xx(101581.1)

CINVESTAV

Centro de Investigación y de Estudios Avanzados del IPN
Unidad Guadalajara

Prueba de Validez de Argumentos Lógicos y Búsqueda de Modelos Utilizando Redes de Petri

Tesis que presenta
José Salvador Navarro Contreras

Para obtener el grado de
Maestro en Ciencias

En la especialidad de
Ingeniería Eléctrica

Guadalajara, Jal., Noviembre de 2001

Prueba de Validez de Argumentos Lógicos y Búsqueda de Modelos Utilizando Redes de Petri

Tesis de Maestría en Ciencias
Ingeniería Eléctrica

Por:

José Salvador Navarro Contreras

Ingeniero en Sistemas Computacionales
Instituto Tecnológico de Tepic 1992-1997

Becario del CONACYT, expediente no. **121130**

Director de Tesis
Dr. Raúl Ernesto González Torres

CINVESTAV del IPN Unidad Guadalajara, Noviembre de 2001

Agradecimientos

Primeramente a Dios por haberme permitido llegar a mi meta.

A mis Padres y hermanos por su apoyo incondicional.

A mi asesor, Dr. Raúl Ernesto González Torres, por ayudarme a llevar a buen fin el presente trabajo.

A los revisores de la tesis, Dr. Antonio Ramírez Treviño, Dr. Manuel Edgardo Guzmán Rentería y Dr. Arturo del Sagrado Corazón Sánchez Carmona, por ayudarme a detectar y corregir los errores de este trabajo.

A los profesores del CINVESTAV y a todos aquellos que han ayudado a mi formación académica y personal.

A mi hermana Lidia, sin su ayuda no hubiese iniciado esta etapa.

A Lupita por su ayuda durante mi estancia en CINVESTAV.

A Diana, Lares, Pilar y demás compañeros en Universidad LaSalle, por la gran ayuda brindada.

Finalmente Ana Raquel, Santiago, Román, Alejandra, Luis Isidro, María Meda, por haberme apoyado durante este tiempo.

A todo ellos muchas gracias.

Becario de CONACYT número 121130.

Índice de Figuras

Índice General

Introducción

La lógica proposicional es un modelo matemático para razonar acerca de la validez de expresiones lógicas. En la lógica proposicional las expresiones sólo toman uno de dos valores posibles: verdadero y falso.

Podemos manipular a las expresiones lógicas de acuerdo a ciertas leyes algebraicas, permitiéndonos razonar formalmente (usando razonamiento deductivo) en base a un conjunto de premisas. Existen varias técnicas que pueden ser utilizadas para probar una hipótesis a partir de un conjunto de premisas. La más conocida y que se estudia en cualquier curso básico de lógica es el que usa tablas de verdad, aunque cabe resaltar que tal método es ineficiente para todos los casos. Una técnica utilizada ampliamente por su sencillez, es la basada en resolución y aunque es mejor que la técnica basada en tablas de verdad, esta técnica no nos dice por dónde comenzar ó continuar en el proceso. Una mejora a la técnica de resolución, es hiper-resolución, la cual engloba varios pasos de resolución en uno sólo. Al igual que sucede con resolución, en hiper-resolución tampoco sabemos por dónde comenzar y/o continuar, en la mayoría de los casos.

Algunas áreas de aplicación en las que los mecanismos deductivos pueden ser utilizados son las siguientes.

- Lenguajes de programación: quizá el lenguaje de programación más conocido basado en el principio de resolución sea PROLOG (Programming in Logic), en el cual se pueden implementar una gran variedad de sistemas (principalmente relacionados con problemas lógicos), así que la aplicación de los mecanismos deductivos se puede extender al de estos sistemas. Algunos lenguajes de programación que han surgido recientemente, son los llamados lenguajes formales, los cuales tratan de prevenir tanto como les es posible, errores lógicos que puedan introducir fallas a la ejecución de la aplicación que se ha implementado en ellos.
- Desarrollo de programas: todo programa es susceptible a fallas, algunas de ellas son producto de errores en su código. Tales errores, pueden ser prevenidos usando herramientas formales basadas en mecanismos deductivos. Aunque lo anterior se realiza durante las fases de desarrollo de los programas, también se pueden utilizar para verificar programas ó hardware existente ó en funcionamiento.
- Representación e inferencia del conocimiento: durante años el hombre ha tratado de introducir cierta inteligencia a las máquinas, hasta el momento, esto no ha sido completado del todo. Los sistemas expertos, presentan cierta inteligencia (dentro de alguna área), a partir de una base de conocimientos y mediante un método de inferencia, pueden determinar la mejor (o una de las mejores) solución para un

problema dentro del área para la que fueron diseñados. La forma en que adquieren nuevo conocimiento es mediante la aplicación de un mecanismo deductivo a su base de conocimientos. Los agentes inteligentes, son una nueva forma de ver los sistemas, existen agentes que toman decisiones, agentes que responden interrogantes y otros más, cada uno debe de tener cierta inteligencia dada por información que conocen y procesan.

- Matemáticas: la aplicación de los mecanismos deductivos tuvo sus orígenes en las matemáticas. Los mecanismos deductivos son usados para probar teoremas en la teoría de conjuntos, en álgebra y en geometría.
- Sistemas: existen varios sistemas para la prueba automática de teoremas, aunque el más famoso es Otter. La mayoría de ellos utilizan resolución en alguna de sus variantes y sólo unos pocos se han extendido a su ejecución en varios procesadores.
- Prueba de circuitos digitales: algunos probadores automáticos de teoremas se usan para verificar el correcto funcionamiento de algunos procesadores.

El trabajo de nuestra tesis se enfocará en desarrollar un método para la prueba de teoremas basado en Redes de Petri. Además, podemos decir que un probador de teoremas es un método deductivo. Debido a que no le asignamos semántica a las expresiones lógicas que utilizamos, podemos decir que las aplicaciones de nuestro método son las mencionadas anteriormente para los mecanismo deductivos.

La mayoría de los probadores automáticos de teoremas se basan en resolución, aunque también tal como lo mencionamos anteriormente, existe el problema de que el principio de resolución no nos indica cuales expresiones considerar, ni el orden. Aquí veremos intuitivamente, que los T-invariantes de la Red de Petri nos delimita el conjunto de expresiones a considerar y que la conexión de los elementos de la Red de Petri, nos indican posibles caminos a seguir, para llegar a obtener un resultado.

La redacción de la tesis se encuentra distribuida de la siguiente manera.

En el capítulo I introducimos los conceptos de la lógica proposicional en los cuales se basa nuestro trabajo. Conceptos tales como: sintaxis de la lógica proposicional, semántica de los conectivos lógicos, tablas de verdad, formas normales y algunos otros. Luego, introducimos el principio de resolución, algunos resultados del mismo y ejemplos. Una mejora al principio de resolución, es hiper-resolución, la cual mencionamos al final. Para mayores referencias de estos conceptos consultar [Nerode and Shore, 1997] y [Fitting, 1995].

De manera similar en el capítulo II, introducimos los conceptos de las Redes de Petri, en los cuales también se basa nuestro trabajo. Definición, representación, marcado, transición habilitada, regla de evolución del marcado, T-invariantes, P-invariantes, son algunos de los conceptos más importantes que abordamos. Algunas modificaciones a algunos de los conceptos anteriores las presentamos al final de este mismo capítulo II. En [Silva, 1985],

[Desel and Esparza, 1995], [Martínez, 1984] y [Colom, 1989] podemos encontrar más detalles de los tópicos que exponemos en el capítulo II.

En la lógica proposicional, un procedimiento de demostración (algoritmo) sólido es el que sólo prueba tautologías, en tanto que es completo un procedimiento que prueba toda tautología. La solidez y completud de nuestro método, la presentamos y probamos en el capítulo III.

Un modelo para un conjunto de expresiones puede ser una situación en la cual falla ó se tiene una situación no deseable, en el sistema que se ha modelado. También puede ser que debido a que encontramos un modelo, podemos decir que no podemos deducir cierta expresión a partir de lo que conocía el sistema (del conjunto de expresiones ó premisas). Por lo anterior, los modelos son importantes y presentamos en el capítulo IV, un método para encontrarlos (cuando existen) basado en la matriz de incidencia de la Red de Petri.

En [Jeffrey et *al.*, 1996], [Lautenbach, 1985], [Peterka and Murata, 1989] y [Sinachopoulos, 1987] podemos encontrar algunos métodos semejantes al que exponemos en el capítulo III. Métodos que buscan una contradicción o modelos en su defecto, los podemos encontrar en [Bibel, 1993] y [Socher and Johann, 1996].

Ya que tenemos todos los elementos teóricos de nuestro método, el paso siguiente es implementarlo. En el apéndice A presentamos una guía de usuario para el editor de Redes de Petri que implementamos para incluir en él las características del método propuesto. Los algoritmos utilizados por el editor anterior los describimos en el apéndice B y C. Para el algoritmo que describimos en el apéndice C utilizamos como referencia los expuestos en [Colom, 1989], [Martínez and Silva, 1981] y [Treves, 1986].

I Lógica Proposicional

1. Introducción

En esta sección describiremos los elementos de la lógica proposicional, necesarios para el desarrollo de nuestro trabajo. Primeramente definiremos la sintaxis de la lógica proposicional. Suponemos que tenemos una colección de sentencias ó enunciados, de los que pensamos expresan proposiciones. Comenzamos con una colección de enunciados elementales o atómicos cuya estructura interna no nos ocuparemos de analizar. De hecho, los representaremos mediante simples letras, las letras proposicionales. Todo lo que nos importará de ellos es si son, o representan, proposiciones verdaderas o falsas, pero no ambas cosas. Luego formaremos enunciados más elaboradas usando los conectivos proposicionales. Nuestra principal preocupación será el ver cómo la verdad o falsedad de los enunciados compuestos depende de la verdad o falsedad de los enunciados atómicos que los forman. En particular, nos interesará saber cuáles enunciados compuestos deben ser verdaderos, independientemente de los valores que posean sus enunciados atómicos. Para la lógica proposicional tales enunciados son llamados tautologías. Gran parte de la lógica proposicional se dedica a tratar de determinar cuáles enunciados son tautologías.

Las tablas de verdad son un mecanismo usado para determinar cuáles enunciados son tautologías. Algunos métodos de demostración, como resolución y árboles semánticos, determinan si un enunciado compuesto es o no una tautología. Una mejora al método de resolución es hiper-resolución, en el cual se basa nuestro trabajo. En la lógica proposicional, un procedimiento de demostración (algoritmo) *sólido* es el que sólo prueba tautologías, en tanto que es *completo* un procedimiento que prueba toda tautología. Luego, para cada método de demostración, tendremos que establecer solidez y completud.

Si las tablas de verdad son un mecanismo de demostración, entonces ¿para qué necesitamos de otros mecanismos? Primero, las tablas de verdad siempre son ineficientes, en tanto que otros mecanismos pueden ser mucho mejores en un análisis por casos.

Una vez analizado todo lo anterior, estará completa la parte de la lógica proposicional necesaria para establecer los fundamentos de nuestro trabajo.

2. Sintaxis de la Lógica Proposicional

La sintaxis de cualquier lenguaje inicia con la descripción de su alfabeto. El alfabeto del lenguaje de la lógica proposicional consiste de:

i. Símbolos proposicionales: p_0, p_1, p_2, ...

ii. Conectivos lógicos: $\wedge, \vee, \rightarrow, \leftrightarrow, \neg$

iii. Constantes: T y F, verdad y falsedad, respectivamente.

iv. Símbolos auxiliares: ")" y "("

Una vez que hemos especificado el alfabeto de nuestro lenguaje, podemos describir los enunciados de la lógica proposicional. La definición siguiente termina de establecer la sintaxis de la lógica proposicional.

Definición 1.1: (*fórmula atómica, proposiciones*)

i. Una *fórmula atómica* es un símbolo proposicional, o cualquiera de las dos constantes T, F.

Denotamos el conjunto de las fórmulas atómicas por *At*.

ii. El *conjunto de las fórmulas proposicionales* o *proposiciones* PROP es el menor de los conjuntos X que satisfacen las condiciones siguientes:

a. Si φ es una fórmula atómica, entonces $\varphi \in X$.

b. Si $\varphi \in X$, entonces $\neg\varphi \in X$.

c. Si $\psi \in X$, entonces $(\varphi \square \psi) \in X$, donde $\square \in \{ \wedge, \vee, \rightarrow, \leftrightarrow \}$.

Ejemplo 1.1:

Sean p, q, r letras proposicionales.

Luego p, q, r también son fórmulas atómicas.

Son fórmulas proposicionales las siguientes:

$\neg p, \neg q, \neg r$.

$(p \wedge q), (q \wedge r), (p \wedge r)$

$(p \wedge q) \rightarrow r$, $(p \wedge q) \leftrightarrow (q \wedge r)$, etc.

La definición de una fórmula proposicional presupone la existencia de un conjunto "más pequeño" (único) que cumpla con las condiciones a, b y c, donde "más pequeño" significa: es un subconjunto de los demás. La existencia de tal conjunto se establece fácilmente. Primero, existen conjuntos que cumplen las condiciones a, b y c, por ejemplo, el conjunto universal consistente de todas las palabras formadas por los símbolos que estamos usando. Luego, es fácil verificar que la intersección de cualquier familia de conjuntos que cumplen con las condiciones a, b y c es otro conjunto que cumple tales condiciones. Ahora, sólo formemos la

intersección de la familia (no vacía) de todos los conjuntos que cumplen las condiciones a, b y c. Este será el conjunto "más pequeño" PROP.

El definir objetos o conjuntos inductivamente permite probar con facilidad propiedades que poseen esos objetos y definir funciones entre conjuntos definidos de esa manera. Esto es posible debido a dos principios básicos los cuales quedan establecidos en los dos teoremas siguientes.

Teorema 1.1: (*principio de inducción estructural*)

Toda fórmula proposicional tiene la propiedad **Q** si

(B) Toda fórmula atómica tiene la propiedad **Q**;

(I) Si φ tiene la propiedad **Q**, entonces $\neg\varphi$ tiene la propiedad **Q**.

Si φ y ψ tienen la propiedad **Q**, entonces $(\varphi \square \psi)$ tiene la propiedad **Q**, donde $\square \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$.

El otro principio básico que necesitamos nos permitirá definir funciones sobre la clase de fórmulas proposicionales.

Teorema 1.2: (*principio de recursión estructural*)

Hay una y sólo una función h definida sobre el conjunto de fórmulas proposicionales PROP tal que

(B) El valor de h está dado explícitamente sobre las fórmulas atómicas.

(I) El valor $h(\neg\varphi)$ está especificado en términos del valor $h(\varphi)$.

El valor $h((\varphi \square \psi))$ está especificado en términos de los valores $h(\varphi)$ y $h(\psi)$, donde $\square \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$.

Una manera de representar una fórmula es mediante un árbol de formación. Esta forma de representar una fórmula es viable para su posterior análisis sintáctico y semántico.

Definición 1.2: (*árbol de formación de una proposición*)

El *árbol de formación* (*parse tree*) de una proposición φ se define como sigue: para φ, $\psi \in$ PROP y $\square \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$

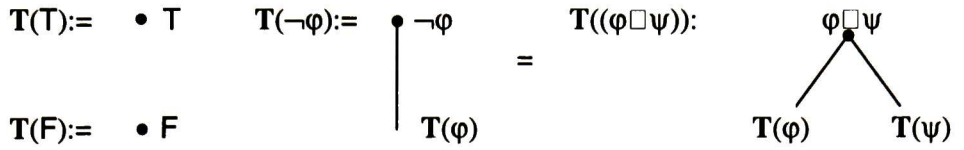


Figura 1.1 Arbol de formación asociado con la proposición que etiqueta su raíz.

El *árbol de formación* T representa o está asociado con la proposición que etiqueta su raíz.

El teorema siguiente garantiza el hecho de que sólo existe un árbol de formación asociado a cada proposición en PROP.

Teorema 1.3:

Cada proposición $\varphi \in$ PROP tiene un único árbol de formación T asociado con ella.

A continuación contemplamos algunas definiciones necesarias para algunas pruebas posteriores.

Definición 1.3: (*profundidad, soporte*)

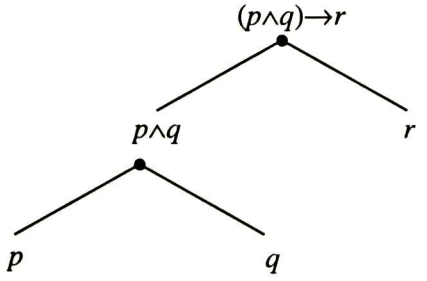
Sea $\varphi \in$ PROP y sea T el árbol de formación asociado a φ.

i. La *profundidad* de φ es la altura de T.

ii. El *soporte* de φ es el conjunto de letras proposicionales que aparecen como etiquetas de las hojas de T. También es el conjunto de fórmulas atómicas presentes en φ. Denotamos por $|\varphi|$ al soporte de φ.

Ejemplo 1.2:

Sea $\varphi = (p \wedge q) \rightarrow r$ una proposición. El árbol de formación T asociado a φ es el siguiente:



La altura de T es 2 por tanto la profundidad de φ es también 2.

Como las etiquetas de las hojas de T son p, q y r, entonces $|\varphi| = \{p, q, r\}$.

3. Semántica de los Conectivos Lógicos

La lógica proposicional es bivaluada, es decir, tomamos como el espacio de los valores de verdad al conjunto VAL:= {V, F}, donde V y F son simplemente dos objetos distintos, pero claro, tal como inferimos, V representa el valor verdadero y F el valor falso. Nuestra intención es asociar a cada fórmula proposicional un solo elemento de VAL. Para ello, primeramente debemos mencionar cómo interpretar sobre VAL cada una de las operaciones indicadas por los conectivos lógicos. Para la negación es fácil, de ahora en adelante asumimos que tenemos una función $\neg$:VAL$\rightarrow$VAL definida por $\neg(V) = F$ y $\neg(F) = V$.

Los conectivos binarios son un asunto bastante más complicado. De ahora en adelante asumiremos que para cada uno de los conectivos tenemos un símbolo de operación binaria correspondiente en el lenguaje de la lógica proposicional. Para hacer más fácil de recordar la relación entre símbolo y operación, vamos a sobrecargar nuestra notación. De ahora en adelante, por ejemplo, $\wedge$ será un símbolo de operación binaria de nuestro lenguaje formal y también denotará una operación sobre VAL, de acuerdo con la tabla mostrada a continuación. Este doble uso no deberá causarnos dificultad, ya que el contexto nos hará ver claro a cuál nos referimos.

Tabla 1.1

		$\vee$	$\wedge$	$\rightarrow$	$\leftrightarrow$
V	V	V	V	V	V
V	F	V	F	F	F
F	V	V	F	V	F
F	F	F	F	V	V

Si tenemos V$\rightarrow$F, de acuerdo a la tabla 1.1 el resultado sería F. Si consideramos V$\vee$F y basándonos también en la tabla anterior, el resultado sería V. Para cualesquiera que sean los valores de verdad que consideremos y la operación indicada por el conectivo binario correspondiente en una proposición, es fácil determinar su valor de verdad basándonos en la tabla 1.1.

3.1 Valuaciones, Tautología, Consecuencia y Modelo

Ahora conectaremos la sintaxis y la semántica. Estamos interesados en aquellas fórmulas proposicionales que por su estructura lógica deben ser verdaderas, sin tomar en cuenta los

significados que posean las fórmulas atómicas que las forman. Tales fórmulas son llamadas tautologías. Interpretaremos a los símbolos proposicionales como que nombran las correspondientes operaciones sobre VAL que hemos denotado en la tabla 1.1.

Definición 1.4: (*valuación booleana*)

Una *valuación booleana*, ó simplemente *valuación*, es una función ν del conjunto de las fórmulas proposicionales PROP al conjunto de valores de verdad VAL que satisface las condiciones siguientes:

i. $\nu(\mathsf{T}) = \mathsf{V}$

ii. $\nu(\mathsf{F}) = \mathsf{F}$

iii. $\nu(\neg\varphi) = \neg\nu(\varphi)$

iv. $\nu(\varphi \square \psi) = \nu(\varphi) \square \nu(\psi)$, para $\square \in \{ \wedge, \vee, \rightarrow, \leftrightarrow \}$

Cuando tenemos una fórmula proposicional y deseamos encontrar una valuación para dicha fórmula, siempre es posible hacerlo. Tal como indicamos en el siguiente lema.

Lema 1.1:

Para cualquier función f del conjunto de fórmulas atómicas *At* al conjunto de valores de verdad VAL, tal que $f(\mathsf{T}) = \mathsf{V}$ y $f(\mathsf{F}) = \mathsf{F}$, existe una valuación ν que concuerda con f sobre las fórmulas atómicas.

Mediante las tablas de verdad podemos encontrar todas las valuaciones posibles para cierta fórmula, aunque tal como mostramos en el siguiente lema, algunas de ellas la podemos omitir.

Lema 1.2:

Si ν_1 y ν_2 son dos valuaciones que concuerdan sobre un conjunto de fórmulas atómicas *S*, entonces ν_1 y ν_2 concuerdan sobre toda fórmula proposicional cuyo soporte esté contenido en *S*.

Ejemplo 1.3:

Sean *p*, *q* y *r* fórmulas atómicas.

Sea v una valuación tal que $v(p) = V$, $v(q) = V$ y $v(r) = F$ y a todas las demás fórmulas atómicas les asigna el valor F. Por el Lema 1.1 existe una valuación que satisface estas condiciones y por el Lema 1.2 sólo hay una.

Ahora bien:

$v((p\wedge q)\rightarrow r) = v(p\wedge q)\rightarrow v(r)$	Por iv de la definición 3.1, aplicado a $\rightarrow$
$= (v(p)\wedge v(q))\rightarrow v(r)$	También por iv de la misma definición aplicado a $\wedge$
$= (V\wedge V)\rightarrow F$	Unicamente sustituimos los valores de verdad asignados
$= V\rightarrow F$	Cambiamos $V\wedge V$ por V, resultado de la operación $\wedge$
$= F$	Finalmente, de la operación $\rightarrow$ obtenemos el valor F

Por lo tanto la proposición $(p\wedge q)\rightarrow r$ es falsa bajo v.

Estamos interesados en aquellas fórmulas proposicionales que no pueden evitar ser verdaderas.

Definición 1.5: (*tautología*)

Sea $\alpha\in$PROP.

Decimos que α es *válida* sii $v(\alpha) = V$, para cualquier valuación v. Si α es *válida* también decimos que es una *tautología*.

Ejemplo 1.4:

Sea $\varphi = p\rightarrow((\neg p\wedge q)\rightarrow r)$

En φ aparecen 3 símbolos proposicionales, p, q y r, a los cuales se les puede asignar cualquiera de los dos valores de verdad en VAL. El número total de valuaciones a considerar para φ es 8, tal como podemos observar en la tabla siguiente. A continuación trataremos de explicar la misma.

En la primera columna podemos apreciar las distintas valuaciones que actúan sobre p, q y r. En el primer renglón aparecen los símbolos proposicionales sobre los cuales operan cada una de las valuaciones y para facilitar el cálculo del valor de verdad de φ, la hemos particionado en dos fórmulas. En el segundo renglón de la tabla aparece v_1 que es la primera valuación que consideramos y tal como lo podemos apreciar, v_1 asigna V a las letras proposicionales p, q y r. Luego, de acuerdo a v_1 calculamos el valor de verdad de las dos fórmulas que son parte de φ. Al final mostramos que bajo v_1 la fórmula φ es V. La explicación

de los subsecuentes renglones es similar a que realizamos para el segundo renglón, por lo que la omitiremos.

	p	q	r	$\neg p \wedge q$	$(\neg p \wedge q) \rightarrow r$	$p \rightarrow ((\neg p \wedge q) \rightarrow r)$
v_1	V	V	V	F	V	V
v_2	V	V	F	F	V	V
v_3	V	F	V	F	V	V
v_4	V	F	F	F	V	V
v_5	F	V	V	V	V	V
v_6	F	V	F	V	F	V
v_7	F	F	V	F	V	V
v_8	F	F	F	F	V	V

Como observamos, sin importar el valor que se asigne a las letras proposicionales p, q y r, la proposición φ siempre es verdadera. Por tanto φ es una tautología.

Cuando una fórmula es satisfacible existe una valuación que le asigna el valor V, ahora bien, cuando toda fórmula en un conjunto es satisfacible, decimos que el conjunto es satisfacible. Formalizamos lo anterior a continuación.

Definición 1.6: (*fórmula satisfacible*)

Sea $\sigma \in$ PROP.

Decimos que la fórmula σ es satisfacible sii existe una valuación v tal que $v(\sigma) = V$, en caso contrario decimos que σ es insatisfacible.

Definición 1.7: (*conjunto de fórmulas satisfacible*)

Sea Σ un conjunto de fórmulas proposicionales.

Decimos que Σ es un conjunto de fórmulas satisfacible sii existe una valuación v que asocia a cada fórmula en Σ el valor V, de lo contrario decimos que Σ no es satisfacible o insatisfacible.

Ejemplo 1.5:

Del ejemplo 1.4 podemos decir que toda fórmula es satisfacible y por tanto el conjunto considerado también decimos que es satisfacible.

$p \wedge \neg p$ sería un ejemplo de una fórmula insatisfacible.

En la siguiente definición introducimos el concepto de consecuencia semántica.

Definición 1.8: (*consecuencia*)

Sean $\sigma \in$ PROP y Σ un conjunto de proposiciones (posiblemente infinito).

Decimos que σ es una *consecuencia* de Σ sii

$v(\sigma) = V$ siempre que $v(\tau) = V$, para toda $\tau \in \Sigma$ y toda valuación v.

Usamos el símbolo $\models$ para indicar una consecuencia.

Si σ es *consecuencia* de Σ, entonces escribimos $\Sigma \models \sigma$.

Si $\Sigma \models \sigma$ y Σ es el conjunto vacío, entonces únicamente escribimos $\models \sigma$. Resulta que esta última expresión es otra forma de afirmar que σ es una tautología.

Definición 1.9: (*modelo*)

Sea Σ un conjunto de fórmulas proposicionales.

Decimos que una valuación v es un *modelo* de Σ sii $v(\sigma) = V$, para toda $\sigma \in \Sigma$.

Denotamos por $M(\Sigma)$ el conjunto de todos los modelos de Σ.

Ejemplo 1.6:

Sean $p, q \in At$.

Sean $\Sigma = \{p, (p \rightarrow q)\}$ y $\sigma = q$.

Probemos que σ es consecuencia de Σ.

Supongamos que, por el contrario, σ no es consecuencia de Σ. Esto implica que existe una valuación v que hace verdadero cada elemento de Σ pero no hace verdadera a σ.

Como $p \in \Sigma$, la valuación v debe asignar V a p, es decir $v(p) = V$.

A su vez, $p \rightarrow q \in \Sigma$ y por tanto v debe hacer verdadera a $p \rightarrow q$. Pero la única forma de conseguirlo es que v asigne V a q, ya que de otra forma $p \rightarrow q$ sería falsa (sí v asignara F a q). Esto contradice la suposición inicial. Por lo tanto, no existe valuación que haga verdadero cada elemento de Σ y falso a σ.

Luego $\Sigma \models \sigma$.

Como otro ejemplo, consideremos el conjunto $\Sigma = \{p, (p \rightarrow q, r), (q, r \rightarrow s)\}$.

El conjunto de fórmulas proposicionales Σ tiene 5 modelos, tal como podemos observar en la siguiente tabla.

	p	q	r	s
v_1	V	V	V	V
v_2	V	V	F	V
v_3	V	V	F	F
v_4	V	F	V	V
v_5	V	F	V	F

Luego $M(\Sigma) = \{v_1, v_2, v_4, v_4, v_5\}$.

3.2 Teorema de Reemplazo

Un resultado análogo a la familiar propiedad de sustitución de la igualdad es aquel que enuncia que uno puede sustituir proposiciones por otras que sean "sus iguales". Este resultado es necesario para facilitar el análisis de las proposiciones.

Cuando escribamos $\varphi[p]$ querremos decir que φ es una fórmula proposicional y la presencia de la fórmula atómica p en φ juega un papel especial.

Por ejemplo, si $\varphi[p]$ es la fórmula proposicional $p \rightarrow (q \vee \neg p)$ y si ψ es $(p \rightarrow r)$, entonces $\varphi[\psi]$ es $(p \rightarrow r) \rightarrow (q \vee \neg(p \rightarrow r))$.

Notemos que, como en este ejemplo, la fórmula de sustitución ψ puede introducir presencias frescas de p. No requerimos que p esté presente en $\varphi[p]$ y esta es una cuestión importante.

Posteriormente, cuando escribamos $\varphi[\psi]$ para alguna fórmula proposicional ψ, nos estaremos refiriendo a la fórmula que resulta de φ cuando todas las presencias de p son sustituidas por presencias de la fórmula ψ.

Teorema 1.4: (*teorema de reemplazo*)

Sean $\varphi[p]$, ψ y τ fórmulas proposicionales.

Sea ν una valuación.

Sí $\nu(\psi) = \nu(\tau)$, entonces $\nu(\varphi[\psi]) = \nu(\varphi[\tau])$.

Ejemplo 1.7:

Sea $\varphi[s]$ la fórmula proposicional $(p \rightarrow q) \wedge (r \rightarrow s)$.

Como $(\neg p \vee q) \leftrightarrow (p \rightarrow q)$ es una tautología, tenemos, por el Teorema 3.1, que también lo es $\varphi[\neg p \vee q] \leftrightarrow \varphi[p \rightarrow q]$. Esto es,

$$\vdash ((p \rightarrow q) \wedge (r \rightarrow (\neg p \vee q))) \leftrightarrow ((p \rightarrow q) \wedge (r \rightarrow (p \rightarrow q))).$$

Hay una manera informal pero útil de describir esto: hemos reemplazado una presencia de $\neg p \vee q$ por su equivalente $p \rightarrow q$ en la fórmula $(p \rightarrow q) \wedge (r \rightarrow (\neg p \vee q))$, para convertirla en su equivalente $(p \rightarrow q) \wedge (r \rightarrow (p \rightarrow q))$.

3.3 Equivalencia Lógica

Un resultado importante para nuestro trabajo es el que establece cuando dos proposiciones son equivalentes. A continuación enunciaremos algunas definiciones y resultados al respecto.

Definición 1.10: (*equivalencia*)

Sean φ, ψ elementos de PROP.

La relación binaria $\equiv$ en PROP se define de tal manera que para cualesquiera φ y ψ:

$\varphi \equiv \psi$ sii la proposición $\varphi \leftrightarrow \psi$ es una tautología.

Cuando se tenga que $\varphi \equiv \psi$ diremos que φ y ψ son (*lógicamente*) *equivalentes*.

Lema 1.3:

Sean φ, φ' ψ, ψ' elementos de PROP.

Si $\varphi \equiv \varphi'$ y $\psi \equiv \psi'$, entonces $\neg\varphi \equiv \neg\varphi'$ y $(\varphi \quad \psi) \equiv (\varphi' \quad \psi')$, donde $\in \{ \wedge, \vee, \rightarrow, \leftrightarrow \}$.

Ahora introducimos algunas leyes de equivalencia que son de gran utilidad para obtener las llamadas formas normales de una fórmula proposicional.

Lema 1.4:

Sean φ, ψ, σ elementos de PROP.

Las equivalencias siguientes son válidas:

i. **Leyes Asociativas:** $((\varphi \vee \psi) \vee \sigma) \equiv (\varphi \vee (\psi \vee \sigma))$

$((\varphi \wedge \psi) \wedge \sigma) \equiv (\varphi \wedge (\psi \wedge \sigma))$

ii. **Leyes Conmutativas:** $(\varphi \vee \psi) \equiv (\psi \vee \varphi)$

$(\varphi \wedge \psi) \equiv (\psi \wedge \varphi)$

iii. **Leyes Distributivas:** $(\varphi \vee (\psi \wedge \sigma)) \equiv ((\varphi \vee \psi) \wedge (\varphi \vee \sigma))$

$(\varphi \wedge (\psi \vee \sigma)) \equiv ((\varphi \wedge \psi) \vee (\varphi \wedge \sigma))$

iv. **Leyes De Morgan:** $\neg(\varphi \vee \psi) \equiv (\neg\varphi \wedge \neg\psi)$

$\neg(\varphi \wedge \psi) \equiv (\neg\varphi \vee \neg\psi)$

v. **Leyes de Idempotencia:** $(\varphi \vee \varphi) \equiv \varphi$

$(\varphi \wedge \varphi) \equiv \varphi$

vi. **Ley de Doble Negación:** $\neg\neg\varphi \equiv \varphi$

vii. **Leyes de Absorción:** $\varphi \vee (\varphi \wedge \psi) \equiv \varphi$

$\varphi \wedge (\varphi \vee \psi) \equiv \varphi$

viii. **Leyes del Cero y el Uno:** $(\varphi \vee \beta) \equiv \varphi$ $\quad (\varphi \wedge \beta) \equiv \beta$

$(\varphi \vee \alpha) \equiv \alpha$ $\quad (\varphi \wedge \alpha) \equiv \varphi$

$(\varphi \vee \neg\varphi) \equiv \alpha$ $\quad (\varphi \wedge \neg\varphi) \equiv \beta$

3.4 Formas Normales

Es preferible estandarizar una fórmula, debido a que de esta manera suele ser más fácil establecer si es una tautología o si al menos es satisfacible. De hecho la mayoría de los métodos de demostración automática de teoremas convierten las fórmulas en algún tipo de forma normal antes de realizar una prueba.

El comportamiento de todos los conectivos puede ser descrito por algún subconjunto propio de los mismos. Este el caso de la conjunción y la disyunción, por lo que sólo trabajaremos con estas. Así que comenzamos por introducir la notación para conjunción generalizada y disyunción generalizada.

Definición 1.11: (*conjunción y disyunción generalizada*)

Sea $\varphi_1, \varphi_2, \ldots, \varphi_n$, una lista de fórmulas proposicionales (posiblemente vacía).

Definimos dos nuevos tipos de fórmulas proposicionales de la siguiente forma:

$\{\varphi_1, \varphi_2, \ldots, \varphi_n\}$ es la *conjunción (generalizada)* de $\varphi_1, \varphi_2, \ldots, \varphi_n$.

$<\varphi_1, \varphi_2, \ldots, \varphi_n>$ es la *disyunción (generalizada)* de $\varphi_1, \varphi_2, \ldots, \varphi_n$.

Si v es una valuación, entonces requerimos que

$v(<\varphi_1, \varphi_2, \ldots, \varphi_n>) = V$ sii v asocia a algún miembro de la lista $\varphi_1, \varphi_2, \ldots, \varphi_n$ el valor V. De lo contrario $v(<\varphi_1, \varphi_2, \ldots, \varphi_n>) = F$.

$v(\{\varphi_1, \varphi_2, \ldots, \varphi_n\}) = V$ sii v asocia a cada miembro de la lista $\varphi_1, \varphi_2, \ldots, \varphi_n$ el valor V. De otra forma $v(\{\varphi_1, \varphi_2, \ldots, \varphi_n\}) = F$.

El principio de resolución se aplica únicamente a fórmulas proposicionales que están en forma de cláusulas, por lo cual introducimos a continuación los conceptos pertinentes.

Definición 1.12: (*literal, literal positiva, literal negativa*)

Una *literal* es una fórmula atómica o la negación de una fórmula atómica.

Si la literal l es una fórmula atómica, decimos que l es una *literal positiva*; pero si l es la negación de una fórmula atómica, entonces decimos que l es una *literal negativa.*

Definición 1.13: (*cláusula, cláusula vacía, cláusula de Horn, cláusula de programa*)

Sea $\varphi_1, \varphi_2, \ldots, \varphi_n$ una lista de literales.

Una *cláusula C* es una disyunción $<\varphi_1, \varphi_2, \ldots, \varphi_n>$. Consecuentemente, la cláusula C es verdadera si al menos uno de sus elementos es verdadero.

Si la lista de literales es vacía, entonces decimos que C es una cláusula vacía. La *cláusula vacía* $<>$ siempre es falsa ya que no tiene elementos verdaderos.

Una cláusula que contiene a lo más una literal positiva es llamada *cláusula de Horn.*

Una cláusula que contiene exactamente una literal positiva es llamada *cláusula de programa.*

Ejemplo 1.8:

Sean $p, \neg q, r, \neg s, t$ literales.

La disyunción $< p, \neg q, r, \neg s, t >$ es una cláusula.

También $< p, \neg q >, < p, r >, <\neg q, \neg s >$ son cláusulas.

Definición 1.14: (*notación de una cláusula*)

Sea $C = < \alpha_1, \alpha_2, \ldots, \alpha_m, \neg\beta_1, \neg\beta_2, \ldots, \neg\beta_n >$ una cláusula, donde $m, n \geq 0$, y sea $|C| = \{\alpha_1, \alpha_2, \ldots, \alpha_m, \beta_1, \beta_2, \ldots, \beta_n\}$ el soporte de C.

Preferimos escribir la cláusula C en la forma ordenada $<\beta \rightarrow \alpha>$, donde α es la lista formada por toda $\alpha_i \in |C|$ para $i = 0, 1, \ldots, m$, es decir, la lista $\alpha_1, \alpha_2, \ldots, \alpha_m$ y β es la lista formada por toda $\beta_j \in |C|$ para $j = 0, 1, \ldots, n$, es decir, la lista $\beta_1, \beta_2, \ldots, \beta_n$. Siempre que no

exista confusión, omitiremos los paréntesis angulares ("<" y ">") al escribir una cláusula. Note además, que el símbolo "→" no indica implicación.

Ejemplo 1.9:

Reescribimos las cláusulas del ejemplo 1.8 de acuerdo a la definición anterior.

$< p, \neg q, r, \neg s, t >$ la reescribimos como $< q, s \rightarrow p, r, t >$.

$< p, \neg q >$ la reescribimos como $<q \rightarrow p>$

$< p, r >$ la reescribimos como $<\leftrightarrow p, r>$

$<\neg q, \neg s >$ la reescribimos como $<q, s \rightarrow>$

Veamos algunas definiciones más.

Definición 1.15: (*forma normal conjuntiva, forma de cláusula ó conjunto de cláusulas*)

Sea $C_1, C_2, ..., C_n$ una lista de cláusulas.

Una fórmula proposicional φ está en *forma normal conjuntiva* (o *forma de cláusula*, o es un *conjunto de cláusulas*) si es una conjunción $\{C_1, C_2, ..., C_n\}$. Por ello φ es verdadera si cada uno de sus elementos es verdadero.

La *fórmula vacía*, que denotamos por { }, es aquella que no contiene cláusulas. La fórmula vacía es siempre verdadera ya que no tiene elementos falsos.

Dentro de una fórmula siempre encerraremos entre paréntesis angulares cada una de sus cláusulas.

Ejemplo 1.10:

De las cláusulas del ejemplo anterior podemos llegar a formar el conjunto de cláusulas:

$\{< q, s \rightarrow p, r, t >, <q \rightarrow p>, <\leftrightarrow p, r>, <q, s \rightarrow>\}$

En la cual, tal como lo establece la definición anterior, cada elemento es una cláusula.

Teorema 1.5: (*forma normal*)

Toda fórmula proposicional se puede convertir a una forma de cláusula. Es decir, para cada fórmula proposicional φ existe una fórmula proposicional φ' en forma de cláusula, tal que $\varphi \equiv \varphi'$

De hecho existen algoritmos para convertir una fórmula proposicional ordinaria a forma de cláusula. La exposición completa de un algoritmo de este tipo la puede encontrar en [Fitting, 1995].

Ejemplo 1.11:

Sea $\varphi = (p \rightarrow \neg q) \wedge (r \vee (\neg p \rightarrow q))$.

Una forma normal conjuntiva equivalente a φ es:

$\{<p, q \rightarrow>, <\rightarrow p, q, r>\}$

3.5 Asignaciones

Cuando estuvimos hablando de fórmulas, introdujimos lo que es una valuación. Ahora al estar hablando de conjuntos de cláusulas, introducimos el concepto de asignación.

Definición 1.16: (*asignación*)

Una *asignación* $\mathbf{A}$ es un conjunto de literales positivas, la cual asigna V a toda $l \in \mathbf{A}$ y asigna F a toda $l \notin \mathbf{A}$.

Para terminar con esta sección y continuar en la siguiente con resolución, introducimos la definición de cuando una asignación satisface una cláusula, cuando satisface un conjunto de cláusulas, cuando una cláusula es satisfacible y cuando un conjunto de cláusulas es satisfacible. Pero antes de ello, especificamos algunas notaciones para las operaciones de unión e intersección, aunque no las utilicemos todas ellas en este momento.

Definición 1.17:

Sean C, C_1 y C_2 cláusulas.

Sea $\square \in \{\cup, \cap, \subset, \subseteq, \not\subset\}$ tal que $\cup$ denota unión, $\cap$ denota intersección, $\subset$ denota subconjunto propio, $\subseteq$ denota subconjunto y $\not\subset$ denota no pertenencia.

i. $C \square \mathbf{A}$ denota la operación indicada entre el conjunto formado por los átomos en $|C|$ y el conjunto formado por los átomos que aparecen en la asignación $\mathbf{A}$.

ii. $C_1 \square C_2$ denota la operación indicada entre el conjunto formado por los átomos en $|C_1|$ y el conjunto formado por los átomos en $|C_2|$.

Definición 1.18:

Sean C una cláusula, S un conjunto de cláusulas y $\mathbf{A}$ una asignación.

i. Decimos que $\mathbf{A}$ *satisface la cláusula* C, y escribimos $\mathbf{A} \models C$, sii $C \cap \mathbf{A} \neq \varnothing$; es decir, $\mathbf{A}$ asigna el valor V cuando menos a una literal en C.

ii. Si existe $\mathbf{A}$ que satisfaga a C, decimos que C es *satisfacible*, en caso contrario decimos que C es *insatisfacible*. La cláusula vacía $< >$ siempre es insatisfacible.

iii. Decimos que *la asignación* $\mathbf{A}$ *satisface a* S, y escribimos $\mathbf{A} \models S$, sii $\mathbf{A}$ satisface toda cláusula en S. Decimos que *la valuación inducida por* $\mathbf{A}$ *es un modelo para* S, si $\mathbf{A}$ satisface a S.

iv. Si existe $\mathbf{A}$ que satisfaga a S, decimos que S *es satisfacible*, en caso contrario decimos que S *es insatisfacible*.

Ejemplo 1.12:

Sean $C_1 = <\leftrightarrow p>$, $C_2 = <p \rightarrow q>$ y $C_3 = <q \rightarrow r>$ cláusulas.

Sean $\mathbf{A_1} = \{p, q\}$ y $\mathbf{A_2} = \{p, q, r\}$ asignaciones.

La asignación $\mathbf{A_1}$ satisface a las cláusulas C_1 y C_2, ya que $C_1 \cap \mathbf{A_1} \neq \varnothing$ y $C_2 \cap \mathbf{A_1} \neq \varnothing$. Pero $\mathbf{A_1}$ no satisface a C_3, debido a que $C_3 \cap \mathbf{A_1} = \varnothing$. Recordando que la cláusula C_2 viene de $<\neg p, q>$. Por otro lado, la asignación $\mathbf{A_2}$ satisface a C_1, C_2 y a C_3.

Sean $S_1 = \{<\leftrightarrow p>, <p \rightarrow q>\}$, $S_2 = \{<p \rightarrow q>, <q \rightarrow r>\}$ y $S_3 = \{<\leftrightarrow p>, <p \rightarrow\}$ conjuntos de cláusulas.

La asignación $\mathbf{A_1}$ satisface a S_1 pero no a S_2. Por otro lado, la asignación $\mathbf{A_2}$ satisface tanto a S_1 como a S_2.

Para que una asignación satisfaga el conjunto S_3, debe contener tanto a la literal p como a su negación, pero no puede haber tal asignación, por la definición 1.16. Consecuentemente el conjunto de cláusulas S_3 es insatisfacible.

4 Resolución

El método de prueba subyacente a PROLOG y a la mayoría de los probadores de teoremas es un sistema de axiomas y reglas particularmente simple llamado resolución. Resolución sólo tiene una regla y es un procedimiento de refutación, es decir, trata de mostrar que la fórmula dada es insatisfacible. Al comenzar asume que la fórmula de interés está en forma normal conjuntiva.

Definición 1.19: (*regla de resolución*)

Sean $C_1 = \langle \varphi \rightarrow \psi, l \rangle$ y $C_2 = \langle \varphi', l \rightarrow \psi' \rangle$ cláusulas.

En nuestra terminología actual, decimos que, de C_1 y C_2 inferimos $C = \langle \varphi, \varphi' \rightarrow \psi, \psi' \rangle$. Otra forma de visualizar el proceso anterior sería:

$$\frac{\langle \varphi \rightarrow \psi, l \rangle \qquad \langle \varphi', l \rightarrow \psi' \rangle}{\langle \varphi, \varphi' \rightarrow \psi, \psi' \rangle}$$

Denotamos por ***R*** al sistema de inferencia para refutación de cláusulas, el cual comprende sólo la regla anterior. La cláusula C es llamada resolvente de C_1 y C_2. También decimos que resolvimos sobre l.

Ejemplo 1.13:

Sea $S = \{C_1, C_2, C_3, C_4\}$, donde

$C_1 = \langle \rightarrow p \rangle$, $C_2 = \langle p \rightarrow q \rangle$, $C_3 = \langle q \rightarrow r \rangle$ y $C_4 = \langle r \rightarrow \rangle$

Apliquemos la regla de resolución a algunas cláusulas de S:

De $\langle \rightarrow p \rangle \qquad \langle p \rightarrow q \rangle$

Obtenemos $\dfrac{}{\langle \rightarrow q \rangle}$

De $\langle p \rightarrow q \rangle \qquad \langle q \rightarrow r \rangle$

Obtenemos $\dfrac{}{\langle p \rightarrow r \rangle}$

De $\langle q \rightarrow r \rangle \qquad \langle r \rightarrow \rangle$

Obtenemos $\dfrac{}{\langle q \rightarrow \rangle}$

Enseguida introducimos algunas definiciones necesarias para demostrar los teoremas de solidez y completud del sistema de resolución. De aquí en delante S denota un conjunto de cláusulas, a menos que se defina de otra forma.

Definición 1.20: ($S \Rightarrow_R S'$)

La expresión $S \Rightarrow_R S'$ significa que S' se obtiene de S por la *aplicación* de la regla de resolución.

Ejemplo 1.14:

Tomemos el conjunto de cláusulas S del ejemplo 1.13. Tenemos que:

$\{<\rightarrow p>,<p \rightarrow q>,<q\rightarrow r>,<r\rightarrow>\} \Rightarrow_R \{<\rightarrow p>,<p\rightarrow q>, <q\rightarrow r>,<r\rightarrow>,<\rightarrow q>\}$,

donde $<\rightarrow q>$ es la resolvente sobre p de C_1 y C_2.

Definición 1.21: (*derivación por resolución*)

Sea S_0 un conjunto de cláusulas.

Una *derivación por resolución de* S_0 es una secuencia $S_0, S_1, \ldots$ de conjuntos de cláusulas, donde para cada número natural n se cumple $S_n \Rightarrow_R S_{n+1}$.

Definición 1.22: (*refutación de un conjunto de cláusulas*)

Una *refutación* de S es una derivación por resolución de S con $<>\in S_n$, para algún $n\geq 0$.

Definición 1.23:

Sean $C = <\varphi\rightarrow\psi>$ y $D = <\varphi'\rightarrow\psi'>$ cláusulas distintas.

Se dice que *la cláusula D es absorbida por la cláusula C* si $\varphi \subseteq \varphi'$ y $\psi \subseteq \psi'$.

Definición 1.24: (*cláusula redundante con respecto a S*)

Una cláusula C *es redundante* (con respecto a S) siempre que $C\in S$, C es una tautología, o C es absorbida por alguna cláusula D en S.

Ejemplo 1.15:

Sea $S = \{C_1 = <p, q\rightarrow r>,\ C_2 = <p\rightarrow>,\ C_3 = <\rightarrow r>,\ C_4 = <p, q\rightarrow s>\}$

Las cláusulas C_1 y C_4 son absorbidas por la cláusula C_2. También, la cláusula C_1 es absorbida por la cláusula C_3.

Por lo que las cláusulas C_1 y C_4 son redundantes con respecto a S.

La cláusula $C_5 = <\rightarrow r>$ también es redundante con respecto a S.

El lema siguiente afirma que una resolvente es consecuencia de las cláusulas de las que se infiere.

Lema 1.5:

Si R es una resolvente de cláusulas C y D, entonces $\{C, D\} \models R$.

Cuando tenemos un conjunto de cláusulas satisfacible y le aplicamos la regla de resolución para obtener un nuevo conjunto de cláusulas, este también es satisfacible. Esto queda establecido en el lema siguiente.

Lema 1.6:

Sean S y S' conjuntos de cláusulas tales que $S \Rightarrow_R S'$ Luego S' es satisfacible sii S lo es. Equivalentemente, S es insatisfacible sii S' lo es.

Ejemplo 1.16:

Consideremos el conjunto de cláusulas S del ejemplo 1.13

$S_0 = S = \{<\neg p>,<p \rightarrow q>,<q \rightarrow r>,<r \rightarrow >\} \Rightarrow_R S_1 = \{<\neg p>,<p \rightarrow q>,<q \rightarrow r>,<r \rightarrow >,<\neg q>\}$

$\Rightarrow_R S_2 = \{<\neg p>,<p \rightarrow q>,<q \rightarrow r>,<r \rightarrow >,<\neg q>,<p \rightarrow r>\}$

$\Rightarrow_R S_3 = \{<\neg p>,<p \rightarrow q>,<q \rightarrow r>,<r \rightarrow >,<\neg q>,<p \rightarrow r>,<q \rightarrow >\}$

$\Rightarrow_R S_4 = \{<\neg p>,<p \rightarrow q>,<q \rightarrow r>,<r \rightarrow >,<\neg q>,<p \rightarrow r>,<q \rightarrow >, < >\}$

La anterior es una refutación de S, ya que $< > \in S_4$.

Como S_4 es insatisfacible, S_3 también lo es. Como se encontró que S_3 es insatisfacible, también lo es S_2. Luego, también son insatisfacibles S_1 y S_0 y, consecuentemente S es insatisfacible (de acuerdo al lema 1.6).

El primer resultado importante que introducimos acerca del sistema de resolución es el de su solidez. Este resultado establece que si la cláusula vacía aparece en algún conjunto de cláusulas que se obtiene en una derivación por resolución, entonces el conjunto de cláusulas considerado inicialmente es insatisfacible.

Teorema 1.6: (*solidez de resolución*)

Si $S_0, S_1, \ldots S_n$ es una derivación por resolución con $< > \in S_n$, entonces S_0 es insatisfacible.

Prueba:

Ya que $< >$ y por tanto S_n es insatisfacible, repetidas aplicaciones del lema anterior garantizan que cada uno de los conjuntos de cláusulas S_i sean también insatisfacible.

Para llegar a establecer la completud del principio de resolución introducimos la siguiente definición.

Definición 1.25: (*conjunto de cláusulas cerrado bajo* ***R***)

Decimos que un conjunto de cláusulas S es *cerrado bajo* ***R*** sii cada resolvente de cláusulas en S es redundante con respecto a S.

Ejemplo 1.17:

Del ejemplo 1.16 podemos decir que S_0 no es cerrado bajo ***R*** ya que la resolvente de C_1 y C_2, que es $<\leftrightarrow q>$ no es redundante con respecto a S_0. La cláusula $<q\rightarrow>$ que es la resolvente de C_3 y C_4, no es redundante con respecto a S_1, ni redundante con respecto a S_2, por ello, S_1 y S_2 no son cerrados bajo ***R***. El conjunto S_3 no es cerrado bajo ***R***, debido a que la cláusula $<>$ no es redundante con respecto a S_3. Por último, el conjunto S_4 tampoco es cerrado bajo ***R***, la cláusula $<p\rightarrow>$ no es redundante con respecto a S_4.

El conjunto $S_6 = \{<\leftrightarrow p>,<p\rightarrow q>,<q\rightarrow r>,<r\rightarrow>,<\leftrightarrow q>,<p\rightarrow r>,<q\rightarrow>,<>,<p\rightarrow>,<\leftrightarrow r>\}$ es cerrado bajo ***R***, ya que cada resolvente de cláusulas en S_6 es redundante con respecto a S_6.

Si tenemos un conjunto cerrado bajo ***R*** y la cláusula vacía no pertenece a ese conjunto, entonces tal conjunto es satisfacible. Lo anterior queda establecido en el lema siguiente. Luego introducimos una definición que posteriormente se utilizará como argumento para la completud de resolución.

Lema 1.7: (*conjunto de cláusulas satisfacible*)

Si S es cerrado bajo ***R*** y si $<> \notin S$, entonces S es satisfacible.

Definición 1.26: (*derivación por resolución justa*)

Una *derivación* por resolución $S_0, S_1,\ldots$ se dice *justa* si el conjunto $S^{\infty} = \bigcup_{k\geq 0}\bigcap_{j\geq k} S_j$ es cerrado bajo ***R***.

Ejemplo 1.18:

Si continuamos con la derivación por resolución del ejemplo 1.16, obtenemos los siguientes conjuntos.

$S_0 = S \Rightarrow_R S_1 \Rightarrow_R S_2 \Rightarrow_R S_3 \Rightarrow_R S_4 = \{<\leftrightarrow p>,<p\rightarrow q>,<q\rightarrow r>,<r\rightarrow>,<\leftrightarrow q>,<p\rightarrow r>,<q\rightarrow>,<>\}$

$\Rightarrow_R S_5 = \{<\leftrightarrow p>,<p\rightarrow q>,<q\rightarrow r>,<r\rightarrow>,<\leftrightarrow q>,<p\rightarrow r>,<q\rightarrow>,<>,<\leftrightarrow r>\}$

$\Rightarrow_R S_6 = \{<\leftrightarrow p>,<p\rightarrow q>,<q\rightarrow r>,<r\rightarrow>,<\leftrightarrow q>,<p\rightarrow r>,<q\rightarrow>,<>,<\leftrightarrow r>,<p\rightarrow>\}$

La derivación $S_0 = S \Rightarrow_R S_1 \Rightarrow_R S_2 \Rightarrow_R S_3 \Rightarrow_R S_4 \Rightarrow_R S_5 \Rightarrow_R S_5$ es justa ya que el conjunto $S^{\infty} = \{<\rightarrow p>, <p\rightarrow q>, <q\rightarrow r>, <r\rightarrow>, <\rightarrow q>, <p\rightarrow r>, <q\rightarrow>, <>, <\rightarrow r>, <p\rightarrow>\}$ es cerrado bajo R.

Si un conjunto de cláusulas es insatisfacible, entonces en una derivación justa por resolución aparecerá en alguno de sus elementos la cláusula vacía.

Teorema 1.7: (*completud de resolución*)

Si S es insatisfacible y si $S_0, S_1, S_2, \ldots$ es una derivación justa por resolución con $S_0 = S$, entonces existe una $n \geq 0$, tal que S_n contiene a la cláusula vacía $<>$.

Podemos ilustrar una derivación por resolución por medio de un árbol de prueba por resolución, en el cual podemos apreciar claramente las cláusulas que tomamos para obtener cada resolvente.

Definición 1.27: (*árbol de prueba por resolución*)

Sean C, C_2 cláusulas y S un conjunto de cláusulas con $C_0, C_1 \in S$.

Un *árbol de prueba por resolución* de S a partir de C es un árbol binario y etiquetado T con las siguientes propiedades:

i. La raíz de T está etiquetada con C.

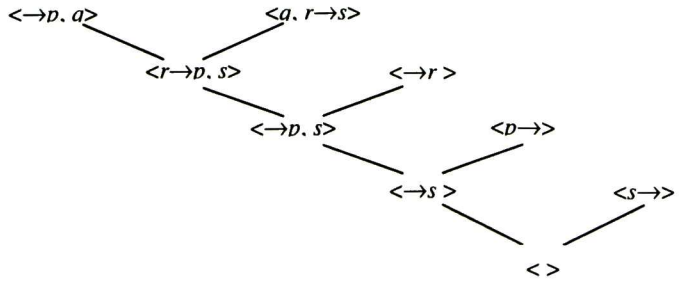
ii. Las hojas de T están etiquetadas con elementos de S.

iii. Si un nodo tiene etiqueta C_2 y sucesores inmediatos con etiqueta C_0, C_1, entonces C_2 es la resolvente de C_0 y de C_1.

Ejemplo 1.19:

Sea $S = \{<\rightarrow p, q>, <\rightarrow r>, <q, r\rightarrow s>, <p\rightarrow>, <s\rightarrow>\}$

Un árbol de prueba por resolución a partir de $<>$ es el siguiente:



En el cual observamos que su raíz está etiquetada con la cláusula vacía. El nodo con etiqueta $<\rightarrow s>$ es la resolvente de $<\rightarrow p, s>$ y $<p\rightarrow>$, ya que son sus sucesores inmediatos.

Si de un conjunto de cláusulas podemos obtener como resolvente una cláusula C, entonces existe un árbol de prueba por resolución a partir de dicha cláusula.

Lema 1.8:

Decimos que S tiene un árbol de prueba por resolución a partir de C sii existe una derivación por resolución de C a partir de S.

5 Hiper-Resolución

El método de demostración conocido como Hiper-resolución fue desarrollado por J. A. Robinson en 1965, y está basado en la idea de combinar varios pasos de resolución en un solo paso grande de resolución.

Ejemplo 1.20:

Sea $S = \{C_1, C_2, C_3\}$ con $C_1 = <s \rightarrow p, q>$, $C_2 = <p, r \rightarrow>$ y $C_3 = <q \rightarrow>$.

Las derivaciones por resolución mostradas en la figura 1.2 ambas llegan a la cláusula $<s, r\rightarrow>$. Un paso de hiper-resolución combinaría los dos pasos individuales de resolución de la figura 1.2 en uno solo y simplificaría por tanto, los dos pasos realizados mediante resolución. Esto lleva a la representación de la figura 1.3 del paso de hiper-resolución correspondiente a las derivaciones de la figura 1.2.

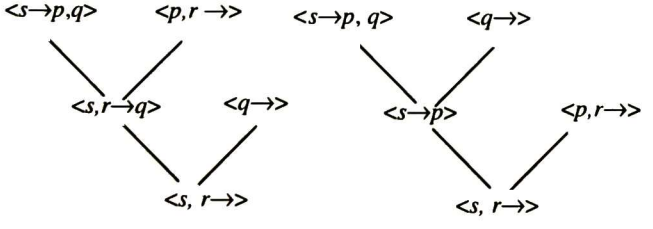


Figura 1.2 Arboles de prueba por Resolución

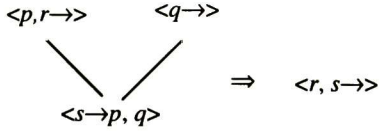


Figura 1.3 Arbol de prueba por Hiper-resolución

Cuando Robinson desarrolló la estrategia de hiper-resolución claramente tenía en mente el modelo del átomo de Bohr. En la anterior derivación por hiper-resolución la cláusula central $<s\rightarrow p, q>$ es llamada el *núcleo*, y las cláusulas satelitales $<p, r\rightarrow>$ y $<q\rightarrow>$ son llamadas los *electrones* del paso de hiper-resolución.

Definición 1.28: (*reglas de hiper-resolución negativa y de hiper-resolución positiva*)

1. Sea $n \geq 1$. La *regla de hiper-resolución negativa* para n cláusulas esta dada por:

$$\frac{\langle \varphi \rightarrow l_1, l_2, \ldots, l_n \rangle \quad \langle \varphi_1, l_1 \rightarrow \rangle \quad \langle \varphi_2, l_2 \rightarrow \rangle \quad \langle \varphi_n, l_n \rightarrow \rangle}{\langle \varphi, \varphi_1, \varphi_2, \varphi_n \rightarrow \rangle}$$

$\langle \varphi \rightarrow l_1, l_2, \ldots, l_n \rangle$ es *núcleo* y $\langle \varphi_1, l_1 \rightarrow \rangle$, $\langle \varphi_2, l_2 \rightarrow \rangle$ y $\langle \varphi_n, l_n \rightarrow \rangle$ son los *electrones*, del paso de hiper-resolución negativa.

2. Sea $n \geq 1$. La *regla de hiper-resolución positiva* para n cláusulas esta dada por:

$$\frac{\langle l_1, l_2, \ldots, l_n \rightarrow \psi \rangle \quad \langle \rightarrow \psi_1, l_1 \rangle \quad \langle \rightarrow \psi_2, l_2 \rangle \quad \langle \rightarrow \psi_n, l_n \rangle}{\langle \rightarrow \psi, \psi_1, \psi_2, \psi_n \rangle}$$

$\langle l_1, l_2, \ldots, l_n \rightarrow \psi \rangle$ es el *núcleo* y $\langle \rightarrow \psi_1, l_1 \rangle$, $\langle \rightarrow \psi_2, l_2 \rangle$ y $\langle \rightarrow \psi_n, l_n \rangle$ son los *electrones*, del paso de hiper-resolución positiva.

Denotamos por ***H*** al sistema de inferencia para refutación de cláusulas, el cual comprende las reglas anteriores. A la cláusula resultante al aplicar la regla de hiper-resolución le llamamos *hiper-resolvente* positiva ó negativa, según sea el caso.

Ejemplo 1.21:

Sea $S = \{ C_1, C_2, C_3, C_4, C_1', C_2', C_3', C_4' \}$, donde

$C_1 = \langle w \rightarrow p, q, r \rangle$, $C_2 = \langle s, p \rightarrow \rangle$, $C_3 = \langle t, q \rightarrow \rangle$, $C_4 = \langle u, r \rightarrow \rangle$

$C_1' = \langle p, q, r \rightarrow w \rangle$, $C_2' = \langle \rightarrow s, p \rangle$, $C_3' = \langle \rightarrow t, q \rangle$, $C_4' = \langle \rightarrow u, r \rangle$

El siguiente es un paso de hiper-resolución negativa:

$$\frac{\langle w \rightarrow p, q, r \rangle \quad \langle s, p \rightarrow \rangle \quad \langle t, q \rightarrow \rangle \quad \langle u, r \rightarrow \rangle}{\langle w, s, t, u \rightarrow \rangle}$$

El siguiente es un paso de hiper-resolución positiva:

$$\frac{\langle p, q, r \rightarrow w \rangle \quad \langle \rightarrow s, p \rangle \quad \langle \rightarrow t, q \rangle \quad \langle \rightarrow u, r \rangle}{\langle \rightarrow s, t, u, w \rangle}$$

Enseguida, introducimos algunas definiciones necesarias para demostrar los teoremas de solidez y completud del sistema de hiper-resolución.

Definición 1.29: ($S \Rightarrow_H S'$)

La expresión $S \Rightarrow_H S'$ significa que S' se obtiene de S por la aplicación de la regla de Hiper-resolución.

Definición 1.30: (*derivación por hiper-resolución de un conjunto de cláusulas*)

Una *derivación por hiper-resolución* de S_0 es una secuencia $S_0, S_1, \ldots$, de conjuntos de cláusulas donde para cada número natural n, $S_n \Rightarrow_H S_{n+1}$.

Definición 1.31: (*refutación de un conjunto de cláusulas*)

Una *refutación* de S es una derivación por hiper-resolución de S con $< > \in S_n$, para algún $n \geq 0$.

Ejemplo 1.22:

Sea $S = \{C_1, C_2, C_3, C_4\}$ donde:

$C_1 = <\rightarrow p>$, $C_2 = <\rightarrow q, r, s>$, $C_3 = <p, q \rightarrow>$ y $C_4 = <r \rightarrow>$.

$S = S_0 = \{<\rightarrow p>, <\rightarrow q, r, s>, <p, q \rightarrow>, <r \rightarrow>\}$

Al aplicar la regla de hiper-resolución a C_1, C_2 y C_3 tenemos:

$\Rightarrow_H S_1 = \{<\rightarrow p>, <\rightarrow q, r, s>, <p, q \rightarrow>, <r \rightarrow>, <\rightarrow r>\}$

En S_1 tenemos las cláusulas $<r \rightarrow>$ y $<\rightarrow r>$, al aplicarles la regla de hiper-resolución llegamos a:

$\Rightarrow_H S_2 = \{<\rightarrow p>, <\rightarrow q, r, s>, <p, q \rightarrow>, <r \rightarrow>, <\rightarrow r>, < >\}$

La anterior es una derivación por hiper-resolución de S. Además, es una refutación ya que $< > \in S_2$.

El lema siguiente enuncia que una hiper-resolvente ya sea positiva ó negativa es consecuencia del núcleo y electrones de los cuales se infiere.

Lema 1.9:

Si D es la hiper-resolvente del núcleo C y electrones $C_1, C_2, \ldots, C_n$, entonces $\{C, C_1, C_2, \ldots, C_n\} \models D$.

Prueba:

Sea $\mathbf{A}$ una asignación tal que $\mathbf{A} \models \{C, C_1, C_2, \ldots, C_n\}$.

Primeramente tomemos el caso de una aplicación de hiper-resolución positiva.

Sean $C = <l_1, l_2, \ldots, l_n \rightarrow \psi>$, $C_i = <\rightarrow \psi_i, l_i>$ y $D = <\rightarrow \psi, \psi_1, \psi_2, \ldots, \psi_n>$.

Como $\mathbf{A} \models C$, tenemos que $l_i \not\subset \mathbf{A}$ para alguna l_i o $\psi \cap \mathbf{A} \neq \varnothing$ y como $\mathbf{A} \models C_i$, tenemos que $(\psi_i \cup l_i) \cap \mathbf{A} \neq \varnothing$.

Queremos probar que $\mathbf{A} \models D$, o bien que $\{\psi, \psi_1, \psi_2, \ldots, \psi_n\} \cap \mathbf{A} \neq \varnothing$.

Si $l_i \not\subset \mathbf{A}$ y como sabemos que $\{\psi_i \cup l_i\} \cap \mathbf{A} \neq \varnothing$, entonces tenemos que $\{\psi_i\} \cap \mathbf{A} \neq \varnothing$ y por tanto $\{\psi, \psi_1, \psi_2, \ldots, \psi_n\} \cap \mathbf{A} \neq \varnothing$, luego $\mathbf{A} \models D$.

Si $\psi \cap \mathbf{A} \neq \varnothing$ y $\{\psi\} \subseteq \{\psi, \psi_1, \psi_2, \ldots, \psi_n\} \cap \mathbf{A} \neq \varnothing$, entonces tenemos que $\{\psi, \psi_1, \psi_2, \ldots, \psi_n\} \cap \mathbf{A} \neq \varnothing$, luego $\mathbf{A} \models D$.

Ahora veamos que sucede con la aplicación de hiper-resolución negativa.

Sean $C = \langle \psi \rightarrow l_1, l_2, \ldots, l_n \rangle$, $C_i = \langle \psi_i, l_i \rightarrow \rangle$ y $D = \langle \psi, \psi_1, \psi_2, \ldots, \psi_n \rightarrow \rangle$.

Como $\mathbf{A} \models C$, tenemos que $\{l_i\} \cap \mathbf{A} \neq \varnothing$ para alguna l_i o $\psi \not\in \mathbf{A}$ y como $\mathbf{A} \models C_i$, tenemos que $\{l_i\} \cap \mathbf{A} = \varnothing$ o $\psi_i \cap \mathbf{A} = \varnothing$.

Queremos probar que $\mathbf{A} \models D$, o bien que $\{\psi, \psi_1, \psi_2, \ldots, \psi_n\} \cap \mathbf{A} = \varnothing$.

Si $\{l_i\} \cap \mathbf{A} \neq \varnothing$, entonces $\psi_i \cap \mathbf{A} = \varnothing$. Luego $\{\psi, \psi_1, \psi_2, \ldots, \psi_n\} \cap \mathbf{A}$ por lo que $\mathbf{A} \models D$.

Si $\psi \not\in I$, entonces $\mathbf{A} \models D$.

Por tanto si D es la hiper-resolvente del núcleo C y electrones $C_1, C_2, \ldots, C_n$, entonces $\{C, C_1, C_2, \ldots, C_n\} \models D$.

Definición 1.32: (*conjunto de cláusulas cerrado bajo H*)

Un conjunto de cláusulas S se dice cerrado bajo ***H*** si cada resolvente de cláusulas en S es redundante con respecto a S.

Cuando tenemos un conjunto de cláusulas satisfacible y le aplicamos la regla de hiper-resolución para obtener un nuevo conjunto de cláusulas, este también es satisfacible. Esto queda establecido en el lema siguiente.

Lema 1.10:

Sean S y S' conjuntos de cláusulas tal que $S \Rightarrow_H S'$ Luego S' es satisfacible sii S lo es. Equivalentemente, S es insatisfacible si S' lo es.

Prueba:

Probemos que si S es satisfacible, entonces S' también lo es.

Si la cláusula que se introduce por la regla de hiper-resolución es una hiper-resolvente D del núcleo C y electrones C_1, C_2,..., C_n en S, entonces $S' = S \cup D$. De acuerdo al lema 1.9 si $\mathbf{A} \models S$, entonces de $\mathbf{A} \models S \cup D$, luego $\mathbf{A} \models S'$

Por lo tanto si S es satisfacible, entonces S' también lo es.

Ahora probemos que si S' es satisfacible, entonces S también lo es.

Suponga que S' es satisfacible y $S' = S \cup D$.

Como existe $\mathbf{A} \models S'$ y $S \subset S'$, entonces $\mathbf{A} \models S$.

Por lo tanto si S' es satisfacible, entonces S también lo es.

Luego S' es satisfacible sii S lo es.

Ejemplo 1.23:

Del conjunto de cláusulas del ejemplo 1.22 podemos decir que los conjuntos S_0 y S_1 no son cerrados bajo $\boldsymbol{H}$, ya que en ninguno de ellos aparece la cláusula $< >$ y tal cláusula no es redundante con respecto a S_0 ni a S_1. El conjunto S_2 es cerrado bajo $\boldsymbol{H}$.

El conjunto de cláusulas S_2 es insatisfacible ya que $< > \in S_2$. De acuerdo al lema 1.10 podemos decir que S_1 también es insatisfacible por serlo S_2, por el mismo lema decimos que S_0 es insatisfacible al serlo S_1. Luego, podemos decir que S es insatisfacible.

El primer resultado importante que introducimos acerca del sistema de hiper-resolución es el de su solidez. Este resultado establece que si la cláusula vacía aparece en algún conjunto de cláusulas obtenido en una derivación por hiper-resolución, entonces el conjunto de cláusulas considerado inicialmente es insatisfacible.

Teorema 1.8: (*solidez de hiper-resolución*):

Si S_0, S_1, ..., S_n es una derivación por hiper-resolución con $< > \in S_n$, entonces S_0 es *insatisfacible*.

Prueba:

Ya que $< >$ y por tanto S_n es insatisfacible, repetidas aplicaciones del lema anterior garantiza que cada uno de los conjuntos de cláusulas S_i sean también insatisfacibles.

Si tenemos un conjunto cerrado bajo $\boldsymbol{H}$ y la cláusula vacía no pertenece a ese conjunto, entonces tal conjunto es satisfacible. Este hecho queda establecido en el lema siguiente.

Lema 1.11:

Sea S un conjunto de cláusulas.

Si S es cerrado bajo $\boldsymbol{H}$ y si $<> \notin S$, entonces S es satisfacible.

Prueba:

Supongamos que S es cerrado bajo $\boldsymbol{H}$, $<> \notin S$ y por el contrario que S es insatisfacible.

Con la construcción de un modelo para S podremos probar este lema.

Sea $S^+ = \{ \phi \rightarrow \psi \mid \phi = \varnothing \}$ el conjunto de las cláusulas positivas en S.

Si $S^+ = \varnothing$, entonces $\mathbf{A} = \varnothing$ es un modelo para S. Así que suponga $S^+ \neq \varnothing$.

Sea X el conjunto de todos los átomos que aparecen en cláusulas de S^+ y sea $\mathbf{A}$ el subconjunto minimal de X el cual cubre a S+.

Probaremos que $\mathbf{A}$ es un modelo para S. Es claro que $\mathbf{A}$ es un modelo para S^+.

Supongamos que existe alguna cláusula $C = <\phi \rightarrow \psi> \in S$, la cual no satisface $\mathbf{A}$.

Luego $\phi \subseteq \mathbf{A}$ y $\psi \cap \mathbf{A} = \varnothing$.

Si $\phi = \varnothing$, entonces $C \in S^+$ y se satisface mediante $\mathbf{A}$, por ello sea $\phi = \{l_1, l_2, \ldots, l_n\}$.

Elija una cláusula $D_i \in S^+$ tal que l_i es el único elemento de $\mathbf{A}$ que aparece en D_i, así:

$C = <l_1, l_2, \ldots, l_n \rightarrow \psi>$ y $D_i = <\rightarrow l_i, \psi_i>$.

donde $\psi \cap \mathbf{A} = \varnothing$ y la hiper-resolvente R de C con $D_1, D_2, \ldots, D_n$ es de la forma, $R = <\rightarrow \psi, \psi_1, \psi_2, \ldots, \psi_n>$. El conjunto S debe contener a R ya que S es cerrado bajo H.

Como $\psi \cap \mathbf{A} = \varnothing$ y $\psi_i \cap \mathbf{A} = \varnothing$, entonces $\{\psi, \psi_1, \psi_2, \ldots, \psi_n\} \cap \mathbf{A} = \varnothing$ y por tanto $\mathbf{A}$ no satisface R, pero R es una cláusula positiva en S y toda cláusula positiva de S se satisface por $\mathbf{A}$.

Hemos construido un modelo para S, por lo que S es satisfacible y lo cual es una contradicción.

Por lo tanto, si S es cerrado bajo $\boldsymbol{H}$ y si $<> \notin S$, entonces S es satisfacible.

Definición 1.33:

Una derivación por hiper-resolución $S_0, S_1, \ldots$ se dice *justa* si el conjunto: $S^{\infty} = \bigcup_{k \geq 0} \bigcap_{j \geq k} S_j$ es cerrado bajo $\boldsymbol{H}$.

El siguiente lema utiliza las definiciones anteriores y es el último que utilizaremos para probar la completud del principio de hiper-resolución.

Lema 1.12:

Si el conjunto de cláusulas S es insatisfacible, y si $S_0, S_1, S_2, \ldots$ es una derivación justa por hiper-resolución con $S_0 = S$, entonces S^∞ es insatisfacible.

Prueba:

Supongamos que S es insatisfacible y S^∞ es satisfacible.

Como $S_0 \subset S^\infty$ y $S_0 = S$, tenemos que $S \subset S^\infty$ Como ya sabemos que si $X \subset T$ y X es insatisfacible, entonces T también lo es. Por lo tanto, S^∞ es insatisfacible.

Ejemplo 1.24:

Siguiendo con la derivación del ejemplo 1.22 obtenemos:

$\Rightarrow_H S_3 = \{<\leftrightarrow p>, <\leftrightarrow q, r, s>, <p, q \rightarrow>, <r \rightarrow>, <\leftrightarrow r>, < >, <q \rightarrow>\}$

Luego, al aplicar la regla de hiper-resolución a $<\leftrightarrow q, r, s>$, $<r \rightarrow>$ y a $<q \rightarrow>$ obtenemos:

$\Rightarrow_H S_4 = \{<\leftrightarrow p>, <\leftrightarrow q, r, s>, <p, q \rightarrow>, <r \rightarrow>, <\leftrightarrow r>, < >, <q \rightarrow>, <\leftrightarrow s>\}$

Después de aplicar la regla de hiper-resolución a $<\leftrightarrow q, r, s>$ y $<q \rightarrow>$ tenemos:

$\Rightarrow_H S_5 = \{<\leftrightarrow p>, <\leftrightarrow q, r, s>, <p, q \rightarrow>, <r \rightarrow>, <\leftrightarrow r>, < >, <q \rightarrow>, <\leftrightarrow s>, <\leftrightarrow r, s>\}$

Luego sobre $<\leftrightarrow q, r, s>$ y $<r \rightarrow>$ se aplicó la regla de hiper-resolución con lo que:

$\Rightarrow_H S_6 = \{<\leftrightarrow p>, <\leftrightarrow q, r, s>, <p, q \rightarrow>, <r \rightarrow>, <\leftrightarrow r>, < >, <q \rightarrow>, <\leftrightarrow s>, <\leftrightarrow r, s>, <\leftrightarrow q, s>\}$

el conjunto $S^\infty = S_6$ es cerrado bajo $\boldsymbol{H}$ y en consecuencia es insatisfacible, por lema 1.12.

Si un conjunto de cláusulas es insatisfacible, entonces en una derivación justa por hiper-resolución aparecerá en alguno de sus elementos la cláusula vacía.

Teorema 1.9: (*completud de hiper-resolución*):

Si S es un conjunto de cláusulas insatisfacible y si $S_0, S_1, S_2, \ldots$ es una derivación justa por hiper-resolución con $S_0 = S$, entonces existe una $n \geq 0$ tal que S_n contiene la cláusula vacía $< >$.

Prueba:

Sea $S_0, S_1, S_2, \ldots$ una derivación justa por hiper-resolución con $S_0 = S$. Entonces S^∞ es cerrado bajo $\boldsymbol{H}$ y es insatisfacible, por el lema anterior. Además, por el lema 1.11, tenemos que $<> \in S^\infty$

Luego, existe una $n \geq 0$ tal que $<> \in \cap_{j \geq n} S_j$. Esto a su vez implica que $<> \in S_n$.

Podemos ilustrar una derivación por hiper-resolución por medio de un árbol de prueba por hiper-resolución, en el cual podemos apreciar claramente las cláusulas que tomamos para obtener cada hiper-resolvente.

Definición 1.34: (*árbol de prueba por hiper-resolución*)

Sean C y D cláusulas.

Sea S un conjunto de cláusulas, donde $\{C_0, C_1, \ldots, C_n\} \subseteq S$.

Un *árbol de prueba por hiper-resolución* de S a partir de C es un árbol binario y etiquetado T con las siguientes propiedades:

i. La raíz de T está etiquetada con C.

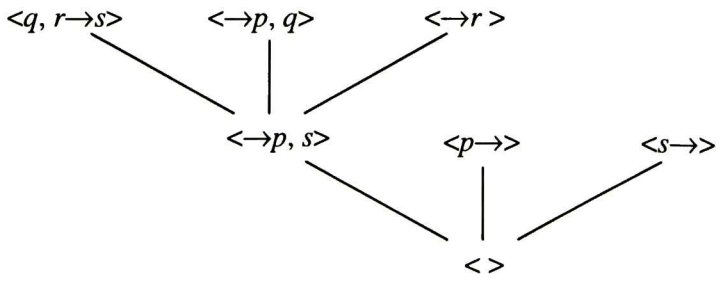
ii. Las hojas de T están etiquetadas con elementos de S.

iii. Si un nodo tiene etiqueta D y sucesores inmediatos con etiqueta $C_0, C_1, \ldots, C_n$, entonces D es la hiper-resolvente de C_0, de $C_1, \ldots$, y de C_n.

Ejemplo 1.25:

Sea $S = \{<\rightarrow p, q>, <\rightarrow r>, <q, r\rightarrow s>, <p\rightarrow>, <s\rightarrow>\}$

Un árbol de prueba por hiper-resolución a partir de $<>$ es el siguiente:



II Redes de Petri

1 Introducción

El primer paso para describir un sistema es elaborar un modelo del mismo. Dependiendo de los objetivos que se persigan se pueden construir modelos de diferentes tipos. Para el trabajo que estamos desarrollando se utilizarán las Redes de Petri (RdP), primeramente para modelar y luego para concluir sobre el problema modelado.

En la sección 2 de este capítulo presentamos los conceptos básicos acerca de las RdP en los cuales fundamentamos nuestro trabajo. Iniciamos con la definición de las RdP generalizadas, así como con las formas de representarlas: gráfica y matricialmente. Luego, definimos a una RdP pura, con las cuales trabajaremos. Posteriormente, para hacer más compacta la escritura introducimos una notación para los lugares de salida y de entrada de una transición, así como para las transiciones de entrada y de salida de un lugar. Ahora bien, una RdP tiene estados, cada uno dado por el marcado actual. Para pasar de un marcado a otro debemos de disparar transiciones habilitadas lo que conlleva a eliminar y añadir marcas en algunos lugares de la RdP. Por ello, definimos a un marcado de la RdP, a la regla para habilitar una transición, a la regla de evolución del marcado, a una secuencia de disparos, al vector característico asociado a una secuencia de disparos y a un marcado alcanzable. Ya al final de la sección 2, definimos a las componentes repetitivas, llamadas T-invariantes y a las componentes conservativas, llamadas P-invariantes.

En el trabajo que exponemos en el capítulo 3 usamos todas las definiciones que presentamos en la sección 2 de este capítulo, aunque con algunas modificaciones en algunas de ellas. Presentamos tales modificaciones en la sección 3 de este capítulo. Iniciamos tal sección con la redefinición de una marca en la RdP, la cual es necesaria debido a la semántica que le damos a las marcas a lo largo de nuestro trabajo. Al redefinir a las marcas, luego redefinimos también al marcado de la RdP, a la regla para habilitar una transición y a la regla de evolución del marcado. Por último, definimos a lo que llamamos un T-invariante válido, concepto fundamental en nuestro trabajo.

2. Definiciones Básicas

En esta sección presentamos algunos conceptos esenciales acerca de las RdP en los cuales se fundamenta nuestro trabajo. Además de presentar tales conceptos, introducimos algunos ejemplos que esperamos ilustren los conceptos a los cuales hacen referencia.

Definición 2.1: (*RdP generalizada*)

Una RdP generalizada ó simplemente RdP es una cuádrupla $N = <P, Tr, \alpha, \beta>$ tal que

P es un conjunto finito y no vacío de lugares.

Tr es un conjunto finito y no vacío de transiciones.

$P \cap Tr = \varnothing$; es decir, lugares y transiciones son conjuntos disjuntos.

$\alpha : P \times Tr \rightarrow \mathbf{N}$ es la función de incidencia previa, donde **N** denota el conjunto de los números naturales.

$\beta : Tr \times P \rightarrow \mathbf{N}$ es la función de incidencia posterior.

Ejemplo 2.1:

Sea $N = <P, Tr, \alpha, \beta>$, tal que $P = \{l_1, l_2, l_3, l_4\}$, $Tr = \{ t_1, t_2, t_3, t_4\}$ y funciones de incidencia:

Previa

α	t_1	t_2	t_3	t_4
l_1	0	1	0	0
l_2	0	0	1	0
l_3	0	0	0	1
l_4	0	0	0	1

Posterior

β	l_1	l_2	l_3	l_4
t_1	1	1	0	0
t_2	0	0	1	0
t_3	0	0	0	1
t_4	0	0	0	0

Con lo cual queda definida *N*.

Podemos representar a una RdP gráfica o matricialmente. Cada una de ellas tienen sus ventajas y dependiendo del objetivo será la representación que elegiremos. Cuando distribuimos adecuadamente los elementos de una RdP en una representación gráfica, podemos visualizar fácilmente algunas características de la misma. Para efectos de análisis utilizamos mayormente una representación matricial.

De aquí en delante $N = <P = \{p_1, p_2, \ldots, p_m\}, Tr = \{t_1, t_2, \ldots, t_n\}, \alpha, \beta>$ denotará una RdP, t una transición y p un lugar, a menos que indiquemos lo contrario.

2.1 Representación gráfica

La representación gráfica de una RdP es mediante un grafo bipartito dirigido. En el cual, representamos a los lugares por circunferencias y a las transiciones por barras. Existe un arco

que va del lugar p_i a la transición t_j sii $\alpha(p_i, t_j) \neq 0$. Análogamente, existe un arco que va de la transición t_k al lugar p_r sii $\beta(t_k, p_r) \neq 0$. Etiquetamos a cada arco con un número natural, $\alpha(p_i, t_j)$ o $\beta(t_j, p_i)$, que se denomina peso del arco. Por convenio, un arco no etiquetado posee un peso unitario.

Ejemplo 2.2:

En la RdP del ejemplo 2.1 hay 4 lugares con etiquetas: l_1, l_2, l_3 y l_4, por lo que en la representación gráfica de N deben de aparecer 4 circunferencias con etiquetas l_1, l_2, l_3 y l_4, respectivamente. De forma similar, las transiciones t_1, t_2, t_3 y t_4 están en tal RdP, por lo que en la gráfica deben de aparecer también 4 barras con etiquetas t_1, t_2, t_3 y t_4, respectivamente.

Ahora bien, un arco va de t_1 a l_1 ya que $\beta(t_1, l_1) \neq 0$. Un arco más va de l_1 a t_2 ya que $\alpha(l_1, t_2) \neq 0$. La existencia de los demás arcos en la gráfica se justifica por razones similares a las expuestas en los dos casos anteriores.

La representación gráfica de la RdP es la que mostramos a continuación. En ella no etiquetamos a los arcos ya que todos tienen peso igual a 1.

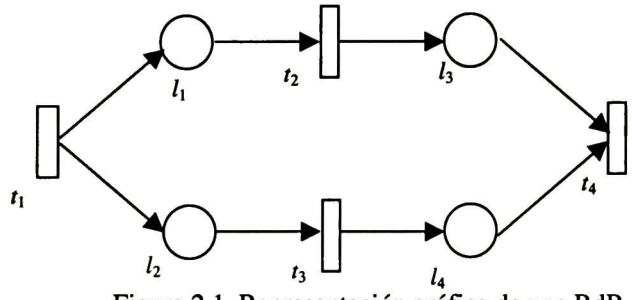


Figura 2.1 Representación gráfica de una RdP

2.2 Representación matricial

Representamos matricialmente a una RdP por medio de dos matrices.

Se denomina matriz de *incidencia previa* a la matriz:

$A^- = [a_{ij}]_{mxn}$, donde $a_{ij} = \alpha(p_i, t_j)$.

Se denomina matriz de *incidencia posterior* a la matriz:

$A^+ = [a_{ij}]_{mxn}$, donde $a_{ij} = \beta(t_j, p_i)$.

Ejemplo 2.3:

Mostramos enseguida a la matriz de incidencia previa y a la matriz de incidencia posterior de la RdP de los ejemplos 2.1 y 2.2.

$$A^- = \begin{array}{c} \\ l_1 \\ l_2 \\ l_3 \\ l_4 \end{array} \begin{array}{c} \begin{array}{cccc} t_1 & t_2 & t_3 & t_4 \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{array} \qquad A^+ = \begin{array}{c} \\ l_1 \\ l_2 \\ l_3 \\ l_4 \end{array} \begin{array}{c} \begin{array}{cccc} t_1 & t_2 & t_3 & t_4 \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \end{array}$$

Cuando una RdP es pura se simplifica su representación matricial, tal como observamos a continuación.

Definición 2.2: (RdP *pura*)

Una RdP es *pura* si ninguna transición contiene un lugar que sea simultáneamente de entrada y de salida. Expresado de otra forma $\alpha(p_i, t_j)*\beta(t_j, p_i) = 0$, para $t_j \in Tr$ y $p_i \in P$

La representación matricial de una RdP pura se simplifica al definir sólo una matriz A, denominada matriz de incidencia.

$$A = A^+ - A^-$$

Ejemplo 2.4:

La RdP de los ejemplos 2.1, 2.2 y 2.3 es pura.

Una RdP que no es pura es la que observamos en la gráfica siguiente.

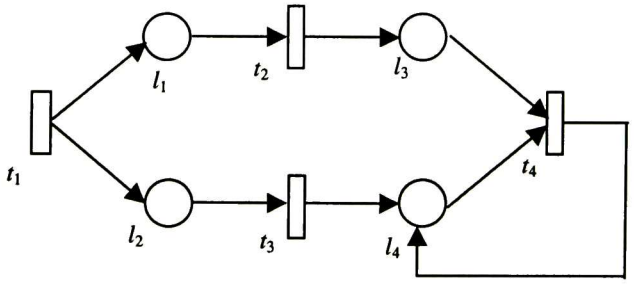


Figura 2.2 Representación gráfica de una RdP que no es pura

La RdP anterior no es pura ya que l_4 es un lugar tanto de entrada como de salida de t_4. Tenemos que utilizar la matriz de incidencia previa y la matriz de incidencia posterior para describir la RdP anterior, ya que no es posible especificar en la matriz de incidencia que un lugar pertenece tanto al conjunto de lugares de entrada a una transición, como al conjunto de lugares de salida de dicha transición.

Definición 2.3: (*conjunto de lugares (transiciones) de entrada y de salida de una transición (un lugar)*)

Definimos los siguientes conjuntos:

a) Conjunto de lugares de entrada de t: $\bullet t = \{p \in P \mid \alpha(p, t)>0\}$

b) Conjunto de lugares de salida de t: $t\bullet = \{p \in P \mid \beta(t, p)>0\}$

c) Conjunto de transiciones de entrada de p: $\bullet p = \{t \in Tr \mid \beta(t, p)>0\}$

d) Conjunto de transiciones de salida de p: $p\bullet = \{t \in Tr \mid \alpha(p, t)>0\}$

Ejemplo 2.5:

Si consideramos a la RdP del ejemplo 2.1 tenemos que:

$\bullet t_1 = \{\ \}$, $\bullet t_2 = \{ l_1\}$, $\bullet t_3 = \{ l_2\}$ y $\bullet t_4 = \{ l_3, l_4\}$

$t_1\bullet = \{ l_1, l_2\}$, $t_2\bullet = \{ l_3\}$, $t_3\bullet = \{ l_4\}$ y $t_4\bullet = \{\ \}$

$\bullet l_1 = \{ t_1\}$, $\bullet l_2 = \{ t_1\}$, $\bullet l_3 = \{ t_2\}$ y $\bullet l_4 = \{ t_3\}$

$l_1\bullet = \{ t_2\}$, $l_2\bullet = \{ t_3\}$, $l_3\bullet = \{ t_4\}$ y $l_4\bullet = \{ t_4\}$

Para la RdP no pura del ejemplo 2.4 tenemos que:

$\bullet t_1 = \{\ \}$, $\bullet t_2 = \{ l_1\}$, $\bullet t_3 = \{ l_2\}$ y $\bullet t_4 = \{ l_3, l_4\}$

$t_1\bullet = \{ l_1, l_2\}$, $t_2\bullet = \{ l_3\}$, $t_3\bullet = \{ l_4\}$ y $t_4\bullet = \{ l_4 \}$

$\bullet l_1 = \{ t_1\}$, $\bullet l_2 = \{ t_1 \}$, $\bullet l_3 = \{ t_2\}$ y $\bullet l_4 = \{ t_3, t_4 \}$

$l_1\bullet = \{ t_2\}$, $l_2\bullet = \{ t_3\}$, $l_3\bullet = \{ t_4\}$ y $l_4\bullet = \{ t_4\}$

Algunas transiciones de nuestro particular interés, son las que mencionamos en la definición siguiente.

Definición 2.4: (*transición fuente*)

Decimos que la transición t es una transición fuente sii $\bullet t = \varnothing$.

Ejemplo 2.6:

En la RdP del ejemplo 2.4 la transición t_1 es una transición fuente, ya que $\bullet t_1=\varnothing$.

Un modelo sin dinámica es como un automóvil que no camina. Las RdP tienen una dinámica, pero antes de que entremos a ese punto, enunciaremos algunos conceptos indispensables para su completa definición.

Definición 2.5: (*marcado*)

El marcado K de N es una aplicación de P en **N**, es decir, la asignación de un número entero no negativo (número de marcas) a cada lugar.

El marcado K lo denotamos por el vector:

$K = [a_i]_m$, en el que $a_i = k$, $k \in \mathbf{N}$ y cada componente a_i está asociada con el lugar l_i en P.

Ejemplo 2.7:

Sean $N = <\{l_1, l_2, l_3\}, \text{Tr}, \alpha, \beta>$ y $K = [1, 0, 2]$ el marcado actual de N.

Si nos encontramos en el marcado K de N, entonces decimos que hay una marca en el lugar l_1, cero marcas en el lugar l_2 y dos marcas en el lugar l_3.

Como observamos en este ejemplo, la correspondencia de cada componente del vector K con cada lugar en N, se realiza considerando un orden lexicográfico de los lugares.

Para que un automóvil camine son necesarias algunas condiciones: gasolina, máquina funcionando, etc.. Ahora bien, para que exista en la RdP una evolución o presente una dinámica, es necesario que existan transiciones habilitadas, las cuales definimos a continuación.

Definición 2.6: (*transición habilitada*)

La transición t está habilitada por el marcado K sii para todo $p \in {}^\bullet t$, el lugar p posee al menos $\alpha(p, t)$ marcas.

Ejemplo 2.8:

Consideremos a la RdP del ejemplo 2.1. Supongamos que para esa RdP $\alpha(l_i, t_j) = 1$, donde $l_i \in \text{P}$, $t_j \in \text{Tr}$ y el marcado actual es $K_1 = [1, 2, 0, 3]$.

La transición t_2 está habilitada ya que cada uno de sus lugares de entrada (l_1) contiene al menos una marca.

Lo mismo sucede para la transición t_3, por tanto también t_3 está habilitada en K_1.

Podemos decir que todos los lugares de entrada de t_1 poseen al menos una marca ($\bullet t_1 = \varnothing$), por ello t_1 está habilitada.

La única transición que no está habilitada en K_1 es t_4, ya que uno de sus lugares de entrada (l_3) no posee al menos una marca.

A continuación describimos con detalle la evolución ó dinámica de una RdP.

Definición 2.7: (*regla de evolución del marcado*)

Sean $t \in$ Tr una transición habilitada y $p_i, p_j \in$ P.

Disparar la transición t es la operación que consiste en eliminar $\alpha(p_i, t)$ marcas a cada lugar $p_i \in \bullet t$ y añadir $\beta(t, p_j)$ marcas a cada lugar $p_j \in t\bullet$.

La notación $K_i \xrightarrow{t} K_j$ significa que t está habilitada en el marcado K_i y que al disparar la transición t a partir del marcado K_i, alcanzamos el marcado K_j.

Ejemplo 2.9:

Consideremos a la RdP del ejemplo anterior y su marcado.

Si disparamos la transición habilitada t_2 en el marcado K_1, entonces alcanzamos el marcado K_2 = [0, 2, 1, 3], ya que al disparo de t_2 se elimina una marca de cada uno de sus lugares de entrada (l_1) y se añade una marca a cada uno de sus lugares de salida (l_3).

Si en el marcado K_1 hubiéramos disparado la transición t_1 hubiéramos alcanzado el marcado K_2 = [2, 3, 0, 3], ya que sólo se hubiera añadido una marca a cada uno de sus lugares de salida.

Al realizar una secuencia de disparos en una RdP consideramos que cada transición estuvo habilitada antes de su disparo. A continuación introducimos la notación para una secuencia de disparos.

Definición 2.8: (*secuencia de disparos*)

Una secuencia de disparos realizable a partir del marcado K_0 se representa por una secuencia de transiciones, tal que el disparo de cada transición conduce a un marcado que habilita la transición siguiente de la secuencia.

Si $K_0 \xrightarrow{t_i} K_1 \xrightarrow{t_j} K_2 \longrightarrow \ldots \xrightarrow{t_r} K_q$, entonces decimos que la secuencia $\sigma = t_i t_j \ldots t_r$ es realizable a partir de K_0. La anterior evolución del marcado se puede condensar escribiendo $K_0 \xrightarrow{\sigma} K_q$

Ejemplo 2.10:

Consideremos a la RdP del ejemplo 2.8 y supongamos que su marcado es $K_0 = [0, 0, 0, 0]$.

En K_0 sólo una transición se encuentra habilitada y es t_1.

Al disparo de t_1 alcanzamos el marcado $K_1 = [1, 1, 0, 0]$. En K_1 se encuentran habilitadas las transiciones t_1, t_2 y t_3. Si en K_1 elegimos disparar de nuevo t_1, entonces alcanzamos el marcado $K_2 = [2, 2, 0, 0]$. En este marcado se encuentran de nuevo habilitadas las transiciones t_1, t_2 y t_3. Si en este caso elegimos disparar t_2, entonces alcanzamos el marcado $K_3 = [1, 2, 1, 0]$. En K_3 si disparamos la transición t_3, entonces alcanzamos el marcado $K_4 = [1, 1, 1, 1]$. Por último si disparamos t_4 en K_4, entonces alcanzamos el marcado $K_5 = [1, 1, 0, 0]$.

Con lo anterior, podemos decir que la secuencia $\sigma = t_1\ t_1\ t_2\ t_3\ t_4$ es realizable a partir de K_0.

Como $K_0 \xrightarrow{t_1} K_1 \xrightarrow{t_1} K_2 \xrightarrow{t_2} K_3 \xrightarrow{t_3} K_4 \xrightarrow{t_4} K_5$, entonces podemos decir que $K_0 \xrightarrow{\sigma} K_5$.

Cuando no nos interesa el orden en el cual realizamos una secuencia de disparos y únicamente es relevante el saber cuantas veces se disparó cada transición en la secuencia, es conveniente utilizar una notación para ello.

Definición 2.9: (*vector característico asociado a una secuencia de disparos*)

Se llama vector característico asociado a una secuencia de disparos σ al vector $\vec{\sigma} \in \mathbf{N}^n$, cuya i-ésima componente es igual al número de ocurrencias de t_i en σ.

Ejemplo 2.11:

El vector característico de la secuencia de disparos del ejemplo anterior es [2, 1, 1, 1] ya que disparamos en dos ocasiones a t_1 y en una ocasión cada una de las otras transiciones.

Definición 2.10: (*marcado alcanzable*)

Decimos que el marcado K es alcanzable a partir de un marcado inicial K_0 sii existe una secuencia de disparos realizable σ a partir de K_0 la cual transforma K_0 en K, es decir, existe σ tal que $K_0 \xrightarrow{\sigma} K$

Ejemplo 2.12:

Del ejemplo 2.10 podemos decir que los marcados $K_1 = [1, 1, 0, 0]$, $K_2 = [2, 2, 0, 0]$, $K_3 = [1, 2, 1, 0]$, $K_4 = [1, 1, 1, 1]$ y $K_5 = [1, 1, 0, 0]$ son alcanzables a partir de K_0, ya que para cada uno de ellos existe una secuencia realizable, tal que del marcado inicial obtenemos tales marcados.

Existen ciertas componentes en la RdP que son llamadas componentes repetitivas, las cuales indican que posiblemente exista una secuencia de disparos σ, tal que si K_0 es el marcado inicial, entonces $K_0 \xrightarrow{\sigma} K_0$.

En las siguientes definiciones A denota la matriz de incidencia de la RdP.

Definición 2.11: (T-invariante)

Al vector $Y \in \mathbf{N}^n$ lo denominamos T-invariante ó componente repetitiva de N, sii Y es la solución del sistema de ecuaciones lineales A, tal que $AY = \mathbf{0}$.

Otras componentes en la RdP llamadas componentes conservativas, son aquellas que indican que posiblemente exista un conjunto de lugares en los cuales la suma de las marcas en ellos siempre es igual a k, es decir, las marcas en tales lugares se conservan.

Definición 2.12: (P-invariante)

Al vector $Y \in \mathbf{N}^m$ lo denominamos P-invariante ó componente conservativa de N, sii Y es la solución del sistema de ecuaciones lineales A, tal que $YA = \mathbf{0}$.

Ejemplo 2.13:

La matriz de incidencia de la RdP del ejemplo 2.1 es:

$$A = \begin{array}{c} \\ l_1 \\ l_2 \\ l_3 \\ l_4 \end{array} \begin{array}{c} \begin{array}{cccc} t_1 & t_2 & t_3 & t_4 \end{array} \\ \begin{bmatrix} 1 & -1 & 0 & 0 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & -1 \end{bmatrix} \end{array}$$

Un T-invariante para la RdP anterior es $Y = [1, 1, 1, 1]^T$ ya que $AY = \mathbf{0}$. Esta RdP no presenta P-invariantes.

La RdP del ejemplo 2.4 no presenta T-invariantes.

3. Modificaciones

Para el desarrollo y prueba de nuestro trabajo, el cual presentamos en el capítulo 3, utilizaremos la definición de una RdP generalizada, sólo que a las marcas les añadiremos información (colores). Por lo anterior, será necesario que redefinamos al marcado, a la regla para habilitar una transición y a la regla de evolución del marcado. Todo esto lo redefinimos en ese orden en esta sección, junto con lo que llamamos T-invariante válido. Este último, introduce un significado especial a las componentes repetitivas o T-invariantes.

Definición 2.13: (*marcas*)

Una *marca M* en la RdP *N* es una pareja $<D, I>$, donde $D \in \{1, 2, \ldots, n\}$ e $I \in \mathbf{N}$.

En el marcado de una RdP generalizada sólo usamos números enteros para denotar el número de marcas en cada lugar. Debido a la definición anterior, ya no es suficiente el saber que en un lugar hay cierto número de marcas, es necesario además, conocer qué marcas se encuentran en cada lugar, es decir, el valor de *D* y de *I*, las cuales conforman a cada marca.

Definición 2.14: (*marcado*)

Sea M^* la colección de todos los multiconjuntos de marcas en *N*.

Un *marcado K* de *N* es una función de P en M^*, el cual denotamos por el vector:

$K=[M_i]_m$, donde cada $M_i \in M^*$ y M_i está asociado con el lugar $l_i \in$ P.

El marcado vacío K_0 es el vector $[\varnothing, \varnothing, \ldots, \varnothing]$.

Para que exista una evolución en la RdP necesitamos también redefinir lo que es una transición habilitada. La cual difiere totalmente a la hecha en 2.6. Pero antes de que enunciemos la regla para habilitar una transición, necesitamos la siguiente definición.

Definición 2.15: (*marcas diferentes*)

Sean $M_1 = <D_1, I_1>$ y $M_2 = <D_2, I_2>$ marcas.

Decimos que la marca M_1 es diferente a la marca M_2 sii $D_1 \neq D_2$ ó $I_1 \neq I_2$.

Definición 2.16: (*regla para habilitar una transición*)

Decimos que *t* está *habilitada* sii todos sus lugares de entrada contienen marcas distintas entre sí.

Con todo lo que hemos redefinido anteriormente, también es necesaria la redefinición de la regla de evolución del marcado en la RdP.

Definición 2.17: (*regla de evolución del marcado*)

Sean $M_1, M_2, \ldots, M_j, j \geq 0$, las marcas que habilitan la transición t_i en N.

Sea k el número de veces que se ha disparado t_i.

Al disparo de t_i se añade una marca $M = <D = i, I = k + 1>$ a cada uno de sus lugares de salida y se elimina cada marca que la habilitó, es decir, si la marca M_r en $p \in \bullet t_i$ habilitó a t_i, entonces se elimina una marca M_r de p. Además, toda marca con referencia M_r se cambia a M.

En la siguiente definición añadimos algunas restricciones a la que presentamos en 2.11.

Definición 2.18 (*T-invariante válido*)

Sea K_0 el marcado vacío.

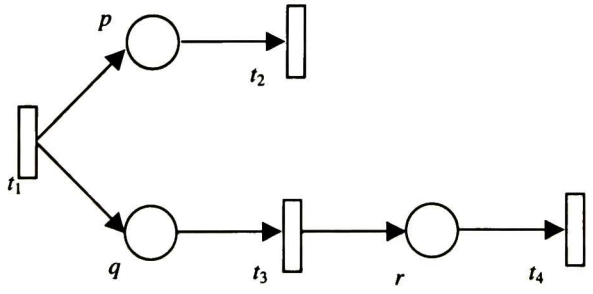
Sea σ una secuencia de disparos.

Decimos que un T-invariante es válido sii existe σ tal que $K_0 \xrightarrow{\sigma} K_0$ y el vector característico asociado a σ es ese T-invariante.

A continuación vemos un ejemplo en el cual podemos seguir fácilmente cada una de las definiciones que enunciamos anteriormente.

Ejemplo 2.14:

Consideremos la RdP que mostramos en la gráfica siguiente.



Si como condición inicial tenemos un marcado vacío, $K_0 = [\varnothing, \varnothing, \varnothing]$, entonces en ese estado la única transición habilitada es t_1 (todos sus lugares contienen marcas con referencia distinta entre sí, t_1 siempre esta habilitada).

Al disparo de t_1 obtenemos una marca <1,1> en el lugar p y otra marca <1, 1> en el lugar q, de acuerdo a la definición 2.17. La referencia D en las marcas anteriores es igual a 1, ya que de acuerdo a la misma definición 2.17, D es igual al subíndice de la transición que disparamos. Ahora bien, la referencia I de esas mismas marcas es también igual a 1, ya que se había disparado cero veces t_1 y también por la definición 2.17, I es igual a las veces que se ha disparado la transición en cuestión más uno. El marcado alcanzado es $K_1 = [<1, 1>, <1, 1>, \varnothing]$.

En K_1 está habilitada t_2 y también t_3. Si elegimos disparar t_3 eliminamos la marca <1, 1> del lugar q, añadimos la marca <3, 1> al lugar r y toda marca con referencia <1, 1> se cambia por <3, 1>, todo esto de acuerdo a la definición 2.17. De este modo llegamos al marcado $K_2 = [<3, 1>, \varnothing, <3, 1>]$.

La transición t_2 está habilitada en K_2, así como t_4. Si disparamos t_2, entonces eliminamos la marca <3, 1> del lugar p y no añadimos ninguna marca, ya que t_2 no tiene ningún lugar de salida, pero sí cambiamos toda marca con referencia <3, 1> por <2, 1> que es la referencia de la marca que hubiéramos añadido a los lugares de salida de t_2. El marcado $K_3 = [\varnothing, \varnothing, <2, 1>]$ es el que obtenemos.

En K_3 está habilitada t_4. Si la disparamos obtenemos el marcado $k_4 = [\varnothing, \varnothing, \varnothing, \varnothing]$.

La secuencia de disparos realizada es $\sigma = t_1 t_3 t_2 t_4$.

La matriz de incidencia A de N es:

$$A = \begin{array}{c} \\ p \\ q \\ r \end{array} \begin{array}{c} \begin{array}{cccc} t_1 & t_2 & t_3 & t_4 \end{array} \\ \begin{bmatrix} 1 & -1 & 0 & 0 \\ 1 & 0 & -1 & 0 \\ 0 & 0 & 1 & -1 \end{bmatrix} \end{array}$$

La RdP contiene el T-invariante $[1, 1, 1, 1]^T$ el cual es igual al vector característico de σ y debido a que de $K_0 \xrightarrow{\sigma} K_0$, se tiene que $[1, 1, 1, 1]^T$ es un T-invariante válido.

III Prueba de Satisfacibilidad de un Conjunto de Cláusulas

1. Introducción

En este capítulo presentamos un método que determina si un conjunto de cláusulas es insatisfacible. Además de presentar el método, probamos su validez y completud.

En la sección 2 hacemos una recopilación de los métodos basados en RdP que hemos estudiado, dando una breve explicación de los mismos. Luego, en la sección 3 presentamos la forma de construir una RdP a partir de un conjunto de cláusulas.

El método propuesto se sustenta en la proposición siguiente: "un conjunto de cláusulas S es insatisfacible sii existe un T-invariante válido en la RdP N asociada a S" La prueba de esta proposición la hemos dividido en dos partes, de manera que en la sección 4 probamos la validez del método y en la sección 5 su completud.

2. Métodos Estudiados

Los métodos que hemos estudiado asocian una RdP a un conjunto de cláusulas y luego determinan la validez de dicho conjunto. Ninguno de ellos es aplicable a cualquier tipo de cláusulas y algunos proponen la realización de una herramienta (programa), donde se ilustre la ejecución del método, pero no la presentan.

Las características principales de tales métodos se describen a continuación.

2.1 El Método de Peterka y Murata

El primer método que estudiamos fue el presentado por [Peterka and Murata, 1989], el cual se restringe al subconjunto de cláusulas de Horn de la lógica de predicados de primer orden con términos cerrados. De aquí tomamos la idea de como asociar una RdP a un conjunto de cláusulas.

Este método únicamente necesita determinar si existe un T-invariante en la RdP y si en el soporte de ese T-invariante (esto es, en el conjunto de transiciones asociadas a las componentes no nulas del T-invariante) se encuentra una transición determinada. Esta transición está asociada a una consulta, es decir, a una fórmula cerrada, y si se cumple lo

anterior, podemos decir que la consulta es válida, esto es, que es consecuencia del conjunto de cláusulas de Horn asociado a la RdP.

2.2 El Método de Lautenbach

Otro método que estudiamos, similar al anterior, es el que se muestra en [Lautenbach, 1985]. Este método usa la misma técnica que el anterior para asociar una RdP a un conjunto de cláusulas. Además, está basado en la afirmación de que un conjunto de cláusulas S asociado con la RdP N es insatisfacible sii se cumplen las dos condiciones siguientes:

a. Existe un T-invariante en N el cual reproduce el marcado vacío y cuyo vector característico de la secuencia realizada es ese T-invariante.

b. La RdP determinada por ese T-invariante (Definición 4.3) no contiene P-invariantes.

Más tarde encontramos, mediante un contraejemplo, que el teorema anterior es válido únicamente en un sentido, es decir, que si S es insatisfacible, entonces se cumplen las condiciones a y b, anteriores.

Este método no se restringe a cláusulas de Horn, trabaja con enunciados de la lógica proposicional y de la lógica de predicados de primer orden, pero no es completo.

2.3 El Método de Sinachopoulos

Después de haber estudiado el método presentado en [Lautenbach, 1985] nos encontramos con que la Dra. Sinachopoulos había presentado un contraejemplo para el método de Lautenbach. Ella emplea la misma técnica que Lautenbach, pero sólo se restringe a cláusulas de Horn de la lógica proposicional. Cabe resaltar que prueba la validez y completud de su método [Sinachopoulos, 1987].

El método de la Dra. Sinachopoulos presenta el resultado siguiente: Un conjunto de cláusulas de Horn S es insatisfacible sii existe un T-invariante en la RdP N asociada a S.

De este artículo se tomó la idea para probar en el método que proponemos, la existencia de un T-invariante en la RdP asociada a un conjunto de cláusulas, cuando el conjunto es insatisfacible.

2.4 El Método de Jeffrey, Lobo y Murata

Este método fue presentado en [Jeffrey et *al.*, 1996] y se aplica a la lógica de predicados de primer orden, aunque se restringe igual que los otros a cláusulas de Horn.

De acuerdo con este método, para construir la RdP N asociada al conjunto de cláusulas S, primeramente se dibuja un lugar por cada predicado en S, luego por cada cláusula en S se

dibuja una transición. Los arcos se dibujan de forma similar a como se hace en los métodos anteriores. La diferencia estriba en que cada arco se etiqueta con las variables o instancias que aparecen en el predicado, el cual está asociado al lugar del que sale o entra dicho arco.

Básicamente, para determinar si una proposición C en forma de cláusula (esto es, una consulta) es consecuencia de S, por cada predicado en C se marca con las variables o instancias correspondientes cada lugar asociado. Posteriormente, se realiza una evolución en N utilizando unificación. Un marcado vacío indica que C es consecuencia de S.

3. Construcción de una RdP a partir de un Conjunto de Cláusulas

Ya en la sección anterior enunciamos algunos métodos que determinan si un conjunto de cláusulas es insatisfacible, todos ellos basados en RdP. Cabe recalcar que los métodos que han probado su validez sólo lo han hecho para un grupo reducido de fórmulas: cláusulas de Horn. Aunque se han intentado ampliar tales métodos, buscando que no sólo se apliquen a cláusulas de Horn, hasta ahora tales intentos han sido fallidos.

El método que presentamos en este capítulo es una extensión del presentado por [Peterka and Murata, 1989]. El adelanto principal es que no se restringe a cláusulas de Horn ni a cláusulas de programa, se aplica a cualquier tipo de cláusulas de la lógica proposicional. Esto es bueno ya que no todo argumento lógico se puede expresar por medio de cláusulas de Horn. La única restricción que debemos imponer, es que una literal y su negación no pueden aparecer en una misma cláusula.

El método comienza asociando una RdP al conjunto de cláusulas dado. Enseguida, calcula los T-invariantes en la RdP y busca una secuencia de disparos σ que reproduzca el marcado vacío. Además, el vector característico asociado a σ debe ser uno de los T-invariantes encontrados o una combinación lineal de ellos.

A continuación definiremos como construir una RdP a partir de un conjunto de cláusulas y luego presentamos un ejemplo. De aquí en delante, a menos que se indique lo contrario, S denotará un conjunto de n cláusulas, es decir $S = \{ C_1, C_2, \ldots, C_n\}$. Además, $l_1, l_2,\ldots, l_k$ denotarán los átomos que aparecen en las cláusulas de S, esto es, $|S| = \{l_1, l_2,\ldots, l_k\}$.

La RdP N asociada con S se construye de la siguiente forma:

- Por cada átomo l_j, $1 \leq j \leq k$, se dibuja un lugar con etiqueta l_j.
- Por cada cláusula C_i, $1 \leq i \leq n$, se dibuja una transición con etiqueta t_i.
- Se dibuja un arco del lugar l_j a la transición t_i, si l_j aparece del lado izquierdo de $\rightarrow$ en C_i, es decir, si l_j corresponde a una literal negativa en C_i.
- Se dibuja un arco de la transición t_i al lugar l_j, si l_j aparece del lado derecho de $\rightarrow$ en C_i, es decir, l_j corresponde a una literal positiva en C_i.

Ejemplo 3.1:

Consideremos el conjunto de cláusulas $S = \{C_1, C_2, C_3, C_4\}$, donde $C_1 = <\rightarrow p, q>$, $C_2 = <p \rightarrow>$, $C_3 = <q \rightarrow r>$, $C_4 = <r \rightarrow>$.

En S ocurren cuatro cláusulas por lo que en la RdP N asociada a S aparecerán también cuatro transiciones, cuyas etiquetas son t_1, t_2, t_3 y t_4.

Además, $|S| = \{p, q, r\}$, por lo que en N aparecerán tres lugares con etiquetas p, q y r.

Un arco va de t_1 a p debido a que p aparece del lado derecho de $\rightarrow$, es decir, p corresponde a una literal positiva en C_1.

Un arco más va de p a t_2 debido a que p aparece del lado izquierdo de $\rightarrow$, es decir, p corresponde a una literal negativa en C_2.

Al terminar de trazar todos los arcos del modo en que se especificó arriba, completamos la RdP N asociada a S y obtenemos la gráfica que muestra la figura 3.1.

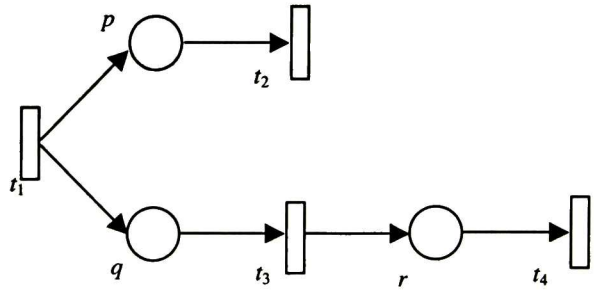


Figura 3.1 RdP asociada al conjunto $S = \{<\rightarrow p, q>, <p \rightarrow >, <q \rightarrow r>, <r \rightarrow>\}$

4. Validez del Método Propuesto

De nada sirve tener un método cuya validez no ha sido verificada. Por lo que en este capítulo enunciamos una proposición respecto al método propuesto. Tal proposición prueba la existencia de un T-invariante válido en caso de que el conjunto de cláusulas considerado sea insatisfacible.

Con el fin de realizar la prueba de la proposición mencionada anteriormente, enunciamos algunas definiciones y lemas. Primeramente, en la definición 3.1 especificamos como representamos una cláusula mediante un vector, en la definición 3.2 expresamos la forma de asociar un vector con una cláusula y así tener un vínculo entre la lógica proposicional y las RdP.

Luego en la sección 4.1, describimos la forma de construir un árbol vectorial asociado con un árbol de resolución, para después en la sección 4.2 determinar un T-invariante a partir de ese árbol vectorial. En base al árbol vectorial y al T-invariante encontrado, en la sección

4.3 describimos la forma de hallar una secuencia de transiciones y probamos, en el lema 3.2, que es realizable.

Después de lo anterior en el teorema 3.1, concluimos que si un conjunto de cláusulas S es insatisfacible, entonces existe un T-invariante válido en la RdP N asociada a S.

Iniciemos con las primeras definiciones de esta sección. Para definir a un árbol vectorial asociado a un árbol de resolución, primeramente debemos definir la forma de asociar una cláusula con un vector y luego cómo asociar un vector con una cláusula.

Definición 3.1 (*representación vectorial de una cláusula*)

Sea $C \in S$.

Decimos que el vector V es la *representación vectorial* de la cláusula C si se cumplen las dos condiciones siguientes:

- Es de tamaño k, donde k es el número de átomos en $|S|$.
- Para cada $i = 1, \ldots, k$, la i-ésima componente de V está asociada con l_i y su valor $V[i]$ es igual a:

  n si l_i aparece n veces del lado derecho de $\rightarrow$ en C (l_i determina una literal positiva)

  0 si l_i no aparece en C

  $-n$ si l_i aparece n veces a la izquierda de $\rightarrow$ en C (l_i determina una literal negativa)

Nota: Estamos asumiendo que una literal no puede aparecer tanto del lado izquierdo como del lado derecho de $\rightarrow$ en una misma cláusula, es decir, en forma positiva y negativa.

Dado un vector también podemos decir qué cláusula, respecto a un conjunto dado, está asociada con ese vector.

Definición 3.2: (*asociación de una cláusula con un vector*).

Sea V un vector de tamaño k.

Para cada $1 \leq i \leq k$, sea l_i el átomo asociado con la i-ésima componente de V.

La cláusula C *asociada a* V se construye de la siguiente forma:

- Si $V[i] < 0$, entonces l_i aparece $|V[i]|$ veces en C del lado izquierdo de $\rightarrow$.
- Si $V[i] = 0$, entonces l_i no aparece en C.
- Si $V[i] > 0$, entonces l_i aparece $V[i]$ veces en C del lado derecho de $\rightarrow$.

Ejemplo 3.2:

Sea $S = \{C_1, C_2, C_3, C_4\}$, donde $C_1 = <p \rightarrow>$, $C_2 = <q \rightarrow p>$, $C_3 = <r \rightarrow p, q>$ y $C_4 = <\rightarrow r>$.

Al ser tres los átomos que aparecen en las cláusulas de S (p, q y r), el tamaño de los vectores que representan tales cláusulas será también tres.

De acuerdo con la definición 3.1, la representación vectorial de C_1 es $V_1 = [-1, 0, 0]$, la de C_2 es $V_2 = [1, -1, 0]$, la de C_3 es $V_3 = [1, 1, -1]$ y la de C_4 es $V_4 = [0, 0, 1]$.

Ahora bien, dados los vectores $V_1 = [1, 0, 0]$, $V_2 = [1, 0, -1]$ y $V_3 = [1, 2, 0]$ y por la definición 3.2, la cláusula asociada a V_1 es $<\rightarrow p>$, a V_2 es $<r \rightarrow p>$ y a V_3 es $<\rightarrow p, q, q>$.

4.1 Arbol Vectorial Asociado a un Arbol de Resolución

Tal como lo hicimos con las cláusulas, ahora veamos como asociamos una representación vectorial a un árbol de prueba por resolución. Más adelante en la sección 4.2, describiremos como determinar un T-invariante a partir de un árbol vectorial y luego en la sección 4.3, como determinar una secuencia de transiciones a partir del mismo árbol vectorial.

Sea $V = \{V_1, V_2, ..., V_n\}$, donde V_i es la representación vectorial de la cláusula C_i en S, $1 \leq i \leq n$.

Sea T un árbol de prueba por resolución de S a partir de $< >$.

El procedimiento para construir el *árbol vectorial* asociado al árbol de resolución T, es el siguiente.

Paso 1. Cada hoja de T está asignada una cláusula $C_i \in S$. Sustituyamos cada una de ellas por su correspondiente representación vectorial.

Paso 2. Para todo nodo h cuyos nodos hijos h_l y h_r estén etiquetados con vectores y sean hojas de T, sustituyamos la etiqueta asignada al nodo h por la suma de los vectores asignados a h_l y a h_r.

Paso 3. Para todo nodo h de T cuya etiqueta es una cláusula y nodos hijos h_l y h_r estén etiquetados con vectores, determinamos la etiqueta del nodo h y realizamos algunas modificaciones al árbol de la siguiente forma:

Sea l_j la literal sobre la que se resolvió para obtener la cláusula asignada al nodo h de T.

Sean V_l y V_r las representaciones vectoriales de las cláusulas asignadas a h_l y a h_r, respectivamente.

Sea la j-ésima componente de V_l y de V_r la asociada con l_j, tal que $|V_l[j]| = a$ y $|V_r[j]| = b$.

Si $a \neq b$, entonces multiplicamos por una constante $c_l = b$ al vector asignado a h_l y por una constante $c_r = a$ al vector asignado en h_r, es decir, cambiamos el vector V_l asignado en h_l por $c_l * V_l$ y el vector V_r asignado en h_r por $c_r * V_r$. Después, sustituyamos la etiqueta del nodo h por la suma de $c_l * V_l$ con $c_r * V_r$, es decir, por la etiqueta $c_l * V_l + c_r * V_r$.

Si $a = b$, entonces sustituyamos la etiqueta del nodo h por la suma de V_l con V_r, es decir, por $V_l + V_r$.

En cualquiera de los dos casos anteriores, tendremos que la j-ésima componente del vector asignado a h es igual a 0.

Paso 4. Por último, para todo nodo interno h de T, en el cual añadimos una constante $c > 1$, realicemos lo siguiente:

1. Sean h_l y h_r los hijos del nodo h.

Sea c la constante que multiplica al vector asignado en h.

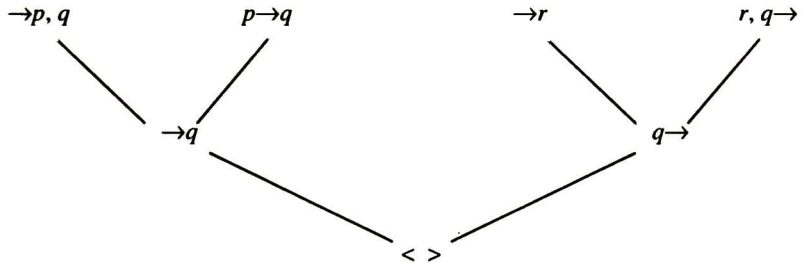
Sean c_l y c_r las constantes que afectan a h_l y a h_r, respectivamente (sino existen tales constantes, entonces se consideran iguales a 1).

Sustituyamos la etiqueta del nodo h_l por $c_l * c * V_l$ y la del nodo h_r por $c_r * c * V_r$.

Si h_l no es nodo hoja y $c_l > 1$, entonces vaya a 1 con $h = h_l$.

Si h_r no es nodo hoja y $c_r > 1$, entonces vaya a 1 con $h = h_r$.

Después de haber completado todas las fases anteriores, tenemos el árbol vectorial

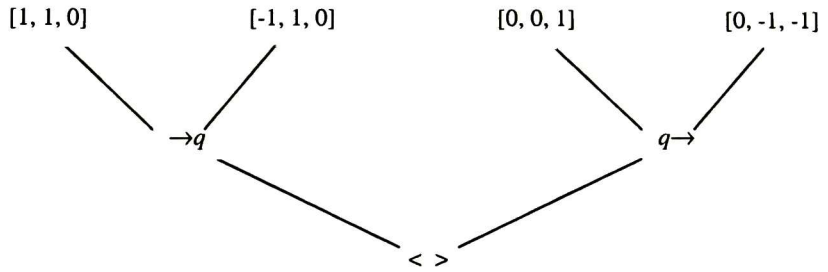


asociado al árbol de resolución T y el cual denominaremos T'

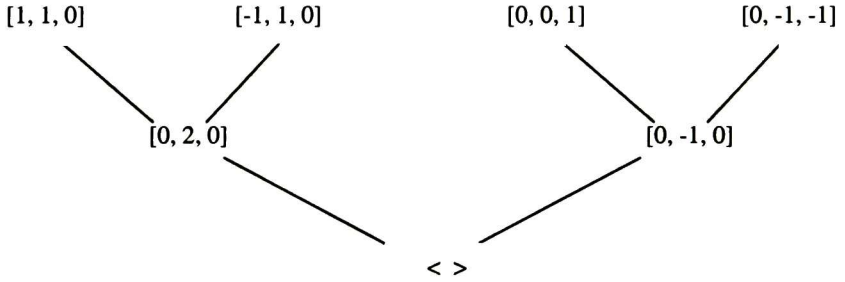
Ejemplo 3.2:

Sea $S = \{C_1, C_2, C_3, C_4\}$, donde $C_1 = <\rightarrow p, q>$, $C_2 = <p \rightarrow q>$, $C_3 = <\rightarrow r>$ y $C_4 = <r, q \rightarrow>$ con el siguiente árbol de resolución T.

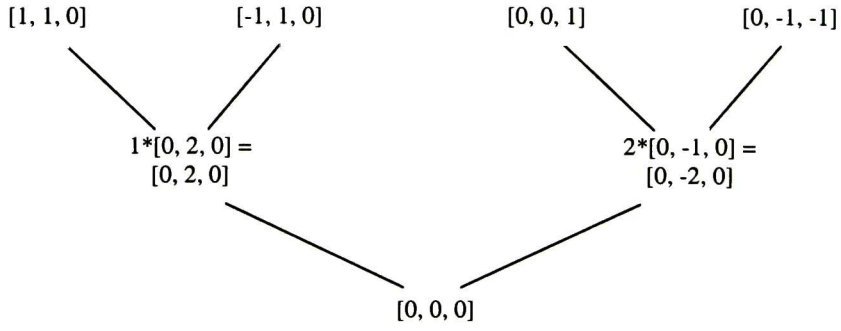
Paso 1.Comencemos a cambiar las etiquetas de las hojas de T por la representación vectorial de las cláusulas asignadas a ellas. Después de cambiar las etiquetas a toda hoja de T, obtenemos el árbol siguiente.



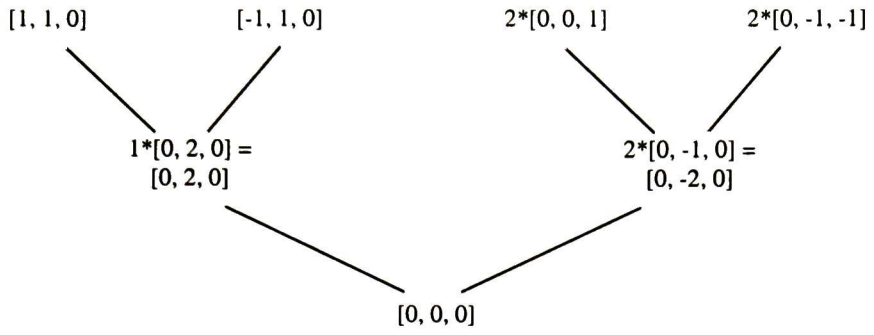
Paso 2. Sumando los vectores asignados en nodos que son hojas de T, obtenemos el vector que asignamos al nodo padre de tales nodos hojas.



Paso 3. La cláusula asignada al nodo raíz de T es producto de aplicar resolución sobre q a dos resolventes de cláusulas de S. En uno de los nodos hijos de la raíz de T (h_l) tenemos que la componente asociada a q es igual a 2 y en el otro nodo (h_r) es igual a -1. Por ello, tendremos dos constantes $c_l = 1$ y $c_r = 2$, las cuales multiplican a los vectores en h_r y en h_l, respectivamente, es decir, la etiqueta de la raíz de T es 1*[0, 2, 0]+2*[0, -1, 0] = [0, 0, 0].



Paso 4. Por último, debemos de actualizar algunas constantes. Por ejemplo, se introdujo una constante 2 en un vector asignado a un nodo de T' por lo que introducimos una constante 2 que multiplica a cada vector asignado en sus nodos hijos. Obtenemos el árbol vectorial T' asociado al de resolución T que se muestra a continuación.



Después de haber construido el árbol vectorial T' asociado al de resolución T, podemos determinar a partir de T', un T-invariante para N.

4.2 T-invariante Determinado a partir de un Arbol Vectorial

A continuación describimos el procedimiento para determinar un T-invariante a partir de un árbol vectorial asociado a uno de resolución. Luego en la sección 4.3, encontramos una secuencia de transiciones cuyo vector característico asociado corresponde a este T-invariante.

Sea T un árbol de prueba por resolución de S a partir de $< >$.

Sea T' el árbol vectorial asociado a T.

Sean $a_1, a_2, \ldots, a_m$, para $m \geq 0$, las constantes que afectan a la representación vectorial V_i de la cláusula C_i. Considerando únicamente a los vectores asignados a hojas de T'

La i-ésima componente del vector de coeficientes V_c es igual a:

0 si $m = 0$, es decir, si V_i no aparece en las etiquetas asignadas a hojas en T'

$a_1+a_2+\ldots+a_m$ si $m>0$, es decir, si V_i está asignado en m hojas de T'

Al realizar lo anterior obtenemos un vector de coeficientes y con lo cual:

$$V_c[1]*V_1 + V_c[2]*V_2 + \quad + V_c[n]*V_n = \mathbf{0} \text{ (vector cero)}$$

En la RdP N tenemos una matriz de incidencia A, la cual está compuesta de vectores columna que corresponden a los vectores V_i en V, es decir, tenemos la matriz de incidencia A de N:

$$A = \begin{bmatrix} V_1[1] & V_2[1] & V_n[1] \\ V_1[2] & V_2[2] & V_n[2] \\ & & \\ V_1[k] & V_2[k] & V_n[k] \end{bmatrix}$$

Por ello, si multiplicamos A por el vector de coeficientes V_c sucede que $A*V_c = \mathbf{0}$.

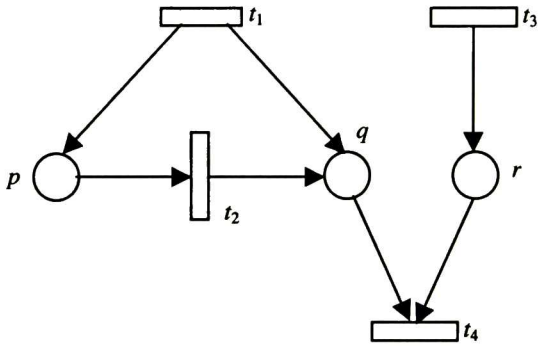
Con lo que hemos encontrado un *T-invariante* de N.

Ejemplo 3.4:

Sea $V = \{V_1 = [1, 1, 0], V_2 = [-1, 1, 0], V_3 = [0, 0, 1], V_4 = [0, -1, -1]\}$ las representaciones vectoriales de las cláusulas en S, del ejemplo 3.3.

Las constantes de los vectores V_1 y V_2 son iguales a 1 y como tales vectores aparecen sólo una vez en las hojas de T' tenemos que $V_c[1] = 1$ y $V_c[2] = 1$. Los vectores V_3 y V_4, también aparecen una sola vez en T' pero las constantes que los afectan son iguales a 2, por tanto $V_c[3] = 2$ y $V_c[4] = 2$. Por lo anterior, $V_c = [1, 1, 2, 2]^T$

La RdP N asociada a $S = \{<\rightarrow p, q>, <p \rightarrow q>, <\rightarrow r>, <r, q \rightarrow>\}$ es:



La matriz de incidencia A de la RdP N es:

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

En esta matriz cada vector columna corresponde a la etiqueta asignada a una hoja de T' Multiplicando la matriz A por el vector de coeficientes V_c obtenemos **0**. Con lo que V_c es un T-invariante de N.

Hemos expuesto en la sección 4.2, un procedimiento para encontrar un T-invariante en N cuando S es insatisfacible. Establecemos lo anterior, en el siguiente lema.

Lema 3.1: (*existencia de un T-invariante*)

Si S es insatisfacible, entonces existe en N un T-invariante.

Prueba:

Si S es insatisfacible, entonces existe un árbol de prueba por resolución T de S a partir de $< >$.

En la sección 4.1 describimos cómo asociar un árbol vectorial a T y en la sección 4.2 expresamos la forma de encontrar un T-invariante. Con lo cual el lema queda establecido.

4.3 Secuencia de Transiciones Construida a partir de un Arbol Vectorial

Hasta el momento hemos encontrado un T-invariante en N. Ahora, tratemos de encontrar una secuencia cuyo vector característico asociado, sea el vector de constantes V_c, es decir, ese T-invariante.

Sea T el árbol de prueba por resolución de S a partir de $< >$.

Sea T' el árbol vectorial asociado a T.

Primero comenzamos por la raíz de T' y le asignamos un nombre a cada vector en nodos de T'.

Sea h la raíz de T'

Sean h_l y h_r los nodos hijos de h.

Sean V_l y V_r los vectores asignados a h_l y a h_r, respectivamente, tal que sólo una componente de V_l es diferente de cero y positiva y, sólo una componente de V_l es diferente de cero y negativa.

Si V_l es una hoja, entonces $l \leq n$, de lo contrario $l = n+1$.

Si V_r es una hoja, entonces $r \leq n$, de lo contrario $r = n+2$.

Además, a los coeficientes que afecten a V_l y a V_r los denominaremos c_l y c_r, respectivamente.

En la suma s_0 siempre consideremos el orden $\mathbf{0} = V_l \ldots V_l + V_r \ldots V_r$, tal que V_l ocurre c_{n+1} veces y V_r aparece c_{n+2} veces (frecuentemente simplificaremos en la suma y escribiremos por ejemplo, $c_{n+1}V_l + c_{n+2}V_r$, en lugar de escribirlo de la forma anterior). La *secuencia de transiciones* σ_0 asociada a la suma s_0, la construimos al sustituir cada vector por su correspondiente transición asociada y al eliminar los operadores presentes, es decir, la secuencia de transiciones es $\sigma_0 = t_l \ldots t_l t_r \ldots t_r$.

Ahora, veamos lo que sucede con vectores en nodos de T' diferentes a la raíz.

Sea $c_m V_m$ un vector en la suma que hemos obtenido hasta el momento.

Sea h_m el nodo en el cual esta asignado el vector $c_m V_m$.

Sean h_j y h_i los nodos hijos de h_m.

Sean $c_j V_j$ y $c_i V_i$ los vectores asignados a h_j y a h_i, respectivamente, tal que una componente negativa de $c_j V_j$ es igual a cero en $c_m V_m$ y una componente positiva de $c_i V_i$ es igual a cero en $c_m V_m$.

En la suma siempre sustituyamos a $c_m V_m$ por $V_i \ldots V_i + V \ldots V_j$, donde V_i aparece c_i veces y V_j aparece c_j veces.

Si V_s no es una hoja y q es el subíndice mayor introducido hasta el momento, entonces el valor de s es $q+1$. Si V_s es una hoja, entonces $s \in \{1, 2, \ldots, n\}$. Lo anterior, primero para $s = j$ y luego para $s = i$.

Si hemos realizado lo anterior k veces con nodos diferentes a la raíz de T', entonces la *secuencia de transiciones* asociada a la suma s_k es σ_k. Obtenemos esta secuencia sustituyendo cada vector por su correspondiente transición asociada y eliminando los operadores presentes en la suma.

Hemos encontrado una secuencia y esperamos que sea realizable. Posteriormente en la proposición 4.1 probaremos este hecho, por lo pronto veamos un ejemplo que ilustre el procedimiento anterior.

Ejemplo 3.5:

Consideremos el árbol T' del ejemplo 3.3, cuyo nodo raíz es el vector cero.

En los vectores asignados a los nodos hijos de la raíz de T' en la cual está asignado **0,** tenemos que uno de ellos sólo una de sus componentes es diferente de cero y positiva, a este vector lo denominaremos V_5 ya que S contiene 4 cláusulas y V_5 no es una hoja. En el otro vector que designaremos V_6 debido a que tampoco es una hoja, sólo una de sus componentes es diferente de cero y negativa. La constante que afecta a V_5 es 1 y a V_6 es 2 por ello:

$\mathbf{0} = V_5 + V_6 V_6$. La secuencia de transiciones asociada a esta suma es $\sigma_0 = t_5 t_6 t_6$

Los hijos del nodo en el que está asignado el vector V_5 son nodos hoja, por ello los denominaremos V_1 y V_2, donde V_1 y V_2 son las representaciones vectoriales de C_1 y C_2, respectivamente.

Sabemos que el vector V_5 es la suma de los vectores V_1 y V_2, por lo que sustituimos en la suma a V_5 por $V_1 + V_2$. Con lo que $\mathbf{0} = V_1 + V_2 + V_6 V_6$ y su secuencia de transiciones asociada es $\sigma_1 = t_1 t_2 t_6 t_6$.

Como V_6 es la suma de dos vectores asignados en nodos hojas de T' a tales vectores los denominaremos V_3 y V_4, donde V_3 y V_4 son la representación vectorial de C_3 y C_4, respectivamente. Por tanto $\mathbf{0} = V_1 + V_2 + V_3 V_3 + V_4 V_4$ y $\sigma_2 = t_1 t_2 t_3 t_3 t_4 t_4$.

Con lo realizado, hemos encontrado una secuencia de transiciones a partir de un árbol vectorial.

Para probar que la secuencia de transiciones determinada a partir de un árbol vectorial es realizable, definimos la manera en que una secuencia de transiciones determina una RdP.

Definición 3.3: (*RdP determinada por una secuencia de transiciones*)

Sea T un árbol de resolución para S a partir de < >.

Sea T' el árbol vectorial asociado a T.

Sea σ_k la secuencia de transiciones construida a partir de T'

Sean $t_{a1}, t_{a2}, \ldots, t_{am}$ las transiciones que aparecen en σ_k, cuyos vectores correspondientes son $V_{a1}, V_{a2}, \ldots, V_{am}$, respectivamente.

Sea A la matriz cuyas columnas son $V_{a1}, V_{a2}, \ldots, V_{am}$.

Sea A' la matriz construida a partir de A, eliminando de A todo renglón que contenga sólo7 valores iguales a cero.

La RdP N_k determinada por σ_k es la RdP cuya matriz de incidencia es A'

Ejemplo 3.6:

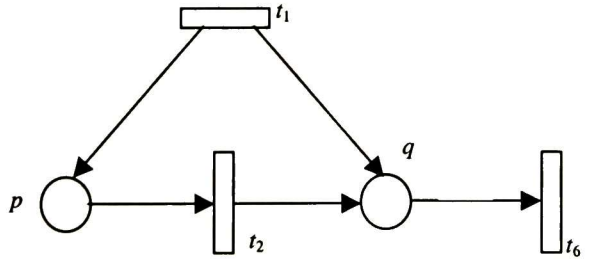
Considere la secuencia de transiciones $\sigma_1 = t_1\, t_2\, t_6\, t_6$ del ejemplo 3.5. La siguiente matriz A contiene como vectores columna a V_1, a V_2 y a V_6:

$$A = \begin{array}{c} \\ p \\ q \\ r \end{array} \begin{array}{c} \begin{array}{ccc} V_1 & V_2 & V_6 \end{array} \\ \begin{bmatrix} 1 & -1 & 0 \\ 1 & 1 & -1 \\ 0 & 0 & 0 \end{bmatrix} \end{array}$$

De A eliminamos el tercer renglón y obtenemos la matriz de incidencia A':

$$A' = \begin{array}{c} \\ p \\ q \end{array} \begin{array}{c} \begin{array}{ccc} t_1 & t_2 & t_6 \end{array} \\ \begin{bmatrix} 1 & -1 & 0 \\ 1 & 1 & -1 \end{bmatrix} \end{array}$$

La matriz de incidencia de la RdP asociada a N_1 es A', cuya representación gráfica es:



La siguiente definición introduce una notación para determinar el valor de la componente asociada con un lugar en un vector columna de la matriz de incidencia A de N.

Definición 3.4:

Sea A la matriz de incidencia de una RdP N.

Sea $\bar{t}_i$ la i-ésima columna de A.

Sea p_k el lugar asociado con el k-ésimo renglón de A.

$\vec{t}_i$ [p_k] denota el valor de la k-esima componente de $\vec{t}_i$

Ya habíamos determinado en la sección 4.3, una secuencia de transiciones para una RdP, nos faltaba probar que dicha secuencia fuera realizable.

Lema 3.2:

Sea S un conjunto de x cláusulas.

Sea T un árbol de prueba por resolución para S a partir de < >.

Sea T' el árbol vectorial asociado a T.

Sea σ_k la secuencia de transiciones construida a partir de T'

Sea N_k la RdP determinada por σ_k. Entonces,

la secuencia σ_k es realizable en N_k.

Prueba:

La demostración se hace por inducción en k:

Caso base:

Sean s_0 la suma $\mathbf{0} = V_{n+1}\ldots V_{n+1}+V_{n+2}\ldots V_{n+2}$ y la secuencia de transiciones $\sigma_0 = t_{n+1}\ldots t_{n+1}t_{n+2}\ldots t_{n+2}$, donde V_{n+1} aparece a_{n+1} veces en s_0 y t_{n+1} aparece a_{n+1} veces en σ_0. De forma similar, V_{n+2} aparece a_{n+2} veces en s_0 y t_{n+2} aparece a_{n+2} veces en σ_0.

Sea N_0 la RdP determinada por σ_0.

Probemos que σ_0 es realizable en N_0.

Sabemos que:

en V_{n+1} sólo una de sus componentes es igual a a, $a \in \mathbf{N}$, digamos que V_{n+1} $[c_{n+1}] = a$.

en V_{n+2} sólo una de sus componentes es igual a $-b$, $b \in \mathbf{N}$, digamos que V_{n+2} $[c_{n+2}] = -b$.

$a_{n+1}V_{n+1} = a_{n+1}\overline{t_{n+1}}$ y $a_{n+2}V_{n+2} = a_{n+2}\overline{t_{n+2}}$ ya que $V_{n+1} = \overline{t_{n+1}}$ y $V_{n+2} = \overline{t_{n+2}}$ y $|a_{n+1}a| = |a_{n+2}b|$.

Sea p_j el único lugar de salida de t_{n+1} y el único lugar de entrada a t_{n+2}.

$\beta(\overline{t_{n+1}}, p_j) = c_{n+1} = a$ y $\alpha(p_j, t_{n+2}) = c_{n+2} = b$.

La secuencia σ_0 es realizable ya que t_{n+1} es una transición fuente que podemos dispararla a_{n+1} veces. Al primer disparo de t_{n+1} se depositan a marcas con referencia $<n+1, 1>$, al segundo disparo de t_{n+1} se depositan a marcas con referencia $<n+1, 2>$, ..., al a_{n+1}-ésimo disparo de t_{n+1} se depositan a marcas con referencia $<n+1, a_{n+1}>$ en el lugar p_j. Como b marcas habilitan a t_{n+2}, entonces podemos dispararla a_{n+2} veces con esas $a_{n+1}c_{n+1}$ marcas en p_j (ya que $a_{n+1}\beta(t_{n+1}, p_j) = a_{n+2}\alpha(p_j, t_{n+2})$).

Por tanto σ_0 es realizable en N_0.

Paso de inducción:

Sea s_j la suma construida a partir de T', para $j < k$.

Sea σ_j la secuencia determinada por s_j.

Suponga que σ_j es realizable en la RdP N_j.

Probemos que la secuencia σ_{j+1} determinada por s_{j+1} es realizable en la RdP N_{j+1} determinada por σ_{j+1}.

Sea V_i el vector en s_j el cual en s_{j+1} lo sustituimos por V_x y por V_y, es decir, sustituimos a a_iV_i en s_j por $a_xV_x+a_yV_y$ para obtener s_{j+1}. Por lo anterior en σ_{j+1} sustituimos a a_it_i por $t_x \ldots t_x t_y \ldots t_y$, tal que t_x aparece a_x veces y t_y aparece a_y veces.

Sabemos que en N_j:

$|V_i[p_s]| = \alpha(p_s, t_i)$ para todo $p_s \in {}^\bullet t_i$

$V_i[p_s] = \beta(t_i, p_s)$ para todo $p_s \in t_i^\bullet$

Sea $E = {}^\bullet t_i$, conjunto de lugares de entrada a t_i.

Sea $O = t_i^\bullet$, conjunto de lugares de salida de t_i.

Sea $E_1 = {}^\bullet t_x$, conjunto de lugares de entrada a t_x.

Sea $E_2 = {}^\bullet t_y$, conjunto de lugares de entrada a t_y.

Sea $O_1 = t_x^\bullet$, conjunto de lugares de salida de t_x.

Sea $O_2 = t_y^\bullet$, conjunto de lugares de salida de t_y.

Sea p_q el lugar de salida de t_x y de entrada a t_y, donde $p_q \notin E$ y $p_q \notin O$.

Sean M_1, M_2, ..., M_r los multiconjuntos de marcas que habilitan a t_i para su disparo a_i veces. Al primer disparo de t_i se eliminan las marcas en M_1, al segundo disparo de t_i se eliminan las marcas de M_2, ... , al r-ésimo disparo de t_i se eliminan las marcas de M_r.

Sabemos que $E = E_1 \cup E_2 - \{p_q\}$

Como $\alpha(p_s, t_i) = \alpha(p_s, t_x) + \alpha(p_s, t_y)$, entonces t_x debe estar habilitada para su disparo a_x veces ya que $E_1 \subset E$.

Después de disparar a_x veces t_x podemos disparar a_y veces t_y, ya que $E_2 - \{p_q\} \subset E$. Si M_i es el conjunto de marcas que utilizó t_i en uno de sus disparos, entonces las marcas que elimina t_x y t_y al disparar una vez cada una de ellas y en ese orden, es igual a M_i junto con las marcas en p_q que añadió t_x a su disparo.

Debido a que $a_i * \alpha(p_s, t_i) = a_x * \alpha(p_s, t_x) + a_y * \alpha(p_s, t_y)$ al disparo de t_i, a_i veces, se eliminan el mismo número de marcas que al disparar t_x, a_x veces y luego de disparar t_y, a_y veces.

Falta verificar que t_x y t_y añaden el mismo número de marcas que t_i y que el cambio de referencia no impide que σ_{j+1} sea realizable.

Como $a_i * \beta(t_i, p_s) = a_m * \beta(t_x, p_s) + a_y * \beta(t_y, p_s)$, entonces se añade el mismo número de marcas. Ahora, analicemos que pasa con la referencia de esas marcas.

Al disparo de t_i obtenemos el multiconjunto de marcas $M_i = <i, I>$, tal que en M_i se encuentran las marcas que añadió t_i a su disparo y las marcas que cambiaron su referencia por ser iguales a las que habilitaron a t_i, también para su disparo.

Al disparo de t_x y luego del disparo de t_y obtenemos el multiconjunto de marcas $M_y = <y, I>$, tal que en M_y se encuentran las marcas que añadieron t_x y t_y a su disparo, además de las marcas que cambiaron su referencia por ser iguales a las que habilitaron a t_x primero, y luego a t_y para sus disparos.

El número de elementos de M_i es igual al número de elementos de M_y, ya que al disparo de t_i se añade el mismo número de marcas que después del disparo de t_x primero y luego del disparo de t_y. Además, las marcas que cambiaron su referencia al disparo de t_i, son las mismas que cambiaron su referencia después del disparo de t_y.

Cómo lo único que ha cambiado es la referencia de las marcas, es decir, las marcas que en σ_j después del disparo de t_i tenían referencia $<i, I>$, en σ_{j+1} tienen referencia $<y, I>$ después del disparo de t_y. Podemos decir que σ_{j+1} es realizable ya que las transiciones están habilitadas cuando sus lugares de entrada contienen marcas con referencia distinta entre sí, no importando cual sea su referencia.

Así la secuencia σ_{j+1} es realizable en N_{j+1}.

Con todo lo enunciado anteriormente, podemos probar la completud del método propuesto.

Teorema 3.1: Sí S es insatisfacible, entonces existe un T-invariante válido en N.

Prueba:

Por el lema 3.1 podemos decir que existe un T-invariante en N y por el lema 3.2 que tal T-invariante es válido.

Por tanto, si S es insatisfacible, entonces existe un T-invariante válido en N. Con lo cual hemos probado la solidez del método propuesto.

5. Completud del Método Propuesto

En esta sección probaremos que cuando tengamos un T-invariante válido en la RdP N, entonces podemos decir que el conjunto de cláusulas S asociado a N es insatisfacible. Con esto habremos probado la completud del método propuesto.

Primeramente en la definición 3.5 describimos cómo asociar una cláusula a una marca. Luego en la definición 3.6, describimos cómo asociar una cláusula con el disparo de una transición. Posteriormente, introducimos dos lemas, uno para establecer que la cláusula asociada con el disparo de una transición fuente es elemento de S y otro, en el cual establecemos que con el disparo de una transición no fuente está asociada una cláusula, la cual es la hiper-resolvente positiva de electrones que son las cláusulas asociadas con las marcas que habilitaron tal transición y núcleo, que es la cláusula que determinó la transición que se disparó.

Por último, en el teorema 3.2 establecemos y probamos la completud del método propuesto.

Veamos primeramente cómo le asociamos una cláusula a una marca que se encuentra en lugares de la RdP N.

Definición 3.5 (*asociación de una cláusula con una marca*)

Sea $M = <D,I>$ la marca en los lugares $p_1, p_2, \ldots, p_i$, $i \geq 0$, con etiquetas $l_1, l_2, \ldots, l_i$, respectivamente.

Sean $a_1, a_2, \ldots, a_i$ la cantidad de ocurrencias de la marca M en l_1, en l_2, ..., en l_i, respectivamente.

La cláusula C_M asociada a M está formada por los átomos l_1, l_2,..., l_i, esto es, C_M = $<\rightarrow l_1,...l_1, l_2,...l_2, ..., l_i,...l_i>$, tal que l_1 ocurre a_1 veces, l_2 ocurre a_2 veces, ..., l_i ocurre a_i veces.

La cláusula C_M contiene átomos que determinan sólo literales positivas.

Ahora veamos, cómo asociar una cláusula con el disparo de una transición de N.

Definición 3.6 (*asociación de una cláusula con el disparo de una transición*)

Sea $C_m = < l_1, l_2, ..., l_i \rightarrow l_{i+1}, l_{i+2},..., l_{i+j}> \in S$, tal que $i, j \geq 0$, pero no ambas iguales a 0.

Sea t_m la transición que determinó C_m.

Sea $M = <m, I>$, $I \in \{1, 2, ..., r\}$, $r > 0$:

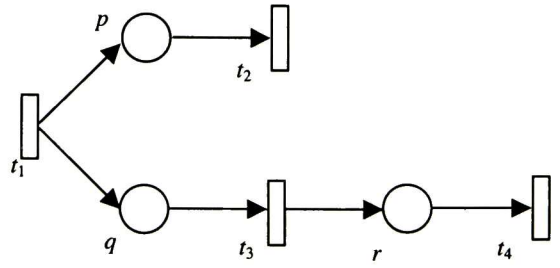
- la marca que añade a su disparo t_m a todo $p \in t_m\bullet$; si $t_m\bullet \neq \varnothing$ ó
- la referencia a la que cambió M_1, M_2, ... y M_i; si $t_m\bullet = \varnothing$ y existen tales marcas en el marcado alcanzado, ó
- la marca que se hubiera añadido al disparo de t_m a todo $p \in t_m\bullet$; si al disparo de t_m ya no quedaron marcas M_1, M_2, ... , M_i que cambiaran su referencia a M y $t_m\bullet = \varnothing$.

La cláusula R_m asociada con el disparo de t_m es la cláusula asociada a M.

Un ejemplo donde ilustramos las definiciones anteriores es el siguiente.

Ejemplo 3.7:

Sea $S = \{C_1, C_2, C_3, C_4\}$, donde $C_1 = <\rightarrow p, q>$, $C_2 = <p \rightarrow>$, $C_3 = <q \rightarrow r>$ y $C_4 = <r \rightarrow>$. Cuya correspondiente RdP asociada N, es la que mostramos a continuación.



Si comenzamos con un marcado vacío K_0 = [∅, ∅, ∅] y si luego disparamos la transición t_1, entonces obtenemos el marcado K_1 = [{<1, 1>}, {<1, 1>}, ∅]. La cláusula asociada con el disparo de t_1 es la asociada con la marca <1, 1>, es decir, la cláusula $C_{<1,1>}$ = <→*p*, *q*>, ya que la marca <1, 1> se encuentra en el lugar *p* y en el lugar *q*, con número de ocurrencias de cada una igual a 1. Además, observemos que $C_{<1,1>}$ es un elemento de *S*.

Si en K_1 disparamos t_2, entonces obtenemos el marcado K_1 = [∅, {<2, 1>}, ∅]. La cláusula asociada con el disparo de t_2 es la asociada con la marca <2, 1>, esto es, la cláusula $C_{<2,1>}$ = <→*q*>, esto debido a que la marca <2, 1> se encuentra en el lugar *q*.

Si disparamos t_3 en K_1, entonces obtenemos el marcado K_2 = [∅, ∅, {<3, 1>}]. La cláusula asociada con el disparo de t_3 es la asociada con la marca <3, 1>, esto es, la cláusula $C_{<3,1>}$ = <→*r*>, esto debido a que la marca <3, 1> se encuentra en el lugar *r*.

Por último, si disparamos t_4 en K_2, entonces obtenemos el marcado K_5 = [∅, ∅, ∅]. La cláusula asociada con el disparo de t_4 es la asociada con la marca <4, 1>, esto es, la cláusula $C_{<4,1>}$ = < >, esto debido a que en ningún lugar aparece la marca <4, 1>.

Tal como los vimos en el ejemplo anterior, en el siguiente lema establecemos que la cláusula asociada con el disparo de una transición fuente es una cláusula perteneciente a *S*.

Lema 3.3: (*la cláusula asociada con el disparo de una transición fuente es un elemento de S*)

Sea C_m = <→l_1, l_2,..., l_j>, $j > 0$, un elemento de *S*.

Sea t_m la transición que determinó C_m.

La cláusula asociada con el disparo de t_m es $C_m \in S$.

Prueba:

Sean p_1, p_2, ..., p_j, los lugares de salida de t_m, con etiquetas l_1, l_2,..., l_j, respectivamente.

Sea M = <*m*, *I*>, $I \in \{1, 2, \ldots, r\}$, $r > 0$, la marca que añade a su disparo t_m a todo $p_i \in t_m\bullet$, para $1 \leq i \leq j$.

Por definición, la cláusula asociada con el disparo de t_m es la asociada a *M*. La cual también por definición, es igual a <→l_1, l_2,..., l_j>, ya que l_i es la etiqueta del lugar p_i y $p_i \in t_m\bullet$. Además l_i sólo aparece una vez en C_m, ya que por la regla de disparo se añade una marca a cada $p_i \in t_m\bullet$.

Por lo tanto, la cláusula asociada con el disparo de t_m es $C_m \in S$.

En el lema siguiente establecemos que la cláusula asociada con el disparo de una transición fuente, es la hiper-resolvente positiva de electrones que son las cláusulas asociadas con las marcas que habilitaron tal transición y núcleo que es la cláusula que determinó la transición que disparamos.

Lema 3.4:

Sean $C_m \in S$ y t_m la transición que determinó C_m.

Sean $p_1, p_2, \ldots, p_i$, $i > 0$, los lugares de entrada a t_m, con referencias $l_1, l_2, \ldots, l_i$, respectivamente.

Sean $p_{i+1}, p_{i+2}, \ldots, p_{i+j}$, $j \geq 0$, los lugares de salida de t_m, cuyas referencias son $l_{i+1}, l_{i+2}, \ldots, l_{i+j}$, respectivamente.

Sean $M_1, M_2, \ldots, M_i$, las marcas que habilitan a t_m, tal que M_1 se encuentra en p_1, M_2 se encuentra en $p_2, \ldots, M_i$ se encuentra en p_i.

Sean $C_{m1} = <\rightarrow\varphi_1, l_1>$, $C_{m2} = <\rightarrow\varphi_2, l_2>, \ldots, C_{mi} = <\rightarrow\varphi_{mi}, l_{mi}>$ las cláusulas asociadas con las marcas $M_1, M_2, \ldots, M_i$, respectivamente. Entonces, la cláusula asociada con el disparo de t_m es la *hiper*-resolvente positiva de electrones $C_{m1}, C_{m2}, \ldots, C_{mi}$ y núcleo C_m.

Prueba:

Sea $M = <m, I>$, $I \in \{1, 2, \ldots, r\}$, $r > 0$:

- la marca que añade a su disparo t_m a todo $p \in t_m\bullet$; si $t_m\bullet \neq \varnothing$ ó
- la referencia a la que cambió $M_1, M_2, \ldots$ y M_i; si $t_m\bullet = \varnothing$ y existen tales marcas en el marcado alcanzado, ó
- la marca que se hubiera añadido al disparo de t_m a todo $p \in t_m\bullet$; si al disparo de t_m ya no quedaron marcas $M_1, M_2, \ldots, M_i$ que cambiaran su referencia a M y $t_m\bullet = \varnothing$.

Sea R_m la cláusula asociada con el disparo de t_m.

Por definición, la cláusula R_m es la cláusula asociada a M. Determinemos tal cláusula.

La marca M_1 antes del disparo de t_m se encontraba en los lugares etiquetados por los átomos en φ_1 y por l_1. Luego del disparo de t_m se cambió la referencia de la marca M_1 por M y se eliminó la marca en l_1.

La marca M_2 antes del disparo de t_m se encontraba en los lugares etiquetados por los átomos en φ_2 y por l_2. Luego del disparo de t_m se cambió la referencia de la marca M_2 por M y se eliminó la marca en l_2.

...

La marca M_i antes del disparo de t_m se encontraba en los lugares etiquetados por los átomos en φ_{mi} y por l_i. Luego del disparo de t_m se cambió la referencia de la marca M_i por M y se eliminó la marca en l_i.

Por lo anterior, la cláusula R_m asociada a M contiene a los átomos en φ_1, en φ_2,... y en φ_{mi}. Además, también por definición, R_m contiene a los átomos l_{i+1}, l_{i+2},..., l_{i+j}, ya que p_{i+1}, p_{i+2},..., p_{i+j} son los lugares de salida de t_m.

Por lo que R_m es igual a $<\rightarrow l_{i+1}, l_{i+2}, \ldots l_{i+j}, \varphi_1, \varphi_2, \varphi_{mi}>$, ya que por definición los átomos que aparecen en R_m determinan literales positivas.

Ahora determinemos la hiper-resolvente positiva del núcleo C_m y electrones C_{m1}, C_{m2},..., C_{mi}.

$$\frac{<l_1, l_2, \ldots, l_i \rightarrow l_{i+1}, l_{i+2}, \ldots, l_{i+j}> \quad <\rightarrow\varphi_1, l_1> \quad <\rightarrow\varphi_2, l_2> \quad ,\ldots, \quad <\rightarrow\varphi_{mi}, l_{mi}>}{<\rightarrow l_{i+1}, l_{i+2}, \ldots, l_{i+j}, \varphi_1, \varphi_2, \varphi_{mi}>}$$

la cual es igual a R_m.

Por lo tanto, la cláusula asociada con el disparo de t_m es la hiper-resolvente positiva de electrones C_{m1}, C_{m2},..., C_{mi} y núcleo C_m.

Si disparamos una transición no fuente que no tiene lugares de salida y si el marcado alcanzado es el marcado vacío, entonces la cláusula asociada con la transición que disparamos es la cláusula vacía. Esto es un resultado inmediato del lema anterior y lo establecemos en el corolario siguiente.

Corolario 3.1:

Sea $C_m \in S$.

Sea t_m la transición que determinó C_m.

Sean p_1, p_2,..., p_i, $i > 0$, los lugares de entrada a t_m, con referencias l_1, l_2, ..., l_i, respectivamente.

Sea $K = [M_1, M_2, \ldots, M_i, M_{i+1}, M_{i+2}, \ldots, M_k]$ el marcado actual de N, tal que M_{i+1}, M_{i+2}, ..., $M_k = \varnothing$.

Sean $C_{m1} = <\rightarrow l_1>$, $C_{m2} = <\rightarrow l_2>$, ..., $C_{mi} = <\rightarrow l_{mi}>$ las cláusulas asociadas con las marcas M_1, M_2, ..., M_i, respectivamente.

La cláusula asociada con el disparo de t_m es la cláusula vacía.

Prueba:

La prueba de este corolario es consecuencia del lema anterior, ya que de acuerdo a este lema la cláusula asociada con el disparo de t_m es la hiper-resolvente de electrones C_{m1}, $C_{m2},\ldots, C_{mi}$ y núcleo C_m, es decir, la cláusula vacía, debido a que:

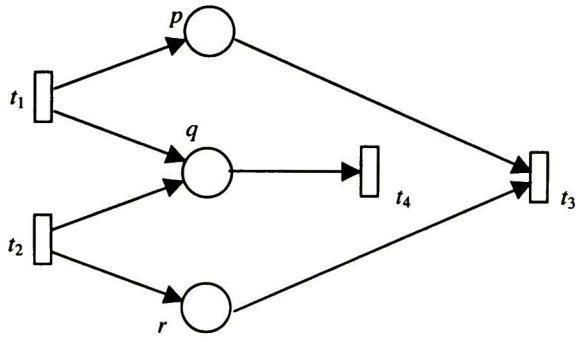
$$\frac{<l_1,l_2,\ldots,l_i\rightarrow> \quad <\rightarrow l_1> \quad <\rightarrow l_2> \quad ,\ldots, \quad <\rightarrow l_{mi}>}{<\ >}$$

Por tanto, la cláusula asociada con el disparo de t_m es la cláusula vacía.

Veamos un ejemplo que ilustre el corolario y lemas anteriores.

Ejemplo 3.8:

Sea $S = \{C_1, C_2, C_3, C_4\}$, donde $C_1 = <\rightarrow p, q>$, $C_2 = <\rightarrow q, r>$, $C_3 = <p, r\rightarrow>$ y $C_4 = <q\rightarrow>$. Cuya RdP asociada N, mostramos a continuación.



Si comenzamos con el marcado vacío $K_0 = [\varnothing, \varnothing, \varnothing]$ y si luego realizamos la secuencia de disparos $\sigma = t_1t_2$, entonces obtenemos el marcado $K_2 = [\{<1, 1>\}, \{<1, 1>, <2, 1>\}, \{<2, 1>\}]$. La cláusula asociada con el disparo de t_1 es la asociada con la marca <1, 1>, es decir, la cláusula $C_{<1, 1>} = <\rightarrow p, q>$. De forma similar, la cláusula asociada con el disparo de t_2 es la asociada con la marca <2, 1>, es decir, la cláusula $C_{<2, 1>} = <\rightarrow q, r>$. Tal como lo establecimos en el lema 3.3, las cláusulas asociadas con el disparo de t_1 y de t_2, $C_{<1, 1>}$ y $C_{<2, 1>}$, son elementos de S.

En el marcado K_2 si disparamos t_3, entonces obtenemos el marcado $K_3 = [\varnothing, \{<3, 1>, <3, 1>\}, \varnothing]$. La cláusula asociada con el disparo de t_3 es la asociada con la marca <3, 1>, es decir, la cláusula $C_{<3, 1>} = <\rightarrow q, q>$. La cual, tal como lo establecimos en el lema 3.4, es la hiper-resolvente positiva de electrones $C_{<1, 1>}$, $C_{<2, 1>}$ y núcleo C_3, donde C_3 es la cláusula que determinó t_3.

Si en el marcado K_3 disparamos t_4, entonces obtenemos el marcado $K_4 = [\varnothing, \{<4, 1>\}, \varnothing]$. La cláusula asociada con el disparo de t_4 es la asociada con la marca <4, 1>, es decir, la cláusula $C_{<4,1>} = <\rightarrow q>$, la cual es la hiper-resolvente del electrón $C_{<3,1>}$ y núcleo C_4.

Si en K_4 disparamos de nuevo t_4, obtenemos el marcado $K_5 = [\varnothing, \varnothing, \varnothing]$, ya que sólo se elimina la marca del lugar q. De acuerdo al corolario 3.1, la cláusula asociada con el disparo de t_4 es la hiper-resolvente del electrón $C_{<4,1>}$ y núcleo C_4, es decir, la cláusula vacía.

Por último, establecemos y probamos que si existe un T-invariante válido en N, entonces S es insatisfacible.

Teorema 3.2.

Si existe un T-invariante válido en N, entonces S es insatisfacible.

Prueba:

Supongamos que existe un T-invariante válido en N, pero que S es satisfacible.

Sea σ la secuencia de transiciones que reproduce el marcado vacío en N, la cual existe por definición de T-invariante válido.

Analicemos el disparo de cada transición en el orden en que aparecen en σ y construyamos un árbol de la siguiente forma.

Sea t_m la transición que disparamos.

1) Si t_m es una transición fuente, entonces creamos un nodo V_i, i = número de nodos creados, y lo etiquetamos con la cláusula asociada con el disparo de t_m.

2) Si t_m no es una transición fuente, entonces consideremos lo siguiente:

   - Sean $m_1, m_2, \ldots, m_j$ las marcas que habilitaron a t_m para su disparo.
   - Sean $C_{m1}, C_{m2}, \ldots, C_{mj}$ las cláusulas asociadas con las marcas $m_1, m_2, \ldots, m_j$, respectivamente
   - Sea C_m la cláusula que determinó a t_m.

En el árbol existen nodos $V_{r1}, V_{r2}, \ldots, V_{rj}$, $\{r_1, r_2, \ldots, r_j\} \subseteq \{1, 2, \ldots, i\}$ e i = número de nodos creados; con etiquetas $C_{m1}, C_{m2}, \ldots, C_{mj}$, respectivamente, ya que tales cláusulas están asociadas con las marcas que habilitaron a t_m para su disparo.

Creamos un nodo V_{i+1} y lo etiquetamos con la cláusula C_m. Luego, creamos un nodo V_{i+2} y lo etiquetamos con la hiper-resolvente de electrones que etiquetan a los nodos $V_{r1}, V_{r2}, \ldots,$

V_{rj} y núcleo que etiqueta a V_{i+1}. Además, el nodo V_{i+2} es el padre de los nodos V_{r1}, V_{r2},..., V_{rj} y de V_{i+1}.

Las hojas del árbol que construimos están etiquetadas con elementos de S (el lema 3.3). Cualquier nodo no hoja de este árbol está etiquetado con una hiper-resolvente de elementos de S ó de hiper-resolventes de S y elementos de S ó de hiper-resolventes de S.

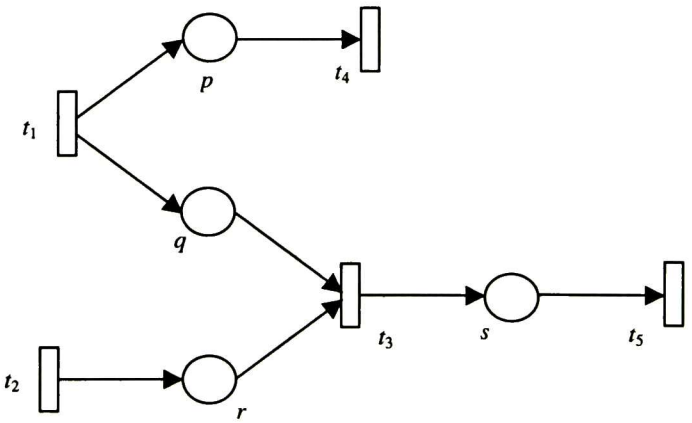
Por ello, este árbol es uno de hiper-resolución, en el cual tenemos al nodo raíz etiquetado con la cláusula vacía. Esto último se debe a que la última transición en σ es una transición sin lugares de salida, a que el marcado alcanzado después de este disparo es el marcado vacío y a que de acuerdo al corolario 3.1, al disparo de tal transición obtenemos la cláusula vacía la cual es insatisfacible. Con lo que llegamos a una contradicción.

Por lo tanto, si existe un T-invariante válido en N, entonces S es insatisfacible.

Un ejemplo donde ilustramos la prueba de la proposición anterior es el siguiente.

Ejemplo 3.9:

Sea $S = \{C_1, C_2, C_3, C_4, C_5\}$, donde $C_1 = <\rightarrow p, q>$, $C_2 = <\rightarrow r>$, $C_3 = <q, r \rightarrow s>$, $C_4 = <p \rightarrow>$ y $C_5 = <s \rightarrow>$. En la gráfica siguiente mostramos la RdP N asociada a S.



A continuación construimos un árbol de hiper-resolución, de acuerdo a lo expuesto en la prueba del teorema 3.2. Para ello comenzamos con el marcado vacío $K_0 = [\varnothing, \varnothing, \varnothing, \varnothing]$ y luego realizamos la secuencia $\sigma = t_1t_2t_3t_4t_5$.

A partir de K_0 si disparamos t_1, obtenemos el marcado $K_1 = [\{<1, 1>\}, \{<1, 1>\}, \varnothing, \varnothing]$.

Al ser t_1 una transición fuente, creamos un nodo V_1 y lo etiquetamos con la cláusula $C_{<1, 1>} = <\rightarrow p, q>$, que es la cláusula asociada con el disparo de t_1.

V_1: $\rightarrow p, q$

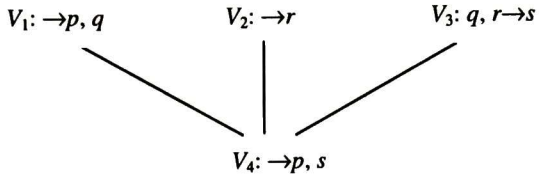
Luego si disparamos t_2, obtenemos el marcado $K_2 = [\{<1, 1>\}, \{<1, 1>\}, \{<2, 1>\}, \varnothing]$.

Por ser t_2 también una transición fuente, creamos un nodo V_2 y lo etiquetamos con la cláusula $C_{<2,1>} = <\rightarrow r>$.

V_1: $\rightarrow p, q$ $\qquad$ V_2: $\rightarrow r$

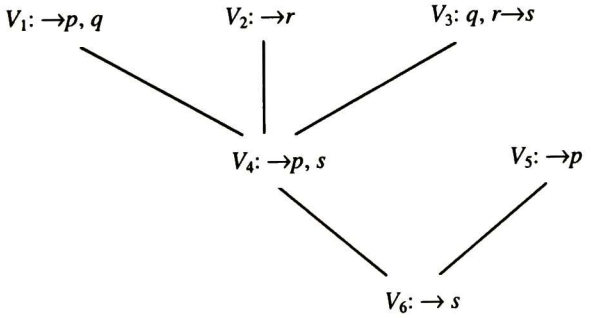
En K_2 si disparamos t_3, obtenemos el marcado $K_3 = [\{<3, 1>\}, \varnothing, \varnothing, \{<3, 1>\}]$.

Creamos el nodo V_3, el cual etiquetamos con la cláusula que determinó t_3, es decir, con la cláusula $C_3 = <q, r \rightarrow s>$. También creamos el nodo V_4, el cual etiquetamos con la hiper-resolvente de electrones $C_{<1, 1>}$, $C_{<2, 1>}$ y núcleo C_3. Además, V_4 es el padre de V_1, de V_2 y de V_3.



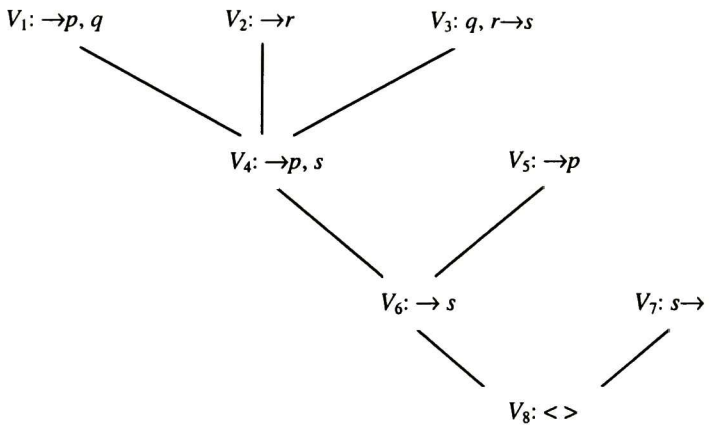
Si Continuamos con la secuencia σ con el marcado K_3 y disparamos t_4, entonces obtenemos el marcado $K_4 = [\varnothing, \varnothing, \varnothing, \{<4, 1>\}]$.

Creamos el nodo V_5 y lo etiquetamos con la cláusula que determinó a t_4, es decir, con la cláusula $C_4 = <p \rightarrow>$. Luego, creamos el nodo V_6, el cual etiquetamos con la hiper-resolvente del electrón que etiqueta a V_5 y núcleo que etiqueta a V_4. Además, V_6 es el padre de V_4 y de V_5.



La única transición que falta disparar para que completemos la secuencia σ es t_5. Al hacerlo, obtenemos el marcado vacío $K_5 = [\varnothing, \varnothing, \varnothing, \varnothing]$

Creamos el nodo V_7 y lo etiquetamos con la cláusula que determinó a t_5, es decir, con la cláusula $C_5 = <s \rightarrow>$. Luego, creamos el nodo V_8 y lo etiquetamos con la hiper-resolvente del electrón que etiqueta a V_6 y núcleo que etiqueta a V_7. Además, V_8 es el padre de V_6 y de V_7.



Este es un árbol de hiper-resolución cuyo nodo raíz está etiquetado con la cláusula vacía, por tanto el conjunto de cláusulas considerado para su construcción es insatisfacible. Como las cláusulas utilizadas para su construcción corresponden a elementos de S, entonces podemos decir que S es insatisfacible.

Si utilizamos el editor (cuyo uso presentamos en el apéndice A) con la RdP del ejemplo 3.9 y si elegimos realizar una simulación automática, entonces podremos observar los mismos marcados y la misma secuencia que las obtenidas en el ejemplo anterior.

IV Construcción de Modelos

1. Introducción

Cuando modelamos un sistema que creemos robusto y encontramos luego que no lo es, no es suficiente con decir "falla en algo", es necesario decir el punto en el que falla.

Al hablar de la lógica proposicional sucede algo similar. Algunas personas no se conforman con oír "el conjunto es satisfacible", sino que esperan ver una manera de satisfacerlo, es decir, requieren de un modelo para el conjunto considerado. La importancia de ello estriba en que un modelo puede indicar una situación en la cual un sistema falla u ocurre una situación no deseable. Tal sistema puede ser uno de manufactura, un circuito digital, un sistema experto. Hablando de sistemas expertos podemos decir además, que un modelo puede indicar falta de información para deducir una respuesta o llegar a una conclusión. En el siguiente acertijo podemos observar que un modelo nos va a servir para llegar a su solución. Por todo lo anterior, un modelo juega el rol que nosotros le hayamos establecido, pudiendo ser alguno de los anteriores u otros más.

Ejemplo 4.1:

Consideremos el siguiente enunciado:

"Si no especifico las condiciones iniciales, entonces mi programa no empezará. Si programo un ciclo infinito, entonces mi programa no terminará. Si el programa no empieza o si no termina, entonces el programa fallará. Puedo decir que si especifico las condiciones iniciales, entonces el programa no fallará."

Ahora introducimos algunos símbolos proposicionales:

I Condiciones iniciales

S Programa empezar

L ciclo infinito

E Programa Terminar

F Programa Falla

A cada enunciado le asociamos una fórmula:

$<\neg I \rightarrow \neg S>$ Si no especifico las condiciones iniciales, entonces el programa no empezará.

$<L \rightarrow \neg E>$ Si programo un ciclo infinito, entonces el programa no terminará.

$<\neg S, \neg E \rightarrow F>$ Si el programa no empieza o no termina, entonces el programa fallará.

Esta ultima proposición la podemos separar en: $<\neg S \rightarrow F>$ y $<\neg E \rightarrow F>$.

De la última oración del texto podemos plantear la siguiente proposición:

$<I \rightarrow \neg F>$ Si especifico las condiciones iniciales, entonces el programa no fallará.

Para determinar la validez de esta última proposición vamos a negarla, con lo que tenemos: $<I \wedge F>$.

El conjunto de cláusulas a considerar es: $S = \{<S \rightarrow I>, <E, L \rightarrow>, <\rightarrow F, S>, <\rightarrow E, F>, <\rightarrow I>, <\rightarrow F>\}$.

La matriz de incidencia A de la RdP asociada con S es:

$$A = \begin{array}{c} \\ E \\ F \\ I \\ L \\ S \end{array} \begin{bmatrix} 0 & \mathbf{-1} & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \mathbf{1} & 0 & \mathbf{1} \\ \mathbf{1} & 0 & 0 & 0 & \mathbf{1} & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 \\ -1 & 0 & \mathbf{1} & 0 & 0 & 0 \end{bmatrix}$$

En A hemos sombreado algunas de sus componentes. Observemos que las componentes elegidas de un mismo renglón, ambas son iguales a 1 (aunque la otra posibilidad es que fueran iguales a -1). Por tal propiedad y por haber elegido una componente de cada columna de A, tales componentes definen un camino abierto P en A. Dicho camino denota el modelo $M = \{I, S, F\}$ para S. El modelo definido por P contiene al átomo I debido a que: la componente en la intersección del renglón asociado con I y la primer columna de A es igual a 1 y también debido a que tal componente está en P Por razones similares también se encuentran en M los átomos S y F. Los conceptos nuevos que introducimos en este párrafo los presentamos en la sección 2 titulada "Conceptos Fundamentales".

Luego, en la sección 3 exponemos un método para construir un modelo a partir de la matriz de incidencia de la RdP asociada a un conjunto de cláusulas. Además, en esta misma sección presentamos algunos teoremas para probar la validez del método en cuestión.

A lo largo de este capítulo asumimos, al igual que en los capítulos anteriores, que:

$S = \{C_1, C_2, \ldots, C_n\}$ es un conjunto de n cláusulas.

$L_1 = \{l_1, l_2, \ldots, l_k\}$ es el conjunto de los átomos pertenecientes al soporte de S.

N es la RdP asociada a S.

Además, para este capítulo asumimos que:

A es la matriz de incidencia de N, donde

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} & & a_{1,n} \\ a_{2,1} & a_{2,2} & & a_{2,n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{k,1} & a_{k,2} & & a_{k,n} \end{bmatrix}$$

2. Conceptos Fundamentales

En esta sección presentamos primeramente en la definición 4.1 los conceptos de camino, camino abierto y camino cerrado, todos ellos en A. Luego en el teorema 4.1, establecemos que si S es satisfacible, entonces existe un camino abierto en A. Por último en la definición 4.2 introducimos una asignación que se induce a partir de un camino abierto en A.

Definición 4.1: (*camino, camino cerrado y camino abierto en A*)

Sea D una pareja $<i, j>$, para $1 \leq i \leq k$ y $1 \leq j \leq n$.

Un camino P en la matriz de incidencia A es un conjunto de n parejas de la forma $\{<i, 1>, <i, 2>, \ldots, <i, n>\}$, tal que si la pareja $<i, j> \in P$, entonces $a_{i,j} \neq 0$.

Un *camino cerrado* en A, es un camino en A en el cual existe una pareja $<p, q>$ y una pareja $<p, r>$, tal que $a_{p,q} * a_{p,r} < 0$, para $1 \leq p \leq k$ y $1 \leq q, r \leq n$.

Un *camino abierto* en A es un camino que no es cerrado.

Ejemplo 4.2:

Sea $S = \{C_1, C_2, C_3, C_4\}$, donde $C_1 = <\leftrightarrow p, q>$, $C_2 = <q \rightarrow r>$, $C_3 = <p, r \rightarrow q>$ y $C_4 = <p, r \rightarrow s>$. La matriz de incidencia A asociada a la RdP N es:

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

Algunos caminos en A son $P_1 = \{<1, 1>, <2, 2>, <1, 3>, <1, 4>\}$, $P_2 = \{<1, 1>, <2, 2>, <3, 3>, <4, 4>\}$ y $P_3 = \{<2, 1>, <3, 2>, <2, 3>, <3, 4>\}$.

El camino P_1 es un camino cerrado, ya que existe en él la pareja $<1, 1>$ y la pareja $<1, 3>$, tal que $a_{1,1} * a_{1,3} < 0$.

El camino P_2 es un camino abierto.

El camino P_3 es un camino cerrado, ya que $a_{3,2} * a_{3,4} < 0$ y ya que $<3, 2>$ y $<3, 4> \in P_3$.

Cuando un conjunto de cláusulas es satisfacible, siempre podemos encontrar un camino abierto en la matriz de incidencia de la RdP asociada a dicho conjunto.

Teorema 4.1:

Si S es satisfacible, entonces existe un camino abierto en A.

Prueba:

Si S es satisfacible, entonces existe una asignación **A**, tal que **A** es un modelo para S.

Si **A** es un modelo para S, entonces **A** satisface cada cláusula en S.

Para cada cláusula $C_j \in S$, $1 \leq j \leq n$, consideremos sólo uno de los dos puntos siguientes y determinemos un conjunto de parejas P.

i. Si el átomo $l_i \in$ **A** y si l_i determina una literal positiva en C_j, entonces $<i, j> \in P$. La componente $a_{i,j} = 1$, ya que l_i determina una literal positiva en C_j.

ii. Si el átomo $l_i \notin$ **A** y si l_i determina una literal negativa en C_j, entonces $<i, j> \in P$. La componente, $a_{i,j} = -1$, ya que l_i determina una literal negativa en C_j.

Al menos uno de los dos puntos anteriores se debe cumplir al considerar cada cláusula en S, ya que al menos un átomo $l_i \in$ **A** ó al menos un átomo $l_i \notin$ **A**, esto debido a que cada cláusula en S se satisface por **A**.

El número de parejas en P es igual a n, ya que para cualquier átomo que pertenezca al soporte de S ó pertenece a **A** ó no pertenece, por lo que debimos de considerarlo en i ó en ii.

Por todo lo anterior, P es un camino.

En P no existe $<q, a>$ y $<q, b>$, tal que $a_{q,a} * a_{q,b} < 0$, ya que un átomo l_q, ó lo consideramos en i ó lo consideramos en ii, pero no en ambos.

Por lo anterior, P es un camino abierto.

Por lo tanto, existe un camino abierto en A.

Luego, Si S es satisfacible, entonces existe un camino abierto en A.

Un ejemplo donde ilustramos el teorema anterior es el siguiente.

Ejemplo 4.3:

Sea $S = \{C_1, C_2, C_3, C_4, C_5\}$, donde $C_1 = <\rightarrow t>$, $C_2 = <p \rightarrow>$, $C_3 = <q, r \rightarrow p>$, $C_4 = <s, t \rightarrow q>$ y $C_5 = <s \rightarrow r>$.

Una asignación que es un modelo para S es **A** = $\{r, t\}$.

Para verificar la existencia de un camino abierto P, consideremos cada cláusula en S.

1. El átomo $t \in$ **A** y determina una literal positiva en C_1, por lo tanto $<5, 1> \in P$

2. El átomo $p \notin$ **A** y determina una literal negativa en C_2, por lo tanto $<1, 2> \in P$

3. El átomo $q \notin$ **A** y determina una literal negativa en C_3, por lo tanto $<2, 3> \in P$

4. El átomo $s \notin$ **A** y determina una literal negativa en C_4, por lo tanto $<4, 4> \in P$

5. El átomo $s \notin \mathbf{A}$ y determina una literal negativa en C_5, por lo tanto $<4, 5> \in P$.

Con lo que tenemos el camino abierto $P = \{<5, 1>, <1, 2>, <2, 3>, <4, 4>, <4, 5>\}$.

La matriz de incidencia A de la RdP asociada a S, en la cual podemos verificar que P es un camino abierto es la siguiente.

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

Si en el punto 5 hubiéramos considerado al átomo r, entonces en lugar de la pareja $<4, 5>$ debería de aparecer en P la pareja $<3, 5>$. Con lo cual tendríamos el camino abierto $\{<5, 1>, <1, 2>, <2, 3>, <4, 4>, <3, 5>\}$.

Otra asignación que es un modelo para S es $\mathbf{A_2} = \{t\}$, de la cual podemos verificar la existencia del camino abierto $P_2 = \{<5, 1>, <1, 2>, <3, 3>, <4, 4>, <4, 5>\}$ ó del camino abierto $P_3 = \{<5, 1>, <1, 2>, <2, 3>, <4, 4>, <4, 5>\}$.

Ya una vez que determinamos la existencia de un camino abierto en A cuando S es satisfacible, ahora veamos cómo inducimos una asignación a partir de un camino abierto en A.

Definición 4.2: (*asignación que se induce a partir de un camino abierto*)

Sea P un camino abierto en A.

Sea la pareja $<i, j> \in P$, para $1 \leq i \leq k$ y $1 \leq j \leq n$.

Sea l_i el átomo asociado con el renglón i de A.

Decimos que $\mathbf{A}$ es una asignación que se induce a partir de P sii $\mathbf{A}$ contiene al átomo l_i y $a_{i,j} = 1$.

3. Método para Construir Modelos y Prueba de su Validez

El método consiste en encontrar un camino abierto P en la matriz de incidencia de la RdP asociada con el conjunto de cláusulas S a considerar. Luego, basándonos en el teorema 4.2 que mostramos posteriormente, podemos decir que la asignación que se induce a partir de P es un modelo para S. Por último, en la proposición 4.1 establecemos que si S es un conjunto satisfacible, entonces existe un camino abierto en A que induce un modelo para S, con lo cual probamos la validez del método.

Teorema 4.2: (*la asignación que se induce a partir de un camino abierto en A es un modelo para S*)

Sea P un camino abierto en A.

Sea **A** la asignación que se induce a partir de P.

La asignación **A** es un modelo para S.

Prueba:

Supongamos que por el contrario, **A** no es un modelo para S.

Luego, existe alguna cláusula en S a la cual no satisface **A**. Digamos que esa cláusula es C_j.

Supongamos además que la columna j de A es la representación vectorial de C_j y que el átomo l_i está asociado con el renglón i de A.

Sabemos por definición, que P debe contener a la pareja $<i, j>$ y que $a_{i,j} \neq 0$.

Tenemos sólo dos casos a considerar, que $a_{i,j} = 1$ ó que $a_{i,j} = -1$.

Caso 1: Si $a_{i,j} = 1$, entonces $l_i \in$ **A** y por tanto **A** satisface a C_j.

Caso 2: Si $a_{i,j} = -1$, entonces $l_i \notin$ **A** y por tanto **A** satisface a C_j.

En cualquiera de los dos casos anteriores se contradice la suposición de que **A** no satisface a la cláusula C_j. Consecuentemente, no existe cláusula en S que no satisfaga **A**.

Por lo tanto, la asignación **A** es un modelo para S.

En el ejemplo siguiente ilustramos el teorema y definiciones anteriores.

Ejemplo 4.4:

Consideremos al camino abierto $P_2 = \{<1, 1>, <2, 2>, <3, 3>, <4, 4>\}$, del ejemplo 4.2 y determinemos la asignación correspondiente.

Cómo $a_{1,1} = 1$, entonces $p \in$ **A**.

Cómo $a_{2,2} = -1$, entonces $q \notin$ **A**.

Cómo $a_{3,3} = -1$, entonces $r \notin$ **A**.

Cómo $a_{4,4} = 1$, entonces $s \in$ **A**.

Con lo anterior **A** = $\{p, s\}$.

La asignación anterior es un modelo para S ya que satisface cada cláusula en S.

Reuniendo los resultados obtenidos en los dos teoremas anteriores, podemos concluir que cuando un conjunto de cláusulas S es satisfacible, siempre nos es posible encontrar un camino abierto P en A, el cual induce un modelo para S. La Proposición que presentamos a continuación establece este hecho.

Proposición 4.1: (*existencia de modelos*)

Si S es un conjunto satisfacible, entonces existe un camino abierto en A que induce un modelo para S.

Prueba:

Supongamos que S es un conjunto satisfacible. Entonces, por el Teorema 4.1, existe un camino abierto P en A. Además, de acuerdo con lo establecido en el Teorema 4.2, la valuación que se induce a partir de P es un modelo para S.

Por lo tanto, existe un camino abierto en A, el cual induce un modelo para S.

Veamos un ejemplo donde apliquemos lo visto en el capítulo anterior y este.

Ejemplo 4.5:

Este ejemplo va enfocado a la validación de circuitos. ¿Un circuito cumple con las especificaciones dadas?.

Primero, veremos si un circuito que fue modelado mediante una RdP cumple con las especificaciones dadas por una tabla de verdad. Posteriormente, trataremos de validar un circuito basándonos en el primero (el cual ya fue probado).

Utilizaremos los siguientes símbolos en los diagramas de los circuitos:

La siguiente tabla de verdad muestra el comportamiento de un sumador completo, en ella las entradas son X y Y, el acarreo es C y la suma es S.

X	Y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Veamos si el siguiente circuito, que únicamente utiliza compuertas NAND, cumple con las especificaciones anteriores.

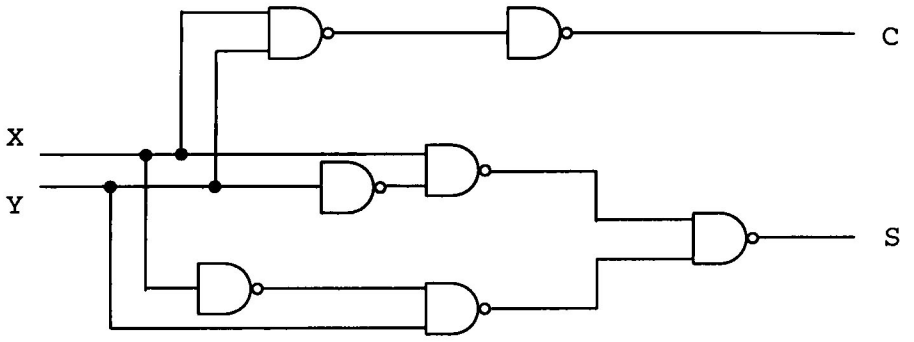


Figura 4.1 Circuito Lógico de un sumador completo (usando sólo compuertas NAND)

El conjunto de cláusulas correspondientes al circuito de la figura 4.1 es S_1 = {<S→X, Y>, <X, Y, S→>, <X→Y, S>, <Y→X, S>, <C→X >, <C→Y>, <X, Y→C>}.

Debemos de verificar las propiedades de la tabla de verdad que mostramos anteriormente. Comencemos por la primera propiedad la cual enuncia que a partir de <X→> y <Y→>, debemos concluir <S→> y <C→>. Consideremos al conjunto formado por estas cláusulas, pero negando la última de ellas (lo que queremos probar) S_2 = {<X→>, <Y→>, <→S, C>}.

Si tomamos la RdP asociada al conjunto $S_1 \tau S_2$ y ejecutamos la simulación automática del editor de RdP (PNToolkit) podremos ver que el conjunto es insatisfacible, al declarar la existencia de un T-invariante válido y su correspondiente secuencia de disparos ($t_{10}t_1t_5t_8t_8t_9$, en este caso). Si tomamos la matriz de incidencia de la RdP en cuestión, no podremos encontrar ningún camino abierto en ella. Por tanto podemos decir que sí cumple con la primera propiedad. Enseguida mostramos en la figura 4.2 la RdP que dibujamos en el editor PNToolkit para llevar a cabo la simulación automática.

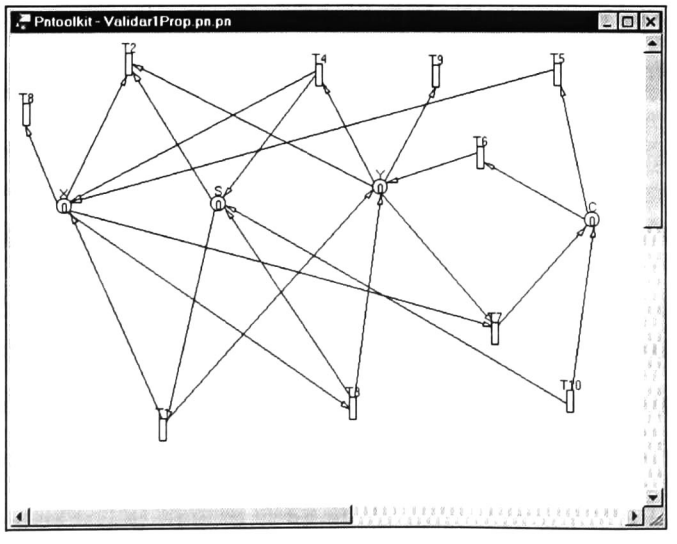


Figura 4.2 RdP tomada del Editor PNToolkit para realizar una simulación automática

De forma similar debemos de probar que a partir de <X→> y <→Y >, podemos concluir <→S > y <C→>. En este caso, consideramos al conjunto de cláusulas S_2 = {<X→>, <→Y>, <S→C>}. Con el conjunto de cláusulas S_1 τ S_2, el editor nos muestra que existe un T-invariante válido y la secuencia: $t_9t_9t_4t_7t_{10}t_5t_5t_8t_8$. Así podemos decir que cumple con la segunda propiedad.

Para probar que cumple con la tercera propiedad consideramos al conjunto de cláusulas S_2 = {<→X>, <Y→>, <S→C>}. El conjunto resultante (S_1 τ S_2) es insatisfacible al encontrar el editor un T-invariante válido y su secuencia $t_8t_3t_9t_{10}t_6t_9$, con lo que se cumple con la tercer propiedad.

La cuarta propiedad queda establecida al encontrar la secuencia $t_8t_8t_8t_9t_9t_9t_7t_5t_7t_6t_7t_{10}t_2$ correspondiente a un T-invariante válido en la RdP asociada al conjunto S_1 τ {<→X>, <→Y>, <C→S>}.

Al haber comprobado las cuatro propiedades de la tabla de verdad que mostramos anteriormente, podemos decir que el circuito se apega a las especificaciones dadas.

Ahora bien, queremos determinar (validar) si el siguiente circuito cumple con las mismas cuatro propiedades que el circuito de la figura 4.1.

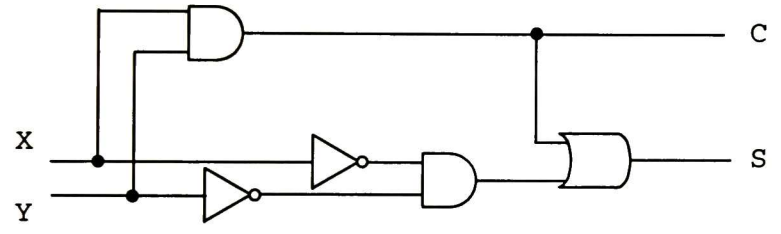


Figura 4.3 Circuito lógico para validar que cumpla con las especificaciones de un sumador completo

El conjunto de cláusulas de este circuito es S_3 = {<S, X→C>, <S, Y→C>, <C→S>, <→ X, Y, S>, <C→X >, <C→Y>, <X, Y→C>}.

No vamos a probar cada una de las propiedades que buscamos cumpla este circuito. En lugar de ello, vamos a compararlo con el circuito que ya validamos anteriormente, es decir, determinaremos si el conjunto de proposiciones asociado a este circuito es consecuencia del conjunto de proposiciones asociado al circuito anterior. Para ello debemos de considerar al conjunto de cláusulas S_1 τ S_3' donde S_3' es la negación de S_3, es decir, S_3 = {<X→C, S>, <Y→C, S>, <S, C→>, <S→X, Y>}. La cuarta cláusula de S_3' (<S→X, Y>) también pertenece a S_1 por lo que vamos a considerar al conjunto S_4 = {<X→C, S>, <Y→C, S>, <S, C→>} en lugar del conjunto de cláusulas S_3'. La matriz de incidencia correspondiente a la RdP asociada con el conjunto de cláusulas S_1τS_4 es la siguiente:

$$A = \begin{array}{c} \\ X \\ Y \\ S \\ C \end{array} \begin{bmatrix} \mathbf{1} & -1 & -1 & 1 & 1 & 0 & -1 & -1 & 0 & 0 \\ 1 & \mathbf{-1} & 1 & -1 & 0 & 1 & -1 & 0 & -1 & 0 \\ -1 & -1 & \mathbf{1} & 1 & 0 & 0 & 0 & 1 & 1 & -1 \\ 0 & 0 & 0 & 0 & \mathbf{-1} & -1 & 1 & 1 & 1 & -1 \end{bmatrix}$$

En la matriz de incidencia existe el camino abierto P = {<1, 1>, <2, 2>, <3, 3>, <4, 5>}. Por tanto existe una asignación que es un modelo para el conjunto y tal modelo es {X, S}. Esto nos indica que el circuito anterior no cumple con las especificaciones buscadas (al haber un modelo).

El modelo que encontramos nos indica que el circuito no cumple con la tercera propiedad, esto es, que dadas las entradas X = 1 y Y = 0, no obtenemos alguna o ninguna de las salidas esperadas (S = 1 y C = 0).

Si hubiéramos encontrado un T-invariante válido, entonces podríamos haber concluido que el nuevo circuito cumplía con las especificaciones dadas.

Conclusiones y Trabajo Futuro

Hemos propuesto un método para determinar la insatisfacibilidad de un conjunto de cláusulas. En caso de concluir satisfacibilidad encuentra un modelo para tal conjunto.

En los capítulos anteriores expusimos sólo la teoría del método. En el apéndice A presentamos una guía para el uso de una herramienta visual que toma como base la teoría y, después de un proceso interno muestra un resultado: un modelo para el conjunto de cláusulas en caso de que sea satisfacible tal conjunto ó en caso contrario, expresa que el conjunto de cláusulas es insatisfacible. Muestra más información de lo que hemos mencionado en este párrafo, pero para profundizar en ello es mejor referirse al apéndice en cuestión.

Como trabajo futuro queda probar si el conjunto de T-invariantes que considera el programa es suficiente, es decir, probar que si existe un T-invariante válido para el conjunto de cláusulas dado, entonces el programa lo considera. También se puede mejorar el algoritmo para determinar si un conjunto de cláusulas es insatisfacible o no, algunas vías posibles son: simplificación de la RdP, mejoramiento del algoritmo para calcular los T-invariantes y en la realización de la secuencia de disparos cuyo vector característico asociado es uno de los T-invariantes que se calcularon. Además, se puede hacer un estudio para determinar la complejidad de los algoritmos para determinar si un T-invariante es válido, así como de los algoritmos usados para la búsqueda de modelos, todos ellos factibles de mejora. Así mismo, queda pendiente compararlo con otros métodos similares, extenderlo a la lógica de predicados, lógica abductiva u otras, y modificarlo de alguna forma para que tenga una aplicación práctica.

Apéndice A

Descripción del Editor PNToolkit

PNToolkit es un editor de Redes de Petri, el cual además de permitirnos plasmar gráficamente una Red de Petri, determina si el conjunto de cláusulas asociado con tal Red de Petri es insatisfacible o presenta un modelo para tal conjunto (en caso de ser satisfacible).

Hemos dividido este apéndice en dos secciones, la primera de ellas nos muestra las diferentes formas en que podemos adquirir la gráfica de una Red de Petri: 1) a partir de un archivo, 2) dibujo manual, 3) utilizando su matriz de incidencia y 4) utilizando su conjunto de cláusulas asociado. En la segunda parte (Herramientas del sistema), mostramos como realizar una simulación sobre una Red de Petri (para determinar si su conjunto de cláusulas asociado es insatisfacible), la cual puede ser de forma automática o guiada por el usuario, por último, introducimos la búsqueda de modelos, la cual se lleva a cabo sobre la matriz de incidencia de la Red de Petri asociada con el conjunto de cláusulas del cual se busca un modelo. La búsqueda de modelos se lleva a cabo de forma automática.

1. Los Pasos Iniciales

Después de que hayamos iniciado el sistema aparecerá una ventana con título "PN Specification Toolkit" la cual contiene todos los menús usados para comenzar la mayoría de las acciones del usuario, así como botones de acceso directo a opciones de los menús anteriores (opciones de uso frecuente).

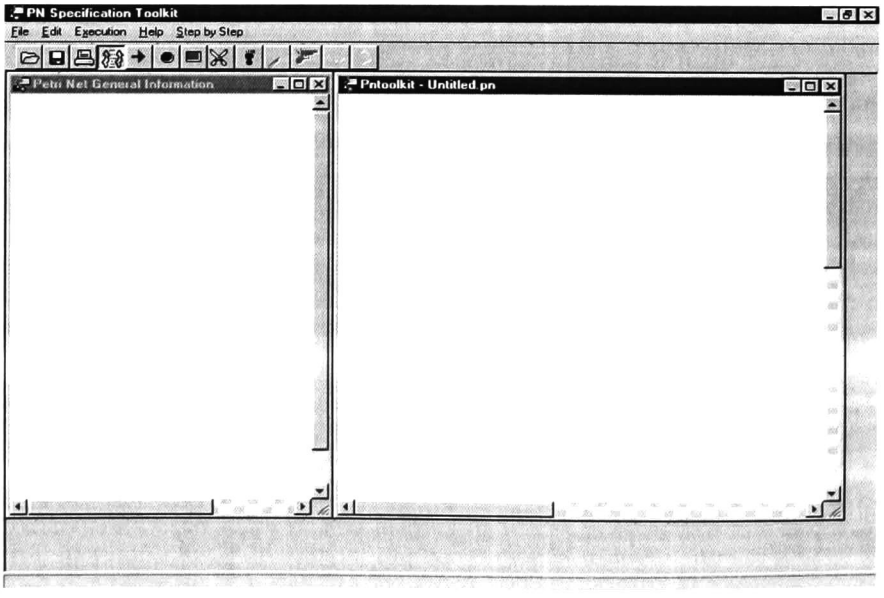


Figura A.1 Ventana al iniciar el sistema

Dentro de la ventana titulada "PN Specification Toolkit" aparecen dos ventanas más, una titulada "PNToolkit" seguido del nombre del archivo en uso (al inicio del sistema se tiene un archivo denominado "Untitled.pn") en la cual podemos ver gráficamente la RdP y que denominaremos ventana de dibujo; la otra ventana titulada "Petri Net General Information" ó con el nombre de un archivo de texto, muestra información de la RdP presente en la ventana de dibujo ó de alguna otra RdP, dependiendo del archivo en uso; información tal como: matriz de incidencia, T-invariantes, P-invariantes, cláusulas en la RdP y otras que se explicarán más adelante. A esta última ventana, la denominaremos ventana de edición. La Figura A.1 muestra el estado del sistema una vez que lo hemos iniciado.

Para adquirir una RdP en particular, podemos hacerlo por medio de alguna de las siguientes vías:

- *Abrir un archivo existente* el cual contiene la RdP.
- Si no queremos trabajar con un archivo existente, es posible dibujar una RdP usando los botones de lugar, transición y arco, lo cual denominaremos una *operación manual.* Para dibujar un lugar, una transición ó un arco en la ventana de dibujo, sólo es necesario que pulsemos el botón izquierdo del ratón sobre el botón requerido y luego que pulsemos el mismo botón izquierdo, en el área en que deseamos dibujar el elemento seleccionado.
- Si no queremos dibujar cada lugar, transición y arco, existe la posibilidad de introducir la *matriz de incidencia* de la RdP que queremos dibujar ó su *conjunto de cláusulas* asociado. El programa toma tales datos y dibuja la RdP determinada por la matriz de incidencia ó la RdP asociada con el conjunto de cláusulas, según sea el caso.

1. 1 Dibujo de la RdP a partir de un archivo

Para trabajar con una RdP especificada en un archivo existente, sólo es necesario que elijamos del menú "File" la opción "Open" o bien que pulsemos el botón correspondiente y luego seleccionemos el archivo requerido. Tales archivos tienen la extensión ".pn"

1. 2 Dibujo manual de la RdP

Como ya lo mencionamos anteriormente, para dibujar una RdP manualmente, únicamente elegimos con el ratón el botón del componente correspondiente que deseamos dibujar (lugar, transición o arco). Luego para el caso de los lugares y transiciones, sólo oprimimos el botón izquierdo del ratón sobre el área dónde deseamos dibujar el lugar ó transición. El área debe estar dentro de la ventana de dibujo y no se permite dibujar un lugar sobre otro ó una transición sobre otra. Para el caso de los arcos, debemos oprimir el botón izquierdo del ratón sobre un lugar y luego sobre una transición ó viceversa.

Se permite mover los lugares y transiciones ya dibujados. Si deseamos mover cualquiera de ellos, primero pulsamos el botón de mover, luego posicionamos el ratón sobre uno de ellos y oprimimos el botón izquierdo del ratón, sin dejar de oprimirlo arrastramos el lugar o transición elegido hasta la nueva posición, la cual queda indicada cuando dejamos de oprimir el botón izquierdo del ratón. Al mover un lugar o una transición, también se mueven los arcos que salen o entran a ellos (sí es que existen).

Para eliminar una componente ya dibujada, primero elegimos el botón de corte, luego posicionamos el ratón sobre la componente a eliminar y oprimimos el botón izquierdo del mismo.

Además, contamos con la herramienta deshacer (undo), la cual sólo nos permite retroceder al estado anterior a la última acción de las siguientes: borrado, movimiento o creación de un lugar, arco ó transición.

1. 3 Dibujo de la RdP a partir de su matriz de incidencia

Para dibujar una RdP a partir de su matriz de incidencia primero elegimos del menú "File" la opción "Matrix" con lo cual aparecerá una ventana titulada "Incidence Matrix" solicitando el número de lugares y el número de transiciones de la RdP a dibujar. Tal número de lugares o transiciones debe estar entre 1 y 20. Una vez que introducimos un número dentro del rango anterior y que oprimimos el botón "OK", se cierra esta ventana y aparece una nueva titulada "FillMatrix" (Figura A.3), la cual contiene una matriz de número de renglones igual al número de lugares y número de columnas igual al número de transiciones.

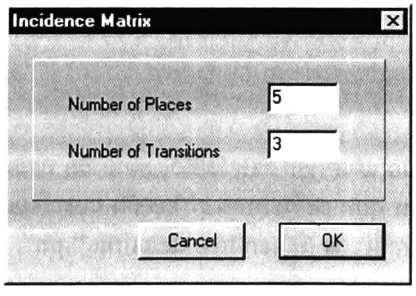


Figura A.2 Ventana "Incidence Matrix"

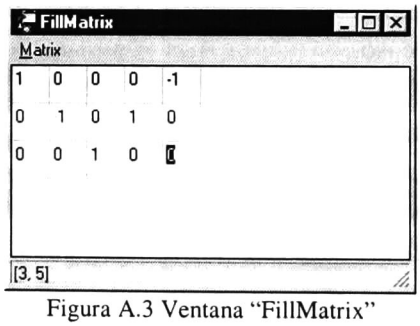


Figura A.3 Ventana "FillMatrix"

Inicialmente en cada celda de la matrix en la ventana titulada "FillMatrix" contiene valores iguales a cero. El usuario puede introducir cualquier valor de 0, 1 y –1 en cada una de tales celdas. En la barra de estados de esta ventana aparece el número de renglón y número de columna actual (**[3, 5]**, indica que la posición actual es renglón 3, columna 5). Una vez que introducimos todos los valores necesarios, podemos elegir del menú "Matrix" la opción "Save and Exit" con lo que se cerrará esta ventana y se dibujará la RdP determinada por la matriz de incidencia que acabamos de introducir, tal como lo muestra la Figura A.4. También

pudimos haber optado por la opción "Exit without saving" con lo que se cancelará la operación de dibujar la RdP a partir de su matriz de incidencia.

Luego de haber creado una RdP a partir de su matriz de incidencia podemos reacomodar sus lugares y transiciones. En la Figura A.5 mostramos un posible acomodo de la RdP creada en este apartado.

Nota: Si elegimos la opción "Save and Exit" en la ventana "FillMatrix", toda componente en la ventana de dibujo será eliminada y reemplazada por la RdP determinada por la matriz de incidencia que introducimos.

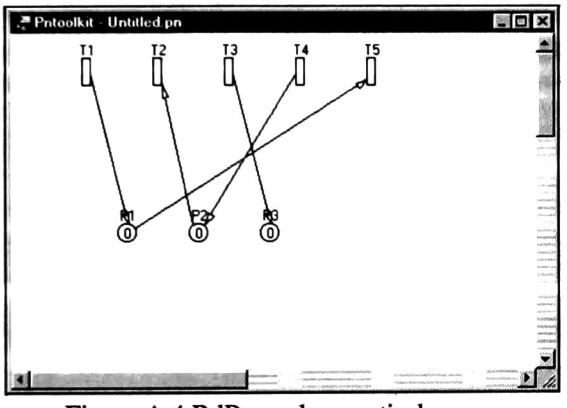


Figura A.4 RdP creada a partir de su matriz de incidencia

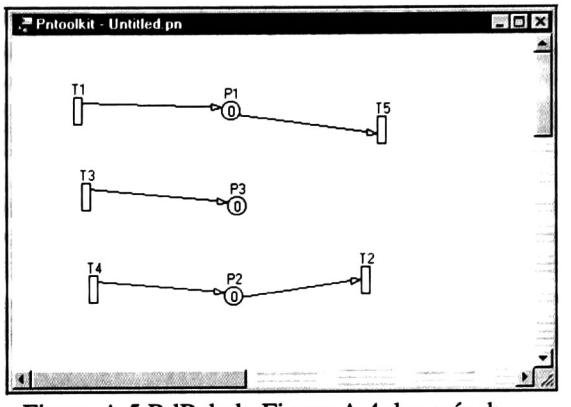


Figura A.5 RdP de la Figura A.4 después de mover sus lugares y transiciones

1.4 Dibujo de la RdP a partir de un Conjunto de Cláusulas

También podemos dibujar una RdP a partir de un conjunto de cláusulas. Primero, debemos elegir del menú "File" la opción "Clause" con lo cual aparecerá una ventana titulada "IntroClauseForm" (figura A.6) solicitando un conjunto de cláusulas. Luego de introducir un conjunto de cláusulas se dibujará la RdP asociada con dicho conjunto.

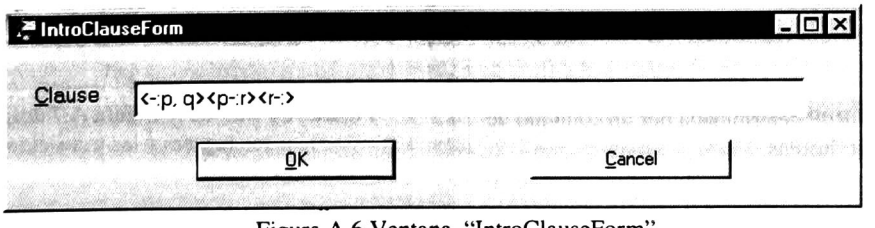


Figura A.6 Ventana "IntroClauseForm"

Cada cláusula debe estar delimitada por los símbolos "**<**" y "**>**". Dentro de las cláusulas cada átomo es una letra minúscula de la "**a**" a la "**z**". Si **l** es un átomo que determina una literal negativa (literal positiva), entonces **l** se encuentra del lado izquierdo (derecho) del

símbolo "-:" (símbolo de implicación). Si existe más de un átomo del lado izquierdo del símbolo "-:" ó del lado derecho del mismo símbolo, entonces utilizamos el símbolo "," para separarlos.

Los siguientes son conjuntos de cláusulas válidos:

1. **<-:p, q> <q-:r> <r-:>**
2. **<-:s>**
3. **<a-:>**
4. **<a-:r, x><x-:z><-:w>**

Los siguientes no son conjuntos de cláusulas válidos:

1. **<A, z>**
2. **<a z>**
3. **<p, q>,<q-:r>**
4. **<p, q, r**
5. **<p:-r>**

Al dibujar una RdP a partir de un conjunto de cláusulas también se restringe a que el máximo número de cláusulas sea 20 y el máximo número de literales apareciendo en tales cláusulas, también sea 20.

Una vez que introducimos un conjunto de cláusulas válido y que pulsamos el botón "OK" de la ventana "IntroClauseForm", aparecerá en la ventana de dibujo la RdP asociada a dicho conjunto (ver Figura A.7). Si movemos los lugares y las transiciones de la Figura A.7, obtenemos la RdP de la Figura A.8.

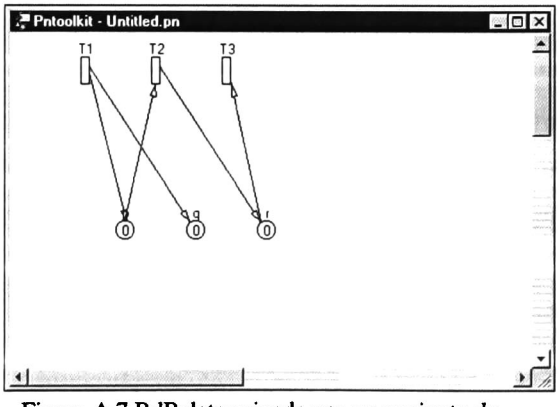


Figura A.7 RdP determinada por un conjunto de cláusulas

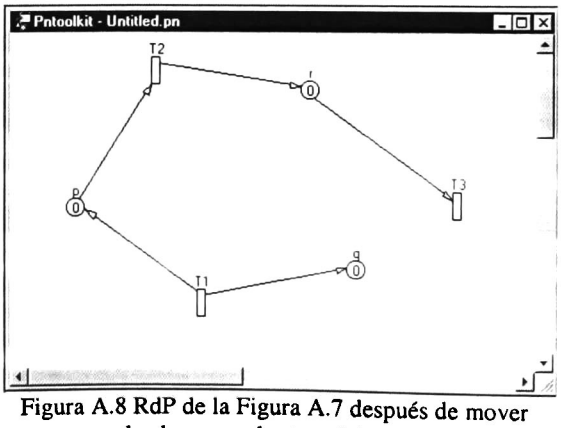


Figura A.8 RdP de la Figura A.7 después de mover los lugares y las transiciones

Nota: Si introducimos un conjunto de cláusulas válido y pulsamos el botón "OK" de la ventana "IntroClauseForm", entonces todo elemento en la ventana de dibujo será eliminado y reemplazado por las componentes asociadas con el conjunto de cláusulas que introducimos.

2. Herramientas del Sistema

Una vez que hemos dibujado una RdP, podemos hacer uso de cualquiera de las dos herramientas principales del sistema: la *simulación*, que determina si el conjunto de cláusulas asociado con la RdP es insatisfacible y la de *búsqueda de modelos*, la cual muestra un modelo para el conjunto de cláusulas asociado con la RdP, sí es que lo hay. La herramienta de simulación puede ser automática o bien paso a paso con interacción del usuario.

Existen algunas otras herramientas secundarias como lo es el determinar: P-invariantes, T-invariantes, cláusulas asociadas a la RdP, matriz de incidencia de la RdP, entre otras.

2.1 Simulación

La herramienta de simulación como ya lo mencionamos anteriormente, puede ser automática o bien guiada por el usuario. A continuación detallamos cada una de ellas.

2.1.1 Simulación Automática

Para realizar una simulación automática primero debemos de dibujar una RdP por cualquiera de los métodos mencionados en la sección 1. Luego, elegimos la opción "Simulation" del menú "Execution" para que se lleve a cabo la simulación. Después de lo anterior, aparece una ventana la cual especifica si hubo un T-invariante válido en la RdP ó si no lo hubo. Además, en la ventana de edición aparecerá todo el proceso de la simulación: T-invariante a probar, el disparo de cada transición perteneciente al T-invariante a probar, el marcado alcanzado después del disparo de cada transición, lo que se ha realizado del T-invariante y al final, la secuencia de disparos realizada.

Supongamos que tenemos la RdP de la Figura A.9.

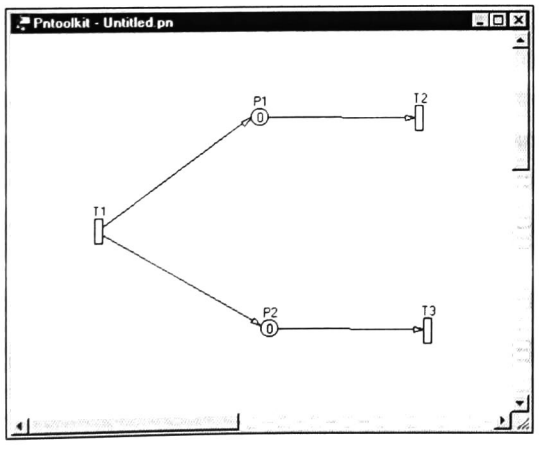


Figura A.9 RdP para llevar a cabo una simulación

Si elegimos del menú "Execution" la opción "Simulation", se visualizará el sistema tal como muestra la Figura A.10. En la ventana de edición se observa la información de la simulación.

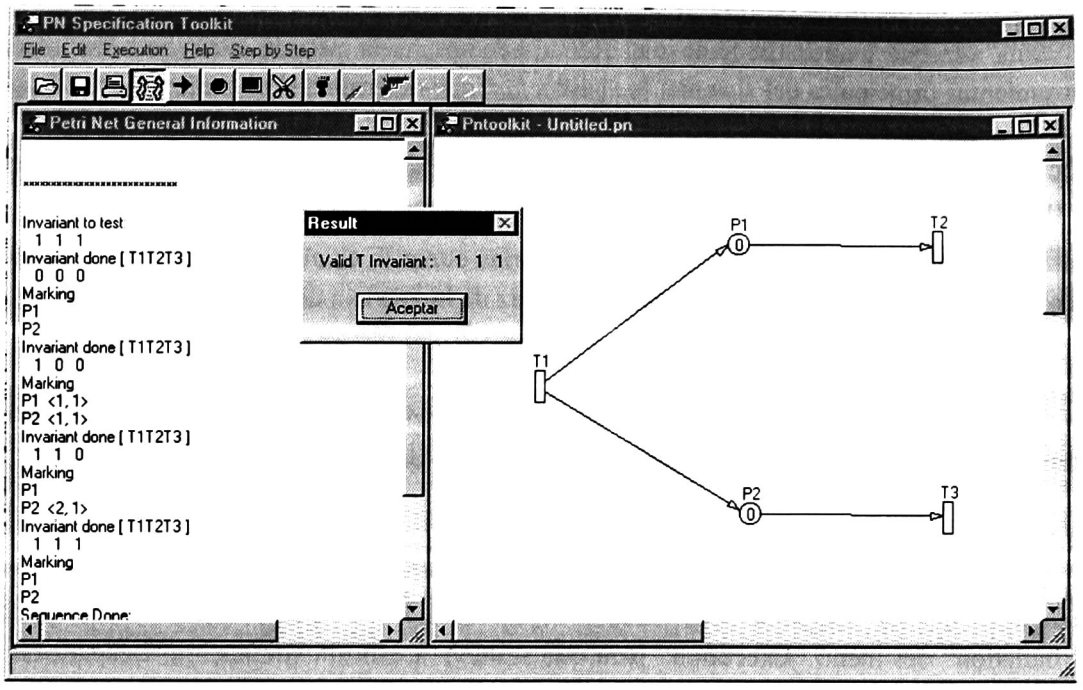


Figura A.10 Estado del sistema después de realizar la simulación

Ahora, analicemos la información de la ventana de edición. Lo primero que aparece es el T-invariante a probar, seguida por el T-invariante realizado (nada aún) y el marcado actual (vacío):

****
Invariant to test:
1 1 1
Invariant done (T1T2T3)
0 0 0
Marking
P1
P2

Luego, al disparo de cada transición se actualiza el T-invariante realizado y el marcado alcanzado. Para el ejemplo en cuestión, si disparamos la transición T1, entonces se incrementa la primer componente del T-invariante realizado y el marcado alcanzado contiene marcas en los lugares P1 y P2.

Invariant done (T1T2T3)
1 0 0
Marking
P1 <1, 1>
P2 <1, 1>

La misma explicación para el bloque anterior, se sigue para el siguiente bloque, el cual indica que hemos disparado la transición T2.

```
Invariant done ( T1T2T3 )
  1  1  0
Marking
P1
P2 <2, 1>
```

En el siguiente bloque el T-invariante realizado es igual al T-invariante a probar y el marcado actual es vacío por lo que hemos terminado. Al final se indica la secuencia de disparos realizada.

```
Invariant done ( T1T2T3 )
  1  1  1
Marking
P1
P2
Sequence Done:
T1 T2 T3
```

Otro tipo de simulación automática es la que tenemos en el menú "Execution" y opción "Test Invariant" Si la elegimos, entonces podemos determinar si un T-invariante en particular es válido ó no. Para ello, debemos de introducir el T-invariante a probar separando cada componente por comas (","). Si hubiéramos introducido el T-invariante "1, 1, 1" para probarlo en la RdP de la figura A.9, entonces hubiéramos obtenido el mismo resultado que el de la figura A.10, es decir, que tal T-invariante es válido.

2.1.2 Simulación guiada por el usuario

Cómo ya lo mencionamos anteriormente, existe la posibilidad de que se lleve a cabo una simulación paso a paso guiada por el usuario. Para iniciarla debemos elegir del menú "Step by Step" la opción "Start Step by Step" ó bien pulsar el botón correspondiente, con lo cual se inhabilitan los botones de lugar, transición, arco, abrir archivo y otros; se habilitan los botones "Show Enabled Transitions", "Fire Enabled Transition", "Clauses In Marking" y "Reset Step by Step Simulation" mismas opciones que se encuentran en el menú "Step by Step"

Una vez que hemos hecho lo anterior, debemos oprimir el botón "Show Enabled Transitions" para que las transiciones habilitadas con el marcado actual se cambien a un color rojo y así localizarlas fácilmente. Luego, podemos elegir la opción "Fire Enabled Transition" para disparar una de las transiciones habilitadas. En cualquier momento de la simulación se puede determinar el conjunto de cláusulas asociado con el marcado actual por medio de la opción "Clauses in Marking" del menú "Step by Step"

Ejemplo1

Consideremos la RdP de la figura A.9 y observemos lo que sucede en una simulación paso a paso.

Después de que oprimimos el botón "Start Step by Step Simulation" se habilita el botón "Show Enabled Transitions", si lo pulsamos, entonces la transición T1 cambia a un color rojo (figura A.11.a)). Luego, si pulsamos el botón "Fire Enabled Transition", entonces aparecerá una ventana (figura A.11.b)), en la cual podemos escribir la transición a disparar. Si escribimos T1 y oprimimos el botón "OK", entonces podemos observar que el marcado en los lugares P1 y P2 ha cambiado de 0 a 1.

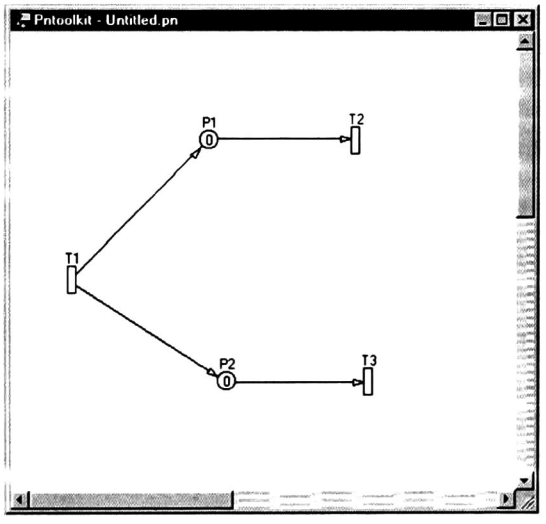


Figura A.11.a) Después de oprimir el botón "Show Enabled Transitions", toda transición habilitada cambia a color rojo

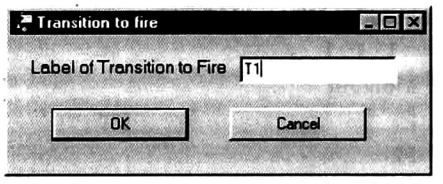


Figura A.11.b) Ventana dónde se introduce la etiqueta de la transición habilitada que se desea disparar

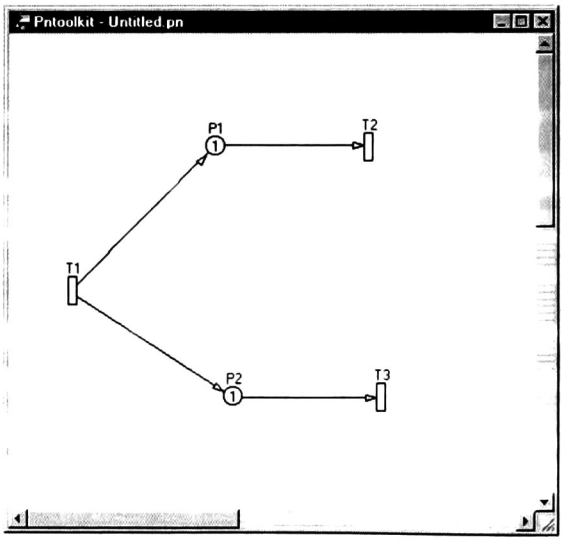


Figura A.11.c) Después de haber disparado T1 se añade una marca a P1 y otra a P2

Si oprimimos el botón "Clauses In Marking", entonces en la ventana de edición aparece lo siguiente:

Mark and Clause associated

<1, 1>< -: P1, P2>

Lo cual nos indica que la cláusula asociada con la marca **<1, 1>** es **<→P1, P2>**.

Sí oprimimos el botón "Show Enabled Transitions", entonces podemos observar que para el marcado actual todas las transiciones se encuentran habilitadas y por tanto podemos

disparar cualquiera de ellas. Si disparamos la transición T2 y pedimos mostrar las cláusulas en el marcado actual aparecerá lo siguiente:

Mark and Clause associated
<2, 1>< -: P2>

Sólo existe una marca en la RdP y su cláusula asociada es **<→P2>**.

Por último, si solicitamos ver las transiciones habilitadas, entonces podemos observar que son T1 y T3. Si elegimos disparar T3, entonces aparece el mensaje que se muestra en la figura A.12, el cual indica que el marcado vacío se ha alcanzado y por tanto el invariante realizado es válido. Además, muestra la secuencia realizada. Con lo que podemos decir que el conjunto de cláusulas asociado con la RdP es insatisfacible.

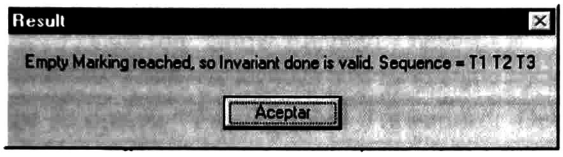


Figura A.12 Resultado de una simulación paso a paso

Después de recibir el mensaje de la figura A.12 y terminar con la simulación paso a paso, debemos pulsar el botón "Reset Step by Step Simulation" ó elegir la opción Reset del menú "Step by Step", para dar por terminada la simulación.

Ejemplo 2.

En este ejemplo, usaremos una RdP un poco más complicada que la del ejemplo anterior. Además, en algunos puntos de la simulación, deberemos elegir las marcas utilizadas para disparar una transición dada, debido a que existe más de una posible combinación de marcas que habilitaron a la transición que deseamos disparar. La RdP a utilizar es la que muestra la figura A.13.

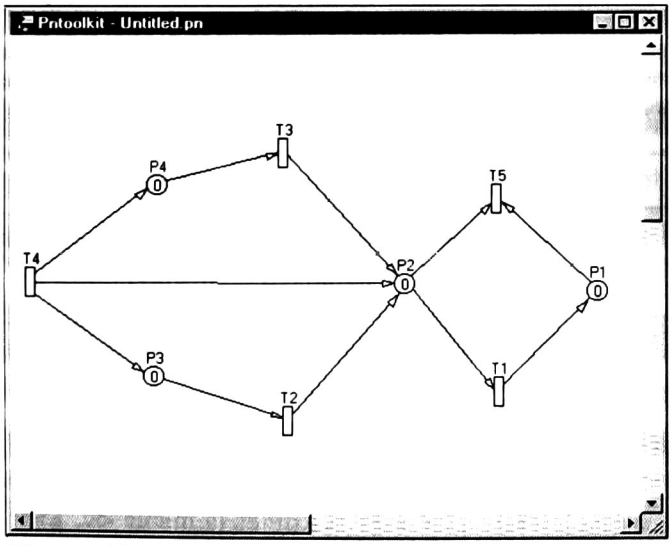


Figura A.13 RdP para realizar una simulación guiada por el usuario

El conjunto de cláusulas asociado con la RdP anterior es {<P2→P1>, <P3→P2>, <P4→P2>, <→P2, P3, P4>, <P1, P2→>}

En la siguiente simulación guiada por el usuario, cada opción elegida será una perteneciente al menú "Step by Step", a menos que se aclare lo contrario. La secuencia de transiciones a realizar en la RdP es σ=T4T4T2T2T3T3T1T1T1T5T5T5.

Iniciemos con la simulación. Para ello elegimos la opción "Start Step by Step" ó pulsamos el botón correspondiente. Luego, elegimos la opción "Show Enabled Transitions" con lo que cambiará a color rojo la transición T4. Para disparar T4 debemos elegir la opción "Fire Enabled Transition" luego en la ventana que recién apareció, escribir T4 y oprimir el botón "OK" Después de realizar lo anterior, el sistema presenta el estado que podemos observar en la figura A.14.

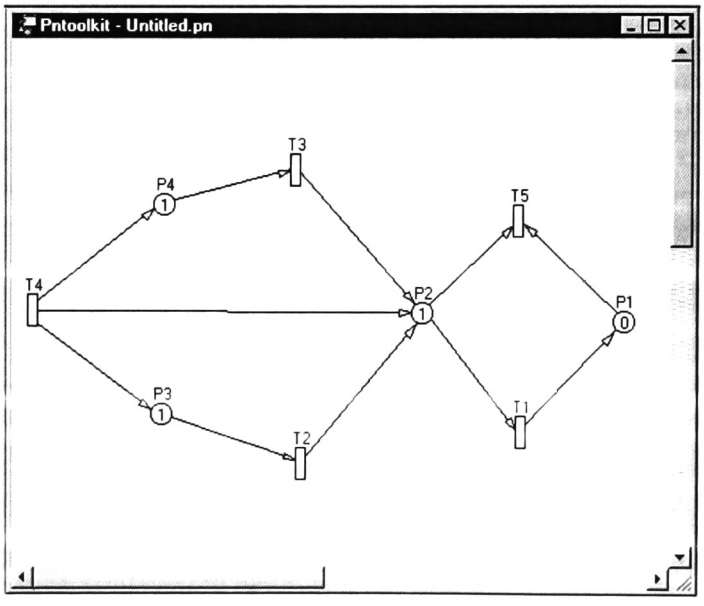


Figura A.14 Estado de la RdP de la figura A.13 después del disparo de T4

En la RdP de la figura A.14 existe una marca en el lugar P2, una marca en el lugar P3 y una marca más en el lugar P4. Si pulsamos el botón derecho del ratón cuando su cursor está sobre el lugar P2, entonces aparecerá una ventana en la cual podemos observar que la referencia de la marca en P2 es <4, 1>. Siguiendo el mismo procedimiento anterior para P3 y para P4, sabremos que la referencia de la marca en tales lugares, también es <4, 1>. Si elegimos la opción "Clauses in Marking", entonces aparecerá lo siguiente en la ventana de edición:

Mark and Clause associated

<4, 1>< -: P2, P3, P4>

Lo cual indica que la única marca en la RdP es la marca <4, 1>, en los lugares P2, P3 y P4. Además, que la cláusula asociada a la marca <4, 1> es <→P2, P3, P4>.

Disparemos la transición T4. Para ello, elegimos la opción "Show Enabled Transitions", con lo cual podemos observar que las transiciones habilitadas en el marcado actual son T1, T2, T3 y T4, luego en la ventana que aparece luego de elegir "Fire Enabled Transition" escribimos T4. Después de lo anterior, si elegimos la opción "Clauses in Marking" podremos observar lo siguiente en la ventana de edición:

Mark and Clause associated
<4, 1>< -: P2, P3, P4>
<4, 2>< -: P2, P3, P4>

Lo cual nos indica que existe una nueva marca <4, 2> en los lugares P2, P3 y P4. Con lo que tenemos una cláusula más y asociada a la marca <4, 2>. Tal cláusula también es <→P2, P3, P4>.

Si elegimos de nuevo la opción "Show Enabled Transitions" sabremos que de nuevo las transiciones habilitadas en el marcado actual son T1, T2, T3 y T4. Disparemos la transición T2. Para ello elegimos la opción "Fire Enabled Transition". Debido a que en el marcado actual existen dos posibles combinaciones de marcas que habilitan a T2, entonces el sistema presenta una ventana para que el usuario elija la combinación de marcas que utilizará T2 para su disparo. Tal ventana es la que mostramos en la figura A.16.a. En este caso, elegimos la marca <4, 1> para disparar T2. Para hacerlo, debemos de pulsar el botón izquierdo del ratón sobre tal marca (figura A.16.b) y luego elegir la opción "Save and Exit" (figura A.16.c).

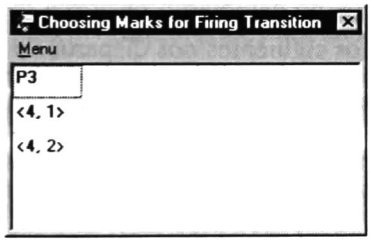


Figura A.16.a) Ventana para elegir la combinación de marcas a utilizar para el disparo de la transición T2

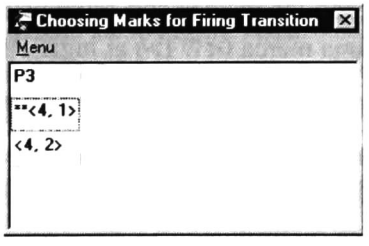


Figura A.16.b) Estado de la ventana de la figura A.16.a, después de elegir la marca <4, 1> para disparar T2

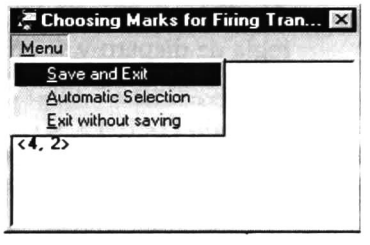


Figura A.16.c) Para llevar a cabo el disparo de T2, debemos de elegir a opción "Save and Exit"

El marcado alcanzado es [∅, {<2, 1>, <2, 1>, <4, 2>}, {<4, 2>}, {<2, 1>, <4, 2>}]. La cláusula asociada con la marca <2, 1> es <→P2, P2, P4> y la asociada con la marca <4, 2> es <→P2, P3, P4>.

Si en el marcado actual disparamos la transición T2 de nuevo, entonces no aparecerá la ventana de la figura A.16, ya que no es necesario que elijamos las marcas a utilizar para el disparo de T2, debido a que sólo una marca habilita a T2 y dicha marca es <4, 2>. Después de lo anterior, el marcado alcanzado es [∅, {<2, 1>, <2, 1>, <2, 2>}, ∅, {<2, 1>, <2, 2>}].

La cláusulas asociada con la marca <2, 1> es <→P2, P2, P4> y la asociada con la marca <2, 2> es <→P2, P2, P4>.

La siguiente transición a disparar es T3. Las marcas que habilitan a T3 son <2, 1> y <2, 2>. Cómo existen dos posibles marcas a utilizar para el disparo de T3, al elegir disparar T3 debe de aparecer la ventana de la figura A.16 para que elijamos una de las dos marcas (<2, 1> ó <2, 2>). En este caso, elegimos la marca <2, 1> para el disparo de T3. Luego de realizar lo anterior observamos que el marcado alcanzado es [∅, {<3, 1>, <3, 1>, <3, 1>, <2, 2>, <2, 2>}, ∅, {<2, 2>}], que la cláusula asociada con la marca <3, 1> es <→P2, P2, P2> y la asociada con la marca <2, 2> es <→P2, P2, P4>.

Las transiciones habilitadas en el marcado actual son T1, T3 y T4. Elegimos disparar T3, al no haber más de una combinación de marcas que habilitaran a T3, no aparece la ventana de la figura A.16. El marcado alcanzado es [∅, {<3, 1>, <3, 1>, <3, 1>, <3, 2>, <3, 2>, <3, 2>}, ∅, ∅]. La cláusula asociada con la marca <3, 1> es <→P2, P2, P2> y la asociada con la marca <3, 2> es <→P2, P2, P2>.

Las únicas transiciones habilitadas en el marcado actual son T1 y T4. Existen 6 posibles marcas a usar para el disparo de T4, aunque realmente son dos combinaciones, ya que 3 de ellas tienen referencia <3, 1> y las otras 3 referencia <3, 2>. Para realizar la secuencia σ, debemos disparar la transición T4 tres veces. Al disparo de T4 elegimos la marca con referencia <3, 1> con ello obtenemos el marcado [{<1, 1>}, {<3, 1>, <3, 1>, <3, 2>, <3, 2>, <3, 2>}, ∅, ∅]. En tal marcado las marcas con referencia <3, 1> no cambiaron debido a la regla de disparo y se añadió una marca (<1, 1>) al lugar P1. En los siguientes dos disparos de T4, elegimos de nuevo la marca <3, 1> para cada uno de ellos. Con lo cual obtenemos el marcado [{<1, 1>, <1, 2>, <1, 3>}, {<3, 2>, <3, 2>, <3, 2>}, ∅, ∅]. Las cláusulas asociadas con las marcas anteriores son <→P1>, <→P1>, <→P1> y <→P2, P2, P2>.

Las transiciones habilitadas en el marcado actual son T1, T4 y T5, pero debemos de disparar la transición T5 tres veces para completar la secuencia σ.

La primera vez que disparamos T5 elegimos la combinación de marcas <1, 1> y <3, 2>. Con lo cual obtenemos el marcado [{<1, 1>, <1, 3>}, {<3, 2>, <3, 2>}, ∅, ∅]. Las cláusulas asociadas con las marcas anteriores son <→P1>, <→P1> y <→P2, P2>.

La segunda vez que disparamos T5 elegimos la combinación de marcas <1, 2> y <3, 2>. Con lo cual obtenemos el marcado [{<1, 3>}, {<3, 2>}, ∅, ∅]. Las cláusulas asociadas con las marcas anteriores son <→P1> y <→P2>.

La tercera y última vez que disparamos T5 no es necesario que elijamos las marcas utilizadas para su disparo ya que sólo existe una combinación (<1, 3> y <3, 2>). Al realizar este último disparo llegamos al marcado vacío con lo que aparece la ventana de la figura A.17, la cual nos indica que el marcado vacío ha sido alcanzado y en consecuencia el conjunto de cláusulas asociado con la RdP es insatisfacible. Tal ventana muestra además la secuencia realizada.

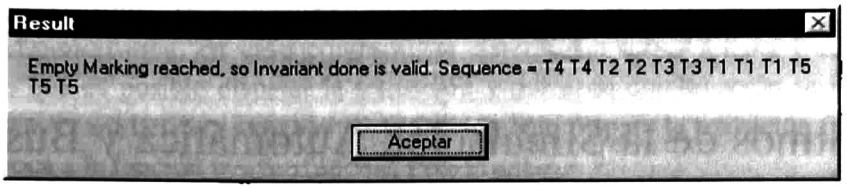


Figura A.17 Ventana indicando que el marcado vacío ha sido alcanzado y la secuencia realizada

Hemos terminado la simulación y para indicarlo, debemos elegir la opción "Reset" u oprimir el botón correspondiente.

2.2 Búsqueda de Modelos

Para llevar a cabo la búsqueda de modelos, primero debemos de tener dibujada una RdP. Posteriormente, sólo elegimos la opción "Models" del menú "Execution" para iniciar la búsqueda de modelos.

Al terminar de realizar la búsqueda de modelos, aparecerá una ventana la cual especifica el modelo para el conjunto de cláusulas asociado a la RdP o bien que no existe un modelo. Para el caso de encontrar un modelo, se especifica cada átomo seguido de un signo "=" y luego el valor de verdad asignado a tal átomo.

Ejemplo 3.

Supongamos que tenemos la RdP de la Figura A.18.

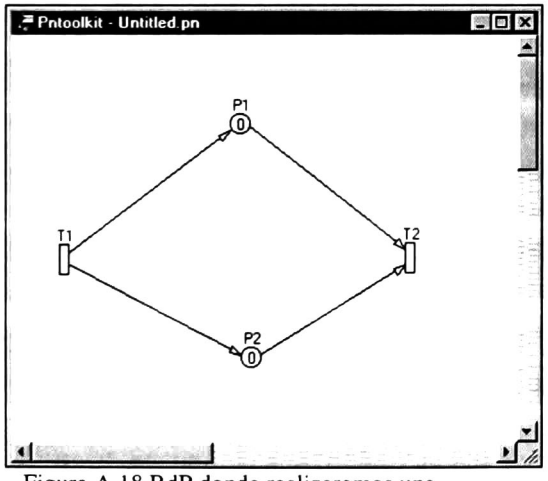


Figura A.18 RdP donde realizaremos una búsqueda de modelos

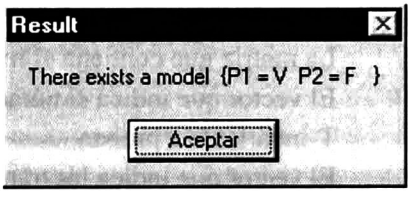


Figura A.19 Modelo para el conjunto de cláusulas asociado con la RdP de la Figura A.18

Después de que elegimos del menú "Execution" la opción "Models" aparece la ventana de la Figura A.19. En tal ventana, aparece el valor de verdad asignado a cada átomo que contempla el modelo encontrado.

Apéndice B

Algoritmos de la Simulación Automática y Búsqueda de Modelos

Para la implementación del método que es objeto esta tesis, utilizamos el compilador C++ Builder de Borland, versión 3.0. No vamos a explicar el código, sólo presentaremos a modo de funciones los algoritmos utilizados para llevar a cabo una simulación automática (determinar si un conjunto de cláusulas es insatisfacible a partir de su RdP asociada) y la búsqueda de modelos (en caso de que el conjunto de cláusulas sea satisfacible).

Antes de cada algoritmo (simulación automática y búsqueda de modelos) presentamos una tabla explicando la notación usada para que sea más fácil su seguimiento. Asumimos que el lector esta familiarizado con los bucles o ciclos, así como con las decisiones.

Simulación Automática

Para la simulación automática se calculan primero los T-Invariantes, debido a la importancia del algoritmo que los calcula, hemos designado un apéndice (Apéndice C) para la explicación del mismo, describimos además, algunas opciones en cada paso del algoritmo y el porqué eligimos una en particular.

Notación

A	La matriz de incidencia de la RdP
TI	La matriz que contiene los T-invariantes de *A*. *TI*[*k*] denota el *k*-ésimo T-invariante en *TI*
M	La matriz que contiene el marcado actual de *A* (inicialmente es el marcado vacío)
ID	El vector que indica el número de veces que se ha disparado cada transición del T-invariante a probar
ET	El vector que indica las transiciones habilitadas
CM	La matriz que almacena la combinación de marcas que habilitan a una transición

Antes de iniciar con la especificación de las funciones, hablaremos un poco acerca del flujo de las mismas, es decir, del orden en que se invocan.

La función que inicia todo es la función `SimulaciónAutomática`, donde se calculan los T-Invariantes, luego con cada T-Invariante que se calculó y con T-Invariantes que son suma de algunos de ellos se invoca a la función `EvolucionConTInvariante`. Esta funcion determina si se han disparado todas las transiciones de algun T-Invariante (con lo que hemos encontrado un T-Invariante válido) ó de no ser así, primero calcula todas las transiciones habilitadas en el marcado actual y luego, con cada transicion habilitada invoca a la función `TinvarianteVálido`. En la función TinvarianteVálido se dispara la transición, si es una

transición fuente sólo se actualiza el marcado y se invoca a la función EvoluciónConTInvariante, en caso contrario, se determinan todas las combinaciones de marcas con las que es posible disparar la transición y por cada combinación se invoca a la función DispararTransiciónConMarcas. Esta función sólo dispara la transición (actualiza el marcado) e invoca a la función EvolucionConTInvariante.

```
Función SimulaciónAutomática
    Se calculan los T-invariantes en A y se almacenan en TI
    Se añaden a TI los vectores que son suma de T-invariantes en TI
    Para k=1 hasta k= Número de T-invariantes en TI
        Si EvoluciónConTInvariante(TI[k], M, ID) entonces
            T-invariante válido
        FSi
        k=k+1
    FPara
FFunción

Función EvoluciónConTInvariante(TI[k], M, ID)
    Si TI[k] = ID  -Se ha realizado el k-ésimo T-invariante en TI
        regresar Verdadero a la función invocadora
    Fsi
    Sino
       Calcular las transiciones habilitadas en A con el marcado actual e
       indicarlas en ET
       Para cada transición habilitada T en ET, perteneciente a TI[k] y que
       no se ha disparado el número de veces que indica TI[k]
           Si TinvarianteVálido(TI[k], M, ID, T)
               regresar Verdadero a la función invocadora
           FSi
           Sino
               regresar Falso a la función invocadora
           Fsino
       FPara
    FSino
FFunción

Función TinvarianteVálido(TI[k]  M, ID, T)
    Si T es una transición fuente entonces
        Se incrementa en 1 el número de veces que se ha disparado T en ID
        Se actualiza el marcado en M
        Si EvoluciónConTInvariante(TI[k]  M, ID) entonces
            regresar Verdadero a la función invocadora
        FSi
        Sino
            regresar Falso a la función invocadora
```

```
        FSino
    FSi
    Sino
        Se almacena en CM las combinaciones de marcas que habilitan a T
        Para n=1 hasta n= número de combinaciones de marcas en CM
        Si DispararTransiciónConMarcas(TI[k], M, ID, T, CM[n]) entonces
            regresar Verdadero a la función invocadora
        FSi
        Sino
            regresar Falso a la función invocadora
        Fsino
        n=n+1
    FSino
FFunción

Función DispararTransiciónConMarcas(TI[k], M, ID, T, CM[n])
    Diparar T y actualizar marcado en M
    Si EvoluciónConTInvariante(TI[k], M, ID)
        regresar Verdadero a la función invocadora
    FSi
    Sino
        regresar Falso a la función invocadora
    FSino
FFunción
```

Búsqueda de Modelos

Para una mejor comprensión de los algoritmos que vamos a presentar, recomendamos leer primero el método para la búsqueda de modelos (capítulo IV).

El algoritmo realiza recorridos en la matriz de incidencia de la red de Petri. Si encontramos un camino abierto (definición 4.1) en la red de Petri podemos decir que el conjunto es satisfacible y presentar un modelo para tal conjunto.

Notación

A	La matriz de incidencia de la Red de Petri
NoN	El Vector que indica las posiciones de elementos no nulos de alguna columna de *A*
R	Vector que representa el camino que se está recorriendo
Col	Columna actual

La función que inicia todo el proceso es la función `EncontrarModelo`, en la cual se calculan todos los elementos no nulos de la primer columna en la matriz de incidencia de la red de Petri que estemos analizando, luego, por cada elemento no nulo invoca a la función `Recorrido`. Esta función primero determina si se encuentra analizando la última columna de

la matriz de incidencia, de ser así, por cada elemento no nulo en dicha columna comprueba si representa un camino abierto (hemos encontrado un modelo) ó un camino cerrado, en caso contrario, por cada elemento no nulo en la misma columna se invoca a si misma (llamado recursiva).

```
Función EncontrarModelo
    Se calculan los elementos no nulos de la primera columna de A, se
    almacenan en NoN
    Para k=1 hasta k=número de elementos de NoN
        Se almacena en R[1] el k-ésimo elemento de NoN
        Si Recorrido(R, 1)
            Modelo encontrado en A, indicado en R
        FSi
        k=k+1
    FPara
FFunción

Función Recorrido(R, Col)
    Si Col=n
    Se calculan los elementos no nulos de la columna Col de A y se almacenan
    en NoN
    Para k=1 hasta k=número de elementos de NoN
        Se almacena en R[Col] el k-ésimo elemento de NoN
        Si R representa un camino abierto entonces
            Regresar Verdadero a la función invocadora
        FSi
        k=k+1
    Fpara
        Regresar Falso a la función invocadora
    FSi
    Sino
       Se calculan los elementos no nulos de la columna Col de A y se
       almacenan en NoN
       Para k=1 hasta k=número de elementos de NoN
           Se almacena en R[Col] el k-ésimo elemento de NoN
           Si R representa un camino abierto entonces
               Si Recorrido(R, k+1)
                   Regresar Verdadero a la función invocadora
               FSi
           FSi
           k=k+1
       Fpara
       Regresar Falso a la función invocadora
    FSino
FFunción
```

Apéndice C

Algoritmo para el Cálculo de P-Invariantes

Los P-invariantes (T-invariantes) son vectores no negativos anuladores izquierdos (derechos) de la matriz de incidencia de una Red de Petri. La importancia de estos vectores reside en su utilidad para el análisis de propiedades.

Los algoritmos conocidos para el cálculo de semiflujos elementales (P y T-Invariantes) en Redes de Petri representan un nuevo redescubrimiento, con diversas mejoras técnicas con respecto a la clase de problemas que tratan, del método de eliminación de variables de Fourier-Motzkin.

Uno de los problemas fundamentales de estos algoritmos radica en su complejidad. Diversos métodos y reglas se han estudiado y comparado para paliar en cierta medida este problema. Cómo resultado, presentamos el algoritmo mejorado que posee gran eficiencia y robustez frente a modelos de Redes de Petri procedentes del mundo real.

Aunque en el texto solo hablamos de P-invariantes, para calcular los T-invariantes de una Red de Petri únicamente debemos de considerar la transpuesta de su matriz de incidencia.

Introducción

La idea de base consiste en eliminar variables del sistema añadiendo a éste todas las desigualdades que resultan de combinaciones positivas de pares de desigualdades del sistema original. El método procede eliminando todas las variables y memorizando para cada desigualdad final los coeficientes de la combinación lineal positiva de las filas originales que la ha generado. Estos coeficientes son los p-invariantes.

Una de las dificultades del método estriba en el gran número de desigualdades añadidas en el proceso de eliminación de variables. Las distintas variantes que aparecen en la literatura persiguen como objetivo común el controlar el crecimiento de nuevas desigualdades eliminando las redundantes.

El algoritmo básico del cual se parte es el siguiente:

(1) $B := A$, {m es el número de lugares de la RdP}

$Y := I_m$ {I_m es una matriz identidad de dimensión n}

(2) Para i:=1 hasta n hacer {n es el número de transiciones de la RdP}

2.1 Añadir a la matriz $[Y \mid B]$ todas las filas que resulten de la combinación lineal positiva de pares de filas de $[Y \mid B]$ y que anulen la i-ésima columna de B.

2.2 Eliminar de $[Y \mid B]$ las filas en las que la i-ésima columna de B sea no nula.

(3) Las filas de la matriz Y son P-invariantes de la RdP Entre éstas se encontrarán todos los P-invariantes mínimos (tras una eventual transformación de las filas de Y en P-invariantes canónicos).

Cada fila de la matriz Y memoriza los coeficientes de la combinación lineal de filas de la matriz A que ha generado la fila de B del mismo índice. En el paso (3) del algoritmo las filas de B son nulas y por tanto cada fila Y_i es un P-invariante: $Y_i^T A = 0$.

Consideraciones generales sobre la implementación y evaluación de prestaciones.

Más adelante analizamos algunas variantes del algoritmo anterior. Dichas variantes se obtienen incorporando a los algoritmos diferentes soluciones algorítmicas de las que, a priori, se puede esperar una mejora de las prestaciones. En la discusión posterior sobre la implementación de cada una de las soluciones partimos del algoritmo básico al que se incorporan de forma gradual, aquellas que se han mostrado eficaces en puntos anteriores al discutido. De esta forma, el algoritmo será mejorado paso a paso conforme avance la exposición (mejora incremental)

Implementación eficiente del algoritmo de cálculo de los P-invariantes partiendo de la matriz de flujo

Aquí analizamos diversas variantes del algoritmo básico, todas ellas elegidas de entre varias [Treves, 1986] como la que presenta mejores prestaciones.

En secciones sucesivas las siguientes cuestiones se aplicarán para obtener el "mejor" algoritmo de este tipo: la ubicación de la comparación de soportes para eliminar P-invariantes no mínimos; heurísticas para la selección de la columna a anular; la eliminación previa de un conjunto máximo de columnas linealmente dependientes de la matriz de flujo; la clasificación de las filas de la matriz cuya columna k-ésima es cero, negativa o positiva, respectivamente; tests rápidos de no minimalidad de P-invariantes. Al final resumimos las conclusiones sobre la implementación del algoritmo.

Ubicación de la comparación de soportes

El problema que planteamos es dónde ubicar la eliminación de semiflujos no mínimos basada en la comparación de sus soportes. Las alternativas a considerar son:

1. En el momento de generar una nueva fila
2. Al final de la anulación de una columna o conjunto de columnas y

3. Al final del algoritmo

Para mejorar la eficiencia del algoritmo memorizamos el soporte de cada fila de Y como un conjunto de índices. Así, el soporte de una nueva fila de Y es la unión de los soportes de las filas de Y que la generan. Por tanto, podemos realizar la comparación de soportes sin cálculos previos adicionales.

De acuerdo a resultados presentados en [Treves, 1986] no existe una variante óptima. Adoptamos la opción 1) como variante a considerar en la solución final, por ser la más robusta, es decir, en los casos desfavorables la diferencia de tiempo con los otros no es excesiva. Sin embargo, en los casos que se generan gran cantidad de P-invariantes, sobre todo no mínimos, presenta una ventaja significativa respecto a las otras dos variantes.

Heurística para la selección de la columna a anular

En este apartado presentamos una heurística que persigue limitar el número de filas a añadir en una iteración, al seleccionar la columna a anular que genere menos filas nuevas.

Sean π_k y φ_k el número de elementos positivos y negativos, respectivamente, de la columna k de B que deseamos anular. El número de nuevas filas a añadir, combinando pares de filas, es $\pi_k\varphi_k$. Las filas $\pi_k + \varphi_k$ utilizadas para anular la columna k se eliminan al final de la iteración. Se denomina factor de expansión al número neto de filas que se añaden en la anulación de la columna k: $F(k) := \pi_k\varphi_k - (\pi_k + \varphi_k)$.

Por tanto, a priori, puede ser una buena política seleccionar como columna a anular la de menor factor de expansión.

Las variantes de las cuales podemos elegir son:

1. Seleccionar la primera columna con factor de expansión negativo, o si no existe la de menor valor $\pi_k\varphi_k$.
2. Las columnas se seleccionan en el orden que aparecen en la matriz de flujo.
3. Seleccionar primero las columnas con mayor factor de expansión.
4. Seleccionar una columna con el menor factor de expansión.

En [Martínez, 1984] se elige la primera opción ya que permite reducir el tiempo de búsquedas secuenciales por las listas que implementan la matriz.

Eliminación previa de un conjunto máximo de columnas linealmente dependientes de la matriz de flujo A.

El número de columnas que se anulan activamente es igual al rango r de A. El resto de las columnas, $n - r$, se anulan durante la anulación de otras columnas. Por tanto, la eliminación

inicial de n r columnas linealmente dependientes aparece como una posible mejora, ya que reducimos, en tiempo polinomial, la ocupación de memoria hasta r columnas. Los tiempos de combinación de filas es de esperar que también disminuyan. Para la implementación de esta mejora potencial consideramos las variantes:

1. Trabajamos directamente con la matriz A.
2. Triangularización de A por columnas. Obtenemos una matriz con r columnas linealmente independientes a la que aplicamos el algoritmo.
3. Selección de r columnas linealmente independientes de A. Elegimos las columnas con menor factor de expansión.

En [Colom, 1989] se recomienda utilizar la primer variante, ya que las otras no presentan una ventaja clara sobre la elegida.

Clasificación de las filas de la matriz

Podemos clasificar a las filas de $U = [\ B \mid Y\ ]$ en tres matrices $U_{(0)}$, $U_{(+)}$ y $U_{(-)}$ que contienen las filas de U, en las que la columna k de B es nula, positiva y negativa, respectivamente. Para anular la columna k de B, combinamos cada fila de $U_{(+)}$ con cada fila de $U_{(-)}$. Añadimos las nuevas filas a la matriz $U_{(0)}$. Una vez que anulamos la columna k, $U_{(0)}$ se convierte en la matriz U inicial para proceder a eliminar la columna k+1. Previamente, eliminamos las matrices $U_{(+)}$ y $U_{(-)}$.

Esta solución reduce el número de tests para seleccionar dos filas cuyos valores en la columna k de B sean no nulos y de signo contrario.

Eliminación previa de las transiciones con un único lugar de entrada y un único lugar de salida: Eliminación de secuencias.

Las transiciones tales que $|\bullet t\ | = |\ t\bullet\ | = 1$ representan secuencias puras. La eliminación inicial de estas transiciones presenta como ventajas:

1. Reducir la matriz en una fila [$F(k) := \pi_k\varphi_k - (\pi_k + \varphi_k) = 1 - 2 = -1$].
2. Después de anular la columna no es necesario realizar la comparación de soportes. En efecto, al generar un P-invariante intermedio cuyo soporte es el único que contiene a los dos lugares $\bullet t = t\bullet$.

Test rápido de no minimalidad.

La caracterización de los P-invariantes no mínimos mediante el rango de ciertas submatrices de A permite circunscribir la determinación de la minimalidad al propio P-

invariante. Por tanto, la aplicación de dicha caracterización puede resultar computacionalmente interesante. Para obviar los cálculos asociados al cálculo del rango de estas submatrices recurrimos a mayorantes, que dan sólo condiciones suficientes de minimalidad.

Corolario: Un p-invariante Y es no mínimo si cardinal($\|Y\|$) > mayorante_del_rango + 1.

Este es un filtro previo para eliminar P-invariantes no mínimos con relación a la comparación de soportes. La bondad de este filtro depende del mayorante del rango considerado.

En [Martínez and Silva, 1981] el mayorante del rango es el número de columnas no nulas de la submatriz formada por las filas de A correspondientes a los lugares que pertenecen al soporte del P-invariante. La aplicación del test en pasos intermedios del algoritmo, considera la matriz que se forma por las columnas que se anularon activamente en lugar de A.

La implementación de la regla anterior admite dos soluciones:

1. Para cada fila memorizamos la unión de los soportes de las filas de A indexadas por el soporte del P-invariante. Para una nueva fila obtenemos a partir de la unión de los que se memorizaron para las dos filas que la generan.
2. Para cada fila calculamos la unión de los soportes de todas las filas de A indexadas por el soporte del nuevo P-invariante.

Posteriormente, en ambos casos, calculamos el cardinal del conjunto obtenido restringido a las columnas anuladas activamente hasta ese momento. Este valor será el mayorante del rango.

En [Colom, 1989] se elige la primera solución ya que permite obtener mejores prestaciones temporales a costa de un incremento de memoria pequeño.

Conclusiones y presentación del algoritmo para la búsqueda de P-invariantes.

Las variantes que elegimos son:

- Comparación de soportes antes de generar una nueva fila como combinación lineal de dos filas anteriores.
- Selección de la columna a anular según la regla: primera columna con factor de expansión negativo, o si no existe la de menor valor $\pi_k\varphi_k$.
- Clasificación de las filas de la matriz antes de anular una columna k en 3 matrices que agrupan las filas cuya columna k es cero, negativa o positiva, respectivamente.
- Eliminación previa de secuencias (transiciones tal que $|{\bullet}t| = |t{\bullet}| = 1$)

- Test rápido de minimalidad utilizando como mayorante del rango el propuesto en [Martínez, 1981].

Antes de presentar el algoritmo final, introducimos la notación utilizada en el mismo.

Notación

A La matriz de flujo de la RdP (dimensiones mxn)

I La matriz identidad (dimensiones mxm)

U^k La matriz $[\ B^k \mid Y^k\ ]$ donde
para $k = 0$, $B^k = A$ y $Y^k = I$
para $k > 0$, B^k es la matriz resultante de haber anulado k columnas de A
Y^k es la matriz que memoriza los coeficientes de combinaciones lineales de filas de A

$U_{(-)}$ La matriz que contiene las filas de U^{k-1} cuya columna k-ésima es negativa

$U_{(+)}$ La matriz que contiene las filas de U^{k-1} cuya columna k-ésima es positiva

π_k, φ_k El número de positivos y negativos en la columna k de B^k

```
0.   Anular todas las columnas k de U⁰ en las que π_k=φ_k=1
1.   Seleccionar la primera columna no nula de U⁰ tal que π_kφ_k-(π_k+φ_k)<0 ó una
     columna k que minimice π_kφ_k.
2.   Mientras k<>nulo hacer
3.      Mover las filas j de U^k-1 tales que B^k-1[j,k]=0 a U^k
4.      Mover las filas j de U^k-1 tales que B^k-1[j, k]<0 a U_(-)
5.      Mover las filas j de U^k-1 tales que B^k-1[j, k]>0 a U_(+)
6.      fila1:= índice_primera_fila (U_(-))
7.      Mientras que fila1 <> nulo hacer
8.         fila2:=índice_primera_fila (U_(+))
9.         Mientras que fila2 <> nulo hacer
10.           ||Y||=||Y_(-)[fila1, -]||∪|| Y_(+)[fila2, -]
              {cálculo del soporte del nuevo p-invariante Y}
11.           Si cardinal(||Y||)<=mayorante_rango+1 Entonces
12.              Si ||Y|| no incluye el soporte de otro p-invariante de Y^k
                 o Y_(+) o Y_(-) Entonces
13.                 Añadir αU(-)[fila1, -]+βU(+)[fila2, -] a U^k
                    Dónde α=|B(+)[fila2, k]| y β=|A(-)[fila1, k]|
14.                 Dividir nueva fila por el m.c.d. de elementos no
                    nulos
15.                 Memorizar ||Y||
                 FSi
              FSi
16.           fila2:=índice_siguiente(fila2)
           Fmientras que
17.        fila1:=índice_siguiente(fila1)
        Fmientras que
18.  Eliminar todas las filas de U(+) y U(-)
19.  Seleccionar la primera columna no nula de U⁰ tal que π_kφ_k-(π_k+φ_k)<0 ó una
     columna k que minimice π_kφ_k.
     FMientras que
```

Índice Alfabético

Bibliografía

[Bibel, 1993] Wolfgang Bibel. *Deduction, Automated Logic.* Academic Press, 1993.

[Colom, 1989] José Manuel Colom Piazuelo. *Análisis Estructural de Redes de Petri, Programación Lineal y Geometría Convexa, Tesis Doctoral.* Universidad de Zaragoza, 1989.

[Desel and Esparza, 1995] Jorg Desel and Javier Esparza. *Free Choice Petri Nets.* Cambridge Tracts in Theoretical Computer Science, Great Britain, 1995.

[Fitting, 1995] Melvin Fitting. *First-Order Logic and Automated Theorem Proving.* Graduate Texts in Computer Science, Springer Verlag, 1995.

[Jeffrey et *al.*, 1996] John Jeffrey, Jorge Lobo, and Tadao Murata. *A High Level Petri Net for Goal-Directed Semantics of Horn Clause Logic.* IEEE Transactions on Knowledge and Data Engineering, Vol. 8, No. 2, April 1996.

[Lautenbach, 1985] Kurt Lautenbach. *On Logical and Linear Dependencies.* Sankt Augustin, Germany, GMD Rep. 147, 1985.

[Martínez, 1984] J. Martínez. *Contribución al Análisis y Modelado de Sistemas Concurrentes mediante Redes de Petri, Tesis Doctoral.* Universidad de Zaragoza, 1984.

[Martínez and Silva, 1981] J. Martínez and M. Silva. *A simple and fast algorithm to obtain all invariants of a generalized Petri Net, Second European Workshop on Application and Theory of Petri Nets.* Bad Honnef, September, pp. 411-422.

[Murata, 1989] Tadao Murata. *Petri nets: Properties, Analysis and Applications.* IEEE, vol 77, No. 4, pp. 541-580, 1989.

[Nerode and Shore, 1997] Anil Nerode and Richard A. Shore. *Logic for Applications.* Springer Verlag, New York, 1997.

[Peterka and Murata, 1989] George Peterka and Tadao Murata. *Proof Procedure and Answer Extraction in Petri Net Model of Logic Programs.* IEEE Transactions on Software Engineering, Vol. 15, No. 2, February 1989.

[Silva, 1985] Manuel Silva. *Las Redes de Petri en la Automática y la Informática.* Ed. AC, Madrid, España.

[Sinachopoulos, 1987] Antonia Sinachopoulos, *Derivation of a contradiction by resolution using Petri nets.* Petri Net Newsletter, vol. 26, pp. 16-29, Apr. 1987.

[Socher and Johann, 1996] Rolf Socher Ambrosius and Patricia Johann. *Deduction Systems.* Graduate Texts in Computer Science, Springer Verlag, 1996.

[Treves, 1986] N. Treves. *Le Calcul d'invarriants dans le Réseaux de Petri a Predicats Transitions Unaires, Thése de Docteur de 3éme cycle.* Universidad de Paris-Sud, Centre d'Orsay, 1986.

CENTRO DE INVESTIGACION Y DE ESTUDIOS AVANZADOS DEL IPN
UNIDAD GUADALAJARA

El Jurado designado por la Unidad Guadalajara del Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional, aprobó la tesis: "**Prueba de validez de Argumentos Lógicos y Búsqueda de Modelos utilizando Redes de Petri**" del Sr. **José Salvador Navarro Contreras,** el día 11 de Diciembre de 2001.

EL JURADO

Dr. Manuel Edgardo Guzmán Rentería
Investigador Cinvestav 3A
CINVESTAV DEL IPN
Guadalajara

Dr. Arturo del Sagrado Corazón Sánchez Carmona
Investigador Cinvestav 3B
CINVESTAV DEL IPN
Guadalajara

Dr. Raúl Ernesto González Torres
Investigador Cinvestav 2C
CINVESTAV DEL IPN
Guadalajara

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