando p existe si y solamente si puede construirse el diagrama de la Figura 2.3, en el que ambos cuadrados han de ser diagramas de pegado.*

Una derivación directa desde un grafo G a un grafo H se obtiene en los tres pasos que se indican en el Algoritmo 2.

$$\begin{tikzcd}
\mathbf{L} \arrow[d, "g"'] & \mathbf{K} \arrow[l, "l"'] \arrow[r, "r"] \arrow[d, "k"] & \mathbf{R} \arrow[d, "h"] \\
\mathbf{G} & \mathbf{D} \arrow[l, "l'"] \arrow[r, "r'"'] & \mathbf{H}
\end{tikzcd}$$

Figura 2.3: Derivación directa como diagrama de pegado doble

Algoritmo 2 Derivación Directa

- Hacer "matching" de la parte izquierda L de la producción con un subgrafo del grafo G y comprobar que la condición de pegado se satisface.
- Considerar la parte izquierda L excepto los puntos de pegado K, es decir, $L - K$, y eliminarlos del grafo G. El resultado es un grafo de contexto D que todavía contiene los puntos de pegado K.
- Añadir la parte derecha R de la producción al grafo de contexto D juntando los puntos de pegado K de D y los puntos de pegado K de R. El resultado es el grafo H.

Definición 97 (Gramática de Grafos) *Una* gramática de grafos $\mathcal{G}$ *es una tupla* $\mathcal{G} = (G_0, P, \pi)$ *donde* G_0 *es el grafo inicial,* P *es un conjunto de nombres de producciones y* π *asocia cada nombre de producción con una producción de grafo. Se escribe* $p : (L \overset{l}{\leftarrow} K \overset{r}{\rightarrow} R)$ *si* $\pi(p) = (L \overset{l}{\leftarrow} K \overset{r}{\rightarrow} R)$*.*

En resumen, en la aproximación de doble pegado una producción de grafos consiste en un grafo parte izquierda L, un grafo parte derecha R y un grafo interfaz (común) K que forma parte tanto de R como de L, como puede verse en la Figura 2.4. Informalmente, para aplicar una de estas reglas a un grafo G hay que encontrar un *match*, una ocurrencia de su parte izquierda L en G. El mecanismo de reescritura primero elimina la parte de la parte izquierda L que no está en el interfaz K produciendo el grafo D, y añadiendo entonces la parte de la parte derecha R que no está en el interfaz K, obteniéndose el grafo H. Formalmente, esto se consigue requiriendo que los dos cuadrados de la Figura 2.4 sean diagramas de pegado en la categoría de grafos, de ahí el nombre de la aproximación. El grafo interfaz K se mantiene: es necesario para llevar a cabo el paso de reescritura, pero no se ve afectado en el paso en sí. Nótese que el interfaz K juega un papel fundamental para especificar cómo la parte derecha tiene que unirse con el grafo D. En las gramáticas de grafos en las que las producciones tienen un grafo interfaz K vacío, la expresividad se decrementa drásticamente: sólo podrán añadirse subgrafos desconectados.

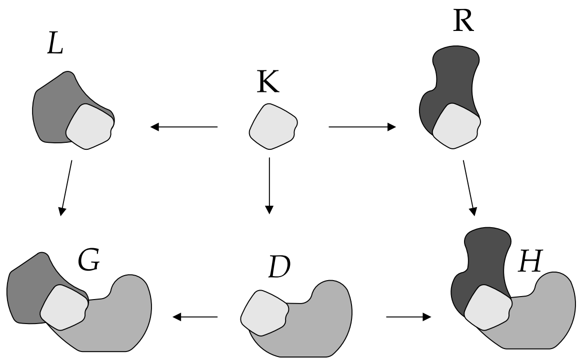


Figura 2.4: Un paso de reescritura de grafos en la aproximación de doble pegado

Las gramáticas de grafos pueden verse como una generalización de las redes de Petri, donde el estado de un sistema se describe mediante un grafo en lugar de mediante un conjunto de tokens. Ésta es la idea fundamental que se sigue en los grupos de investigación COMPUGRAPH de Berlín y Pisa. Andrea Corradini, perteneciente al grupo de Pisa, en [Cor95] demuestra que conceptos relevantes de la semántica concurrente de las redes pueden extenderse a las gramáticas. Básicamente, una red de Petri puede verse como un sistema de transformación de grafos que actúa sobre un tipo restringido de grafos, llamados grafos discretos etiquetados (que pueden considerarse como conjuntos de tokens etiquetados por lugares), y las producciones son las transiciones de la red. Desde este punto de vista, los sistemas de transformación de grafos generales son una verdadera extensión de las redes de Petri ordinarias en dos aspectos:

1. permiten especificar reescrituras dependientes del contexto, donde parte del estado es requerido pero no se ve afectado por el paso de reescritura;

2. permiten una descripción más estructurada del estado, un grafo arbitrario, posiblemente no discreto.

En el ejemplo de la Figura 2.5 (extraído de [Cor95]) se muestra una red desde el punto de vista de las gramáticas de grafos. La transición de la Figura

2.5 (a) se representa como la producción de grafo de la Figura 2.5 (b): esta producción consume los tokens de las precondiciones y produce tokens en las postcondiciones de la transición, siendo el grafo interfaz siempre vacío.

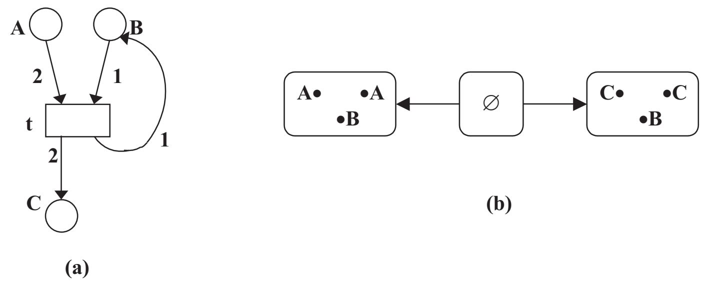


Figura 2.5: (a) Una transición de una red de Petri. (b) La correspondiente producción de grafos.

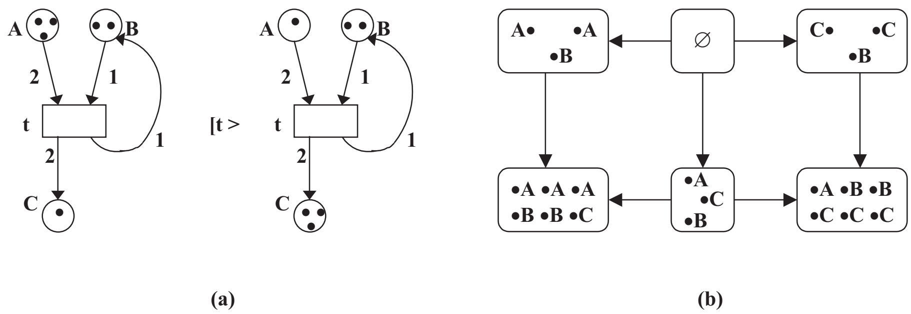


Figura 2.6: (a) Disparo de una transición de una red de Petri. (b) Diagrama de pegado doble correspondiente.

Un marcado se representa por el conjunto de nodos etiquetados por el lugar en el que se encuentran. En esta representación la estructura topológica de la red no se representa, evitando una excesiva complejidad en la producción. Una producción puede aplicarse a un marcado dado si y sólo si la correspondiente transición está habilitada; el diagrama de disparo doble produce el mismo marcado que el disparo de la transición; dos producciones pueden ocurrir paralela e independientemente en un marcado si y sólo si las correspondientes transiciones son concurrentes; etc. Por ejemplo, en la Figura 2.6 (a) se muestra el disparo de la transición de la Figura 2.5 (a) a partir del marcado formado por 3 tokens en A, 2 en B y 1 en C, y en la Figura 2.6 (b) se muestra la correspondiente derivación directa utilizando la producción de la Figura 2.5 (b).

Existe un "gap" de abstracción entre las gramáticas de grafos y las redes. El problema reside en el hecho de que los grafos son más concretos que los marcados: los nodos de un grafo, aunque tengan la misma etiqueta, tienen una identidad precisa, mientras que los tokens en un mismo lugar de una red no se pueden distinguir. Formalmente, el homólogo exacto de un marcado (multiconjunto) es un isomorfismo de grafos discretos. Por lo tanto, hay que imponer equivalencias convenientes si se quiere obtener una correspondencia exacta con las computaciones de las redes.

Capítulo 3

Revisión de Trabajos Relacionados

A lo largo del trabajo de la tesis se han estudiado otros modelos existentes que permiten la descripción de sistemas concurrentes dinámicos complejos. En este capítulo mostramos aquellas aproximaciones basadas en redes de Petri que, aunque tienen una formalización sintáctica muy distinta entre sí y tratar de expresarla de forma unificada resultaría complicado, tienen en común la idea de la reescritura que se sigue de las gramáticas de grafos. Vamos a presentar:

- las *redes móviles* de Asperti y Busi [AB96], en la Sección 3.1, originadas a partir de la combinación de redes de Petri L/T con técnicas de manejo de nombres típicas del π-cálculo [Mil91] y las *redes dinámicas* de Buscemi y Sassone [BS01], en la Sección 3.2, inspiradas en el join-cálculo [FG96, FGL$^+$96], que permiten la creación dinámica de componentes como en nuestra propuesta,

- las *redes* Δ de Gradit y Vernadat [GVA99, GV99], en la Sección 3.3, un formalismo de reescritura que integra las ventajas de las redes de Petri y las gramáticas de grafos, respectivamente para la especificación del comportamiento y las transformaciones topológicas de un sistema workflow [EN93],

- el *formalismo POP* introducido por Engelfriet, Leih y Rozenberg [ELR91], en la Sección 3.4, que puede verse como una gramática de grafos, y

- el modelo relacionado de los *autómatas cooperativos* de Badouel, Darondeau y Tokmakoff [BDT99], en la Sección 3.5, en los que los tokens son elementos activos con comportamiento dinámico.

Para todos ellos, como en nuestro modelo, la descripción de los cambios es interna e incremental y su tratamiento es local. La idea de reescritura también subyace en todas las propuestas, una configuración del sistema se describe como una red de Petri y un cambio de configuración como una regla de reescritura de grafos que reemplaza la parte del sistema que hace matching con la parte izquierda de la regla de reescritura por la correspondiente parte derecha. Con respecto al poder expresivo, todos ellos son equivalentes a la máquina de Turing, como nuestro modelo. Pero el modelo de los sistemas de reescritura de redes es mucho más intuitivo puesto que es un modelo más cercano a las redes de Petri.

3.1. Las Redes Móviles

La propuesta de las *redes móviles* [AB96] se plantea como una solución al problema de que las redes de Petri no ofrecen una manera directa de expresar el cambio de estructura que pueden sufrir los sistemas. Se originan a partir de la fusión de redes de Petri L/T con las técnicas de manejo de nombres típicas del π-cálculo [MPW92]. Son una clase de redes de alto nivel con movilidad y características dinámicas. La movilidad se añade en el sentido de R. Milner en el π-cálculo [Mil91], es decir, pasar una referencia a un proceso (un nombre de canal) en una comunicación. Desde el punto de vista de las redes de Petri, los canales se interpretan como los lugares y los tokens como los nombres.

Las redes móviles son un modelo adecuado para representar sistemas reconfigurables dinámicamente, en los que la interconexión (así como el número) de componentes puede cambiar durante la ejecución. Son una variante de las *redes coloreadas* [Jen92], en las que los colores de los tokens son tuplas de nombres. Así, las principales características de las redes móviles son:

- El postset de una transición no es fijo sino que depende del color de los tokens consumidos y algunos lugares en el preset se definen dinámicamente, como veremos más adelante.

- La estructura de la red puede cambiar dinámicamente debido al disparo de una transición: nuevas redes pueden añadirse, sus lugares pueden fusionarse con los lugares existentes y las transiciones pueden eliminarse.

La introducción de movilidad en la salida (esto es, que el postset no sea fijo a priori) está basada en la siguiente idea: las transiciones de las redes L/T pueden verse como instantáneas (es decir, pueden producir tokens en el postset mientras consumen tokens del preset), ya que no hay ninguna dependencia funcional del postset sobre el preset. Si nos centramos en las redes coloreadas, el color de los tokens en el postset depende del color de los tokens en el preset; así, el disparo de una transición consiste en tres fases: consumir los tokens del preset, calcular el color de los tokens a producir y producir los tokens en el postset. Ya que los tokens en el postset pueden producirse sólo después de procesarse el color de los tokens consumidos, también podría hacerse que los lugares dependieran de los tokens consumidos. Esta es la forma en la que se obtienen redes móviles. El mecanismo de procesado del color se realiza por simple sustitución de nombres de lugares, ya que los tokens se colorean con (tuplas de) nombres de lugares. Por ejemplo, considerar la transición $a(x), b(y) \rightarrow x\langle y\rangle$: un token de un lugar a y un token de un lugar b se consumen y un token, con el mismo color que el token consumido de b, se produce en el lugar especificado por el color del token consumido de a.

La extensión a movilidad en la entrada (un preset no fijo) está motivada por el siguiente razonamiento: cuando se consideran redes con unificación de nombres en el preset (es decir, la transición $a(x), b(x) \rightarrow x$ está habilitada únicamente si ambos lugares a y b contienen un token del mismo color), se tiene una dependencia entre los colores de los tokens consumidos. En las redes móviles con movilidad en la entrada, esta dependencia se extiende a los lugares del preset: el preset se especifica mediante una lista (en lugar de un multiconjunto), y el lugar de un ítem en el preset puede especificarse mediante el color de un token de un ítem precedente. Por ejemplo, la transición $a(x), x() \rightarrow b$ consume un token del lugar a y un token (sin color) del lugar especificado por el color del token consumido del lugar a (y produce un token sin color en el lugar b).

Para tratar con redes que crecen dinámicamente, en lugar de marcados simples, se permite que el postset de una transición sea una subred. Cuando tal transición se dispara, los nuevos lugares y transiciones de la red postset se añaden a la estructura actual de la red; además, se requiere que algunos de los lugares de la subred se fusionen con los lugares ya existentes. Este tipo de redes se denominan *redes dinámicas*.

Si se tiene en cuenta la posibilidad de que una red pueda disminuir de tamaño, es muy difícil permitir una eliminación directa de lugares, debido a la generación de referencias que se quedan colgadas, que se corresponden con la ocurrencia de los nombres de los lugares eliminados en otras partes de la red. Por esta razón, se permite solamente la eliminación directa de transiciones; en cuanto a los lugares, los que ya no se utilizan se eliminan mediante una forma implícita de recogida de basura.

El mecanismo de borrado de transiciones consiste en lo siguiente: toda transición está etiquetada (una etiqueta que no tiene por que ser única); además del preset y postset, a una transición se le asocia también el conjunto a borrar, es decir, un conjunto de etiquetas correspondientes a las transiciones que se quieren eliminar cuando se efectúe su disparo.

Finalmente, se presenta una forma implícita de recogida de basura: los lugares que quedan aislados, es decir, su nombre no ocurre en ningún lugar de la red, lo que implica que ninguna transición puede producir/consumir tokens desde ellos, pueden eliminarse sin que cambie el comportamiento de la red; lo mismo ocurre con los lugares vacíos, cuyos nombres ocurren como mucho en el preset de alguna transición (en este caso, también se eliminan las transiciones con un lugar de este tipo en el preset).

3.2. Las Redes Dinámicas

La analogía entre los términos join del join-cálculo [FG96, FGL$^+$96] y las redes de Petri es relativamente sencilla y se puso de manifiesto en [AB96]. Nombres, mensajes y definiciones elementales del join-cálculo se corresponden con lugares, tokens y transiciones de las redes de Petri. La correspondencia, sin embargo, acaba pronto porque las redes no son un formalismo de paso de valores y no pueden expresar movilidad. Desafortunadamente, tienen una topología de red estática e inmutable. Lo que sugiere la búsqueda de extensiones de redes de Petri que permitan el estudio de redes con movilidad. En [BS01], M. Buscemi y V. Sassone presentan tres extensiones de las redes de Petri L/T obtenidas añadiendo, de manera jerárquica, las características necesarias para alcanzar toda la expresividad del join-cálculo. Dichas características y dichas extensiones son:

1. El paso por valor que se consigue con (una versión de) las *redes coloreadas.*

2. La movilidad vista como reconfigurabilidad de redes, obtenida con las *redes reconfigurables.*

3. El crecimiento dinámico modelado mediante la noción de *redes dinámicas*.

En las redes coloreadas [Jen92] los tokens conllevan cierta información: los colores. Los arcos entre lugares y transiciones están etiquetados con expresiones que se evalúan según los colores de los tokens después de ligar variables libres con colores. Las expresiones de los arcos de entrada y de salida describen, respectivamente, los recursos necesarios para que sea posible el disparo y los recursos generados por dicho disparo. El disparo de una transición se permite sólo si todos los arcos de entrada de la transición tienen el mismo color. Los nombres en los marcados representan colores y en las transiciones representan variables ligadas. Los disparos sólo afectan al marcado, es decir, las redes de Petri coloreadas tienen una estructura estática. Esto es totalmente inadecuado para representar estructuras que cambian dinámicamente. Las redes reconfigurables generalizan a las redes coloreadas añadiéndoles un ingrediente: cada disparo de una transición puede dar lugar a un conjunto diferente de lugares de salida. Esto equipa a las redes con un mecanismo para modelizar redes con topologías reconfigurables, es decir, redes en las que el conjunto de componentes es fijo, pero la conectividad entre ellos puede cambiar con el tiempo. Formalmente, se consigue con una alteración muy suave de la definición de redes coloreadas (que permite a los marcados y mensajes contener nombres privados) y de su regla de disparo (los nombres recibidos incluyen lugares de salida además de nombres en los arcos de salida).

La generalización obvia de las redes reconfigurables es permitir la creación dinámica de componentes, lo que se consigue mediante las nociones de redes de Petri dinámicas. La idea que hay detrás de tales estructuras es que el disparo de una transición asigna un nuevo parámetro de red en los valores actuales de los nombres recibidos. Como la red puede consistir simplemente en un marcado y ninguna transición, esto incluye la definición estándar de las redes L/T. Por tanto, las redes coloreadas y reconfigurables son tipos especiales de redes dinámicas. El rasgo característico de las redes dinámicas, que las distingue de otras aproximaciones y dibuja una conexión con [AB96], es que, como para las redes reconfigurables, los arcos de entrada nunca se modifican. Aunque es posible modificar dinámicamente el postset de una transición, y también generar nuevas subredes, no es posible añadir lugares a los presets de las transiciones. Esto nos permite generalizar las funciones postset de las redes permitiendo que sean redes dinámicas. Lo que significa que la definición de estas redes es recursiva.

3.3. Las Redes Δ

Las *redes* Δ, introducidas por P. Gradit, F. Vernadat y P. Azéma en [GVA99, GV99], son un formalismo de reescritura basado en las redes de Petri [Rei85] y en las gramáticas de grafos [Ehr87][Sch93][Cor95], que se utilizan para definir sistemas workflow [EN93]. En un sistema workflow se distinguen dos aspectos complementarios: el comportamiento de los objetos y la dinámica de las interacciones entre objetos. Las redes de Petri y las gramáticas de grafos son apropiadas, respectivamente, para especificar el comportamiento y para transformaciones en la topología. Una red Δ combina los dos modelos para especificar dos niveles asociados con cada uno de los aspectos del workflow. El *nivel de coordinación* expresa los aspectos de comportamiento y se interpreta como una gramática de grafos mientras que el *nivel de procesamiento* expresa aspectos estructurales y se corresponde con la aproximación de redes de Petri. A continuación, se presentan las definiciones formales del modelo de redes Δ.

Definición 98 (Red Δ) *Una* red Δ *es una tupla* $\Delta = (C, V, S, T)$ *donde* C *es un conjunto de constantes,* V *es un conjunto de variables,* S *es un conjunto de figuras y* T *es un conjunto de transiciones. Formalmente,* $T \subset (P \times \mathcal{P}(P)) \times (P \times P)$ *siendo* $P = \mathcal{P}(V \times S) \times \mathcal{P}(V \times V)$ *el* conjunto de patrones.

Un patrón $p \in P$ *viene dado por un par* (p_S, p_G)*, siendo* p_S *un elemento de* $\mathcal{P}(V \times S)$ *(par figura) y* p_G *un elemento de* $\mathcal{P}(V \times V)$ *(par arco).*

Una transición $t \in T$ *viene dada por una condición, compuesta por un patrón* (t_0) *y un conjunto de patrones prohibidos* $\{t_i^{\neg}\}$*, y un par de patrones: consumido* (t^-) *y producido* (t^+)*.*

Notación 99 *Se denota por* $\rho(p) \in \mathcal{P}(P)$ *el conjunto de variables presentes en el patrón* p*, por* $\rho^0(t) = \rho(t^-) \cup \rho(t^0)$ *el conjunto de referencias requeridas en una transición y por* $\rho^+(t) = \rho(t^+) \cup \rho^0(t)$ *el conjunto de referencias creadas por una transición.*

Definición 100 (Red Δ marcada) *Una* red Δ marcada *es una red* Δ *y un marcado* M *donde* $M \in \mathcal{P}(C \times S) \times \mathcal{P}(C \times C)$*. Un* marcado *es un patrón de constantes. La misma notación se utiliza tanto para marcados (construidos sobre el conjunto de constantes* C*) como para patrones (construidos sobre el conjunto de variables* V*).* $\gamma(M) \in \mathcal{P}(C)$ *denota el conjunto de constantes utilizadas.*

Notación 101 *Sea Σ el conjunto de sustituciones desde V a C. Sea $\sigma \in \Sigma$ una sustitución. La extensión canónica de σ, que asocia un patrón ($\in \mathcal{P}(V \times P) \times \mathcal{P}(V \times V)$) con un marcado ($\in \mathcal{P}(C \times P) \times \mathcal{P}(C \times C)$), también se denota por σ.*

Definición 102 (Transición habilitada) *Se dice que un marcado M* habilita una transición t *a través de una sustitución $\sigma \in \Sigma$ si y sólo si:*

(a) La imagen a través de σ de los patrones requeridos y consumidos está "presente" en el marcado.

(b) No está presente ninguna imagen de patrón prohibido.

(c) La imagen de las referencias creadas es distinta a la de las referencias usadas en el marcado actual M.

(d) La imagen del patrón producido es distinta en el marcado actual M.

$$
\begin{array}{ccc}
M \overset{\sigma(t)}{\rightarrow} \; sii & (\sigma(t^- \cup t^0) \subset M) & (a) \\
\wedge & \begin{array}{c}[\forall \tau \in \Sigma \; tal \; que \; \tau_{|\rho(t)} = \sigma_{|\rho(t)}, \\ \neg(\exists t_i^\neg \; / \; \tau(t_i^\neg) \subset M)]\end{array} & (b) \\
\wedge & (\sigma(p^+(t)) \cap (\gamma(M)) = \emptyset) & (c) \\
\wedge & (\sigma(t^+) \cap M = \emptyset) & (d)
\end{array}
$$

Definición 103 (Disparo de una transición) *El* disparo de una transición habilitada *eliminará la imagen del patrón consumido y añadirá la imagen del patrón producido.*

$$
Si \; M \overset{\sigma(t)}{\rightarrow} \; entonces \; M \overset{\sigma(t)}{\rightarrow} (M - \sigma(t^-)) \cup \sigma(t^+)
$$

Las redes Δ pueden considerarse como un sistema de reescritura que integra las ventajas de las gramáticas de grafos y las redes de Petri. Mediante un ejemplo, en [GVA99] se ilustra el interés del formalismo propuesto en dos aspectos complementarios de los CSCW [DME98]: la topología dinámica para el nivel de coordinación y la descripción del comportamiento para el nivel de procesamiento. Además, se presenta también un operador de composición de redes Δ que permite construir sistemas más complejos a partir de componentes elementales.

Con respecto al poder computacional, en [GV99] se demuestra que una red Δ es equivalente a una máquina de Turing.

3.4. El Formalismo POP

En [ELR91], J. Englefriet, G. Leih y G. Rozenberg introducen dos formalismos teóricos para modelizar sistemas paralelos basados en objetos: uno a nivel de sistemas de transición y el otro a nivel de programa. Ambos formalismos se basan en las redes de Petri, en la idea de que una red de Petri (en concreto los *sistemas de redes Elementales*, formados por una red simple y pura y un marcado inicial) puede verse como un sistema de transición para sistemas paralelos, similar a un autómata finito para sistemas secuenciales. Por ello, para modelizar un sistema paralelo basado en objetos lo que hacen es traducirlo a una red de Petri y añadirle algunas características básicas basadas en objetos, obteniendo lo que se denomina un *sistema Paralelo de Transiciones basado en Objetos* (a *Parallel Object-based Transition system - POT* -), también llamado *red de Petri con Transiciones basada en Objetos.* Así pues, una manera de describir el comportamiento de un sistema paralelo basado en objetos será traducirlo a un POT (normalmente infinito). Para estandarizar la definición de tales redes de Petri infinitas definen un lenguaje de programación con las características básicas de los lenguajes de programación basados en objetos: *Programa Paralelo basado en Objetos* (*Parallel based-Object Program - POP* -). Para formalizar la semántica del lenguaje POP, se asocia un POT con cada POP, de una manera directa, lo cual permite describir el comportamiento de un sistema paralelo basado en objetos traduciéndolo a un POP (finito). Los POPs se pueden comparar con las redes de Petri de alto nivel, pueden verse como "abreviaturas" de redes de Petri (por esta razón los POPs también se llaman *redes de Petri Programadas basadas en Objetos*).

En un sistema paralelo basado en objetos la tarea a realizar se divide en distintas subtareas, cada una de ellas llevada a cabo por un componente del sistema (llamado *objeto*). Cada objeto tiene su propia identidad y puede actuar de manera autónoma, secuencial y en paralelo con otros objetos. Los objetos tienen un ciclo de vida ya que al terminar ciertas tareas y empezar otras, pueden ser destruidos y creados. Cada objeto tiene su propia memoria local, en la que almacena sus datos privados (enteros, lógicos, referencias a otros objetos, un programa a ejecutar por el objeto, etc). Los objetos pueden comunicarse entre sí "citándose" y "acordando" intercambiar información. Por tanto, un objeto encapsula datos que pueden cambiar activamente, debido al propio objeto o a la comunicación con otros objetos. Los objetos se comunican participando en un evento síncrono y atómico durante el cual las memorias de los objetos participantes pueden consultarse y actualizarse, algunos objetos pueden destruirse y otros pueden crearse. En el sistema no se tienen en cuenta clases o herencia; por ello se llaman sistemas basados en objetos en lugar de orientados a objetos.

Notación 104 *En las definiciones que siguen se supone* $Ob = \{b_0, b_1, b_2, \ldots\}$ *un conjunto infinito de objetos. Formalmente no se distingue entre objetos y referencias a (direcciones de, nombres de) objetos. Se denota por* $\varphi_{fin}(Ob)$ *el conjunto de todos los subconjuntos finitos de* Ob*.*

Definición 105 (Estructura de Memoria) *Una* estructura de memoria *es una tupla* $(Var, Val, \perp, ref)$ *donde* Var *es un conjunto finito de variables,* Val *es un conjunto de valores, con* $Ob \subseteq Val$*,* $\perp \in Val$ *es el valor indefinido, con* $\perp \notin Ob$ *y* $ref : Val \to \varphi_{fin}(Ob)$ *es la función de referencia, con* $ref(b) = \{b\}$ *para todo* $b \in Ob$ *y* $ref(\perp) = \emptyset$*.*

Fijamos un conjunto infinito de variables de objeto $Ovar$. Si $B \in Ovar$ denota un objeto $b \in Ob$ y v es una variable de Var (para una estructura de memoria dada), entonces $B \cdot v$ denota la variable privada v de b.

Definición 106 (Programa Paralelo Basado en Objetos) *Un* Programa Paralelo basado en Objetos *(o una red de Petri Programada basada en objetos), POP, es un par* (M, T) *en el que*

- $M = (Var, Val, \perp, ref)$ *es una estructura de memoria y*
- T *es un conjunto finito de transiciones.*

Toda transición en T *es de la forma* $(L \to R, cond, asig)$ *donde*

- L *y* R *son subconjuntos finitos de* $Ovar$*, con* $L \cup R \neq \emptyset$*,*
- *cond es un conjunto finito de condiciones de la forma* $f(u_1, \ldots, u_n)$ *tal que*
    - $n \geq 1$*,*
    - $u_i \in L \cup R$ *o*
    - $u_i = A \cdot v$ *con* $A \in L$ *y* $v \in Var$ *(para* $1 \leq i \leq n$*),*

y f es una función $Val^n \to \{cierto, falso\}$, y

- *asig es un conjunto finito de asignaciones de la forma $B \cdot v := L \to f(u_1, \ldots, u_n)$, exactamente una para cada $B \in R$ y $w \in Var$, tal que*
  - $n \geq 0$,
  - $u_i \in L \cup R$ *o*
  - $u_i = A \cdot v$ *con* $A \in L$ *y* $v \in Var$ *(para* $1 \leq i \leq n$*),*

  y f es una función $Val^n \to Val$ que respeta las referencias, es decir, $\forall x_1, \ldots, x_n \in Val$, $ref(f(x_1, \ldots, x_n)) \subseteq (ref(x_i) \mid 1 \leq i \leq n)$.

Intuitivamente, en una transición $t = (L \to R, cond, asig)$, $L \cup R$ denota el conjunto de objetos que participan en t; los objetos (denotados mediante variables objeto) de L se comunican en t, los objetos de $R - L$ se crean en t, los objetos en $L - R$ se destruyen en t, y los objetos en $L \cap R$ sobreviven en t. Las condiciones se dan sobre los objetos de las variables de L y sobre (los nombres de) los objetos participantes. Las asignaciones expresan los valores de las variables de los objetos de R en términos de las variables de L y en términos de (los nombres de) los objetos participantes. El uso de tales asignaciones implica que la comunicación es local: únicamente la información presente en los objetos participantes puede utilizarse. Es lo que se denomina "propiedad de paso de información" de los POPs.

Los POPs pueden verse como gramáticas de grafos, en los que cada transición $t = (L \to R, cond, asig)$ es una producción de la gramática. El par $L \to R$ son los nodos de la parte izquierda y la parte derecha de la producción, mientras que parte de *cond* describe los arcos entre los nodos de L y parte de *asig* describe los arcos entre los nodos de R. El resto de *cond* describe condiciones de aplicación de la producción y el resto de *asig* describe el mecanismo de *embedding* de la producción. El tipo concreto de la gramática de grafos que se obtiene depende de qué funciones se permitan en las condiciones y las asignaciones.

3.5. Los Autómatas Cooperativos

Los Autómatas Cooperativos [BDT99] surgen como un formalismo basado en redes de Petri para modelar la coordinación de agentes en un sistema distribuido. Un sistema puede verse como una red de Petri de alto nivel en la que los tokens son elementos activos formados por un autómata y una memoria privada. Las transiciones de la red son vectores de sincronización que dan todas las posibles interacciones entre los distintos objetos dinámicos. La idea básica es que los objetos deben sincronizarse de acuerdo con su comportamiento (el servicio que ofrecen o el servicio que reciben) prescindiendo de su estado interno. Por ejemplo, si un proceso quiere imprimir un fichero, no le importa qué impresora lo imprima, cualquier objeto puede ser reemplazado por otro objeto que ofrezca el mismo servicio incluso si, ante otras peticiones, se comportan de forma muy distinta. Tal sincronización requiere la creación de objetos nuevos (tokens) y puede también eliminar otros objetos.

Definición 107 (Autómata Agente) *Un* Autómata Agente *es una estructura $A = (\Sigma, S, T)$ tal que Σ es un conjunto finito de nombres de acciones, S es un conjunto finito de estados en el que $S_\perp = S \cup \{\perp\}$ es el conjunto que se obtiene al añadir a S el valor indefinido $\perp \notin S$. $T \subseteq S_\perp \times \Sigma \times S_\perp$ es un sistema de transiciones para el que se asume que el conjunto de nombres de acciones puede dividirse en tres subconjuntos disjuntos $\Sigma = \Sigma_1 + \Sigma_2 + \Sigma_3$ tal que $T = T_1 + T_2 + T_3$ siendo $T_1 \subseteq \{\perp\} \times \Sigma_1 \times S$, $T_2 \subseteq S \times \Sigma_2 \times S$, y $T_3 \subseteq S \times \Sigma_3 \times \{\perp\}$. Los elementos de Σ_1, Σ_2, y Σ_3 se denominan respectivamente* creaciones, acciones regulares *y* destrucciones.

Definición 108 (Autómata Cooperativo) *Un* sistema de Autómatas Cooperativos *(AC) es una tupla (Σ, S, T, Att, R) donde (Σ, S, T) es un Autómata Agente (AA), Att es un conjunto finito de atributos, y R es un conjunto finito de reglas de transición.*

Cada regla $r = (synch, guard, update)$ está formada por un vector synch de acciones sincronizadas, requerido por los respectivos agentes que participan en la transacción, una condición guard que especifica las relaciones que pueden tener lugar entre ellos para habilitar la transacción y una actualización update de las relaciones para finalizar la transacción.

- *Un* vector de sincronización *es un vector $synch = A_1 \cdot a_1 + \cdots + A_n \cdot a_n$ en el que los a_i son nombres de acciones ($a_i \in \Sigma$) y los A_i son nombres de agentes con un alcance limitado a la regla (acotado en cada instanciación para los tokens agentes con estados locales s_i proporcionando acciones a_i).*
- *Una* condición *guard es una conjunción de restricciones de igualdad (resp. desigualdad) $A_i \cdot u = A_j \cdot v$ o $A_i \cdot u = 0$ (resp. $A_i \cdot u \neq A_j \cdot v$*

o $A_i \cdot u \neq 0$*) donde* a_i *y* a_j *son acciones regulares o de destrucción en* $^\bullet r = \{A_i \mid a_i \in \Sigma_1 \cup \Sigma_3\}$ *y* u *y* v *son atributos.*

- *Una* actualización *update es un conjunto finito de asignaciones* $A_i \cdot u := A_j \cdot v$ *o* $A_i \cdot u = new$ *donde* a_i *es una acción regular o de creación en* $r^\bullet = \{A_i \mid a_i \in \Sigma_1 \cup \Sigma_2\}$ *y* u *y* v *son atributos, tales que no ocurren asignaciones múltiples.*

Un token del autómata es un par $x = (s, \rho)$ donde $s \in S$ es el estado del token x (escrito $x \cdot state = s$), y $\rho : Att \to \mathbb{N}$ almacena los valores $x \cdot u = \rho(u)$ de los atributos u del token x.

Un estado del sistema, o *marcado*, es un multiconjunto finito de tokens.

El disparo en un marcado M de una regla $r = (synch, guard, update)$ con vector de sincronización $synch = A_1 \cdot a_1 + \cdots + A_n \cdot a_n$ lleva al marcado $M' = M - L + R$, siendo $L = \{x_i \mid A_i \in {}^\bullet r\}$ y $R = \{x_i' \mid A_i \in r^\bullet\}$ multiconjuntos de tokens consumidos o producidos, respectivamente, en el disparo.

Definición 109 (Regla habilitada) *Un enlace* $\{A_i = x_i \mid A_i \in {}^\bullet r \wedge x_i \in M\}$ *habilita* r *si y sólo si:*

1. $L = \{x_i \mid A_i \in {}^\bullet r\}$ *es más pequeño que M;*
2. $x_i \cdot state \xrightarrow{a_i} s_i$ *en AA siempre que* $A_i \in \Sigma_2$*;*
3. $x_i \cdot state \xrightarrow{a_i} \perp$ *en AA siempre que* $A_i \in \Sigma_3$*;*
4. *la condición guard es válida bajo este enlace.*

Definición 110 (Disparo de una regla) *El disparo de* r *bajo un enlace que habilite implica:*

1. *la producción de tokens agentes* x_i' *tales que* $\perp \xrightarrow{a_i} x_i' \cdot state$ *en AA para todas las acciones de creación* $a_i \in \Sigma_1$*;*
2. *la transformación de tokens* x_i *en tokens* x_i' *tales que* $x_i' \cdot state = s_i$ *para todas las acciones regulares* $a_i \in \Sigma_2$*;*
3. *el borrado de tokens agente* x_i *para todas las acciones de destrucción* $a_i \in \Sigma_3$.

Definición 111 (Actualización de las relaciones) *Los atributos $u, v \ldots$ de los tokens producidos en $R = \{x_i^{'} \mid A_i \in r^{\bullet}\}$ tienen los valores $x_i^{'} \cdot u$, $x_i^{'} \cdot v \ldots$ de manera que se cumple lo siguiente para $A_i = x_i$ para $A_i \in {}^{\bullet}r$ y $A_i^{'} = x_i^{'}$ para $A_i \in r^{\bullet}$:*

1. *$A_i^{'} \cdot u = A_j \cdot v$ siempre que $A_i \cdot u := A_j \cdot v$ sea una actualización update y $A_j \in {}^{\bullet}r$;*
2. *$A_i^{'} \cdot u = A_i \cdot u$ siempre que $A_i \in {}^{\bullet}r \cap r^{\bullet}$ y A_i no esté asignada en update;*
3. *$A_i^{'} \cdot u = 0$ siempre que $A_i \in r^{\bullet} \backslash {}^{\bullet}r$ y A_i no esté asignada en update;*
4. *todos los valores $A_i^{'} \cdot u$ tales que $A_i \cdot u := new$ esté en update son diferentes (para todo i, u) y difieren de los valores de los atributos de todos los tokens de M;*
5. *$A_i^{'} \cdot u = A_j^{'} \cdot v$ siempre que $A_i \cdot u := A_j \cdot v$ esté en update y $A_j \in {}^{\bullet}r \cap r^{\bullet}$.*

Definición 112 (Sistema Marcado de Autómatas Cooperativos) *Un* sistema marcado de Autómatas Cooperativos *es un Autómata Cooperativo con un marcado inicial.*

La memoria local de los tokens se presenta como listas atributo/valor. El número de atributos da una medida de la expresividad del modelo en términos de flujo de información entre objetos. En base a esto, se establece una jerarquía de AC con tres clases:

- la clase total (con dos o más atributos) es adecuada para el diseño de aplicaciones *groupware*,
- la clase intermedia (con un atributo) es adecuada para aplicaciones workflow y
- el modelo básico (sin atributos) permite solamente coordinación pura de eventos sin flujo de información.

No pueden obtenerse herramientas de verificación automática para la clase total de autómatas cooperativos puesto que está demostrado que el modelo tiene la complejidad de Turing. Sin embargo, sí que existen técnicas de verificación para la subclase de automátas cooperativos workflow; en particular, la verificación de la acotabilidad puede obtenerse construyendo el árbol de cobertura correspondiente.

Capítulo 4

Redes Reconfigurables. Propuesta inicial

En [BO98a] aparece por primera vez la definición de *redes reconfigurables* como una clase de redes de Petri de alto nivel que pueden cambiar dinámicamente su propia estructura reescribiendo alguna de sus componentes. Esta clase de redes de Petri soporta cambios dinámicos dentro de los sistemas concurrentes y distribuidos, en particular, se pueden utilizar para describir los sistemas workflow, como en [EKR95]. Una red de Petri reconfigurable es una red de Petri con reglas de modificación de la estructura locales que reemplazan una de sus subredes por otra subred. Los tokens de los lugares eliminados se transfieren a los lugares creados. La técnica utilizada por Van der Aalst [VDA97, VDAVH02] para verificar la seguridad de las redes workflow puede generalizarse a las redes reconfigurables. La acotabilidad de una red reconfigurable puede decidirse construyendo una forma simplificada del árbol de cobertura de Karp y Miller; sin embargo, esta construcción no permite decidir si un lugar está acotado o no. Es posible traducir una red reconfigurable a una red automodificante equivalente, de forma que, las redes reconfigurables aparecen como una subclase de las redes automodificantes para las que la acotabilidad puede decidirse. Sin embargo, la traducción utiliza características como arcos reset cuando se sabe que (como se ha visto en el Capítulo 1) la acotabilidad para la clase de redes de Petri con arcos reset es indecidible.

El modelo de las redes automodificantes permite modelizar sistemas concurrentes que se adaptan a situaciones cambiantes. Sin embargo, el uso de arcos reset y de transferencia que hace que aumente la expresividad de las redes automodificantes frente a las redes de Petri, hace al modelo intratable. Es posible obtener modelos más tratables con características similares. La idea es utilizar las redes automodificantes para modelizar sistemas consistentes en un conjunto de redes de Petri, llamadas *configuraciones*, junto con algún mecanismo que permita al sistema evolucionar de una configuración a otra bajo ciertas circunstancias. Para describir tales sistemas, introducimos el modelo de las *Redes de Petri Controladas por un Autómata* (a partir de ahora, RPCA). Una RPCA es una familia de redes de Petri paramétricas, esto es, redes de Petri cuyas relaciones de flujo son funciones lineales que dependen de varios parámetros, junto con algún control dado por un autómata finito cuyos estados determinan los valores concretos de dichos parámetros. Una RPCA es, por tanto, la composición en cascada (o producto semidirecto) de un autómata (control) con una red de Petri paramétrica. En este sentido, es bastante similar a un *Sistema de Adición de Vectores con Estados* (SAVE) que es el producto directo de un autómata con una red de Petri. Lo que implica la posibilidad de asociar cada RPCA con algún SAVE equivalente (debilitado). Esta noción de equivalencia débil es suficiente para asegurar que ambos modelos son equivalentes con respecto a ciertas propiedades fundamentales como acotabilidad de un lugar, alcanzabilidad y vivacidad. Puesto que todas estas propiedades son decidibles para los SAVE, lo son también para las RPCA.

En los trabajos iniciales, que vamos a detallar en este capítulo, el estudio se centra en una subclase de las redes reconfigurables, las llamadas *redes reconfigurables reversibles*, cuyas reglas de modificación de la estructura son formalmente invertibles (pueden aplicarse en ambos sentidos). Esta asunción es bastante natural ya que se supone que el sistema a modelizar no puede tener un comportamiento degradado, es decir, no puede perder ninguna de sus habilidades a lo largo de su ejecución. Estas redes pueden verse como la composición en cascada de un autómata con una red de Petri paramétrica, es decir, como una RPCA, por tanto, acotabilidad, alcanzabilidad y vivacidad son decidibles para las redes reconfigurables reversibles. Además, demostramos que, bajo algunas asunciones adicionales, una red reconfigurable reversible es equivalente a una red de Petri estratificada.

En las dos primeras secciones de que consta este capítulo se presentan los estudios previos sobre redes reconfigurables que son el punto de partida de la presente Tesis Doctoral. Finalmente, en la Sección 4.3 mostramos algunos ejemplos de sistemas concurrentes sujetos a cambios dinámicos, que modelizamos utilizando redes reconfigurables reversibles, redes de Petri y redes de Petri estratificadas, mostrando que la elección de un modelo particular depende de la naturaleza del problema a modelizar.

4.1. Cambios Dinámicos en Sistemas Workflow: Redes Reconfigurables

En esta sección se presentan una clase de redes de Petri de alto nivel, llamadas *redes reconfigurables* [BO98a], que pueden modificar su propia estructura reescribiendo alguno de sus componentes, permitiendo cambios dinámicos en *sistemas workflow* [EN93]. Las redes reconfigurables son una extensión natural de las redes de Petri, de manera que la mayoría de los resultados teóricos y de las herramientas automáticas que existen para las redes de Petri pueden utilizarse o adaptarse para ellas. Por ejemplo, la acotabilidad de una red reconfigurable puede decidirse construyendo su árbol de cobertura.

Presentamos, en primer lugar, los sistemas workflow en la Subsección 4.1.1. Las redes reconfigurables se presentan en la Subsección 4.1.2 en la que también se indica cómo pueden utilizarse para modelizar redes workflow con una estructura dinámica, las llamadas *redes workflow reconfigurables*. En la Subsección 4.1.3 se demuestra que puede decidirse si una red reconfigurable está acotada construyendo su árbol de cobertura, que la seguridad de una red workflow reconfigurable implica su acotabilidad y que puede también verificarse utilizando el árbol de cobertura. Se prueba también una analogía de un resultado de [VDA97] demostrando que la seguridad de una red workflow reconfigurable se reduce al acotamiento y vivacidad de la red reconfigurable obtenida añadiendo una transición extra conectando el lugar de salida de la red workflow reconfigurable a su lugar de entrada. En la Subsección 4.1.4 se demuestra que cualquier red reconfigurable puede simularse con una red automodificante.

4.1.1. Sistemas Workflow

Hay diferentes tipos de trabajo, tales como hacer pan, montar una cama, diseñar una casa o recoger resultados para obtener una estadística. En todos estos ejemplos, se puede ver el objeto tangible que se produce o modifica (pan, cama, casa, estadística). A tal objeto se le llama *caso*. Un caso puede ser algo más abstracto como una reclamación de seguros.

Todo caso tiene un principio y un final, y debe poderse distinguir de cualquier otro caso. Cada caso implica realizar un *proceso*. Un proceso consiste en un número de *tareas* que se llevan a cabo y un conjunto de *condiciones* que determinan el orden de las tareas. Una tarea es una unidad lógica de trabajo que se lleva a cabo de una sola vez por un *recurso*. Un recurso es el nombre genérico de una persona, máquina o grupo de personas o máquinas que pueden llevar a cabo tareas específicas.

La organización del trabajo, tanto dentro de como entre compañías, es cada vez más y más complicada. Por ello, se han desarrollado sistemas de información (automáticos) que administran procesos y los coordinan. Un sistema de información tiene que controlar, monitorizar y soportar la logística de todos los procesos de una empresa (*bussiness process*). En otras palabras, el sistema de información tiene que administrar el flujo de trabajo (*workflow*) a través de la organización.

Una definición de *workflow* consiste en la definición de un proceso, un resumen de los recursos que requiere y la clasificación de tales recursos en clases. Un *sistema workflow* es aquel que administra los workflows en una situación específica de una empresa y organiza el camino que deben seguir los datos entre los recursos humanos y a través de los programas de aplicación. Un *sistema de administración de workflows* es un paquete software que implementa un sistema workflow. Un sistema workflow consiste, generalmente, en un sistema de administración de workflows junto con la definición de los recursos y los procesos, aplicaciones, un sistema de bases de datos, etc.

Los conceptos básicos de la administración de workflows se pueden consultar en la web de la *Coalición de Administración de Workflows* (*Workflow Management Coalition* - WfMC) [Webe], una organización internacional que pretende estandarizar los productos de administración de workflows.

Las redes de Petri se han utilizado para modelizar, analizar y verificar propiedades de los sistemas workflow [EN93, EKR95][DME98][VDA97, VDAVH02].

4.1.2. Redes Reconfigurables

Las redes reconfigurables son redes de Petri de alto nivel que soportan cambios dinámicos en los sistemas workflow, es decir, cambios locales en la planificación de las tareas requeridas para procesar un caso que esté fluyendo en ese momento por el sistema. Veamoslo con un ejemplo.

Ejemplo 113 (Orden de Pedido [EKR95]) *Consideremos una gestión muy sencilla de las tareas a realizar por el departamento de ventas de una com-*

pañía. Cuando un cliente solicita un pedido por mail o en persona comienza el trabajo. El administrador rellena un formulario, se comprueba si hay existencias del producto (Comprobar Inventario) y, a continuación, se comprueba el crédito del cliente (Comprobar Crédito). Después de esta evaluación, bien se envía al cliente una carta de rechazo o bien la orden se aprueba y pasa a Facturación y Envío. El departamento de Envío hará que el pedido se envíe al cliente y el departamento de Facturación enviará la factura al cliente. Por último, se procederá a Archivar.

Así pues, el caso es una orden de pedido de un cliente y las tareas implicadas son Comprobar Inventario, Comprobar Crédito, Enviar, Facturar y Archivar.

Un cambio dinámico puede habilitar la ejecución paralela de dos tareas (en el Ejemplo 113 podrían ser Envío y Facturación) que, anteriormente, se realizaban secuencialmente en algún orden; alguna tarea también puede refinarse en tareas más elementales siguiendo cierto orden. En cualquier caso, se asume que el conjunto de tareas implicadas así como el conjunto de cambios locales se conocen con antelación y pueden listarse. Esta asunción implica que el conjunto de instancias de tareas es finito. Este conjunto constituye el conjunto de transiciones de una red reconfigurable. El cambio de una configuración a otra debido a un cambio local se representa introduciendo un nuevo tipo de contenido en dicho lugar que denota si un lugar existe o no en el estado actual del sistema. Por tanto, definimos una red reconfigurable como sigue.

Definición 114 (Red Reconfigurable) *Una* red reconfigurable *es una estructura $N = (P, T, F, R)$ en la que*

- $P = \{p_1, \ldots, p_m\}$ *es un conjunto no vacío y finito de lugares,*
- $T = \{t_1, \ldots, t_n\}$ *es un conjunto no vacío y finito de transiciones disjunto de P ($P \cap T = \emptyset$),*
- $F : (P \times T) \cup (T \times P) \to \mathbb{N}$ *es una relación de flujo ponderada, y*
- $R = \{r_1, \ldots, r_k\}$ *es un conjunto finito de reglas de modificación de la estructura.*

Una regla de modificación de la estructura *es una aplicación $r : P_1 \to P_2$ cuyo dominio y codominio son conjuntos disjuntos de lugares ($P_1, P_2 \subseteq P$ y $P_1 \cap P_2 = \emptyset$).*

Un marcado *de una red N es una aplicación $M : P \to \mathbb{N} \cup \{\alpha\}$ donde $\alpha \notin \mathbb{N}$, cuando $M(p) = \alpha$ se dice que el lugar p no existe en el marcado M mientras que $M(p) = n \in \mathbb{N}$ expresa que p existe en el marcado M y tiene valor n.*

Decimos que $E = T \cup R$ denota el conjunto de eventos *de una red reconfigurable.*

$M\,[e\rangle\,M'$ denota el hecho de que el evento e está habilitado en el marcado M y que la red alcanza el marcado M' cuando se dispara este evento. Esta relación de transición se define como sigue.

- *Una* transición $t \in T$ *está* habilitada *en el marcado M si:*

  $$\forall p \in P \ M(p) \neq \alpha \Longrightarrow M(p) \geq F(p,t)$$

  Cuando una transición t se dispara en un marcado M, el marcado resultante $M\,[t\rangle\,M'$ es tal que $\forall p \in P$

  $$\begin{array}{lcl} M(p) = \alpha & \Longrightarrow & M'(p) = \alpha \\ M(p) \neq \alpha & \Longrightarrow & M'(p) = M(p) - F(p,t) + F(t,p) \end{array}$$

- *Una* regla de modificación de la estructura $r \in R$ *está* habilitada *en el marcado M si:*

  $$\begin{array}{ll} \forall p \in P_1 & M(p) \neq \alpha \\ \forall p \in P_2 & M(p) = \alpha \end{array}$$

  El disparo de esta regla habilitada r produce el nuevo marcado M' definido como sigue:

  $$\begin{array}{ll} \forall p \in P_1 & M'(p) = \alpha \\ \forall p \in P_2 & M'(p) = \sum\{M(q) | q \in P_1 \wedge r(q) = p\} \\ \forall p \in P \backslash (P_1 \cup P_2) & M'(p) = M(p) \end{array}$$

Una red reconfigurable marcada *es una red reconfigurable junto con un marcado inicial.*

La política de disparos de transiciones es la misma que en la red de Petri obtenida descartando los lugares no existentes. Esta red de Petri se denomina *configuración* de la red reconfigurable. Siempre que ninguna regla de modificación de la estructura tenga lugar, la red reconfigurable se comporta exactamente igual que dicha red de Petri. Las reglas de modificación de la estructura producen cambios estructurales en la red eliminando lugares existentes y creando nuevos lugares, es decir, cambian el sistema de una configuración a otra. Cuando un lugar se elimina, los tokens de ese lugar no desaparecen sino que se mueven a otros lugares de la red. Así pues, el número de tokens permanece constante durante la aplicación de reglas de modificación de la estructura. La regla define cómo los tokens deben moverse por la red. Los lugares del conjunto P_2 que no están en el rango de r son lugares creados por la regla de modificación de la estructura e inicialmente no contienen tokens.

Una red reconfigurable puede verse como un conjunto de redes de Petri (sus configuraciones) que se corresponden con los distintos *modos* de operación del sistema. Las reglas de modificación de la estructura permiten cambiar de un modo de operación a otro mientras el marcado actual se modifica adecuadamente: los casos que fluyen por el sistema no se procesan durante el disparo de una regla de modificación de la estructura, por lo tanto, no hay creación ni desaparición de tokens, dichos tokens simplemente se desplazan de los lugares desaparecidos a los creados. Así pues, un sistema modelado mediante una red reconfigurable tiene la habilidad de cambiar dinámicamente su propia estructura cuando se dan ciertas condiciones. Por ejemplo, si el contenido de algún lugar es demasiado grande (hay una gran cantidad de casos esperando a ser procesados), se pueden duplicar las transiciones de salida de este lugar, técnicamente reemplazamos dicho lugar por uno nuevo teniendo duplicadas tantas transiciones de salida actuando como las transiciones de salida del lugar original.

Utilizando redes reconfigurables se pueden también implementar fácilmente los *cambios dinámicos aplazados* de [EKR95]. Estos cambios estructurales dinámicos también llamados *cambios synthetic cut-over* se definen como sigue, según [EKR95].

> *[en un cambio synthetic cut-over] ambas regiones de cambio, la nueva y la antigua, se mantienen en el nuevo procedimiento. Esto asegura que los tokens que ya estaban en la región de cambio antigua continuarán su progreso como si el cambio no hubiera tenido lugar inmediatamente (lo que justifica el atributo 'aplazado'). Sin embargo, los tokens que estaban evolucionando en el contexto de la región de cambio antigua nunca entrarán en la región de cambio antigua (y posiblemente región de cambio nueva); esto quiere decir que para estos tokens el cambio es inmediato.*

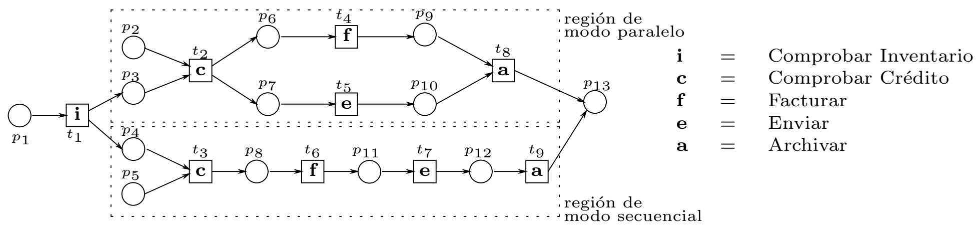


Figura 4.1: Cambio *synthetic cut-over*

Podemos ilustrar el cambio synthetic cut-over con el ejemplo de la Figura 4.1. Esta red reconfigurable describe cómo proceder con una orden de pedido de un cliente (Ejemplo 113). Hay dos modos de operación correspondientes a regiones distintas en la representación gráfica de la red, una en la que las operaciones de Facturación y Envío se procesan secuencialmente y otra en la que se procesan en paralelo. La regla de modificación de la estructura $r : \{p_4; p_2\} \rightarrow \{p_3; p_5\}$ dada por $r(p_4) = p_3$ y $r(p_2) = p_5$ permite cambiar de modo de operación secuencial a paralelo. A la inversa, la regla de modificación de la estructura r^{-1} realiza el cambio en el sentido contrario. La Figura 4.2 representa un fragmento del grafo de marcado de esta red reconfigurable, un lugar está representado gráficamente en un estado dado si y sólo si ese lugar existe en el marcado actual (es decir, su valor es distinto de α). Obsérvese que cuando se cambia de modo secuencial a paralelo, los tokens que están en la región antigua (región de modo secuencial) *continúan su progreso como si el cambio no hubiera ocurrido* pero los tokens en el contexto (en el lugar p_1) alcanzan ahora la región nueva (la región de modo paralelo).

El conjunto de lugares que existen en el marcado M, sea $D(M) = \{p \in P | M(p) \neq \alpha\}$, se denomina *dominio* de M. Dos marcados se dicen equivalentes cuando tienen el mismo dominio: $M_1 \equiv M_2 \iff D(M_1) = D(M_2)$. Un *modo de operación* es una clase de equivalencia para $\equiv$, puede identificarse con un subconjunto de lugares $D \subseteq P$. Normalmente, una red reconfigurable está implícitamente asociada a un subconjunto fijo de modos de operación. En el ejemplo anterior distinguimos dos modos de operación, el secuencial y el paralelo, cuyos respectivos dominios son $P \backslash \{p_3; p_5\}$ y $P \backslash \{p_2; p_4\}$; cualquier marcado cuyo dominio sea diferente de estos dos conjuntos, intuitivamente, no debería corresponder a ningún estado del sistema.

Además, como en [VDA97, VDAVH02], se considera que las redes que modelizan sistemas workflow tienen dos lugares distintivos, un *lugar de entrada i*

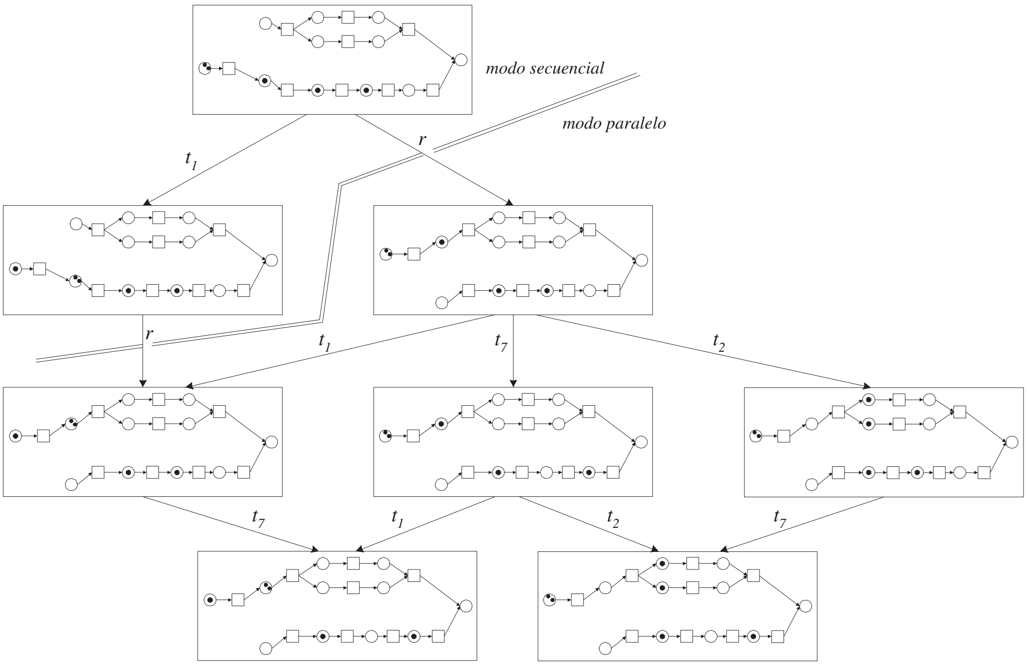


Figura 4.2: Fragmento del grafo de marcado de la red reconfigurable de la Figura 4.1

que es un lugar fuente, es decir, un lugar que no tiene transiciones de entrada: $\forall t \in T\ F(t,i) = 0$; y un *lugar de salida o* que es un lugar sumidero, es decir, un lugar que no tiene transiciones de salida $\forall t \in T\ F(o,t) = 0$. Estos lugares se corresponden, respectivamente, con el principio y el final del proceso de un caso. En el ejemplo de la Figura 4.1 el lugar de entrada es el lugar p_1 y el lugar de salida es el lugar p_{13}.

Redes Workflow Reconfigurables

Se presenta, a continuación, la definición de red workflow reconfigurable que es una adaptación de la noción de red workflow de van der Aalst [VDA97, VDAVH02].

Definición 115 (Red Workflow Reconfigurable) *Una* red workflow reconfigurable $\mathcal{N} = (N, \mathcal{O}, i, o)$ *es una red reconfigurable* $N = (P, T, F, R)$ *con un conjunto explícito de modos de operación* $\mathcal{O} \subseteq 2^P$ *y dos lugares distinguidos* $i, o \in P$ *donde* i *es un* lugar fuente *y* o *es un* lugar sumidero.

Todo modo de operación $\Omega \in \mathcal{O}$ *contiene lugares i y o; además, el conjunto*

$\mathcal{O}$ es fuertemente conectado en el sentido de que todo $\Omega \in \mathcal{O}$ deriva de otro $\Omega' \in \mathcal{O}$ por una secuencia (finita) de reglas de modificación de la estructura $r_1, \ldots, r_n$, donde Ω deriva desde Ω' a través de una regla $r : P_1 \rightarrow P_2$ cuando $\Omega = (\Omega' \backslash P_1) \cup P_2$.

Finalmente, si Ω y Ω' son subconjuntos de lugares tales que $\Omega = (\Omega' \backslash P_1) \cup P_2$ para alguna regla de modificación de la estructura $r : P_1 \rightarrow P_2$ entonces $\Omega \in \mathcal{O} \Longleftrightarrow \Omega' \in \mathcal{O}$.

El conjunto de modos de operación de una red workflow reconfigurable es una componente conectada del grafo dirigido cuyos vértices son subconjuntos de lugares y cuyos arcos son pares (Ω, Ω') tales que $\Omega = (\Omega' \backslash P_1) \cup P_2$ para alguna regla de modificación de la estructura $r : P_1 \rightarrow P_2$; y además esta componente está fuertemente conectada. Si $\Omega \in \mathcal{O}$ es un modo de operación, i_Ω representa el marcado del dominio Ω con un token en el lugar i y ningún otro token en ningún otro lugar y, de forma similar, o_Ω es el marcado del dominio Ω con un token en el lugar o y ningún otro token en ningún otro lugar. Si $U \subseteq E$ es un subconjunto de eventos (normalmente T, R o el propio E), se escribe $M\,[U\rangle\, M'$ si el marcado M' puede alcanzarse desde el marcado M disparando una secuencia de eventos de U. Debido a la conexión fuerte, cualquier modo particular de operación puede reestablecerse antes de empezar (o después de finalizar) el proceso de un caso: es decir, $i_\Omega\,[R\rangle\, i_{\Omega'}$ y $o_\Omega\,[R\rangle\, o_{\Omega'}$ para cualquier par $\Omega, \Omega' \in \mathcal{O}$ de modos de operación.

Un token en el lugar de entrada corresponde a un caso entrando en el sistema. Este caso *fluye* a través del sistema hasta que un token aparece en el lugar de salida indicando la terminación del caso. Mientras tanto, el rol que juega el marcado es doble. Por una parte, indica el estado actual (distribuido) de progreso del caso; por otra parte, indica el estado actual del sistema que procesa el caso. Sin pérdida de generalidad, se puede asumir que el estado del sistema está vacío cuando empieza un nuevo caso. El modo de operación, no obstante, puede haber cambiado pero esto no tiene importancia ya que un caso sería ignorado en el modo actual de operación del sistema. Finalmente, ya que un sistema está preparado para procesar casos indefinidamente, no debería tener un comportamiento degradado: si una transición llega a estar muerta en algún estado (es decir, si no puede ser disparada desde ese estado), entonces dicha transición debería descartarse en primer lugar. Tales restricciones son tenidas en cuenta en la definición siguiente que es una adaptación de la definición similar para redes workflow [VDA97, VDAVH02]. Se escribe $M \sqsubseteq M'$ cuando M y M' son marcados con el *mismo dominio* D tal que $\forall p \in D\ M(p) \leq M'(p)$

y $\mathcal{M}$ denota el conjunto de marcados.

Definición 116 (Red Workflow Reconfigurable Segura) *Una* red workflow reconfigurable *es* segura *si se cumplen las siguientes condiciones.*

1. *Todo proceso de un caso termina:*

$$\forall M \in \mathcal{M}\ \forall \Omega, \Omega' \in \mathcal{O}\ i_\Omega [E\rangle M \Longrightarrow$$
$$\exists M' \in \mathcal{M}\ \exists \Omega' \in \mathcal{O}\ (M [E\rangle M' \wedge o_{\Omega'} \sqsubseteq M')$$

2. *Toda terminación de un caso restablece el estado inicial del sistema (y posiblemente el modo de operación):*

$$\forall M \in \mathcal{M}\ \forall \Omega, \Omega' \in \mathcal{O}\ (i_\Omega [E\rangle M \wedge o_{\Omega'} \sqsubseteq M) \Longrightarrow M = o_{\Omega'}$$

3. *No existen transiciones muertas:*

$$\forall t \in T\ \forall \Omega \in \mathcal{O}\ \exists M, M' \in \mathcal{M}\ i_\Omega [E\rangle M \wedge M [t\rangle M'$$

Obsérvese que el proceso de un caso puede no terminar y que un cambio de modo de operación puede ser necesario para terminar. La definición anterior establece algunas propiedades del comportamiento esperado de un sistema workflow procesando un caso individual pero no dice nada del sistema cuando varios casos están procesándose concurrentemente. En las aplicaciones reales todo caso tiene una *identidad*, un agente que realiza una tarea dentro de un sistema workflow conoce que caso está siendo procesado actualmente. Se puede considerar que a cualquier caso nuevo que entra en el sistema se le da un *color* y que ese color es distinto de los colores de otros casos que estén fluyendo por el sistema en ese momento. La red de Petri coloreada resultante se comporta como sigue: para dispararse, a una transición sólo se le permite "tomar" de sus lugares de entrada tokens del mismo color y entonces produce tokens de ese mismo color en sus lugares de salida. Esto significa que el proceso concurrente de múltiples casos se representa como la superimposición no interfiriente de los procesos de casos individuales. Una red workflow reconfigurable se considera sólo con respecto a un caso individual, lo que justifica la definición anterior. Una forma débil de interferencia entre casos existe, sin embargo, debido al hecho que el estado actual del sistema puede tener un efecto de decisión como si una regla de modificación de la estructura permitida fuera invocada. Por ejemplo, si el número de tokens de un lugar determinado excede un límite dado, se puede regular el flujo invocando alguna regla de modificación de la estructura, la modificación contraria puede invocarse después cuando el contenido de dicho lugar esté bajo otro límite; en este sentido, la manera en que un caso se procesa puede estar influenciada por otros casos. Nótese que tales límites no aparecen en la definición de reglas de modificación de la estructura, lo que refleja el hecho de que las decisiones que conciernen a la invocación de estas reglas son externas al sistema. Además, esta interferencia concierne solamente a los modos de operación que se están utilizando cuando se procesa un caso y el formalismo permite al diseñador del sistema asegurar que el proceso de un caso no es sensible a los cambios dinámicos que ocurren dentro del sistema. Se puede enriquecer, si es necesario, la descripción de la red añadiendo lugares y transiciones extra para asegurar que el proceso de los casos sea independiente de los cambios dinámicos. Por ejemplo, se puede identificar una lista de propiedades que caracterizan el hecho de que un caso se ha procesado correctamente. En general, tal propiedad corresponde a la finalización de una tarea. Estas propiedades se representan mediante lugares extra $p_{n+1}, \ldots, p_{n+k}$. Otro lugar extra se introduce como el nuevo lugar de salida de la red enriquecida. Este lugar es “filled” por una transición extra cuyas precondiciones son los lugares $p_{n+1}, \ldots, p_{n+k}$ junto con el antiguo lugar de salida. De esta forma, un token en el nuevo lugar de salida indica que el caso ha sido procesado correctamente sin tener en cuenta los cambios dinámicos que hayan ocurrido.

4.1.3. Acotamiento de una Red Reconfigurable

El árbol de alcanzabilidad de una red reconfigurable marcada es el árbol cuya raíz está etiquetada con el marcado inicial y tal que si V es un vértice cualquiera de este árbol etiquetado con marcado M, los arcos que tienen origen en V están en correspondencia biyectiva con los disparos $M\,[e\rangle\,M'$, el arco asociado con $M\,[e\rangle\,M'$ está etiquetado con el evento e y su fin está etiquetado con M'. El árbol de alcanzabilidad es, por tanto, el “desplegado” del grafo de marcado de la red marcada. Si la red no está acotada, este árbol es infinito. Igual que para las redes de Petri ordinarias, puede construirse una aproximación finita del árbol de alcanzabilidad denominada *árbol de cobertura*. Dos propiedades son la base del algoritmo de Karp y Miller [KM69]. Se corresponden con las dos proposiciones que siguen.

Proposición 117 *La relación de orden entre marcados es un buen orden.*

Demostración:

Recordemos (ver e.g. [Die97]) que una relación de orden $(X, \leq)$ es un buen orden si para cualquier secuencia infinita $(x_i, i \in \mathbb{N})$ pueden encontrarse índices $i < j$ tales que $x_i \leq x_j$, lo que es equivalente a decir que toda secuencia en X tiene una subsecuencia creciente infinita. El orden usual sobre $\mathbb{N}$ es un buen orden. Además, si extraemos subsecuencias iterativamente (n veces), notaremos que si $(X, \leq)$ es un buen orden, entonces X^n con el orden punto fijo es también un buen orden. Finalmente, $\mathbb{N} \cup \{\alpha\}$ con el orden $x \leq y \Longleftrightarrow (x = y = \alpha \vee [x, y \in \mathbb{N} \wedge x \leq y])$ es un buen orden. Efectivamente, si $(x_i, i \in \mathbb{N})$ es una secuencia en $\mathbb{N} \cup \{\alpha\}$, entonces podemos extraer una subsecuencia que es constantemente igual a α o una secuencia creciente de enteros según tengamos un número infinito de índices $i \in \mathbb{N}$ tales que $x_i = \alpha$ o $x_i \in \mathbb{N}$. Por tanto, el orden entre marcados es un buen orden.

■

Proposición 118 *La regla de disparo es monótona:*

$$(M_1 \, [e\rangle \, M_2 \wedge M_1 \sqsubseteq M_1') \Longrightarrow \exists M_2'(M_1' \, [e\rangle \, M_2' \wedge M_2 \sqsubseteq M_2')$$

Además, $|M_1'| - |M_1| = |M_2'| - |M_2|$.

Así pues, si $M \, [u\rangle \, M'$ *con* $M \sqsubseteq M'$ *entonces la secuencia de disparos* $u \in E^*$ *puede reproducirse, es decir,* $\exists M \, [u^n\rangle \, M^{(n)}$ *para todo* $n \in \mathbb{N}$, *y entonces* $|M^{(n)}| = |M| + kn$ *siendo* $k = |M'| - |M|$.

Demostración:

- Si $e = t \in T$, entonces $(M_1 \, [t\rangle \, M_2 \wedge M_1 \sqsubseteq M_1') \Longrightarrow \exists M_2'(M_1' \, [t\rangle \, M_2' \wedge M_2 \sqsubseteq M_2')$ y $M_1' - M_1 = M_2' - M_2$ (*propiedad de efecto constante*) igual que se cumple para las redes de Petri.
- Si $e = r \in R$, la primera propiedad es trivial mientras que la propiedad del efecto constante se debilita: como los tokens se mueven de unos lugares a otros, sólo se conserva el número total de tokens, es decir, $|M_1'| - |M_1| = |M_2'| - |M_2|$.

■

Algoritmo 3 Árbol de Cobertura de una Red Reconfigurable Marcada

- Inicialmente, el árbol se reduce a su raíz etiquetada M_0 y señalada como un vértice "nuevo".
- Mientras existan vértices "nuevos" hacer:
    - Seleccionar un nuevo vértice V, sea M su etiqueta.
    - Para todo disparo $M\,[e\rangle\,M'$ hacer:
        - Crear un nuevo vértice V' etiquetado M' y un arco desde V a V' etiquetado con e.
        - Si existe algún nodo V'' en el camino desde la raíz al vértice V cuya etiqueta M'' es tal que $M'' \sqsubseteq M'$
entonces
    Si $M'' = M'$
    entonces señalar el vértice V' como "viejo"
    sino señalarlo como "no acotado".
sino señalar V' como "nuevo".
    - Retirar V del conjunto de vértices "nuevos".

La observación clave para el algoritmo de Karp y Miller para redes de Petri es tal que, a causa de la propiedad de efecto constante, cuando se encuentran dos marcados M y M' y una secuencia de disparos $u \in T^*$ tales que $M\,[u\rangle\,M'$ y $M \sqsubseteq M'$, podemos deducir $M\,[u^n\rangle\,M^{(n)}$ para todo $n \in \mathbb{N}$. Con lo que para todo lugar $p \in P$ tal que $M(p) < M'(p)$, se tiene que $M^{(n)} = M(p) + nk$ siendo $k = M'(p) - M(p) > 0$ y, por tanto, este lugar no está acotado. Debido a la forma débil de la propiedad de efecto constante, esta observación no se da para las redes reconfigurables. Sin embargo, si no nos interesa el acotamiento de un lugar en particular de la red sino el acotamiento de toda la red (es decir, si existe algún lugar en la red que no esté acotado), entonces las proposiciones anteriores son suficientes y el acotamiento puede verificarse utilizando la siguiente versión simplificada del árbol de cobertura. El árbol de cobertura de una red reconfigurable marcada (N, M_0) se construye mediante el Algoritmo 3.

Proposición 119 *El árbol de cobertura de una red reconfigurable marcada es finito.*

Demostración:

Puesto que la relación de orden en el conjunto de marcados es un buen orden, el árbol de cobertura de una red reconfigurable marcada no tiene ramas infinitas. Puesto que además, cada vértice tiene como mucho $|E|$ sucesores, por el lema de König [Reu89] se deduce que el árbol es finito.

∎

Proposición 120 *Una red reconfigurable marcada está acotada si y solamente si ningún vértice de su árbol de cobertura se señala como "no acotado".*

Demostración:

- Si el árbol de cobertura no contiene ningún vértice señalado como "no acotado" entonces el conjunto de etiquetas de sus vértices coincide con el conjunto de marcados de la red reconfigurable alcanzables desde el estado inicial (etiqueta de la raíz), por lo tanto, la red reconfigurable marcada está acotada.

- Si, por el contrario, el árbol de cobertura contiene algún vértice V' señalado "no acotado", en ese caso, existe algún vértice V en el camino desde la raíz tal que $M_0\,[u\rangle\, M\,[v\rangle\, M'$ donde u etiqueta el camino desde la raíz al vértice V, v etiqueta el camino desde el vértice V al vértice V', $M \sqsubseteq M'$ y $M \neq M'$. Entonces, por la Proposición 118, $M\,[v^n\rangle\, M^{(n)}$ con $\left|M^{(n)}\right| = |M| + kn$ siendo $k = |M'| - |M| > 0$. Ya que existe un número finito de lugares, la red no está acotada.

∎

Corolario 121 *El acotamiento de una red reconfigurable es decidible.*

Acotamiento y seguridad de una red workflow reconfigurable

Una red workflow reconfigurable N se dice que está acotada si la red reconfigurable marcada (N, i_Ω) para algún modo de operación $\Omega \in \mathcal{O}$ está acotada. Esta definición no depende de la elección de $\Omega \in \mathcal{O}$ ya que $i_\Omega\,[R\rangle\, i_{\Omega'}$ para todo par $\Omega, \Omega' \in \mathcal{O}$.

Proposición 122 *Una red workflow reconfigurable segura está acotada y se puede decidir si una red workflow reconfigurable es segura.*

Demostración:

Si (N, i_Ω) no está acotada entonces como se puede ver en la demostración de la Proposición 120, existen marcados M y M' tales que $i_\Omega [E\rangle M$, $M [E\rangle M'$, $M \sqsubseteq M'$ y $M(p) < M(p')$ para algún lugar p. Ya que N es segura se tiene que $M [u\rangle o_{\Omega'}$ para alguna secuencia $u \in E^*$. Por monotonía, $M' [u\rangle M''$ con $o_{\Omega'} \sqsubseteq M''$ y $M'' \neq o_{\Omega'}$ lo que contradice el hecho de que N es segura. Una vez comprobado el acotamiento, las propiedades 1. a 3. de la Definición 116 pueden comprobarse directamente en el árbol de cobertura que coincide con el árbol de alcanzabilidad.

■

Sin embargo, como subraya Hack en [Hac74]:

> "*La talla de la construcción de Karp y Miller en su procedimiento de decisión para acotamiento y cobertura puede crecer tan rápido como la función de Ackermann en la talla de la red de Petri*"

lo que demuestra la intratabilidad de la verificación de la propiedad de seguridad vía la construcción del árbol de cobertura.

Van der Aalst demostró en [VDA97] que la seguridad de una red workflow se reduce al acotamiento y la vivacidad de una red de Petri obtenida añadiendo una transición extra conectando su lugar de salida a su lugar de entrada. Puesto que es posible decidir acotamiento y vivacidad de las redes de Petri sin elección en tiempo polinómico, [DE95] dedujo que la seguridad de las redes workflow sin elección también puede decidirse en tiempo polinómico. Se demuestra que la construcción de Van der Aalst puede trasladarse a las redes reconfigurables sin cambios significativos, desafortunadamente, de ahí no se puede deducir directamente un algoritmo polinómico para la decisión de la seguridad de las redes workflow reconfigurables cuyas configuraciones son todas redes de Petri sin elección.

Definición 123 (Red Workflow Reconfigurable Marcada) *Si* $\mathcal{N} = (N, \mathcal{O}, i, o)$ *es una red workflow reconfigurable donde* $N = (P, T, F, R)$ *y* $\Omega \in \mathcal{O}$ *es un modo de operación,* $\overline{\mathcal{N}}_\Omega = (\overline{N}, i_\Omega)$ *es la* red workflow reconfigurable marcada *formada por la red reconfigurable* $\overline{N} = (P, \overline{T}, \overline{F}, R)$ *y por un marcado inicial* i_Ω*, siendo* $\overline{T} = T \cup \{t^*\}$ *con* $t^* \notin T$ *una nueva transición y la relación de flujo extendida* $\overline{F} : (P \times \overline{T}) \cup (\overline{T} \times P) \to \mathbb{N}$ *viene dada por*

$$\overline{F}(t,p) = \begin{cases} F(t,p) & \text{si } t \in T \text{ y } p \in P \\ -1 & \text{si } t = t^* \text{ y } p = o \\ 1 & \text{si } t = t^* \text{ y } p = i \\ 0 & \text{en otro caso} \end{cases}$$

$\overline{N}$ se obtiene a partir de N añadiendo una nueva transición t^* que se habilita cuando un caso ha alcanzado la terminación (hay un token en el lugar de salida) y entonces se elimina ese caso del sistema y se introduce uno nuevo (añadiendo un token en el lugar de entrada).

Proposición 124 *La red workflow reconfigurable* $\mathcal{N} = (N, \mathcal{O}, i, o)$ *es segura si y solamente si la red reconfigurable* $\overline{\mathcal{N}}_\Omega$ *está viva y acotada.*

Demostración:

Primero nótese que ya que cada estado inicial i_Ω es alcanzable desde cualquier otro estado mediante reglas de modificación de la estructura ($i_\Omega\,[R\rangle\, i_{\Omega'}$), la red reconfigurable $\overline{\mathcal{N}}_\Omega$ está viva y acotada si y solamente si $\overline{\mathcal{N}}_{\Omega'}$ está viva y acotada para algún $\Omega' \in \mathcal{O}$.

$\Longrightarrow$ Se demuestra en primer lugar que si $\overline{\mathcal{N}}_{\Omega'}$ está viva y acotada entonces $\mathcal{N}$ es una red workflow reconfigurable segura. Puesto que $\overline{\mathcal{N}}_\Omega$ está viva, la trasición t^* es potencialmente disparable en todo marcado alcanzable, es decir, la condición 1. de la Definición 116 se satisface. Si $\Omega, \Omega' \in \mathcal{O}$, $<\Omega, \Omega'>$ denota el conjunto de enteros $n \in \mathbb{N}$ para los que existe algún marcado M tal que $i_\Omega\,[E\rangle \cup \{t^*\} > M$, $i_{\Omega'} \sqsubseteq M$ y $|M| = n + 1$. Es decir, por la Proposición 118, que $<\Omega, \Omega'>$ registra todos los posibles incrementos de la talla de los marcados a través de las computaciones en $\overline{\mathcal{N}}_\Omega$ desde algún marcado mayor que i_Ω a algún marcado mayor que $i_{\Omega'}$. Así pues, $(n \in <\Omega, \Omega'> \wedge\, m \in <\Omega', \Omega''>) \Longrightarrow n + m \in <\Omega, \Omega''>$. Ahora $<\Omega, \Omega'> \neq \emptyset$. En realidad, ya que la condición 1. de la Definición 116 se satisface, $i_\Omega\,[E\rangle\, M$ para algún M tal que $o_{\Omega''} \sqsubseteq M$ para algún $\Omega'' \in \mathcal{O}$; puesto que $o_{\Omega''}\,[R\rangle\, o_{\Omega'}$ y por la Proposición 118 se deduce $M\,[R\rangle\, M'$ con $o_{\Omega'} \sqsubseteq M'$ y entonces $M'\,[t^*\rangle\, M''$ con $i_{\Omega'} \sqsubseteq M''$ como se requiere. Por tanto, ya que $\overline{\mathcal{N}}_\Omega$ está acotada, se deduce que $<\Omega, \Omega'>= \{0\}$ para todo

$\Omega, \Omega' \in \mathcal{O}$, y así $(i_\Omega\,[E\rangle\, M \wedge o_{\Omega'} \sqsubseteq M) \Longrightarrow M = o_{\Omega'}$, es decir, la condición 2. de la Definición 116 se satisface. La condición 3. de la Definición 116 se sigue del hecho que $\overline{\mathcal{N}}_\Omega$ está viva para todo $\Omega \in \mathcal{O}$.

$\Longleftarrow$ A la inversa, se asume que $\mathcal{N}$ es segura.

1. En primer lugar se demuestra que $\overline{\mathcal{N}}_\Omega$ está acotada. Ya que $\mathcal{N}$ es segura, la red extendida $\overline{\mathcal{N}}$ vuelve a algún estado inicial i_Ω cuando t^* se dispara, es entonces suficiente comprobar que la red reconfigurable marcada (N, i_Ω) está acotada para todo $\Omega \in \mathcal{O}$. Si este no es el caso, entonces, por la construcción del árbol de cobertura, se deduce que existen los marcados M_1 y M_2 tales que $i_\Omega\,[E\rangle\, M_1$, $M_1\,[E\rangle\, M_2$, $M_1 \sqsubseteq M_2$ y $M_1 \neq M_2$. Por la seguridad de $\mathcal{N}$, se deduce $M_1\,[u\rangle\, o_{\Omega'}$ para algún $u \in E^*$ y por la Proposición 118 $M_2\,[E\rangle\, M_2'$ con $o_{\Omega'} \sqsubseteq M_2'$ y $M_2' \neq o_{\Omega'}$ (puesto que $|M_2'| - 1 = |M_2| - |M_1| > 0$) lo que contradice la seguridad de $\mathcal{N}$.
2. En segundo lugar, se demuestra que $\overline{\mathcal{N}}_\Omega$ está viva. Al ser $\mathcal{N}$ segura, la transición t^* es potencialmente disparable en cualquier marcado alcanzable y sus disparos siempre llevan a algún estado inicial i_Ω; además, puesto que $i_\Omega\,[R\rangle\, i_{\Omega'}$ para un par arbitrario de modos de operación, se deduce que la red $\overline{\mathcal{N}}_\Omega$ es cíclica (todo estado inicial i_Ω y, por tanto, cualquier marcado alcanzable es alcanzable desde cualquier marcado alcanzable). Ya que además no existen transiciones muertas en $\overline{\mathcal{N}}_\Omega$ (por la condición 3. de la Definición 116 y por el hecho de que t^* no es una transición muerta), se deduce que esta red marcada está viva.

■

4.1.4. Redes Reconfigurables como Redes Automodificantes

El propósito de esta sección es demostrar que las redes reconfigurables son redes automodificantes. Las redes automodificantes [Val78a, Val78b, Val81] son generalizaciones de las redes Lugar/Transición en las que la relación de flujo entre un lugar y una transición depende del marcado. Recordamos, a continuación, la definición de Red Automodificante ya introducida en el Capítulo 1.

Definición 125 (Red Automodificante) *Una* red automodificante *es una estructura $N = (P, T, F)$ en la que $P = \{p_1, \ldots, p_m\}$ es un conjunto no vacío y finito de lugares, $T = \{t_1, \ldots, t_n\}$ es un conjunto no vacío y finito de transiciones disjunto de P y $F : (P \times T) \cup (T \times P) \to \mathbb{N}^{P*}$ es la relación de flujo donde $P_* = P \cup \{*\}$ y $* \notin P$.*

Un vector $\varphi \in \mathbb{N}^{P*}$ puede representarse mediante una suma formal $\varphi = \lambda_0 + \sum_{i=1}^{m} \lambda_i \cdot p_i$ donde el coeficiente constante es la entrada correspondiente al lugar ficticio: $\lambda_0 = \varphi(*)$ y $\lambda_i = \varphi(p_i)$.

Un marcado *de la red N es una aplicación $M : P \to \mathbb{N}$. Si $M \in \mathbb{N}^P$ es un marcado y $\varphi \in \mathbb{N}^{P*}$, $\varphi(M) = \lambda_0 + \sum_{i=1}^{m} \lambda_i \cdot M(p_i)$ denota la evaluación de la función afín φ en el marcado M.*

Representamos por $M\,[t\rangle\, M'$ cuando la transición t está habilitada *en el marcado M y conduce al marcado M'. Esta relación de transición viene dada por:*

$$M\,[t\rangle\, M' \iff \forall p \in P \; M(p) \geq F(p,t)(M) \wedge M' = M - F(p,t)(M) + F(t,p)(M)$$

Una red automodificante marcada *es una red automodificante junto con un marcado inicial.*

Proposición 126 *Cualquier red reconfigurable marcada puede asociarse con una red automodificante marcada con grafo de marcado isomorfo y cuyo conjunto de transiciones es el conjunto de eventos de la red reconfigurable.*

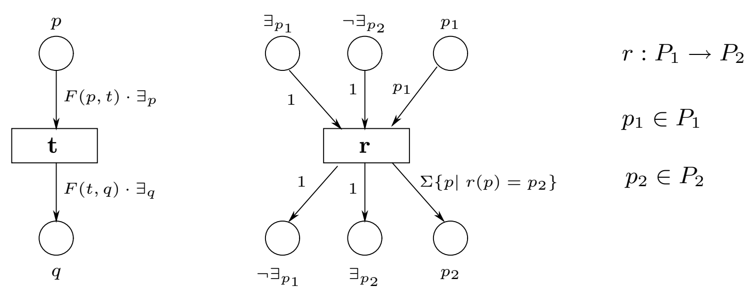


Figura 4.3: Traducción de una red reconfigurable en una red automodificante equivalente

Demostración:

La traducción de una red reconfigurable en una red automodificante equivalente es directa: representamos cada lugar p de la red reconfigurable mediante tres lugares $\exists p$, $\neg\exists p$ y p. Los dos primeros lugares son lugares complementarios 1-acotados, cuyo contenido indica si el lugar p existe en el marcado actual y el tercer lugar, denotado por p, tiene el mismo contenido que el lugar p original cuando dicho lugar existe. En la Figura 4.3 se muestra la representación de la traducción.

Cualquier red reconfigurable $N = (P, T, F, R)$ se asocia con una red automodificante $\widetilde{N} = (\widetilde{P}, \widetilde{T}, \widetilde{F})$ definida como sigue:

- El conjunto de lugares $\widetilde{P} = \exists P \cup \neg\exists P \cup P$ es la unión disjunta de tres copias del conjunto P cuyos elementos típicos se denotan, respectivamente, por $\exists p$, $\neg\exists p$ y p para p en el rango de P.
- El conjunto de transiciones $\widetilde{T} = T \cup R$ está formado por las transiciones del conjunto original junto con su conjunto de reglas de modificación de la estructura, es decir, su conjunto de eventos.
- Finalmente, la relación de flujo $\widetilde{F}$ viene dada por las siguientes identidades donde $\widetilde{p}$, t y $r : P_1 \rightarrow P_2$ toman valores en $\widetilde{P}$, T y R, respectivamente.

$$\widetilde{F}(\widetilde{p}, t) = \begin{cases} F(p,t) \cdot \exists p & \text{si } \widetilde{p} = p \in P \\ 0 & \text{en otro caso} \end{cases}$$

$$\widetilde{F}(t, \widetilde{p}) = \begin{cases} F(t,p) \cdot \exists p & \text{si } \widetilde{p} = p \in P \\ 0 & \text{en otro caso} \end{cases}$$

$$\widetilde{F}(\widetilde{p}, r) = \begin{cases} 1 & \begin{array}{l} \text{si } \widetilde{p} = \exists p_1 \wedge p_1 \in P_1 \\ \text{o } \widetilde{p} = \neg\exists p_2 \wedge p_2 \in P_2 \end{array} \\ p_1 & \text{si } \widetilde{p} = p_1 \in P_1 \\ 0 & \text{en otro caso} \end{cases}$$

$$\widetilde{F}(r, \widetilde{p}) = \begin{cases} 1 & \begin{array}{l} \text{si } \widetilde{p} = \neg\exists p_1 \wedge p_1 \in P_1 \\ \text{o } \widetilde{p} = \exists p_2 \wedge p_2 \in P_2 \end{array} \\ \sum\{p \mid r(p) = p_2 & \text{si } \widetilde{p} = p_2 \in P_2 \\ 0 & \text{en otro caso} \end{cases}$$

Un marcado M de una red reconfigurable N se asocia con el marcado $\widetilde{M}$ de la red automodificante $\widetilde{N}$ dado por

$$\begin{array}{lcl} \widetilde{M}(\exists p) & = & \text{si } M(p) \neq \alpha \text{ entonces } 1 \text{ sino } 0 \\ \widetilde{M}(\neg\exists p) & = & \text{si } M(p) \neq \alpha \text{ entonces } 0 \text{ sino } 1 \\ \widetilde{M}(p) & = & \text{si } M(p) \neq \alpha \text{ entonces } M(p) \text{ sino } 0 \end{array}$$

Las relaciones anteriores inducen una correspondencia biyectiva entre los marcados de N y aquellos marcados $\widetilde{M}$ de $\widetilde{N}$ tales que $\forall p \in P$ $\widetilde{M}(\exists p)$, $\widetilde{M}(\neg\exists p) \in \{0; 1\}$ y $\widetilde{M}(\exists p) = 1 \Longleftrightarrow \widetilde{M}(\neg\exists p) = 0$ y $\widetilde{M}(\exists p) = 0 \Longrightarrow \widetilde{M}(p) = 0$.

Una comparación directa de la Definición 114 y de la Definición 125 muestra que un evento $e \in T \cup R$ de N está habilitado en un marcado M de la red reconfigurable N si y solamente si una transición de la red automodificante $\widetilde{N}$ está habilitada en el marcado asociado $\widetilde{M}$; por otra parte, $M\,[e\rangle\, M'$ en N si y sólo $\widetilde{M}\,[e\rangle\, \widetilde{M'}$ en $\widetilde{N}$.

■

El acotamiento no es decidible para las redes automodificantes (como hemos visto en el Capítulo 1) mientras que sí que lo es para las redes reconfigurables. Las redes reconfigurables son, por tanto, una subclase de las redes automodificantes para las que el acotamiento puede decidirse.

4.2. Redes Reconfigurables Reversibles

En la Sección 4.1 se han presentado las redes reconfigurables como una clase de redes de Petri de alto nivel que se basan en la reescritura de alguno de sus componentes para modificar dinámicamente su propia estructura. En esta sección nos vamos a centrar en una subclase de las redes reconfigurables, las llamadas redes reconfigurables reversibles, en las que las reglas de modificación de la estructura pueden aplicarse tanto en un sentido como en el sentido contrario. Además, estas redes pueden verse como la composición en cascada de un autómata con una red de Petri paramétrica; y, bajo alguna asunción adicional, son equivalentes a las redes de Petri estratificadas. Desde un punto de vista práctico, la elección de un modelo particular (redes de Petri, redes automodificantes, redes de Petri estratificadas, redes reconfigurables) depende de la naturaleza del problema a modelizar.

En la Subsección 4.2.1 se introduce el modelo de las *Redes de Petri Controladas por un Autómata* (RPCA), para describir sistemas consistentes en un conjunto de redes de Petri, o configuraciones, junto con algún mecanismo de cambio de una configuración a otra. Se demuestra que cualquier RPCA puede asociarse con algún *Sistema de Adición de Vectores con Estados* (SAVE) equivalente y que, por tanto, el acotamiento de un lugar, la alcanzabilidad, el interbloqueo y la vivacidad son propiedades decidibles para las RPCA. En la Subsección 4.2.2 se considera la subclase de las *Redes Reconfigurables Reversibles* para las que se establece una propiedad fundamental. Esta propiedad se utiliza en la Subsección 4.2.3 para demostrar que el grafo de marcado de una red reconfigurable reversible es equivalente al grafo de marcado de una RPCA. Así pues, el acotamiento de un lugar, la alcanzabilidad, el interbloqueo y la vivacidad son también propiedades decidibles para las redes reconfigurables reversibles. Finalmente, la propiedad fundamental de las redes reconfigurables reversibles también se utiliza en la Subsección 4.2.4 para traducir las redes reconfigurables reversibles a redes de Petri estratificadas.

4.2.1. Redes de Petri Controladas por un Autómata

En esta sección se introduce la noción de Red de Petri Controlada por un Autómata (RPCA). Se muestra una equivalencia débil entre las RPCAs y los Sistemas de Adición de Vectores con Estados (SAVE [Reu89]) que implica que el acotamiento de un lugar, la alcanzabilidad, el interbloqueo y la vivacidad sean propiedades decidibles para las RPCAs. En particular, se da una definición del árbol de cobertura de una RPCA que es homóloga a la del árbol de cobertura del SAVE asociado.

Definición 127 (Red de Petri Paramétrica) *Una* red de Petri paramétrica *es una estructura* $(\Pi, P, E, Pre, Post, M_0)$ *siendo:*

- Π*, P y E conjuntos finitos disjuntos dos a dos cuyos elementos se denominan, respectivamente,* parámetros*,* lugares *y* eventos*.*
- $Pre, Post : P \times E \to \mathbb{N}^\Pi$ *son la* matriz preset *y* postset*, respectivamente.*
- $M_0 \in \mathbb{N}^P$ *es el* marcado inicial*.*

Si $^\bullet\Lambda, \Lambda^\bullet : \mathbb{N}^\Pi \to (P \times E \to \mathbb{N})$ *vienen dados por* $^\bullet\Lambda(\alpha)(p,e) = Pre(p,e) \cdot \alpha$ *y* $\Lambda^\bullet(\alpha)(p,e) = Post(p,e) \cdot \alpha$ *para* $\alpha \in \mathbb{N}^\Pi$*,* $e \in E$ *y* $p \in P$*, entonces una red de Petri paramétrica forma parte de una familia de redes de Petri* $N_\alpha = (P, E, ^\bullet\Lambda(\alpha), \Lambda^\bullet(\alpha))$ *indexadas por* $\alpha \in \mathbb{N}^\Pi$*.*

La relación de disparo *en N_α viene dada por*

$$M\,[e/\alpha\rangle\, M' \Leftrightarrow \forall p \in P \begin{cases} (i) & M(p) \geq Pre(p,e)\cdot\alpha \\ (ii) & M'(p) = M(p) - Pre(p,e)\cdot\alpha + Post(p,e)\cdot\alpha \end{cases}$$

Definición 128 (Red de Petri Controlada por un Autómata (RPCA)) *Una* red de Petri controlada por un autómata *es una tripleta $\mathcal{N} = (A, \Delta, N)$ en la que:*

- $A = (S, E, T, s_0)$ *es un autómata determinista finito con un conjunto de estados S, un conjunto de eventos E, una relación de transición $T \subseteq S \times E \times S$ y un estado inicial $s_0 \in S$.*
- $N = (\Pi, P, E, Pre, Post, M_0)$ *es una red de Petri paramétrica con el mismo conjunto de eventos que el autómata.*
- $\Delta : S \to \mathbb{N}^\Pi$ *es llamada función de codificación de estados y da el interfaz entre la parte de control (el autómata) y la parte controlada (la red de Petri paramétrica).*

Realmente, N puede construirse como la familia de redes N_α controladas por el autómata A.

Los marcados *de $\mathcal{N}$ son los pares $(M, s) \in \mathbb{N}^P \times S$ y su* relación de transición entre marcados *viene dada por*

$$(M, s) \xrightarrow{e} (M', s') \Longleftrightarrow s \xrightarrow{e} s' \wedge M\,[e/\Delta(s)\rangle\, M'$$

Su grafo de marcado *es el autómata, denotado por $N \rtimes_\Delta A$, obtenido restringiendo su relación de transición al conjunto de marcados alcanzables desde el marcado inicial (M_0, s_0).*

Una red de Petri controlada por un autómata consiste pues en dos sistemas compuestos en cascada: un autómata A y una red de Petri paramétrica N. El interfaz entre los dos sistemas viene dado por la función de codificación de estados que puede representarse en forma de una matriz $\Delta : S \rtimes \Pi \to \mathbb{N}$. La expresión $N \rtimes_\Delta A$ se abrevia por $N \rtimes A$ cuando Δ es una codificación de estados *trivial*, es decir, cuando $S = \Pi$ y Δ es la matriz identidad. Las redes de Petri controladas por autómata son muy similares a los Sistemas de Adición de Vectores con Estados.

Definición 129 (Sistema de Adición de Vectores con Estados (SAVE)) *Un* sistema de adición de vectores con estados $(V, Arcs, l)$ *[Reu89] está formado por un grafo orientado finito $G = (V, Arcs)$, un entero $m \geq 1$ y una aplicación $l : Arcs \to \mathbb{Z}^m$, es decir, un SAVE es un grafo orientado etiquetado en $\mathbb{Z}^m$.*

Un marcado *es un par (M, v) formado por un vector con entradas no negativas $M \in \mathbb{N}^m$ y un vértice $v \in V$.*

La notación $(M, v) \xrightarrow{a} (M', v')$ significa que existe un arco desde v a v' y $M' = M + l(a)$.

El marcado (M', v') se dice que es alcanzable *desde el marcado (M, v) si $(M, v) \xrightarrow{*} (M', v')$ siendo $\xrightarrow{*}$ el cierre transitivo y reflexivo de la relación de alcanzabilidad en un paso dada por: $(M, v) \to (M', v') \iff \exists a \in Arcs\ (M, v) \xrightarrow{a} (M', v')$.*

Si el SAVE dispone de un marcado inicial (M_0, v_0), su grafo de marcado *es el grafo cuyos arcos son las tripletas $(M, v) \xrightarrow{a} (M', v')$ donde (M, v) (y, por tanto, (M', v')) son marcados alcanzables desde el marcado inicial.*

Claramente, una RPCA $\mathcal{N} = (A, \Delta, N)$ donde N es una red de Petri *pura* es equivalente al SAVE $\mathcal{N}^\circ = (V, Arcs, l)$ cuyos vértices son los estados de A: $V = S$ y se tiene un arco en $Arcs$ desde s a s' etiquetado por el vector $V(e, s) \in \mathbb{Z}^n$ dado por

$$V(e, s)(i) = [Post(p_i, e) - Pre(p_i, e)] \cdot \Delta(s)$$

siendo $P = \{p_1, \ldots, p_n\}$ una enumeración fija del conjunto de lugares de N. En realidad, $\mathcal{N}$ y $\mathcal{N}^\circ$ tienen los mismos marcados y $(M, s) \xrightarrow{e} (M', s')$ aparece en el grafo de marcado de $\mathcal{N}$ si y sólo si $(M, s) \xrightarrow{V(e,s)} (M', s')$ aparece en el grafo de marcado de $\mathcal{N}^\circ$.

Suponiendo que la red N sea impura, lo que significa que tanto $Pre(p, e) \cdot \Delta(s)$ como $Post(p, e) \cdot \Delta(s)$ son distintos de cero para algún lugar p, evento e y estado s, la configuración de N asociada con el estado s ($N_{\Delta(s)}$) es una red de Petri impura. Así podemos transformar fácilmente cualquier red de Petri en una red de Petri pura dividiendo sus eventos. Esta transformación que se ilustra a la izquierda de la Figura 4.4 consiste en dividir cada evento e en dos eventos, el "comienzo de e" denotado por e^+ y el "fin de e" denotado por e^-. Todo lugar de entrada del evento e en la red original es un lugar de

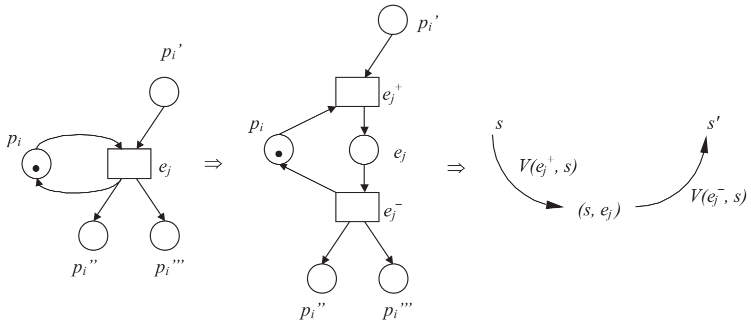


Figura 4.4: Desde una red impura a una red pura

entrada de e^+ con el mismo peso. De forma similar, todo lugar de salida del evento e en la red original es un lugar de salida de e^- con el mismo peso. Finalmente, en la red dividida tenemos un lugar extra por cada evento de la red original (con el mismo nombre de dicho evento) que es un lugar de salida para e^+ y un lugar de entrada para e^- con pesos únicos, su contenido indica el número de instancias del evento e que están siendo ejecutadas actualmente. La red dividida da una descripción más fina del fenómeno modelizado en el que romper la atomicidad de los eventos permite una descripción explícita de la coocurrencia de eventos. Sin embargo, si no se está interesado en la independencia de eventos, que es el caso para el estudio de propiedades como alcanzabilidad, acotamiento, vivacidad o interbloqueo, entonces la red dividida es equivalente a la red original (ver [Bad96]).

Podemos pensar en aplicar esta operación a cada configuración de N y utilizar la construcción anterior para RPCA puras. Sin embargo, la atomicidad de eventos requiere que todo disparo de un evento tenga lugar en alguna configuración particular del sistema, y esta propiedad no se conserva en la división de eventos, ya que e^- puede ocurrir en una configuración distinta a la correspondiente para e^+. Para evitar este problema, se fuerza a que el evento e^+ ocurra inmediatamente después del evento e^-. De esta forma, perdemos toda la información sobre la concurrencia de eventos, pero como ya se ha mencionado, esto no tiene efecto sobre las propiedades que nos interesan. Así pues, toda RPCA $\mathcal{N} = (A, \Delta, N)$ se asocia con un SAVE $\mathcal{N}^\circ = (V, Arcs, l)$ con los componentes que siguen:

- Los vértices son o bien estados de la RPCA, o bien pares (s, e) formados por un estado y un evento habilitado en ese estado:

$$V = S \cup \{(s, e) \in S \times E | \exists s' \in S \; s \xrightarrow{e} s'\}.$$

- Para toda transición $s \xrightarrow{e} s'$ tenemos dos arcos en el SAVE: uno desde el vértice s al vértice (s,e) etiquetado $V(e^+,s)$ y otro desde el vértice (s,e) al vértice s' etiquetado $V(e^-,s)$ donde $V(e^+,s)$ y $V(e^-,s)$ son vectores de $\mathbb{Z}^{n+k}$ que vienen dados por

$$V(e^+,s)(i) = \begin{cases} -Pre(p_i,e)\cdot\Delta(s) & \text{si } 1 \leq i \leq n \\ 1 & \text{si } i > n \text{ y } e = e_{i-n} \\ 0 & \text{si } i > n \text{ y } e \neq e_{i-n} \end{cases}$$

y

$$V(e^-,s)(i) = \begin{cases} Post(p_i,e)\cdot\Delta(s) & \text{si } 1 \leq i \leq n \\ -1 & \text{si } i > n \text{ y } e = e_{i-n} \\ 0 & \text{si } i > n \text{ y } e \neq e_{i-n} \end{cases}$$

donde $P = \{p_1,\ldots,p_n\}$ y $E = \{e_1,\ldots,e_k\}$ son enumeraciones fijas del conjunto de lugares y de eventos de la RPCA.

Entonces, los marcados de $\mathcal{N}$ son también marcados de $\mathcal{N}^\circ$ y $(M,s) \xrightarrow{e} (M',s')$ aparece en el grafo de marcado de $\mathcal{N}$ si y sólo si la secuencia de transiciones $(M,s) \overset{V(e^+,s)}{\to} (M''\cup\{e\},s) \overset{V(e^-,s)}{\to} (M',s')$ aparece en el grafo de marcado de $\mathcal{N}^\circ$ siendo $M'' : P \to \mathbb{N}$.

Por otra parte, $(M,s) \overset{V(e^+,s)}{\to} (M''\cup\{e\},s)$ es la única transición que llega al marcado $(M''\cup\{e\},s)$ y $(M''\cup\{e\},s) \overset{V(e^-,s)}{\to} (M',s')$ es la única transición desde este marcado puesto que $V(e^+,s)$ que añade un token al lugar e y $V(e^-,s)$ que elimina un token de dicho lugar, son los únicos vectores que modifican el valor del lugar e.

Si restringimos nuestra atención a los marcados alcanzables desde algún marcado inicial $(M_0,s_0) \in \mathbb{N}^P \times S$ se sigue que los marcados alcanzables de $\mathcal{N}^\circ$ son marcados alcanzables de $\mathcal{N}$ junto con algún marcado de la forma $(M''\cup\{e\},s)$ donde $M'' : P \to \mathbb{N}$, $e \in E$ y $s \in S$.

Así pues, la RPCA y su SAVE asociado son equivalentes con respecto a las siguientes propiedades: acotamiento de un lugar, alcanzabilidad, interbloqueo y vivacidad, ya que estas propiedades son decidibles para los SAVEs (ver [Reu89]).

Proposición 130 *Acotamiento de un lugar, alcanzabilidad, interbloqueo y vivacidad son propiedades decidibles para las RPCAs.*

En concreto, podemos decidir el acotamiento de un lugar de una RPCA construyendo el árbol de cobertura de su SAVE asociado. Podemos simplificar la construcción de este árbol de cobertura amalgamando los arcos sucesivos asociados con $V(e^+, s)$ y $V(e^-, s)$. Concluimos esta sección con la descripción de este árbol de cobertura.

Se define una relación de orden entre marcados de una RPCA como sigue

$$(M, s) \sqsubseteq (M', s') \Longleftrightarrow (s = s') \wedge (\forall p \in P \ M(p) \leq M'(p))$$

De forma similar a lo que se hace para las redes de Petri, una aproximación finita del árbol de alcanzabilidad llamada árbol de cobertura puede construirse. Con este propósito, un nuevo valor ω que representa un entero "arbitrariamente grande" se añade al conjunto $\mathbb{N}$ de enteros no negativos; con la relación de orden y las operaciones extendidas de suma y sustracción de forma que $\forall n \in \mathbb{N}, n < \omega$ $\omega + n = n + \omega = \omega + \omega = \omega$, $\omega - n = \omega$ y $n - \omega$ no está definida.

Definición 131 (Marcados Generalizados) *Un* marcado generalizado *de una RPCA $\mathcal{N}$ es un par $(\widetilde{M}, s)$ donde $\widetilde{M}$ es una aplicación $\widetilde{M} : P \to \mathbb{N} \cup \{\omega\}$ y $s \in S$ es un estado.*

La relación de orden anterior puede extenderse a los marcados generalizados:

$$(\widetilde{M}, s) \sqsubseteq (\widetilde{M}', s') \Longleftrightarrow (s = s') \wedge (\forall p \in P \ \widetilde{M}(p) \leq \widetilde{M}'(p))$$

El árbol de cobertura de una RPCA $\mathcal{N}$ se construye según el Algoritmo 4.

Proposición 132 *El árbol de cobertura de una red de Petri controlada por un autómata es finito.*

Una red de Petri controlada por un autómata está acotada si y solamente si ningún vértice de su árbol de cobertura está etiquetado por (M, s) en donde M contiene una componente ω. Más precisamente, si (M, s) es algún marcado generalizado etiquetando un nodo del árbol de cobertura entonces, para todo entero N, existe un marcado alcanzable asociado con el mismo estado: (M', s) (es decir, $M' : P \to \mathbb{N}$ no tiene ninguna componente ω) tal que:

1. $M(p_i) = \omega \Longrightarrow M'(p_i) \geq N$;
2. $M(p_i) \in \mathbb{N} \Longrightarrow M'(p_i) = M(p_i)$.

Algoritmo 4 Árbol de Cobertura de una RPCA

- Inicialmente, el árbol se reduce a su raíz etiquetada (M_0, s_0) y señalada como un vértice "nuevo".
- Mientras existan vértices "nuevos" hacer:
  - Seleccionar un nuevo vértice V, sea (M, s) su etiqueta.
  - Si la etiqueta es distinta de todas las etiquetas de los nodos en el camino desde la raíz hasta el vértice V entonces para todo disparo $(M, s) \xrightarrow{e} (M', s')$ hacer:
    - Crear un nuevo vértice V' y un arco desde V a V' etiquetado con e y señalar este vértice V' como "nuevo".
    - Etiquetar el vértice V' con la configuración $(\widetilde{M}', s')$ definida como sigue.

      Si existe algún nodo V'' en el camino desde la raíz al vértice V cuya etiqueta (M'', s'') es tal que $(M'', s'') \sqsubseteq (M', s')$ y $M''(p) < M'(p)$,
      entonces $\widetilde{M}'(p) = \omega$.
      En otro caso, $\widetilde{M}'(p) = M'(p)$.
  - Retirar V del conjunto de vértices "nuevos".

Demostración:

Se sigue del resultado análogo para los SAVEs [Reu89].

■

4.2.2. Redes Reconfigurables Reversibles

Ya hemos visto en la Subsección 4.1.2 que las redes reconfigurables pueden cambiar su propia estructura dinámicamente reescribiendo alguna de sus componentes. Una red reconfigurable puede verse como un conjunto de redes de Petri (sus configuraciones) que se corresponden con los distintos modos de operación del sistema. Las reglas de modificación de la estructura permiten cambiar de un modo de operación a otro mientras el marcado actual se modifica adecuadamente desplazando los tokens de los lugares desaparecidos a los creados. En esta subsección presentamos una subclase de redes reconfigurables, llamadas *reversibles*, en las que toda regla de modificación de la estructura puede invertirse de manera formal. Por ejemplo, la red reconfigurable de la Figura 4.1 es reversible si todas sus reglas de modificación de la estructura son biyecciones.

Definición 133 (Red Reconfigurable Reversible (RRR)) *Una* red reconfigurable reversible *es una red reconfigurable en la que todas las reglas de modificación de la estructura son biyecciones.*

La primera propiedad que se verifica trivialmente en las redes reconfigurables reversibles y que justifica la terminología elegida, es el codeterminismo de sus grafos de marcado:

Observación 134 *Las redes reconfigurables reversibles tienen grafos de marcado codeterministas:*

$$M' \, [e\rangle \, M \wedge M'' \, [e\rangle \, M \Longrightarrow M' = M''$$

Toda regla de modificación de la estructura (biyectiva) $r : P_1 \to P_2$ induce una permutación del conjunto de lugares φ_r dada por:

$$\varphi_r(p) = \begin{cases} r(p) & \text{si } p \in P_1 \\ r^{-1}(p) & \text{si } p \in P_2 \\ p & \text{si } p \in P \setminus (P_1 \cup P_2) \end{cases}$$

Toda secuencia de eventos $u \in E^*$ induce también una permutación φ_u dada por:

$$\varphi_u = \begin{cases} id_P & \text{si } u \in T^* \\ \varphi_r \circ \varphi_v & \text{si } u = vr,\ v \in E^* \text{ y } r \in R \end{cases}$$

Proposición 135 (Propiedad Fundamental de las RRRs) *Sea* $N = (P, T, F, R)$ *una red reconfigurable reversible y* $u \in E^*$ *una secuencia de eventos. Sean* D_u^1, $D_u^2 \subseteq P$, $\sigma_u : P \to \mathbb{N}$ *y* $\Delta_u : P \to \mathbb{Z}$ *definidos como sigue:*

$$D_u^1 = \begin{cases} 0 & \textit{si } u \in T^* \\ P_1 & \textit{si } u = r \in R \quad r : P_1 \to P_2 \\ D_v^1 \cup \varphi_v^{-1}(D_e^1) & \textit{si } u = ve,\ v \in E^* \textit{ y } e \in E \end{cases}$$

$$D_u^2 = \begin{cases} 0 & si\ u \in T^* \\ P_2 & si\ u = r \in R \quad r : P_1 \to P_2 \\ D_v^2 \cup \varphi_v^{-1}(D_e^2) & si\ u = ve,\ v \in E^*\ y\ e \in E \end{cases}$$

$$\sigma_u(p) = \begin{cases} 0 & si\ u \in R^* \\ F(p,t) & si\ u = t \in T \\ \max\{\sigma_v(p); \sigma_e(\varphi_v(p)) - \Delta_v(p)\} & si\ u = ve,\ v \in E^*\ y\ e \in E \end{cases}$$

$$\Delta_u(p) = \begin{cases} 0 & si\ u \in R^* \\ F(t,p) - F(p,t) & si\ u = t \in T \\ \Delta_v(p) + \Delta_e(\varphi_v(p)) & si\ u = ve,\ v \in E^*\ y\ e \in E \end{cases}$$

Para los marcados M y M' se tiene

$$M\,[u\rangle\,M' \iff \begin{cases} (i) & D_u^1 \subseteq D(M)\ y\ D(M) \subseteq D_u^2 = \emptyset \\ (ii) & \forall p \in D(M)\ M(p) \geq \sigma_u(p) \\ (iii) & D(M') = \varphi_u(D(M)) \\ (iv) & \forall p \in D(M)\ M'(\varphi_u(p)) = M(p) + \Delta_u(p) \end{cases}$$

Demostración:

Sea $M \xrightarrow{u} M'$ cuando las condiciones *(i)* a *(iv)* se satisfacen. Este conjunto de transiciones es determinista. Más exactamente, escribimos $M \xrightarrow{u} M' \iff M \xrightarrow{u} \wedge M' = M \cdot u$ donde $M \xrightarrow{u} \iff (i) \wedge (ii)$ y $M' = M \cdot u \iff (iii) \wedge (iv)$. La relación ternaria $R(M,u,M') \iff [M' = M \cdot u]$ es, como se sugiere por su notación, funcional: dados M y u, las condiciones (iii) y (iv) caracterizan completamente el elemento M'. En concreto, siempre existe un M' y $M \xrightarrow{u}$ es lógicamente equivalente a $\exists M'\ M \xrightarrow{u} M'$. Vamos a demostrar que $M\,[u\rangle\,M' \iff M \xrightarrow{u} M'$ por inducción sobre la longitud de u. Los casos base se dan cuando $|u| \leq 1$:

- Sea $u = \epsilon$ la cadena vacía. Por convención, ϵ está habilitada en cualquier marcado M y $M\,[\epsilon\rangle\,M' \implies M' = M$. Puesto que $D_\epsilon^1 = D_\epsilon^2 = \emptyset$ y $\sigma_e(p) = 0$, se deduce que $M \xrightarrow{\epsilon}$ siempre se cumple. Puesto que $\varphi_\epsilon = id_P$ y $\Delta_\epsilon(p) = 0$, se deduce $M \cdot \epsilon = M$.
- Sea $u = t \in T$ una transición. Ya que $D_t^1 = D_t^2 = \emptyset$ y $\sigma_t(p) = F(p,t)$, $M \xrightarrow{t} \iff \forall p \in D(M)\ M(p) \geq F(p,t)$. Ya que $\Delta_t(p) = F(t,p) - F(p,t)$

y $\varphi_t = id_P$, $M' = M \cdot t \iff D(M') = D(M) \wedge \forall p \in D(M)\ M'(p) = M(p) + \Delta_t(p)$. Es decir, $M \xrightarrow{t} M' \iff M\,[t\rangle\, M'$.

- Sea $u = r \in R$ una regla de modificación de la estructura que es una biyección $r : P_1 \to P_2$. Puesto que $D_r^1 = P_1$, $D_r^2 = P_2$ y $\sigma_r(p) = 0$, $M \xrightarrow{t} \iff P_1 \subseteq D(M) \wedge D(M) \cap P_2 = \emptyset$. Puesto que $\Delta_r(p) = 0$, $M' = M \cdot r \iff D(M') = \varphi_r(D(M)) \wedge \forall p \in D(M)\ M'(\varphi_r(p)) = M(p)$, es decir, $M' = M \cdot r \iff M = M' \circ \varphi_r$. Es decir, $M \xrightarrow{r} M' \iff M\,[r\rangle\, M'$.

Asumimos $M\,[u\rangle\, M' \iff M \xrightarrow{u} M'$ para $|u| < n$ con $n \geq 1$. Sea $u = ve$ una palabra de longitud n, donde $v \in E^*$ y $e \in E$. $M\,[u\rangle\, M' \iff \exists M''\ M\,[v\rangle\, M'' \wedge M''\,[e\rangle\, M'$. Por hipótesis de inducción, $M\,[u\rangle$ es equivalente a $M \xrightarrow{v} \wedge\ M \cdot v \xrightarrow{e}$, es decir, a las siguientes condiciones

$$
\begin{array}{ll}
(i) & D_v^1 \subseteq D(M) \text{ y } D(M) \subseteq D_v^2 = \emptyset \\
(ii) & \forall p \in D(M)\ M(p) \geq \sigma_v(p) \\
(i)' & D_e^1 \subseteq D(M'') \text{ y } D(M'') \subseteq D_e^2 = \emptyset \\
(ii)' & \forall p \in D(M'')\ M''(p) \geq \sigma_e(p) \\
 & \text{donde el marcado } M'' \text{ viene dado por:} \\
(iii) & D(M'') = \varphi_v(D(M)) \\
(iv) & \forall p \in D(M)\ M''(\varphi_v(p)) = M(p) + \Delta_v(p)
\end{array}
$$

Como $D(M'') = \varphi_v(D(M))$, $(i) \wedge (i)' \iff D_v^1 \cup \varphi_v^{-1}(D_e^1) \subseteq D(M) \wedge [D_v^2 \cup \varphi_v^{-1}(D_e^2)] \cap D(M) = \emptyset$ es equivalente a

$$(i)''\ D_u^1 \subseteq D(M) \text{ y } D(M) \cap D_u^2 = \emptyset$$

$(ii) \wedge (ii)' \iff$ [*por (iii) y (iv)*]$\forall p \in D(M)\ M(p) \geq \sigma_v(p) \wedge M(p) + \Delta_v(p) \geq \sigma_e(\varphi_v(p)) \iff \forall p \in D(M)\ M(p) \geq \max\{\sigma_v(p); \sigma_e(\varphi_v(p)) - \Delta_v(p)\} = \sigma_u(p)$. Por tanto, $(ii) \wedge (ii)' \iff (ii)''$ donde:

$$(ii)''\ \forall p \in D(M)\ M(p) \geq \sigma_u(p)$$

Así pues, $M\,[u\rangle$ si y sólo si $M \xrightarrow{u}$. Ahora ambos sistemas de transiciones son deterministas y para los casos base $M\,[e\rangle\, M' \iff M \xrightarrow{e} M'$ para todo $e \in E$, por tanto, $M\,[u\rangle\, M' \iff M \xrightarrow{u} M'$ como se requería.

∎

Utilizamos la propiedad fundamental de las redes reconfigurables reversibles para demostrar que cualquier red reconfigurable reversible es equivalente a alguna red de Petri controlada por un autómata y para dar la definición de árbol de cobertura para redes reconfigurables reversibles. Esta definición mejora la construcción del árbol de cobertura introducida en [BO98b] para las redes reconfigurables que no sólo permite decidir acotamiento sino también acotamiento de un lugar. En la Sección 4.2.4, identificamos una subclase de redes reconfigurables reversibles que pueden simularse mediante una red de Petri estratificada [BD97], es decir, mediante una red de Petri automodificante con una estratificación del conjunto de lugares en niveles tales que la relación de flujo asociada a un lugar tiene que ver únicamente con el contenido de los lugares de niveles más bajos. Las redes de Petri automodificantes utilizadas en [BO98b] para simular redes reconfigurables no son estratificadas. Esta característica es esencial ya que las redes reconfigurables, a diferencia de las redes de Petri estratificadas, no son, en general, codeterministas en el sentido que no podemos para todo disparo $M\,[e\rangle\, M'$ deducir el marcado M a partir del evento e y el marcado M'. Como ya se ha mencionado, las redes reconfigurables son codeterministas si todas las reglas de modificación de la estructura se asumen biyecciones, es decir, si son redes reconfigurables reversibles.

4.2.3. Redes Reconfigurables Reversibles y Redes de Petri Controladas por un Autómata

Es sólo por conveniencia que se asume que el autómata y la red de Petri parámetrica tienen el mismo conjunto de eventos. Este no es el caso habitualmente, pero puede asumirse con seguridad adjuntando *eventos idle*:

- Un *evento idle de la parte de control* (autómata) es un $e \in E$ tal que $\forall s \in S\ s \xrightarrow{e} s$.
- Un *evento idle de la parte controlada* (red de Petri paramétrica) es un $e \in E$ tal que $\forall p \in P\ Pre(p, e) = Post(p, e) = 0$.

La Figura 4.5 muestra la red de Petri controlada por un autómata correspondiente a la red reconfigurable reversible de la Figura 4.1, nótese que los eventos idle no están representados como siempre era el caso.

Si A y B son autómatas deterministas con el mismo alfabeto de eventos, denotamos $A \preccurlyeq B$ y decimos que *A está cubierto por B* cuando existe un morfismo saturado de autómatas desde B a A, es decir, existe una aplicación σ del conjunto de estados de B al conjunto de estados de A que relaciona los

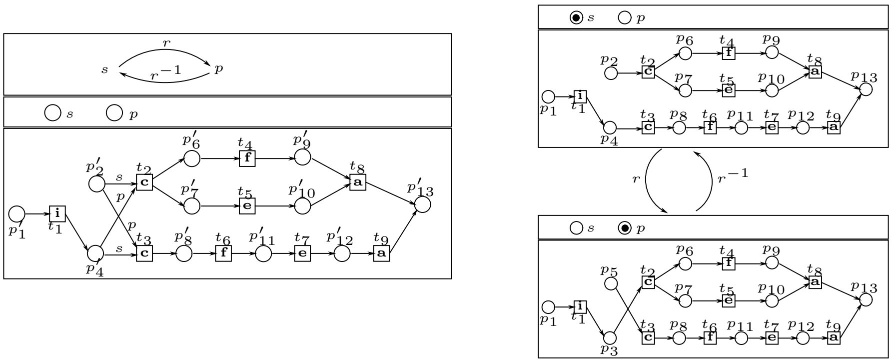


Figura 4.5: Red Reconfigurable Reversible de la Figura 4.1 como una Red de Petri Controlada por un Autómata

correspondientes estados iniciales y es tal que $s_1 \xrightarrow{e} s_2 \Longrightarrow \sigma(s_1) \xrightarrow{e} \sigma(s_2)$ y $\sigma(s_1) \xrightarrow{e} s_2' \Longrightarrow \exists s_2\ s_1 \xrightarrow{e} s_2 \wedge \sigma(s_2) = s_2'$.

Si todo estado es alcanzable desde el estado inicial en ambos autómatas, entonces dicho morfismo cuando existe es único y es sobreyectivo.

Además, la equivalencia generada por la relación de cobertura (es decir, la relación de equivalencia menor que contiene $\preccurlyeq$) identifica exactamente a aquellos autómatas que tienen el mismo lenguaje (donde cada estado es un estado terminal).

Proposición 136 *El grafo de marcado de una red reconfigurable reversible marcada está cubierto por (y es, por tanto, equivalente a) el grafo de marcado de una red de petri controlada por un autómata con codificación de estados trivial:*

$$\mathbf{mg}(N, M_0) \preccurlyeq N(\Phi) \times A(\Phi)$$

Si $N = (P, T, F, R)$ *entonces* $A(\Phi) = (S, E, T', s_0)$ *y* $N(\Phi) = (\Pi, P', E, Pre, Post, M_0')$ *donde:*

- $S = \Pi$ *es el grupo de permutaciones* Φ *generadas por las permutaciones* φ_r *asociadas con alguna regla de modificación de la estructura* $r \in R$ *y* s_0 *es el elemento neutro de ese grupo.*
- $E = T \cup R$ *es el conjunto de eventos de* N.
- $P' = D(M_0) = \{p \in P | M_0(p) \neq \alpha\}$ *es el dominio del marcado* M_0 *y*

- *M'_0 es la restricción de M_0 a su dominio:*

$$\forall p \in P' = D(M_0) \; M'_0(p) = M_0(p)$$

Los eventos no idle de la parte de control son reglas de modificación de la estructura $r \in R$ con

$$\varphi \xrightarrow{r} \varphi' \Longleftrightarrow r : P_1 \to P_2 \wedge P_1 \subseteq \varphi(P') \wedge P_2 \cap \varphi(P') = \emptyset \wedge \varphi' = \varphi_r \circ \varphi$$

Los eventos no idle de la parte controlada son las transiciones $t \in T$ de la red reconfigurable con $Pre(p,t) = \sum F(\varphi(p),t)\cdot\varphi$ y $Post(p,t) = \sum F(t,\varphi(p))\cdot\varphi$, es decir,

$$M\,[t/\varphi\rangle\,M' \Longleftrightarrow \forall p \in P \begin{cases} (i) & M(p) \geq F(\varphi(p),t) \\ (ii) & M'(p) = M(p) - F(\varphi(p),t) + F(t,\varphi(p)) \end{cases}$$

Demostración:

Por la Proposición 135 la relación de transición de N viene dada por

$$M\,[u\rangle\,M' \Longleftrightarrow \begin{cases} (i) & D_u^1 \subseteq D(M) \text{ y } D(M) \subseteq D_u^2 = \emptyset \\ (ii) & \forall p \in D(M) \; M(p) \geq \sigma_u(p) \\ (iii) & D(M') = \varphi_u(D(M)) \\ (iv) & \forall p \in D(M) \; M'(\varphi_u(p)) = M(p) + \Delta_u(p) \end{cases}$$

Supongamos que M es un marcado alcanzable, es decir, $M_0\,[u\rangle\,M$ para alguna secuencia de disparos $u \in E^*$. El marcado $M/u = M \circ \varphi_u$ es tal que $D(M/u) = D(M_0)$ y el marcado M puede ser recuperado desde M/u y φ_u utilizando la identidad $M = M/u \circ \varphi_u^{-1}$. Vamos a considerar los pares (M,φ) en los que φ pertenece al grupo Φ generado por las permutaciones φ_r para una regla de modificación de la estructura r y M un marcado con dominio $P' = D(M_0)$. Y asociamos un par (M,φ) con el marcado $M \circ \varphi^{-1}$. Así pues

$$M\circ\varphi^{-1}\,[u\rangle\,M'\circ\varphi'^{-1} \Longleftrightarrow \begin{cases} (i) & D_u^1 \subseteq \varphi(P') \text{ y } \varphi(P') \cap D_u^2 = \emptyset \\ (ii) & \forall p \in P' \; M(p) \geq \sigma_u(\varphi(p)) \\ (iii) & \varphi'(P') = \varphi_u(\varphi(P')) \\ (iv) & \forall p \in P' \; M'(\varphi'^{-1} \circ \varphi_u \circ \varphi(p)) = M(p) + \Delta_u(\varphi(p)) \end{cases}$$

Nótese que $M \circ \varphi^{-1} [u\rangle\, M' \circ \varphi'^{-1}$ no implica que $\varphi' = \varphi_u \circ \varphi$. Sea $(M, \varphi) \xrightarrow{u} (M', \varphi')$ cuando $M \circ \varphi^{-1} [u\rangle\, M' \circ \varphi'^{-1}$ y $\varphi' = \varphi_u \circ \varphi$. Así pues,

$$(M,\varphi) \xrightarrow{u} (M',\varphi') \Longleftrightarrow \begin{cases} \textit{(i)} & D_u^1 \subseteq \varphi(P') \text{ y } \varphi(P') \cap D_u^2 = \emptyset \\ \textit{(ii)} & \forall p \in P'\ M(p) \geq \sigma_u(\varphi(p)) \\ \textit{(iii)} & \varphi' = \varphi_u \circ \varphi \\ \textit{(iv)} & \forall p \in P'\ M'(p) = M(p) + \Delta_u(\varphi(p)) \end{cases}$$

Cuando $u = t \in T$ es una transición de N, estas condiciones se reducen a

$$(M,\varphi) \xrightarrow{t} (M',\varphi') \Longleftrightarrow \begin{cases} \textit{(i)} & \square \\ \textit{(ii)} & \forall p \in P'\ M(p) \geq F(\varphi(p), t) \\ \textit{(iii)} & \varphi' = \varphi \\ \textit{(iv)} & \forall p \in P'\ M'(p) = M(p) - F(\varphi(p), t) + F(t, \varphi(p)) \end{cases}$$

Cuando $u = r \in R$ es una regla de modificación de la estructura de N, lo anterior se reduce a

$$(M,\varphi) \xrightarrow{r} (M',\varphi') \Longleftrightarrow \begin{cases} \textit{(i)} & P_1 \subseteq \varphi(P') \text{ y } P_2 \cap \varphi(P') = \emptyset \\ \textit{(ii)} & \square \\ \textit{(iii)} & \varphi' = \varphi_r \circ \varphi \\ \textit{(iv)} & M' = M \end{cases}$$

Por lo tanto, la restricción del conjunto de transiciones $(M, \varphi) \xrightarrow{e} (M', \varphi')$ al conjunto de configuraciones (M, φ) alcanzable desde la configuración inicial (M_0, id) es la red de Petri controlada por un autómata $N(\Phi) \times A(\Phi)$. Ahora, la aplicación $f : N(\Phi) \times A(\Phi) \to \mathbf{mg}(N, M_0)$ dada por $f(M, \varphi) = M \circ \varphi^{-1}$ es, por definición de $M \circ \varphi^{-1} [u\rangle\, M' \circ \varphi'^{-1}$, un morfismo del autómata. Es un morfismo saturado porque $M \circ \varphi^{-1} [e\rangle\, M'$, que es la conjunción de las siguientes condiciones

$$\begin{array}{ll} \textit{(i)} & D_e^1 \subseteq \varphi(P') \text{ y } \varphi(P') \cap D_e^2 = \emptyset \\ \textit{(ii)} & \forall p \in P'\ M(p) \geq \sigma_e(\varphi(p)) \\ \textit{(iii)} & D(M') = \varphi_e(\varphi(P')) \\ \textit{(iv)} & \forall p \in P'\ M'(\varphi_e \circ \varphi(p)) = M(p) + \Delta_e(\varphi(p)) \end{array}$$

lo que significa que tenemos una transición $(M, \varphi) \xrightarrow{e} (M' \circ \varphi_e \circ \varphi, \varphi_e \circ \varphi)$ cuya target es enviada a M': $f(M' \circ \varphi_e \circ \varphi, \varphi_e \circ \varphi) = M'$.

■

Una red reconfigurable reversible y su RPCA asociada son equivalentes con respecto a las siguientes propiedades: acotamiento de un lugar, alcanzabilidad, vivacidad e interbloqueo.

Proposición 137 *El acotamiento de un lugar, la alcanzabilidad, la vivacidad y el interbloqueo son propiedades decidibles para las redes reconfigurables reversibles.*

El único punto en la afirmación anterior que no está inmediatamente claro es la propiedad de acotamiento de un lugar. Efectivamente, distintos lugares de la red reconfigurable reversible se representan por el mismo lugar en la red de Petri controlada por un autómata asociada. La componente estado aumenta la ambigüedad: por la Proposición 132 el lugar p de una red reversible reconfigurable está acotado si y solamente si para todo vértice del árbol de cobertura de la RPCA asociada ocurre que si (M, φ) es la etiqueta de dicho vértice, $M(\varphi^{-1}(p)) \neq \omega$.

Para concluir esta sección vamos a reformular la definición de árbol de cobertura de la RPCA asociada con una red reconfigurable reversible. Recordemos que la relación de orden entre marcados (generalizados) viene dada por

$$(M, \varphi) \sqsubseteq (M', \varphi') \iff (\varphi = \varphi') \wedge (\forall p \in \Omega_0 \ M(p) \leq M'(p))$$

Así, una componente ω se añade a la construcción de la cobertura cuando se alcanza un vértice V' asociado con el marcado generalizado (M', φ') tal que existe algún vértice V en el camino desde la raíz a V' etiquetado con el marcado generalizado (M, φ) tal que $(M, \varphi) \sqsubseteq (M', \varphi')$ y $M(p) < M'(p)$ para algún p. Con lo que $\varphi = \varphi'$ y $\varphi_u = id$ siendo u la etiqueta del camino desde V a V'.

El árbol de cobertura de una red reconfigurable reversible marcada (N, M_0) se construye según el Algoritmo 5. En este algoritmo, una secuencia se denomina *iterating* cuando la permutación inducida es la aplicación identidad.

4.2.4. Redes Reconfigurables Reversibles y Redes de Petri Estratificadas

En [BO98b] se presentó una traducción de una red reconfigurable a una red automodificante equivalente. Las redes automodificantes [Val78a, Val78b,

Algoritmo 5 Árbol de Cobertura de una RRR

- Inicialmente, el árbol se reduce a su raíz etiquetada M_0 y señalada como un vértice "nuevo".
- Mientras existan vértices "nuevos" hacer:
  - Seleccionar un nuevo vértice V, sea M su etiqueta.
  - Si la etiqueta es distinta de todas las etiquetas de los nodos en el camino desde la raíz hasta el vértice V entonces para todo disparo $M[e\rangle M'$ hacer:
    - Crear un nuevo vértice V' y un arco desde V a V' etiquetado con e y señalar este vértice V' como "nuevo".
    - Etiquetar el vértice V' con el marcado generalizado $\widetilde{M'}$ definida como sigue.

      Si existe algún nodo V'' en el camino desde la raíz al vértice V cuya etiqueta M'' es tal que $M'' \sqsubseteq M'$ y $M''(p) < M'(p)$ y tal que la etiqueta del camino desde V'' a V' es una secuencia *iterating* de eventos, entonces $\widetilde{M'}(p) = \omega$.
      En otro caso, $\widetilde{M'}(p) = M'(p)$.
  - Retirar V del conjunto de vértices "nuevos".

Val81] son generalizaciones de redes Lugar/Transición en las que la relación de flujo entre un lugar y una transición depende del marcado. La traducción que se propone en la Figura 4.3 implica arcos reset. Este hecho puede sorprender desde el punto de vista del resultado de Dufourd según el cual el acotamiento de las redes de Petri con arcos reset es indecidible. En algunos casos, sin embargo, tales arcos no se necesitan, por ejemplo, la Figura 4.6 muestra una red automodificante equivalente a la red reconfigurable reversible de la Figura 4.1. La red de la Figura 4.6 es estratificada [BD97], lo que significa que existe una estratificación del conjunto de lugares en niveles de forma que las relaciones de flujo ligadas a un lugar involucran solamente al contenido de lugares de niveles más bajos.

Definición 138 (Red de Petri Estratificada) *Una* red de Petri estratificada *es una red automodificante $N = (P, T, F)$ cuyo conjunto de lugares P*

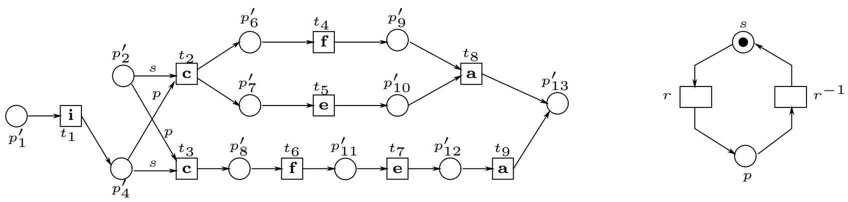


Figura 4.6: Red de Petri estratificada equivalente a la red reconfigurable de la Figura 4.1

puede ordenarse parcialmente de forma que si $F(p,t)$ *o* $F(t,p)$ *es de la forma* $\lambda_0 + \sum_{i=1}^{m} \lambda_i \cdot p_i$ *entonces* $\lambda_i = 0$ *para todo índice i tal que* $p \leq p_i$*.*

Más exactamente, ya que toda regla de modificación de la estructura es una biyección, el número de lugares que existe en un modo de operación actual de una red reconfigurable reversible no cambia cuando ocurre una regla de modificación de la estructura. Por lo tanto, un lugar p que exista en el marcado inicial puede representar al conjunto de lugares $\varphi(p)$ para φ tomando valores en el grupo de permutaciones del conjunto de lugares generado por las permutaciones φ_r asociadas con alguna regla de modificación de la estructura r. Por ejemplo, el lugar p'_2 (respectivamente el lugar p'_4) de la Figura 4.6 equivale al lugar p_2 (resp. el lugar p_4) de la Figura 4.1 cuando el sistema está en el modo secuencial de operación; y equivale al lugar p_5 (resp. el lugar p_3) cuando el sistema está en el modo paralelo de operación. Una red reconfigurable reversible puede verse como una red de Petri paramétrica cuyas instancias son sus configuraciones, controladas por un autómata que describe cómo cambiar de una configuración a otra.

Observación 139 *Un autómata* $A = (S, E, T, s_0)$ *es isomorfo al grafo de marcado de una red de Petri estratificada si y sólo si*

$$A \cong N_k \rtimes (N_{k-1} \rtimes \ldots \rtimes N_2 \rtimes (N_1 \rtimes B_E) \ldots)$$

donde $B_E = (\{*\}, E, \{*\} \times E \times \{*\}, *)$ *es un bouquet [BML80] y* N_1 *a* N_k *son redes de Petri paramétricas.*

En particular, el grafo de marcado de una red de Petri es de la forma $N \rtimes B_E$, es decir, una red de Petri puede interpretarse como N_α para una red de Petri paramétrica N con un parámetro ($\Pi = \{*\}$) donde $\alpha(*) = 1$. El significado del símbolo $\rtimes$ se introdujo en la Definición 128.

En la Proposición 136 dejamos de tener un isomorfismo puesto que el dominio del marcado inicial puede "mapearse" al mismo conjunto mediante diferentes permutaciones de Φ. Si excluimos redes para las que este fenómeno ocurre, tenemos un isomorfismo como se demuestra en la proposición siguiente.

Definición 140 (Red Reconfigurable Reversible Simple) *Una* red reconfigurable reversible *se dice* simple *si la única permutación de Φ (el grupo generado por las permutaciones φ_r asociadas con la regla de modificación de la estructura $r \in R$) que "mapea" el dominio del marcado inicial en sí mismo es la identidad.*

Sea $\Omega_0 = D(M_0)$ el dominio del marcado inicial entonces, equivalentemente, la red es simple si $\forall \varphi, \psi \in \Phi \; \varphi(\Omega_0) = \psi(\Omega_0) \Longrightarrow \varphi = \psi$.

Proposición 141 *El grafo de marcado de una red reconfigurable reversible simple es isomorfo al grafo de marcado de una red de Petri controlada por un autómata:*

$$\mathbf{mg}(N, M_0) \cong N(\Phi) \rtimes A(\Phi)$$

Además, en este caso tenemos también un isomorfismo

$$N(\Phi) \rtimes A(\Phi) \cong N(\Phi) \rtimes_\Delta A(\mathcal{O})$$

siendo $A(\mathcal{O}) = (S, E, T', s_0)$ el autómata cuyos estados son los modos de operación de la red marcada: $S = \mathcal{O}(N, M_0)$, cuyo estado inicial $s_0 = \Omega_0$ es el modo de operación $\Omega_0 = D(M_0)$ del marcado inicial. Los eventos no idle de $A(\mathcal{O})$ son reglas de modificación de la estructura $r \in R$ con

$$\Omega \xrightarrow{r} \Omega' \Longleftrightarrow r : P_1 \to P_2 \wedge P_1 \subseteq \Omega \wedge P_2 \cap \Omega = \emptyset \wedge \Omega' = \varphi_r(\Omega)$$

La matriz función de codificación de estados $\Delta : \mathcal{O}(N, M_0) \times \Phi \to \mathbb{N}$ viene dada por

$$\Delta(\Omega, \varphi) = \textit{si } \Omega = \varphi(\Omega_0) \textit{ entonces } 1 \textit{ sino } 0$$

Demostración:

Decimos que la aplicación $f : N(\Phi) \rtimes A(\Phi) \to \mathbf{mg}(N, M_0)$ dado por $f(M, \varphi) = M \circ \varphi^{-1}$ es un morfismo saturado de autómata. Puesto que los autómatas correspondientes son deterministas y accesibles, el morfismo saturado es sobreyectivo y es un isomorfismo si y sólo si es inyectivo. Suponer que $f(M,\varphi) = f(M',\varphi')$, sea Ω el dominio de $M \circ \varphi^{-1} = M' \circ \varphi'^{-1}$. Entonces $\Omega = \varphi(\Omega_0) = \varphi'(\Omega_0)$ donde $\Omega_0 = D(M_0)$ es el dominio del marcado inicial y, por simplicidad, se sigue que $\varphi = \varphi'$ y, por tanto, $M = M'$, lo que demuestra que f es inyectiva. Así pues,

$$\mathbf{mg}(N, M_0) \cong N(\Phi) \rtimes A(\Phi)$$

Ahora, tenemos una correspondencia biyectiva entre las configuraciones alcanzables de $N(\Phi) \rtimes A(\Phi)$ y las configuraciones alcanzables de $N(\Phi) \times_\Delta A(\mathcal{O})$ tomando (M,φ) como $(M,\varphi(\Omega_0))$ y, a la inversa, (M,Ω) como $(M,\Delta(\Omega))$. Esta correspondencia es un isomorfismo de autómata, ya que si $\varphi = \Delta(\Omega)$ y $\varphi' = \Delta(\Omega')$ se tiene

$$\begin{array}{ll}
\multicolumn{2}{l}{(M,\Omega) \xrightarrow{e} (M',\Omega') \text{ en } N(\Phi) \rtimes_\Delta A(\mathcal{O})} \\
\Longleftrightarrow & \begin{array}{ll} [\text{si } e = t] & M[t/\Delta(\Omega)\rangle M' \text{ en } N(\Phi) \wedge \Omega' = \Omega \\ [\text{si } e = r] & M = M' \wedge P_1 \subseteq \Omega \wedge P_2 \cap \Omega = \emptyset \wedge \Omega' = \varphi_r(\Omega) \end{array} \\
\Longleftrightarrow & \begin{array}{ll} [\text{si } e = t] & M[t/\varphi\rangle M' \text{ en } N(\Phi) \wedge \varphi' = \varphi \\ [\text{si } e = r] & M = M' \wedge P_1 \subseteq \varphi(\Omega_0) \wedge P_2 \cap \varphi(\Omega_0) = \emptyset \wedge \varphi' = \varphi_r \circ \varphi \end{array} \\
\multicolumn{2}{l}{(M,\varphi) \xrightarrow{e} (M',\varphi') \text{ en } N(\Phi) \rtimes A(\Phi)}
\end{array}$$

■

La ventaja de esta segunda representación (ver Figura 4.7) es que el autómata $A(\mathcal{O})$ es isomorfo al grafo de marcado de una red de Petri 1-segura. De esta forma tenemos una representación distribuida del control. Más exactamente, dos reglas de modificación de la estructura son independientes si tienen que ver con subconjuntos disjuntos de lugares de la red de Petri paramétrica controlada.

A causa de la presencia de una función de codificación de estados no trivial, la red de Petri controlada por un autómata $N(\Phi) \rtimes_\Delta A(\mathcal{O})$ no conduce directamente a una representación mediante una red de Petri estratificada. Es necesario codificar los parámetros $\varphi \in \Phi$ de la red $N(\Phi)$ mediante lugares de una red de Petri estratificada con grafo de marcado isomorfo a $A(\mathcal{O})$.

Proposición 142 *Sea $\mathcal{N} = (P,T,F,M_0)$ una red de Petri marcada 1-segura y $X \subseteq 2^P$ un conjunto de 'propiedades' de $\mathcal{N}$. Existe una red de Petri estratificada marcada $\mathcal{N}(X) = (\widetilde{P},T,\widetilde{F},\widetilde{M_0})$ tal que*

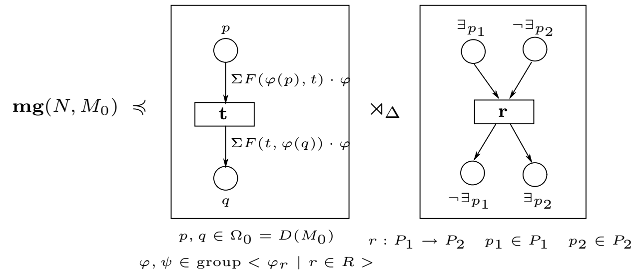


Figura 4.7: Red reconfigurable reversible como una composición en cascada de redes de Petri

- *$\mathcal{N}(X)$ es una extensión conservativa de $\mathcal{N}$ en el sentido que $P \subseteq \widetilde{P}$, $\widetilde{F}(p,t) = F(p,t)$ y $\widetilde{M}_0(p) = M_0(p)$ para todo $p \in P$ y la aplicación $(\widetilde{M} \subseteq \widetilde{P}) \mapsto (\widetilde{M} \cap P \subseteq P)$ induce un isomorfismo entre los respectivos grafos de marcado de $\mathcal{N}$ y $\mathcal{N}(X)$, y*

- *toda propiedad de X se representa en $\mathcal{N}(X)$: para todo $x \in X$ existe un lugar $p_x \in \widetilde{P}$ tal que para todo marcado alcanzable $\widetilde{M}$ de $\mathcal{N}(X)$ se tiene que $\widetilde{M}(p_x) = 1$ si y sólo si $x \subseteq \widetilde{M} \cap P$.*

Demostración:

Consideremos primero el caso en el que $X = 2^P$. Sea $\widetilde{P} = X = 2^P$ y se define $\widetilde{F}(\widetilde{p}, t)$ y $\widetilde{F}(t, \widetilde{p})$ para $\widetilde{p} \in \widetilde{P}$ como sigue. La relación de flujo $\widetilde{F}$ es una extensión de F, es decir, $\widetilde{F}(\{p\}, t) = F(p,t)$ y $\widetilde{F}(t, \{p\}) = F(t,p)$ para $p \in P$. Sea $x \in \widetilde{P}$ con cardinalidad al menos dos, si $\exists p \in x$, $F(p,t) = 1$ y $\forall p \in x$, $F(t,p) = 0$ entonces $\widetilde{F}(x,t) = \{p \in x | F(p,t) = 0\}$ (es decir, $\widetilde{F}(x,t) = y \in \widetilde{P}$ con $y \subset x$) sino $\widetilde{F}(x,t) = 0$. Simétricamente, si $\exists p \in x$, $F(t,p) = 1$ y $\forall p \in x$, $F(p,t) = 0$ entonces $\widetilde{F}(t,x) = \{p \in x | F(t,p) = 0\}$ sino $\widetilde{F}(t,x) = 0$. Finalmente, $\widetilde{M}_0(x) = 1$ si $\forall p \in x\ M_0(p) = 1$ sino $\widetilde{M}_0(x) = 0$. $\mathcal{N}(X)$ denota la red de Petri estratificada obtenida $(\widetilde{P}, T, \widetilde{F}, \widetilde{M}_0)$. Si identificamos $p \in P$ con el único elemento del conjunto $\{p\} \in \widetilde{P}$, se tiene que $P \subseteq \widetilde{P}$, $\widetilde{F}(p,t) = F(p,t)$ y $\widetilde{M}_0(p) = M_0(p)$ para todo $p \in P$. Hay que probar que la aplicación que toma el marcado M de $\mathcal{N}$ como el marcado $\widetilde{M}$ de $\mathcal{N}(X)$ dada por

$$\widetilde{M}(x) = \begin{cases} 1 & \text{si } \forall p \in x\ M(p) = 1 \\ 0 & \text{en otro caso} \end{cases}$$

es un isomorfismo entre grafos de marcado. Lo que se sigue a partir de que la red de Petri estratificada $\mathcal{N}(X)$ es 1-segura. Además, los marcados accesibles de $\mathcal{N}(X)$ son de la forma $\widetilde{M}$ para M marcado accesible de $\mathcal{N}$ y el isomorfismo inverso se sigue a partir de la identidad $M = \widetilde{M} \cap P$ donde los marcados de las redes 1-segura se ven como conjuntos de lugares. Así pues, el lugar x de $\mathcal{N}(X)$ representa a x visto como una propiedad $x \subseteq P$ de $\mathcal{N}$ ya que $\widetilde{M}(x) = 1$ si y sólo si $\forall p \in x\ M(p) = 1$, es decir, $x \subseteq M = \widetilde{M} \cap P$ para todo marcado accesible $\widetilde{M}$. Por tanto, si la aplicación $M \mapsto \widetilde{M}$ es un isomorfismo, la red $\mathcal{N}(X)$ es una extensión conservativa de $\mathcal{N}$ en la que toda propiedad de X se representa. Ya que los grafos de marcado de las redes son deterministas y alcanzables, para verificar que $M \mapsto \widetilde{M}$ es un isomorfismo, es suficiente comprobar que (i) t está habilitada en un marcado M si y sólo si está habilitada en $\widetilde{M}$ y (ii) si $M[t\rangle N$ y $\widetilde{M}[t\rangle\mathbf{N}$ entonces $\mathbf{N} = \widetilde{N}$; lo que equivale a demostrar que para toda transición $t \in T$ y propiedad $x \subseteq P$ se tiene que

$$\begin{array}{ll} \textit{(i)} & M[t\rangle \Longrightarrow \widetilde{F}(x,t) \cdot \widetilde{M} \leq \widetilde{M}(x) \\ \textit{(ii)} & M[t\rangle N \Longrightarrow \widetilde{N}(x) = \widetilde{M}(x) - \widetilde{F}(x,t) \cdot \widetilde{M} + \widetilde{F}(t,x) \cdot \widetilde{M} \end{array}$$

Vamos a realizar un análisis por casos:

1. Asumir que existe $p \in x$ tal que $F(p,t) = 1$ y $q \in x$ tal que $F(t,q) = 1$. Entonces, si $M[t\rangle M'$, necesariamente se tiene que $M(q) = 0$ y $M'(p) = 0$ y, por tanto, $\widetilde{M}(x) = \widetilde{M'}(x) = 0$. Puesto que $\widetilde{F}(x,t) = \widetilde{F}(t,x) = 0$, las condiciones (i) e (ii) se satisfacen.

2. Asumir que para todo $p \in x$, $F(t,p) = 0$ y existe algún $q \in x$ tal que $F(q,t) = 1$. Entonces $\widetilde{F}(x,t) = y$ siendo $y = \{p \in x | F(p,t) = 0\}$ y $\widetilde{F}(t,x) = 0$. Si $M[t\rangle M'$ entonces $\widetilde{M}(x) = \widetilde{M}(y)$ porque todo lugar de entrada de t está marcado en M y $\widetilde{M'}(x) = 0$. Por tanto, las condiciones (i) e (ii) se satisfacen.

3. Asumir que para todo $p \in x$, $F(p,t) = 0$ y existe algún $q \in x$ tal que $F(t,q) = 1$. Entonces $\widetilde{F}(t,x) = y$ siendo $y = \{p \in x | F(t,p) = 0\}$ y $\widetilde{F}(x,t) = 0$. Si $M[t\rangle M'$ entonces $\widetilde{M}(x) = 0$ y $\widetilde{M'}(x) = \widetilde{M'}(y)$ porque todo lugar de salida de t está marcado en M'. Por tanto, las condiciones (i) e (ii) se satisfacen.

4. Asumir que para todo $p \in x$, $F(p,t) = F(t,p) = 0$, entonces $\widetilde{F}(x,t) = \widetilde{F}(t,x) = 0$ y $\widetilde{M}(x) = \widetilde{M'}(x)$ para todo par de marcados M y M' tales que $M[t\rangle M'$. Por tanto, las condiciones (i) e (ii) se satisfacen.

Ahora, si $X \subset P$, la red definida previamente por $X = 2^P$, es decir, $\mathcal{N}(2^P)$, satisface los requerimientos de la proposición. Sin embargo, otra solución más pequeña se obtiene restringiendo $\mathcal{N}(2^P)$ al conjunto de lugares $\widetilde{P} = \{y \subseteq P | \exists x \in X \; y \Subset x\}$ siendo $\Subset$ el cierre reflexivo-transitivo de la relación $\ll$ dada por $y \ll x \iff \exists t \in T$

$$(\forall p \in x \; F(t,p) = 0 \wedge \exists p \in x \; F(p,t) = 1 \wedge y = \{p \in x | F(p,t) = 0\}) \vee$$
$$(\forall p \in x \; F(p,t) = 0 \wedge \exists p \in x \; F(t,p) = 1 \wedge y = \{p \in x | F(t,p) = 0\})$$

es decir, primero consideramos en $\mathcal{N}(X)$ todos los lugares de $\mathcal{N}(2^P)$ que son propiedades en X e iterativamente, añadimos todos los otros lugares de $\mathcal{N}(2^P)$ que aparecen como inscripciones de los arcos de lugares ya incorporados en $\mathcal{N}(X)$. Es, de alguna manera, la subred estratificada de $\mathcal{N}(2^P)$ inducida por X.

■

Puesto que $\mathcal{N}(X)$ es más pequeño que $\mathcal{N}(Y)$ cuando $X \subseteq Y$, existe menos 'pérdida de concurrencia' cuando se reemplaza la red $\mathcal{N}$ por $\mathcal{N}(X)$ que cuando se reemplaza por $\mathcal{N}(Y)$. Es decir, el internamiento de más propiedades se paga con una pérdida en concurrencia. En una situación extrema, si queremos representar todos los subconjuntos de lugares entonces el sistema resultante pasa a ser completamente secuencial.

A partir de la Proposición 141 y la Proposición 142 se sigue que

Proposición 143 *Cualquier red reconfigurable reversible simple marcada puede asociarse con una red de Petri estratificada marcada con grafo de marcado isomorfo y cuyo conjunto de transiciones es el conjunto de eventos de la red reconfigurable original.*

La red de Petri estratificada correspondiente se representa en la Figura 4.8. Más exactamente, si $\mathcal{N} = (P, T, R, F, M_0)$ es una red reconfigurable reversible simple marcada, la red de Petri estratificada marcada asociada es $\mathcal{N}' = (P', T', F', M_0')$ definida como sigue.

- $P' = \Omega_0 \cup \exists P \cup \neg\exists P \cup \mathcal{O}(N, M_0) \downarrow$ siendo Ω_0 el dominio del marcado inicial, $\exists P$ y $\neg\exists P$ copias disjuntas de P cuyos respectivos elementos típicos se denotan por $\exists p$ y $\neg\exists p$ para p tomando valores en P y $\mathcal{O}(N, M_0) \downarrow = \{\gamma \subseteq P | \exists \Omega \in \mathcal{O}(N, M_0) \; \gamma \Subset \Omega\}$ es el cierre descendente

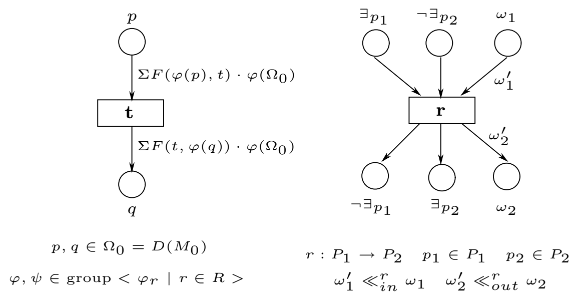


Figura 4.8: Red de Petri estratificada equivalente a una red reconfigurable reversible simple

del conjunto $\mathcal{O}(N, M_0)$ de modos de operación para el cierre reflexivo-transitivo $\Subset$ de la relación $\ll = \bigcup_{r \in R}(\ll^r_{in} \cup \ll^r_{out})$ donde

$$\gamma' \ll^r_{in} \gamma \iff [(P_1 \cup \neg P_2) \cap \gamma \neq \emptyset \wedge (\neg P_1 \cup P_2) \cap \gamma = \emptyset \wedge \gamma = \gamma' \setminus (P_1 \cup \neg P_2)]$$

y

$$\gamma' \ll^r_{out} \gamma \iff [(\neg P_1 \cup P_2) \cap \gamma \neq \emptyset \wedge (P_1 \cup \neg P_2) \cap \gamma = \emptyset \wedge \gamma = \gamma' \setminus (\neg P_1 \cup P_2)]$$

- $T' = T \cup R$ es el conjunto de eventos de $\mathcal{N}$.
- La relación de flujo viene dada por las siguientes identidades donde $t \in T$ y $r : P_1 \to P_2 \in R$

$$F'(p,t) = \begin{cases} \sum_{\varphi \in \Phi} F(\varphi(p), t) \cdot \varphi(\Omega_0) & \text{si } p \in \Omega_0 \\ 0 & \text{en otro caso} \end{cases}$$

$$F'(t,p) = \begin{cases} \sum_{\varphi \in \Phi} F(t, \varphi(p)) \cdot \varphi(\Omega_0) & \text{si } p \in \Omega_0 \\ 0 & \text{en otro caso} \end{cases}$$

$$F'(r,p) = \begin{cases} 1 & \text{si } p \in P_1 \cup \neg P_2 \\ \gamma' & \text{si } p = \gamma \in \mathcal{O}(N, M_0) \downarrow \text{ y } \gamma' \ll^r_{in} \gamma \\ 0 & \text{en otro caso} \end{cases}$$

$$F'(p,r) = \begin{cases} 1 & \text{si } p \in \neg P_1 \cup P_2 \\ \gamma' & \text{si } p = \gamma \in \mathcal{O}(N, M_0) \downarrow \text{ y } \gamma' \ll^r_{out} \gamma \\ 0 & \text{en otro caso} \end{cases}$$

- El marcado inicial M_0' viene dado por

$$M_0'(p) = \begin{cases} M_0(p) & \text{si } p \in \Omega_0 \\ 1 & \text{si } p = \exists q \text{ donde } q \in \Omega_0 \text{ o } p = \neg\exists q \text{ donde } q \notin \Omega_0 \\ 0 & \text{si } p = \exists q \text{ donde } q \notin \Omega_0 \text{ o } p = \neg\exists q \text{ donde } q \in \Omega_0 \\ 1 & \text{si } p = \gamma \in \mathcal{O}(N, M_0) \downarrow \text{ y } \gamma \subseteq M_0 \\ 0 & \text{si } p = \gamma \in \mathcal{O}(N, M_0) \downarrow \text{ y } \gamma \nsubseteq M_0 \end{cases}$$

4.3. Algunos ejemplos

En esta sección presentamos algunos ejemplos de sistemas concurrentes sujetos a cambios dinámicos. Están modelizados, respectivamente, por una red reversible reconfigurable, una red de Petri y una red de Petri estratificada. La elección de un modelo particular depende de la naturaleza del problema a modelizar.

4.3.1. Envío de paquetes de información en una red de transmisión

Las redes reconfigurables reversibles se adaptan bien para la descripción de grandes sistemas distribuidos en los que los cambios dinámicos pueden ocurrir en diferentes lugares y tienen efectos locales, incluso si dichos cambios no son totalmente independientes. Cuando los cambios requieren una sincronización global del sistema, como en el ejemplo de la siguiente sección, entonces las redes de Petri son más adecuadas. Vamos a presentar un ejemplo de talla pequeña para ilustrar este tipo de situación. Consideraremos una red de transmisión que recibe desde dos máquinas (representadas por las transiciones t_1 y t_8 en la Figura 4.9) bloques de información que son tratados por un servidor principal (representado por el lugar p_{12}).

Esta red tiene dos partes diferentes delimitadas por los lugares p_1, p_6 y p_{12}. En la primera parte, existen cuatro posibilidades dependiendo de la decisión de un operador. Puede decidir enviar los paquetes de uno en uno, de dos en dos, de tres en tres o de cuatro en cuatro, lo que se representa respectivamente por los caminos $t_2p_5t_3$, $t_2p_4t_3$, $t_2p_3t_3$ y $t_2p_2t_3$. En la segunda parte, la red ofrece tres posibilidades para el seguimiento de los datos en la red:

- paquetes que siguen el camino normal $t_4p_8t_5p_{10}t_6p_{11}t_7p_{12}$;

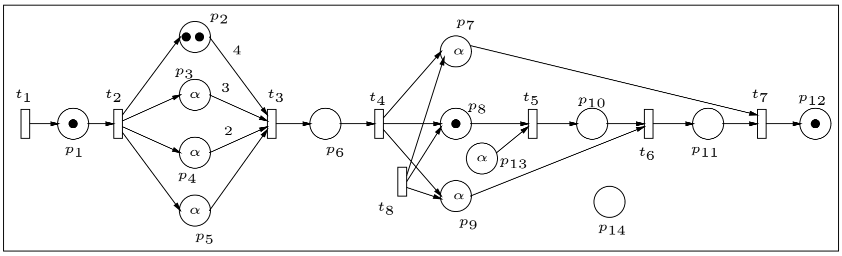


Figura 4.9: Red reconfigurable reversible marcada

- paquetes que evitan la transición t_5 y siguen el camino $t_4p_9t_6p_{11}t_7p_{12}$; y finalmente,
- paquetes que van directamente a través del camino $t_4p_7t_7p_{12}$.

El sistema descrito por la red reconfigurable reversible de la Figura 4.9 consiste en 14 lugares, 8 transiciones y 6 reglas de modificación de la estructura $R = \{r_1, r_2, r_3, r_4, r_5, r_6\}$ tales que:

$$\begin{array}{ll} r_1 = \{p_5\} \to \{p_2\} & r_2 = \{p_4\} \to \{p_3\} \\ r_3 = \{p_3\} \to \{p_2\} & r_4 = \{p_7, p_{13}\} \to \{p_8, p_{14}\} \\ r_5 = \{p_9\} \to \{p_7\} & r_6 = \{p_8, p_{14}\} \to \{p_9, p_{13}\} \end{array}$$

El marcado inicial representado en la Figura 4.9 es

$$M_0 = \{1, 2, \alpha, \alpha, \alpha, 0, \alpha, 1, \alpha, 0, 0, 1, \alpha, 0\}.$$

Esta red tiene 12 posibles configuraciones que se muestran en la Figura 4.10.

La correspondiente RPCA se muestra en la Figura 4.11. Los pesos de los arcos numerados en la Figura 4.11 son:

(1)	$S_0 + S_6 + S_7$	(2)	$4S_0 + 4S_6 + 4S_7$
(3)	$S_1 + S_4 + S_5$	(4)	$3S_1 + 3S_4 + 3S_6$
(5)	$S_3 + S_{10} + S_{11}$	(6)	$2S_3 + 2S_{10} + 2S_{11}$
(7)	$S_2 + S_8 + S_9$	(8)	$S_2 + S_8 + S_9$
(9)	$S_4 + S_6 + S_8 + S_{11}$	(10)	$S_0 + S_1 + S_2 + S_{10}$
(11)	$S_3 + S_5 + S_7 + S_9$	(12)	$S_4 + S_6 + S_8 + S_{11}$
(13)	$S_0 + S_1 + S_2 + S_{10}$	(14)	$S_3 + S_5 + S_7 + S_9$
(15)	$S_4 + S_6 + S_8 + S_{11}$	(16)	$S_0 + S_1 + S_2 + S_{10}$
(17)	$S_3 + S_4 + S_5 + S_6 + S_7 + S_8 + S_9 + S_{11}$	(18)	$S_3 + S_5 + S_7 + S_9$

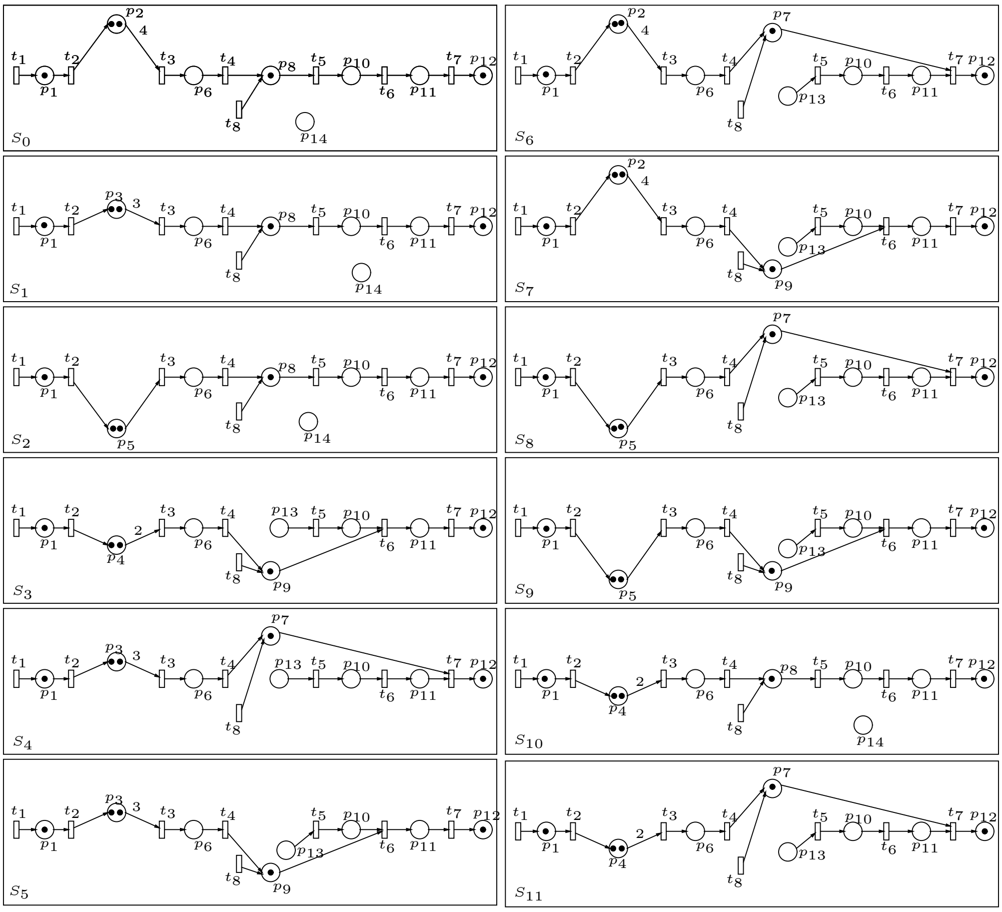


Figura 4.10: Las 12 configuraciones de la red reconfigurable reversible de la Figura 4.9

El autómata correspondiente que aquí no aparece puede calcularse a partir de el conjunto de configuraciones y el conjunto de reglas de modificación de la estructura.

4.3.2. Una fábrica de ensamblaje con diferentes modos de operación

Al contrario que en la sección anterior, el ejemplo aquí considerado requiere una sincronización de todo el sistema antes de que cualquier reconfiguración pueda tener lugar. Por esta razón, las redes reconfigurables no son una buena solución y optamos por una solución utilizando redes de Petri. El sistema es

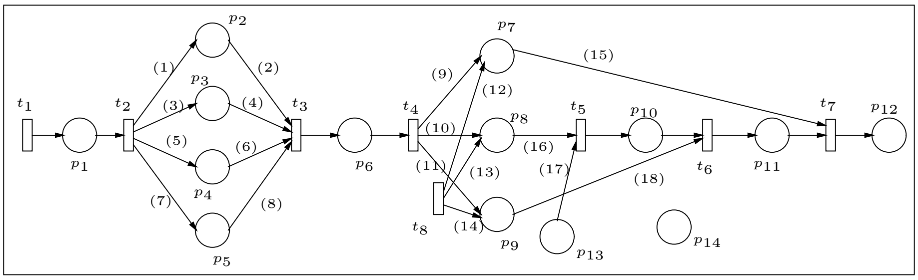


Figura 4.11: RPCA de la red reconfigurable reversible de la Figura 4.9

una fábrica que consiste en cadenas de montaje que ensamblan piezas a través de cintas transportadoras. Cada cadena de montaje está formada por distintas máquinas para procesar las piezas, almacenes y un robot que coge y deja las piezas de/en las cintas transportadoras y que carga y descarga máquinas y almacenes. Dependiendo del producto que la fábrica esté en ese momento fabricando, cada cadena de montaje sigue un modo distinto de operación. La Figura 4.12 muestra una cadena de montaje con tres modos distintos de operación. En el Modo I, el robot recibe una pieza de la cinta A, la carga en la máquina M_1 que la procesa de algún modo, y la pieza se almacena temporalmente hasta que pueda cargarse en la máquina M_2 para otra operación y entonces se envía a otra cadena de montaje a través de la cinta B.

Los lugares de la red de Petri asociada, a la izquierda de la Figura 4.12, se corresponden con las siguientes operaciones.

p_1 : la máquina M_1 espera una pieza
p_2 : el robot carga la máquina M_1 con una pieza recibida desde la cinta A
p_3 : la máquina M_1 trabajando
p_4 : la máquina M_1 preparada para descarga
p_5 : el robot descarga la máquina M_1 y almacena la pieza en el $Almacen_1$
p_6 : la máquina M_2 trabajando
p_7 : la máquina M_2 preparada para descarga
p_8 : el robot descarga la máquina M_2 y envía la pieza a la cinta B
p_9 : la máquina M_2 espera una pieza
p_{10} : el robot carga la máquina M_2 con una pieza del $Almacen_1$
A : piezas preparadas para cogerse desde la cinta A
B : piezas en la cinta B
C : piezas en la cinta C
D : número de espacios libres en el $Almacen_1$

E : número de piezas en el $Almacen_1$
F : número de espacios libres en el $Almacen_2$
G : número de piezas en el $Almacen_2$

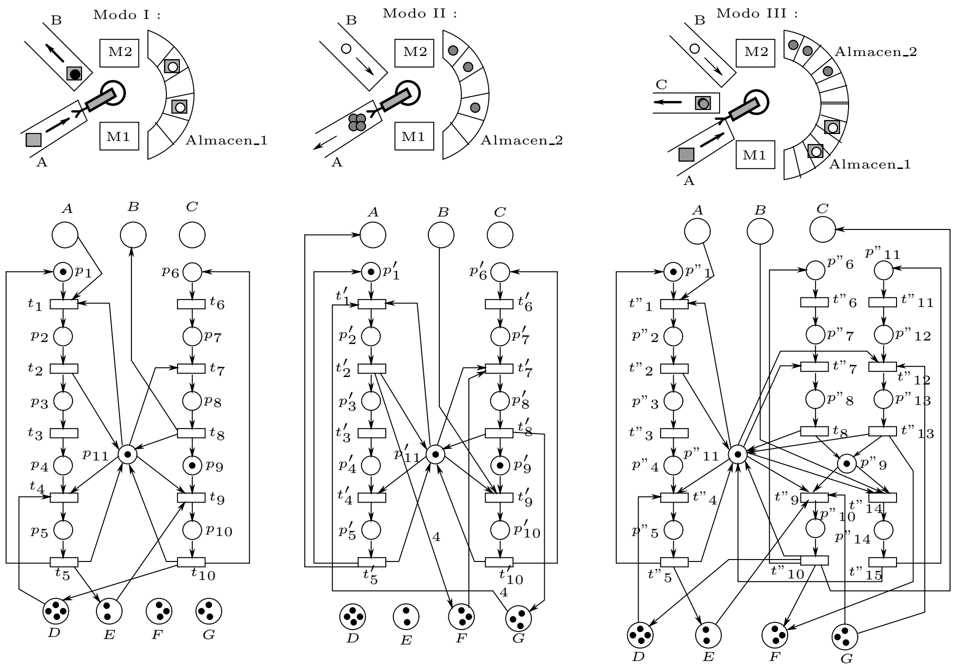


Figura 4.12: Fábrica de ensamblaje con diferentes modos de operación

Si nos olvidamos de los lugares A, B, C, D, E, F y G asociados a las cintas transportadoras y a los almacenes, la red de Petri consiste en tres componentes de una máquina de estados asociados respectivamente con las máquinas M_1 y M_2 y el robot, en particular, todos los lugares p_i son 1-seguros y pueden interpretarse como propiedades. Con lo que la propiedad $estable = p_1 \wedge p_{11} \wedge p_9$ que indica que las máquinas M_1 y M_2 están en su estado *idle*, esperando a que se les cargue una pieza, y el robot está disponible, es una propiedad *estable* que significa que desde cualquier marcado accesible es posible alcanzar algún marcado que verifique esta propiedad. Los otros dos modos de operación, que se muestran en la partes central y derecha de la Figura 4.12, se definen de manera similar. En el Modo II, las piezas provienen de la cinta B, se cargan primero en la máquina M_2 que lleva a cabo alguna transformación y se dejan en el $Almacen_2$. Si hay cuatro piezas disponibles pueden cogerse desde este almacén para ensamblarse en la máquina M_1 y formar una nueva pieza que se envía por la cinta A. En el Modo III, la máquina M_1 puede llevar a cabo la misma operación que en el Modo I sobre las piezas de la cinta A, la piezas resultantes se almacenan en el $Almacen_1$. La máquina M_2 puede realizar la misma operación que en el Modo II sobre las piezas de la cinta B, la piezas resultantes se almacenan en el $Almacen_2$. La máquina M_2 también puede ensamblar una pieza proveniente de cada almacén y enviar la pieza resultante por la cinta C. Nótese que cada almacén siempre contiene el mismo tipo de piezas, que la cinta C se utiliza sólo en el Modo III y que la dirección de las cintas transportadoras puede cambiar cuando se cambia de un modo de operación a otro.

Supóngase que un cambio en la fábrica requiere que la cadena de montaje cambie del Modo I al Modo II. Para no confundirse con distintos tipos de piezas y puesto que la dirección de algunas cintas transportadoras debe cambiar, es importante que, antes de que el cambio tenga lugar, el sistema se encuentre en un estado estable en el que cada máquina está idle sin ninguna pieza cargada y que no quede ninguna pieza en las cintas transportadoras, únicamente el contenido de los almacenes es irrelevante. Por esta razón, el cambio requiere una sincronización de todo el sistema. Para no complicar demasiado este ejemplo, asumiremos que existe un mecanismo para retirar piezas desde las cintas transportadoras cuando el sistema quiere realizar un cambio (las piezas se devuelven a la cinta correspondiente tan pronto como el sistema vuelva a la configuración adecuada). El cambio se lleva a cabo en dos pasos como se ilustra en la Figura 4.13.

Primero el sistema pide un cambio desde el Modo I al Modo II (transición I/II^+) moviendo el token desde el lugar I al lugar "I a II". Al hacer esto, las transiciones t_1 y t_9 que comprueban si el sistema está en el Modo I se inhiben, las máquinas M_1 y M_2 rechazan procesar ninguna nueva pieza antes de que el cambio tenga lugar. La red de Petri correspondiente al Modo I alcanzará necesariamente el estado estable cuando los lugares p_1, p_{11} y p_9 contengan un token cada uno (los otros lugares p_i están vacíos). El cambio puede entonces tener lugar (disparo de la transición I/II^-). Así, la red de Petri correspondiente al Modo I deja de estar activa ya que todos sus lugares p_i están vacíos mientras que la red de Petri correspondiente al Modo II se activa. Es obvio que las transiciones I/II^+ y I/II^- realizan cambios similares para todas aquellas cadenas de montaje que están implicadas en el cambio. Este ejemplo, aunque es bastante sencillo, nos sirve como muestra de cambios dinámicos en sistemas distribuidos.

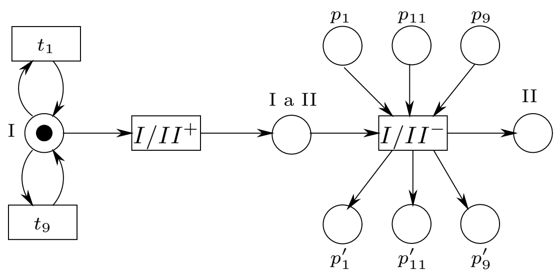


Figura 4.13: Cambio del modo I al modo II

4.3.3. Flujos en una red

Este último ejemplo es representativo para una gran clase de sistemas en los que los cambios derivan de las acciones de los agentes que regulan algún sistema distribuido. Las redes de Petri estratificadas parecen ser un formalismo que se adapta bien a este tipo de situaciones. Consideramos un sistema workflow que consiste en agentes intercambiando datos mediante paso asíncrono de mensajes. El sistema viene descrito por un grafo finito dirigido acíclico con un nodo fuente y un nodo sumidero. Los nodos del grafo representan agentes mientras que los arcos del grafo representan canales de comunicación entre ellos. Los diferentes caminos se corresponden con las distintas formas de llevar a cabo una tarea. Un flujo de este grafo da una división de caminos desde la fuente al sumidero. Para este propósito es conveniente añadir un arco ficticio ℓ_0 desde el sumidero a la fuente. Un flujo es una función que asigna pesos íntegros no negativos a cada arco, de forma que en cada nodo el peso total de los arcos entrantes es igual al peso total de los arcos salientes. Por ejemplo, el flujo representado en la Figura 4.14 se corresponde con cierta política de reparto para las tareas que entran en el sistema: por cada seis tareas que entran en el sistema, dos se dirigen al nodo 2 y cuatro al nodo 3; el nodo 5 recibe tres tareas desde el nodo 3 por cada tarea recibida desde el nodo 2, y estas cuatro tareas se dirigen todas al nodo 8; etc. Cada uno de tales flujos determina un ratio de carga para cada nodo, por ejemplo, en la Figura 4.14 el ratio de carga del nodo 8 es $\frac{4}{6} = \frac{2}{3}$, mientras que el ratio de carga del nodo 7 es $\frac{1}{6}$. En el caso de que se modifique la representación de algún nodo, el flujo actual debe modificarse. Por ejemplo decrementando en uno el peso de los arcos ℓ_4, ℓ_8 y ℓ_{11} y, al mismo tiempo, incrementando en uno el peso de los arcos ℓ_3, ℓ_7 y ℓ_{10}.

La diferencia entre dos flujos es un ciclo. La Figura 4.15 da un conjunto generador para los ciclos del grafo de la Figura 4.14. Por ejemplo, el ciclo σ_1 añade una unidad al peso de ℓ_2 y ℓ_5, y sustrae una unidad al peso de ℓ_1 y ℓ_4.

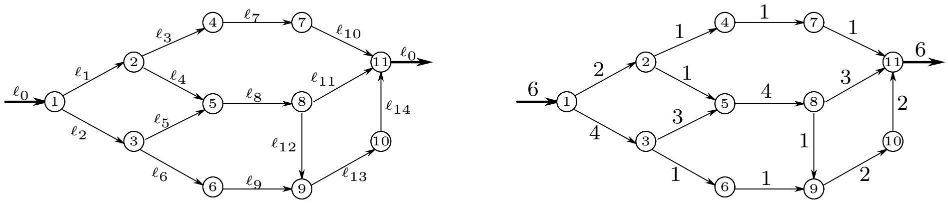
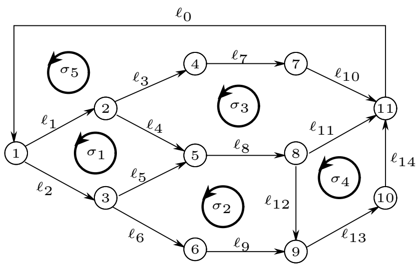


Figura 4.14: Un flujo (en la parte derecha) del sistema workflow (en la parte izquierda)



$$\sigma_1 = \ell_2 + \ell_5 - \ell_4 - \ell_1$$
$$\sigma_2 = \ell_6 + \ell_9 - \ell_{12} - \ell_8 - \ell_5$$
$$\sigma_3 = \ell_4 + \ell_8 + \ell_{11} - \ell_{10} - \ell_7 - \ell_3$$
$$\sigma_4 = \ell_{12} + \ell_{13} + \ell_{14} - \ell_{11}$$
$$\sigma_5 = \ell_1 + \ell_3 + \ell_7 + \ell_{10} + \ell_0$$

Figura 4.15: Un conjunto generador de ciclos

Los ciclos pueden verse como transiciones de una red de Petri cuyos lugares se asocian con los arcos del grafo. La Figura 4.16 muestra la red de Petri (en negrita y sobrepuesta en el grafo) resultante a partir del conjunto generador de la Figura 4.15. Se ha añadido un ciclo extra σ_6 y un lugar extra $\overline{\ell_0}$ cuyo contenido inicial da una cota superior del peso total de un flujo (es decir, el valor del arco ficticio ℓ_0). Se puede querer verificar que el conjunto de marcados accesibles de la Figura 4.16 es el conjunto de flujos del grafo cuyo peso total no excede el valor inicial de $\overline{\ell_0}$. Si dicho valor inicial es 10, como se indica en la figura, el grafo de marcado tiene ya 5005 estados y 21164 transiciones. Ahora el sistema completo puede modelizarse mediante una red estratificada formada por la red de Petri de la Figura 4.16 junto con una red de Petri paramétrica adjuntada a cada nodo del grafo, cuyos parámetros sean los lugares asociados con los arcos conectados a dicho nodo. La red de Petri asociada con el nodo n_5 viene dada, por ejemplo, en la Figura 4.17. Puede verse como una cadena de montaje, similar a la del ejemplo de la Sección 4.3.2, que recibe y envía paquetes cuyo tamaño viene dado por los valores de los parámetros. Cuando existen varios arcos resultantes, los paquetes se envían por turnos por cada una de las cintas transportadoras resultantes, respetando los ratios dados por el flujo actual.

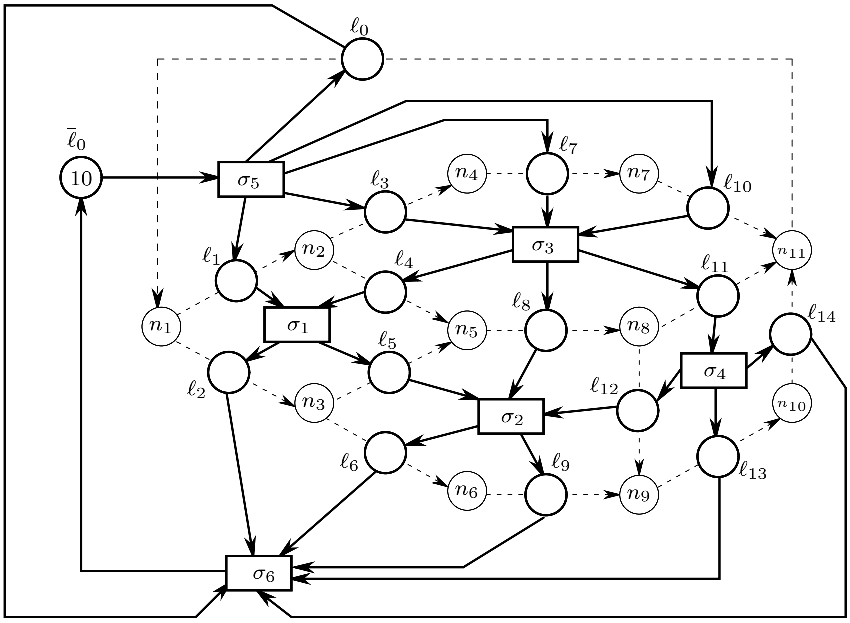
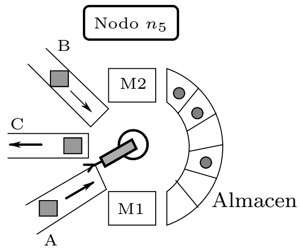
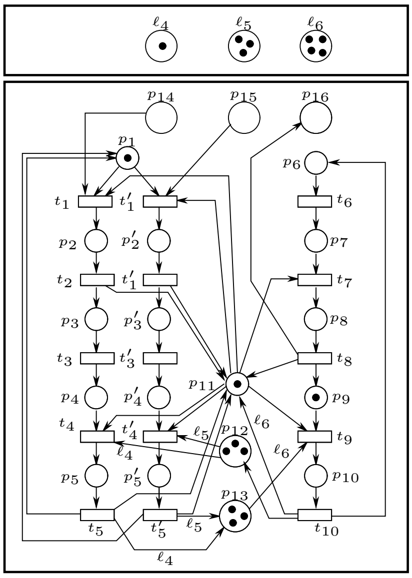


Figura 4.16: Red asociada al conjunto generador de ciclos de la Figura 4.15





p_1	:	la máquina M_1 espera un paquete
p_2, p'_2	:	el robot carga la máquina M_1 con un paquete de una de las dos cintas A o B
p_3, p'_3	:	la máquina M_1 descarga un paquete
p_4, p'_4	:	la máquina M_1 preparada para descargar
p_5, p'_5	:	el robot descarga la máquina M_1 y deja las piezas en el almacén
p_6	:	la máquina M_2 empaqueta las piezas en un nuevo paquete
p_7	:	la máquina M_2 preparada para enviar el paquete
p_8	:	el robot descarga la máquina M_2 y envía el paquete por la cinta transportadora C
p_9	:	la máquina M_2 espera nuevas piezas desde el almacén
p_{10}	:	el robot carga la máquina M_2 con piezas del almacén
p_{11}	:	el robot está disponible
p_{12}	:	número de huecos libres en el almacén
p_{13}	:	número de piezas en el almacén
p_{14}	:	número de paquetes en la cinta transportadora A
p_{15}	:	número de paquetes en la cinta transportadora B
p_{16}	:	número de paquetes en la cinta transportadora C

Figura 4.17: Comportamiento de un nodo de la red

Capítulo 5

Sistemas de Reescritura de Redes. Redes Reconfigurables

En el Capítulo 4 se ha presentado una clase de redes de Petri de alto nivel, denominadas *redes reconfigurables*. Dichas redes pueden cambiar dinámicamente su propia estructura reescribiendo alguno de sus componentes; de esta forma, se pueden permitir cambios dinámicos dentro de los sistemas concurrentes y distribuidos. En particular, estas redes pueden utilizarse para describir los *sistemas workflow* [EN93][VDA97, VDAVH02]. En concreto, una red de Petri reconfigurable es una red de Petri con reglas de modificación de la estructura locales que reemplazan una de sus subredes por otra subred, transfiriéndose los tokens de los lugares eliminados a los lugares creados.

En el Capítulo 4 se ha presentado también una subclase de las redes reconfigurables, llamadas *reversibles*, cuyas reglas de modificación de la estructura son biyecciones y para las que hemos demostrado que las propiedades de acotabilidad, alcanzabilidad y vivacidad son decidibles. La reversibilidad es bastante natural en los sistemas en los que estamos interesados ya que el sistema que se quiere modelizar se supone que se ejecutará siempre y, por tanto, su comportamiento no se degradará: reversibilidad significa que desde cualquier configuración accesible debe poder alcanzarse la configuración inicial; si el sistema no lo permite significa que puede perder alguna de sus habilidades (ese es el sentido de "comportamiento degradado") pero estamos interesados únicamente en los comportamientos asintóticos del sistema y cada uno de ellos es reversible. Esta asunción extra de reversibilidad permitirá reformular la definición de redes reconfigurables de una manera más directa y cercana a lo descrito en la Sección 1 del Capítulo 2 (redes automodificantes de R. Valk). Es decir, un sistema se modelizará como un grupo de redes de Petri, llamadas sus *configuraciones*, junto con un mecanismo que permita al sistema evolucionar de una configuración a otra bajo ciertas circunstancias.

La definición de Sistemas de Reescritura de Redes que introducimos en la Sección 5.1 de este capítulo, combina redes de Petri con sistemas de reescritura de grafos. Una configuración del sistema se describe como una red de Petri y un cambio de configuración como una regla de reescritura de grafos (equivalente a una producción de las gramáticas de grafos, con parte izquierda, parte derecha e interfaz). Dicho cambio de configuración va a consistir en reemplazar parte del sistema (que empareja -hace "matching"- con la parte izquierda de la regla de reescritura) por otro grafo (que viene dado por la parte derecha de la regla de reescritura), lo que es similar a las derivaciones directas en las gramáticas de grafos.

Introducimos también la definición de red reconfigurable, en la Sección 5.2, como un sistema de reescritura de redes en el que un cambio de configuración se limita a modificar las relaciones de flujo de los lugares implicados en la regla de reescritura. Veremos, en la Sección 5.4, que la traducción de este modelo a redes de Petri es automática. Esta equivalencia asegura que todas las propiedades fundamentales de las redes de Petri siguen siendo decidibles para las redes reconfigurables y, por tanto, para este modelo es posible obtener herramientas de verificación automática. Por el contrario, demostramos, en la Sección 5.3, que la clase de los sistemas de reescritura de redes tiene el poder expresivo de la máquina de Turing con lo que la construcción de herramientas de verificación automática no es posible para dicha clase. Por último, en la Sección 5.5 presentamos un ejemplo de modelización, mediante una red reconfigurable, de un sistema distribuido en el que se dan cambios dinámicos locales.

5.1. Sistema de Reescritura de Redes

En esta sección se introduce el modelo de los *Sistemas de Reescritura de Redes*. Para ello necesitamos las siguientes definiciones previas: Si $R \subset X \times Y$ es una relación binaria, se dice que $X'R = \{y \in Y \mid \exists x \in X' \ (x, y) \in R\}$ denota la *imagen* de $X' \subseteq X$ y $RY' = \{x \in X \mid \exists y \in Y' \ (x, y) \in R\}$ denota la *imagen inversa* de $Y' \subseteq Y$; el *dominio* de R es entonces $Dom(R) = RY$ y el *codominio* de R es $Cod(R) = XR$.

Definición 144 (Sistema de Reescritura de Redes (SRR)) *Un* sistema de reescritura de redes *es una estructura* $N = (\mathcal{R}, \Gamma_0, M_0)$ *donde* $\mathcal{R} = \{r_1, \ldots, r_h\}$ *es un conjunto finito de reglas de reescritura,* $\Gamma_0 = (P_0, T_0, F_0)$ *es una red de Petri y* $M_0 : P_0 \to \mathbb{N}$ *es un marcado asociado a* Γ_0*.*

Definición 145 (Regla de Reescritura para SRRs) *Una* regla de reescritura $r \in \mathcal{R}$ *es una estructura* $r = (L, R, \tau, {}^\bullet\tau, \tau^\bullet)$ *donde*

1. $L = (P_L, T_L, F_L)$ *y* $R = (P_R, T_R, F_R)$ *son redes de Petri llamadas* parte izquierda *y* parte derecha *de* r*, respectivamente.*

2. $\tau \subseteq (P_L \times P_R) \cup (T_L \times T_R)$*, llamada* relación de transferencia *de* r*, es una relación binaria que relaciona lugares de* L *con lugares de* R *y transiciones de* L *con transiciones de* R*:*

$$P_L\tau \subseteq P_R \qquad \tau P_R \subseteq P_L \qquad T_L\tau \subseteq T_R \qquad \tau T_R \subseteq T_L$$

3. ${}^\bullet\tau \subseteq \tau$*, y* $\tau^\bullet \subseteq \tau$ *son subrelaciones de las relaciones de transferencia llamadas* relación interfaz de entrada *y* relación interfaz de salida*, respectivamente.*

Definición 146 (Configuración de un SRR) *Una* configuración *de un sistema de reescritura de redes* N *es una red de Petri* $\Gamma = (P, T, F)$*.*

Definición 147 (Estado de un SRR) *Un* estado *de un sistema de reescritura de redes* N *es un par* (Γ, M) *formado por una red de Petri* $\Gamma = (P, T, F)$ *(configuración) y un marcado* $M : P \to \mathbb{N}$ *para* Γ*, es decir, es una red de Petri marcada. El par* (Γ_0, M_0) *es el* estado inicial *del sistema de reescritura de redes.*

Definición 148 (Evento de un SRR) *Un* evento *de un sistema de reescritura de redes es o bien una transición o bien una regla de reescritura:* $E = T \cup R$*.*

Presentamos a continuación un ejemplo sencillo para ilustrar esta definición.

Ejemplo 149 (Impresoras) *El sistema de reescritura de redes de la Figura 5.1 modeliza la impresión de varias copias de un mismo trabajo en distintas impresoras. El "token" en el lugar* buffer_printer *representa una copia del trabajo a imprimir. En este estado* (Γ, M)*, para obtener tres copias, el trabajo debe enviarse a imprimir tres veces y estas tres copias sólo pueden imprimirse secuencialmente (una detrás de otra). La regla de reescritura* triple *ofrece la posibilidad de imprimir cada una de ellas en una impresora distinta. La relación de transferencia* τ *viene dada por* $\tau = \{(\{p\_ini\}, \{p\_1, p\_2, p\_3\}), (\{t\_0\}, \{t\_1, t\_2, t\_3\}), (\{p\_end\}, \{p\_end\})\}$ *y las relaciones interfaz de entrada e interfaz de salida son, en este caso,* ${}^{\bullet}\tau = \{(\{p\_ini\}, \{p\_1, p\_2, p\_3\})\}$ *y* $\tau^{\bullet} = \{(\{p\_end\}, \{p\_end\})\}$*, respectivamente.*

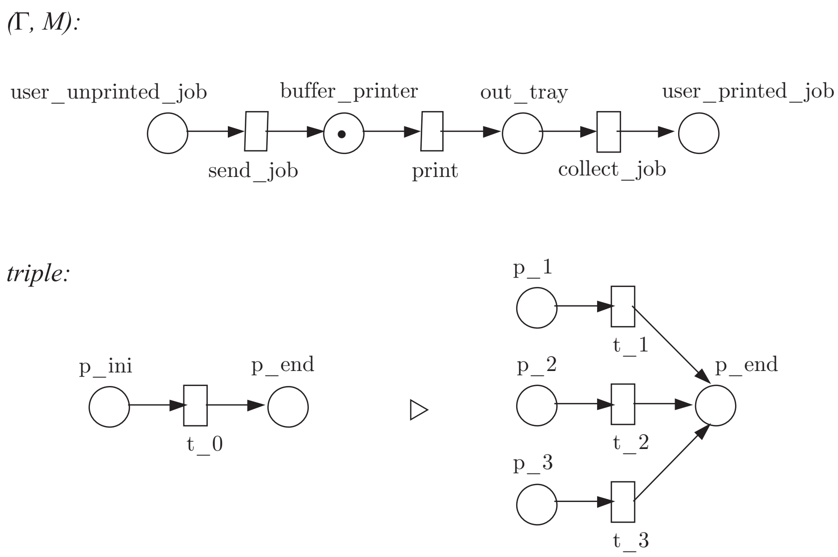


Figura 5.1: Sistema de Reescritura de Redes modelizando un sistema de impresoras

Definición 150 (Redes de Petri Isomorfas) *Sean* $\mathcal{N} = (P, T, F)$ *y* $\mathcal{N}' = (P', T', F')$ *dos redes de Petri. Decimos que* $\mathcal{N}$ *y* $\mathcal{N}'$ *son* isomorfas *si existe una biyección* $\varphi : (P \cup T) \to (P' \cup T')$ *tal que* $F(x, y) \in \mathcal{N} = F'(\varphi(x), \varphi(y)) \in \mathcal{N}'$ *para todo* $x, y \in P \cup T$.

Para poder aplicar una regla de reescritura r en una configuración Γ debemos:

1. Identificar una subred total Γ' de Γ que sea isomorfa a la parte izquierda de la regla. Los elementos de Γ (lugares o transiciones) que no pertenezcan a Γ' constituyen el *contexto* de la regla.

2. Para que una regla esté habilitada, además se requiere que un elemento x' de Γ' tenga un elemento x de su *preset* que pertenezca al contexto sólo si x' pertenece al *interfaz de entrada* de la regla y, simétricamente, que un elemento x' de Γ' tenga un elemento x de su *postset* que pertenezca al contexto sólo si x' pertenece al *interfaz de salida* de la regla.

3. Cuando estas condiciones se dan, la reescritura puede tener lugar y se reemplaza la subred Γ' por la parte derecha R de la regla, fijándose las conexiones entre los elementos de R y los del contexto de acuerdo con la relación de interfaz. La relación de transferencia no sólo se utiliza para reescribir la parte izquierda por la parte derecha de la regla sino también para transferir los tokens de Γ' a R (de ahí su nombre). Nótese que, ya que la relación de transferencia puede ser cualquier tipo de relación, los tokens pueden duplicarse o pueden, incluso, desaparecer.

Definición 151 (Embedding total) *Llamamos* embedding total *de una red de Petri* $\Gamma = (P, T, F)$ *en una red de Petri* $\Gamma' = (P', T', F')$ *a una función inyectiva* $f : P \cup T \to P' \cup T'$ *que hace corresponder lugares con lugares y transiciones con transiciones (*$f(P) \subseteq P'$ *y* $f(T) \subseteq T'$*) tal que para cualquier par de elementos* $x, y \in P \cup T$ *se tiene que* $F(x, y) = F'(f(x), f(y))$*. La imagen de* Γ *por* f *se denomina entonces* subred total *de* Γ'.

Definición 152 (Grafo de estados de un SRR) *El* grafo de estados *de* un sistema de reescritura de redes $N = (\mathcal{R}, \Gamma_0, M_0)$ *es el grafo dirigido etiquetado cuyos nodos son los estados de* N*, es decir, redes de Petri marcadas y cuyos arcos (etiquetados con eventos de* N*) son de dos clases:*

- disparo de una transición: *arcos desde el estado* (Γ, M) *al estado* (Γ', M') *etiquetados con la transición* t *cuando la transición* t *puede dispararse en la red con marcado* M *y lleva al marcado* M'*:*

$$(\Gamma, M) \xrightarrow{t} (\Gamma', M') \Longleftrightarrow (\Gamma = \Gamma' \;\wedge\; M[t\rangle M' \text{ en } \Gamma)$$

- cambio de configuración: *arcos desde el estado* (Γ, M) *al estado* (Γ', M') *etiquetados con la regla* $r = (L, R, \tau, {}^{\bullet}\tau, \tau^{\bullet}) \in \mathcal{R}$*, cuando existe un embedding total* $f : L \to \Gamma$ *tal que para todo* $x \notin f(L)$ *e* $y \in L$ *se tiene*

$$x \in {}^{\bullet}f(y) \Rightarrow y \in Dom({}^{\bullet}\tau) \text{ y } x \in f(y)^{\bullet} \Rightarrow y \in Dom(\tau^{\bullet})$$

y, siendo $\Gamma = (P, T, F)$ *y* $\Gamma' = (P', T', F')$*, se cumple lo siguiente:*

$$P' = P - f(P_L) + P_R \textit{ tal que } P_L\tau = P_R$$

$$T' = T - f(T_L) + T_R \textit{ tal que } T_L\tau = T_R$$

donde el significado de +(-) es añadir(eliminar) lugares/transiciones a(de) Γ*. Los nombres de los lugares* P_R*(transiciones* T_R*) añadidos a* Γ *deben ser nombres nuevos para evitar conflictos.*

$$M'(p) = \begin{cases} M(p) & \textit{si } p \notin R \\ \sum_{p' \in \ ^{\tau}p} M(p') & \textit{si } p \in R \end{cases}$$

$$F'(x,y) = \begin{cases} F(x,y) & \textit{si } x \notin R \ \wedge \ y \notin R \\ F_R(x,y) & \textit{si } x \in R \ \wedge \ y \in R \\ \sum_{y_i \in \ ^{\bullet\tau}y} F(x, f(y_i)) & \textit{si } x \notin R \ \wedge \ y \in R \\ \sum_{x_i \in \ ^{\tau\bullet}x} F(f(x_i), y) & \textit{si } x \in R \ \wedge \ y \notin R \end{cases}$$

Ejemplo 153 (Impresoras) *La Figura 5.2 muestra el nuevo estado* (Γ', M') *debido al cambio de configuración causado por la regla de reescritura* triple *en el Ejemplo 149. Como puede apreciarse, se ha reemplazado una subred por otra y el número de tokens ha aumentado (se ha triplicado).*

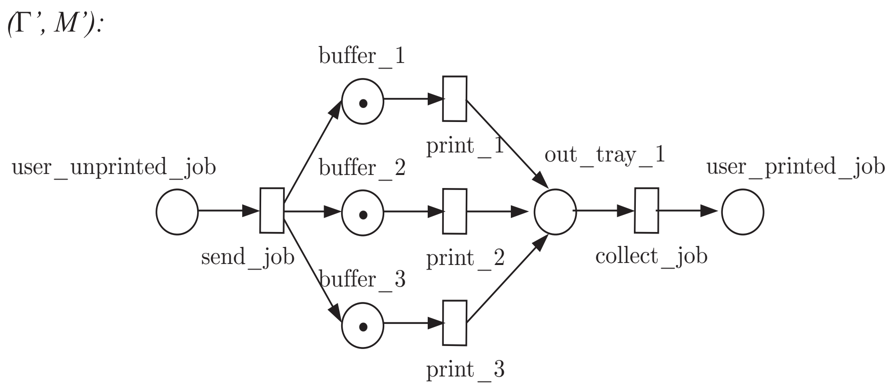


Figura 5.2: Cambio de configuración debido a la regla de reescritura *triple*

5.2. Redes Reconfigurables

En esta sección se presenta una clase específica de sistemas de reescritura de redes que se corresponde con el modelo de redes reconfigurables introducidas en el Capítulo 4. Más exactamente, la definición que sigue es una reformulación de lo que allí se llamaba red reconfigurable *reversible*.

Definición 154 (Red Reconfigurable (Primera versión)) *Una* red reconfigurable *es un sistema de reescritura de redes tal que:*

1. *la relación de transferencia τ es una biyección y*
2. *las interfaces $\tau^\bullet = {}^\bullet\tau$ son restricciones de la relación de transferencia τ al conjunto de transiciones:*

$$\tau^\bullet = {}^\bullet\tau = T_L \times T_R \cap \tau$$

La primera condición afirma que el conjunto de lugares y transiciones no sufre cambios por la aplicación de reglas de reescritura. La segunda condición asegura que un cambio de configuración implica la modificación de las relaciones de flujo de los lugares en el dominio de la regla implicada, de acuerdo a esa regla e independientemente del contexto en el que se aplique esta reescritura. Así pues, la definición anterior puede reformularse de manera directa (esto es, sin recurrir a sistemas de reescritura de redes) utilizando la siguiente definición que, a partir de ahora, consideraremos como la definición final de redes reconfigurables (ya que la reversibilidad se asumirá siempre implícitamente).

Definición 155 (Red Reconfigurable (RR)) *Una* red reconfigurable *es una estructura $N = (P, T, \mathcal{R}, \gamma_0)$ siendo $P = \{p_1, \ldots, p_n\}$ un conjunto finito y no vacío de lugares, $T = \{t_1, \ldots, t_m\}$ un conjunto finito y no vacío de transiciones disjunto de P ($P \cap T = \emptyset$), $\mathcal{R} = \{r_1, \ldots, r_h\}$ un conjunto finito de reglas de reescritura, y γ_0 el estado inicial.*

Definición 156 (Regla de Reescritura para RRs) *Una* regla de reescritura *$r \in \mathcal{R}$ es una estructura $r = (D, {}^\bullet r, r^\bullet)$ donde*

- *$D \subseteq P$ es el* dominio *de r,*

- ${}^\bullet r : (D \times T) \cup (T \times D) \to \mathbb{N}$ *y* $r^\bullet : (D \times T) \cup (T \times D) \to \mathbb{N}$ *son la* precondición *y* postcondición *de* r*, respectivamente. Es decir, son las relaciones de flujo de los lugares del dominio antes y después del cambio de configuración debido a la regla* r*.*

Definición 157 (Configuración de una RR) *Una* configuración *de una red reconfigurable es una red de Petri* $\Gamma = (P, T, F)$*. Por* $Conf(N)$ *se denota el* conjunto de configuraciones *de* N*.*

Definición 158 (Estado de una RR) *Un* estado γ *de una red reconfigurable es un par* $\gamma = (\Gamma, M)$ *donde* Γ *es una configuración y* $M : P \to \mathbb{N}$*. Por tanto, un estado es una red de Petri marcada asociada con el conjunto de lugares y transiciones de la red reconfigurable.*

Definición 159 (Evento de una RR) *Los* eventos *de una red reconfigurable son sus transiciones junto con sus reglas de reescritura:* $E = T \cup \mathcal{R}$*.*

Notación 160 *Representamos una regla de reescritura utilizando la notación de sumas formales como*

$$r = \sum_{p \in D} p((\sum_{t \in T} {}^\bullet r(p,t) \cdot t, \sum_{t \in T} {}^\bullet r(t,p) \cdot t)) \rhd \sum_{p \in D} p((\sum_{t \in T} r^\bullet(p,t) \cdot t, \sum_{t \in T} r^\bullet(t,p) \cdot t))$$

Un caso particular de redes reconfigurables que puede ser interesante para la resolución de problemas específicos son las redes reconfigurables puras.

Definición 161 (Red Reconfigurable Pura) *Una red reconfigurable se llama pura si su estado inicial* $\gamma_0 = (P, T, F_0, M_0)$ *es una red de Petri pura, es decir,*

$$\forall p \in P \ \forall t \in T \quad F(p,t) \neq 0 \Rightarrow F(t,p) = 0$$

y la precondición y la postcondición de cada una de las reglas de reescritura son también redes de Petri puras, es decir,

$$\forall r = (D, {}^\bullet r, r^\bullet) \ \forall p \in D \ \forall t \in T \begin{cases} {}^\bullet r(p,t) \neq 0 \Rightarrow {}^\bullet r(t,p) = 0 \\ r^\bullet(p,t) \neq 0 \Rightarrow r^\bullet(t,p) = 0 \end{cases}$$

Notación 162 *Para redes reconfigurables puras, la notación anterior puede simplificarse a la siguiente*

$$r = \sum_{p\in D} p(\sum_{t\in T} {}^{\circ}r(p,t)\cdot t) \triangleright \sum_{p\in D} p(\sum_{t\in T} r^{\circ}(p,t)\cdot t)$$

siendo ${}^{\circ}r(p,t) = {}^{\bullet}r(t,p) - {}^{\bullet}r(p,t)$ *y de igual forma* $r^{\circ}(p,t) = r^{\bullet}(t,p) - r^{\bullet}(p,t)$.

Definición 163 (Grafo de Configuración de una RR) *El* grafo de configuración $G(N)$ *de una red reconfigurable* $N = (P, T, \mathcal{R}, \gamma_0)$ *es el grafo dirigido etiquetado cuyos nodos son las configuraciones, y tal que existe un arco desde la configuración* Γ *a la configuración* Γ' *etiquetado con la regla* $r = (D, {}^{\bullet}r, r^{\bullet}) \in \mathcal{R}$, *que denotamos* $\Gamma[r\rangle\Gamma'$, *si y solamente si se cumple lo siguiente:*

$$\forall p \in D \quad \begin{cases} F(p,t) = {}^{\bullet}r(p,t) \text{ y } F(t,p) = {}^{\bullet}r(t,p) \\ F'(p,t) = r^{\bullet}(p,t) \text{ y } F'(t,p) = r^{\bullet}(t,p) \end{cases}$$

$$\forall p \notin D \qquad F(p,t) = F'(p,t) \text{ y } F(t,p) = F'(t,p)$$

La evolución dinámica de una red reconfigurable viene dada por su grafo de estados.

Definición 164 (Grafo de Estados de una RR) *El* grafo de estados *de una red reconfigurable* $N = (P, T, \mathcal{R}, \gamma_0)$ *es el grafo dirigido etiquetado cuyos nodos son los estados de* N *y cuyos arcos (etiquetados con eventos) son de dos clases:*

- disparo de una transición: *arcos desde el estado* (Γ, M) *al estado* (Γ, M') *etiquetados con la transición* t *cuando la transición* t *puede dispararse en la red* Γ *con marcado* M *y conduce al marcado* M'.
- cambio de configuración: *arcos desde el estado* (Γ, M) *al estado* (Γ', M) *etiquetados con la regla* $r \in \mathcal{R}$ *si* $\Gamma[r\rangle\Gamma'$ *es una transición del grafo de configuración de* N.

Es decir, el conjunto de transiciones (arcos etiquetados) del grafo de estados de N *viene dado por*

$$\{(\Gamma, M) \xrightarrow{t} (\Gamma, M') \mid M[t\rangle M' \text{ en } \Gamma\} \cup \{(\Gamma, M) \xrightarrow{r} (\Gamma', M) \mid \Gamma[r\rangle\Gamma' \text{ en } G(N)\}$$

A partir de la definición anterior, es obvio que la evolución de un sistema modelado mediante una red reconfigurable depende de qué evento ocurra. Si se dispara una transición, sólo se ve afectado el marcado de la red (como en las redes de Petri). En cambio, si se aplica una regla de reescritura, lo que se modifica es la estructura de la red manteniéndose el marcado original, es decir, los cambios de configuración son ortogonales al comportamiento de la red subyacente. Por otro lado, en el caso general de los sistemas de reescritura de redes, la aplicación de una regla de reescritura no sólo afecta a la estructura de la red sino también al marcado de la red y, por tanto, el comportamiento es distinto al de la red que subyace. No obstante, en la medida de que en este caso sólo se dispone de herramientas de modelización y simulación y que no es difícil restringir el comportamiento para condicionar los cambios de configuración a ciertas propiedades locales de los sistemas, será siempre posible tener en cuenta dichas restricciones, al menos a nivel de simulador.

Todo cambio de configuración que puede tener lugar en una red reconfigurable se representa mediante una regla de reescritura. Para definir estas reglas de una forma genérica, agrupamos lugares y transiciones de acuerdo a una decisión del diseñador. Esto no cambia la naturaleza del modelo considerado pero facilita su descripción.

Definición 165 (Rol) *Decimos que dos lugares (transiciones) tienen el mismo* rol *si el diseñador los agrupa en la misma clase de lugar (transición).*

Ejemplo 166 (Productores/Consumidores) *La Figura 5.3 representa dos líneas de comunicación* half-duplex *entre dos productores/consumidores.*

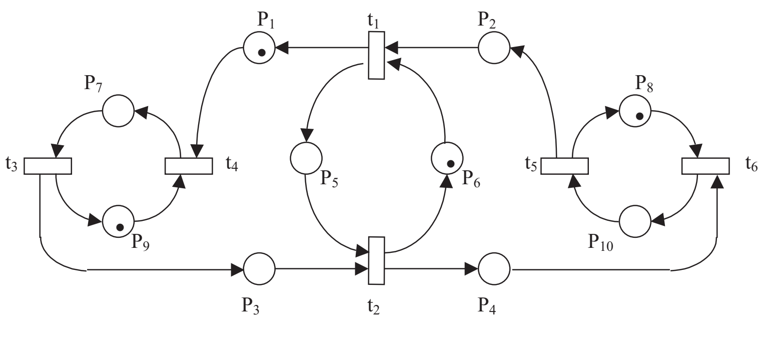


Figura 5.3: Sistema con dos líneas *half-duplex* entre dos productores/consumidores

En esta red reconfigurable van a poder tener lugar varios cambios de configuración. Cada uno de ellos se representa mediante una regla de reescritura. En su definición, distinguimos los roles P, Q y S entre lugares y los roles A y B entre transiciones.

$$\begin{array}{ll} P = \{P_1, P_2, P_3, P_4\} & A = \{t_1, t_2\} \\ Q = \{P_5, P_6\} & B = \{t_3, t_4, t_5, t_6\} \\ S = \{P_7, P_8, P_9, P_{10}\} & \end{array}$$

Definimos cuatro reglas de reescritura:

1. *Cambio de modo secuencial a paralelo:*

$$R_1 : Q(A - A) + Q(-A + A) \rhd Q(\emptyset) + Q(\emptyset)$$

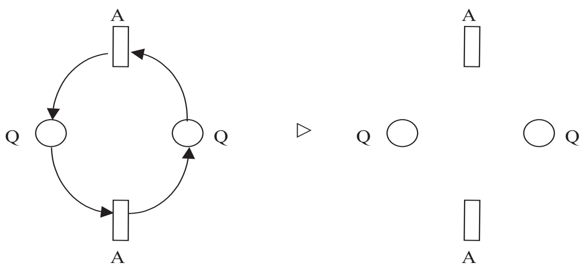


Figura 5.4: Regla de reescritura R_1

2. *Cambio de sentido en una línea de comunicación:*

$$R_2 : P(-A) + P(A) \rhd P(A) + P(-A)$$

Figura 5.5: Regla de reescritura R_2

3. *Cambio de productor/consumidor a consumidor:*

$$
\begin{aligned}
R_3 : & S(B - B) + P(-B) + S(-B + B) + P(B) \\
& \rhd S(B - B) + P(B) + S(-B + B) + P(\emptyset)
\end{aligned}
$$

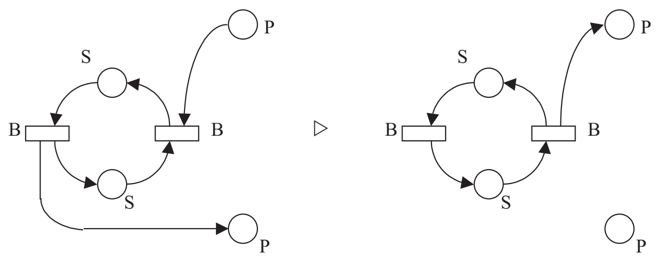


Figura 5.6: Regla de reescritura R_3

4. *Cambio de productor/consumidor a productor:*

$$
\begin{aligned}
R_4 : & S(B - B) + P(-B) + S(-B + B) + P(B) \\
& \rhd S(B - B) + P(-B) + S(-B + B) + P(\emptyset)
\end{aligned}
$$

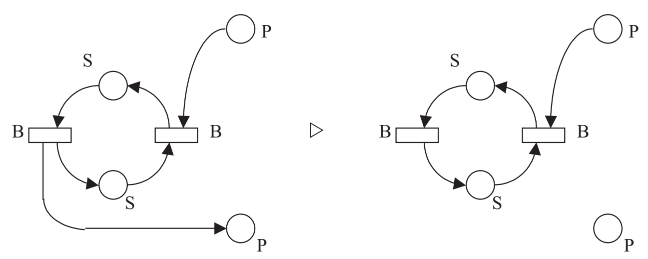


Figura 5.7: Regla de reescritura R_4

Por ejemplo, si cambiamos de modo secuencial a paralelo, en la red en el estado de la Figura 5.3 aplicamos la regla de reescritura R_1 y obtenemos el siguiente estado (Figura 5.8) en el que únicamente las relaciones de flujo han cambiado (el marcado es el marcado original).

La Figura 5.9 muestra parte del grafo de estados de la red reconfigurable del ejemplo.

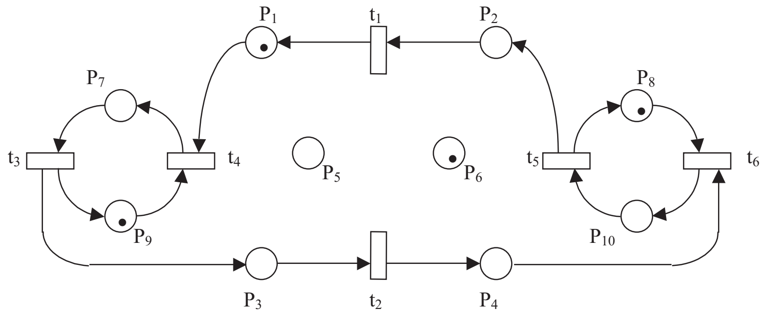


Figura 5.8: Estado obtenido tras aplicar la regla R_1 al estado representado en la Figura 5.3

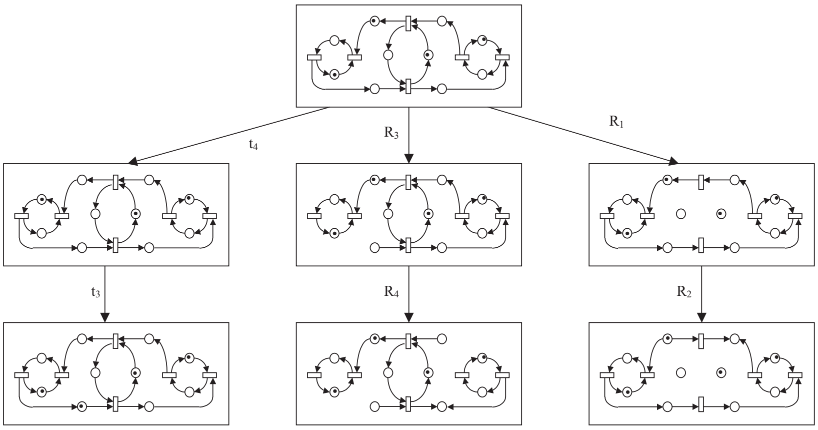


Figura 5.9: Fragmento del grafo de estados del sistema de la Figura 5.3

Podemos observar que, dependiendo de qué evento tenga lugar, cambian el marcado o las relaciones de flujo de la red. El disparo de una transición afecta al marcado de la red, mientras que el disparo de una regla de reescritura deja el marcado intacto y cambia la estructura de la red modificando sus relaciones de flujo.

5.3. Simulación de Máquinas de Turing con Sistemas de Reescritura de Redes

En esta sección vamos a demostrar que, si relajamos ligeramente la restricción sobre las reglas reescritura permitiendo que sean biyecciones parciales en lugar de biyecciones totales, obtenemos una clase de sistemas de reescritura de redes que ya tiene la expresividad de la máquina de Turing. Recordemos la definición de biyección parcial.

Definición 167 (Biyección parcial) *Una* función parcial *de un conjunto X es aquélla cuyo dominio es un subconjunto de X, es decir, hay algunos elementos de X que no tienen imagen. Una relación $\rho \subseteq X \times Y$ es una* biyección parcial *si induce una biyección cuando se restringe a su dominio y codominio, es decir, cada elemento de X tiene como mucho una imagen en Y y cada elemento de Y tiene como mucho una co-imagen en X.*

Que la relación de transferencia τ sea una biyección parcial significa que:

- hay un subconjunto de $(P_L \cup T_L)$ que es una biyección y
- que sólo los elementos de ese subconjunto tienen imagen.

Así, hay elementos de $(P_L \cup T_L)$ que no tienen imagen y hay elementos de $(P_R \cup T_R)$ que no tienen dominio.

La prueba de que las máquinas de Turing puedan simularse mediante sistemas de reescritura de redes cuyas relaciones de transferencia son biyecciones parciales es directa si se tiene en cuenta que el modelo de los sistemas de reescritura de redes no cambia si añadimos restricciones de orden sobre lugares y transiciones. En lo que sigue, esto se demuestra formalmente.

5.3.1. Sistemas de Reescritura de Redes K-ordenadas

En esta sección se presentan los *sistemas de reescritura de redes K-ordenadas*, una extensión de los sistemas de reescritura de redes sobre cuyos lugares y transiciones se han añadido ciertas restricciones de orden y que van a ser los sistemas de reescritura de redes utilizados en la siguiente sección para simular a una máquina de Turing.

Definición 168 (Restricción) *Una red $\Gamma = (P, T, F)$ es la* restricción *de una red $\Gamma' = (P', T', F')$ si $P \subseteq P'$, $T \subseteq T'$ y $F = F' \cap (P \times T \cup T \times P)$; se denota $\Gamma \sqsubseteq \Gamma'$.*

Un sistema de reescritura de redes $N = (\mathcal{R}, \Gamma_0, M_0)$ es la restricción de un sistema *$N' = (\mathcal{R}', \Gamma'_0, M'_0)$ si:*

1. $\Gamma_0 \sqsubseteq \Gamma'_0$ *y*
2. $\exists$ *una biyección* $\varphi : \mathcal{R} \rightarrow \mathcal{R}'$ *tal que* $\forall r \in \mathcal{R}$ $L_r \sqsubseteq L'_r$ *y* $R_r \sqsubseteq R'_r$*; se denota* $N \sqsubseteq N'$.

Si $N \sqsubseteq N'$*, la* restricción del grafo de estados *de* N' *a* N *es el sistema de transición* $(\Gamma, M) \xrightarrow{x} (\Gamma', M')$ *tal que*

- Γ *y* Γ' *son las configuraciones de* N*,*
- $x \in T \cup \mathcal{R}$*,*
- M *y* M' *son los marcados de* Γ *y* Γ'*, respectivamente y*
- $\exists(\widetilde{\Gamma}, \widetilde{M}) \xrightarrow{\widetilde{x}} (\widetilde{\Gamma'}, \widetilde{M'})$ *una transición del grafo de estados de* N' *tal que:*
  - $\widetilde{x} = \begin{cases} x & \text{si } x \in T_\Gamma \\ \varphi_x & \text{si } x \in \mathcal{R} \end{cases}$,
  - $\Gamma \sqsubseteq \widetilde{\Gamma}$ *y* $\Gamma' \sqsubseteq \widetilde{\Gamma'}$*,*
  - $M = \widetilde{M}$ *restringido a los lugares de* Γ *y* $M' = \widetilde{M'}$ *restringido a los lugares de* Γ'.

Definición 169 (Extensión conservativa) N' *es una* extensión conservativa *de* N *si* $N \sqsubseteq N'$ *y el grafo de estados de* N *coincide (es isomorfo) con la restricción del grafo de estados de* N' *a* N.

Definición 170 (Red K-ordenada) *Una* red K-ordenada *es una red de Petri* (P, T, F) *y una aplicación* $\kappa : P \cup T \rightarrow K$ *en la que* K *es un conjunto finito de redes de Petri* $K = \{k_1, \ldots, k_m\}$ *tal que* $\forall k_i = (P_i, T_i, F_i) \in K$: $F_i(P_i \times T_i \cup T_i \times P_i) > F(P \times T \cup T \times P)$.

Definición 171 (Sistema de Reescritura de Redes K-ordenadas) *Un* sistema de reescritura de redes K-ordenadas *es un sistema de reescritura de redes* $N = (\mathcal{R}, \Gamma_0, M_0)$ *donde:*

- Γ_0 *es un red K-ordenada,*

- M_0 *es un marcado para* Γ_0,
- $\mathcal{R}$ *es un conjunto de reglas de reescritura tal que* $r \in \mathcal{R}$ *es de la forma* $r = (L, R, \tau, {}^\bullet\tau, \tau^\bullet)$ *donde*

  1. $L = (P_L, T_L, F_L)$ *y* $R = (P_R, T_R, F_R)$ *son redes de Petri K-ordenadas llamadas la* parte izquierda *y la* parte derecha *de r respectivamente.*
  2. $\tau \subseteq (P_L \times P_R) \cup (T_L \times T_R)$, *llamada la* relación de transferencia *de r, es una relación binaria que relaciona lugares de L con lugares de R y transiciones de L con transiciones de R:*

$$\begin{array}{ll} P_L\tau \subseteq P_R & \tau P_R \subseteq P_L \\ T_L\tau \subseteq T_R & \tau T_R \subseteq T_L \end{array}$$

  3. ${}^\bullet\tau \subseteq \tau$ *y* $\tau^\bullet \subseteq \tau$ *son subrelaciones de la relación de transferencia llamadas la* relación interfaz de entrada *y la* relación interfaz de salida, *respectivamente.*

Definición 172 (Estado de un SRR K-ordenadas) *Un* estado *es un par* (Γ, M) *en el que* Γ *es una red K-ordenada y* M *es un marcado para* Γ.

Definición 173 (Evento de un SRR K-ordenadas) *Un* evento *de un sistema de reescritura de redes K-ordenadas es una transición o una regla de reescritura:* $E = T \cup \mathcal{R}$.

Definición 174 (Grafo de estados de un SRR K-ordenadas) *El* grafo de estados *de un sistema de reescritura de redes K-ordenadas* $N = (\mathcal{R}, \Gamma_0, M_0)$ *es el grafo dirigido etiquetado cuyos nodos son los estados de* N, *es decir, redes K-ordenadas marcadas, y cuyos arcos (etiquetados con eventos de* N*) son:*

- *el* disparo de una transición*:* $(\Gamma, M) \xrightarrow{t} (\Gamma, M')$ *donde* $t \in T_\Gamma \iff M[t\rangle M'$ *en* Γ.
- *el* cambio de configuración*:* $(\Gamma, M) \xrightarrow{r} (\Gamma', M')$ *donde* $r = (L, R, \tau, {}^\bullet\tau, \tau^\bullet) \in \mathcal{R}$ *y* $\exists$ *un embedding* $f : L \to \Gamma$ *compatible con el orden y tal que para todo* $x \notin f(L)$, $y \in L$ *se tiene*

$$\begin{array}{lll} x \in {}^\bullet f(y) & \Rightarrow & y \in Dom({}^\bullet\tau) \text{ y} \\ x \in f(y)^\bullet & \Rightarrow & y \in Dom(\tau^\bullet) \end{array}$$

  y, si $\Gamma = (P, T, F)$ *y* $\Gamma' = (P', T', F')$, *se tiene:*

$$P' = P - f(P_L) + P_R \ \textit{tal que}\ P_L\tau = P_R$$

$$T' = T - f(T_L) + T_R \ \textit{tal que}\ T_L\tau = T_R$$

donde el significado de +(-) es añadir(eliminar) lugares/transiciones a(de) Γ. Los nombres de los lugares P_R(transiciones T_R) añadidos a Γ deben ser nombres nuevos para evitar conflictos.

$$M'(p) = \begin{cases} M(p) & si\ p \notin R \\ \sum_{p' \in\ ^{\tau}p} M(p') & si\ p \in R \end{cases}$$

$$F'(x,y) = \begin{cases} F(x,y) & si\ x \notin R\ \wedge\ y \notin R \\ F_R(x,y) & si\ x \in R\ \wedge\ y \in R \\ \sum_{y_i \in\ ^{\bullet\tau}y} F(x, f(y_i)) & si\ x \notin R\ \wedge\ y \in R \\ \sum_{x_i \in\ ^{\tau\bullet}x} F(f(x_i), y) & si\ x \in R\ \wedge\ y \notin R \end{cases}$$

Proposición 175 *Para todo sistema de reescritura de redes K-ordenadas N, existe un sistema de reescritura de redes N' que es una extensión del sistema de reescritura de redes subyacente N_0 a N tal que la restricción del grafo de estados de N' a N_0 coincide con el grafo de estados de N.*

Analizando esta proposición llegamos a las siguientes conclusiones:

- Que N_0 sea el sistema de reescritura de redes subyacente (sin órdenes) al sistema de reescritura de redes K-ordenadas N significa que $N_0 \sqsubseteq N$, es decir, N_0 es la restricción de N.
- Que N' sea una extensión del sistema N_0 significa que $N_0 \sqsubseteq N'$ y el grafo de estados de N_0 coincide con la restricción del grafo de estados de N' a N_0.

Por tanto, según la proposición precedente, podemos deducir que:

Corolario 176 *Si el grafo de estados de N_0 coincide con el grafo de estados de N, los lugares y las transiciones del conjunto de órdenes son* inobservables*, es decir, no tienen ningún efecto en el grafo de estados de N.*

En resumen, si K es un conjunto finito de órdenes, una *red de Petri K-ordenada* es una red de Petri (P, T, F) junto con una función $\kappa : P \cup T \to K$ que asocia todo elemento de la red con un orden en K. Un *sistema de reescritura de redes K-ordenadas* es, por tanto, un sistema de reescritura de redes en el que todos los componentes de las redes de Petri que lo constituyen (configuración inicial y partes izquierda y derecha de las reglas) están K-ordenados. El *grafo de estados de un sistema de reescritura de redes K-ordenadas* no es más que el grafo de estados del sistema de reescritura de redes subyacente con la excepción que para poder aplicar un cambio de configuración se requiere, además, que el *embedding* (de la parte izquierda de la regla en la configuración actual) sea compatible con los órdenes, es decir, que haga corresponder un elemento (lugar o transición) con un elemento con el mismo orden.

Para demostrar que la expresividad de las máquinas de Turing y los sistemas de reescritura de redes es la misma, vamos a partir, como ya hemos dicho, de una subclase de sistemas de reescritura de redes cuyas relaciones de transferencia van a ser biyecciones parciales. Puesto que hemos demostrado que el modelo de los sistemas de reescritura de redes no cambia al añadir restricciones de orden a sus lugares y transiciones, podemos asociar todo sistema de reescritura de redes K-ordenadas cuyas relaciones de transferencia sean biyecciones parciales con un sistema de reescritura de redes equivalente cuyas relaciones de transferencia sean también biyecciones parciales. Para este propósito, notamos que, ya que las relaciones de transferencia son biyecciones parciales, las relaciones de flujo de sus distintas configuraciones estarán acotadas superiormente por el máximo entre los valores de las relaciones de flujo de la configuración inicial y los valores de las relaciones de flujo de la parte derecha de las reglas. En la Figura 5.10 representamos que una transición t tiene un orden n y que un lugar p tiene un orden m mediante subredes concretas añadidas a las redes originales.

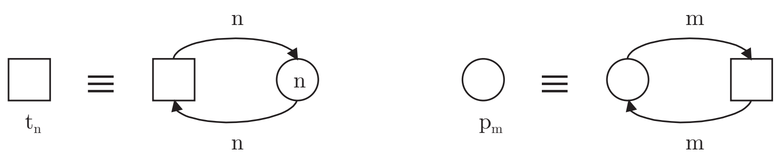


Figura 5.10: Representación del orden en una transición y un lugar

Estos enteros se elegirán lo suficientemente grandes para que no se confundan con los valores de las relaciones de flujo de alguna configuración alcanzable.

Es decir, vamos a exigir que todas las relaciones de flujo del sistema tengan un peso inferior al peso de las relaciones de flujo de los órdenes.

5.3.2. Simulación de una Máquina de Turing

Vamos a demostrar que podemos simular una máquina de Turing a partir de un sistema de reescritura de redes K-ordenadas cuya relación de transferencia τ es una biyección parcial. Así pues, no podremos obtener herramientas de verificación automáticas para esta clase de sistemas de reescritura y, por consiguiente, para la clase general de sistemas de reescritura de redes tampoco.

En primer lugar, recordamos la definición de máquina de Turing.

Definición 177 (Máquina de Turing (MT) [HU79][Sud88]) *Una* máquina de Turing *está constituida por los siguientes elementos:*

- *una cinta infinita a izquierda y a derecha dividida en un número infinito de celdas consecutivas, cada celda conteniendo un* 0 *o un* 1*,*
- *una cabeza de lectura/escritura que puede leer y escribir en la celda actual (aquélla sobre la que la cabeza se encuentra) y que puede también desplazarse de una celda hacia la izquierda o hacia la derecha,*
- *un conjunto finito de estados internos* Q*, y*
- *un conjunto finito de instrucciones que tienen la forma* $[q, v, w, q']$ *siendo* q *y* q' *estados internos,* $v \in \{0, 1\}$ *un posible valor de una celda y* $w \in \{0, 1, L, R\}$ *una operación. Esta instrucción se lee: si la máquina de Turing está en el estado* q *y la celda actual contiene el símbolo* v*, entonces la máquina realiza la operación* w *y pasa al estado* q'*. Según la operación* w*, la cabeza de lectura/escritura es capaz de:*
  - *escribir* 0 *(*1*) en la celda actual si* $w = 0$ *(*$w = 1$*) y*
  - *moverse de una celda hacia la izquierda o hacia la derecha si* $w = L$ *o* $w = R$*.*

Definición 178 (Configuración de una MT) *Una* configuración *es un par* $(q, u) \in Q \times \{0, 1\}^{\mathbb{Z}}$ *en el que* $q \in Q$ *es un estado y* $u : \mathbb{Z} \to \{0, 1\}$ *es una función que describe el estado actual de la cinta (*0 *es la posición de la cabeza). La* configuración inicial *es* (q_0, u_0) *dónde* $u_0(n) = 0$ *para todo* $n \in \mathbb{Z}$*, es decir, inicialmente todas las celdas tienen el valor* 0*.*

Definición 179 (Relaciones de Transición de una MT) *Las* relaciones de transición *entre configuraciones son las siguientes:*

$$(q,u) \xrightarrow{x} (q',u') \iff \left\{ \begin{array}{c} (x = [q,u(0),u'(0),q'] \wedge \forall i \neq 0 \ u'(i) = u(i)) \vee \\ (x = [q,u(0),R,q'] \wedge \forall i \in \mathbb{Z} \ u'(i) = u(i+1)) \vee \\ (x = [q,u(0),L,q'] \wedge \forall i \in \mathbb{Z} \ u'(i) = u(i-1)) \end{array} \right.$$

Una máquina de Turing es *determinista* si no tiene dos instrucciones diferentes $[q_1,\ v_1,\ w_1,\ q_1']$ y $[q_2,\ v_2,\ w_2,\ q_2']$ tales que $q_1 = q_2$ y $v_1 = v_2$. Esto quiere decir que en cada configuración de la máquina como mucho puede aplicarse una instrucción. El comportamiento de una máquina de Turing determinista puede representarse mediante la secuencia de las configuraciones obtenidas a partir de la configuración inicial. Así, una máquina de Turing puede *parar* si alcanza una configuración en la que no es posible aplicar instrucciones, o puede quedarse en un *ciclo*, si alcanza alguna configuración dos veces, o *no está acotada* si alcanza un número infinito de configuraciones diferentes. No es decidible si la máquina de Turing parará y tampoco si no está acotada. Puede demostrarse que una máquina de Turing está acotada si y sólo si la cabeza escanea solamente una parte finita de la cinta.

Vamos a mostrar que una máquina de Turing determinista puede simularse mediante un sistema de reescritura de redes K-ordenadas en el que τ es una biyección parcial, de forma que una máquina de Turing está acotada si y sólo si su SRR K-ordenadas asociado está acotado.

Proposición 180 *El acotamiento de un SRR K-ordenadas tal que τ es una biyección parcial no es decidible.*

Puesto que cualquier configuración alcanzable a partir de la configuración inicial tiene solamente un número finito de celdas con un contenido no nulo, en todo momento es suficiente representar únicamente una parte finita de la cinta. En la Figura 5.11 mostramos la equivalencia entre una configuración de una máquina de Turing y una configuración de un sistema de reescritura de redes K-ordenadas.

Una celda de la máquina es un lugar con el orden 0 o 1 representando su contenido. Cada estado de la máquina de Turing también se considera un orden y la celda actual se representa por el lugar que tiene una transición de entrada con el estado actual de la máquina como orden.

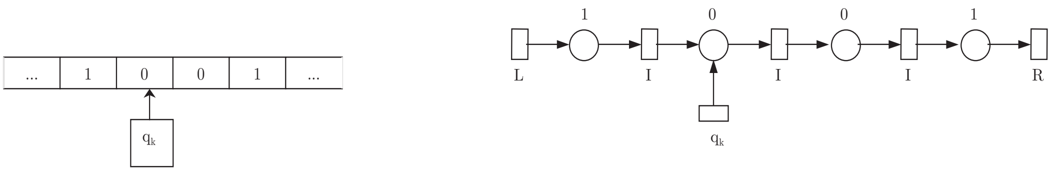


Figura 5.11: Implementación de una máquina de Turing con un sistema de reescritura de redes K-ordenadas

Representamos que una celda es la más a la izquierda, interna o la más a la derecha en función del orden de las transiciones de entrada y de salida del lugar que interviene.

Las relaciones de transición entre las configuraciones de la máquina son las reglas de reescritura del sistema de reescritura.

Proposición 181 *Un Sistema de Reescritura de Redes K-ordenadas cuya relación de transferencia τ es una biyección parcial tiene el mismo poder computacional que la Máquina de Turing.*

Demostración:

Formalmente, la máquina de Turing determinista se representa por el sistema de reescritura de redes K-ordenadas como sigue.

Sea $N = (\mathcal{R}, \Gamma_0, M_0)$ un sistema de reescritura de redes K-ordenadas en el que τ es una biyección parcial y donde

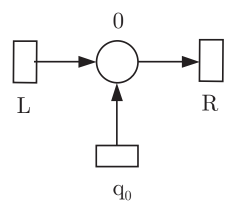
- $K = \{0, 1\} \cup Q \cup \{L, R, I\}$ es el conjunto de órdenes tal que
  - $\{0, 1\}$ son los órdenes de los lugares e indican el contenido de la celda que cada lugar representa, y
  - las transiciones pueden estar ordenados con $Q \cup \{L, R, I\}$ indicando:
    - L es el orden de la transición de entrada al lugar que representa la celda más a la izquierda de la cinta,
    - R es el orden de la transición de salida del lugar que representa la celda más a la derecha de la cinta,
    - I es el orden de las transiciones entre los lugares que representan celdas internas, y

- ◦ Q es un conjunto finito de órdenes que representan los estados internos de la máquina. El estado de la configuración actual q_k está representado por una transición de entrada al lugar de la celda actual con el orden $q_k \in Q$;

- $\Gamma_0 = (P_0, T_0, F_0)$ es la configuración inicial donde $P_0 = \{p_0\}$, $T_0 = \{q_0, L, R\}$ y F_0 se define como sigue:

$$\begin{array}{lll} F_0(p_0, q_0) = 0 & F_0(p_0, L) = 0 & F_0(p_0, R) = 1 \\ F_0(q_0, p_0) = 1 & F_0(L, p_0) = 1 & F_0(R, p_0) = 0 \end{array}$$

Gráficamente,

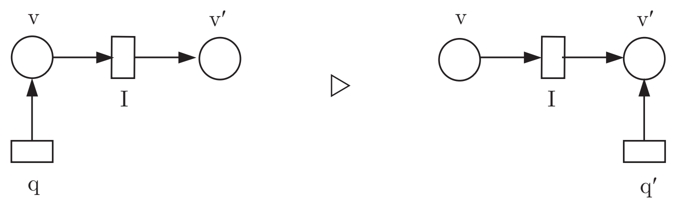


- M_0 es el marcado inicial tal que $M_0(p_0) = 0$,

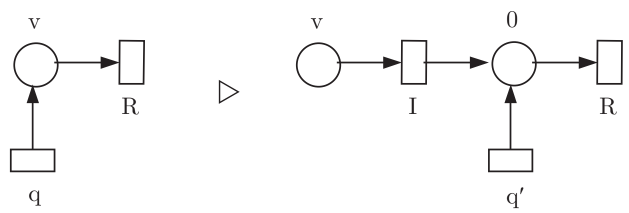
- $\mathcal{R}$ es el conjunto de reglas de reescritura y representa las relaciones de transición entre configuraciones de la máquina de Turing. Las reglas son las siguientes:

  - **Escritura de un valor en la celda actual**. Tenemos una regla de reescritura por cada instrucción de la máquina de Turing de la forma $[q, v, v', q'] \in Q \times \{0,1\}^2 \times Q$.

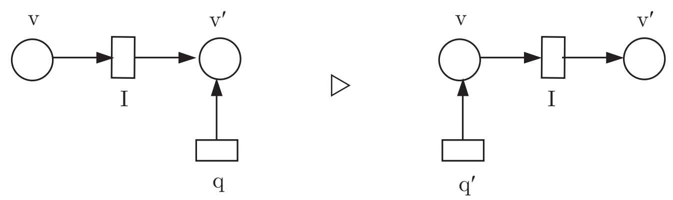
- **Movimiento de la cabeza a derecha**. Tenemos dos reglas de reescritura por cada instrucción de la máquina de Turing de la forma $[q, v, R, q']$ según si la celda actual es la más a la derecha o no.
  - Movimiento de la cabeza a derecha si la celda activa no es la más a la derecha:



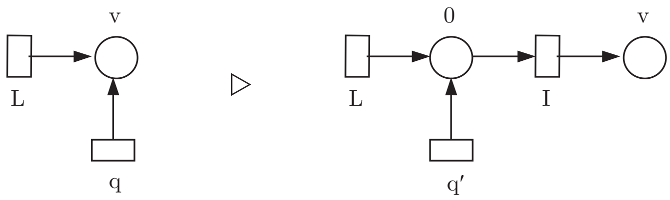
  - Movimiento de la cabeza a derecha si la celda activa es la más a la derecha:



- **Movimiento de la cabeza a izquierda**. Tenemos dos reglas de reescritura por cada instrucción de la máquina de Turing de la forma $[q, v, L, q']$ según si la celda actual es la más a la izquierda o no.
  - Movimiento de la cabeza a izquierda si la celda activa no es la más a la izquierda:



- ∘ Movimiento de la cabeza a izquierda si la celda activa es la más a la izquierda:



■

Ya hemos visto cómo simular una máquina de Turing determinista a partir de un sistema de reescritura de redes K-ordenadas cuya relación de transferencia τ es una biyección parcial. Esta simulación puede extenderse a la clase general de sistemas de reescritura de redes y, por tanto, no podremos obtener herramientas de verificación automáticas para la clase general de sistemas de reescritura de redes.

Corolario 182 *Los Sistemas de Reescritura de Redes tienen el mismo poder computacional que las Máquinas de Turing.*

En la siguiente sección vamos a mostrar que las redes reconfigurables son equivalentes a las redes de Petri.

5.4. Implementación de Redes Reconfigurables con Redes de Petri

El propósito de esta sección es demostrar que las redes reconfigurables son equivalentes a las redes de Petri. A simple vista, puede parecer que no son equivalentes ya que el comportamiento de una red reconfigurable y una red de Petri es distinto debido al conjunto de reglas de reescritura de la primera. Cuando en una red reconfigurable se dispara una regla de reescritura, tiene lugar un cambio de configuración (por $\Gamma_i[r\rangle\Gamma_j$ denotamos el cambio de configuración debido a una regla de reescritura r desde la configuración Γ_i a la configuración Γ_j), es decir, un cambio en la estructura de la red. Obtener una red de Petri equivalente significa que esos cambios de configuración estén presentes, es decir, que todas las configuraciones posibles estén representadas, de alguna forma, en la red. Se deduce pues, que el número de configuraciones debe ser finito para que sea posible su representación. Esta afirmación es obvia puesto que el número de reglas de reescritura es finito.

Recordemos que llamamos *configuración* de una red reconfigurable a una red de Petri $\Gamma = (P, T, pre, post)$ tal que pre y $post$ son un par de aplicaciones $pre,\ post : P \times T \to \mathbb{N}$. Si una configuración es *alcanzable* entonces se verifica:

$\forall p \in P, \forall t \in T : pre(p,t) = n \Longrightarrow \exists (p',t') \in P \times T$ tal que

$$\begin{cases} pre_0(p',t') = n \text{ ó} \\ \exists r \in \mathcal{R},\ n = r^{\bullet}(p',t') \end{cases}$$

$\forall p \in P, \forall t \in T : post(p,t) = n \Longrightarrow \exists (p',t') \in P \times T$ tal que

$$\begin{cases} post_0(p',t') = n \text{ ó} \\ \exists r \in \mathcal{R},\ n = r^{\bullet}(t',p') \end{cases}$$

Es decir, en cualquier configuración alcanzable no tenemos más que arcos de la configuración inicial o arcos resultado de la aplicación de reglas de reescritura. Vamos a denominar *configuración local* a una función $T \to \mathbb{N} \times \mathbb{N}$ que representa las relaciones de flujo entrantes y salientes con respecto a un conjunto fijo de transiciones T de algún lugar de una configuración de la red reconfigurable. Una configuración de una red reconfigurable equivale pues a asociar cada lugar (del conjunto fijo de lugares) con la correspondiente configuración local. Así pues, una configuración local de una red reconfigurable podrá ser bien una configuración local de la configuración inicial o bien una configuración local de la parte derecha de una regla. Por tanto, sólo existen un número finito de configuraciones locales y (puesto que el conjunto de lugares es, a su vez, fijo), sólo un número finito de configuraciones. Por $Conf(N) = \{\Gamma_0, \Gamma_1, \ldots, \Gamma_k\}$ denotamos el conjunto de configuraciones de una red reconfigurable N, donde Γ_0 es la *configuración inicial* o configuración del estado inicial $\gamma_0 = (\Gamma_0, M_0)$.

Vamos a ver el procedimiento a seguir para obtener una red de Petri a partir de una red reconfigurable. Para tener una visión espacial del modelo, imaginamos los lugares y las transiciones de una configuración $\Gamma = (P, T, F)$ como si estuvieran situados en dos planos paralelos diferentes, es decir, un plano con el conjunto de lugares y un plano con el conjunto de transiciones que estarán conectados por las relaciones de flujo F de la configuración Γ representada.

La red de Petri equivalente a una red reconfigurable $N = (P, T, \mathcal{R}, \gamma_0)$ se construye mediante el Algoritmo 6.

Algoritmo 6 Red de Petri equivalente a una Red Reconfigurable

- Partimos de la configuración inicial. Para dicha configuración Γ_0, obtenemos todas las posibles configuraciones debidas a los disparos de todas las reglas de reescritura autorizadas en Γ_0, es decir, obtenemos los nodos del primer nivel del grafo de configuración $G(N)$. Tendremos un único plano de lugares y tantos planos de transiciones como configuraciones diferentes se den. Cada plano de transiciones estará ligado al plano de lugares por las relaciones de flujo de la configuración representada.
- Añadimos un lugar q_i por cada plano de transiciones.
- Añadimos también una transición r_{0i} entre el lugar q_0 de la configuración inicial y cada lugar q_i de cada configuración accesible desde q_0, representando el cambio de configuración debido a la regla r, $\Gamma_0[r\rangle\Gamma_i$.
- Repetimos todo este procedimiento para cada configuración distinta de la configuración inicial, es decir, para cada plano de transiciones obtenido a partir del inicial.

Este procedimiento es finito porque tenemos un número finito de lugares, un número finito de transiciones y un número también finito de reglas de reescritura. Así, llegaremos a configuraciones que ya están representadas en el modelo y acabaremos.

Proposición 183 *Toda red reconfigurable* $N = (P, T, \mathcal{R}, \gamma_0)$ *es equivalente a una red de Petri* $\widetilde{N} = (\widetilde{P}, \widetilde{T}, \widetilde{F}, \widetilde{M}_0)$.

Demostración:

La traducción del modelo de las redes reconfigurables al modelo de redes de Petri es automática. La red de Petri asociada a N es $\widetilde{N} = (\widetilde{P}, \widetilde{T}, \widetilde{F}, \widetilde{M}_0)$ y está compuesta por

- Un conjunto de lugares $\widetilde{P} = P \cup \{q_0, \ldots, q_k\}$ donde cada q_i es un nuevo lugar asociado a la configuración Γ_i.
- Un conjunto de transiciones $\widetilde{T} = (\{q_0, \ldots, q_k\} \times T) \cup \widetilde{R}$ siendo $\widetilde{R}$ el conjunto de transiciones tales que $r_{ij} \in \widetilde{R}$ si $\exists r \in R$ tal que $\Gamma_i[r\rangle\Gamma_j$ en $G(N)$.

- Un conjunto de relaciones de flujo $\widetilde{F}$ tal que

$$\widetilde{F}(p,(q_i,t)) = F_i(p,t) \text{ siendo } \Gamma_i = (P,T,F_i)$$
$$\widetilde{F}((q_i,t),p) = F_i(t,p) \text{ siendo } \Gamma_i = (P,T,F_i)$$

$$\widetilde{F}(p,r) = \widetilde{F}(r,p) = 0 \ \forall r \in \widetilde{R}$$

$$\widetilde{F}(q_i,r_{ij}) = \begin{cases} 1 & \text{si } \exists j : \Gamma_i[r\rangle\Gamma_j \\ 0 & \text{sino} \end{cases} \quad \widetilde{F}(r_{ij},q_j) = \begin{cases} 1 & \text{si } \exists i : \Gamma_i[r\rangle\Gamma_j \\ 0 & \text{sino} \end{cases}$$

$$\widetilde{F}(q_i,(q_j,t)) = \begin{cases} 1 & \text{si } i=j \\ 0 & \text{sino} \end{cases} \quad \widetilde{F}((q_j,t),q_i) = \begin{cases} 1 & \text{si } i=j \\ 0 & \text{sino} \end{cases}$$

- Un marcado inicial $\widetilde{M}_0$ tal que

$$\widetilde{M}_0(p) = M_0(p) \quad \text{si } p \in P$$

$$\widetilde{M}_0(q_i) = \begin{cases} 1 & \text{si } i=0 \\ 0 & \text{sino} \end{cases}$$

■

Observación 184 *(Correspondencia entre Red Reconfigurable y Red de Petri)*

Si $\widetilde{M} : \widetilde{P} \to \mathbb{N}$ *es un marcado alcanzable de* $\widetilde{N}$ *entonces* $\sum_{i=0}^{k} \widetilde{M}(q_i) = 1$.

Denotamos $\gamma_{\widetilde{M}} = (\Gamma_{\widetilde{M}}, M)$ *donde* $M : P \to \mathbb{N}$ *es un marcado de* $\Gamma_{\widetilde{M}}$ *y* $\Gamma_{\widetilde{M}} = \Gamma_i$ *es la configuración asociada a* $\widetilde{M}$ *tal que* $\widetilde{M}(q_i) = 1$.

Inversamente, si $\gamma = (\Gamma, M)$ *es un estado alcanzable de* N *le asociamos la aplicación* $\widetilde{M}_\gamma : \widetilde{P} \to \mathbb{N}$ *con*

$$\widetilde{M}_\gamma(p) = M(p) \quad \textit{si } p \in P$$

$$\widetilde{M}_\gamma(q_i) = \begin{cases} 1 & \textit{si } \Gamma = \Gamma_i \\ 0 & \textit{sino} \end{cases}$$

Observación 185 $\widetilde{M}_{(\gamma_{\widetilde{M}})} = \widetilde{M}$ *y* $\gamma_{(\widetilde{M}_\gamma)} = \gamma$.

Observación 186 $\gamma_{(\widetilde{M}_0)} = (\Gamma_0, M_0)$ *y* $\widetilde{M}_{\gamma_0} = \widetilde{M}_0$.

Proposición 187 *Si* $\gamma = (\Gamma, M)$ *es un estado alcanzable de* N *entonces*

1. $\gamma[r\rangle\gamma' \Longleftrightarrow \widetilde{M}_\gamma[r\rangle\widetilde{M}_{\gamma'}$ *siendo* $\gamma' = (\Gamma', M)$.
2. $\gamma[t\rangle\gamma' \Longleftrightarrow \widetilde{M}_\gamma[(q_i, t)\rangle\widetilde{M}_{\gamma'}$ *siendo* $\gamma = (\Gamma_i, M)$ *y* $\gamma' = (\Gamma_i, M')$.

Observación 188 (q_i, t) *no es disparable más que en marcados* $\widetilde{M}$ *tales que* $\gamma_{\widetilde{M}} = (\Gamma_i, M)$.

La proposición y la observación precedentes indican que:

1. Si cambiamos del estado $\gamma = (\Gamma_i, M)$ al estado $\gamma' = (\Gamma_j, M)$ en la red reconfigurable debido al disparo de una regla de reescritura, lo que cambia en la red de Petri equivalente es el marcado de los lugares q_i y q_j:

$$\begin{array}{lcl} \widetilde{M}_\gamma(q_i) = 1 & \Longrightarrow & \widetilde{M}_{\gamma'}(q_i) = 0 \\ \widetilde{M}_\gamma(q_j) = 0 & \Longrightarrow & \widetilde{M}_{\gamma'}(q_j) = 1 \end{array}$$

   Y también en el otro sentido.

2. Si cambiamos del estado $\gamma = (\Gamma_i, M)$ al estado $\gamma' = (\Gamma_i, M')$ en la red reconfigurable debido al disparo de una transición t, lo que cambia en la red de Petri equivalente es el marcado de los lugares $p \in P$ afectados por el disparo de la transición (q_i, t). Y viceversa.

Corolario 189 *Los marcados* $\widetilde{M}_\gamma$ *en los que* γ *recorre los estados alcanzables de* N *son los marcados alcanzables de* $\widetilde{N}$ *y el grafo de marcado de* $\widetilde{N}$ *es isomorfo al de* N.

Así pues, las redes reconfigurables son equivalentes a las redes de Petri pero proporcionan representaciones más compactas de sistemas concurrentes cuya estructura evoluciona en tiempo de ejecución. Esto se puede apreciar en la Figura 5.12 que ilustra un fragmento de la red de Petri equivalente a la red reconfigurable del Ejemplo 166. Para facilitar la comprensión de la implementación de una red reconfigurable con una red de Petri, sólo mostramos cómo representar un cambio de configuración (el cambio de modo secuencial a paralelo) y solamente para los lugares y las transiciones implicados. Con este pequeño fragmento podemos hacernos una idea del tamaño de la red de Petri completa y de la manera fácil y concisa en la que podemos modelizar el sistema con una red reconfigurable.

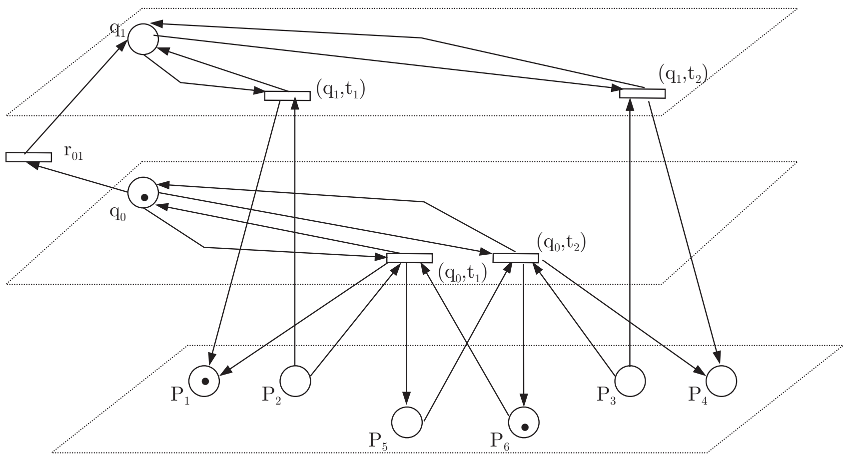


Figura 5.12: Parte de la red de Petri equivalente a la red reconfigurable del Ejemplo 166

En general, si la red reconfigurable $N = (P, T, \mathcal{R}, \gamma_0)$ de la que partimos tiene n lugares, m transiciones, r reglas de reescritura y el número de configuraciones es $k+1$, la red de Petri equivalente $\widetilde{N} = (\ \widetilde{P},\ \widetilde{T},\ \widetilde{F},\ \widetilde{M_0})$ que obtendremos consta de:

- $n + (k+1)$ lugares y
- $((k+1)*m)+z$ transiciones, siendo z es el número de transiciones de $\widetilde{R}$, es decir, $z = \sum_{i=0}^{k}$número de disparos de reglas de $\mathcal{R}$ en la configuración Γ_i. Para cada configuración Γ_i dicho número de disparos se obtiene a

partir del grafo de configuración de N, $G(N)$, como la suma de los arcos de salida y de entrada al nodo que representa tal configuración, es decir, $|\Gamma_i^\bullet| + |{}^\bullet\Gamma_i|$. Así pues, $z = \sum_{i=0}^{k}(|\Gamma_i^\bullet| + |{}^\bullet\Gamma_i|)$.

Podemos concluir, por tanto, que las redes reconfigurables son un modelo abreviado de las redes de Petri; es decir, la expresividad es la misma pero con las redes reconfigurables podemos modelizar más fácil y directamente sistemas que cambian su estructura dinámicamente. Así pues, todas las propiedades que sean decidibles para las redes de Petri (ver Tabla 1.3) también lo serán para las redes reconfigurables.

Proposición 190 *La terminación, la cobertura, el acotamiento, el acotamiento de un lugar, el acotamiento estructural, el interbloqueo, la alcanzabilidad, la vivacidad y la t-vivacidad son propiedades decidibles para las redes reconfigurables.*

5.5. Ejemplo

En la Subsección 4.3.1 presentábamos un ejemplo de modelización de un sistema distribuido en el que pueden ocurrir cambios dinámicos estructurales, utilizando redes reconfigurables reversibles. Se trata de una red de transmisión de paquetes en la que desde dos máquinas se reciben bloques de información y se gestionan por un servidor principal. Vamos a modelizar el mismo sistema pero utilizando la propuesta final de las redes reconfigurables.

La Figura 5.13 representa el estado inicial $\gamma_0 = (\Gamma_0, M_0)$ de la red en el que las dos máquinas desde las que se reciben paquetes están representadas por las transiciones t_1 y t_8 y el servidor principal por el lugar p_7. El marcado inicial de la red es $M_0 = (1, 2, 0, 1, 0, 0, 1)$.

En la red diferenciamos dos partes delimitadas por los lugares p_1, p_3 y p_7. En cada parte, pueden ocurrir varios cambios de configuración, representados mediante reglas de reescritura:

- En la primera parte, se puede decidir enviar los paquetes de uno en uno, de dos en dos, de tres en tres o de cuatro en cuatro. Es decir, el arco que va desde el lugar p_2 a la transición t_3 puede tener pesos de 1 a 4.

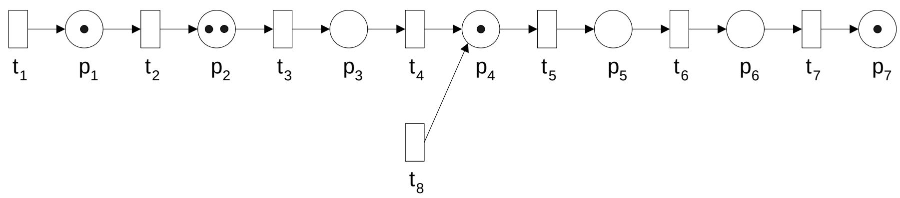


Figura 5.13: Estado inicial de la red de transmisión

- En la segunda parte, los datos pueden seguir distintos caminos en la red hasta llegar a p_7:
  - paquetes que siguen el camino normal $t_4p_4t_5p_5t_6p_6t_7p_7$;
  - paquetes que evitan la transición t_5 y siguen el camino $t_4p_4t_6p_6t_7p_7$; y finalmente,
  - paquetes que van directamente a través del camino $t_4p_4t_7p_7$, evitando las transiciones t_5 y t_6.

En la definición de las reglas de reescritura, cada lugar y cada transición tiene asociado un rol distinto. Distinguimos los roles P, Q, S, T, U, V y W entre lugares y los roles A, B, C, D, E, F, G y H entre transiciones.

$$\begin{array}{llll} P=\{p_1\} & Q=\{p_2\} & S=\{p_3\} & T=\{p_4\} \\ U=\{p_5\} & V=\{p_6\} & W=\{p_7\} & \end{array}$$

$$\begin{array}{llll} A=\{t_1\} & B=\{t_2\} & C=\{t_3\} & D=\{t_4\} \\ E=\{t_5\} & F=\{t_6\} & G=\{t_7\} & H=\{t_8\} \end{array}$$

Definimos ocho reglas de reescritura utilizando la notación de sumas formales (Notación 160) de la Sección 5.2. Su representación gráfica se muestra en la Figura 5.14.

1. Cambio en el envío de paquetes de 1 en 1 a de 2 en 2:

$$R_1 : Q(B-C) \triangleright Q(B-2C)$$

2. Cambio en el envío de paquetes de 1 en 1 a de 3 en 3:

$$R_2 : Q(B-C) \triangleright Q(B-3C)$$

3. Cambio en el envío de paquetes de 1 en 1 a de 4 en 4:

$$R_3 : Q(B - C) \rhd Q(B - 4C)$$

4. Cambio en el envío de paquetes de 2 en 2 a de 3 en 3:

$$R_4 : Q(B - 2C) \rhd Q(B - 3C)$$

5. Cambio en el envío de paquetes de 2 en 2 a de 4 en 4:

$$R_5 : Q(B - 2C) \rhd Q(B - 4C)$$

6. Cambio en el envío de paquetes de 3 en 3 a de 4 en 4:

$$R_6 : Q(B - 3C) \rhd Q(B - 4C)$$

7. Cambio de camino evitando la transición t_5:

$$R_7 : T(D + H - E) + U(E - F) \rhd T(D + H - F) + U(\emptyset)$$

8. Cambio de camino evitando las transiciones t_5 y t_6:

$$R_8 : T(D+H-E)+U(E-F)+V(F-G) \rhd T(D+H-G)+U(\emptyset)+V(\emptyset)$$

Así pues, la red reconfigurable que describe al sistema consta de 7 lugares, 8 transiciones y 8 reglas de reescritura. Puesto que la reversibilidad está implícita en las reglas de reescritura, también hubiera sido posible definir únicamente 5 reglas: R_1, R_2, R_3, R_7 y R_8; de manera que si, por ejemplo, se están enviando los paquetes de 2 en 2 y el operador decide que se envíen de 4 en 4, se tendría que aplicar la regla R_1, en sentido inverso, y seguidamente la regla R_3. Tal como hemos definido la red reconfigurable (con 8 reglas), el cambio de envío de paquetes de 2 en 2 a de 4 en 4 se hará aplicando directamente la regla R_5.

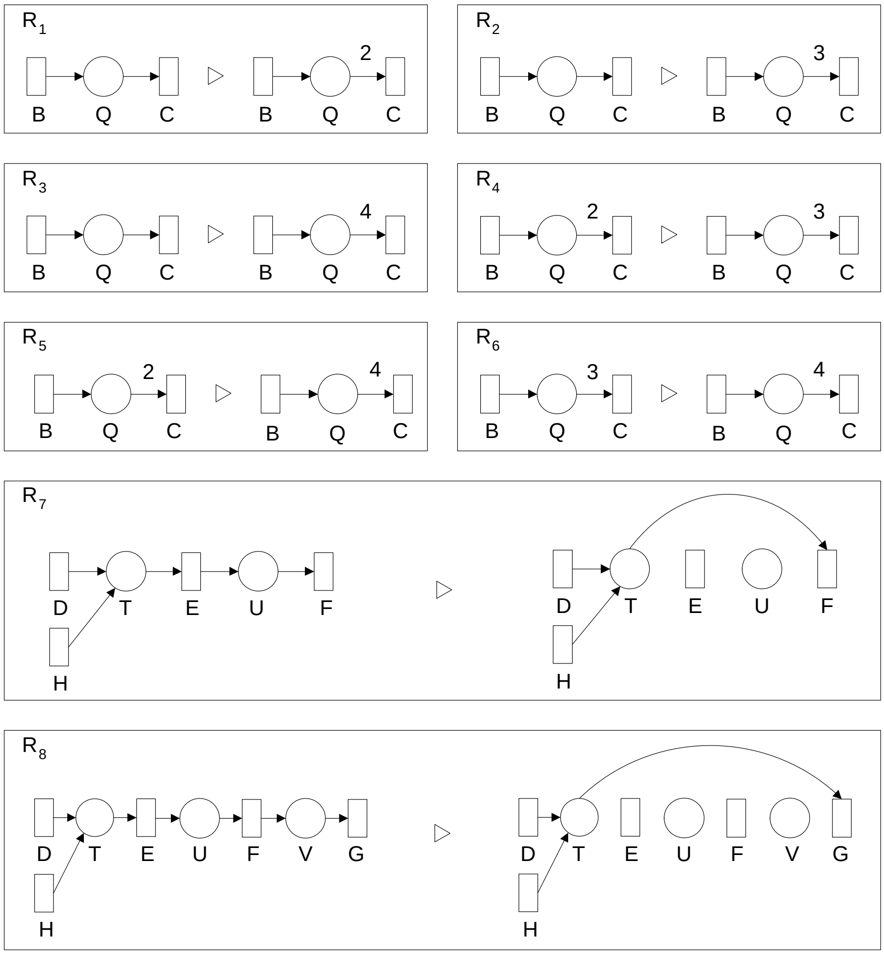


Figura 5.14: Las 8 reglas de reescritura de la red reconfigurable de la Figura 5.13

Las 12 posibles configuraciones de esta red son las que se muestran en la Figura 5.15.

En la Figura 5.16 se presenta el grafo de configuración (ver Definición 163) de la red reconfigurable de la Figura 5.13. A partir de este grafo pueden deducirse las reglas que pueden dispararse desde cualquier configuración, es decir, los cambios de configuración inmediatos desde una configuración dada y también los cambios de configuración sucesivos que deben ocurrir para cambiar a otra configuración no inmediata (los posibles caminos desde un nodo a otro del grafo).

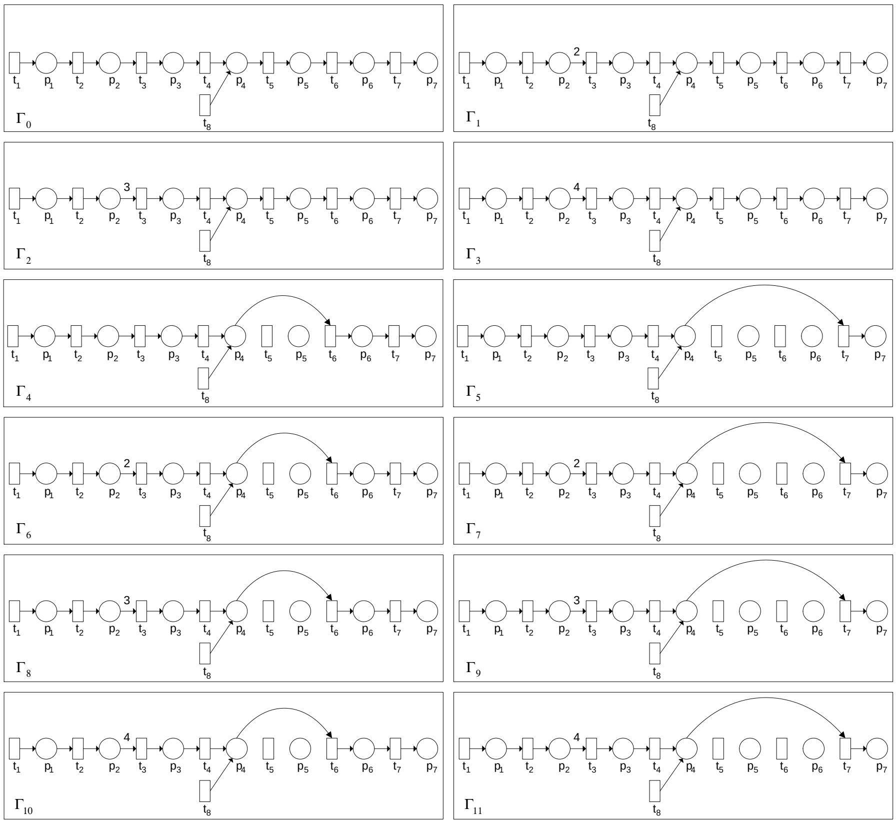


Figura 5.15: Las 12 configuraciones de la red reconfigurable de la Figura 5.13

En el grafo de configuración existe un arco etiquetado con la regla R_8 desde el nodo etiquetado con Γ_0 al nodo etiquetado con Γ_5, lo que significa que puede haber un cambio de configuración $\Gamma_0[R_8\rangle\Gamma_5$ debido a la regla R_8 desde la configuración inicial Γ_0 a la configuración Γ_5. En la Figura 5.17 se muestra el estado que se alcanza cuando se aplica la regla de reescritura R_8 al estado inicial representado en la Figura 5.13. En este nuevo estado, como se puede observar, los paquetes se envían de uno en uno y siguen el camino $t_4p_4t_7p_7$, evitando las transiciones t_5 y t_6. En el grafo de estados, este cambio se representará con un arco etiquetado con R_8 desde el nodo asociado al estado (Γ_0, M_0) al nodo asociado al estado (Γ_5, M_0), es decir, $(\Gamma_0, M_0) \xrightarrow{R_8} (\Gamma_5, M_0)$.

La Figura 5.18 ilustra un fragmento de la red de Petri equivalente a la red reconfigurable del Ejemplo. Sólo se muestra cómo representar un cambio de configuración y sólo para los lugares y las transiciones implicadas. Es el cambio de configuración $\Gamma_0[R_8\rangle\Gamma_5$ debido a la regla R_8, comentado anterior-

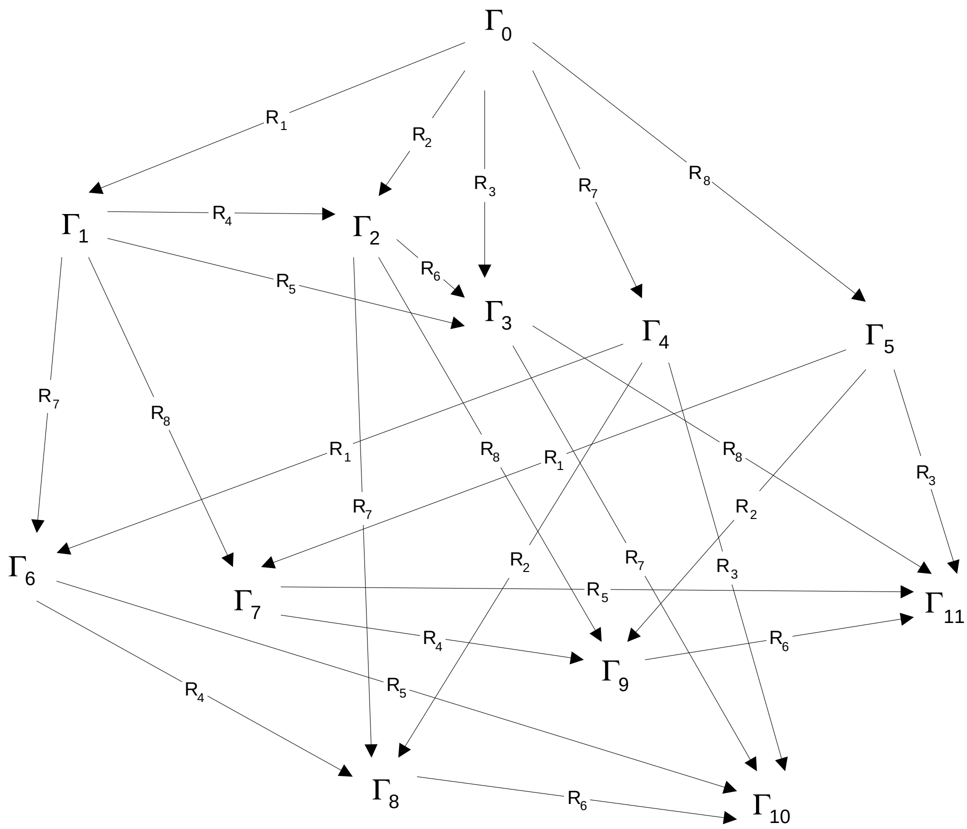


Figura 5.16: Grafo de configuración de la red reconfigurable de la Figura 5.13

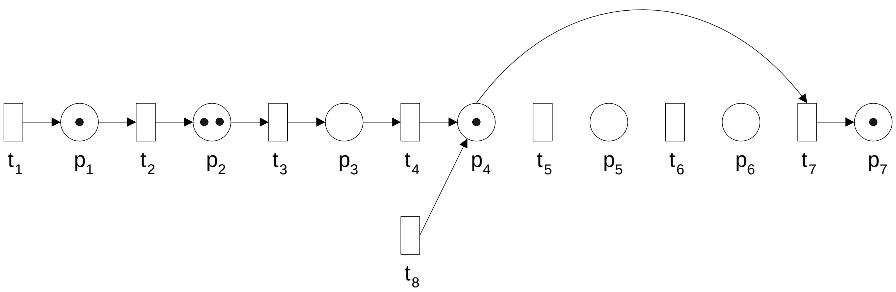


Figura 5.17: Nuevo estado alcanzado tras la aplicación de la regla R_8 al estado de la Figura 5.13

mente. Para hacernos una idea del incremento en tamaño que supone la red de Petri equivalente, calculamos el número de lugares y transiciones de dicha red en función del número de componentes de la red reconfigurable original (lugares, transiciones, reglas de reescritura y configuraciones), según lo visto en la Sección 5.4.

- La red reconfigurable de la que partimos consta de: $n = 7$ lugares, $m = 8$ transiciones, $r = 8$ reglas de reescritura y $k + 1 = 12$ configuraciones.
- La red de Petri equivalente que obtenemos tiene $n+(k+1) = 7+12 = 19$ lugares. El número de transiciones de dicha red se calcula aplicando la fórmula $((k+1)*m)+z = (12*8)+z = 96+z$ donde $z = \sum_{i=0}^{k}(|\Gamma_i^\bullet|+|^\bullet\Gamma_i|)$. El valor de z se obtiene directamente a partir del grafo de configuración de la Figura 5.16, $z = (5+0)+(4+1)+(3+2)+(2+3)+(3+1)+(3+1)+(2+2)+(2+2)+(1+3)+(1+3)+(4+0)+(4+0) = 52$. Así pues, la red de Petri equivalente consta de $96 + 52 = 148$ transiciones.

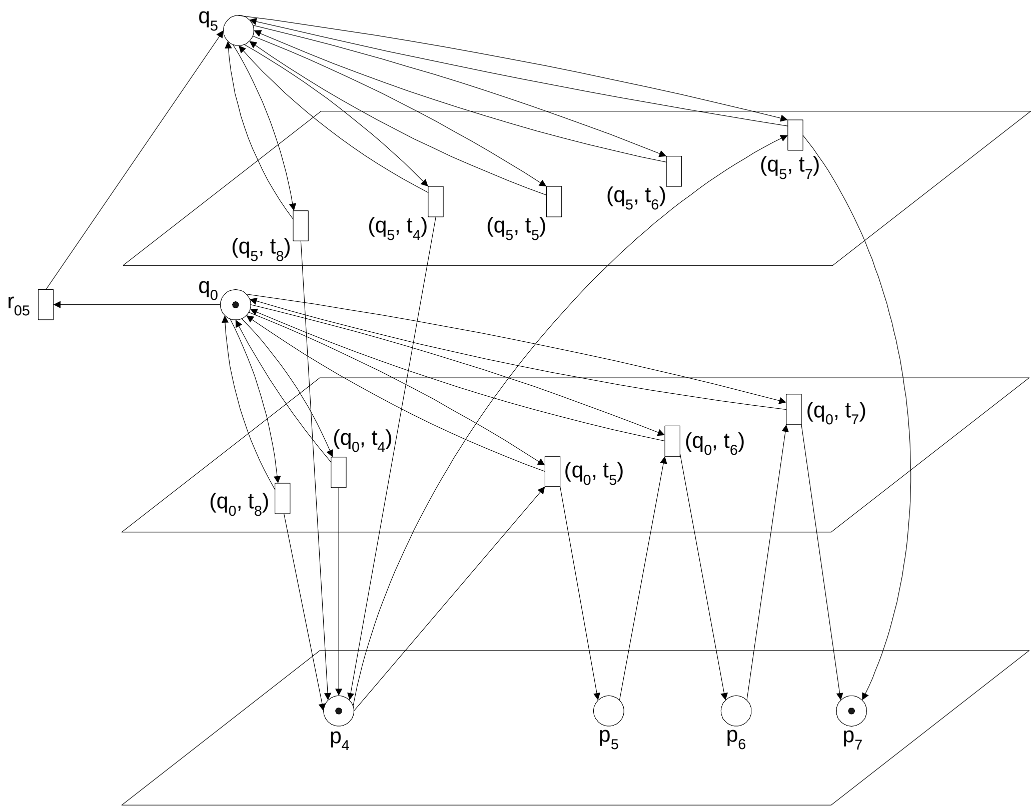


Figura 5.18: Red de Petri equivalente a la Figura 5.13

Capítulo 6

Verificación

Uno de los principales objetivos de la modelización de los sistemas utilizando redes de Petri es su análisis automático. En general, verificar la corrección de un sistema modelado consiste en determinar que ciertas propiedades se satisfacen. Recordemos del Capítulo 1 el significado de algunas propiedades:

- Una red de Petri está *acotada* cuando cada uno de sus lugares está acotado, es decir, cuando el número de tokens que contiene es menor que cierta cota para cualquier marcado alcanzable (cuando dicha cota es 1, el sistema se dice *seguro*).
- Una red de Petri está *viva* cuando cada una de sus transiciones está viva, es decir, cuando desde cualquier marcado alcanzable puede dispararse cualquier transición.
- Una red de Petri está *libre de interbloqueos* cuando en cualquier marcado alcanzable hay alguna transición habilitada.
- Un marcado es un *estado home* cuando es alcanzable desde cualquier estado alcanzable y una red es *reversible* cuando el marcado inicial es un estado *home*.
- Dos lugares están en *exclusión mutua* cuando nunca están marcados simultáneamente.

La acotabilidad excluye desbordamientos, la vivacidad asegura que ninguna acción en el sistema pueda resultar inalcanzable, la existencia de estados *home* informa de la posibilidad de volver a ciertos estados y la exclusión mutua se requiere entre lugares que representan el uso de un recurso común o la presencia en una sección crítica.

Los métodos de análisis de las redes de Petri [Mur89][STC98] se clasifican como sigue (ver Sección 1.3):

- Técnicas de enumeración: Si el sistema está acotado, el grafo de alcanzabilidad se utiliza como modelo computacional para un sistema de pruebas o para procedimientos de decisión y herramientas automáticas de verificación. Los dos grandes problemas de esta aproximación son el tamaño del espacio de estados de un sistema concurrente y la necesidad de repetir el análisis para cada marcado inicial de interés. Los sistemas no acotados pueden analizarse parcialmente utilizando una aproximación similar.

- Técnicas estructurales: La idea básica es obtener información útil sobre el comportamiento razonando sobre la estructura de la red y el marcado inicial. Dos ventajas cruciales de esta aproximación son el conocimiento profundo del comportamiento del sistema y la eficiencia de los algoritmos. Hay dos familias de técnicas íntimamente relacionadas que se han utilizado extensamente: la teoría de grafos y el álgebra lineal.

- Técnicas de transformación: Para facilitar el análisis de sistemas grandes y complejos, se pueden transformar (normalmente, reducir) de forma que preserven las propiedades a analizar. Las reglas de transformación, de alguna forma, preservan el comportamiento y, a menudo, son soportadas por argumentos estructurales como condiciones suficientes, eficientes y sencillas.

Todas estas técnicas no se entienden mutuamente exclusivas sino que deben combinarse para un análisis práctico.

En el Capítulo 5 hemos mostrado la equivalencia entre las redes reconfigurables y las redes de Petri, lo que asegura que todas las propiedades fundamentales de las redes de Petri siguen siendo decidibles para las redes reconfigurables. Para este modelo podemos encontrar, por tanto, herramientas de verificación automáticas. Hemos visto que la traducción a redes de Petri equivalentes incrementa considerablemente el tamaño de la red y, por ello, resultará más eficiente implementar los métodos de verificación de propiedades de las redes de Petri directamente en el modelo de las redes reconfigurables.

En la Sección 6.1 se analizan las propiedades de una red reconfigurable mediante una técnica de enumeración, la construcción del árbol de cobertura. En la Sección 6.2, el análisis se realiza utilizando la ecuación de estados, una técnica de tipo estructural.

6.1. Árbol de Cobertura de una Red Reconfigurable

En el árbol de alcanzabilidad de una red reconfigurable deben representarse tanto los cambios de estado debidos al disparo de una transición como los cambios de estado debidos al disparo de una regla de reescritura. Como se ha visto en el Capítulo 5, si se dispara una transición, sólo se ve afectado el marcado de la red (exactamente igual que en las redes de Petri) mientras que si lo que se aplica es una regla de reescritura, el marcado se mantiene pero la estructura de la red cambia, es decir, tiene lugar un cambio de configuración. Recordemos las definiciones de regla de reescritura y cambio de configuración de una red reconfigurable.

Una *regla de reescritura* $r \in \mathcal{R}$ es una estructura $r = (D, {}^\bullet r, r^\bullet)$ donde $D \subseteq P$ es el *dominio* de r, ${}^\bullet r : (D \times T) \cup (T \times D) \to \mathbb{N}$ y $r^\bullet : (D \times T) \cup (T \times D) \to \mathbb{N}$ son la *precondición* y *postcondición* de r, respectivamente. Es decir, son las relaciones de flujo de los lugares del dominio antes y después del cambio de configuración debido a la regla r.

Denotamos un *cambio de configuración* $\Gamma[r\rangle\Gamma'$, si y solamente si se cumple lo siguiente:

$$\forall p \in D \quad \begin{cases} F(p,t) = {}^\bullet r(p,t) \text{ y } F(t,p) = {}^\bullet r(t,p) \\ F'(p,t) = r^\bullet(p,t) \text{ y } F'(t,p) = r^\bullet(t,p) \end{cases}$$

$$\forall p \notin D \qquad F(p,t) = F'(p,t) \text{ y } F(t,p) = F'(t,p)$$

Es decir, un cambio de configuración sólo afecta a las relaciones de flujo de los lugares en el dominio de la regla aplicada, no afecta al marcado de los mismos.

Definimos una *relación de orden* entre estados de una red reconfigurable como sigue:

$$(\Gamma, M) \sqsubseteq (\Gamma', M') \iff (\Gamma = \Gamma') \text{ y } (\forall p \in P \ M(p) \leq M'(p))$$

Algoritmo 7 Árbol de Cobertura de una Red Reconfigurable

- Inicialmente, el árbol se reduce a su raíz etiquetada (Γ_0, M_0) y señalada como un vértice "nuevo".
- Mientras existan vértices "nuevos" hacer:
  - Seleccionar un nuevo vértice V, sea (Γ, M) su etiqueta.
  - Si la etiqueta es distinta de todas las etiquetas de los nodos en el camino desde la raíz hasta el vértice V entonces para todo disparo $(\Gamma, M) \xrightarrow{e} (\Gamma', M')$ hacer:
    - Crear un nuevo vértice V' y un arco desde V a V' etiquetado con e y señalar este vértice V' como "nuevo".
    - Etiquetar el vértice V' con la configuración $(\Gamma', \widetilde{M'})$ definida como sigue.

      Si existe algún nodo V'' en el camino desde la raíz al vértice V cuya etiqueta (Γ'', M'') es tal que $(\Gamma'', M'') \sqsubseteq (\Gamma', M')$ y $M''(p) < M'(p)$,
      entonces $\widetilde{M'}(p) = \omega$.
      En otro caso, $\widetilde{M'}(p) = M'(p)$.
  - Retirar V del conjunto de vértices "nuevos".

Para construir una aproximación finita del árbol de alcanzabilidad para redes reconfigurables, el árbol de cobertura, definimos los marcados generalizados. Un *marcado generalizado* $\widetilde{M}$ es una función $\widetilde{M} : P \rightarrow \mathbb{N} \cup \{\omega\}$ donde ω es un nuevo valor que representa un entero arbitrariamente grande añadido al conjunto de $\mathbb{N}$ tal que $\forall n \in \mathbb{N}$, $n < \omega$ $\omega + n = n + \omega = \omega + \omega = \omega$, $\omega - n = \omega$ y $n - \omega$ está indefinido.

La anterior relación de orden entre estados de una red reconfigurable puede extenderse a estados generalizados como sigue:

$$(\Gamma, \widetilde{M}) \sqsubseteq (\Gamma', \widetilde{M'}) \Longleftrightarrow (\Gamma = \Gamma') \text{ y } (\forall p \in P \ \widetilde{M}(p) \leq \widetilde{M'}(p))$$

El árbol de cobertura de una red reconfigurable marcada (N, M_0) se construye según el Algoritmo 7.

Proposición 191 *El árbol de cobertura de una red reconfigurable es finito.*

Una red reconfigurable está acotada si y solamente si ningún vértice de su árbol de cobertura está etiquetado por (Γ, M) *en donde* M *contiene una componente* ω*. Más exactamente, si* (Γ, M) *es algún marcado generalizado etiquetando un nodo del árbol de cobertura entonces, para todo entero* N*, existe un marcado alcanzable asociado con el mismo estado* (Γ, M') *tal que:*

1. $M(p_i) = \omega \Longrightarrow M'(p_i) \geq N$;

2. $M(p_i) \in \mathbb{N} \Longrightarrow M'(p_i) = M(p_i)$.

Demostración:

Se sigue del resultado análogo para los SAVEs [Reu89].

■

En el Capítulo 7 vamos a presentar el estado actual de una herramienta automática de simulación y verificación de propiedades para sistemas modelizados mediante redes reconfigurables. En dicha herramienta, entre otras posibilidades, se mecanizará el Algoritmo 7 para obtener el árbol de cobertura y comprobar si el sistema satisface o no las distintas propiedades.

6.2. Ecuación de Estados

El uso del álgebra lineal y la programación entera para propósitos de verificación tiene una larga tradición en la teoría de las redes de Petri [Mur89]. Una de las técnicas más conocidas es la *ecuación de estados* [Mur89][STC98]. Como hemos visto en el Capítulo 1, es una ecuación lineal que puede derivarse directamente de la descripción de la red y de su marcado inicial (en tiempo lineal). Puede verse como un conjunto de restricciones lineales $\mathcal{L}$ que todo marcado alcanzable debe satisfacer. En otras palabras, las soluciones de $\mathcal{L}$ son un superconjunto de los marcados alcanzables. Para hacer uso de la ecuación de estados, se le añaden nuevas restricciones lineales $\mathcal{L}_P$, especificando los marcados que no satisfacen una propiedad deseable P. Entonces, se utiliza la programación entera para resolver el sistema $\mathcal{L} \bigcup \mathcal{L}_P$: si el sistema no tiene solución, todo marcado alcanzable satisface P.

La desventaja de esta técnica es el hecho de que los marcados que satisfacen $\mathcal{L}$ son un superconjunto de los marcados alcanzables: las soluciones de $\mathcal{L} \bigcup \mathcal{L}_P$

pueden o no pueden corresponderse con marcados alcanzables. Así pues, la ecuación de estados es únicamente un método de semidecisión. Su principal ventaja es que no explora el espacio de estados y, por tanto, evita el problema de explosión de estados.

Un *problema de programación lineal* o problema lineal es un sistema $A \cdot X \leq B$ de (des)igualdades lineales llamadas *restricciones*, junto con una función lineal $C^T \cdot X$, llamada *función objetivo*. Una solución del problema es un vector de números racionales que satisface las restricciones. Una solución es óptima si maximiza el valor de la función objetivo (sobre el conjunto de todas las soluciones).

Un *problema de programación entera* consta de los mismos elementos que el problema de programación lineal, pero sólo se permiten soluciones enteras. En un *problema de programación mixta*, algunas variables tienen valores racionales y otras, valores enteros.

Un problema de programación lineal, entera o mixta es *factible o viable* si tiene solución. En caso contrario, es *inviable*.

Todo lugar p de una red tiene asociada una *ecuación de conservación de tokens*. Dada una secuencia de disparo $M_0 \xrightarrow{\sigma} M$, el número de tokens que p contiene en el marcado M es igual al número de tokens que contiene en el marcado M_0, más los tokens añadidos por (los disparos de) las transiciones de entrada de p, menos los tokens eliminados por las transiciones de salida. Si denotamos por $\#(\sigma, t)$ el número de veces que una transición t ocurre en σ, podemos escribir la ecuación de conservación de tokens para p como:

$$M(p) = M_0(p) + \sum_{t \in {}^\bullet p} \#(\sigma, t) W(t, p) - \sum_{t \in p^\bullet} \#(\sigma, t) W(p, t)$$

La ecuación de conservación de tokens para cada lugar se escribe habitualmente de la siguiente forma:

$$M = M_0 + A^T \cdot \overrightarrow{\sigma}$$

donde $\overrightarrow{\sigma} = (\#(\sigma, t_1), \ldots, \#(\sigma, t_n))$ es el llamado *vector de Parikh* de σ y A es la matriz de incidencia. Recordemos la Definición 40 de matriz de incidencia dada en el Capítulo 1.

La *matriz de incidencia* $A = [a_{ij}]$ de una red de Petri con n transiciones y m lugares es una matriz $(n \times m)$ de enteros tal que $a_{ij} = a_{ij}^+ - a_{ij}^-$ siendo

$a_{ij}^{+} = W(i,j)$ el peso del arco desde la transición i a su lugar de salida j y $a_{ij}^{-} = W(j,i)$ el peso del arco a la transición i desde su lugar de entrada j.

Si un marcado dado M es alcanzable desde M_0, entonces existe una secuencia σ que satisface $M_0 \xrightarrow{\sigma} M$. Así, el siguiente problema tiene al menos una solución:

$$\begin{array}{l} Variables : x = \overrightarrow{\sigma}, entera \\ M = M_0 + A^T \cdot x \\ x \geq 0 \end{array}$$

La ecuación $M = M_0 + A^T \cdot x$ es la llamada *ecuación de marcados* o *ecuación de estados*. Si esta ecuación no tiene solución entonces M no es alcanzable desde M_0.

Utilizando este método podemos analizar propiedades como la existencia o no existencia de marcados y/o secuencias de disparo que satisfagan algunas restricciones expresadas en términos de desigualdades lineales. Para la existencia, obtendremos condiciones necesarias, y para la no existencia, condiciones suficientes. También podemos computar (cotas para) el máximo de una función lineal, o semidecidir si existe. Además, como se ha visto en la Sección 1.3.2 del Capítulo 1, las propiedades estructurales (acotabilidad estructural, conservatividad, repetitividad, consistencia, etc.) también pueden definirse a partir de la matriz de incidencia y de sus ecuaciones asociadas.

Para obtener la ecuación de estados, en primer lugar, tenemos que construir la matriz de incidencia de una red reconfigurable. Para las redes reconfigurables, en dicha matriz, deben estar representados, de alguna forma, todos los cambios de configuración que pueden ocurrir como consecuencia de la aplicación de reglas de reescritura. Un cambio de configuración implica un cambio en el peso de los arcos de los lugares y las transiciones de la regla que se dispara. Así pues, el valor de un elemento de la matriz a_{ij} vendrá dado en función de los distintos cambios de configuración ocasionados por reglas que afecten a la transición i y el lugar j.

Actualmente, estamos realizando el estudio de la función que nos permita expresar todos los posibles cambios de peso que pueden tener lugar entre un lugar y una transición, debidos a la aplicación de las diferentes reglas de reescritura. A partir de dicha función, podremos construir la matriz de incidencia de una red reconfigurable y, por tanto, la ecuación de estados. Una vez obtenida la ecuación de estados se tratará de resolverla y de añadirle las restricciones que se consideren convenientes para analizar distintas propiedades.

Además, como trabajo futuro y en la misma línea, pensamos definir directamente las nociones de invariantes lineales, sifones y trampas para una red reconfigurable que nos permitirán extraer conclusiones sobre propiedades estructurales en la red.

Capítulo 7

Desarrollo de una herramienta software

Para poder estudiar sistemas de la vida real que puedan modelarse mediante las redes reconfigurables será útil disponer de alguna herramienta software que nos permita analizar la estructura y el comportamiento dinámico del sistema modelado y así, evaluarlo y sugerir mejoras o cambios. Hemos visto en el Capítulo 5 que el modelo de las redes reconfigurables es equivalente al modelo de las redes de Petri y, por tanto, las propiedades que son decidibles en las redes de Petri lo siguen siendo en nuestro modelo (ver Sección 5.4). Así pues, es factible la obtención de una herramienta de verificación automática para las redes reconfigurables. Nuestro objetivo es el desarrollo de una herramienta para el diseño y verificación de sistemas modelados utilizando redes reconfigurables, que integre un editor gráfico, un simulador y herramientas de análisis de distintas propiedades. Además, puesto que, como hemos demostrado en el Capítulo 5, los sistemas de reescritura de redes tienen el mismo poder expresivo que las máquinas de Turing y no es posible la obtención de una herramienta de verificación automática (ver Sección 5.3), nuestra herramienta, para aquellos sistemas modelados mediante sistemas de reescritura de redes, permitirá únicamente la simulación.

Durante la última década se han desarrollado muchas herramientas automáticas de diseño y verificación basadas en redes de Petri tanto para uso comercial como académico [Webd]. La mayoría de ellas están equipadas con un interfaz gráfico que permite la animación de los tokens y pueden llevar a cabo algún tipo de análisis como la alcanzabilidad o la verificación de una clase específica de propiedades. Más recientemente, la investigación en herramientas de redes de Petri se ha centrado en dos objetivos principales:

- desarrollar técnicas de análisis más potentes y
- conseguir la interacción de unas herramientas de redes de Petri con otras y también con otras herramientas que implementen otros métodos formales.

Como se indica en [Stö98], hay tres formas posibles de obtener una herramienta que integre un editor gráfico, un simulador y técnicas de análisis: implementándola uno mismo, añadiendo e integrando un conjunto de herramientas especializadas o adaptando y mejorando una herramienta integrada ya existente.

1. La primera aproximación garantiza que se cumplan todos los requerimientos, pero su inconveniente es que se necesita un enorme esfuerzo en programación.
2. La integración de herramientas especializadas también supone un gran esfuerzo debido a que las herramientas existentes utilizan distintos formatos y la documentación que las acompaña es muy pobre.
3. Respecto a la tercera aproximación, elegir una herramienta integrada y adaptarla/mejorarla, es necesario que la herramienta tenga una estructura abierta que nos permita adaptarla a nuestras necesidades y mejorarla. Sin embargo, obtener la información relevante sobre la estructura, el interfaz y las representaciones internas de las herramientas (comerciales o no) no es fácil.

Cualquiera de estas tres aproximaciones supone un esfuerzo considerable en implementación, tendremos que decidir en base a nuestros objetivos por cual de ellas nos decidimos. Para ello, será fundamental establecer cuáles son esos objetivos. Además, será importante también tener en cuenta la disponibilidad física de las aplicaciones existentes y los requerimientos específicos de hardware y software de dichas aplicaciones.

A continuación se exponen los principales objetivos que ha de cumplir la aplicación que pretendemos construir:

- Debe tener una GUI intuitiva, consistente y estable.
- Desde el editor gráfico el usuario introducirá la red cuyo comportamiento quiera simular y analizar. El modelo producido por el editor será, por tanto, el utilizado para validación y verificación. Por ello, conviene que dicho editor permita la creación de gráficos estéticamente agradables (alineación de los elementos de la red, color, estilo y anchura de líneas, estilo y tamaño de texto, etc), permita imprimir partes o todo el modelo, compruebe la sintaxis del grafo (p.e. la aparición de transiciones sin arcos de entrada o de salida, arcos no conectados, etc) y, en general, tenga todas las facilidades de cualquier editor gráfico.
- El simulador debe permitir distintos modos de simulación como la simulación continua o paso a paso, permitiendo la presentación en modo gráfico para ver los cambios que se producen en la estructura de la red tras el disparo de algún evento.
- Las herramientas de análisis deben permitir el análisis de cobertura y alcanzabilidad, el análisis de propiedades estructurales y otras propiedades específicas del modelo. Por ejemplo, a partir de la construcción del árbol de cobertura podrá determinarse si se cumplen distintas propiedades en la red, como la alcanzabilidad, acotabilidad, vivacidad, terminación, etc.

Establecidas las características fundamentales que vamos a requerir a la herramienta a desarrollar y teniendo en cuenta la disponibilidad de herramientas que ya implementan muchas de tales características, tenemos dos alternativas de implementación:

A Optar por la utilización de herramientas ya existentes que manipulen redes L/T, basándonos en la equivalencia entre las redes reconfigurables y las redes de Petri (ver Sección 5.3). Será necesario un traductor de redes reconfigurables a redes de Petri. El sistema a modelar se diseñará mediante una red reconfigurable que traduciremos a su red de Petri equivalente. Esta red de Petri será la utilizada por las herramientas ya existentes para simulación y análisis.

B Basándonos en que, tanto las configuraciones de una red reconfigurable como las partes izquierda y derecha de sus reglas de reescritura, son redes de Petri, hacer uso de un editor gráfico existente que nos permita representarlas y acceder a dicha representación para su adaptación y posterior manipulación por un simulador y un analizador implementados íntegramente por nosotros mismos.

En cualquier caso, nuestra herramienta va a consistir en tres partes diferenciadas: editor gráfico, simulador y analizador. Describimos, a continuación, cada una de ellas.

7.1. Editor gráfico

Tras el estudio de las herramientas disponibles con software libre y de fácil acceso, nos decantamos por PED, un editor de redes de Petri que será el soporte gráfico para capturar la red.

7.1.1. PED. Un editor de redes de Petri

PED es un editor de redes de Petri [Webb], que permite la construcción fragmentada y jerárquica de redes L/T, y su exportación a un rango amplio de formatos, incluyendo los aceptados por herramientas de análisis como PEP [Webc] e INA [Weba]. El concepto de jerarquía implica tanto diseño top-down (refinamiento) como bottom-up. Un navegador jerárquico integrado proporciona distintas facilidades de edición y navegación.

Las facilidades de edición son:

- Copiado y pegado de partes seleccionadas de las redes.
- Funcionalidades de alineación y rotación.
- Arcos inhibidores, arcos reset, arcos de test, varios tipos de arcos punteados y discontinuos.
- División y fusión de lugares y transiciones.
- Una función de búsqueda para encontrar elementos en las redes con atributos específicos.
- Otras facilidades de edición y exportación.

PED es un software libre para uso no comercial. Hay versiones disponibles para Solaris y Linux.

La interfaz gráfica de PED permite el diseño gráfico del sistema que queramos modelar tanto si optamos por la implementación **A** como por la implementación **B** (de la página anterior). El diseño, detallado a continuación, se almacena en ficheros, cuya información es la que tratamos y adaptamos a nuestro modelo de las redes reconfigurables.

7.1.2. Diseño de la red reconfigurable en PED

El usuario dibuja el estado inicial de la red reconfigurable del sistema a modelar utilizando PED, así como las distintas reglas de reescritura. Se obtienen distintos ficheros, uno para el estado inicial y uno para cada una de las reglas de reescritura.

Para extraer la información de los ficheros realizamos un análisis léxico de los mismos. De los lugares nos interesa su nombre y su marcado. De las transiciones su nombre. Y de los arcos, su nombre, su peso y los lugares y transiciones que conectan. Toda esta información se almacenará en distintas estructuras: un array de lugares, un array de transiciones y una matriz para los arcos, con tantas filas como lugares y tantas columnas como transiciones. Además, para cada elemento se añadirá también el nº de línea del fichero original en la que aparece.

En el caso de las reglas de reescritura, además, habría que relacionar los lugares y transiciones de la parte izquierda de la regla con los correspondientes lugares y transiciones de la parte derecha.

En el caso (**A**), implementamos un traductor que obtiene la red de Petri equivalente a la red reconfigurable inicial, siguiendo los pasos del Algoritmo 6 de la Sección 5.4. El resultado del traductor es un fichero con el mismo formato que los ficheros de PED, de forma que la red de Petri obtenida es visible desde dicho editor gráfico. El propio PED ofrece la posibilidad de exportar el diseño a otras herramientas de simulación y de análisis, como se indica en las secciones que siguen.

En el caso (**B**), las estructuras utilizadas para representar a la red reconfigurable serán las que utilicemos en la implementación de nuestro simulador y analizador.

Hasta este momento, disponemos, ya en funcionamiento, de la fase de diseño de la red reconfigurable. En la actualidad, nos encontramos desarrollando el resto de la herramienta. El lenguaje de programación utilizado para la implementación es Java 2 SDK, versión 1.4.1.

7.2. Simulador

7.2.1. Simulador A

En la implementación **A**, tendremos que elegir un simulador de los disponibles que sea compatible con el formato de la red resultado del editor PED. Nuestra elección es la herramienta PEP [Webc], un entorno de modelización, simulación y verificación basado en redes de Petri. Es un software libre disponible para Solaris y Linux.

La herramienta PEP

La herramienta PEP [Webc][BG98][Ste01] es un Entorno de Programación basado en redes de Petri. Permite el desarrollo, simulación y verificación de programas paralelos con una variedad de herramientas y formalismos mediante una interfaz de usuario gráfica.

PEP es, principalmente, un entorno de desarrollo integrado que incluye una herramienta completa para la edición y simulación de redes de Petri de alto y bajo nivel, y una gran base de herramientas de análisis y verificación basadas en dichas redes.

La estructura completa de la herramienta PEP consiste en tres niveles: el nivel de desarrollo, el nivel de red y el nivel análisis.

En el nivel de desarrollo se encuentran los módulos de entrada, distintos editores textuales y gráficos. Estas componentes facilitan el diseño de sistemas paralelos mediante programas paralelos, álgebra de procesos o redes de Petri. Los compiladores de PEP generan redes de Petri a partir de tales modelos.

En el nivel de red, se encuentran los simuladores de PEP que permiten la simulación de las redes de forma automática o dirigida por el usuario.

El nivel de análisis incluye varios algoritmos de verificación de redes de Petri para testear, por ejemplo, propiedades de alcanzabilidad y ausencia de interbloqueos, así como algoritmos de model checking.

PEP también está provisto de una buena interfaz para el paquete INA de Starke [RS99], en el que, como veremos, están implementados una gran variedad de algoritmos para el análisis estructural y de comportamiento.

7.2.2. Simulador B

En la implementación **B**, el simulador debe ofrecer, partiendo de un estado inicial, distintos tipos de simulación:

1. Continua. La aplicación es quien decide aleatoriamente de los posibles eventos habilitados, cual se dispara, sin consulta alguna al usuario, permitiéndole la interrupción de la simulación en cualquier momento.

2. Paso a paso. En cada estado alcanzado se muestran los distintos eventos habilitados (transiciones y reglas de reescritura) y el usuario es quien decide cuál de ellos debe dispararse.

3. Mixta con respecto a las configuraciones. Permite al usuario el cambio de configuración en cualquier momento. Consiste en: interrumpir la simulación en un estado, mostrar sólo las distintas reglas de reescritura que estén habilitadas en dicho estado, elegir una para cambiar de configuración (es decir, cambiar la estructura, no el marcado) y continuar la simulación de forma automática. De esta forma, la simulación se convierte en automática para una configuración dada y es el usuario el que decide el cambio a otra de las configuraciones posibles para el estado actual de la red.

Además, como se ha comentado anteriormente, este simulador no sólo va a permitir simular redes reconfigurables sino también sistemas de reescritura de redes.

7.3. Analizador

7.3.1. Analizador A

En el caso de la implementación **A**, igual que para el simulador, elegiremos de entre las herramientas de análisis disponibles, alguna que sea compatible con la representación indicada, por ejemplo, la herramienta INA [Weba] que describimos a continuación. Es una herramienta de software libre, con versiones para Solaris, Linux y MS-Windows.

La herramienta INA

La herramienta INA [Weba][RS99] es una herramienta de análisis interactivo que incorpora un gran número de métodos potentes de análisis de redes L/T. Estos métodos incluyen el análisis de:

1. propiedades estructurales, tales como análisis de traps e interbloqueos, análisis mediante P- y T-invariantes, acotamiento estructural;

2. propiedades de comportamiento, tales como acotabilidad, seguridad, vivacidad, ausencia de interbloqueos y de conflictos dinámicos;

3. propiedades específicas definidas por el usuario, como las definidas mediante predicados y fórmulas de la lógica temporal de restricciones.

Todos estos análisis emplean varias técnicas como métodos de álgebra lineal (para invariantes), grafo de cobertura y alcanzabilidad, grafo reducido de alcanzabilidad basado en conjuntos stubborn y simetrías.

La herramienta INA utiliza una combinación de técnicas interactivas, donde el usuario especifica y pregunta, y técnicas de procesamiento de ficheros. El formato básico de fichero de red de Petri es compatible con otras herramientas, tales como PED y PEP.

7.3.2. Analizador B

El analizador a desarrollar, en la implementación **B**, aplicará las distintas técnicas presentadas en el Capítulo 6 para realizar la verificación de las diferentes propiedades.

Los ficheros obtenidos como resultado del diseño de la red reconfigurable serán la entrada para este analizador. El usuario podrá elegir entre la construcción del árbol de cobertura (según el Algoritmo 7 de la Sección 6.1) o la aplicación de técnicas estructurales basadas en la obtención de la ecuación de estados (ver Sección 6.2).

Capítulo 8

Conclusiones y trabajos futuros

Bajo el título

"Redes Reconfigurables. Modelización y Verificación"

en esta Tesis Doctoral se ha desarrollado formalmente una extensión de las redes de Petri que permite el modelado, la simulación y la verificación de sistemas concurrentes sujetos a cambios dinámicos: las *redes reconfigurables*, que son un caso particular de los *sistemas de reescritura de redes*.

Partiendo de que las redes de Petri no ofrecen la posibilidad de representar directamente el cambio dinámico en la propia estructura de la red, nuestro fin ha sido conseguir un modelo en el que dicho cambio se permita y además se trate de manera local, interna e incremental, es decir, teniendo en cuenta los cambios estructurales desde el inicio del proceso de diseño y en cualquier etapa del mismo.

Inicialmente se han revisado y comparado los formalismos existentes que mejoran la expresividad del modelo básico de redes de Petri y permiten la descripción de sistemas concurrentes dinámicos, centrándonos en aquellas características similares y de interés para la formulación de nuestro modelo.

Las dos líneas de investigación en las que nos hemos basado y que mejoran la expresividad del modelo básico de las redes de Petri para soportar la descripción de sistemas concurrentes sujetos a cambios dinámicos son: las *gramáticas de grafos* y las *redes automodificantes de Valk*.

- Se han presentado las *gramáticas de grafos* [Ehr87][Sch93][Cor95][Bal00] como un formalismo alternativo a las redes de Petri para describir sistemas concurrentes; en concreto, hemos visto cómo las reglas de reescritura de grafos permiten modelizar los cambios dinámicos de la estructura de la red. En esta línea, se han definido los sistemas de reescritura de redes como combinación de redes de Petri con sistemas de reescritura de grafos. Las reglas de reescritura de los sistemas de reescritura de redes son muy similares a las producciones de las gramáticas de grafos (ambas constan de parte izquierda, parte derecha e interfaz) y la aplicación de una regla de reescritura es como una derivación directa en las gramáticas de grafos (bajo ciertas condiciones, cuando se detecta una ocurrencia (*match*) de la parte izquierda de una producción en un grafo, puede reemplazarse por su parte derecha).

- Se han presentado las *redes automodificantes* [Val78a, Val78b, Val81] resaltando los aspectos más relevantes con respecto al problema de la modelización de cambios dinámicos dentro de los sistemas concurrentes. También se ha analizado la decidibilidad o indecidibilidad de las propiedades de las distintas subclases de redes automodificantes. Como en estas redes, modelamos un sistema como un conjunto de redes de Petri, llamadas *configuraciones*, junto con algún mecanismo que permite al sistema evolucionar de una configuración a otra bajo ciertas circunstancias.

Ambas propuestas dan lugar a modelos expresivos que aportan indudables ventajas con respecto a los modelos ya existentes. Sin embargo, la mayoría (si no todas) las propiedades básicas decidibles de las redes de Petri (acotabilidad de lugares, alcanzabilidad, interbloqueo y vivacidad) se pierden en estos modelos. Como consecuencia no pueden construirse herramientas automáticas de verificación para dichos modelos extendidos.

Los *sistemas de reescritura de redes* de [BLO02, BLO03][LO01, LO03b, LO03c, LO03a] combinan las redes de Petri con sistemas de reescritura de grafos. La idea es describir una configuración del sistema como una red de Petri y un cambio de configuración como una regla de reescritura de grafos que consiste en reemplazar parte del sistema (que hace *matching* con la parte izquierda de la regla de reescritura) por otra (que viene dada por la parte derecha de la regla). Hemos demostrado que los sistemas de reescritura de redes tienen el poder expresivo de la máquina de Turing, es decir, las propiedades decidibles de las redes de Petri se pierden en nuestro modelo. Y, por ello, no es posible la verificación automática de propiedades en el mismo.

Hemos definido las *redes reconfigurables* [BLO02, BLO03][LO01, LO03b, LO03c, LO03a] como una subclase de sistemas de reescritura de redes, en la que el cambio de configuración se limita a la modificación de las relaciones de flujo de los lugares implicados en la regla de reescritura. Las *redes reconfigurables* combinan las características más significativas de las dos aproximaciones en las que nos basamos de manera que las propiedades fundamentales son decidibles, con lo que en este modelo es factible la verificación automática.

Antes de mostrar la definición de nuestra propuesta final, se han presentado los estudios previos sobre las redes reconfigurables, que comprenden la definición de dichas redes, el estudio de la decidibilidad de algunas propiedades estructurales mediante distintas técnicas de análisis como la construcción del árbol de cobertura y la equivalencia de estas redes con otros formalismos (como las *redes de Petri Controladas por un Autómata*). Nuestra definición de red reconfigurable es una reformulación de las llamadas *redes reconfigurables reversibles.*

Hemos visto que en la literatura hay otros modelos que permiten la descripción de complejos sistemas concurrentes dinámicos:

- las *redes móviles* de Asperti y Busi [AB96] y las *redes dinámicas* de Buscemi y Sassone [BS01] que permiten la creación dinámica de componentes como en nuestra propuesta,
- las *redes* Δ de Gradit y Vernadat [GVA99, GV99], que integran las ventajas de las redes de Petri y las gramáticas de grafos, respectivamente, para modelizar sistemas workflow,
- el *formalismo POP* introducido por Engelfriet, Leih y Rozenberg [ELR91], y el modelo relacionado de los *autómatas cooperativos* de Badouel, Darondeau y Tokmakoff [BDT99], en los que los tokens son elementos activos con comportamiento dinámico.

Para todos ellos, como en nuestro modelo, la descripción de los cambios es interna e incremental y su tratamiento es local. La idea de la reescritura también subyace en todas las propuestas, la configuración del sistema se describe como una red de Petri y un cambio de configuración como una regla de reescritura de grafos que reemplaza la parte del sistema que hace *matching* con la parte izquierda de la regla de reescritura por la correspondiente parte derecha. Con respecto a la expresividad, todas son equivalentes a la máquina de Turing, como nuestro modelo. Pero el modelo de los sistemas de reescritura de redes es más intuitivo puesto que es, pensamos, un modelo más cercano a las redes de Petri.

Hemos distinguido dos clases particulares de sistemas de reescritura de redes de acuerdo con su relación de transferencia τ. El modelo de las redes reconfigurables es un sistema de reescritura de redes en el que la relación de transferencia τ es una biyección. Este modelo, aunque formalmente es equivalente a las redes de Petri, nos permite expresar de una forma más fácil y concisa sistemas en los que pueden ocurrir cambios estructurales dinámicamente. No obstante, la traducción automática a redes de Petri asegura que todas las propiedades fundamentales de las redes de Petri siguen siendo decidibles para las redes reconfigurables. Para este modelo podemos encontrar, por tanto, herramientas de verificación automáticas. La traducción a redes de Petri equivalentes incrementa considerablemente el tamaño de la red y, por ello, resulta más eficiente implementar los métodos de verificación de propiedades de las redes de Petri directamente en el modelo original. En concreto, uno de los métodos de análisis (por enumeración), la construcción del árbol de cobertura para una red reconfigurable nos ha permitido decidir las propiedades de acotamiento de un lugar, alcanzabilidad, vivacidad e interbloqueo.

Para hacer uso de métodos de análisis estructurales directamente en una red reconfigurable es necesario construir una matriz de incidencia que refleje todos los posibles cambios de configuración que pueden ocurrir debidos a la aplicación de reglas de reescritura. A partir de dicha matriz de incidencia, se obtiene la ecuación de estados, cuya solución como problema lineal es una condición necesaria pero no suficiente para la alcanzabilidad. Además, las nociones de invariantes lineales, sifones y trampas también pueden definirse directamente para una red reconfigurable y nos van a permitir decidir, entre otras, propiedades de comportamiento como acotabilidad y vivacidad y también propiedades estructurales. Estos son los tópicos de nuestra investigación actual.

Estamos desarrollando una herramienta para el diseño y verificación de sistemas modelados utilizando redes reconfigurables, que integra un editor gráfico, un simulador y un analizador de distintas propiedades. El analizador mecanizará los métodos de análisis por enumeración y estructurales comentados anteriormente. Para la clase de los sistemas de reescritura de redes puesto que tiene el poder computacional de la máquina de Turing, no es posible la verificación automática. Sin embargo, este modelo sigue siendo interesante como herramienta de modelización y simulación. Uno de nuestros objetivos futuros es implementar un simulador para sistemas de reescritura de redes.

Otra posible línea de actuación futura es tener en cuenta en la definición de red reconfigurable el hecho de que muchas reconfiguraciones en sistemas reales sólo ocurren bajo ciertas circunstancias que dependen del estado del sistema, es decir, hacer que la aplicación de una regla de reescritura no dependa únicamente de la topología de la red sino también del marcado de la misma [BLO04]. La idea básica es incorporar un mecanismo de control que permita al usuario la decisión del cambio de configuración.

Referencias

[AB96] A. Asperti and N. Busi. Mobile Petri Nets. Technical Report UBLCS-96-10, University of Bologna (Italy). Department of Computer Science, 1996.

[AK77] T. Araki and T. Kasami. Some Decision Problems Related to the Reachability Problem for Petri Nets. *Theoretical Computer Science*, 3(1):85–104, 1977.

[Bad96] E. Badouel. Splitting of Actions, Higher Dimensional Automata and Net Synthesis. Technical Report PI-3013, Inria Research Report, France, 1996.

[Bal00] P. Baldan. *Modelling Concurrent Computations: From Contextual Petri Nets to Graph Grammars*. Phd thesis, Computer Science Department, University of Pisa (Italy), 2000.

[BD97] E. Badouel and PH. Darondeau. Stratified Petri Nets. In *Proc. of Fundamentals of Computation Theory (FCT'97)*, volume 1279 of *Lecture Notes in Computer Science*, pages 117–128. Springer Verlag, 1997.

[BDT99] E. Badouel, Ph. Darondeau, and A. Tokmakoff. Modelling Dynamic Agents Systems with Cooperating Autómata. In *Int. Conf. on Parallel and Distributed Processing Techniques and Applications (PDPTA'99)*, pages 11–17, Las Vegas, Nevada (USA), June 28-July 1 1999. CSREA Press.

[BG98] E. Best and B. Grahlmann. *PEP: Programming Environement based on Petri nets. Documentation and User Guide Version 1.8.* University of Oldenburg, Germany, December 1998.

[BLO02] E. Badouel, M. Llorens, and J. Oliver. Modelling Concurrent Systems using Reconfigurable Nets. In *Actes du 6eme Seminaire Atelier en Algèbre, Logique, et ses Applications*, Yaoundé, Cameroon, Mars 25-29 2002.

[BLO03] E. Badouel, M. Llorens, and J. Oliver. Modelling Concurrent Systems: Reconfigurable Nets. In H.R. Arabnia and Youngsong Mun, editors, *Int. Conf. on Parallel and Distributed Processing Techniques and Applications (PDPTA'03)*, volume IV, pages 1568–1574, Las Vegas, Nevada (USA), June 23-26 2003. CSREA Press.

[BLO04] E. Badouel, M. Llorens, and J. Oliver. Marked-Controlled Reconfigurable Nets: Petri Nets with Structural Dynamic Changes. In *Int. Meeting On High Performance Computing for Computational Science (VECPAR'04)*, Lecture Notes in Computer Science, Valencia (Spain), Junio 28-30 2004. Springer Verlag. Pendiente de aceptación.

[BML80] G. Birkhoff and S. Mac Lane. *Álgebra Moderna*. Vicens Vives, 1980.

[BO98a] E. Badouel and J. Oliver. Reconfigurable Nets, a Class of High Level Petri Nets Supporting Dynamic Changes Within Workflow Systems. In Deparment of Mathematics Eindhoven University of Technology and Computing Science, editors, *Proc. of the Workshop Workflow Management Systems: Net-Based Concepts, Models, Techniques and Tools (WFM'98)*, volume 98/07 of *CSR*, pages 129–145, Lisbon, Portugal, 1998.

[BO98b] E. Badouel and J. Oliver. Reconfigurable Nets, a Class of High Level Petri Nets Supporting Dynamic Changes Within Workflow Systems. Technical Report PI-1163, Inria Research Report, France, 1998.

[BO99] E. Badouel and J. Oliver. Dynamic Changes in Concurrent Systems: Modelling and Verification. Technical Report PI-3708, Inria Research Report, France, 1999.

[Bou96] Z. Bouziane. *Algorithmes Primitifs Récursifs et Problèmes EXPSPACE-complets pour les Réseaux de Petri Cycliques*. Phd thesis, École Normale Supérieure de Cachan (France), 1996.

[Bou98] Z. Bouziane. A Primitive Recursive Algorithm for the General Petri Net. In *Proc. of 39th. IEEE Symposium Foundations of Computer Science (FOCS'98)*, pages 130–136, Palo Alto, Ca, (USA), 1998. IEEE Computer Society Press.

[BS01] M. Buscemi and V. Sassone. High-level Petri Nets as Type Theories in the Join Calculus. In *Proc. of the 4th. Int. Conf. on Foundations of Software Science and Computation Structures (FoSSaCS'01)*, volume 2030 of *Lecture Notes in Computer Science*, pages 104–120, Genova, Italy, 2001. Springer.

[Cia94] G. Ciardo. Petri Nets with Marking-Dependent Arc Cardinality: Properties and Analysis. In *Proc. of Int. Conf. on Application and Theory of Petri Nets (ICATPN'94)*, volume 815 of *Lecture Notes in Computer Science*, pages 179–198. Springer Verlag, 1994.

[CLM76] E. Cardoza, R. Lipton, and A. Meyer. Exponential Space Complete Problems for Petri Nets and Commutative Semigroups. In *Proc. of 8th. ACM Symposium on Theory of Computing*, pages 50–54, Hershey, PA (USA), 1976. ACM Press.

[Cor95] A. Corradini. Concurrent Computing: From Petri Nets to Graph Grammars. *Electronic Notes in Theoretical Computer Science*, 2, 1995.

[Cor96] A. Corradini. Concurrent Graph and Term Graph Rewriting. In *Proc. of 7th. Int. Conf. on Concurrency Theory (CONCUR'96)*, volume 1119 of *Lecture Notes in Computer Science*, pages 438–464, Pisa, Italy, 1996. Springer-Verlag.

[DE95] J. Desel and J. Esparza. Free Choice Petri Nets. In *Cambridge Tracts in Theoretical Computer Science*, volume 40. Cambridge University Press, Cambridge, UK, 1995.

[DF97] C. Dufourd and A. Finkel. Polynomial-Time Many-One Reductions for Petri Nets. In *Proc. of 17th. Conf. Foundation of Software Technology and Theory of Computer Science (FST and TCS'97*, volume 1346 of *Lecture Notes in Computer Science*, pages 312–326, Kharagpur, India, 1997. Springer Verlag.

[DFS98] C. Dufourd, A. Finkel, and P. Schnoebelen. Reset Nets Between Decidability and Undecidability. In Kim Guldstrand Larsen, Sven Skyum, and Glynn Winskel, editors, *Proc. of 25th. Int. Coll. on Automata, Languages, and Programming (ICALP'98)*, volume 1443 of *Lecture Notes in Computer Science*, pages 103–115, Aalborg, Denmark, 1998. Springer Verlag.

[Die97] R. Diestel. Graph Theory. In *Graduate Texts in Mathematics*, volume 173. Springer-Verlag, New York, 1997.

[DJS99] C. Dufourd, P. Jancar, and P. Schnoebelen. Boundedness of Reset P/T Nets. In *Proc. of 26th. Int. Coll. on Automata, Languages, and Programming (ICALP'99)*, volume 1644 of *Lecture Notes in Computer Science*, pages 301–310, Prague, Czech Republic, 1999. Springer Verlag.

[DME98] G. De Michelis and C. A. Ellis. Computer Supported Cooperative Work and Petri Nets. *Lecture Notes in Computer Science*, 1492:125–153, 1998.

[Duf98] C. Dufourd. *Réseaux de Petri avec Reset/Transfert: Décidabilité et Indécidabilité*. Phd thesis, École Normale Supérieure de Cachan (France), 1998.

[Ehr87] H. Ehrig. Tutorial Introduction to the Algebraic Approach of Graph Grammars. In G. Rozenberg H. Ehrig, M.Ñagl and A. Rosenfeld, editors, *Proc. of 3rd. Int. Workshop on Graph Grammars and Their Application to Computer Science*, volume 291 of *Lecture Notes in Computer Science*, pages 3–14, Warrenton, Virginia, USA, 1987. Springer-Verlag.

[EKR95] C. A. Ellis, K. Keddara, and G. Rozenberg. Dynamic Change Within Workflow Systems. In *Proc. of the Conf. on Organizational Computing Systems (COOCS'95)*, Business Processes and Workflows, pages 10–21. ACM Press, New York, 1995.

[ELR91] J. Engelfriet, G. Leih, and G. Rozenberg. Net Based Description of Parallel Object-Based Systems, or POTs and POPs. In W.P. de Roever J.W. de Bakker and G. Rozenberg, editors, *Workshop Foundations of Object-Oriented Languages (FOOL'90)*, volume 489 of *Lecture Notes in Computer Science*, pages 229–273, Noordwijkerhout, Netherlands, 1991. Springer-Verlag.

[EN93] C. A. Ellis and G. J. Nutt. Modeling and Enactment of Workflow Systems. In M. Ajmone Marsan, editor, *Proc. of Int. Conf. on Application and Theory of Petri Nets (ICATPN'93)*, volume 691 of *Lecture Notes in Computer Science*, pages 1–16. Springer Verlag, 1993.

[EN94] J. Esparza and M. Nielsen. Decidability Issues on Petri Nets - a Survey. *Bulletin of the European Association for Theoretical Computer Science (EATCS)*, 52:254–262, 1994.

[Esp98] J. Esparza. Decidability and Complexity of Petri Nets Problems - An Introduction. In G. Rozenberg and W. Reisig, editors, *Lectures on Petri Nets I: Basic Models. Advances in Petri Nets*, volume 1491 of *Lecture Notes in Computer Science*, pages 374–428. Springer Verlag, 1998.

[FG96] C. Fournet and G. Gonthier. The Reflexive Chemical Abstract Machine and the Join-Calculus. In *Proc. of 23rd. ACM Symposium on Principles of Programming Languages (POPL'96)*, pages 372–385, St. Petersburg Beach, Florida (USA), 1996. ACM Press.

[FGL+96] C. Fournet, G. Gonthier, J. Lévy, L. Maranget, and D. Rémy. A Calculus of Mobile Agents. In *Proc. of 7th. Int. Conf. on Concurrency Theory (CONCUR'96)*, volume 1119 of *Lecture Notes in Computer Science*, pages 406–421, Pisa, Italy, 1996. Springer-Verlag.

[Fin90] A. Finkel. Reduction and Covering of Infinite Reachability Trees. *Information and Computation*, 89(2):144–179, 1990.

[Fin93] A. Finkel. The Minimal Coverability Graph Algorithm. volume 674 of *Lecture Notes in Computer Science*, pages 210–243. Springer-Verlag, 1993.

[FMP96] A. Finkel, P. Mckenzie, and C. Picaronny. Minimal Coverability Trees for High-Level Nets. Technical Report 96-3, École Normale Supérieure de Cachan, France, Mars 1996.

[GV99] P. Gradit and F. Vernadat. Integrating Topology in Petri Nets by Use of Rewriting. Technical Report 96039, Laboratoire dÁnalyse et dÁrquitecture des Systèmes (LAAS), France, January 1999.

[GVA99] P. Gradit, F. Vernadat, and P. Azéma. Layered Delta-Net Specification of a Workshop. In *Int. Conf. on Parallel and Distributed Processing Techniques and Applications (PDPTA'99)*, volume VI, pages 2808–2814, Las Vegas, (USA), 26 June - 1 July 1999.

[Hac74] M. Hack. The recursive equivalence of the reachability problem and the liveness problem for Petri nets and Vector addition systems. In *Proc. of the 15th Annual Symposium on Switching and Automata Theory (SWAT)*, pages 156–164, New Orleans, USA, 1974. IEEE Computer Society Press.

[Hac76a] M. Hack. *Decidability Questions for Petri Nets*. Phd thesis, 1976.

[Hac76b] M. Hack. Petri Net Languages. Technical Report 159, Massachusetts Institute of Technology (M.I.T.), Cambridge, Massachusetts, March 1976.

[HU79] J.E. Hopcroft and J.D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, 1979.

[Jan00] P. Jančar. Bouziane's Algorithm for the Petri Net Reachability Problem is Incorrect. Technical report, Technical University of Ostrava, Czech Republic, 2000.

[Jen92] K. Jensen. *Coloured Petri Nets: Basic Concepts, Analysis, Methods and Practical Use*. EATCS Monographs in Computer Science. Springer Verlag, 1992.

[KM69] R.M. Karp and R.E. Miller. Parallel Program Schemata. *Journal of Computer System Science*, 3(2):147–195, 1969.

[Kos82] R. Kosaraju. Decidability of Reachability in Vector Addition Systems. In *Proc. of 14th. ACM Symposium on Theory of Computing*, pages 267–281, San Francisco (USA), 1982. ACM Press.

[Lip76] R.J. Lipton. The Reachability Problem Requires Exponential Space. Technical Report 62, Yale University, Department of Computer Science, USA, 1976.

[LO01] M. Llorens and J. Oliver. Modelización de Sistemas Concurrentes mediante Redes Reconfigurables. In *IX Jornadas de Concurrencia*, pages 213–224, Sitges, Barcelona (Spain), 2001.

[LO03a] M. Llorens and J. Oliver. Cambios Estructurales en Sistemas Concurrentes: Redes Reconfigurables. *Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial*, 2003. Pendiente de aceptación.

[LO03b] M. Llorens and J. Oliver. Sistemas de Reescritura de Redes. In *Concurrencia y Sistemas Distribuidos. Actas de las XI Jornadas de Concurrencia*, pages 237–250, Benicassim, Castellón (Spain), 2003.

[LO03c] M. Llorens and J. Oliver. Structural Changes in Concurrent Systems: Reconfigurable Nets. *IEEE Transactions on Computers*, 2003. Pendiente de aceptación.

[May84] E.W. Mayr. An Algorithm for the General Petri Net Reachability Problem. *SIAM Journal on Computing*, 13(3):441–460, 1984.

[Mil91] R. Milner. The Polyadic π-Calculus: A Tutorial. Technical Report ECS-LFCS-91-180, October 1991.

[MPW92] R. Milner, J. Parrow, and D. Walker. A Calculus of Mobile Processes. *Journal of Information and Computation*, 100(1):1–77, 1992.

[Mur89] T. Murata. Petri Nets: Properties, Analysis and Applications. In *Proc. of the IEEE*, volume 77, pages 541–580, April 1989.

[Pet62] C.A. Petri. *Kommunikation mit Automaten.* Phd thesis, 1962.

[Pet81] J. L. Peterson. *Petri Net Theory and the Modeling of Systems.* Englewood Cliffs, NJ: Prentice-Hall, USA, 1981.

[Rac78] C. Rackoff. The Covering and Boundedness Problems for Vector Addition Systems. *Theoretical Computer Science*, 6(2):223–231, 1978.

[Rei85] W. Reisig. Petri Nets: An Introduction. In W. Brauer, G. Rozenberg, and A. Salomaa, editors, *EATCS Monographs on Theoretical Computer Science*, volume 4. Springer-Verlag, Berlin, Germany, 1985.

[Rei95] K. Reinhardt. Reachability in Petri Nets with Inhibitor Arcs. 1995.

[Reu89] CH. Reutenauer. *Aspects Mathématics Des Réseaux de Petri.* Masson, Paris, 1989.

[RS99] S. Roch and P.H. Starke. *INA: Integrated Net Analyzer, Version 2.2.* Humboldt-Univerität zu Berlin, Institut für Informatik, April 1999.

[Sch93] H. Schneider. Graph Grammars as a Tool to Define the Behavior of Processes Systems: From Petri Nets to Linda. In *Fourth International Conference on Graph Grammars*, Williamsburg, USA, 1993.

[STC98] M. Silva, E. Teruel, and J.M. Colom. Linear Algebraic and Linear Programming Techniques for the Analysis of Place/Transition Net Systems. In G. Rozenberg and W. Reisig, editors, *Lectures on Petri Nets I: Basic Models. Advances in Petri Nets*, volume 1491 of *Lecture Notes in Computer Science*, pages 309–373. Springer Verlag, 1998.

[Ste01] C. Stehno. PEP Version 2.0. Tool demonstration ICATPN 2001, Newcastle upon Tyne, 2001.

[Stö98] H. Störrle. An Evaluation of High-End Tools for Petri Nets. Technical report, Ludwig-Maximilians-Universität München, Institut für Informatik, June 1998.

[Sud88] T.A. Sudkamp. *Languages and Machines. An Introduction to the Theory of Computer Science.* Addison-Wesley, 1988.

[Val78a] R. Valk. On the Computational Power of Extended Petri Nets. In *Proc. of 7th Int. Symposium on Mathematical Foundations of Computer Science (MFCS'78)*, volume 64 of *Lecture Notes In Computer Science*, pages 526–535, Zakopane, Poland, 1978. Springer-Verlag.

[Val78b] R. Valk. Self-Modifying Nets, a Natural Extension of Petri Nets. In Giorgio Ausiello and Corrado Böhm, editors, *Proc. of the 5th Int. Coll. Automata, Languages and Programming (ICALP'78)*, volume 62 of *Lecture Notes In Computer Science*, pages 464–476, Udine, Italy, 1978. Springer-Verlag.

[Val81] R. Valk. Generalizations of Petri Nets. In Jozef Gruska and Michal Chytil, editors, *Proc. of 10th. Symposium on Mathematical Foundations of Computer Science (MFCS'81)*, volume 118 of *Lecture Notes in Computer Science*, pages 140–155, Strbske Pleso, Czechoslovakia, 1981. Springer-Verlag.

[VDA97] W. M. P. Van Der Aalst. Verification of Workflow Nets. In P. Azéma and G. Balbo, editors, *Proc. of Int. Conf. on Application and Theory of Petri Nets (ICATPN'97)*, volume 1248 of *Lecture Notes in Computer Science*, pages 407–426. Springer Verlag, 1997.

[VDAVH02] W. M. P. Van Der Aalst and K. Van Hee. *Workflow Management: Models, Methods and Systems.* Cambridge: MIT Press, 2002.

[Weba] INA. http://www.informatik.uh-berlin.de/∼starke/ina.html - The Home Page of INA (an Integrated Net Analyzer).

[Webb] PED. http://www-dssz.Informatik.TU-Cottbus.de/∼wwwdssz - The Home Page of PED (a Hierarchical Petri Net Editor).

[Webc] PEP. http://theoretica.informatik.uni-oldenburg.de/∼pep/ - The Home Page of PEP (a Programming Environement based of Petri Nets).

[Webd] The Home Page of Petri Net Tools on the Web: http://www.daimi.aau.dk/petrinet/tools/.

[Webe] WfMC. http://www.wfmc.org/ - The Home Page of the WfMC (Workflow Management Coalition).