UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE CIENCIAS

PROGRAMAS ORIENTADOS A ESTADOS: SU RELACION CON PROGRAMAS LOGICOS Y LENGUAJES FORMALES

T E S I S

QUE PARA OBTENER EL TITULO DE

MATEMATICO

P R E S E N T A :

MAURICIO ALVAREZ MANILLA VILLANUEVA

MEXICO, D.F. 1995

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

VNIVERSIDAD NACIONAL
AVFNºMA DE
MEXICO

M. EN C. VIRGINIA ABRIN BATULE
Jefe de la División de Estudios Profesionales
Facultad de Ciencias
P r e s e n t e

Los abajo firmantes, comunicamos a Usted, que habiendo revisado el trabajo de Tesis que realiz(ó)ron el pasante(s) MAURICIO ALVAREZ MANILLA VILLANUEVA

con número de cuenta 9052392-6 con el Título:

PROGRAMAS ORIENTADOS A ESTADOS: SU RELACION CON PROGRAMAS LOGICOS Y LENGUAJES FORMALES.

Otorgamos nuestro **Voto Aprobatorio** y consideramos que a la brevedad deberá presentar su Examen Profesional para obtener el título de MATEMÁTICO

GRADO	NOMBRE(S)	APELLIDOS COMPLETOS	FIRMA
DOCTOR	DAVID ARTURO	ROSENBLUETH LAGUETTE	
Director de Tesis			
M. en C.	JOSE ALFREDO	AMOR MONTAÑO	
M. en C.	CARLOS	TORRES ALCARAZ	
M. en C.	ELISA	VISO GUROVICH	
Suplente			
MAT.	CARLOS	VELARDE VELAZQUEZ	
Suplente			

ॐ गुरु ॐ

A Gurumayi y al linaje de Siddha Yoga
con gratitud.

Agradecimientos

Quiero aprovechar este espacio para agradecer no sólo a quienes cooperaron en la elaboración de esta tesis sino también a aquellas personas que contribuyeron en mi formación durante la carrera.

En primer lugar y muy especialmente quiero agradecer a David Rosenblueth quien aceptó dirigir esta tesis y desde mi llegada al IIMAS me brindó innumerables facilidades, además de su apoyo, atenciones y tiempo, siempre con la mejor disposición y auténtico interés; sin dejar de mencionar todas las revisiones, correcciones y sugerencias que hizo a este trabajo y la paciencia que me ha tenido en los periodos en que mis ideas no han sido del todo claras. También quiero agradecer a Carlos Velarde quien revisó muy detalladamente esta tesis; proporcionó la información y archivos necesarios para la incorporación de las gráficas; además de facilitarme su cubículo y computadora mientras se encontraba de sabático. A Julio Peralta y Enrique Silva les doy gracias por su ayuda con la tipografía y los buenos ratos que hemos pasado juntos.

Al M. en C. Angel Carrillo y al Dr. Gustavo Ayala quiero agradecerles sus consejos y el apoyo e interés constantes que han tenido por mi formación desde que los conozco.

A los siguientes profesores quiero agradecerles todas las cosas que aprendí de ellos, tanto académicas como no académicas. Sus cursos fueron la base de mi carrera y además aprecio mucho su calidad como personas: Angel Carrillo, José Alfredo Amor, Carlos Torres, Mario Magidin, Elisa Viso, Luis Briseño, el Dr. Alberto Barajas y Javier Páez.

A José Alfredo Amor, Carlos Torres y Elisa Viso les agradezco además haber aceptado ser sinodales de esta tesis.

Quiero agradecer a Martha Sánchez y al Departamento de Supercómputo de la Dirección General de Servicios de Cómputo Académico por la oportunidad que tuve de participar en el plan de becarios y el conocimiento que ahi adquirí.

Por último quiero agradecer con mucho cariño a mis padres Ethel y José Manuel que me han apoyado todos estos años emocional y económicamente en todas (o casi todas) mis locuras y a quienes debo muchas de las cosas que hoy disfruto.

Indice

Introducción 1

PARTE I

1 Gramáticas y Lenguajes Formales 4

1.1 Definiciones Básicas 4

1.2 Clasificación de Chomsky 9

2 Resultados sobre Programas Orientados a Estados 12

2.1 Funciones Asociadas y Gramáticas Adjuntas 12

2.2 Programas Asociados a Gramáticas 20

PARTE II

3 Analizadores por Cartas 25

4 Lógica de Primer Orden 35

4.1 Definiciones Básicas 35

4.2 Modelos de Herbrand 45

4.3 Sistemas Formales 47

5 Programación Lógica 50

5.1 Definiciones Básicas 50

5.2 Unificación 53

5.3 Teorema de Herbrand 55

5.4 Resolución SLD 60

5.5 Completud y Validez de la Resolución SLD para Programas Definidos 65

5.6 Refutaciones Izquierdas 67

6 Analizadores por Cartas como Sistemas Formales para Programas Cadena **70**

PARTE III

7 Relación entre Programas Cadena y Programas Orientados a Estados **90**

Conclusiones **99**

Bibliografía **101**

Introducción

La teoría de la programación lógica surge a principios de los años setenta a raíz de las investigaciones en demostración automática de teoremas e inteligencia artificial. Buena parte de la demostración automática se centraba en la construcción de algoritmos eficientes que permitieran obtener consecuencias lógicas de un conjunto de fórmulas de un lenguaje de primer orden. El problema principal al que se enfrentaban aquellos trabajos, era la fuerte explosión combinatoria que ocurría en los espacios de búsqueda. De modo que los demostradores automáticos resultaban ineficientes y computacionalmente caros [Llo87, JLM86].

En 1965, Robinson publica un artículo clave [Rob65] donde propone lo que él denomina: el *principio de resolución*. Restringiéndose a un tipo particular de fórmulas lógicas llamadas cláusulas, Robinson demuestra que el principio de resolución además de ser computacionalmente eficiente, es un sistema formal completo y correcto para la lógica de primer orden. En 1972, basándose en el principio de resolución y restringiéndose (todavía más) a las llamadas cláusulas definidas, Kowalski propone utilizar la lógica de primer orden como lenguaje de programación [Kow74], dando lugar al surgimiento de la programación lógica. En 1973, Colmerauer implementa el primer sistema Prolog.

Si bien, desde el punto de vista de la lógica matemática y la demostración automática de teoremas, restringir nuestra atención a conjuntos finitos de cláusulas definidas y sus consecuencias lógicas, es quedarse con muy poco, desde el punto de vista de los lenguajes de programación no. Los programas lógicos resultaron estar en la misma jerarquía que las máquinas de Turing [Llo87, pp. 53-54] e inauguraron un vasto campo de exploración dentro de la programación declarativa. Hoy en día sería difícil enumerar todas las aportaciones que la programación lógica ha hecho a la inteligencia artificial, los sistemas basados en conocimiento y la teoría de las bases de datos, por mencionar algunas áreas.

Existe una relación estrecha entre la programación lógica y la teoría de los lenguajes formales que creemos no ha recibido la atención que se merece. De hecho, hay mucha herramienta de la teoría de los lenguajes formales que puede ser aplicada a la programación lógica [Mel91]. Tal es el caso de los analizadores sintácticos[1]. Si examinamos la estructura de los programas

[1] En inglés, parsers.

lógicos veremos que existe una fuerte similitud con las gramáticas libres de contexto. La idea general es que los analizadores sintácticos pueden funcionar, bajo ciertas condiciones, como sistemas formales para programas lógicos.

El trabajo desarrollado en esta tesis consiste, primero, en mostrar la relación que existe entre un tipo muy general de programas, llamados programas orientados a estados, y las gramáticas libres de contexto. Segundo, demostrar cómo un tipo particular de analizadores sintácticos junto con ciertas reglas, resultan ser un sistema formal correcto y completo (relativo[2] a la base de Herbrand del lenguaje), para una tipo restringido de programas lógicos llamados programas cadena. Tercero, establecer la equivalencia semántica (también restringida) entre los programas orientados a estados y los programas cadena.

La tesis está dividida en tres partes. La parte I (capítulos 1 y 2) trata sobre la relación de las gramáticas libres de contexto con los programas orientados a estados. La parte II (capítulos 3, 4, 5 y 6) se enfoca en el uso de los analizadores por cartas como sistemas formales para programas cadena. La parte III (capítulo 7) sirve de enlace entre las partes I y II. El capítulo 1 contiene el material básico sobre gramáticas y lenguajes formales. En el capítulo 2, se definen los programas orientados a estados y se demuestran dos teoremas sobre la relación que guardan con las gramáticas libres de contexto. En el capítulo 3, se definen los analizadores por cartas y se da un ejemplo detallado de su utilización. El capítulo 4 contiene nociones básicas de la lógica de primer orden. En el capítulo 5, se da una breve introducción a la teoría de programación lógica y se demuestra una versión del teorema de Herbrand. En el capítulo 6, se definen los programas cadena y se prueba la completud y validez de los analizadores por cartas como sistemas formales para programas cadena. En el capítulo 7, se prueba la equivalencia entre los programas orientados a estados y los programas cadena.

Aunque esta tesis es esencialmente autocontenida, por cuestiones de espacio resultaría imposible incluir las demostraciones de todos los antecedentes teóricos que se usaron. En tales casos se incluyeron referencias bibliográficas adecuadas y los resultados que aparecen sin demostración están rotulados como proposiciones. También, en algunas demostraciones, se obviaron algunos pasos lógicos para no hacer tediosa la lectura.

[2] Ver página 49.

Parte I

1. Gramáticas y Lenguajes Formales

En este capítulo presentamos las nociones básicas de gramáticas y lenguajes formales que usaremos a lo largo de la tesis.

1.1 Definiciones Básicas

Definimos un *alfabeto* Σ como un conjunto finito y no vacío de símbolos. Donde por *símbolo* se entiende una unidad indivisible. e.g., $\Sigma = \{a, b, c\}$. Llamamos *cuerda* a una sucesión finita de símbolos de un alfabeto. Por ejemplo, $w = aacbbabab cabab$ es una cuerda del alfabeto $\Sigma = \{a, b, c\}$. La *cuerda vacía* es aquella que no tiene símbolos. La denotamos λ. Llamemos Σ^* al conjunto de todas las cuerdas del alfabeto Σ incluyendo a la cuerda vacía y definamos $\Sigma^+ = \Sigma^* - \{\lambda\}$.

Por ejemplo:

Si $\Sigma = \{a, b\}$ entonces

$$\Sigma^* = \{\lambda, a, b, aa, ab, ba, bb, aaa, aab, \ldots\}$$

$$\Sigma^+ = \{a, b, aa, ab, ba, bb, aaa, aab, \ldots\}$$

Definición 1.1 Lenguaje

Si $L \subseteq \Sigma^*$ decimos que L es un lenguaje *sobre el alfabeto Σ.*

Por ejemplo:

Si $\Sigma = \{a, b, c\}$ entonces $L = \{abc, aabbcc, aaabbbccc, aaaabbbbcccc, \ldots\}$ es un lenguaje sobre Σ.

De aquí en adelante asumimos que todas las definiciones son sobre un alfabeto Σ conocido de antemano.

Definición 1.2 Concatenación de cuerdas

Sean $w = a_1 a_2 \ldots a_n$ y $v = b_1 b_2 \ldots b_m$ cuerdas. Definimos la concatenación *de w con v como $wv = a_1 a_2 \ldots a_n b_1 b_2 \ldots b_m$. Decimos que w es un prefijo y v un sufijo de wv.*

Por ejemplo:

Si $w = aaaa$ y $v = bbbb$ entonces $wv = aaaabbbb$.

Se puede extender la definición de concatenación a cualquier número finito de cuerdas, en este caso dicha operación es asociativa sobre Σ^*.

También notemos que para toda cuerda w del alfabeto $w = \lambda w = w\lambda$.

Si w es una cuerda del alfabeto, entonces definimos $w^0 = \lambda$ y para todo entero positivo n:

$$w^n = \overbrace{ww\cdots w}^{n \text{ veces}}$$

formalmente, $w^n = w^{n-1}w = ww^{n-1}$ si $n > 0$.

Por ejemplo:

Si $w = abc$, entonces $w^3 = abcabcabc$.

Definición 1.3 Longitud de una cuerda

La longitud de una cuerda w es el número de símbolos del alfabeto que contiene la cuerda. La denotamos por $|w|$ y se define inductivamente de la siguiente manera:

1. $|\lambda| = 0$
2. *Si $a \in \Sigma$ y w es una cuerda, entonces* $|wa| = |w| + 1$

Por ejemplo:

$$|abcabc| = 6.$$

Afirmación 1.4 *Si w y v son cuerdas entonces* $|wv| = |w| + |v|$.

Demostración Por inducción sobre $|v|$.

Primero observemos que de la definición de longitud de cuerda se desprende que $|v| = 0$ si y sólo si $v = \lambda$. Por lo tanto, si $|v| = 0$, entonces $|wv| = |w| = |w| + 0 = |w| + |v|$.

Supongamos como hipótesis inductiva que $|wx| = |w| + |x|$ para toda cuerda x tal que $|x| \leq k$.

Si $|v| = k+1$, entonces de la definición de longitud de cuerda inferimos que $v = xa$ con x una cuerda, $a \in \Sigma$, y $|x| = k$. Por lo tanto:

$$\begin{aligned} |wv| = |w(xa)| = |(wx)a| &= |wx| + 1 \quad \text{por definición de longitud de cuerda} \\ &= |w| + |x| + 1 \quad \text{por hipótesis inductiva} \\ &= |w| + (|x| + 1) \\ &= |w| + |xa| \quad \text{por definición de longitud de cuerda} \\ &= |w| + |v| \end{aligned}$$

◇

Definición 1.5 Concatenación de lenguajes

Sean L_1 y L_2 lenguajes. Definimos la concatenación *de L_1 con L_2 como*

$$L_1 \cdot L_2 = \{wv \mid w \in L_1 \ y \ v \in L_2\}$$

Si L es un lenguaje, definimos $L^0 = \{\lambda\}$ y para todo entero positivo n

$$L^n = \overbrace{L \cdot L \cdot \ldots \cdot L}^{n \text{ veces}}$$

formalmente $L^n = L^{n-1} \cdot L = L \cdot L^{n-1}$ si $n > 0$.

Por ejemplo:

Si $L_1 = \{a, aa, aaa, \ldots\}$ y $L_2 = \{b, bb, bbb, \ldots\}$ entonces:

$L_1 \cdot L_2 = \{ab, abb, aab, aabb, abbb, aabb, aaab, \ldots\}$

$(L_1)^3 = \{aaa, aaaa, aaaaa, \ldots\} = L_1 - \{a, aa\}$

Definición 1.6 Cerradura de Kleene y cerradura positiva

Sea L un lenguaje. Definimos la cerradura de Kleene *o cerradura* estrella *de L como:*

$$\begin{aligned} L^* &= L^0 \cup L^1 \cup L^2 \cup \ldots \\ &= \bigcup \{L^n \mid n \geq 0\} \\ &= \{w_1 w_2 \ldots w_n \mid n \geq 0 \ y \ w_i \in L \ \ \forall i = 1, \ldots, n\} \end{aligned}$$

y la cerradura positiva *de L como*

$$\begin{aligned} L^+ &= L^1 \cup L^2 \cup \ldots \\ &= \bigcup \{L^n \mid n \geq 1\} \end{aligned}$$

Obsérvese que si $\lambda \in L$ entonces $L^* = L^+$.

Definición 1.7 Gramática Formal

$\mathcal{G} = \langle N, T, S, P \rangle$ *es una* gramática formal *si:*

N *es un conjunto finito de símbolos llamados* variables *o* no terminales.

$T \subseteq \Sigma$ *es un conjunto finito de símbolos llamados* símbolos terminales.

$S \in N$ *es un símbolo distinguido llamado* símbolo inicial *o* variable inicial.

$P \subseteq (N \cup T)^+ \times (N \cup T)^*$ *es un conjunto finito cuyos elementos llamamos* producciones.

Para todo $(x, y) \in P$ *escribimos* $x \rightarrow y$. *Además* N *y* T *son tales que* $N \neq \emptyset, T \neq \emptyset, N \cap T = \emptyset$.

Ejemplo 1.8

$N = \{S, A, B\}$

$T = \{a, b, c\}$

$P = \{S \rightarrow AB,\ A \rightarrow AA,\ A \rightarrow a,\ B \rightarrow BB,\ B \rightarrow b,\ AaB \rightarrow c\}$

Por comodidad usualmente numeramos las producciones de la siguiente manera:

1. $S \rightarrow AB$
2. $A \rightarrow AA$
3. $A \rightarrow a$
4. $B \rightarrow BB$
5. $B \rightarrow b$
6. $AaB \rightarrow c$

Definición 1.9 Forma sentencial

Si $\mathcal{G} = \langle N, T, S, P \rangle$ *es una gramática decimos que* α *es una* forma sentencial *de* $\mathcal{G}$ *si* $\alpha \in (N \cup T)^*$.

Las gramáticas formales proporcionan reglas específicas para derivar formas sentenciales a partir de otras. Esto es, si v y w son formas sentenciales y $x \rightarrow z$ es una producción en P decimos que vzw se *deriva* de vxw usando la producción $x \rightarrow z$ y escribimos $vxw \Rightarrow vzw$.

Si $w_1, w_2, \ldots, w_n$ son formas sentenciales y $w_1 \Rightarrow w_2 \Rightarrow \cdots \Rightarrow w_n$ decimos que w_n se deriva de w_1 y lo denotamos $w_1 \overset{*}{\Rightarrow} w_n$. A la sucesión

$w_1, w_2, \ldots, w_n$ la llamamos una ***derivación*** de w_n a partir de w_1 en $\mathcal{G}$. También es válido decir que $w \overset{*}{\Rightarrow} w$.

Retomando el ejemplo 1.8 (los números de las producciones usadas aparecen abajo de las flechas):

$$S \underset{1}{\Rightarrow} AB \underset{2}{\Rightarrow} AAB \underset{4}{\Rightarrow} AABB \underset{4}{\Rightarrow} AABBB \underset{3}{\Rightarrow} AaBBB \underset{6}{\Rightarrow} cBB \underset{5}{\Rightarrow} cbB \underset{5}{\Rightarrow} cbb \tag{1.1}$$

Por lo tanto $S \overset{*}{\Rightarrow} cbb$.

El número de pasos de una derivación es el número de producciones (no necesariamente distintas) que se aplicaron en ella. Por ejemplo (1.1) es una derivación de ocho pasos aunque la producción $B \to BB$ se aplicó dos veces; mientras que $AB \overset{*}{\Rightarrow} AB$ es una derivación de cero pasos. Si w_n se deriva de w_1 en al menos un paso escribimos $w_1 \overset{+}{\Rightarrow} w_n$. Si w_n se deriva de w_1 en m pasos escribimos $w_1 \overset{m}{\Rightarrow} w_n$.

Definición 1.10 Lenguaje generado por una gramática

Sea $\mathcal{G} = \langle N, T, S, P \rangle$ una gramática. Definimos el lenguaje generado *por $\mathcal{G}$ como*

$$L(\mathcal{G}) = \left\{ w \in T^* \mid S \overset{*}{\Rightarrow} w \right\}$$

Por ejemplo:

Por (1.8) sabemos que $S \overset{*}{\Rightarrow} cbb$ por lo tanto $cbb \in L(\mathcal{G})$.

Definición 1.11 Lenguaje generado por una gramática a partir de un símbolo

Sea $\mathcal{G} = \langle N, T, S, P \rangle$ una gramática. Sea $A \in N$. Definimos el lenguaje generado por $\mathcal{G}$ a partir *de A como*

$$L[\mathcal{G}, A] = \left\{ w \in T^* \mid A \overset{*}{\Rightarrow} w \right\}$$

Cuando no haya confusión, simplemente se denotará como $L[A]$. En particular $L[S] = L(\mathcal{G})$.

Retomando el ejemplo 1.8:

$b \in L[\mathcal{G}, B]$ ya que $B \to b \in P$, sin embargo $b \notin L(\mathcal{G})$.

Definición 1.12 Gramáticas equivalentes

Dos gramáticas $\mathcal{G}_1 = \langle N_1, T_1, S_1, P_1 \rangle$ *y* $\mathcal{G}_2 = \langle N_2, T_2, S_2, P_2 \rangle$ *son* equivalentes *si* $L(\mathcal{G}_1) = L(\mathcal{G}_2)$.

1.2 Clasificación de Chomsky

Si $\mathcal{G} = \langle N, T, S, P \rangle$ es una gramática, entonces podemos catalogarla según la forma de sus producciones dentro de alguno de los siguientes cuatro tipos:

Gramáticas no restringidas (tipo 0)

No existe restricción sobre sus producciones. Por definición todas son de la forma $x \to w$ con $x \in (N \cup T)^+$ y $w \in (N \cup T)^*$. Incluso es válido que w sea λ, en cuyo caso la longitud de la forma sentencial derivada podría disminuir. Por este motivo a estas gramáticas también se les llama contraibles.

Gramáticas dependientes de contexto (tipo 1)

Sus producciones son de la forma $x \to w$ donde $|w| \geq |x|$. Se les llama dependientes del contexto porque esta restricción equivale (ver [Sal73, p.p. 82-83]) a pedir que las producciones tengan la forma $xAw \to xvw$ con $x, w, v \in (N \cup T)^*, v \neq \lambda$ y $A \in N$. Entonces decimos que A se puede reemplazar por v siempre que aparezca en el contexto de x y w.

Gramáticas libres de contexto (tipo 2)

Sus producciones son de la forma $A \to w$ donde $A \in N$ y $w \in (N \cup T)^+$. Se les llama libres de contexto ya que A puede reemplazarse por w independientemente del contexto donde aparezca.

Gramáticas regulares (tipo 3)

Una gramática es lineal derecha si todas sus producciones son de la forma $A \to xB$ ó $A \to x$ con $A, B \in N$ y $x \in T$. Análogamente una gramática es lineal izquierda si todas sus producciones son de la forma $A \to Bx$ ó $A \to x$ con $A, B \in N$ y $x \in T$. Una gramática es regular si es lineal izquierda o lineal derecha.

En todos los casos permitimos la producción especial $S \to \lambda$ siempre que S (el símbolo inicial) no aparezca del lado derecho de ninguna producción. Dada cualquier gramática de los tipos 1,2 ó 3 siempre existe una gramática equivalente del mismo tipo en la cual S no aparece del lado derecho de

ninguna producción (ver [HU69, p.p. 15-16]). Esto implica que λ puede o no pertenecer al lenguaje generado por cualquiera de las gramáticas.

Puede verse que cada tipo de gramática contiene al siguiente, es decir que una gramática regular es a su vez libre de contexto, dependiente del contexto y no restringida. Los lenguajes generados por estas gramáticas se clasifican de acuerdo a la gramática del tipo más grande que los genera. Así tenemos lenguajes tipo 0, lenguajes tipo 1 ó dependientes de contexto, lenguajes tipo 2 ó libres de contexto y lenguajes tipo 3 ó regulares. Las contenciones entre los tipos de gramáticas son propias, lo cual se muestra exhibiendo lenguajes de cada tipo que no pertenecen al siguiente. A continuación damos algunos ejemplos. No incluimos las demostraciones debido a que requieren herramienta teorica que está fuera de los objetivos de esta tesis.

$L_1 = \{a^n b^n | \ n \geq 1\} \cup \{a^n b^{2n} | \ n \geq 1\}$ es una lenguaje libre de contexto ya que está generado por una gramática con las siguientes producciones:

$S \rightarrow X$

$S \rightarrow Y$

$X \rightarrow ab$

$X \rightarrow aXb$

$Y \rightarrow abb$

$Y \rightarrow aYbb$

Sin embargo L_1 no es regular (ver [Lin90, p.p. 204-207,216]).

$L_2 = \{a^n b^n c^n | \ n \geq 1\}$ es un lenguaje dependiente de contexto ya que está generado (ver [Lin90, p.p. 306-307]) por una gramática con las siguientes producciones:

$S \rightarrow abc$

$S \rightarrow aXbc$

$Xb \rightarrow bX$

$Xc \rightarrow Ybcc$

$bY \rightarrow Yb$

$aY \rightarrow aa$

$aY \rightarrow aaX$

Sin embargo L_2 no es libre de contexto (ver [Lin90, p.p. 216]).

Un ejemplo de un lenguaje tipo 0 que no es dependiente de contexto puede consultarse en [Lin90, p.p. 309-310] ó en [HU69, p.p. 117-118].

Existen diversos mecanismos matemáticos que dado un lenguaje nos permiten saber si una cuerda pertenece o no a él. Los lenguajes tipo 0 son reconocidos por máquinas de Turing; los lenguajes dependientes del contexto por autómatas de dos "stacks" o autómatas linealmente acotados; los lenguajes libres de contexto son reconocidos por autómatas de stack y los lenguajes regulares por autómatas finitos.

2. Resultados sobre Programas Orientados a Estados

En este capítulo demostramos resultados referentes a la relación entre las gramáticas libres de contexto y los programas orientados a estados.

2.1 Funciones Asociadas y Gramáticas Adjuntas

Definición 2.1 Relación binaria sobre un conjunto

R es una relación binaria *sobre un conjunto X si $R \subseteq X \times X$.*

Definición 2.2 Composición de relaciones binarias

Sean $R_1, R_2, \ldots, R_n$ relaciones binarias sobre X. Definimos la composición[1] *todas ellas como:*

$$\begin{aligned} R_1; R_2; \cdots; R_n &= R_1 \circ R_2 \circ \cdots \circ R_n \\ &= \{ (a_1, a_{n+1}) \mid \text{existen } a_2, a_3, \ldots, a_n \text{ en } X \text{ tales que} \\ &\qquad (a_j, a_{j+1}) \in R_j \text{ para todo } j = 1, 2, \ldots, n \} \end{aligned}$$

Lema 2.3 *Sean $R_1, R_2, \ldots, R_n$ relaciones binarias sobre X. Sean $Q_1, Q_2, \ldots, Q_n$ conjuntos tales que $Q_i \subseteq R_i$ para todo $i = 1, 2, \ldots, n$, entonces:*

$$Q_1; Q_2; \cdots; Q_n \subseteq R_1; R_2; \cdots; R_n$$

Demostración Sea $(a_1, a_{n+1}) \in Q_1; Q_2; \cdots; Q_n$ entonces por definición existen $a_2, a_3, \ldots, a_n \in X$ tales que $(a_j, a_{j+1}) \in Q_j$ para todo $j = 1, 2, \ldots, n$. Como por hipótesis $Q_j \subseteq R_j$ para todo $j = 1, 2, \ldots, n$, entonces $(a_j, a_{j+1}) \in R_j$ para todo $j = 1, 2, \ldots, n$. Por lo tanto $(a_1, a_{n+1}) \in R_1; R_2; \cdots; R_n$. ◇

Definición 2.4 Programa orientado a estados

$\mathcal{P} = \langle D, C, V, I, S \rangle$ es un programa orientado a estados *(POE) si:*

1. *D es un conjunto (posiblemente infinito) llamado conjunto de* estados.
2. *C es un conjunto finito de relaciones binarias sobre D llamadas* comandos.
3. *V es un conjunto de símbolos llamados* etiquetas.

[1] *Nótese que el orden de la composición de relaciones es contrario al que se usa en la composición de funciones.*

4. *I es un conjunto finito de* inclusiones *de la forma:*

$$R_0 \supseteq R_1; R_2; \cdots; R_n$$

donde R_0 es una etiqueta y $R_1, \cdots, R_n$ son etiquetas o comandos no necesariamente distintos y $n > 0$.

5. *S es una etiqueta distinguida que aparece del lado izquierdo de alguna inclusión.*

Denotemos con $Pot(X)$ al conjunto potencia de un conjunto X. i.e.,

$$Pot(X) = \{ A \mid A \subseteq X \}$$

Definición 2.5 Función asociada a un programa orientado a estados

Sea $\mathcal{P} = \langle D, C, V, I, S \rangle$ un POE y $\Phi : V \cup C \longrightarrow Pot(D \times D)$. Decimos que Φ es función asociada *a $\mathcal{P}$ si:*

1. *Para toda inclusión $R_0 \supseteq R_1; R_2; \cdots; R_n$ en I, tenemos que $\Phi(R_0) \supseteq \Phi(R_1); \Phi(R_2); \cdots; \Phi(R_n)$.*
2. *Para todo comando Q en C, se cumple $\Phi(Q) = Q$ (esto tiene sentido ya que los elementos de C son relaciones binarias sobre D).*

Decimos entonces que Φ satisface I.

Cabe aclarar que el dominio de Φ consiste por un lado en las etiquetas en V y por otro en los comandos en C. El rango de Φ consta de relaciones binarias sobre D. Los comandos en C ya son relaciones binarias sobre D por lo cual hacemos que $\Phi \mid_C$ sea la identidad. Si bien esta definición no es del todo elegante, sí resulta adecuada para la demostración del lema 2.11 como veremos después.

Si $\mathcal{P}$ es un POE, definimos $\Gamma(\mathcal{P}) = \{\Phi : V \cup C \longrightarrow Pot(D \times D) \mid \Phi$ es función asociada a $\mathcal{P}\}$, es decir, el conjunto de todas las funciones asociadas a $\mathcal{P}$. Cuando no haya confusión se denotara simplemente por Γ.

Afirmación 2.6 *Si $\mathcal{P} = \langle D, C, V, I, S \rangle$ es un POE, entonces existe una función asociada a $\mathcal{P}$. i.e., $\Gamma(\mathcal{P}) \neq \emptyset$.*

Demostración Definamos $\Phi_D : V \cup C \longrightarrow Pot(D \times D)$ como $\Phi_D(R) = D \times D$ para todo $R \in V$ y $\Phi_D \mid_C$ como la identidad en C.

1. Si $R_0 \supseteq R_1; R_2; \cdots; R_n$ está en I entonces $\Phi_D(R_1); \Phi_D(R_2); \cdots; \Phi_D(R_n)$ es también una relación binaria sobre D (ya que el rango de Φ_D son relaciones binarias sobre D) y por lo tanto

$$\Phi_D(R_0) = D \times D \supseteq \Phi_D(R_1); \Phi_D(R_2); \cdots; \Phi_D(R_n)$$

2. Para todo $Q \in C$, tenemos que $\Phi_D(Q) = Q$ por definición.

Por lo tanto Φ_D es una función asociada a $\mathcal{P}$ y $\Phi_D \in \Gamma(\mathcal{P})$. ◇

Teorema 2.7 *Si $\mathcal{P} = \langle D, C, V, I, S \rangle$ es un POE, entonces existe una única función Ω mínima asociada a $\mathcal{P}$.*

Demostración Para todo $R \in V$ definamos $\Omega(R) = \bigcap \{ \Phi(R) \mid \Phi \in \Gamma \}$ y $\Omega \mid_C$como la identidad en C. Nótese que tiene sentido considerar esta intersección ya que $D \times D = \Phi_D(R) \in \{ \Phi(R) \mid \Phi \in \Gamma \}$ y por lo tanto $\{ \Phi(R) \mid \Phi \in \Gamma \} \neq \emptyset$. Por otro lado, como $\Phi(R) \subseteq D \times D$ para todo $\Phi \in \Gamma$ y $R \in V$, tenemos que $\Omega(R) \subseteq D \times D$ para todo $R \in V$. Por lo tanto, $\Omega : V \cup C \longrightarrow Pot(D \times D)$ está bien definida.

Por construcción $\Omega(R) \subseteq \Phi(R)$ para todo $\Phi \in \Gamma$ y $R \in V$. Entonces basta mostrar que Ω es una función asociada.

1. Si $R_0 \supseteq R_1; R_2; \cdots; R_n$ está en I, entonces

$$\Phi(R_0) \supseteq \Phi(R_1); \Phi(R_2); \cdots; \Phi(R_n)$$
$$\supseteq \Omega(R_1); \Omega(R_2); \cdots; \Omega(R_n) \text{ para todo } \Phi \in \Gamma$$

esta última contención, por el lema 2.3. Por lo tanto

$$\Omega(R_0) = \bigcap \{ \Phi(R_0) \mid \Phi \in \Gamma \} \supseteq \Omega(R_1); \Omega(R_2); \cdots; \Omega(R_n).$$

2. Para todo $Q \in C$ se tiene $\Omega(Q) = Q$ por definición.

Por lo tanto Ω es una función asociada, i.e., $\Omega \in \Gamma$.

Para la unicidad basta observar que si existiera otra mínima función asociada Ω', entonces para todo $R \in V$ tendríamos que $\Omega(R) \subseteq \Omega'(R)$ por ser Ω' función asociada, y $\Omega'(R) \subseteq \Omega(R)$ por ser Ω' mínima. De aquí que $\Omega = \Omega'$. ◇

Definición 2.8 Relación principal y relaciones derivadas de un programa orientado a estados

Sea $\mathcal{P} = \langle D, C, V, I, S \rangle$ un POE. Sea Ω la mínima función asociada a $\mathcal{P}$. Decimos entonces que $\Omega(S)$ es la relación principal *y $\{\Omega(R) \mid R \in V\}$ es el conjunto de* relaciones derivadas *de $\mathcal{P}$.*

Ejemplo 2.9

Definamos $\mathcal{P} = \langle D, C, V, I, S \rangle$ de la siguiente manera:

$D = \{\text{los enteros positivos}\}$

$C = \{Q\}$ donde $Q = \{(n, 2n) \mid n \text{ entero positivo}\}$

$V = \{S\}$

$I = \{S \supseteq S;S, \quad S \supseteq Q\}$

Supongamos que queremos saber cuáles son las relaciones derivadas de $\mathcal{P}$.

Si Φ es cualquier función asociada a $\mathcal{P}$, entonces debe satisfacer las inclusiones en I. De modo que $\Phi(S) \supseteq \Phi(Q) = Q$. Pero también $\Phi(S) \supseteq \Phi(S);\Phi(S)$. Por lo tanto, por el lema 2.3, $\Phi(S) \supseteq Q;Q$. Repitiendo el argumento obtenemos que $\Phi(S) \supseteq Q;Q;Q$, y generalizando por inducción tenemos que:

$$\Phi(S) \supseteq \overbrace{Q;Q;\cdots;Q}^{m \text{ veces}} \text{ para todo } m \text{ entero positivo.}$$

Por lo tanto

$$\Phi(S) \supseteq \bigcup_{m \geq 1} \overbrace{Q;Q;\cdots;Q}^{m \text{ veces}} \tag{2.1}$$

De modo que toda función Φ asociada a $\mathcal{P}$ satisface (2.1). Y parece razonable conjeturar que (2.1) es lo menos que deberíamos exigir a una función para considerarla asociada; en cuyo caso Ω, la mínima de las funciones asociadas, tendría que satisfacer:

$$\begin{aligned} \Omega(S) &= \bigcup_{m \geq 1} \overbrace{Q;Q;\cdots;Q}^{m \text{ veces}} \\ &= \bigcup_{m \geq 1} \{(n, 2^m n) \mid n \text{ entero positivo}\} \end{aligned}$$

Ahora definamos la siguiente gramática libre de contexto $\mathcal{G} = \langle N, T, \overline{S}, P \rangle$ donde:

$T = \{\overline{Q}\}$

$N = \{\overline{S}\}$

$$P = \left\{ \overline{S} \to \overline{SS},\ \overline{S} \to \overline{Q} \right\}$$

Es fácil ver que $L(\mathcal{G}) = \left\{ \overline{Q} \right\}^+ = \left\{ \overline{Q}, \overline{QQ}, \overline{QQQ}, \ldots \right\}$ y existe entonces una conexión inmediata entre la gramática y el programa orientado a estados, de forma tal que:

$$\Omega(S) \supseteq \bigcup_{\underbrace{\overline{QQ}\cdots\overline{Q}}_{m \text{ veces}} \in L(\mathcal{G}),\ m \geq 1} \overbrace{Q;Q;\cdots;Q}^{m \text{ veces}}$$

Y de ser cierta la conjetura:

$$\Omega(S) = \bigcup_{\underbrace{\overline{QQ}\cdots\overline{Q}}_{m \text{ veces}} \in L(\mathcal{G}),\ m \geq 1} \overbrace{Q;Q;\cdots;Q}^{m \text{ veces}}$$

Todo esto se puede formalizar y efectivamente caracteriza a las relaciones derivadas de $\mathcal{P}$ como veremos a continuación.

Definición 2.10 Gramática adjunta a un programa orientado a estados

Sea $\mathcal{P} = \langle D, C, V, I, S \rangle$ un POE. Definamos la gramática adjunta *a $\mathcal{P}$ como $\mathcal{GR}(\mathcal{P}) = \left\langle N, T, \overline{S}, P \right\rangle$ donde:*

$$N = \left\{ \overline{R} \mid R \in V \right\}$$

$$T = \left\{ \overline{Q} \mid Q \in C \right\}$$

i.e., los terminales de la gramática corresponden a los comandos y los no terminales a las etiquetas.

Para toda inclusión $R_0 \supseteq R_1; R_2; \cdots; R_n$ en I, entonces la producción $\overline{R}_0 \to \overline{R}_1\overline{R}_2\cdots\overline{R}_n$ esta en P, y éstas son las únicas producciones en P.

Notemos que está definición es correcta, ya que R_0 es una etiqueta, y por lo tanto $\overline{R}_0$ es un símbolo no terminal. Observemos también que $\lambda \notin L(\mathcal{GR}(\mathcal{P}))$.

Lema 2.11 *Sea* $\mathcal{P} = \langle D, C, V, I, S\rangle$ *un POE y* $\mathcal{GR}(\mathcal{P}) = \left\langle N, T, \overline{S}, P\right\rangle$ *la gramática adjunta a* $\mathcal{P}$. *Sea* Φ *cualquier función asociada a* $\mathcal{P}$ *y* $R_0 \in V$. *Si* $\overline{R}_0 \stackrel{+}{\Rightarrow} \overline{R}_1\overline{R}_2\cdots\overline{R}_m$ *donde* $\overline{R}_j \in (N \cup T)^*$ *para todo* $j = 1, 2, \ldots, m$ *entonces* $\Phi(R_0) \supseteq \Phi(R_1); \Phi(R_2); \cdots; \Phi(R_m)$.

Demostración Por inducción sobre n, el número de pasos de la derivación.

Si $n = 1$

La derivación se obtuvo usando una producción:

$$\overline{R}_0 \rightarrow \overline{R}_1\overline{R}_2\cdots\overline{R}_m$$

Entonces debe existir una inclusión correspondiente en I:

$$R_0 \supseteq R_1; R_2; \cdots; R_m$$

Dado que Φ satisface I (por ser una función asociada) tenemos que

$$\Phi(R_0) \supseteq \Phi(R_1); \Phi(R_2); \cdots; \Phi(R_n)$$

De modo que la afirmación se cumple para $n = 1$.

Ahora supongamos que la afirmación se cumple para $n \leq k$

Si $\overline{R}_0 \stackrel{k+1}{\Rightarrow} \overline{R}_1\overline{R}_2\cdots\overline{R}_m$, podemos suponer que

$$\overline{R}_0 \stackrel{k}{\Rightarrow} \overline{R}_1\overline{R}_2\cdots\overline{R}_{i-1}\overline{T}_{ij}\overline{R}_{j+1}\cdots\overline{R}_m$$

y luego se aplicó la producción $\overline{T}_{ij} \rightarrow \overline{R}_i\cdots\overline{R}_j$ donde $\overline{T}_{ij} \in N$, $i \leq j$. Aplicando la hipótesis de inducción tenemos:

$$\Phi(R_0) \supseteq \Phi(R_1); \Phi(R_2); \cdots; \Phi(R_{i-1}); \Phi(T_{ij}); \Phi(R_{j+1}); \cdots; \Phi(R_m)$$

y

$$\Phi(T_{ij}) \supseteq \Phi(R_i); \cdots; \Phi(R_j)$$

Por el lema 2.3 concluimos

$$\Phi(R_0) \supseteq \Phi(R_1); \Phi(R_2); \cdots; \Phi(R_{i-1}); \Phi(R_i); \cdots; \Phi(R_j); \Phi(R_{j+1}); \cdots; \Phi(R_m)$$

◇

Teorema 2.12 *Sea $\mathcal{P} = \langle D, C, V, I, S\rangle$ un POE. Sea $\mathcal{GR}(\mathcal{P}) = \langle N, T, \overline{S}, P\rangle$ la gramática adjunta a $\mathcal{P}$. Entonces podemos caracterizar a Ω, la mínima función asociada a $\mathcal{P}$, de la siguiente manera:*

Para todo $R \in V$,

$$\Omega(R) = \bigcup \{\, Q_1; Q_2; \cdots; Q_n \mid \overline{Q}_1\overline{Q}_2\cdots\overline{Q}_n \in L[\overline{R}],\ n \geq 1\}$$

Demostración Definamos $\Omega' : V \cup C \longrightarrow Pot(D \times D)$ tal que $\Omega' \mid_C$ es la identidad en C y $\Omega'(R) = \bigcup \{\, Q_1; Q_2; \cdots; Q_n \mid \overline{Q}_1\overline{Q}_2\cdots\overline{Q}_n \in L[\overline{R}],\ n \geq 1\}$ para todo $R \in V$. Notemos que como $\overline{Q}_1\overline{Q}_2\cdots\overline{Q}_n \in L[\overline{R}]$, entonces $\overline{Q}_1, \overline{Q}_2, \ldots, \overline{Q}_n$ son símbolos terminales de la gramática adjunta y por lo tanto $Q_1, Q_2, \ldots, Q_n$ son comandos (no necesariamente distintos).

Es claro que el teorema queda demostrado si Ω' coincide con Ω. Con tal propósito veremos que:

1. $\Omega'(R) \subseteq \Omega(R)$ para todo $R \in V$.
2. Ω' es una función asociada.

Como por hipótesis Ω es la mínima de tales funciones, entonces forzosamente coinciden.

1. Sea $R \in V$ y $\alpha \in \Omega'(R) = \bigcup \{\, Q_1; Q_2; \cdots; Q_n \mid \overline{Q}_1\overline{Q}_2\cdots\overline{Q}_n \in L[\overline{R}]\,\}$. Entonces existe $\overline{Q}_1\overline{Q}_2\cdots\overline{Q}_n \in L[\overline{R}]$ tal que $\alpha \in Q_1; Q_2; \cdots; Q_n$. Entonces $\overline{R} \stackrel{+}{\Rightarrow} \overline{Q}_1\overline{Q}_2\cdots\overline{Q}_n$. Dado que Ω es una función asociada, aplicando el lema 2.11 se tiene que:

$$\begin{aligned}\Omega(R) &\supseteq \Omega(Q_1); \Omega(Q_2); \cdots \Omega(Q_n)\\ &= Q_1; Q_2; \cdots; Q_n\end{aligned}$$

   Esta última igualdad se sigue de que $Q_1, Q_2, \ldots, Q_n$ son comandos y de que $\Omega \mid_C$ es la identidad, por ser función asociada. Por lo tanto $\alpha \in \Omega(R)$ y concluimos $\Omega'(R) \subseteq \Omega(R)$ para todo $R \in V$.

2. Veamos que Ω' es una función asociada:

   (a) Si $R_0 \supseteq R_1; R_2; \cdots; R_n$ es una inclusión en I, entonces la producción $\overline{R}_0 \rightarrow \overline{R}_1\overline{R}_2\cdots\overline{R}_n$ está en P. Recordemos que si R_i es un comando entonces $\Omega'(R_i) = R_i$ y si R_i es una etiqueta entonces $\Omega'(R_i) = \bigcup \{Q_1; Q_2; \cdots; Q_m \mid \overline{Q}_1\overline{Q}_2\cdots\overline{Q}_m \in L[\overline{R}_i],\ m \geq 1\}$. Para simplificar la demostración, si R_i es un comando, podemos pensar que $L[\overline{R}_i] = \{\overline{R}_i\}$ entonces

$$\Omega'(R_i) = \bigcup \left\{ Q_1; Q_2; \cdots; Q_m \mid \overline{Q}_1 \overline{Q}_2 \cdots \overline{Q}_m \in L[\overline{R}_i],\ m \geq 1 \right\}$$

en ambos casos.

Sea $(a_1, a_{n+1}) \in \Omega'(R_1); \Omega'(R_2); \cdots; \Omega'(R_n)$. Por definición de composición de relaciones binarias, existen $a_1, a_2, \ldots, a_n \in D$ tales que:

$$(a_1, a_2) \in \Omega'(R_1) = \bigcup \left\{ Q_1; Q_2; \cdots; Q_m \mid \overline{Q}_1 \overline{Q}_2 \cdots \overline{Q}_m \in L[\overline{R}_1], m \geq 1 \right\}$$

$$(a_2, a_3) \in \Omega'(R_2) = \bigcup \left\{ Q_1; Q_2; \cdots; Q_m \mid \overline{Q}_1 \overline{Q}_2 \cdots \overline{Q}_m \in L[\overline{R}_2], m \geq 1 \right\}$$

$$\vdots$$

$$(a_n, a_{n+1}) \in \Omega'(R_n) = \bigcup \left\{ Q_1; Q_2; \cdots; Q_m \mid \overline{Q}_1 \overline{Q}_2 \cdots \overline{Q}_m \in L[\overline{R}_n], m \geq 1 \right\}$$

entonces existen

$$\begin{aligned} \overline{w}_1 &= \overline{Q}_{11} \overline{Q}_{12} \cdots \overline{Q}_{1m_1} \in L[\overline{R}_1] \\ \overline{w}_2 &= \overline{Q}_{21} \overline{Q}_{22} \cdots \overline{Q}_{2m_2} \in L[\overline{R}_2] \\ &\vdots \\ \overline{w}_n &= \overline{Q}_{n1} \overline{Q}_{n2} \cdots \overline{Q}_{nm_n} \in L[\overline{R}_n] \end{aligned}$$

tales que

$$(a_1, a_2) \in w_1 = Q_{11}; Q_{12}; \cdots; Q_{1m_1}$$

$$(a_2, a_3) \in w_2 = Q_{21}; Q_{22}; \cdots; Q_{2m_2}$$

$$\vdots$$

$$(a_n, a_{n+1}) \in w_n = Q_{n1}; Q_{n2}; \cdots; Q_{nm_n}$$

Por lo tanto $(a_1, a_{n+1}) \in w_1; w_2; \cdots; w_n$ y como sabíamos que $\overline{R}_0 \rightarrow \overline{R}_1 \overline{R}_2 \cdots \overline{R}_n$ está en P y $\overline{w}_1 \in L[\overline{R}_1]$, $\overline{w}_2 \in L[\overline{R}_2], \ldots, \overline{w}_n \in L[\overline{R}_n]$, entonces $\overline{w}_1 \overline{w}_2 \cdots \overline{w}_n \in L[\overline{R}_0]$.

Concluimos que:

$$(a_1, a_{n+1}) \in w_1; w_2; \cdots; w_n \text{ con } \overline{w}_1 \overline{w}_2 \cdots \overline{w}_n \in L[\overline{R}_0]$$

o equivalentemente

$$(a_1, a_{n+1}) \in Q_{11}; \cdots; Q_{1m_1}; \cdots\cdots; Q_{n1}; \cdots; Q_{nm_n}$$

con $\overline{Q}_{11}\cdots\overline{Q}_{1m_1}\cdots\cdots\overline{Q}_{n1}\cdots\overline{Q}_{nm_n} \in L[\overline{R}_0]$, entonces

$$(a_1, a_{n+1}) \in \bigcup\left\{ Q_1; Q_2; \cdots; Q_m \mid \overline{Q}_1\overline{Q}_2\cdots\overline{Q}_m \in L[\overline{R}_0],\ m \geq 1\right\}$$
$$= \Omega'(R_0)$$

Por lo tanto, $\Omega'(R_0) \supseteq \Omega'(R_1); \Omega'(R_2); \cdots; \Omega'(R_n)$ ya que (a_1, a_{n+1}) se escogió arbitrariamente.

(b) Por definición, para todo $Q \in C$ tenemos que $\Omega'(Q) = Q$

Por lo tanto Ω' es una función asociada a $\mathcal{P}$, y concluimos que $\Omega'(R) = \Omega(R)$. ◇

2.2 Programas Asociados a Gramáticas

Hemos visto como podemos asociar de manera natural una gramática libre de contexto a un programa orientado a estados. Es importante señalar que la gramática adjunta, por sí sola no caracteriza al POE. Retomando el ejemplo 2.9:

Si $\mathcal{P} = \langle D, C, V, I, S\rangle$ es tal que:

$D = \mathcal{N}$ (los naturales)

$C = \{Q\}$ donde $Q = \{\ (n, 2n) \mid n \in \mathcal{N}\ \}$

$V = \{S\}$

$I = \{S \supseteq S; S,\quad S \supseteq Q\}$

entonces $\mathcal{GR}(\mathcal{P}) = \left\langle N, T, \overline{S}, P\right\rangle$ donde

$T = \left\{\overline{Q}\right\}$

$N = \left\{\overline{S}\right\}$

$P = \left\{\overline{S} \to \overline{S}\,\overline{S},\quad \overline{S} \to \overline{Q}\right\}$

Si tuviéramos que $Q = \{\ (n, n+1) \mid n \in \mathcal{N}\ \}$ la gramática adjunta sería la misma pero el significado del programa totalmente distinto.

No obstante, como veremos a continuación, a una gramática libre de contexto podemos asociarle un POE particular, de modo que la relación principal del programa determine totalmente al lenguaje generado por la gramática.

Definición 2.13 Representación cociente de un lenguaje sobre un alfabeto

Sea L un lenguaje sobre un alfabeto Σ. Definimos la representación cociente *de L sobre Σ como*

$$\mathcal{Q}(L) = \{(xw, w) \mid x \in L \text{ y } w \in \Sigma^*\}$$

Puede observarse que un lenguaje esta totalmente determinado por su representación cociente y viceversa.

Lema 2.14 *Si L y M son lenguajes sobre un alfabeto Σ entonces $\mathcal{Q}(L \cdot M) = \mathcal{Q}(L); \mathcal{Q}(M)$.*

Demostración Sabemos

$$\mathcal{Q}(L \cdot M) = \{(xw, w) \mid x \in L \cdot M \text{ y } w \in \Sigma^*\} = \{(zvw, w) \mid z \in L,\ v \in M \text{ y } w \in \Sigma^*\}$$

y

$$\mathcal{Q}(L) = \{(xw, w) \mid x \in L \text{ y } w \in \Sigma^*\}$$

$$\mathcal{Q}(M) = \{(xw, w) \mid x \in M \text{ y } w \in \Sigma^*\}$$

$\boxed{\subseteq}$ Sea $(xw, w) \in \mathcal{Q}(L \cdot M)$. Entonces $x = zv$ con $z \in L, v \in M$. Por lo tanto $(zvw, vw) \in \mathcal{Q}(L)$ y $(vw, w) \in \mathcal{Q}(M)$, y por definición de composición de relaciones binarias $(xw, w) = (zvw, w) \in \mathcal{Q}(L); \mathcal{Q}(M)$.

$\boxed{\supseteq}$ Sea $(xw, w) \in \mathcal{Q}(L); \mathcal{Q}(M)$. Entonces por definición existe $z \in \Sigma^*$ tal que $(x, z) \in \mathcal{Q}(L)$ y $(z, w) \in \mathcal{Q}(M)$. Pero si $(x, z) \in \mathcal{Q}(L)$ entonces $(x, z) = (yz, z)$ con $y \in L$. Análogamente si $(z, w) \in \mathcal{Q}(M)$ entonces $(z, w) = (vw, w)$ con $v \in M$ y $w \in \Sigma^*$. Por lo tanto $x = yz = yvw$ con $y \in L, v \in M$ y $w \in \Sigma^*$, de donde $(xw, w) \in \mathcal{Q}(L \cdot M)$. ◇

Lema 2.15

Si $\{L_i\}_{i \in I}$ es una familia numerable de lenguajes sobre un alfabeto Σ entonces

$$\mathcal{Q}(\bigcup\{L_i \mid i \in I\}) = \bigcup\{\mathcal{Q}(L_i) \mid i \in I\}$$

Demostración

$$\begin{aligned}
\mathcal{Q}(\bigcup\{L_i \mid i \in I\}) &= \mathcal{Q}(\{x \mid x \in L_i \text{ para algún } i \in I\}) \\
&= \{(xw, w) \mid x \in L_i \text{ para algún } i \in I \text{ y } w \in \Sigma^*\} \\
&= \bigcup\{\{(xw, w) \mid x \in L_i \text{ y } w \in \Sigma^*\} \mid i \in I\} \\
&= \bigcup\{\mathcal{Q}(L_i) \mid i \in I\}
\end{aligned}$$

◇

Afirmación 2.16 *Si L y M son lenguajes sobre un alfabeto Σ, entonces $M \subseteq L$ si y solo si $\mathcal{Q}(M) \subseteq \mathcal{Q}(L)$.*

Demostración

$\boxed{\Longrightarrow}$

$$\begin{aligned} \mathcal{Q}(M) &= \{(xw, w) \mid x \in M \text{ y } w \in \Sigma^*\} \\ &\subseteq \{(xw, w) \mid x \in L \text{ y } w \in \Sigma^*\} \text{ ya que } M \subseteq L \text{ por hipótesis} \\ &= \mathcal{Q}(L) \end{aligned}$$

$\boxed{\Longleftarrow}$ Sea $x \in M$, y tomemos $w \in \Sigma^*$ arbitrario entonces $(xw, w) \in \mathcal{Q}(M)$. Se sigue, por hipótesis, que $(xw, w) \in \mathcal{Q}(L)$, y por lo tanto $x \in L$. ◇

Definición 2.17 Programa asociado a una gramática libre de contexto

Si $\mathcal{G} = \langle N, T, S, P\rangle$ una gramática libre de contexto, definamos el POE asociado a esta gramática como $\mathcal{PR}(\mathcal{G}) = \left\langle D, C, V, I, \widehat{S}\right\rangle$ donde:

$$D = \Sigma^*$$

$$V = \left\{ \widehat{A} \mid A \in N \right\}$$

$$C = \left\{ \widehat{B} \mid B \in T \right\} \text{ donde } \widehat{B} = \mathcal{Q}(\{B\}) \text{ para todo } B \in T$$

(recordemos que $\mathcal{Q}(\{B\}) = \{(Bw, w) \mid w \in \Sigma^*\}$).

Para toda producción $A_0 \to A_1 A_2 \cdots A_n$ en P, entonces la inclusión $\widehat{A}_0 \supseteq \widehat{A}_1; \widehat{A}_2; \cdots; \widehat{A}_n$ está en I, y éstas son las únicas inclusiones en I. Notemos que esta definición es correcta ya que, como A_0 es un símbolo no terminal, entonces $\widehat{A}_0$ es una etiqueta.

Teorema 2.18 *Sea $\mathcal{G} = \langle N, T, S, P\rangle$ una gramática libre de contexto y $\mathcal{PR}(\mathcal{G}) = \left\langle D, C, V, I, \widehat{S}\right\rangle$ el programa asociado a $\mathcal{G}$. Entonces la relación principal del programa es igual a la representación cociente del lenguaje generado por la gramática. i.e., Si Ω es la mínima función asociada a $\mathcal{PR}(\mathcal{G})$, entonces $\Omega(\widehat{S}) = \mathcal{Q}(L(\mathcal{G}))$.*

Demostración Si $\mathcal{GR}(\mathcal{PR}(\mathcal{G})) = \left\langle N', T', \overline{S}, P'\right\rangle$ es la gramática adjunta al programa asociado a $\mathcal{G}$, entonces $\mathcal{GR}(\mathcal{PR}(\mathcal{G}))$ y $\mathcal{G}$ son la misma gramática módulo nombres de símbolos, ya que:

1. $T' = \left\{ \overline{\widehat{B}} \mid \widehat{B} \in C \right\} = \left\{ \overline{\widehat{B}} \mid B \in T \right\}$

2. $N' = \left\{ \overline{\widehat{A}} \mid \widehat{A} \in V \right\} = \left\{ \overline{\widehat{A}} \mid A \in N \right\}$

3. Para toda producción $A_0 \to A_1 A_2 \cdots A_n$ en P tenemos que la inclusión $\widehat{A}_0 \supseteq \widehat{A}_1; \widehat{A}_2; \cdots; \widehat{A}_n$ está en I, entonces la producción $\overline{\widehat{A}}_0 \to \overline{\widehat{A}}_1 \overline{\widehat{A}}_2 \cdots \overline{\widehat{A}}_n$ está en P'.

Y como éstas son las únicas inclusiones y producciones entonces

$$P' = \left\{ \overline{\widehat{A}}_0 \to \overline{\widehat{A}}_1 \overline{\widehat{A}}_2 \cdots \overline{\widehat{A}}_n \mid A_0 \to A_1 A_2 \cdots A_n \in P \right\}$$

Si ahora aplicamos el teorema 2.12 a $\mathcal{PR}(\mathcal{G})$ tenemos que:

$$\Omega(\widehat{S}) = \bigcup \left\{ \widehat{B}_1; \widehat{B}_2; \cdots; \widehat{B}_n \mid \overline{\widehat{B}}_1 \overline{\widehat{B}}_2 \cdots \overline{\widehat{B}}_n \in L[\overline{\widehat{S}}] = L(\mathcal{GR}(\mathcal{PR}(\mathcal{G}))),\ n \geq 1 \right\}$$

como $\mathcal{GR}(\mathcal{PR}(\mathcal{G}))$ y $\mathcal{G}$ son la misma gramática módulo nombres de símbolos

$$= \bigcup \left\{ \widehat{B}_1; \widehat{B}_2; \cdots; \widehat{B}_n \mid B_1 B_2 \cdots B_n \in L[S] = L(\mathcal{G}),\ n \geq 1 \right\}$$

$$= \bigcup \{ \mathcal{Q}(\{B_1\}); \mathcal{Q}(\{B_2\}); \cdots; \mathcal{Q}(\{B_n\}) \mid B_1 B_2 \cdots B_n \in L(\mathcal{G}),\ n \geq 1 \}$$

por definición

$$= \bigcup \{ \mathcal{Q}(\{B_1\} \cdot \{B_2\} \cdot \ldots \cdot \{B_n\}) \mid B_1 B_2 \cdots B_n \in L(\mathcal{G}),\ n \geq 1 \}$$

por el lema 2.14

$$= \bigcup \{ \mathcal{Q}(\{B_1 B_2 \cdots B_n\}) \mid B_1 B_2 \cdots B_n \in L(\mathcal{G}),\ n \geq 1 \}$$

por la definición 1.5

$$= \mathcal{Q}\left(\bigcup \{ \{B_1 B_2 \cdots B_n\} \mid B_1 B_2 \cdots B_n \in L(\mathcal{G}),\ n \geq 1 \} \right)$$

por el lema 2.15

$$= \mathcal{Q}(L(\mathcal{G}))$$ ◇

Entonces el programa asociado $\mathcal{PR}(\mathcal{G})$ determina totalmente a la gramática $\mathcal{G}$ de la cual se obtuvo. Contrariamente a lo que ocurría con la gramática adjunta $\mathcal{GR}(\mathcal{P})$ a un POE $\mathcal{P}$. Esto se debe a que los programas orientados a estados tienen la misma jerarquía que las máquinas de Turing. Por lo tanto al adjuntar una gramática libre de contexto a un POE posiblemente perdemos expresividad.

Parte II

3. Analizadores por Cartas

Los analizadores sintácticos son procedimientos que nos permiten saber si una determinada cuerda pertenece al lenguaje generado por una gramática dada. En este capítulo trabajaremos con un tipo particular de analizadores sintácticos llamados a veces analizadores por "cartas"[1]. Los analizadores por cartas al aplicarse a gramáticas libres de contexto, tienen la particularidad de proporcionar todos los posibles árboles de análisis sintáctico[2] para una cuerda dada.

Definición 3.1 Cartas

Sean E y B conjuntos arbitrarios. Llamamos a E conjunto de estados *y a B conjunto de* nombres. *Intuitivamente una* carta *es una gráfica dirigida cuyos nodos son elementos de E y cuyas aristas toman nombres en B. Formalmente definimos una carta como una relación $C \subseteq E \times E \times B$, tal que si $\langle e_1, e_2, n_1 \rangle \in C$ y $e_1 \neq e_2$ entonces para todo $n_2 \in B$ se cumple $\langle e_2, e_1, n_2 \rangle \notin C$.*

Un *analizador por cartas* se compone de un conjunto de cartas y un conjunto de reglas en el metalenguaje.

Observación 3.2 *En el resto de la tesis hablaremos de "inicializar" una carta C y de "agregar" nuevas aristas a una carta C. Estas expresiones matemáticamente carecen de sentido, puesto que C es un conjunto, pero pueden traducirse a las siguientes definiciones formales:*

1. *Al inicializar la carta definimos un conjunto de aristas C_0.*
2. *Para todo entero positivo n, definimos C_{n+1} como C_n unión el conjunto de aristas que se obtiene aplicando alguna regla a aristas en C_n.*
3. *Definimos $C = \bigcup_{n \geq 0} C_n$.*

De este modo conservamos la idea computacional de carta; es decir, una gráfica a la que agregamos aristas a intervalos discretos de tiempo de acuerdo con ciertas reglas fijas.

[1] En inglés, chart parsers. El nombre obedece a una supuesta similitud con los mapas cartográficos.

[2] Que no definiremos aquí.

Ahora daremos un ejemplo de cómo implementar un analizador por cartas para reconocer el lenguaje generado por una gramática libre de contexto y de paso obtendremos varias definiciones importantes.

Sea $\mathcal{G} = \langle N, T, S, P \rangle$ la gramática tal que:

$$N = \{E, S\}$$
$$T = \{I, +, -, *, /, (,)\}$$

y P consta de las producciones:

1. $S \to X$
2. $X \to (X)$
3. $X \to X + X$
4. $X \to X - X$
5. $X \to I$

Donde I representa cualquier número natural. Podríamos formalizar esto último agregando las producciones: $I \to 0$, $I \to 1, \ldots, I \to 9$, $I \to 0I$, $I \to 1I, \ldots, I \to 9I$. Pero no lo haremos por simplicidad.

Entonces el lenguaje generado por $\mathcal{G}$ resulta ser el conjunto de las expresiones aritméticas bien formadas de suma y resta de naturales escritos en base 10.

Por ejemplo:

$$I + I - I$$

puede obtenerse como:

$$S \underset{1}{\Rightarrow} X \underset{3}{\Rightarrow} X + X \underset{4}{\Rightarrow} X + X - X \overset{*}{\underset{5}{\Rightarrow}} I + I - I$$

Definición 3.3 Producción punto

Sea $\mathcal{H} = \langle N, T, S, P \rangle$ una gramática. Entonces para toda $x \to vw$ producción de P tal que $w, v \in (N \cup T)^*$, decimos que $x \to v \bullet w$ es una producción punto. *Al conjunto de todas las producciones punto, le llamamos conjunto de producciones punto asociadas a $\mathcal{H}$.*

En nuestro ejemplo la producción $X \to (X)$ genera las siguientes producciones punto:

$X \to \bullet(X)$

$X \rightarrow (\bullet X)$

$X \rightarrow (X\bullet)$

$X \rightarrow (X)\bullet$

Ahora sea $E = \{[n] \mid n$ es un número natural $\}$ y sea B el conjunto de los símbolos terminales unión el conjunto de producciones punto asociadas a la gramática. Para cada cuerda w del alfabeto, definamos una carta $C(w)$ cuyo conjunto de estados es E y con nombres en B .

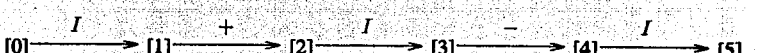
Por ejemplo, tomemos la cuerda

$$I + I - I$$

e inicialicemos la carta de acuerdo con la siguiente regla:

(i) Agregar la arista $\langle [n-1], [n], t\rangle$ a la carta, siempre que el símbolo terminal t aparezca en la posición n de la cuerda.

De este modo tenemos que $\langle [0], [1], I\rangle$, $\langle [1], [2], +\rangle$, $\langle [2], [3], I\rangle$, $\langle [3], [4], -\rangle$, $\langle [4], [5], I\rangle$ están en $C(I + I - I)$:

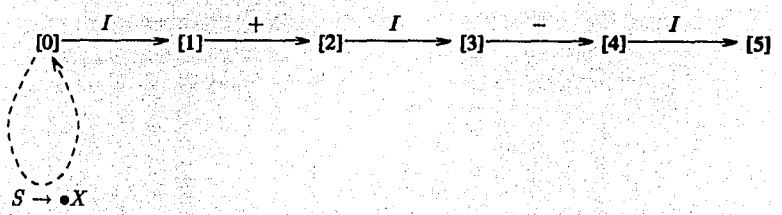


Establezcamos ahora las siguientes reglas:

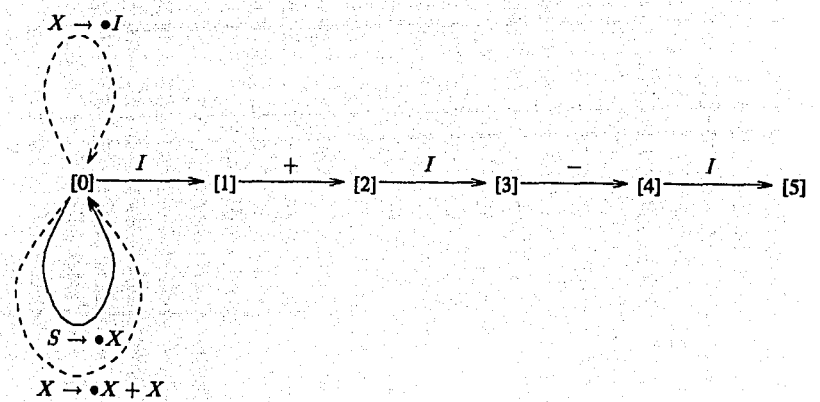
(ii) Si S es el símbolo inicial de la gramática, entonces para toda producción de la forma $S \rightarrow w$ en P agregar la arista $\langle [0], [0], S \rightarrow \bullet w\rangle$ a la carta.

(iii) Siempre que agreguemos una arista del tipo $\langle [n], [m], A \rightarrow w \bullet Bv\rangle$ donde $B \in N$ y $w, v \in (N \cup T)^*$ agregamos aristas $\langle [m], [m], B \rightarrow \bullet z\rangle$ para toda producción de la forma $B \rightarrow z$ en P (esto es para toda producción que comience con B).

De modo que aplicando la regla (ii) obtenemos $\langle [0], [0], S \rightarrow \bullet X \rangle$:



Y aplicando la regla (iii) a $\langle [0], [0], S \rightarrow \bullet X \rangle$ obtenemos $\langle [0], [0], X \rightarrow \bullet X + X \rangle$ y $\langle [0], [0], X \rightarrow \bullet I \rangle$:



Definición 3.4 Aristas activas y pasivas

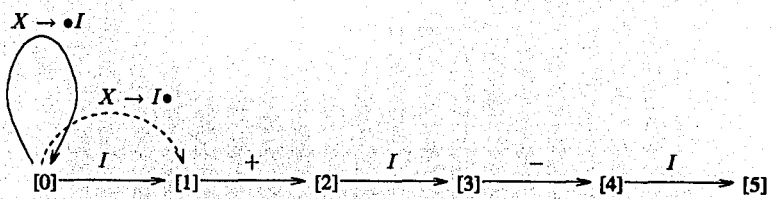
Decimos que una arista es pasiva *si está etiquetada con un símbolo terminal, o con una producción punto de la forma* $A \rightarrow w\bullet$ *donde* $w \in (N \cup T)^+$ *(i.e., si el punto es el último símbolo del nombre). En caso contrario decimos que la arista es* activa.

Ahora enunciamos la regla fundamental de los analizadores por cartas:

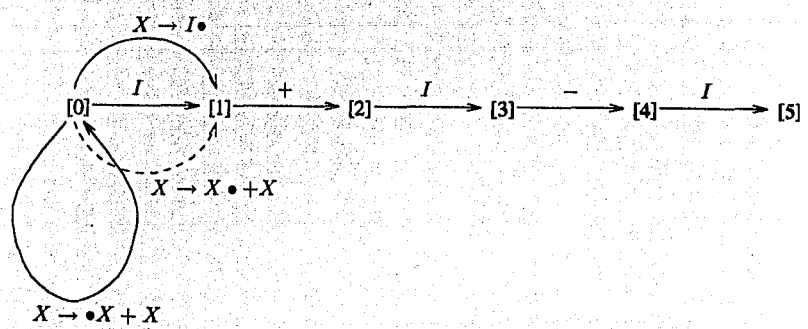
(RF) Siempre que tengamos una arista activa de la forma $\langle [l], [m], A \to w \bullet Bv \rangle$ donde $v, w \in (N \cup T)^*$, y una arista pasiva $\langle [m], [n], B \to z\bullet \rangle$ con $B \in N$ ó $\langle [m], [n], B \rangle$ con $B \in T$ agregamos la arista $\langle [l], [n], A \to wB \bullet v \rangle$ a la carta.

Aplicando la regla fundamental en nuestro ejemplo:

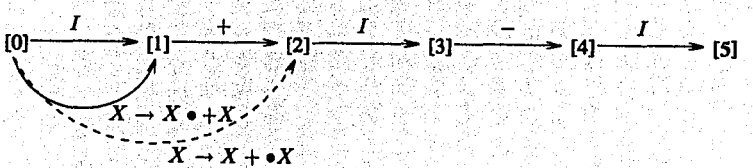
$\langle [0], [0], X \to \bullet I \rangle$ y $\langle [0], [1], I \rangle$ generan $\langle [0], [1], X \to I\bullet \rangle$:



$\langle [0], [0], X \to \bullet X + X \rangle$ y $\langle [0], [1], X \to I\bullet \rangle$ generan $\langle [0], [1], X \to X \bullet + X \rangle$:

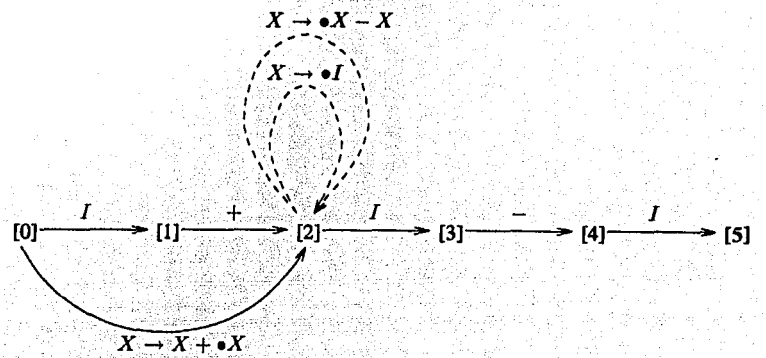


$\langle [0], [1], X \to X \bullet + X \rangle$ y $\langle [1], [2], + \rangle$ generan $\langle [0], [2], X \to X + \bullet X \rangle$:



Aplicando la regla (iii) a $\langle [0],[2], X \to X + \bullet X \rangle$ obtenemos:

$\langle [2],[2],\ X \to \bullet X - X \rangle$ y $\langle [2],[2], X \to \bullet I \rangle$:

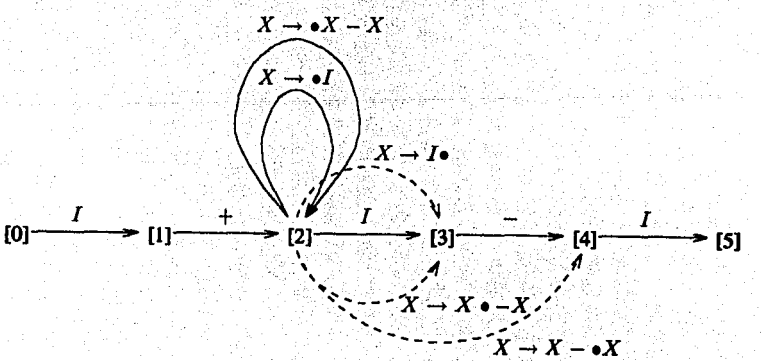


Aplicando la regla fundamental

$\langle [2],[2], X \to \bullet I \rangle$ y $\langle [2],[3], I \rangle$ generan $\langle [2],[3], X \to I \bullet \rangle$

$\langle [2],[2],\ X \to \bullet X - X \rangle$ y $\langle [2],[3], X \to I \bullet \rangle$ generan $\langle [2],[3],\ X \to X \bullet - X \rangle$

$\langle [2],[3],\ X \to X \bullet - X \rangle$ y $\langle [3],[4], - \rangle$ generan $\langle [2],[4],\ X \to X - \bullet X \rangle$

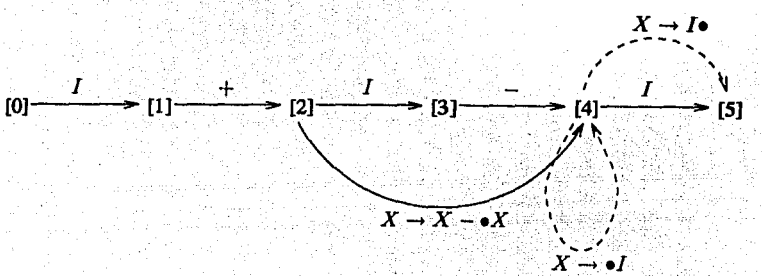


A su vez aplicando la regla (iii) a $\langle [2],[4],\ X \to X - \bullet X \rangle$ obtenemos:

$\langle [4],[4], X \to \bullet I \rangle$

y aplicando la regla fundamental

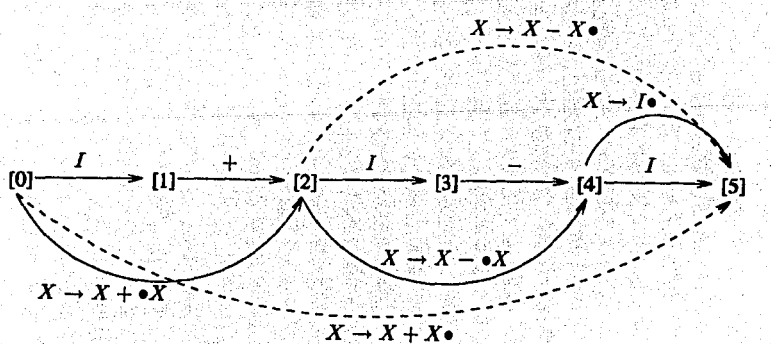
$\langle [4],[4], X \to \bullet I \rangle$ y $\langle [4],[5], I \rangle$ generan $\langle [4],[5], X \to I \bullet \rangle$



Aplicando la regla fundamental

$\langle [2],[4],\ X \to X - \bullet X \rangle$ y $\langle [4],[5], X \to I \bullet \rangle$ generan $\langle [2],[5],\ X \to X - X \bullet \rangle$

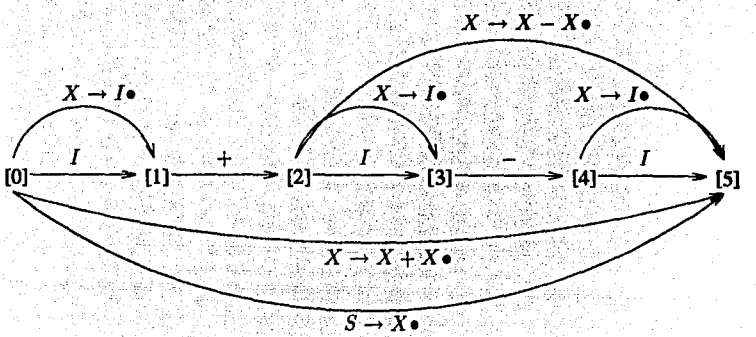
$\langle [0],[2], X \to X + \bullet X \rangle$ y $\langle [2],[5],\ X \to X - X \bullet \rangle$ generan $\langle [0],[5], X \to X + X \bullet \rangle$



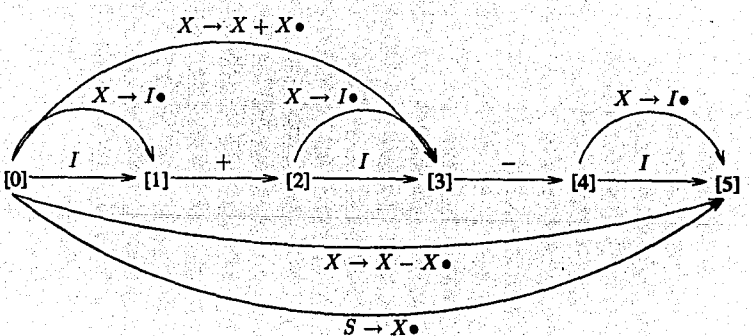
Por último por la regla fundamental

$\langle [0],[0], S \to \bullet X \rangle$ y $\langle [0],[5], X \to X + X \bullet \rangle$ generan $\langle [0],[5], S \to X \bullet \rangle$

De modo que la gráfica contiene las aristas:



Por un desarrollo análogo, podríamos ver que la carta también contiene las aristas:



Que corresponden a la derivación[3]:

$$S \underset{1}{\Rightarrow} X \underset{4}{\Rightarrow} X - X \underset{3}{\Rightarrow} X + X - X \overset{*}{\underset{5}{\Rightarrow}} I + I - I$$

Podemos resumir todo lo anterior de la siguiente manera:

Sea $\mathcal{G} = \langle N, T, S, P \rangle$ una gramática libre de contexto. Sea $t_1 \cdots t_n$ una cuerda del alfabeto. Sea $E = \{[n] \mid n \text{ es un número natural }\}$ y sea B el conjunto de símbolos terminales unión el conjunto de producciones punto asociadas a la gramática. Para cada cuerda w del alfabeto, definimos una carta $C(w)$ cuyo conjunto de estados es E y con nombres en B. Inicializamos $C(w)$ de acuerdo con la siguiente regla:

(i) Agregamos aristas $\langle [k-1], [k], t_k \rangle$ para todo $k = 1, \ldots, n$

Luego establecemos las siguientes reglas:

(ii) Si S es el símbolo inicial de la gramática, entonces para toda producción de la forma $S \rightarrow w$ en P agregamos la arista $\langle [0], [0], S \rightarrow \bullet w \rangle$ a la carta.

(ii) Siempre que agreguemos una arista del tipo $\langle [n], [m], A \rightarrow w \bullet Bv \rangle$ donde $B \in N$ y $w, v \in (N \cup T)^*$ agregamos aristas $\langle [m], [m], B \rightarrow \bullet z \rangle$ para toda producción de la forma $B \rightarrow z$ en P.

(RF) Siempre que tengamos una arista activa de la forma $\langle [l], [m], A \rightarrow w \bullet Bv \rangle$ donde $v, w \in (N \cup T)^*$, y una arista pasiva $\langle [m], [n], B \rightarrow z \bullet \rangle$ con $B \in N$ ó $\langle [m], [n], B \rangle$ con $B \in T$ agregamos la arista $\langle [l], [n], A \rightarrow wB \bullet v \rangle$ a la carta.

(ii) y (iii) se conocen como reglas de arriba-abajo, y el analizador por cartas así construido es un analizador de *arriba a abajo*. El análisis termina cuando ya no es posible agregar nuevas aristas[4]. Notemos que siempre termina, ya que la longitud de la cuerda y el conjunto de producciones punto son finitos. Cuando el análisis termina, para todo $k = 1, \ldots, n$ tenemos:

$t_1 \cdots t_k \in L(\mathcal{G})$ si y sólo si existe una arista pasiva de la forma $\langle [0], [k], S \rightarrow w \bullet \rangle$ en la carta y $S \rightarrow w$ es una producción en P.

Esta última afirmación es consecuencia indirecta del teorema 2.18 y los resultados del capítulo 4. Su demostración puede consultarse en [AU72, p.p. 320-

[3] También pueden corresponder a otras derivaciones, por ejemplo:

$$S \underset{1}{\Rightarrow} X \underset{4}{\Rightarrow} X - X \underset{5}{\Rightarrow} X - I \underset{3}{\Rightarrow} X + X - I \overset{*}{\underset{5}{\Rightarrow}} I + I - I$$

[4] Formalmente $C = C_n$ para algún n en los naturales.

331] aunque no la usaremos en el resto de la tesis. Para un tratamiento detallado de los analizadores por cartas y su implementación en Prolog, puede consultarse [GM89, p.p. 179-215].

4. Lógica de Primer Orden

Damos una breve introducción a algunos aspectos de lógica de primer orden necesarios para poder hablar de programas lógicos. Nos basamos en los textos de Enderton[End87] y Lloyd[Llo87].

4.1 Definiciones Básicas

Definimos un lenguaje de primer orden, partiendo de un conjunto infinito de símbolos. Suponemos que cada símbolo es una entidad indivisible y que ningún símbolo es una sucesión de otros símbolos. Agrupamos a los símbolos de la siguiente manera:

- *Símbolos lógicos*

  Paréntesis: (,).

  Conectivos: $\neg, \longrightarrow, \wedge, \vee$.

  Variables: v_n donde n es cualquier entero positivo.

- *Parámetros*

  Símbolo de *cuantificador*: $\forall$.

  Símbolos de *predicado*: Para cada entero positivo n, un conjunto, posiblemente vacío, de símbolos, llamados símbolos de predicado n-ario.

  Símbolos de *constante*: Un conjunto, posiblemente vacío, de símbolos.

  Símbolos de *función*: Para cada entero positivo n un conjunto, posiblemente vacío, de símbolos, llamados símbolos de función n-aria.

Ejemplo 4.1

A la teoría elemental de los números podemos asociarle el lenguaje de primer orden cuyos parámetros son:

Símbolos de predicado: $\{<, \approx\}$

Símbolos de constante: $\{\mathbf{0}\}$

Símbolos de función unaria: {**S** (para sucesor)}.

Símbolos de función binaria: {+ (para suma), · (para producto), **E** (para exponenciación)}.

Usaremos este ejemplo a lo largo del capítulo para ilustrar las definiciones que siguen.

Definición 4.2 Expresión

Definimos una expresión como cualquier sucesión finita de símbolos del lenguaje de primer orden.

Por ejemplo:

$$\mathbf{SS} << \mathbf{0SS} + \mathbf{S}$$

es una expresión.

Definición 4.3 Término

Definimos el conjunto de los términos inductivamente, de la siguiente manera:

1. *Las variables y las constantes son* términos.
2. *Si f es un símbolo de función n-aria y $t_1, t_2, \ldots, t_n$ son términos, entonces $ft_1t_2\ldots t_n$ es un* término.
3. *Solamente las expresiones obtenidas a partir de (1) y (2) son términos.*

Por ejemplo:

0

S v_1

SSSS0

+ SSSS0 S v_1

son términos (agregamos los espacios en blanco para facilitar la lectura pero no son parte del lenguaje).

Definición 4.4 Fórmula atómica o átomo

Una fórmula atómica *o* átomo *es una expresión de la forma $Pt_1t_2\ldots t_n$ donde P es un símbolo de predicado n-ario y $t_1, t_2, \ldots, t_n$ son términos.*

Por ejemplo:

$$< \mathbf{0} + \mathbf{SSSS0}\ \mathbf{S}v_1$$

es una fórmula atómica.

Si bien estas expresiones son adecuadas para el desarrollo formal de la teoría, resultan un tanto difíciles de entender. De modo que constantemente usaremos abreviaturas para facilitar la lectura. El lector debe tener presente que se trata únicamente de notación. Así por ejemplo:

$$< \mathbf{0} + \mathbf{SSSS0}\mathbf{S}v_1$$

lo denotamos

$$< (\mathbf{0},\ +(\mathbf{S}^4\mathbf{0},\ \mathbf{S}v_1)\ )$$

e inclusive por tratarse de predicados y funciones binarias

$$\mathbf{0}\ <\ \mathbf{S}^4\mathbf{0} + \mathbf{S}v_1$$

A veces también usaremos metavariables x, w, y, z para simbolizar cualquier variable. e.g.,

$$\mathbf{0}\ <\ \mathbf{S}^4\mathbf{0} + \mathbf{S}x$$

Definición 4.5 Literal

Decimos que α es una literal *si α es una fórmula atómica o la negación de un fórmula atómica. i.e., $\alpha = (\neg\beta)$ con β fórmula atómica.*

Por ejemplo:

$$\mathbf{0} < \mathbf{S}^4\mathbf{0} + \mathbf{S}v_1$$

y

$$\neg(\mathbf{0} < \mathbf{S}^4\mathbf{0} + \mathbf{S}v_1)$$

son literales

Definición 4.6 Fórmula

Definimos el conjunto de fórmulas inductivamente de la siguiente manera:

1. *Las fórmulas atómicas son* fórmulas.

2. *Si α y β son fórmulas y v_i es una variable*[1] *entonces:*

$$(\neg\alpha)$$
$$(\alpha \longrightarrow \beta)$$
$$(\alpha \wedge \beta)$$
$$(\alpha \vee \beta)$$
$$\forall v_i \alpha$$

son fórmulas.

3. *Solamente las expresiones obtenidas a partir de (1) y (2) son fórmulas.*

Por ejemplo:

$$(\neg\forall v_1(\, \mathbf{0} \;<\; \mathbf{S^4 0} + \mathbf{S}v_1 \longrightarrow \mathbf{0 \cdot 0} \approx \mathbf{1}\,))$$

es una fórmula.

Definición 4.7 Presencia libre

Sea x cualquier variable y δ cualquier fórmula. Definimos inductivamente x aparece libre *en δ de la siguiente manera:*

1. *Si δ es atómica, x* aparece libre *en δ, si x aparece en (es un símbolo de) δ.*
2. *Si $\delta = (\neg\alpha)$, entonces x* aparece libre *en δ si x aparece libre en α.*
3. *Si δ es $(\alpha \longrightarrow \beta), (\alpha \wedge \beta)$ ó $(\alpha \vee \beta)$, entonces x* aparece libre *en δ si x aparece libre en α ó en β.*
4. *Si $\delta = \forall v_i \alpha$, entonces x* aparece libre *en δ si x aparece libre en α y $x \neq v_i$.*

A las variables que aparecen libres en una fórmula les llamamos variables libres *de la fórmula. Si x aparece en δ, pero no es una variable libre de δ, decimos que está* cuantificada.

Por ejemplo:

v_1 aparece libre en $\forall v_1\ v_1 < \mathbf{0} \longrightarrow \mathbf{S}v_1 < \mathbf{S0}$, (nótese que sus dos primeras presencias no son libres) y está cuantificada en $\forall v_1(v_1 < \mathbf{0} \longrightarrow \mathbf{S}v_1 < \mathbf{S0})$

[1] i es cualquier entero positivo.

Definición 4.8 Enunciado

Definimos un enunciado como una fórmula que no tiene variables libres.

Por ejemplo:

$$\forall v_1 \; v_1 < \mathbf{0} \longrightarrow \forall v_2 \; \mathbf{S}v_2 < \mathbf{S0}$$

Definición 4.9 Estructura

Sea $\mathcal{L}$ un lenguaje de primer orden. Definimos una estructura Ψ *para $\mathcal{L}$ como una función cuyo dominio es el conjunto de parámetros y tal que:*

1. *Ψ asocia al símbolo de cuantificador $\forall$ un conjunto* no vacío *$|\Psi|$ llamado universo de Ψ.*
2. *Ψ asocia a cada símbolo de predicado n-ario P una relación n-aria $P^{\Psi} \subseteq |\Psi|^n$.*
3. *Ψ asocia a cada símbolo de constante c un elemento c^{Ψ} de $|\Psi|$.*
4. *Ψ asocia a cada símbolo de función n-aria f una función n-aria $f^{\Psi} : |\Psi|^n \longrightarrow |\Psi|$.*

Ejemplo 4.10

Podemos asociar al lenguaje del ejemplo 4.1 la estructura μ tal que:

$|\mu| = \mathcal{N}$ los números naturales

$<^{\mu} = \{ (n,m) \mid n, m \in \mathcal{N} \text{ y } n < m \}$

$\approx^{\mu} = \{ (n,n) \mid n \in \mathcal{N} \}$

$\mathbf{0}^{\mu} = 0$

$\mathbf{S}^{\mu} : \mathcal{N} \longrightarrow \mathcal{N}$ es tal que $\mathbf{S}^{\mu}(n) = n + 1$

$+^{\mu} : \mathcal{N} \times \mathcal{N} \longrightarrow \mathcal{N}$ tal que $+^{\mu}(n,m) = n + m$

$\cdot^{\mu} : \mathcal{N} \times \mathcal{N} \longrightarrow \mathcal{N}$ tal que $\cdot^{\mu}(n,m) = n \cdot m$

$\mathbf{E}^{\mu} : \mathcal{N} \times \mathcal{N} \longrightarrow \mathcal{N}$ tal que $\mathbf{E}^{\mu}(n,m) = n^m$

Es importante aclarar que $< \approx \mathbf{0} + \cdot$ (en negritas) son símbolos del lenguaje de primer orden, mientras que $<$, $=$, 0, $+$, $\cdot$ son las relaciones menor que, igual a, el cero, y las operaciones de suma y producto en los números naturales, respectivamente. $< = 0 + \cdot$ son símbolos del metalenguaje y no pertenecen al lenguaje de primer orden.

Algunos autores llaman interpretaciones a las estructuras. Entonces se dice que:

$$<^{\mu},\ \approx^{\mu},\ \mathbf{0}^{\mu},\ \mathbf{S}^{\mu},\ +^{\mu},\ \cdot^{\mu},\ \mathbf{E}^{\mu}$$

son las interpretaciones bajo μ de los símbolos $< \approx \mathbf{0}\ \mathbf{S}\ + \cdot\ \mathbf{E}$ respectivamente.

En adelante, aunque no se mencione, asumimos que todas las fórmulas, estructuras y construcciones están definidas respecto a un mismo lenguaje de primer orden dado.

Definición 4.11 Asignación de variables

Sea $\mathcal{L}$ un lenguaje de primer orden y Ψ una estructura para el lenguaje. Sea $V = \{\ v_n \mid\ n$ entero positivo $\}$ el conjunto de todas las variables de $\mathcal{L}$. Decimos que la función h es una asignación de variables *para Ψ si $h : V \longrightarrow |\Psi|$.*

Definición 4.12 Ψ satisface δ con h

Sea Ψ una estructura para el lenguaje, δ una fórmula y h una asignación de variables para Ψ. Decimos que Ψ satisface *δ con h, lo cual denotamos:*

$$\models_{\Psi} \delta \quad [h]$$

si y solo si la traducción de δ determinada por Ψ, donde la variable x se traduce como $h(x)$ siempre que aparece libre en δ, es verdadera.

Formalmente esto se expresa de la siguiente manera:

1. **Términos**

   Sea T el conjunto de todos los términos de $\mathcal{L}$. Definimos la extensión $\overline{h} : T \longrightarrow |\Psi|$ como:

   (a) Para cada variable x, $\overline{h}(x) = h(x)$.

   (b) Para cada símbolo de constante c, $\overline{h}(c) = c^{\Psi}$.

   (c) Si f es un símbolo de función n-aria y $t_1, t_2, \ldots, t_n$ son términos, entonces:

   $$\overline{h}(f t_1 t_2 \ldots t_n) = f^{\Psi}(\overline{h}(t_1), \overline{h}(t_2), \ldots, \overline{h}(t_n))$$

   i.e., la extensión $\overline{h}$ asocia a cada término un elemento particular de $|\Psi|$.

2. **Fórmulas**

(a) Si P es un símbolo de predicado n-ario,

$$\models_\Psi Pt_1t_2\ldots t_n\ [h] \text{ si y solo si } (\overline{h}(t_1),\overline{h}(t_2),\ldots,\overline{h}(t_n)) \in P^\Psi$$

(b) $\models_\Psi \neg\delta\ [h]$ si y solo si no se cumple $\models_\Psi \delta\ [h]$.

(c) $\models_\Psi (\alpha \longrightarrow \beta)\ [h]$ si y solo si $\models_\Psi \beta\ [h]$ ó no se cumple $\models_\Psi \alpha\ [h]$ (o ambas).

(d) $\models_\Psi (\alpha \wedge \beta)\ [h]$ si y solo si $\models_\Psi \alpha\ [h]$ y $\models_\Psi \beta\ [h]$

(e) $\models_\Psi (\alpha \vee \beta)\ [h]$ si y solo si $\models_\Psi \alpha\ [h]$ ó $\models_\Psi \beta\ [h]$ (o ambas).

(f) $\models_\Psi \forall x\delta\ [h]$ si y solo si para todo $d \in |\Psi|$, tenemos que $\models_\Psi \delta\ [h(x \mid d)]$, donde $h(x \mid d)$ es una función que coincide con h en todo su dominio excepto que en la variable x toma el valor d. I.e., $h(x \mid d)(z) = h(z)$ para todo $z \neq x$ y $h(x \mid d)(x) = d$.

Ejemplo 4.13

Retomando el ejemplo 4.10:

Si x, y son variables y h es una asignación de variables para μ tal que $h(x) = 1$y $h(y) = 2$, entonces

$$\models_\mu x+y \approx \mathbf{S^3 0}\ \ [h]$$

ya que esta fórmula es una abreviatura de

$$\models_\mu \approx +xy\mathbf{SSS0}\ \ [h]$$

y por definición de $\overline{h}$ tenemos que

$$\begin{aligned}
\overline{h}(+xy) &= +^\mu(\overline{h}(x),\overline{h}(y)) \\
&= \overline{h}(x)+\overline{h}(y) \\
&= h(x)+h(y) \\
&= 1+2 \\
&= 0+1+1+1 \\
&= \overline{h}(\mathbf{0})+1\ +1+1 \\
&= \mathbf{S}^\mu(\overline{h}(\mathbf{0}))\ +1+1 \\
&= \overline{h}(\mathbf{S0})+1\ +1 \\
&= \mathbf{S}^\mu(\overline{h}(\mathbf{S0}))\ +1 \\
&= \overline{h}(\mathbf{SS0})+1 \\
&= \mathbf{S}^\mu(\overline{h}(\mathbf{SS0})) \\
&= \overline{h}(\mathbf{SSS0})
\end{aligned}$$

y $\overline{h}(+xy) = \overline{h}(\mathbf{SSS0})$ implica $(\overline{h}(+xy),\overline{h}(\mathbf{SSS0})) \in \approx^\mu$ y por lo tanto

$$\models_\mu \approx +xy\mathbf{SSS0} \quad [h]$$

Aquí resulta conveniente introducir las siguientes abreviaturas:

Si α y β son fórmulas y x cualquier variable entonces:

$(\alpha \longleftarrow \beta)$ abrevia a $(\beta \longrightarrow \alpha)$

$(\alpha \longleftrightarrow \beta)$ abrevia a $(\alpha \longrightarrow \beta) \wedge (\beta \longrightarrow \alpha)$

$\exists x\alpha$ abrevia a $(\neg\forall x(\neg\alpha))$

Así tenemos que si h es una asignación de variables para la estructura Ψ :

$\models_\Psi \exists x\alpha\ [h]$ syss $\models_\Psi (\neg\forall x(\neg\alpha))\ [h]$
syss no se cumple $\models_\Psi \forall x(\neg\alpha)\ [h]$
syss no se cumple que para todo $d \in |\Psi|$ $\models_\Psi \neg\alpha\ [h(x \mid d)]$
syss no se cumple que para todo $d \in |\Psi|$ no $\models_\Psi \alpha\ [h(x \mid d)]$
syss hay una $d \in |\Psi|$ tal que $\models_\Psi \alpha\ [h(x \mid d)]$

Los símbolos $\longleftarrow, \longleftrightarrow$ funcionan como conectivos, mientras que $\exists$ juega el papel de cuantificador existencial.

De nuevo para simplificar la lectura y evitar paréntesis innecesarios establecemos la siguiente jerarquía de precedencia al asociar los símbolos:

$$\neg$$
$$\forall, \exists$$
$$\wedge$$
$$\vee$$
$$\longleftarrow, \longleftrightarrow, \longrightarrow$$

Definición 4.14 Consecuencia lógica

Sea Γ un conjunto de fórmulas y δ una fórmula. Decimos que δ es una consecuencia lógica *de Γ o que Γ implica lógicamente a δ; lo cual denotamos $\Gamma \models \delta$, si y solo si, para toda estructura Ψ del lenguaje y para toda asignación de variables h tal que Ψ satisface a todos los elementos de Γ con h, tenemos que $\models_\Psi \delta\ [h]$.*

Si α y β son fórmulas, escribimos $\alpha \models \beta$ en lugar de $\{\alpha\} \models \beta$.

Ejemplo 4.15

Si $\Gamma = \{\forall x(x < \mathbf{S}x)\}$ *entonces* $\Gamma \models (\mathbf{0} < \mathbf{S0})$.

Demostración Sea Ψ una estructura para el lenguaje y h una asignación de variables. Supongamos que Ψ satisface a todas las fórmulas en Γ con la asignación h.

entonces $\models_\Psi \ \forall x(x < \mathbf{S}x) \quad [h]$

entonces para todo $d \in |\Psi|$ tenemos que $\models_\Psi \ x < \mathbf{S}x \quad [h(x \mid d)]$

entonces para todo $d \in |\Psi|$ tenemos que $(\overline{h(x \mid d)}(x), \overline{h(x \mid d)}(\mathbf{S}x)) \in <^\Psi$

entonces para todo $d \in |\Psi|$ tenemos que $(d, \mathbf{S}^\Psi(d)) \in <^\Psi$

en particular si $d = \mathbf{0}^\Psi$ tenemos que $(\mathbf{0}^\Psi, \mathbf{S}^\Psi(\mathbf{0}^\Psi)) \in <^\Psi$

entonces $(\overline{h}(\mathbf{0}), \overline{h}(\mathbf{S0})) \in <^\Psi$

entonces $\models_\Psi \ \mathbf{0} < \mathbf{S0} \quad [h]$

Por lo tanto $\Gamma \models (\mathbf{0} < \mathbf{S0})$. ◇

Nótese que solamente supusimos que Ψ satisfizo todas las fórmulas en Γ con la asignación h. Exceptuando esto, la elección de Ψ y h es totalmente arbitraria, de modo que $|\Psi|$ podría ser "cualquier" conjunto y no necesariamente el de los números naturales.

Definición 4.16 Equivalencia lógica

Si α *y* β *son fórmulas tales que* $\alpha \models \beta$ *y* $\beta \models \alpha$ *decimos que son* lógicamente equivalentes.

Por ejemplo:

$(\neg\alpha \vee \neg\beta)$ y $\neg(\alpha \wedge \beta)$

$(\alpha \vee \beta)$ y $((\neg\alpha) \longrightarrow \beta)$

$(\alpha \wedge \beta)$ y $\neg(\alpha \longrightarrow (\neg\beta))$

$(\alpha \longrightarrow \beta)$ y $(\neg\alpha \vee \beta)$

Definición 4.17 Cerradura universal y cerradura existencial

Si α *es una fórmula cuyas variables libres son* $x_1, x_2, \ldots, x_n$ *entonces definimos la* cerradura universal *de* α *como* $\forall(\alpha) = \forall x_1 \forall x_2 \ldots \forall x_n \alpha$ *y la* cerradura existencial *de* α *como* $\exists(\alpha) = \exists x_1 \exists x_2 \ldots \exists x_n \alpha$.

Por ejemplo:

Si $\alpha = x{<}y \longrightarrow (\exists z\ x + z{\approx}y)$, entonces

$$\forall(\alpha) = \forall x \forall y(x{<}y \longrightarrow (\exists z\ x + z{\approx}y))$$

$$\exists(\alpha) = \exists x \exists y(x{<}y \longrightarrow (\exists z\ x + z{\approx}y))$$

Definición 4.18 Verdad y modelo

Si δ es un enunciado y Ψ una estructura entonces sucede exactamente alguna de:

(a) Ψ satisface δ con cualquier asignación de variables.

(b) Ψ no satisface δ con ninguna asignación de variables.

Ya que los enunciados no tienen variables libres, y resulta irrelevante la asignación de variables que elijamos.

Si ocurrió (a) decimos que δ es verdadero *respecto a Ψ o que Ψ es un* modelo *de δ. Escribimos*

$$\models_{\Psi} \delta$$

Si ocurrió (b) decimos que δ es falso *respecto a Ψ o que Ψ no es modelo de δ.*

Si Γ es un conjunto de enunciados, decimos que Ψ es un modelo de Γ si y sólo si es modelo de cada uno de sus elementos. De modo que $\Gamma \models \delta$ si y sólo si todo modelo de Γ es también un modelo de δ.

Definición 4.19 Conjunto satisfactible de fórmulas

Sea Γ un conjunto de fórmulas. Decimos que Γ es satisfactible *si existen una estructura Ψ para el lenguaje y una asignación de variables h tal que Ψ satisface a todos los elementos de Γ con h. En caso contrario decimos que Γ es* insatisfactible.

Notemos que si Γ es un conjunto de enunciados, entonces Γ es satisfactible si y sólo si tiene un modelo.

Observación 4.20 *Si Γ es un conjunto de fórmulas insatisfactible, entonces el conjunto de todas estructuras Ψ del lenguaje con sus respectivas asignaciones de variables h tal que Ψ satisface a todos los elementos de Γ con h es vacío. Entonces para cualquier fórmula δ se cumple $\Gamma \models \delta$ por vacuidad. El hecho que Γ sea insatisfactible es equivalente a pedir que exista una formula α tal que $\Gamma \models (\alpha \wedge \neg\alpha)$.*

Teorema 4.21 *Sea* Γ *un conjunto de fórmulas y* δ *una fórmula entonces* $\Gamma \models \delta$ *si y solo si* $\Gamma \cup \{\neg\delta\}$ *es insatisfactible.*

Demostración

$[\Longrightarrow]$ Sea Ψ cualquier estructura para el lenguaje y h cualquier asignación de variables. Si Ψ satisface a todos los elementos de Γ con h, tenemos por hipótesis que $\models_\Psi \delta\ [h]$. Entonces no se cumple $\models_\Psi \neg\delta\ [h]$. Como la elección de Ψ y h fue arbitraria $\Gamma \cup \{\neg\delta\}$ es insatisfactible.

$[\Longleftarrow]$ Si Γ es insatisfactible, entonces por la observación 4.20 tenemos $\Gamma \models \delta$. Si Γ es satisfactible, sea Ψ cualquier estructura para el lenguaje y h cualquier asignación de variables, tal que Ψ satisface a todos los elementos de Γ con h. Como por hipótesis $\Gamma \cup \{\neg\delta\}$ es insatisfactible, entonces no puede suceder $\models_\Psi \neg\delta\ [h]$. Por lo tanto $\models_\Psi \delta\ [h]$. Como la elección de Ψ y h fue arbitraria concluimos que $\Gamma \models \delta$. ◇

4.2 Modelos de Herbrand

Definición 4.22 Universo de Herbrand

Sea $\mathcal{L}$ *un lenguaje de primer orden. Definimos* $U_\mathcal{L}$ *el* universo de Herbrand *para el lenguaje* $\mathcal{L}$ *como el conjunto de todos los términos sin variables del lenguaje* $\mathcal{L}$*, i.e., los términos formados a partir de los símbolos de constante. Si* $\mathcal{L}$ *no tiene símbolos de constante agregamos uno con este propósito.*

En el ejemplo 4.1:

$$
\begin{aligned}
U_\mathcal{L} = \{\ & \mathbf{0},\\
& \mathbf{0+0},\quad \mathbf{0\cdot 0},\quad \mathbf{E(0,0)},\\
& \mathbf{S0},\\
& \mathbf{0+S0},\quad \mathbf{0\cdot S0},\quad \mathbf{E(0,S0)},\\
& \mathbf{S0+0},\quad \mathbf{S0\cdot 0},\quad \mathbf{E(S0,0)},\\
& \mathbf{S0+S0},\quad \mathbf{S0\cdot S0},\quad \mathbf{E(S0,S0)},\\
& \mathbf{SS0},\ldots\}
\end{aligned}
$$

Definición 4.23 Base de Herbrand

Sea $\mathcal{L}$ *un lenguaje de primer orden. Definimos* $B_\mathcal{L}$ *la* base de Herbrand *para el lenguaje* $\mathcal{L}$ *como el conjunto de todas las fórmulas atómicas sin variables. Formalmente cualquier expresión de la forma* $Pt_1t_2\ldots t_n$ *donde* P *es un símbolo de predicado n-ario y* $t_1, t_2, \ldots, t_n$ *son términos sin variables.*

De modo que la base de Herbrand se obtiene adjuntando los símbolos de predicado de $\mathcal{L}$ a los elementos de $U_{\mathcal{L}}$.

En el ejemplo 4.1:

$$\begin{aligned} B_{\mathcal{L}} = \{\ & \mathbf{0} < \mathbf{0}, \\ & \mathbf{0} \approx \mathbf{0}, \\ & \mathbf{0} < \mathbf{0}+\mathbf{0}, \quad \mathbf{0} < \mathbf{0}\cdot\mathbf{0}, \quad \mathbf{0} < \mathbf{E}(\mathbf{0},\mathbf{0}), \\ & \mathbf{0} \approx \mathbf{0}+\mathbf{0}, \quad \mathbf{0} \approx \mathbf{0}\cdot\mathbf{0}, \quad \mathbf{0} \approx \mathbf{E}(\mathbf{0},\mathbf{0}), \\ & \mathbf{0}+\mathbf{0} < \mathbf{0}, \quad \mathbf{0}\cdot\mathbf{0} < \mathbf{0}, \quad \mathbf{E}(\mathbf{0},\mathbf{0}) < \mathbf{0}, \ldots \} \end{aligned}$$

Definición 4.24 Estructura de Herbrand

Sea $\mathcal{L}$ un lenguaje de primer orden. Decimos que Ψ es una estructura de Herbrand *para el lenguaje si:*

1. $|\Psi| = U_{\mathcal{L}}$
2. *Para todo símbolo de constante c, $c^{\Psi} = c$. Es decir las constantes son asignadas a sí mismas.*
3. *Si f es un símbolo de función n-aria entonces $f^{\Psi} : (U_{\mathcal{L}})^n \longrightarrow U_{\mathcal{L}}$ es tal que $f^{\Psi}(t_1, t_2, \ldots, t_n) = ft_1, t_2, \ldots, t_n$.*

Notemos que en las estructuras de Herbrand la asignación del universo, las constantes y los símbolos de función es fija. De modo que la estructura queda totalmente determinada por las relaciones asignadas a los símbolos de predicado. A su vez notemos que si P es un símbolo de predicado n-ario entonces $P^{\Psi} \subseteq (U_{\mathcal{L}})^n$ de modo que, las relaciones quedan totalmente determinadas por un subconjunto de $B_{\mathcal{L}}$, la base de Herbrand. Por ejemplo:

El conjunto

$$\left\{\mathbf{0} < \mathbf{S0},\ \mathbf{0} < \mathbf{S^2 0},\ \mathbf{0} < \mathbf{S^3 0},\ \mathbf{S0} \approx \mathbf{S0},\ \mathbf{S0} \approx \mathbf{S^2 0},\ \mathbf{S0} \approx \mathbf{S^3 0}\right\} \subseteq B_{\mathcal{L}}$$

determina que

$$\begin{aligned} <^{\Psi} &= \{(\mathbf{0},\mathbf{S0}),\ (\mathbf{0},\mathbf{S^2 0}),\ (\mathbf{0},\mathbf{S^3 0})\} \\ \approx^{\Psi} &= \{(\mathbf{S0},\mathbf{S0}),\ (\mathbf{S0},\mathbf{S^2 0}),\ (\mathbf{S0},\mathbf{S^3 0})\} \end{aligned}$$

para una estructura de Herbrand Ψ

Definición 4.25 Conjunto asociado a una estructura de Herbrand

Sea $\mathcal{L}$ un lenguaje de primer orden. Sea Ψ una estructura de Herbrand para el lenguaje $\mathcal{L}$. Definimos $H(\Psi)$, el conjunto asociado a la estructura Ψ, *como:*

$$H(\Psi) = \{Pt_1t_2\ldots t_n \mid \begin{array}{l}(t_1, t_2, \ldots, t_n) \in P^{\Psi} \text{ donde} \\ P \text{ es cualquier símbolo de predicado } n\text{-ario en } \mathcal{L} \\ y\ n \geq 1\ \}\end{array}$$

Notemos que $H(\Psi) \subseteq B_{\mathcal{L}}$. De hecho $H(\Psi)$ caracteriza a Ψ en el sentido de que si $I \subseteq B_{\mathcal{L}}$, entonces existe una única estructura de Herbrand Ψ tal que $H(\Psi) = I$

Definición 4.26 Modelo de Herbrand

Sea Γ un conjunto de enunciados en un lenguaje de primer orden. Decimos que Ψ es un modelo de Herbrand *para Γ si, Ψ es un modelo de Γ y Ψ es estructura de Herbrand para el lenguaje dado.*

Retomando el ejemplo 4.1:

Si $\Gamma = \{\ \forall x\ \mathbf{0} < \mathbf{S}x\ \}$ entonces la estructura de Herbrand Ψ determinada por el conjunto

$$\begin{array}{rcl} H(\Psi) & = & \{\mathbf{0} < \mathbf{S0}, \\ & & \mathbf{0} < \mathbf{S}(\mathbf{0}+\mathbf{0}),\ \ \mathbf{0} < \mathbf{S}(\mathbf{0}\cdot\mathbf{0}),\ \ \mathbf{0} < \mathbf{S}(\mathbf{E}(\mathbf{0},\mathbf{0})), \\ & & \mathbf{0} < \mathbf{SS0}, \\ & & \mathbf{0} < \mathbf{S}(\mathbf{0}+\mathbf{S0}),\ \ \mathbf{0} < \mathbf{S}(\mathbf{0}\cdot\mathbf{S0}),\ \ \mathbf{0} < \mathbf{S}(\mathbf{E}(\mathbf{0},\mathbf{S0})),\ldots\} \\ & = & \{\mathbf{0} < \mathbf{S}t \mid t \in U_{\mathcal{L}}\} \end{array}$$

es un modelo de Herbrand para Γ.

4.3 Sistemas Formales

Ahora daremos algunas definiciones de sistemas formales. Nos restringiremos exclusivamente a lenguajes de primer orden; para un tratamiento más general puede consultarse [Men64].

Definición 4.27 Regla de Inferencia

Sea $\mathcal{L}$ un lenguaje de primer orden. Sea $\mathcal{F}$ el conjunto de todas las fórmulas en $\mathcal{L}$. Una regla de inferencia *de aridad n, es una relación $\Phi \subseteq \mathcal{F}^n$, tal que para cualesquiera $\delta_1, \delta_2, \ldots, \delta_n$ (con $n > 0$) fórmulas del lenguaje existe una manera efectiva[2] de decidir si $(\delta_1, \delta_2, \ldots, \delta_n) \in \Phi$.*

En tal caso decimos que δ_n es una consecuencia directa *de $\delta_1, \delta_2, \ldots, \delta_{n-1}$ en virtud de la regla Φ.*

[2] Una discusión detallada sobre el significado de la efectividad puede consultarse en [End87, p.p. 93-97].

Definición 4.28 Sistema formal

Sea $\mathcal{L}$ un lenguaje de primer orden. Un sistema formal *es un conjunto finito de reglas de inferencia en $\mathcal{L}$.*

Definición 4.29 Deducción formal

Sea $\mathcal{L}$ un lenguaje de primer orden. Sea $\Gamma \cup \{\delta\}$ un conjunto de fórmulas de $\mathcal{L}$. Sea SF un sistema formal para $\mathcal{L}$. Decimos que δ se deduce *de Γ en SF si y solo si existe una sucesión finita $\delta_1, \delta_2, \ldots, \delta_n$ de fórmulas en L tales que:*

1. $\delta = \delta_n$
2. *Para todo $k = 1, \ldots, n$ se cumple $\delta_k \in \Gamma$ ó δ_k es una consecuencia directa de un subconjunto de $\{\delta_1, \delta_2, \ldots, \delta_{k-1}\}$ en virtud de alguna regla en SF.*

Escribimos entonces $\Gamma \vdash_{SF} \delta$.

Ejemplo 4.30

En el ejemplo 4.1, sea $\mathcal{M}$ la regla de inferencia conocida como *modus ponens*, i.e.,

$$\mathcal{M} = \{ (\alpha, \alpha \longrightarrow \beta, \beta) \mid \alpha, \beta \text{ fórmulas de } \mathcal{L} \}$$

Sea D el sistema formal cuya única regla de inferencia es $\mathcal{M}$.

Si $\Gamma = \{\forall x\ \mathbf{0} < x \longrightarrow \mathbf{0} < \mathbf{S0},\ \forall x\ \mathbf{0} < x\}$ entonces

$$\delta_1 = \forall x\ \mathbf{0} < x$$

$$\delta_2 = \forall x\ \mathbf{0} < x \longrightarrow \mathbf{0} < \mathbf{S0}$$

$$\delta_3 = \mathbf{0} < \mathbf{S0}$$

es una deducción formal de $\mathbf{0} < \mathbf{S0}$ a partir de Γ. Por lo tanto:

$$\Gamma \vdash_D \mathbf{0} < \mathbf{S0}$$

Definición 4.31 Validez de un sistema formal

Sea SF un sistema formal para un lenguaje de primer orden L. Sea Γ un conjunto de fórmulas en $\mathcal{L}$. Decimos que SF es un sistema formal válido *para Γ si, para toda fórmula δ del lenguaje, si $\Gamma \vdash_{SF} \delta$ entonces $\Gamma \models \delta$.*

Definición 4.32 Completud de un sistema formal

Sea SF un sistema formal para un lenguaje de primer orden L. Sea Γ un conjunto de fórmulas en $\mathcal{L}$. Decimos que SF es un sistema formal completo *para Γ, si para toda fórmula δ del lenguaje, si $\Gamma \models \delta$ entonces $\Gamma \vdash_{SF} \delta$.*

En los capítulos siguientes haremos referencia a la completud y validez relativas a un conjunto restringido de fórmulas Σ, de un sistema formal SF. Es decir que para toda fórmula δ elemento del conjunto Σ, se cumpla $\Gamma \models \delta$ si y sólo si $\Gamma \vdash_{SF} \delta$.

5. Programación Lógica

Ahora presentamos la herramienta básica de la programación lógica y enunciamos la completud y validez de la refutación SLD como sistema formal para programas lógicos. Nos basamos en el texto de Lloyd[Llo87]. Muchos ejemplos y resultados están tomados de ahí sin demostración.

5.1 Definiciones Básicas

Como en el capítulo anterior, para todas las definiciones y teoremas trabajaremos sobre un mismo lenguaje de primer orden $\mathcal{L}$.

Definición 5.1 Listas

Las listas son un tipo especial de términos que definimos inductivamente de la siguiente manera:

1. *nil es una* lista.
2. *Si l es una lista y t es un término entonces $cons(t, l)$ es una lista. A t se le llama* cabeza *y a l* cuerpo *de la lista.*

nil es un símbolo de constante y cons un símbolo de función binaria. A nil le llamamos lista vacía. *En la práctica usamos las siguientes abreviaturas:*

[] *para denotar nil*

$[t \mid l]$ *para denotar* $cons(t, l)$

Ejemplo 5.2

$$[23 \mid [34 \mid [56 \mid []]]]$$

abrevia a la lista

$$cons(23, cons(34, cons(56, nil)))$$

que también denotamos como

$[23, 34, 56]$

$[23 \mid [34, 56]]$

$[23 \mid [34 \mid [56]]]$

Nótese que $[56]$ abrevia a la lista $[56 \mid []]$. Hay que tener cuidado con esta notación, ya que por ejemplo, $[[23] \mid [34, 56]]$ denota la lista $[[23], 34, 56]$ y no la lista $[23, 34, 56]$ como podría pensarse. Formalmente:

$$[] = nil$$

y

$$\begin{aligned}[a_1, a_2, \ldots, a_n] &= [a_1 \mid [a_2, \ldots, a_n]] \\ &= cons(a_1, [a_2, \ldots, a_n]) \text{ para todo } n \geq 1\end{aligned}$$

Definición 5.3 Cláusula

Decimos que una fórmula es una cláusula, si es de la forma:

$$\forall(l_1 \vee l_2 \vee \ldots \vee l_n)$$

donde las l_j son literales y $n \geq 0$. Cuando $n = 0$, la llamamos cláusula vacía *(ver definición 5.9).*

Por ejemplo:

Si p, q, r son símbolos de predicado binario, f es un símbolo de función binario y X, Y, Z son variables para el lenguaje entonces:

$$\forall X \forall Y \forall Z(p(X, Z) \vee \neg q(X, Y) \vee \neg r(Y, Z))$$

$$\forall X \forall Y(\neg p(X, Y) \vee r(f(X, Y), a)))$$

son cláusulas.

En los siguientes ejemplos supondremos que las letras mayúsculas denotan variables y las letras minúsculas denotan símbolos de predicado, símbolos de función o constantes para el lenguaje de primer orden. La aridad de todas ellas quedará determinada por el contexto.

Como toda cláusula se compone de literales, sin perder generalidad, podemos suponer que tiene la forma:

$$\forall X_1 \forall X_2 \ldots \forall X_s(a_1 \vee \ldots \vee a_k \vee \neg b_1 \vee \ldots \vee \neg b_h) \qquad (5.1)$$

donde las a_i y las b_j son átomos y $X_1, X_2, \ldots, X_s$ son todas las variables que aparecen en ellos. Notemos que (5.1) es lógicamente equivalente a:

$$\forall X_1 \forall X_2 \ldots \forall X_s((a_1 \vee \ldots \vee a_k) \vee \neg(b_1 \wedge \ldots \wedge b_h))$$

que a su vez es lógicamente equivalente a:

$$\forall X_1 \forall X_2 \ldots \forall X_s((a_1 \vee \ldots \vee a_k) \longleftarrow (b_1 \wedge \ldots \wedge b_n))$$

y en adelante la denotaremos simplemente como

$$a_1, \ldots, a_k \longleftarrow b_1, \ldots, b_n$$

A $b_1, \ldots, b_n$ le llamamos el *antecedente* y a $a_1, \ldots, a_k$ el *consecuente* de la cláusula. Obsérvese que todas las variables están cuantificadas universalmente, y que las comas en el antecedente denotan conjunción, mientras que en el consecuente denotan disyunción.

Definición 5.4 Cláusula definida

Una cláusula definida *es una cláusula de la forma:*

$$a \longleftarrow b_1, \ldots, b_n$$

Donde $a, b_1, b_2, \ldots, b_n$ *son átomos y* $n \geq 0$. *A* a *le llamamos* cabeza *y a* $b_1, \ldots, b_n$ cuerpo *de la cláusula.*

Definición 5.5 Cláusula unitaria

Una cláusula unitaria *es una cláusula definida*

$$a \longleftarrow$$

i.e., una cláusula definida cuyo cuerpo es vacío.

Definición 5.6 Programa definido

Un programa definido *es un conjunto finito y no vacío de cláusulas definidas.*

Si q es cabeza de alguna cláusula en un programa definido Γ, entonces al conjunto de cláusulas en Γ cuya cabeza es q le llamamos *definición* de q.

Ejemplo 5.7

$busca(T, P) \longleftarrow compara(T, P)$

$busca([X \mid Y], P) \longleftarrow busca(Y, P)$

$compara(X, []) \longleftarrow$

$compara([X \mid Y], [X \mid Z]) \longleftarrow compara(Y, Z)$

es un programa definido. La definición del predicado *busca* es:

$busca(T, P) \longleftarrow compara(T, P)$

$busca([X \mid Y], P) \longleftarrow busca(Y, P)$

Definición 5.8 Meta

Una meta *es una cláusula de la forma:*

$$\longleftarrow b_1, b_2, \ldots, b_n$$

con $n > 0$, esto es, una cláusula con consecuente vacío. A cada b_i le llamamos submeta.

Definición 5.9 Cláusula vacía

La cláusula vacía *es aquella cláusula con antecedente y consecuente vacíos. La denotamos como $\square$. Si en un sistema formal deducimos la cláusula vacía significa que se ha inferido una contradicción.*

5.2 Unificación

Definición 5.10 Expresión simple

Definimos una expresión simple[1] *como un término, una literal o una conjunción o disyunción de literales.*

Definición 5.11 Sustitución

Una sustitución *es un conjunto finito $\theta = \{X_1/t_1, X_2/t_2, \ldots, X_n/t_n\}$ donde $n \geq 0$, $X_1, X_2, \ldots, X_n$ son variables distintas y cada t_i es un término distinto de X_i para todo $i = 1, \ldots n$.*

Definición 5.12 Instancia

Sea $\theta = \{X_1/t_1, X_2/t_2, \ldots, X_n/t_n\}$ una sustitución y β una expresión simple. Entonces definimos $\beta\theta$, la instancia *de β al aplicarle θ, como la expresión que se obtiene al sustituir todas las presencias de la variable X_i en β por el término t_i para todo $i = 1, \ldots, n$. A cada elemento X_i/t_i de la sustitución le llamamos* liga.

Por ejemplo:

Si $\beta = p(X, f(Y, b), Z, W)$ y $\theta = \{X/a, Y/Z, Z/g(c, d)\}$, entonces $\beta\theta = p(a, f(Z, b), g(c, d), W)$.

La definición se extiende a un conjunto finito de expresiones simples $S = \{\beta_1, \beta_2, \ldots, \beta_m\}$, de la siguiente manera: $S\theta = \{\beta_1\theta, \beta_2\theta, \ldots, \beta_m\theta\}$.

[1] Aquí nos apartamos de la definición en [Llo87, p.p. 20] para ser consistentes con nuestra definición 4.2.

Definición 5.13 Composición de sustituciones

Sean $\theta = \{U_1/s_1, U_2/s_2, \ldots, U_m/s_m\}$ *y* $\sigma = \{X_1/t_1, X_2/t_2, \ldots, X_n/t_n\}$ *sustituciones. Entonces definimos la* composición $\theta\sigma$ *como el conjunto que se obtiene de:*

$$\{U_i/s_1\sigma, U_2/s_2\sigma, \ldots, U_m/s_m\sigma, X_1/t_1, X_2/t_2, \ldots, X_n/t_n\}$$

quitando toda liga $U_i/s_i\sigma$ *tal que* $U_i = s_i\sigma$ *y toda liga* X_j/t_j *tal que* $X_j \in \{U_1, U_2, \ldots, U_m\}$.

Por ejemplo:

Si $\theta = \{X/f(Y), Y/Z\}$ y $\sigma = \{X/a, Y/b, Z/Y\}$, entonces

$$\theta\sigma = \{X/f(b), Z/Y\}$$

Definición 5.14 Sustitución identidad

A la sustitución dada por el conjunto vacío le llamamos sustitución identidad *y la denotamos como* ε.

Definición 5.15 Variantes

Sean α *y* β *expresiones simples. Decimos que* α *y* β *son* variantes *si existen sustituciones* θ *y* σ *tales que* $\alpha = \beta\theta$ *y* $\beta = \alpha\sigma$. *También decimos que* α *es una variante de* β *y viceversa.*

Por ejemplo:

$p(f(X,Y), g(Z), a)$ es una variante de $p(f(Y,X), g(U), a)$, pero $p(X,X)$ no es una variante de $p(X,Y)$.

Proposición 5.16 *Si* α *y* β *son variantes, entonces* α *se obtiene de* β *renombrando variables y viceversa.*

Demostración Ver [Llo87, p.p. 22]

Proposición 5.17 *Si* θ, σ, τ *son sustituciones, entonces:*

(a) $\theta\varepsilon = \varepsilon\theta = \theta$

(b) $(\beta\theta)\sigma = \beta(\theta\sigma)$ *para toda expresión simple* β

(c) $(\theta\sigma)\tau = \theta(\sigma\tau)$

Demostración Ver [Llo87, p.p. 21]

Definición 5.18 Unificador

Sea S un conjunto finito de términos o átomos. Decimos que una sustitución θ es un unificador *para S si $S\theta$ tiene solamente un elemento. Un unificador θ para S es llamado unificador más general (mgu) de S, si para todo unificador σ de S, existe una sustitución τ tal que $\sigma = \theta\tau$.*

Por ejemplo:

$S = \{p(f(X), Z),\ p(Y, a)\}$ tiene como unificador a $\sigma = \{Y/f(a),\ X/a,\ Z/a\}$, y como unificador más general a $\theta = \{Y/f(X),\ Z/a\}$.

El unificador más general no es único, pero cualesquiera dos de ellos son iguales módulo nombres de variables (ver [Llo87, p.p. 22-23]). Si no existe unificador para el conjunto, decimos que no es unificable.

Proposición 5.19 *Sea S un conjunto finito de términos o átomos. Entonces existe un algoritmo tal que:*

(a) Si S es unificable, el algoritmo termina y da un mgu para S

(b) Si S no es unificable, el algoritmo termina sin éxito.

Demostración Ver [Llo87, p.p. 25].

5.3 Teorema de Herbrand

Proposición 5.20 *Sea δ una conjunción o disyunción de literales, X una variable en δ, t un término sin variables, μ una estructura para el lenguaje de primer orden y s una asignación de variables para μ. Si θ es la sustitución $\theta = \{X/t\}$, entonces:*

$$\models_\mu \delta\theta \quad [s] \quad syss \quad \models_\mu \delta \quad [s(X \mid \overline{s}(t))]$$

Demostración Es un corolario inmediato de un resultado de la lógica de primer orden llamado el lema de sustitución. Su demostración puede consultarse en [End87, p.p. 179-180].

Teorema 5.21 *Si Γ es un conjunto de cláusulas en un lenguaje de primer orden $\mathcal{L}$, entonces Γ tiene un modelo si y sólo si Γ tiene un modelo de Herbrand.*

Demostración

$\boxed{\Leftarrow}$ Es obvio.

$\boxed{\Longrightarrow}$ Recordemos que por la definición 4.24, para definir una estructura de Herbrand sólo tenemos que dar la asignación para los símbolos de predicado. Por hipótesis existe una estructura μ que es un modelo para Γ. Sea $U_{\mathcal{L}}$ el universo de Herbrand para el lenguaje de primer orden. Sea Ψ la estructura de Herbrand determinada por los conjuntos:

$$p^{\Psi} = \{(t_1, t_2, \ldots, t_n) \mid \models_{\mu} pt_1t_2 \ldots t_n \quad y \quad t_i \in U_{\mathcal{L}}\ \forall i = 1, \ldots, n\}$$

para todo p símbolo de predicado del lenguaje de aridad n, y para todo n entero positivo.

Afirmamos que Ψ es un modelo de Herbrand para Γ.

Demostración

Primero notemos que si p es un símbolo de predicado n-ario y $t_1, t_2, \ldots, t_n$ son términos sin variables, entonces

$$\models_{\Psi} pt_1t_2 \ldots t_n \text{ syss } \models_{\mu} pt_1t_2 \ldots t_n$$

por definición de p^{Ψ}.

Análogamente

$$\begin{array}{ll} \models_{\Psi} \neg pt_1t_2 \ldots t_n & \text{syss no se cumple } \models_{\Psi} pt_1t_2 \ldots t_n \\ & \text{syss } pt_1t_2 \ldots t_n \notin p^{\Psi} \\ & \text{syss no se cumple } \models_{\mu} pt_1t_2 \ldots t_n \text{ por definición de } p^{\Psi} \\ & \text{syss } \models_{\mu} \neg pt_1t_2 \ldots t_n \end{array}$$

Se sigue que si α es una disyunción de literales que no tiene variables, entonces:

$$\models_{\Psi} \alpha \Longleftrightarrow \models_{\mu} \alpha \tag{5.2}$$

Sabemos que cualquier elemento de Γ es de la forma $\forall(\alpha)$, donde α es una disyunción de literales. Sea h cualquier asignación de variables para Ψ. Sean $X_1, X_2, \ldots, X_n$ son todas las variables en α. Queremos mostrar que:

$$\models_{\Psi} \forall X_1 \forall X_2 \cdots \forall X_n \alpha \quad [h] \tag{5.3}$$

Por ser Ψ una estructura de Herbrand sabemos que $h : V \longrightarrow U_{\mathcal{L}}$. De modo que (5.3) es equivalente a mostrar que para cualesquiera $t_1, t_2, \ldots, t_n$ términos sin variables se cumpla:

$$\models_\Psi \alpha \quad [h(X_1 \mid t_1,\ X_2 \mid t_2, \ldots, X_n \mid t_n)] \tag{5.4}$$

De nuevo como Ψ es una estructura de Herbrand, es fácil mostrar que para todo término sin variables t se tiene $\overline{h}(t) = t$. Entonces (5.4) es equivalente a

$$\models_\Psi \alpha \quad [h(X_1 \mid \overline{h}(t_1),\ X_2 \mid \overline{h}(t_2), \ldots, X_n \mid \overline{h}(t_n))] \tag{5.5}$$

Si definimos la sustitución θ como $\theta = \{X_1/t_1,\ X_2/t_2, \ldots, X_n/t_n\}$, y aplicamos repetidamente la proposición 5.20, resulta que (5.5) es equivalente a:

$$\models_\Psi \alpha\theta \quad [h]$$

y esto último es equivalente a $\models_\Psi \alpha\theta$ ya que $\alpha\theta$ no tiene variables. Por lo tanto,

$$\models_\Psi \forall X_1 \forall X_2 \cdots \forall X_n \alpha\ [h] \Longleftrightarrow \models_\Psi \alpha\theta \tag{5.6}$$

Pero recordemos que μ es un modelo de Γ. Entonces si s es cualquier asignación de variables para μ, tenemos que:

$$\models_\mu \forall X_1 \forall X_2 \cdots \forall X_n \alpha \quad [s]$$

En particular,

$$\models_\mu \alpha \quad [s(X_1 \mid \overline{s}(t_1),\ X_2 \mid \overline{s}(t_2), \ldots, X_n \mid \overline{s}(t_n))] \tag{5.7}$$

Nuevamente, aplicando repetidamente la proposición 5.20, tenemos que (5.7) es equivalente a:

$$\models_\mu \alpha\theta \quad [s]$$

entonces, $\models_\mu \alpha\theta$, ya que $\alpha\theta$ no tiene variables. Como $\alpha\theta$ es una disyunción de literales, entonces por (5.2) sabemos que:

$$\models_\Psi \alpha\theta \text{ syss } \models_\mu \alpha\theta$$

y por lo tanto,

$$\models_\Psi \alpha\theta$$

Por (5.6) concluimos que Ψ es un modelo de Herbrand para Γ.

$\Diamond$

Corolario 5.22 *Si Γ es un conjunto de cláusulas en un lenguaje de primer orden $\mathcal{L}$, entonces Γ es insatisfactible si y sólo si Γ no tiene modelos de Herbrand.*

Demostración Es inmediato de la definición 4.19 (conjunto satisfactible).

Observación 5.23 *En la demostración del teorema 5.21 es esencial que Γ sea un conjunto de cláusulas. En general el resultado no se cumple si Γ es un conjunto arbitrario de enunciados, como se ilustra a continuación:*

Ejemplo 5.24

Sea $\Gamma = \{p(a), \exists X \neg p(X)\}$ en un lenguaje cuyos únicos símbolos de predicado y constante son p y a respectivamente. Sea μ la estructura tal que:

$$|\mu| = \{0, 1\}$$
$$a^\mu = 0$$
$$p^\mu = \{0\}$$

Entonces

$$\models_\mu p(a)$$

ya que $a^\mu = 0$ y $0 \in \{0\} = p^\mu$. Análogamente si h es cualquier asignación de variables, entonces

$$\models_\mu \neg p(X) \quad [h(X \mid 1)]$$

ya que $1 \notin \{0\} = p^\mu$. Por lo tanto

$$\models_\mu \exists X \neg p(X)$$

Por lo tanto μ es un modelo de Γ. Sin embargo los únicos posibles modelos de Herbrand para Γ, serían Ψ y Λ donde: $p^\Psi = \emptyset$ y $p^\Lambda = \{a\}$. Pero $p(a)$ no es verdadero en Ψ y $\exists X \neg p(X)$ no es verdadero en Λ. Por lo tanto, Γ no tiene modelos de Herbrand.

Teorema 5.25 *Todo programa definido es satisfactible.*

Demostración Sea Γ un programa definido. Por la definición 4.25, sea Ψ la única estructura de Herbrand tal que:

$$H(\Psi) = B_\mathcal{L}$$

donde $B_{\mathcal{L}}$ es la base de Herbrand para el lenguaje de primer orden y $H(\Psi)$ el conjunto asociado a la estructura de Herbrand Ψ. Es decir, $p^{\Psi} = (U_{\mathcal{L}})^n$ para todo p símbolo de predicado del lenguaje con aridad n, variando n sobre los enteros positivos.

Afirmamos que Ψ es un modelo de Herbrand para Γ.

Demostración

Todos los elementos de Γ son cláusulas definidas, por lo que tienen la forma:

$$a \longleftarrow b_1, \ldots, b_n \text{ con } n \geq 0$$

que denota:

$$\forall(a \vee \neg(b_1 \wedge \ldots \wedge b_n)) \text{ donde } a, b_1, \ldots, b_n \text{ son átomos.}$$

Entonces queremos mostrar que:

$$\models_{\Psi} \forall(a \vee \neg(b_1 \wedge \ldots \wedge b_n))$$

para lo cual es suficiente probar que:

$$\models_{\Psi} \forall(a)$$

Supongamos que $X_1, X_2, \ldots, X_s$ son todas las variables en a. Sean $t_1, t_2, \ldots, t_s$ términos sin variables escogidos arbitrariamente. Definamos la sustitución:

$$\theta = \{X_1/t_1,\ X_2/t_2, \ldots, X_s/t_s\}$$

Dado que Ψ es una estructura de Herbrand, por un argumento análogo al usado en el teorema 5.21, basta entonces ver que:

$$\models_{\Psi} a\theta$$

pero esto es cierto ya que $a\theta$ no tiene variables y por lo tanto $a\theta \in B_{\mathcal{L}} = H(\Psi)$.

Por lo tanto Ψ es un modelo de Herbrand para Γ.

$\Diamond$

Observación 5.26 *El teorema 5.25 no se cumple para conjuntos arbitrarios de cláusulas.*

Por ejemplo el conjunto:

$$\{\forall X\ p(X),\ \forall X \neg p(X)\}$$

no es satisfactible.

5.4 Resolución SLD

Supongamos que tenemos un programa definido Γ y una meta G. Si $G = \leftarrow b_1, b_2, \ldots, b_n$ y $X_1, X_2, \ldots, X_s$ son todas las variables que aparecen en G entonces:

$$G = \forall X_1 \forall X_2 \ldots \forall X_s(\neg(b_1 \wedge b_2 \wedge \ldots \wedge b_n))$$

que es lógicamente equivalente a:

$$\neg(\exists X_1 \exists X_2 \ldots \exists X_s(b_1 \wedge b_2 \wedge \ldots \wedge b_n))$$

Ahora supongamos que queremos mostrar que $\exists X_1 \exists X_2 \ldots \exists X_s(b_1 \wedge b_2 \wedge \ldots \wedge b_n)$ (que es lógicamente equivalente a $\neg G$) es una consecuencia lógica de Γ. Por el teorema 4.21, basta ver que $\Gamma \cup \{G\}$ es insatisfactible. A su vez, por la observación 4.20, esto se logra encontrando una fórmula α tal que $\Gamma \cup \{G\} \models (\alpha \wedge \neg\alpha)$. Es decir, mostrando que una contradicción es consecuencia lógica de $\Gamma \cup \{G\}$. En particular como la cláusula vacía significa contradicción, es suficiente probar que $\Gamma \cup \{G\} \models \square$.

Ahora supongamos que ya sabemos que $\Gamma \models \neg G$. Por el teorema 5.25, Γ tiene un modelo de Herbrand Ψ, entonces:

$$\models_\Psi \exists X_1 \exists X_2 \cdots \exists X_s(b_1 \wedge b_2 \wedge \cdots \wedge b_n)$$

entonces, existen $t_1, t_2, \cdots, t_s$ términos sin variables, tales que:

$$\models_\Psi (b_1 \wedge b_2 \wedge \cdots \wedge b_n) \quad [h(X_1 \mid t_1,\ X_2 \mid t_2, \cdots, X_s \mid t_s)]$$

para cualquier asignación de variables h. Si definimos la sustitución θ como $\theta = \{X_1/t_1,\ X_2/t_2, \ldots, X_s/t_s\}$, entonces:

$$\models_\Psi (b_1 \wedge b_2 \wedge \ldots \wedge b_n)\theta$$

Más aún, como la elección de Ψ fue arbitraria, entonces $\models_\Psi (b_1 \wedge b_2 \wedge \ldots \wedge b_n)\theta$ para todo Ψ modelo de Herbrand para Γ. Entonces $\models_\Psi \neg(b_1 \wedge b_2 \wedge \ldots \wedge b_n)\theta$ es falso para todo Ψ modelo de Herbrand y se sigue que:

$$\Gamma \cup \{\neg(b_1 \wedge b_2 \wedge \ldots \wedge b_n)\theta\}$$

no tiene modelos de Herbrand. Observemos $\neg(b_1 \wedge b_2 \wedge \ldots \wedge b_n)\theta$ es una cláusula, ya que $\neg(b_1 \wedge b_2 \wedge \ldots \wedge b_n)\theta = \forall(\neg(b_1 \wedge b_2 \wedge \ldots \wedge b_n)\theta)$ por no tener variables. Entonces, por el corolario 5.22

$$\Gamma \cup \{\neg(b_1 \wedge b_2 \wedge \ldots \wedge b_n)\theta\} \text{ es insatisfactible}$$

y por lo tanto

$$\Gamma \models (b_1 \wedge b_2 \wedge \ldots \wedge b_n)\theta \tag{5.8}$$

Las refutaciones SLD como veremos a continuación, nos dan un procedimiento para mostrar que $\Gamma \cup \{G\} \models \square$ y encontrar una sustitución θ que satisfaga (5.8).

Definición 5.27 Respuesta correcta

Sea Γ un programa definido y $G = \longleftarrow b_1, b_2, \ldots, b_n$ una meta. Decimos que θ es una respuesta correcta *para $\Gamma \cup \{G\}$, si θ es una sustitución para las variables en $b_1 \wedge b_2 \wedge \ldots \wedge b_n$ tal que*

$$\Gamma \models \forall((b_1 \wedge b_2 \wedge \ldots \wedge b_n)\theta)$$

y θ no necesariamente contiene ligas para todas las variables en $b_1 \wedge b_2 \wedge \ldots \wedge b_n$ (véanse los ejemplos 5.32 y 5.39).

Definición 5.28 Regla de resolución

Sean $G = \longleftarrow a_1, \ldots, a_m, \ldots, a_k$ una meta y $C = a \longleftarrow b_1, b_2, \ldots, b_n$ una cláusula definida. Decimos que la meta G' se deriva *de G y C usando el mgu θ si se cumple:*

(a) θ es un mgu para a_m y a.

(b) G' es la meta $\longleftarrow (a_1, \ldots, a_{m-1}, b_1, \ldots, b_n, a_{m+1}, \ldots, a_k)\theta$

A a_m se le llama átomo seleccionado *en G y a G' resolvente para G y C.*

Observemos que la regla de resolución es una regla de inferencia de aridad 3, cuyos elementos son de la forma (G, C, G'). Donde G es una meta y G' se deriva de G y una cláusula definida C usando un mgu θ.

Definición 5.29 Derivación SLD

Sea Γ un programa definido y G una meta. Una derivación SLD *para $\Gamma \cup \{G\}$ consiste en:*

(a) Una sucesión de metas $G_0, G_1, \ldots$

(b) Una sucesión de variantes de cláusulas en Γ: $C_1, C_2, \ldots$

(c) Una sucesión de mgus $\theta_1, \theta_2, \ldots$

tales que $G_0 = G$ y cada G_{i+1} se deriva de G_i y C_{i+1} usando θ_{i+1}.

Una derivación SLD puede ser finita o infinita. Observemos que las derivaciones SLD no son otra cosa que deducciones formales en un sistema formal cuya única regla de inferencia es la regla de resolución.

Definición 5.30 Refutación SLD

Una refutación SLD *de n pasos para $\Gamma \cup \{G\}$, es una derivación SLD para $\Gamma \cup \{G\}$ finita y tal que la última meta es $G_n = \square$, la cláusula vacía. A la composición $\theta = \theta_1\theta_2 \cdots \theta_n$ de los unificadores, restringida a las variables de G se le llama* respuesta calculada.

Notemos que una refutación SLD es una deducción formal de la cláusula vacía a partir de $\Gamma \cup \{G\}$ (en el sistema formal que tiene a la regla de resolución como única regla de inferencia).

Ejemplo 5.31

Retomando el programa definido Γ del ejemplo 5.7:

1. $busca(T, P) \longleftarrow compara(T, P)$
2. $busca([X \mid Y], P) \longleftarrow busca(Y, P)$
3. $compara(X, []) \longleftarrow$
4. $compara([X \mid Y], [X \mid Z]) \longleftarrow compara(Y, Z)$

El significado deseado del programa Γ, es que $\Gamma \models busca(T, P)$ si y solo si P es una sublista de T. Proceduralmente la cláusula (2) avanza T, la cláusula (1) empieza la comparación, la cláusula (4) va revisando si coinciden y la cláusula (3) termina la comparación.

Sea G la meta $\longleftarrow busca([c, a, d, e, n, a],\ [a, d])$ entonces:

$G_0 = \longleftarrow busca([c \mid [a, d, e, n, a]],\ [a, d])$

$C_1 = busca([X_1 \mid Y_1],\ P_1) \longleftarrow busca(Y_1, P_1)$ seleccionando (2)

$\theta_1 = \{X_1/c,\ Y_1/[a, d, e, n, a],\ P_1/[a, d]\}$

$G_1 = \longleftarrow busca([a, d, e, n, a],\ [a, d)$

$C_2 = busca(T_2, P_2) \longleftarrow compara(T_2, P_2)$ seleccionando (1)

$\theta_2 = \{T_2/[a, d, e, n, a],\ P_2/[a, d]\}$

$G_2 = \longleftarrow compara([a \mid [d, e, n, a]],\ [a \mid [d]])$

$C_3 = compara([X_3 \mid Y_3],\ [X_3 \mid Z_3]) \longleftarrow compara(Y_3, Z_3)$ seleccionando (4)

$\theta_3 = \{X_3/a,\ Y_3/[d,e,n,a],\ Z_3/[d]\}$

$G_3 = \leftarrow compara([d \mid [e,n,a]],\ [d \mid []])$

$C_4 = compara([X_4 \mid Y_4],\ [X_4 \mid Z_4]) \leftarrow compara(Y_4, Z_4)$ seleccionando (4)

$\theta_4 = \{X_4/d,\ Y_4/[e,n,a],\ Z_4/[]\}$

$G_4 = \leftarrow compara([e,n,a],\ [])$

$C_5 = compara(X_5, []) \leftarrow$ seleccionando (3)

$\theta_5 = \{X_5/[e,n,a]\}$

$G_5 = \leftarrow = \square$

Es una refutación SLD de cinco pasos para $\Gamma \cup \{G\}$. La respuesta calculada es ε ya que G no tiene variables. Notemos que las variables en cada C_i están subindicadas con i, esto se conoce como ajenizar las variables[2]. En el siguiente ejemplo se verá la importancia de hacer esto.

También puede verse, por ejemplo que:

$$busca([c \mid [a,d,e,n,a]],\ [a,d])$$

es el átomo seleccionado en $G_0 = \leftarrow busca([c \mid [a,d,e,n,a]],\ [a,d])$ y unifica con el antecedente de la cláusula:

$$C_1 = busca([X_1 \mid Y_1], P_1) \leftarrow busca(Y_1, P_1)$$

usando $\theta_1 = \{X_1/c, Y_1/[a,d,e,n,a], P_1/[a,d]\}$ como mgu. Así la meta

$$G_1 = \leftarrow busca([a,d,e,n,a],\ [a,d)$$

se deriva de G_0 y C_1 usando θ_1.

Ejemplo 5.32

Con el mismo programa Γ sea $G = \leftarrow busca([a,b,Y],[a \mid X])$, entonces:

$G_0 = \leftarrow busca([a,b,Y],\ [a \mid X])$

$C_1 = busca(T_1, P_1) \leftarrow compara(T_1, P_1)$ seleccionando (1)

$\sigma_1 = \{T_1/[a,b,Y],\ P_1/[a \mid X]\}$

$G_1 = \leftarrow compara([a \mid [b,Y]],\ [a \mid X])$

[2] En inglés, standarizing apart.

$C_2 = compara([X_2 \mid Y_2], [X_2 \mid Z_2]) \longleftarrow compara(Y_2, Z_2)$ seleccionando (4)

$\sigma_2 = \{X_2/a,\ Y_2/[b,Y],\ Z_2/X\}$

$G_2 = \longleftarrow compara([b,Y], X)$

$C_3 = compara(X_3, []) \longleftarrow$ seleccionando (3)

$\sigma_3 = \{X_3/[b,Y],\ X/[]\}$

$G_3 = \longleftarrow = \square$

es una refutación SLD de tres pasos para $\Gamma \cup \{G\}$. De este modo

$\sigma_1\sigma_2\sigma_3 = \{T_1/[a,b,Y],\ P_1/[a \mid []],\ X_2/a,\ Y_2/[b,Y],\ Z_2/[],\ X_3/[b,Y],\ X/[]\}$

y si restringimos esta composición a las variables en G, obtenemos la respuesta calculada

$$\sigma = \{X/[]\}$$

Notemos que de no haber ajenizado las variables habría confusión entre X, X_2 y X_3.

Observación 5.33 *Si tenemos una refutación SLD de n pasos para $\Gamma \cup \{G\}$, tal que:*

$G_0 = G, G_1, \ldots, G_n = \square$ es la sucesión de metas.

$C_1, \ldots, C_n$ son las variantes de la cláusulas en Γ.

$\theta_1, \ldots, \theta_n$ es la sucesión de mgus.

Entonces para todo $i = 1, 2, \ldots, n$, la secuencia de metas $G_i, G_{i+1}, \ldots, G_n$ determina una refutación SLD de $n - i$ pasos para $\Gamma \cup \{G_i\}$ con respuesta calculada $\theta_{i+1}\theta_{i+2}\cdots\theta_n$ restringida a las variables en G_i.

Definición 5.34 Resolución SLD

Definimos la resolución SLD *como el sistema formal que consta de la siguiente regla de inferencia:*

- *La fórmula $\neg G$ es una consecuencia directa de las cláusulas definidas $C_1, C_2, \ldots, C_n$, si G es una meta; $\Gamma = \{C_1, C_2, \ldots, C_n\}$ es un programa definido y existe una refutación SLD para $\Gamma \cup \{G\}$.*

Observése que $\neg G$ no es una cláusula.

Es fácil demostrar a partir de esta definición que si Γ es un programa definido y G es una meta, entonces $\Gamma \vdash_{SLD} \neg G$ si y sólo si existe una refutación SLD para $\Gamma \cup \{G\}$.

5.5 Completud y Validez de la Resolución SLD para Programas Definidos

Ahora enunciamos los resultados de validez y completud de la resolución SLD. Cabe aclarar que la completud es relativa[3] al conjunto de las fórmulas de la forma $\neg G$ donde G es una meta. Sus demostraciones no se incluyen porque requieren material que excede los objetivos de esta tesis.

Proposición 5.35 *Sea Γ un programa definido y G una meta. Entonces $\Gamma \models \neg G$ si y sólo si existe una refutación SLD para $\Gamma \cup \{G\}$.*

Demostración Ver [Llo87, p.p. 43-44,49].

Observación 5.36 *Si no se cumple $\Gamma \models \neg G$, entonces es inmediato que no existe una refutación SLD para $\Gamma \cup \{G\}$. Entonces, para cualquier derivación que empiece en G sucede una de dos cosas:*

(a) A partir de algún paso ya no es posible unificar ninguna submeta con ninguna cláusula del programa, y por lo tanto no se puede extender la derivación.

(b) Siempre es posible extender la derivación pero nunca deducimos la cláusula vacía (obtenemos una derivación de longitud infinita).

Proposición 5.37 *Sea Γ un programa definido y G una meta.*

(a) Toda respuesta calculada σ para $\Gamma \cup \{G\}$, es también una respuesta correcta.

(b) Para toda respuesta correcta θ para $\Gamma \cup \{G\}$, existe una respuesta calculada σ y una sustitución τ tal que $\theta = \sigma\tau$.

Demostración Ver [Llo87, p.p. 43-44,48].

Observación 5.38 *Este teorema nos dice que las respuestas correctas se obtienen asignando variables de una respuesta calculada, de modo que una respuesta calculada es una "forma general" o "molde" de muchas respuestas correctas.*

También observemos que si $\Gamma \models \neg G$ y G es una meta sin variables, entonces ε (la sustitución identidad) es una respuesta correcta y por lo tanto existe una refutación SLD para $\Gamma \cup \{G\}$ con respuesta calculada σ. Pero recordemos que la respuesta calculada se obtiene restringiendo la composición de los mgus a las variables de G. Entonces $\sigma = \varepsilon$.

[3] Ver el último párrafo del capítulo anterior en la página 49.

Ejemplo 5.39

Retomando el ejemplo 5.32, sabemos que $\sigma = \{X/[]\}$ es una respuesta calculada para el programa definido:

1. $busca(T, P) \longleftarrow compara(T, P)$
2. $busca([X \mid Y], P) \longleftarrow busca(Y, P)$
3. $compara(X, []) \longleftarrow$
4. $compara([X \mid Y], [X \mid Z]) \longleftarrow compara(Y, Z)$

y la meta

$$G = \longleftarrow busca([a, b, Y], [a \mid X])$$

entonces, por la proposición 5.37, σ es también una respuesta correcta y se cumple:

$$\Gamma \models \forall(busca([a, b, Y], [a \mid X])\sigma)$$

esto es:

$$\Gamma \models \forall Y\ busca([a, b, Y], [a])$$

recordando que $[a \mid []] = [a]$. Lo cual es intuitivamente obvio, ya que no importa cuál sea el valor que tome Y, $[a]$ siempre es una sublista de $[a, b, Y]$. También podemos observar que σ no es única. De hecho existen otras refutaciones SLD para $\Gamma \cup \{G\}$ que generan las respuestas calculadas:

$\{X/[b]\}$

$\{X/[b, Y]\}$

y junto con σ son las únicas respuestas calculadas. Ahora bien, no toda respuesta correcta es respuesta calculada. Por ejemplo:

$$\theta = \{X/[b],\ Y/c\}$$

es una respuesta correcta pero no existe refutación SLD de $\Gamma \cup \{G\}$ de la cual sea respuesta calculada. Sin embargo, la proposición 5.37 nos garantiza la existencia de la respuesta calculada $\sigma = \{X/[b]\}$ y la sustitución $\tau = \{Y/c\}$, tales que:

$$\theta = \{X/[b],\ Y/c\} = \{X/[b]\}\ \{Y/c\} = \sigma\tau$$

5.6 Refutaciones Izquierdas

Proposición 5.40 Switching lemma

Sea Γ *un programa definido y* $G = \longleftarrow \alpha$ *una meta. Supongamos que* $\Gamma \cup \{G\}$ *tiene una refutación SLD donde:*

$G_0 = G, G_1, \ldots, G_{k-1}, G_k, G_{k+1}, \ldots, G_n = \square$ *es la sucesión de metas.*

$C_1, \ldots, C_n$ *son las variantes de la cláusulas en* Γ.

$\sigma_1, \ldots, \sigma_n$ *es la sucesión de mgus.*

σ *es la respuesta calculada.*

Supongamos que:

$$\begin{array}{lcl} G_{k-1} & = & \longleftarrow b_1, \ldots, b_{i-1}, b_i, \ldots, b_{j-1}, b_j, \ldots, b_n \\ G_k & = & \longleftarrow (b_1, \ldots, b_{i-1}, C_k^-, \ldots, b_{j-1}, b_j, \ldots, b_n)\sigma_k \\ G_{k+1} & = & \longleftarrow (b_1, \ldots, b_{i-1}, C_k^-, \ldots, b_{j-1}, C_{k+1}^-, \ldots, b_n)\sigma_k\sigma_{k+1} \end{array}$$

donde C_k^- *y* C_{k+1}^- *denotan los cuerpos de la cláusulas* C_k *y* C_{k+1} *respectivamente.*

Entonces existe una refutación SLD para $\Gamma \cup \{G\}$ *con respuesta calculada* τ, *tal que:*

(a) El átomo b_j *se selecciona en* G_{k-1} *en lugar de* b_i *y a su vez* b_i *se selecciona en* G_k *en lugar de* b_j.

(b) La respuesta calculada τ, *es tal que* $\alpha\tau$ *y* $\alpha\sigma$ *son variantes.*

Demostración Ver [Llo87, p.p. 50-51].

Definición 5.41 Refutación izquierda

Definimos una refutación izquierda *como una refutación SLD tal que en cada paso se seleccionó el átomo más a la izquierda de cada meta.*

Teorema 5.42 *Sea Γ un programa definido y $G = \longleftarrow \alpha$ una meta. Supongamos que existe una refutación SLD de n pasos para $\Gamma \cup \{G\}$ con respuesta calculada θ, entonces:*

(a) Existe una refutación izquierda de n pasos para $\Gamma \cup \{G\}$ con respuesta calculada τ

(b) La respuesta calculada τ, es tal que $\alpha\tau$ y $\alpha\theta$ son variantes.

Demostración Por inducción sobre el número de pasos de la refutación:

- **Si la refutación se obtuvo en un paso**

  $G_0 = \longleftarrow b_1$
  $G_1 = \square$

  De modo que forzosamente se escogió el átomo más a la izquierda de G_1 y el resultado se cumple en este caso.

- **Supongamos como hipótesis inductiva que la afirmación se cumple para refutaciones con k o menos pasos**

  Supongamos que la refutación fue de $k+1$ pasos y tal que:

  $G_0 = G, G_1, \ldots, G_{k-1}, G_k, G_{k+1} = \square$ es la sucesión de metas.

  $C_1, \ldots, C_{k+1}$ son las variantes de la cláusulas en Γ.

  $\theta_1, \ldots, \theta_{k+1}$ es la sucesión de mgus.

  Si en el primer paso se seleccionó el átomo b_j con $j > 1$, entonces:

  $G_0 = \longleftarrow b_1, b_2, \ldots, b_j, \ldots, b_n$
  $G_1 = \longleftarrow (b_1, b_2, \ldots, C_1^-, \ldots, b_n)\theta_1$

  y el átomo b_1 se seleccionó en la meta G_m para algún $2 \leq m \leq k$. Aplicando $m-1$ veces el lema 5.40 obtenemos una refutación SLD para $\Gamma \cup \{G\}$ con respuesta calculada θ' donde, en el primer paso se seleccionó b_1, es decir:

  $G_0 = \longleftarrow b_1, b_2, \ldots, b_j, \ldots, b_n$
  $G_1' = \longleftarrow (C_m^-, b_2, \ldots, b_j, \ldots, b_n)\theta_1'$
  $G_2' = \longleftarrow (C_m^-, b_2, \ldots, C_1^-, \ldots, b_n)\theta_1'\theta_2'$

  y tal que $\alpha\theta$ y $\alpha\theta'$ son variantes. Por la observación 5.33 también tenemos una refutación SLD para $\Gamma \cup \{G_1'\}$ de k pasos con respuesta

calculada: $\sigma = \theta'_2 \cdots \theta'_{k+1}$ restringida a las variables en G'_1. Aplicando la hipótesis de inducción sabemos que existe una refutación izquierda de k pasos para $\Gamma \cup \{G'_1\}$ con respuesta calculada τ tal que $(C_m^-, b_2, \ldots, b_j, \ldots, b_n)\theta'_1\sigma$ y $(C_m^-, b_2, \ldots, b_j, \ldots, b_n)\theta'_1\tau$ son variantes. Por lo tanto, hemos construido una refutación izquierda de $k+1$ pasos para $\Gamma \cup \{G\}$ con respuesta calculada $\theta'_1\tau$ (restringida a las variables en G). Es fácil mostrar que $\alpha\theta$ y $\alpha\theta'_1\tau$ también son variantes. $\diamond$

Este teorema es un caso particular de un resultado más general llamado independencia de la regla de selección (ver [Llo87, p.p. 50-52]).

6. Analizadores por Cartas como Sistemas Formales para Programas Cadena

En este capítulo demostramos la validez y completud (relativa a la base de Herbrand) de los analizadores por cartas usados como sistemas formales para un subconjunto de los programas lógicos. Nos apoyamos fuertemente en la completud y validez de la resolución SLD para programas lógicos.

De nuevo suponemos que para todas las definiciones y teoremas el lenguaje de primer orden $\mathcal{L}$ está dado.

Definición 6.1 *Si t es un término de un lenguaje, entonces $var(t)$ denota al conjunto de variables que aparecen en t.*

Definición 6.2 Programa cadena

Un programa cadena *es un programa definido tal que todas sus cláusulas son de la forma:*

$p_0(X_0, X_m) \longleftarrow p_1(X_0, X_1), \ldots, p_k(X_{k-1}, X_k), \ldots, p_m(X_{m-1}, X_m)$

y

$p_0(t, t') \longleftarrow$

donde $X_0, X_1, \ldots, X_m$ son variables distintas, $p_0, p_1, \ldots, p_m$ son símbolos de predicado binario no necesariamente distintos y $var(t') \subseteq var(t)$.

Ejemplo 6.3

$av([St, [Y \mid Z]], [[Y \mid St], Z]) \longleftarrow$

$av([St, [Y \mid Z]], [St, Z]) \longleftarrow$

$re([[Y \mid St], Z], [St, [Y \mid Z]]) \longleftarrow$

$sb([St, []], [St, []]) \longleftarrow$

$sb(X_0, X_2) \longleftarrow av(X_0, X_1), sb(X_1, X_2)$

$sb(X_0, X_3) \longleftarrow av(X_0, X_1), sb(X_1, X_2), re(X_2, X_3)$

Lema 6.4 *Si Γ es un programa cadena y*

$$G = \longleftarrow p_1(t_0, t_1), \ldots, p_m(t_{m-1}, t_m)$$

una meta tal que t_0 es un término sin variables, entonces para toda respuesta calculada σ para $\Gamma \cup \{G\}$ se cumple que $t_1\sigma$, $t_2\sigma, \ldots, t_m\sigma$ son términos sin variables.

Demostración Primero probaremos el resultado para refutaciones izquierdas por inducción sobre el número de pasos de la refutación. Sea una refutación izquierda para $\Gamma \cup \{G\}$ con respuesta calculada σ:

Si la refutación se obtuvo en un paso

$G_0 = \longleftarrow p_1(t_0, t_1)$

$G_1 = \square$

de modo que forzosamente se escogió una cláusula en Γ de la forma

$$C_1 = p_1(s, s') \longleftarrow$$

donde $t_1\sigma = s'\sigma$ y $s\sigma = t_0\sigma = t_0$ ya que t_0 es un término sin variables por hipótesis. Como Γ es un programa cadena, tenemos que $var(s') \subseteq var(s)$ entonces $var(s'\sigma) \subseteq var(s\sigma) = var(t_0) = \emptyset$. Por lo tanto $t_1\sigma = s'\sigma$ es un término sin variables.

Ahora supongamos como hipótesis inductiva que la afirmación se cumple para refutaciones con k o menos pasos

Si la refutación fue de $k+1$ pasos, entonces

$$G_0 = G = \longleftarrow p_1(t_0, t_1), \ldots, p_j(t_{j-1}, t_j), \ldots, p_m(t_{m-1}, t_m)$$

y el átomo seleccionado en el primer paso de la refutación fue $p_1(t_0, t_1)$ ya que es el que está más a la izquierda. Entonces, ocurrió alguno de los siguientes casos:

(a) Se escogió una cláusula de la forma

$$C_1 = p_1(s, s') \longleftarrow$$

entonces

$$\begin{aligned} G_1 &= \longleftarrow (p_2(t_1, t_2), \ldots, p_m(t_{m-1}, t_m))\sigma_1 \\ &= \longleftarrow p_2(t_1\sigma_1, t_2\sigma_1), \ldots, p_m(t_{m-1}\sigma_1, t_m\sigma_1) \end{aligned}$$

donde $t_1\sigma_1 = s'\sigma_1$ y $s\sigma_1 = t_0\sigma_1 = t_0$ ya que t_0 es un término sin variables por hipótesis. Pero como Γ es un programa cadena, tenemos que $var(s') \subseteq var(s)$, entonces $var(s'\sigma_1) \subseteq var(s\sigma_1) = var(t_0) = \emptyset$. Por lo tanto:

$$G_1 = \longleftarrow p_2(t_1\sigma_1, t_2\sigma_1), \ldots, p_m(t_{m-1}\sigma_1, t_m\sigma_1)$$

y $t_1\sigma_1 = s'\sigma_1$ es un término sin variables.

(b) Se escogió una cláusula de la forma:

$$C_1 = p_1(Y_0, Y_n) \longleftarrow q_1(Y_0, Y_1), \ldots, q_n(Y_{n-1}, Y_n)$$

entonces

$\sigma_1 = \{Y_0/t_0,\ Y_n/t_1\}$

$G_1 = \longleftarrow (q_1(Y_0, Y_1), \ldots, q_n(Y_{n-1}, Y_n), p_2(t_1, t_2), \ldots, p_m(t_{m-1}, t_m))\sigma_1$

Pero observemos que las variables de las cláusula C_1 están ajenizadas, entonces σ_1 no actúa sobre variables en $t_0, t_1, \ldots, t_m$ de modo que $t_i = t_i\sigma_1$ para todo $i = 0, 1, \ldots m$. Por lo tanto:

$$G_1 = \longleftarrow q_1(t_0, Y_1), \ldots, q_n(Y_{n-1}, t_1), p_2(t_1, t_2), \ldots, p_m(t_{m-1}, t_m)$$

y t_0 es un término sin variables por hipótesis.

Puede verse que en ambos casos G_1 es un cláusula de la forma

$$\longleftarrow b_1(s_0, s_1), \ldots, b_u(s_{u-1}, s_u)$$

donde s_0 es un término sin variables y

$$\{t_i\sigma_1 \mid i = 1, \ldots, m\} \subseteq \{s_j \mid j = 0, \ldots, u\} \tag{6.1}$$

Entonces por la observación 5.33 tenemos una refutación SLD de k pasos para $\Gamma \cup \{G_1\}$ con respuesta calculada $\sigma' = \sigma_2\sigma_3 \cdots \sigma_{k+1}$ restringida a las variables de G_1. Aplicando la hipótesis inductiva tenemos que $s_1\sigma', s_2\sigma', \ldots, s_u\sigma'$ son términos sin variables. Pero recordemos que $\sigma = \sigma_1\sigma'$ restringida a las variables de G_0 por definición. Entonces por (6.1) para todo $i = 1, \ldots, m$ tenemos:

$$t_i\sigma = t_i\sigma_1\sigma_2\sigma_3 \cdots \sigma_{k+1} = (t_i\sigma_1)\sigma' = s_j\sigma' \text{ para algún } j$$

Por lo tanto $t_1\sigma,\ t_2\sigma, \ldots, t_m\sigma$ son términos sin variables.

Ahora probemos el caso general:

Sea una refutación SLD para $\Gamma \cup \{G\}$ con respuesta calculada σ. Por el teorema 5.42 sabemos que existe una refutación izquierda para $\Gamma \cup \{G\}$ con respuesta calculada τ tal que $(p_1(t_0,t_1) \wedge \ldots \wedge p_m(t_{m-1},t_m))\tau$ y $(p_1(t_0,t_1) \wedge \ldots \wedge p_m(t_{m-1},t_m))\sigma$ son variantes. Entonces por la proposición 5.16 difieren sólo en nombres de variables. Pero por lo que acabamos de demostrar $t_1\tau,\ t_2\tau,\ldots,t_m\tau$ son términos sin variables entonces $t_j\tau = t_j\sigma$ para todo $j = 1,2,\ldots,m$ y por lo tanto $t_1\sigma,\ t_2\sigma,\ldots,t_m\sigma$ son términos sin variables.◇

Teorema 6.5 *Sea Γ un programa cadena y $p_0,p_1,\ldots,p_m$ símbolos de predicado binario no necesariamente distintos. Si $\Gamma \models p_1(t_0,t_1) \wedge \ldots \wedge p_m(t_{m-1},t_m)$ y t_0 es un término sin variables, entonces $t_1,\ldots,t_m$ son términos sin variables.*

Demostración Sea $\alpha = p_1(t_0,t_1) \wedge \ldots \wedge p_m(t_{m-1},t_m)$. Sabemos que las cláusulas son un tipo particular de enunciados lógicos y Γ es un conjunto de cláusulas. Por la definición 4.14, $\Gamma \models \alpha$ si y sólo si para cualquier estructura Ψ y cualquier asignación de variables h tal que Ψ sea un modelo de Γ, se tiene que $\models_\Psi \alpha\ [h]$. Por lo tanto, si Ψ es un modelo de Γ :

$$\models_\Psi \alpha\ [h] \text{ para cualquier asignación de variables } h$$

es equivalente a (ver definición 4.12)

$$\models_\Psi \forall(\alpha)$$

Entonces $\Gamma \models \alpha$ si y sólo si $\Gamma \models \forall(\alpha)$ (este resultado puede obtenerse directamente usando el teorema de generalización de la lógica de primer orden[1]). Por lo tanto $\Gamma \models \forall(\alpha)$.

Sea G la meta

$$\begin{aligned} G &= \forall(\neg\alpha) \\ &= \forall(\neg(p_1(t_0,X_1) \wedge \ldots \wedge p_m(X_{m-1},X_m))) \\ &= \longleftarrow p_1(t_0,X_1),\ldots,p_m(X_{m-1},X_m) \end{aligned}$$

y

$$\theta = \{X_1/t_1,\ X_2/t_2,\ldots,X_m/t_m\}.$$

Como $\Gamma \models \forall(\alpha)$ entonces $\Gamma \models \forall((p_1(t_0,X_1) \wedge \ldots \wedge p_m(X_{m-1},X_m))\theta)$. De modo que θ es una respuesta correcta para $\Gamma \cup \{G\}$. Por la proposición 5.37 sabemos que para toda respuesta correcta θ existe una refutación SLD para

[1] Ver [End87, p.p. 157].

$\Gamma \cup \{G\}$ con respuesta calculada σ y una sustitución de variables τ, tales que $\theta = \sigma\tau$. Pero por el lema 6.4 sabemos que $X_1\sigma,\ X_2\sigma, \ldots, X_m\sigma$ son términos sin variables, entonces $X_i\sigma\tau = X_i\sigma$ para todo $i = 1, \ldots, m$. Por lo tanto $t_i = X_i\theta = X_i\sigma\tau = X_i\sigma$ es un término sin variables para todo $i = 1, \ldots, m$. ◇

Ahora construimos una versión modificada de un analizador por cartas que servirá como sistema formal para programas cadena.

Sea un programa cadena Γ. Comenzamos definiendo la carta $C(\Gamma)$ cuyo conjunto de estados es $U_{\mathcal{L}}$, el universo de Herbrand del lenguaje (el conjunto de términos sin variables) y establecemos las siguientes reglas:

(i) Si agregamos a la carta una arista del tipo $\langle t,\ t',\ q_0 \longleftarrow q_k \cdots q_m\rangle$, (donde t y t' podrían ser iguales) entonces para toda cláusula de la forma:

$$q_k(Y_0, Y_s) \longleftarrow r_1(Y_0, Y_1), \ldots, r_s(Y_{s-1}, Y_s)$$

en Γ, agregamos la arista:

$$\langle t',\ t',\ q_k \longleftarrow r_1 \cdots r_s\rangle$$

(ii) Si la carta contiene una arista del tipo $\langle t,\ t',\ q_0 \longleftarrow q_k \cdots q_m\rangle$ (donde t y t' podrían ser iguales), y $q_k(s, s') \longleftarrow$ es una cláusula en Γ y existe un mgu θ tal que $s\theta = t'\theta = t'$ (ya que t' es un término sin variables) entonces agregamos la arista:

$$\langle t', s'\theta, q_k \longleftarrow\rangle$$

Notando que como Γ es un programa cadena, tenemos que $var(s') \subseteq var(s)$, entonces $var(s'\theta) \subseteq var(s\theta) = var(t') = \emptyset$ Por lo tanto, $s'\theta$ es un término sin variables.

(RF) Si la carta contiene aristas $\langle t_0,\ t_1,\ q_0 \longleftarrow q_k q_{k+1} \cdots q_m\rangle$ (donde t_0 y t_1 podrían ser iguales) y $\langle t_1,\ t_2, q_k \longleftarrow\rangle$ entonces agregamos la arista:

$$\langle t_0,\ t_2,\ q_0 \longleftarrow q_{k+1} \cdots q_m\rangle$$

Consideramos a las aristas del tipo $\langle t,\ t',\ q_0 \longleftarrow q_k \cdots q_m\rangle$ como activas y a las del tipo $\langle t, t', q_k \longleftarrow\rangle$ como pasivas.

Con todo lo anterior, pretendemos que las aristas representen cláusulas como sigue:

$\langle t, t', q \longleftarrow b_r \cdots b_{r+k}\rangle$ con $q \neq S$ representa la cláusula

$$q(t, X_{r+k}) \longleftarrow b_r(t', X_r), \ldots, b_{r+k}(X_{r+k-1}, X_{r+k})$$

$\langle t, t', q \longleftarrow \rangle$ con $q \neq S$ representa la cláusula

$$q(t, t') \longleftarrow$$

$\langle t, t', S \longleftarrow b_r \cdots b_{r+k} \rangle$ representa la meta

$$\longleftarrow b_r(t', X_r), \ldots, b_{r+k}(X_{r+k-1}, X_{r+k})$$

$\langle t, t', S \longleftarrow \rangle$ representa la cláusula vacía $\square$.

El símbolo especial S no aparece en ninguna cláusula de Γ, y se agrega únicamente para no enunciar las reglas (i), (ii), y (RF) en dos versiones. Esto no representa problema ya que las aristas *per se* no tienen significado adscrito y podemos interpretarlas como queramos (en este caso como cláusulas o metas de un programa cadena).

Lema 6.6 *Sea Γ un programa cadena. Sea*

$$G = \longleftarrow p_1(t_0, t_1), \ldots, p_m(t_{m-1}, t_m)$$

una meta tal que t_0 es un término sin variables. Supongamos que existe una refutación izquierda para $\Gamma \cup \{G\}$ con respuesta calculada σ y que $C(\Gamma)$ contiene una arista del tipo:

$$\langle t, t_0, q \longleftarrow p_1 \cdots p_m \rangle$$

entonces después de un número finito de aplicaciones de las reglas (i), (ii) y (RF), la carta $C(\Gamma)$ contendrá las aristas pasivas[2]*:*

$\langle t_0, t_1\sigma, p_1 \longleftarrow \rangle$

$\langle t_1\sigma, t_2\sigma, p_2 \longleftarrow \rangle$

$\vdots$

$\langle t_{m-1}\sigma, t_m\sigma, p_m \longleftarrow \rangle$

[2] De acuerdo con la observación 3.2 esto se enuncia formalmente como:

[...] , y que $C_k(\Gamma)$ contiene una arista del tipo $\langle t, t_0, q \longleftarrow p_1 \cdots p_m \rangle$, entonces existe n en los naturales, tal que $C_{k+n}(\Gamma)$ contiene a las aristas pasivas:

$\langle t_0, t_1\sigma, p_1 \longleftarrow \rangle$
$\langle t_1\sigma, t_2\sigma, p_2 \longleftarrow \rangle$
$\vdots$
$\langle t_{m-1}\sigma, t_m\sigma, p_m \longleftarrow \rangle$

Demostración Observemos que como t_0 es un término sin variables, el lema 6.4 nos garantiza que $t_1\sigma, \ldots, t_m\sigma$ son términos sin variables. De modo que tiene sentido considerar a estos términos como posibles estados de $C(\Gamma)$.

Supongamos entonces que $C(\Gamma)$ contiene una arista

$$\langle t, t_0, q \longleftarrow p_1 \cdots p_m \rangle$$

y que existe una refutación izquierda para $\Gamma \cup \{G\}$ con respuesta calculada σ. Sean:

n el número de pasos de la refutación

$G_0, G_1, \ldots, G_n$ la sucesión de metas

$C_1, C_1, \ldots, C_n$ la sucesión de variantes de cláusulas en Γ

$\sigma_1, \sigma_2, \ldots, \sigma_n$ la sucesión de mgus usados

Definamos:

$\theta_j = \sigma_1 \cdots \sigma_j$ para todo $j = 1, \ldots, n$ y $\theta_0 = \varepsilon$

$\tau_j = \sigma_{j+1} \cdots \sigma_n$ para todo $j = 0, \ldots, n-1$ y $\tau_n = \varepsilon$

de este modo $\theta_j \tau_j = \sigma_1 \cdots \sigma_n$ para todo $j = 0, 1, \ldots, n$.

Notemos que por la observación 5.33, para cada meta G_i con $i = 0, 1, \ldots, n$, tenemos una refutación SLD para $\Gamma \cup \{G_i\}$ de $n - i$ pasos cuya respuesta calculada es $\tau_i = \sigma_{i+1} \cdots \sigma_n$ restringida a las variables en G_i. En particular $\sigma = \tau_0 = \sigma_1 \cdots \sigma_n$ restringida a las variables en G.

Vamos ahora a demostrar que después de un número finito de aplicaciones de las reglas (i), (ii) y (RF), la carta $C(\Gamma)$ contiene las aristas pasivas:

$$\langle t_0, t_1\sigma, p_1 \longleftarrow \rangle$$
$$\langle t_1\sigma, t_2\sigma, p_2 \longleftarrow \rangle$$
$$\vdots$$
$$\langle t_{m-1}\sigma, t_m\sigma, p_m \longleftarrow \rangle$$

La demostración será por inducción sobre n, el número de pasos de la refutación.

Si la refutación se obtuvo en un paso

Entonces:

$G_0 = \longleftarrow p_1(t_0, t_1)$

$G_1 = \square$

de modo que forzosamente se escogió una cláusula en Γ de la forma:

$$C_1 = p_1(s, s') \longleftarrow$$

y σ fue tal que $s'\sigma = t_1\sigma$ y $s\sigma = t_0\sigma = t_0$ ya que t_0 es un término sin variables por hipótesis. También por hipótesis $C(\Gamma)$ contiene una arista activa del tipo:

$$\langle t, t_0, q \longleftarrow p_1 \rangle$$

Aplicando la regla (ii) agregamos la arista:

$$\langle t_0, s'\sigma, p_1 \longleftarrow \rangle$$

que es lo mismo que:

$$\langle t_0, t_1\sigma, p_1 \longleftarrow \rangle$$

Por lo tanto la afirmación se cumple cuando $n = 1$.

Ahora supongamos como hipótesis inductiva que la afirmación se cumple cuando la refutación tiene k o menos pasos

Supongamos que $n = k + 1$. Entonces:

$$G_0 = \longleftarrow p_1(t_0, t_1), \ldots, p_m(t_{m-1}, t_m)$$

y el átomo seleccionado en el primer paso de la refutación fue $p_1(t_0, t_1)$ ya que es el que aparece más a la izquierda. Entonces, ocurrió alguno de los siguientes casos:

caso (a) Se escogió una cláusula de la forma

$$C_1 = p_1(s, s') \longleftarrow$$

entonces

$$\begin{aligned} G_1 &= \longleftarrow (p_2(s', t_2), \ldots, p_m(t_{m-1}, t_m))\sigma_1 \\ &= \longleftarrow p_2(s'\sigma_1, t_2\sigma_1), \ldots, p_m(t_{m-1}\sigma_1, t_m\sigma_1) \end{aligned}$$

donde $t_1\sigma_1 = s'\sigma_1$ y $s\sigma_1 = t_0\sigma_1 = t_0$ ya que t_0 es un término sin variables por hipótesis. Pero como Γ es un programa cadena, tenemos que $var(s') \subseteq var(s)$ entonces $var(s'\sigma_1) \subseteq var(s\sigma_1) = var(t_0) = \emptyset$. Por lo tanto:

$$G_1 = \longleftarrow p_2(t_1\sigma_1, t_2\sigma_1), \ldots, p_m(t_{m-1}\sigma_1, t_m\sigma_1)$$

donde $t_1\sigma_1 = s'\sigma_1$ es un término sin variables. Por hipótesis $C(\Gamma)$ contiene una arista activa del tipo:

$$\langle t, t_0, q \longleftarrow p_1 p_2 \cdots p_m \rangle \tag{6.2}$$

Aplicando la regla (ii) a agregamos la arista:

$$\langle t_0, s'\sigma_1, p_1 \longleftarrow \rangle$$

que es lo mismo que:

$$\langle t_0, t_1\sigma_1, p_1 \longleftarrow \rangle \tag{6.3}$$

Aplicando (RF) a las aristas (6.2) y (6.3) obtenemos la arista

$$\langle t, t_1\sigma_1, q \longleftarrow p_2 \cdots p_m \rangle \tag{6.4}$$

Pero por la observación 5.33 tenemos una refutación izquierda para $\Gamma \cup \{G_1\}$ de k pasos con respuesta calculada τ_1 (restringida a las variables de G_1). Aplicando la hipótesis inductiva sabemos que después de un número finito de aplicaciones de las reglas (i), (ii) y (RF), la carta $C(\Gamma)$ contiene las aristas pasivas:

$$\langle t_1\sigma_1,\ t_2\sigma_1\tau_1, p_2 \longleftarrow \rangle$$
$$\langle t_2\sigma_1\tau_1,\ t_3\sigma_1\tau_1, p_3 \longleftarrow \rangle$$
$$\vdots$$
$$\langle t_{m-1}\sigma_1\tau_1,\ t_m\sigma_1\tau_1, p_m \longleftarrow \rangle$$

que son las mismas que:

$$\langle t_1\sigma,\ t_2\sigma,\ p_2 \longleftarrow \rangle$$
$$\langle t_2\sigma,\ t_3\sigma,\ p_3 \longleftarrow \rangle$$

$$\vdots$$

$$\langle t_{m-1}\sigma,\ t_m\sigma, p_m \longleftarrow \rangle$$

ya que $\sigma_1\tau_1 = \sigma_1\sigma_2\cdots\sigma_n$ y como $t_1\sigma_1$ es un término sin variables entonces $t_1\sigma_1 = t_1\sigma_1\sigma_2\cdots\sigma_n = t_1\sigma$. Por esta misma razón la arista (6.3) es en realidad:

$$\langle t_0, t_1\sigma, p_1 \longleftarrow \rangle$$

Por lo tanto la afirmación se cumple en este caso.

caso (b) Se escogió una cláusula de la forma

$$C_1 = p_1(Y_0, Y_s) \longleftarrow q_1(Y_0, Y_1), \ldots, q_s(Y_{s-1}, Y_s)$$

entonces,

$$\begin{array}{lcl} \sigma_1 & = & \{Y_0/t_0, Y_s/t_1\} \\ G_1 & = & \longleftarrow (q_1(Y_0, Y_1), \ldots, q_s(Y_{s-1}, Y_s), p_2(t_1, t_2), \ldots, p_m(t_{m-1}, t_m))\sigma_1 \\ & = & \longleftarrow q_1(t_0, Y_1), \ldots, q_s(Y_{s-1}, t_1), p_2(t_1, t_2), \ldots, p_m(t_{m-1}, t_m) \end{array}$$

ya que las variables de las cláusula C_1 están ajenizadas y por lo tanto σ_1 no actúa sobre variables en $t_0, t_1, \ldots, t_m$ de modo que $t_i = t_i\sigma_1$ para todo $i = 0, 1, \ldots m$. (♣)

Por hipótesis $C(\Gamma)$ contiene una arista activa del tipo:

$$\langle t, t_0, q \longleftarrow p_1 p_2 \cdots p_m \rangle \tag{6.5}$$

Aplicando la regla (i) a (6.5) agregamos la arista

$$\langle t_0, t_0, p_1 \longleftarrow q_1 q_2 \cdots q_s \rangle \tag{6.6}$$

Como supusimos que para cada G_i seleccionábamos el átomo de más a la izquierda, forzosamente se refutaron las submetas con símbolos de predicado $q_1, q_2, \cdots, q_s$ en G_1. De modo que para algún $j < n$ en el paso j de la refutación se obtuvo la meta:

$$G_j = \longleftarrow p_2(t_1\theta_j, t_2\theta_j), \ldots, p_m(t_{m-1}\theta_j, t_m\theta_j)$$

recordando que $\theta_j = \sigma_1 \cdots \sigma_j$ para todo $j = 1, \ldots, n$. Ahora notemos que la secuencia de metas que aparece entre G_1 y G_j determina una refutación izquierda "implícita" para $\Gamma \cup \{\longleftarrow q_1(t_0, Y_1), \ldots, q_s(Y_{s-1}, t_1)\}$ de $j-1$ pasos y con respuesta calculada $\sigma_2 \cdots \sigma_j$ (restringida a las variables en $\beta =$

$\{t_0, Y_1, \ldots, Y_{s-1}, t_1\}$). Pero por lo que observamos en el párrafo (♣) $\sigma_1 = \{Y_0/t_0, Y_s/t_1\}$ no contiene ligas para variables en β. Entonces $\beta\sigma_1 = \beta$ y por lo tanto

$$\beta\sigma_2 \cdots \sigma_j = \beta\sigma_1\sigma_2 \cdots \sigma_j = \beta\theta_j.$$

También notemos que $j - 1 < n - 1 = k$. Entonces existe una refutación izquierda para $\Gamma \cup \{\longrightarrow q_1(t_0, Y_1), \ldots, q_s(Y_{s-1}, t_1)\}$ con menos de k pasos y con respuesta calculada θ_j (restringida a las variables en β). Como la arista (6.6) está en la carta podemos aplicar la hipótesis de inducción y afirmar que después de un número finito de aplicaciones de las reglas (i), (ii) y (RF), la carta $C(\Gamma)$ contiene aristas pasivas:

$$\langle t_0,\ Y_1\theta_j, q_1 \longrightarrow\rangle$$
$$\langle Y_1\theta_j,\ Y_2\theta_j, q_2 \longrightarrow\rangle$$
$$\vdots$$
$$\langle Y_{s-2}\theta_j,\ Y_{s-1}\theta_j, q_{s-1} \longrightarrow\rangle$$
$$\langle Y_{s-1}\theta_j,\ t_1\theta_j, q_s \longrightarrow\rangle$$

donde $t_0, Y_1\theta_j, \ldots, Y_{s-1}\theta_j,\ t_1\theta_j$ son términos sin variables (por el lema 6.4).

Si ahora aplicamos $s-1$ veces la regla (RF) a estas aristas[3] y (6.6) obtenemos las aristas

$$\langle t_0,\ Y_1\theta_j, p_1 \longrightarrow q_2 q_3 \ldots q_s\rangle \tag{6.7}$$

$$\langle t_0,\ Y_2\theta_j, p_1 \longrightarrow q_3 \ldots q_s\rangle \tag{6.8}$$
$$\vdots$$
$$\langle t_0,\ Y_{s-1}\theta_j, p_1 \longrightarrow q_s\rangle$$

$$\langle t_0, t_1\theta_j, p_1 \longrightarrow\rangle \tag{6.9}$$

y aplicando (RF) a (6.9) y (6.5) obtenemos

$$\langle t, t_1\theta_j, q \longrightarrow p_2 \cdots p_m\rangle \tag{6.10}$$

[3] En realidad aplicamos (RF) a la primera arista y a (6.6) y obtenemos la arista (6.7). Luego aplicamos (RF) a la segunda arista y a (6.7) y obtenemos (6.8) y así sucesivamente.

Nuevamente por la observación 5.33 tenemos una refutación izquierda para $\Gamma \cup \{G_j\}$ de $n-j = k+1-j \leq k$ pasos con respuesta calculada τ_j (restringida a las variables de G_j). Como la arista (6.10) está en la carta podemos aplicar la hipótesis de inducción y afirmar que después de un número finito de aplicaciones de las reglas (i), (ii) y (RF), la carta $C(\Gamma)$ contiene aristas pasivas:

$$\langle t_1\theta_j, t_2\theta_j\tau_j, p_2 \longleftarrow \rangle$$
$$\langle t_2\theta_j\tau_j,\ t_3\theta_j\tau_j, p_3 \longleftarrow \rangle$$
$$\vdots$$
$$\langle t_{m-1}\theta_j\tau_j,\ t_m\theta_j\tau_j, p_m \longleftarrow \rangle$$

que son en realidad (recordemos que $\sigma = \sigma_1 \cdots \sigma_n$ restringida a las variables en G):

$$\langle t_1\sigma,\ t_2\sigma,\ p_2 \longleftarrow \rangle$$
$$\langle t_2\sigma,\ t_3\sigma,\ p_3 \longleftarrow \rangle$$
$$\vdots$$
$$\langle t_{m-1}\sigma,\ t_m\sigma, p_m \longleftarrow \rangle$$

ya que $\theta_j\tau_j = \sigma_1 \cdots \sigma_n$ para todo $j = 0, 1, \ldots, n$. Como $t_1\theta_j$ es un término sin variables entonces $t_1\theta_j = t_1\theta_j\tau_j = t_1\sigma$. Por esta misma razón la arista (6.9) es en realidad $\langle t_0, t_1\sigma, p_1 \longleftarrow \rangle$.

Por lo tanto la afirmación también se cumple en este caso y el teorema queda demostrado. ◇

Ahora presentamos el teorema de completud (relativa a la base de Herbrand) de los analizadores por cartas como sistemas formales para programas cadena.

Teorema 6.7 *Sea Γ un programa cadena y $p_0, p_1, \ldots, p_m$ símbolos de predicado binario no necesariamente distintos. Si*

$$\Gamma \models p_1(t_0, t_1) \wedge \ldots \wedge p_m(t_{m-1}, t_m)$$

y t_0 es un término sin variables, entonces existe un analizador por cartas tal que después de un número finito de aplicaciones de las reglas (i),(ii) y (RF) la carta $C(\Gamma)$ contiene las aristas:

$\langle t_0, t_1, p_1 \longleftarrow \rangle$

$\langle t_1, t_2, p_2 \longleftarrow \rangle$

$\vdots$

$\langle t_{m-1}, t_m, p_m \longleftarrow \rangle$

Demostración Sean $\alpha = p_1(t_0, X_1) \wedge \ldots \wedge p_m(X_{m-1}, X_m)$ y

$$G = \longleftarrow \alpha = \longleftarrow p_1(t_0, X_1), \ldots, p_m(X_{m-1}, X_m)$$

Suponemos que el símbolo especial S, no aparece en ninguna cláusula de Γ. Definamos el analizador por cartas $A(\Gamma, G)$ que consta de las reglas (i),(ii) y (RF) junto con la carta $C(\Gamma)$ inicializada con la arista:

$$\langle t_0, t_0, S \longleftarrow p_1 \cdots p_m \rangle$$

Por el Teorema 6.5 sabemos que $t_1, \ldots, t_m$ son términos sin variables. Sea:

$$\theta = \{X_1/t_1, X_2/t_2, \ldots, X_m/t_m\}$$

entonces por hipótesis:

$$\Gamma \models (p_1(t_0, X_1) \wedge \ldots \wedge p_m(X_{m-1}, X_m))\theta$$

Como los términos $t_0, t_1, \ldots, t_m$ no tienen variables, esto es equivalente a:

$$\Gamma \models \forall((p_1(t_0, X_1) \wedge \ldots \wedge p_m(X_{m-1}, X_m))\theta)$$

Por lo tanto θ es una respuesta correcta para $\Gamma \cup \{G\}$. Por la proposición 5.37 sabemos que para toda respuesta correcta θ existe una refutación SLD para $\Gamma \cup \{G\}$ con respuesta calculada σ y una sustitución de variables τ, tales que $\theta = \sigma\tau$. Pero por el lema 6.4 sabemos que $X_1\sigma, X_2\sigma, \ldots, X_m\sigma$ son términos sin variables, entonces $X_i\sigma\tau = X_i\sigma$ para todo $i = 1, \ldots, m$. Por lo tanto $t_i = X_i\theta = X_i\sigma\tau = X_i\sigma$ para todo $i = 1, \ldots, m$. Como σ contiene ligas únicamente para variables en G (por ser una respuesta calculada) entonces $\sigma = \theta$. Por lo tanto θ es una respuesta calculada.

Por el teorema 5.42 existe una refutación izquierda para $\Gamma \cup \{G\}$ con respuesta calculada θ' tal que $\alpha\theta$ y $\alpha\theta'$ son variantes, i.e., difieren sólo en nombres de variables. (recordemos que $G = \longleftarrow \alpha$).

Pero $X_i\theta = t_i$ es un término sin variables para todo $i = 1, \ldots, m$. Entonces $\alpha\theta = p_1(t_0, t_1) \wedge \ldots \wedge p_m(t_{m-1}, t_m)$ es una fórmula sin variables. Entonces $\alpha\theta = \alpha\theta'$ y por lo tanto $\theta = \theta'$ ya que ambas actúan exclusivamente sobre las variables de α por ser respuestas calculadas.

De este modo tenemos una refutación izquierda para $\Gamma \cup \{G\}$ con respuesta calculada θ. Por otro lado al inicializar $C(\Gamma)$ agregamos la arista:

$$\langle t_0, t_0, S \longleftarrow p_1 \cdots p_m \rangle$$

Entonces se satisfacen la hipótesis del Lema 6.6 y por lo tanto después de un número finito de aplicaciones de las reglas (i), (ii) y (RF), la carta $C(\Gamma)$ contiene las aristas pasivas:

$$\langle t_0, X_1\theta, p_1 \longleftarrow \rangle$$
$$\langle X_1\theta, X_2\theta, p_2 \longleftarrow \rangle$$
$$\vdots$$
$$\langle X_{m-1}\theta, X_m\theta, p_m \longleftarrow \rangle$$

Como por definición $\theta = \{X_1/t_1,\ X_2/t_2, \ldots, X_m/t_m\}$ entonces estas aristas son en realidad:

$$\langle t_0, t_1, p_1 \longleftarrow \rangle$$
$$\langle t_1, t_2, p_2 \longleftarrow \rangle$$
$$\vdots$$
$$\langle t_{m-1}, t_m, p_m \longleftarrow \rangle$$ ◇

Observación 6.8 *Es importante recalcar que cuando usamos un analizador por cartas para reconocer una cuerda de una gramática libre de contexto, llega un momento en que ya no es posible aplicar las reglas y agregar aristas nuevas. En tal caso el algoritmo termina. Pero para un programa cadena el analizador podría generar infinidad de aristas y nunca acabar.*

Por ejemplo, considérese el programa Λ:

(a) $suc(X, sX) \longleftarrow$

(b) $mq(X,Y) \longleftarrow suc(X,Y)$

(c) $mq(X,Y) \longleftarrow suc(X,Z), mq(Z,Y)$

donde $suc(X,Y)$ representa la relación Y es el sucesor de X y $mq(X,Y)$ representa la relación X es menor que Y. Entonces:

$\Lambda \models mq(0, s0)$

$\Lambda \models mq(0, ss0)$

$\Lambda \models mq(0, sss0)$

$\Lambda \models mq(0, ssss0)$

$\Lambda \models mq(0, sssss0)$

$\vdots$

Por lo tanto la meta

$$G = \longleftarrow mq(0, X)$$

tiene infinidad de respuestas calculadas:

$\sigma_1 = \{X/s0\}$

$\sigma_2 = \{X/ss0\}$

$\sigma_3 = \{X/sss0\}$

$\sigma_4 = \{X/ssss0\}$

$\vdots$

Si al inicializar $C(\Lambda)$ agregamos la arista

$$\langle 0, 0, S \longleftarrow mq \rangle \tag{6.11}$$

por el teorema 6.7 después de un número finito de aplicaciones de las reglas la carta contendrá las aristas:

$\langle 0, s0, mq \longleftarrow \rangle$

$\langle 0, ss0, mq \longleftarrow \rangle$

$\langle 0, sss0, mq \longleftarrow \rangle$

$\langle 0, ssss0, mq \longleftarrow \rangle$

$\vdots$

Y esto sucederá de la siguiente manera:

Aplicando la regla (i) a (6.11) y (b) obtenemos

$$\langle 0, 0, mq \longleftarrow suc \rangle \tag{6.12}$$

aplicando la regla (ii) a (6.12) y (a) obtenemos

$$\langle 0, s0, suc \longleftarrow \rangle \tag{6.13}$$

aplicando (RF) a (6.12) y (6.13) agregamos

$$\langle 0, s0, mq \longleftarrow \rangle \tag{6.14}$$

Por otro lado aplicando la regla (i) a (6.11) y (c) se tiene

$$\langle 0, 0, mq \longleftarrow suc, mq \rangle \tag{6.15}$$

aplicando (RF) a (6.13) y (6.15) obtenemos

$$\langle 0, s0, mq \longleftarrow mq \rangle \tag{6.16}$$

aplicando la regla (i) a (6.16) y (b) obtenemos

$$\langle s0, s0, mq \longleftarrow suc \rangle \tag{6.17}$$

aplicando la regla (ii) a (6.17) y (a) obtenemos

$$\langle s0, ss0, suc \longleftarrow \rangle \tag{6.18}$$

aplicando (RF) a (6.17) y (6.18) agregamos

$$\langle s0, ss0, mq \longleftarrow \rangle \tag{6.19}$$

por último aplicando (RF) a (6.16) y (6.19) generamos

$$\langle 0, ss0, mq \longleftarrow \rangle$$

y de manera análoga agregamos:

$\langle 0, sss0, mq \longleftarrow \rangle$

$\langle 0, ssss0, mq \longleftarrow \rangle$

$\langle 0, sssss0, mq \longleftarrow \rangle$

etc.

Ahora mostramos la validez de los analizadores por cartas como sistemas formales para programas cadena.

Teorema 6.9 *Sea* Γ *un programa cadena y*

$$G = \longleftarrow p_1(t_0, X_1), \ldots, p_m(X_{m-1}, X_m)$$

una meta tal que t_0 *es un término sin variables. Entonces el analizador por cartas* $A(\Gamma, G)$ *definido en el teorema 6.7 es tal que todas las aristas en* $C(\Gamma)$ *representan consecuencias lógicas de* $\Gamma \cup \{G\}$.

Demostración Recordemos que $\langle t, t', q \longleftarrow b_r \cdots b_{r+k} \rangle$ con $q \neq S$ representa la cláusula:

$$q(t, X_{r+k}) \longleftarrow b_r(t', X_r), \ldots, b_{r+k}(X_{r+k-1}, X_{r+k})$$

$\langle t, t', q \longleftarrow \rangle$ con $q \neq S$ representa la cláusula:

$$q(t, t') \longleftarrow$$

$\langle t, t', S \longleftarrow b_r \cdots b_{r+k} \rangle$ representa la meta:

$$\longleftarrow b_r(t', X_r), \ldots, b_{r+k}(X_{r+k-1}, X_{r+k})$$

$\langle t, t', S \longleftarrow \rangle$ representa la cláusula vacía $\square$.

S es un símbolo especial que no aparece en ninguna cláusula de Γ.

Después de la inicialización la carta $C(\Gamma)$, únicamente contiene la arista:

$$\langle t_0, t_0, S \longleftarrow p_1 \cdots p_m \rangle$$

pero $\Gamma \cup \{G\} \models G$ trivialmente y $G = \longleftarrow p_1(t_0, X_1), \ldots, p_m(X_{m-1}, X_m)$.

Por lo tanto la afirmación se cumple para la arista inicial.

Como la única manera de agregar nuevas aristas a $C(\Gamma)$ es a través de las reglas (i), (ii) y (RF), entonces basta mostrar que la aplicación de las reglas preserva la propiedad de ser consecuencia lógica y concluir por inducción sobre la obtención de aristas[4].

[4] De acuerdo con la observación 3.2 ésto se enuncia formalmente como:

(i) Las aristas en $C_0(\Gamma)$ son consecuencias lógicas de $\Gamma \cup \{G\}$.
(ii) Si las aristas en $C_n(\Gamma)$ son consecuencias lógicas de $\Gamma \cup \{G\}$, entonces las aristas en $C_{n+1}(\Gamma)$ también.

Regla (i) Basta observar que si:

$q_k(Y_0, Y_s) \longleftarrow r_1(Y_0, Y_1), \ldots, r_s(Y_{s-1}, Y_s) \in \Gamma$

entonces, $\Gamma \cup \{G\} \models q_k(Y_0, Y_s) \longleftarrow r_1(Y_0, Y_1), \ldots, r_s(Y_{s-1}, Y_s)$

entonces, $\Gamma \cup \{G\} \models \forall(q_k(Y_0, Y_s) \longleftarrow r_1(Y_0, Y_1) \wedge \ldots \wedge r_s(Y_{s-1}, Y_s))$

Si t' es un término sin variables

entonces, $\Gamma \cup \{G\} \models \forall(q_k(t', Y_s) \longleftarrow r_1(t', Y_1) \wedge \ldots \wedge r_s(Y_{s-1}, Y_s))$.

Por lo tanto la arista $\langle t', t',\ q_k \longleftarrow r_1 \cdots r_s\rangle$ representa una consecuencia lógica de $\Gamma \cup \{G\}$.

Regla (ii) Basta observar que si $q_k(s, s') \longleftarrow$ es una cláusula en Γ entonces

$$\Gamma \cup \{G\} \models \forall(q_k(s, s'))$$

si t' es un término sin variables y existe un mgu θ tal que $s\theta = t'$. Entonces como Γ es un programa cadena, tenemos que $var(s') \subseteq var(s)$ entonces $var(s'\theta) \subseteq var(s\theta) = var(t') = \emptyset$ y $s'\theta$ es un término sin variables. Entonces:

$\Gamma \cup \{G\} \models (q_k(s, s'))\theta$

entonces,

$\Gamma \cup \{G\} \models q_k(t', s'\theta)$,

entonces como t' y $s'\theta$ son términos sin variables:

$\Gamma \cup \{G\} \models \forall(q_k(t', s'\theta))$

Por lo tanto la arista $\langle t', s'\theta, q_k \longleftarrow\rangle$ representa una consecuencia lógica de $\Gamma \cup \{G\}$.

Regla (RF) Supongamos que las aristas

$$\langle t_0,\ t_1,\ q_0 \longleftarrow q_k q_{k+1} \cdots q_m\rangle \text{ y } \langle t_1,\ t_2, q_k \longleftarrow\rangle$$

representan consecuencias lógicas de $\Gamma \cup \{G\}$ y t_0, t_1 y t_2 son términos sin variables, entonces

$\Gamma \cup \{G\} \models q_0(t_0, X_m) \longleftarrow q_k(t_1, X_k), q_{k+1}(X_k, X_{k+1}), \ldots, q_m(X_{m-1}, X_m)$

$\Gamma \cup \{G\} \models q_k(t_1, t_2) \longleftarrow$

pero la primera implicación es en realidad:

$\Gamma \cup \{G\} \models$

$\forall(q_0(t_0, X_m) \longleftarrow q_k(t_1, X_k) \wedge q_{k+1}(X_k, X_{k+1}) \wedge \cdots \wedge q_m(X_{m-1}, X_m))$.

Como t_0, t_1 y t_2 son términos sin variables, entonces

$\Gamma \cup \{G\} \models$

$\forall(q_0(t_0, X_m) \longleftarrow q_k(t_1, t_2) \wedge q_{k+1}(t_2, X_{k+1}) \wedge \cdots \wedge q_m(X_{m-1}, X_m))$.

Como $\Gamma \cup \{G\} \models q_k(t_1, t_2) \longleftarrow$ se sigue que

$\Gamma \cup \{G\} \models \forall(q_0(t_0, X_m) \longleftarrow q_{k+1}(t_2, X_{k+1}) \wedge \cdots \wedge q_m(X_{m-1}, X_m))$.

Por lo tanto la arista $\langle t_0,\ t_2,\ q_0 \longleftarrow q_{k+1} \cdots q_m \rangle$ representa una consecuencia lógica de $\Gamma \cup \{G\}$.

$\diamond$

Ahora podemos enunciar el teorema de validez y completud (relativa a la base de Herbrand) de los analizadores por cartas como sistemas formales para programas cadena.

Teorema 6.10 *Si Γ es un programa cadena y*

$$G = \longleftarrow p_1(t_0, X_1), \ldots, p_m(X_{m-1}, X_m)$$

es una meta tal que t_0 es un término sin variables. Entonces el analizador por cartas $A(\Gamma, G)$ es tal que:

(a) Todas las aristas en $C(\Gamma)$ representan consecuencias lógicas de $\Gamma \cup \{G\}$.

(b) Si $\Gamma \models p_1(t_0, t_1) \wedge \ldots \wedge p_m(t_{m-1}, t_m)$ entonces después de un número finito de aplicaciones de las reglas (i),(ii) y (RF) la carta $C(\Gamma)$ contiene las aristas (pasivas):

$\langle t_0, t_1, p_1 \longleftarrow \rangle$

$\langle t_1, t_2, p_2 \longleftarrow \rangle$

$\vdots$

$\langle t_{m-1}, t_m, p_m \longleftarrow \rangle$

Demostración Es inmediata de los teoremas 6.7 y 6.9.

Parte III

7. Relación entre Programas Cadena y Programas Orientados a Estados

En este capítulo estudiamos la relación que existe entre los programas cadena y los programas orientados a estados. La idea principal, es que bajo ciertas condiciones, podemos establecer una correspondencia biunívoca entre un subconjunto de los programas orientados a estados y los programas cadena de modo tal que las relaciones derivadas del primero determinen al conjunto de consecuencias lógicas en la base de Herbrand del segundo.

Sea Γ un programa cadena. Podemos suponer, sin perder generalidad, que si un símbolo de predicado aparece en una cláusula unitaria, entonces no aparece en la cabeza de cláusulas no unitarias. En caso de que esta restricción no se cumpla, siempre existe un programa cadena Γ' lógicamente equivalente con tal característica. Por ejemplo, consideremos el programa cadena del ejemplo 6.3:

$av([St,[Y \mid Z]],\ [[Y \mid St],Z]) \longleftarrow$

$av([St,[Y \mid Z]],\ [St,Z]) \longleftarrow$

$re([[Y \mid St],Z],\ [St,[Y \mid Z]]) \longleftarrow$

$sb([St,[]],\ [St,[]]) \longleftarrow$

$sb(X_0,X_2) \longleftarrow av(X_0,X_1), sb(X_1,X_2)$

$sb(X_0,X_3) \longleftarrow av(X_0,X_1), sb(X_1,X_2), re(X_2,X_3)$

Puede verse que el predicado sb no satisface la restricción. Entonces construimos el siguiente programa cadena:

$av([St,[Y \mid Z]],\ [[Y \mid St],Z]) \longleftarrow$

$av([St,[Y \mid Z]],\ [St,Z]) \longleftarrow$

$re([[Y \mid St],Z],\ [St,[Y \mid Z]]) \longleftarrow$

$\widehat{sb}([St,[]],\ [St,[]]) \longleftarrow$

$sb(X_0,X_2) \longleftarrow av(X_0,X_1), sb(X_1,X_2)$

$sb(X_0,X_3) \longleftarrow av(X_0,X_1), sb(X_1,X_2), re(X_2,X_3)$

$sb(X_0, X_1) \longleftarrow \widehat{sb}(X_0, X_1)$

Donde se sustituyó la cláusula unitaria correspondiente al predicado sb y se agregó una cláusula intermedia $\widehat{sb}$, para preservar el significado del programa. Ahora asociemos a Γ el programa orientado a estados $\mathcal{P} = \langle D, C, V, I, SB \rangle$, tal que:

$D = U_{\mathcal{L}}$ donde $U_{\mathcal{L}}$ es el universo de Herbrand para el lenguaje de primer orden.

$V = \{R \mid r$ es un símbolo de predicado en Γ y r no aparece en ninguna cláusula unitaria$\}$.

Si $q(s_1, s_2) \longleftarrow$ es una cláusula unitaria en Γ entonces $Q = \{(t_1, t_2) \in D \times D \mid$ existe una sustitución θ tal que $q(t_1, t_2) = q(s_1\theta, s_2\theta)\}$, es un comando en C.

Si $p_0(X_0, X_m) \longleftarrow p_1(X_0, X_1), \ldots, p_m(X_{m-1}, X_m)$ es una cláusula en Γ, entonces la inclusión $P_0 \supseteq P_1; P_2; \cdots; P_m$ está en I (esta asociación tiene sentido ya que supusimos que si un símbolo de predicado aparece en una cláusula unitaria, entonces no aparece en la cabeza de cláusulas no unitarias).

En nuestro ejemplo tenemos que:

$C = \{AV, \widehat{SB}, RE\}$ donde

$$
\begin{aligned}
AV &= \{(t_1, t_2) \in U_{\mathcal{L}} \times U_{\mathcal{L}} \mid \text{existe } \theta \text{ tal que } av(t_1, t_2) = av([St, [Y \mid Z]]\theta, [[Y \mid St], Z]\theta)\} \\
&\cup \{(t_1, t_2) \in U_{\mathcal{L}} \times U_{\mathcal{L}} \mid \text{existe } \theta \text{ tal que } av(t_1, t_2) = av([St, [Y \mid Z]]\theta, [St, Z]\theta)\} \\
\widehat{SB} &= \{(t_1, t_2) \in U_{\mathcal{L}} \times U_{\mathcal{L}} \mid \text{existe } \theta \text{ tal que } \widehat{sb}(t_1, t_2) = \widehat{sb}([St, []]\theta, [St, []]\theta)\} \\
RE &= \{(t_1, t_2) \in U_{\mathcal{L}} \times U_{\mathcal{L}} \mid \text{existe } \theta \text{ tal que } re(t_1, t_2) = re([[Y \mid St], Z]\theta, [St, [Y \mid Z]]\theta)\}
\end{aligned}
$$

$V = \{SB\}$

I consta de las siguientes inclusiones:

$SB \supseteq AV; SB$

$SB \supseteq AB; SB; RE$

$$SB \supseteq \widehat{SB}$$

Ahora supongamos que $\mathcal{P} = \langle D, C, V, I, S\rangle$ es un POE y queremos asociarle un programa cadena Γ. Como D es un conjunto arbitrario y los comandos en C son relaciones binarias sobre D, esta asociación *no* es posible en general; de hecho sólo es posible cuando se cumple la siguiente condición:

(*) Existe una función suprayectiva[1] $\overline{(\ )} : U_{\mathcal{C}} \longrightarrow D$ y para todo comando Q en C, existe un conjunto finito Δ_Q de cláusulas unitarias de la forma $q(s_1, s_2) \longleftarrow$, con $var(s_2) \subseteq var(s_1)$ tales que:

$(\overline{t_1}, \overline{t_2}) \in Q$ si y sólo si existen $q(s_1, s_2) \longleftarrow \in \Delta_Q$ y una sustitución θ tal que $q(t_1, t_2) = q(s_1\theta, s_2\theta)$.

En caso de que $\mathcal{P}$ satisfaga esta condición basta entonces asociarle el programa cadena que consta de las cláusulas unitarias en los Δ_Q para todo Q en C, y las cláusulas $p_0(X_0, X_m) \longleftarrow p_1(X_0, X_1), \ldots, p_m(X_{m-1}, X_m)$ para toda inclusión $P_0 \supseteq P_1; P_2; \cdots; P_m$ en I.

Observación 7.1 *Cuando asociamos un programa orientado a estados $\mathcal{P}$ a un programa cadena Γ, haciendo $\overline{(\ )} : U_{\mathcal{C}} \longrightarrow U_{\mathcal{C}}$ la identidad en $U_{\mathcal{C}}$, puede verse que se cumple la condición (*) y que el programa cadena asociado a $\mathcal{P}$ resulta ser precisamente Γ.*

Observación 7.2 *Ya sea que asociemos un programa orientado a estados $\mathcal{P} = \langle D, C, V, I, S\rangle$ a un programa cadena Γ, o viceversa, se cumple que:*

1. *La inclusión $P_0 \supseteq P_1; P_2; \cdots; P_m$ está en I si y solo si la cláusula $p_0(X_0, X_m) \longleftarrow p_1(X_0, X_1), \ldots, p_m(X_{m-1}, X_m)$ está en Γ.*
2. *Si t_1 y t_2 son términos sin variables entonces: Q es un comando y $(\overline{t_1}, \overline{t_2}) \in Q$ si y sólo si existen una cláusula unitaria $q(s_1, s_2) \longleftarrow$ en Γ y una sustitución θ tales que $q(t_1, t_2) = q(s_1\theta, s_2\theta)$.*
3. *r es un símbolo de predicado en Γ si y sólo si $R \in V \cup C$.*

[1] $\overline{(\ )}$ tiene que ser suprayectiva ya que de otro modo no podríamos hacer referencia todos los estados en D.

Lema 7.3 *Sea Γ un programa cadena y $\mathcal{P} = \langle D, C, V, I, S\rangle$ un POE asociado a Γ (o viceversa). Sea Φ cualquier función asociada a $\mathcal{P}$ y $G =\!\longleftarrow q_1(t_0,t_1),\ldots,q_m(t_{m-1},t_m)$ una meta tal que t_0 es un término sin variables. Entonces para toda respuesta calculada σ para $\Gamma \cup \{G\}$ se cumple que:*

$(\overline{t_0\sigma}, \overline{t_1\sigma}) \in \Phi(Q_1)$

$(\overline{t_1\sigma}, \overline{t_2\sigma}) \in \Phi(Q_2)$

$\vdots$

$(\overline{t_{m-1}\sigma}, \overline{t_m\sigma}) \in \Phi(Q_m)$

Demostración Primero observemos que, por el lema 6.4, $t_0\sigma, t_1\sigma, t_2\sigma, \ldots, t_m\sigma$ son términos sin variables, de modo que tiene sentido considerar a los $(\overline{t_{j-1}\sigma}, \overline{t_j\sigma})$ como posibles elementos de $\Phi(Q_j)$ (recordemos que $\overline{\ \ }$ tiene como dominio al conjunto de términos sin variables y como rango a los elementos de D).

Sea cualquier refutación SLD para $\Gamma \cup \{G\}$ con respuesta calculada σ. Usando un argumento similar al que se dió en la parte final del lema 6.4 podemos suponer, sin perder generalidad, que la refutación es izquierda. La demostración será por inducción sobre el número de pasos de la refutación.

Si la refutación se obtuvo en un paso

$G_0 =\!\longleftarrow q_1(t_0,t_1)$

$G_1 = \square$

De modo que forzosamente se escogió una cláusula en Γ de la forma $C_1 = q_1(s_0,s_1) \longleftarrow$ donde $t_1\sigma = s_1\sigma$ y $s_0\sigma = t_0\sigma$. Entonces por la parte (2) de la observación 7.2 tenemos que Q_1 es un comando y $(\overline{t_0\sigma}, \overline{t_1\sigma}) \in Q_1$. Por ser Φ una función asociada y Q_1 un comando, entonces $\Phi(Q_1) = Q_1$. Por lo tanto $(\overline{t_0\sigma}, \overline{t_1\sigma}) \in \Phi(Q_1)$ y la afirmación se cumple para refutaciones de un paso.

Ahora supongamos como hipótesis inductiva que la afirmación se cumple para refutaciones con k o menos pasos

Si la refutación fue de $k+1$ pasos, entonces

$$G_0 = G =\!\longleftarrow q_1(t_0,t_1),\ldots,q_j(t_{j-1},t_j),\ldots,q_m(t_{m-1},t_m)$$

y el átomo seleccionado en el primer paso fue $q_1(t_0,t_1)$ ya es el que está más a la izquierda.

Entonces, ocurrió alguno de los siguientes casos:

(a) Se escogió una cláusula de la forma

$$C_1 = q_1(s_0, s_1) \longleftarrow$$

entonces,

$$G_1 = \longleftarrow (q_2(t_1, t_2), \ldots, q_m(t_{m-1}, t_m))\sigma_1$$

donde $t_1\sigma_1 = s_1\sigma_1$ y $s_0\sigma_1 = t_0\sigma_1 = t_0$ ya que t_0 es un término sin variables. Pero como Γ es un programa cadena $var(s_1) \subseteq var(s_0)$ entonces

$$var(t_1\sigma_1) = var(s_1\sigma_1) \subseteq var(s_0\sigma_1) = var(t_0) = \emptyset$$

Por lo tanto $t_1\sigma_1$ es un término sin variables. Ahora sea $\sigma' = \sigma_2\sigma_3 \cdots \sigma_{k+1}$ entonces

$$t_0\sigma = (t_0\sigma_1)\sigma' = (s_0\sigma_1)\sigma' = s_0\sigma$$

$$t_1\sigma = (t_j\sigma_1)\sigma' = (s_1\sigma_1)\sigma' = s_1\sigma$$

Como $t_0\sigma$ y $t_1\sigma$ son términos sin variables y $q_1(s_0, s_1) \longleftarrow$ es una cláusula unitaria, entonces por la parte (2) de la observación 7.2 tenemos que Q_1 es un comando y $(\overline{t_0\sigma}, \overline{t_1\sigma}) \in Q_1$. Por ser Φ una función asociada y un Q_1 comando, entonces $\Phi(Q_1) = Q_1$ y por lo tanto:

$$(\overline{t_0\sigma}, \overline{t_1\sigma}) \in \Phi(Q_1)$$

Por otro lado:

$$G_1 = \longleftarrow q_2(t_1\sigma_1, t_2\sigma_1), \ldots, q_m(t_{m-1}\sigma_1, t_m\sigma_1)$$

Por la observación 5.33 tenemos una refutación izquierda para $\Gamma \cup \{G_1\}$ de k pasos con respuesta calculada σ' (restringida a las variables en G_1). Como ya habiamos observado que $t_1\sigma_1$ es un término sin variables aplicando la hipótesis inductiva concluimos que:

$$(\overline{t_1\sigma}, \overline{t_2\sigma}) = (\overline{t_1\sigma_1\sigma'}, \overline{t_2\sigma_1\sigma'}) \in \Phi(Q_2)$$

$$\vdots$$

$$(\overline{t_{m-1}\sigma}, \overline{t_m\sigma}) = (\overline{t_{m-1}\sigma_1\sigma'}, \overline{t_m\sigma_1\sigma'}) \in \Phi(Q_m)$$

y como ya teníamos que $(\overline{t_0\sigma}, \overline{t_1\sigma}) \in \Phi(Q_1)$, el resultado se cumple en este caso.

(b) Se escogió una cláusula de la forma:

$$C_1 = q_1(Y_0, Y_n) \longleftarrow r_1(Y_0, Y_1), \ldots, r_n(Y_{n-1}, Y_n)$$

entonces:

$$\begin{aligned} \sigma_1 &= \{Y_0/t_0, Y_n/t_1\} \\ G_1 &= \longleftarrow (r_1(Y_0, Y_1), \ldots, r_n(Y_{n-1}, Y_n), \\ &\qquad q_2(t_1, t_2), \ldots, q_m(t_{m-1}, t_m))\sigma_1 \\ &= \longleftarrow r_1(t_0, Y_1\sigma_1), \ldots, r_n(Y_{n-1}\sigma_1, t_1), \\ &\qquad q_2(t_1\sigma_1, t_2\sigma_1), \ldots, q_m(t_{m-1}\sigma_1, t_m\sigma_1) \end{aligned}$$

Como las variables de la cláusula C_1 están ajenizadas, σ_1 no actúa sobre variables en t_0 y t_1, entonces $t_0\sigma_1 = t_0$ y $t_1\sigma_1 = t_1$. Por lo tanto:

$$\begin{aligned} G_1 &= \longleftarrow r_1(t_0\sigma_1, Y_1\sigma_1), \ldots, r_n(Y_{n-1}\sigma_1, t_1\sigma_1), \\ &\qquad q_2(t_1\sigma_1, t_2\sigma_1), \ldots, q_m(t_{m-1}\sigma_1, t_m\sigma_1) \end{aligned}$$

Por la observación 5.33 tenemos una refutación izquierda de k pasos para $\Gamma \cup \{G_1\}$ con respuesta calculada $\sigma' = \sigma_2\sigma_3 \cdots \sigma_{k+1}$ restringida a las variables en G_1. Observando que $\sigma = \sigma_1\sigma'$ (restringida a las variables en G_0) y que $t_0\sigma_1$ es un término sin variables, aplicamos la hipótesis inductiva y obtenemos que:

$(\overline{t_1\sigma}, \overline{t_2\sigma}) \in \Phi(Q_2)$

$\vdots$

$(\overline{t_{m-1}\sigma}, \overline{t_m\sigma}) \in \Phi(Q_m)$

y

$(\overline{t_0\sigma}, \overline{Y_1\sigma}) \in \Phi(R_1)$

$(\overline{Y_1\sigma}, \overline{Y_2\sigma}) \in \Phi(R_2)$

$\vdots$

$(\overline{Y_{n-1}\sigma}, \overline{t_1\sigma}) \in \Phi(R_n)$

donde $Y_1\sigma, Y_2\sigma, \ldots, Y_{n-1}\sigma$ son términos sin variables por el lema 6.4.

De modo que sólo resta mostrar que $(\overline{t_0\sigma}, \overline{t_1\sigma}) \in \Phi(Q_1)$, pero $C_1 = q_1(Y_0, Y_n) \longleftarrow r_1(Y_0, Y_1), \ldots, r_n(Y_{n-1}, Y_n)$ es una cláusula en Γ, entonces por la parte (1) de la observación 7.2 $Q_1 \supseteq R_1; \cdots; R_n$ es una inclusión en I. Por ser Φ una función asociada resulta que:

$$\Phi(Q_1) \supseteq \Phi(R_1); \cdots; \Phi(R_n)$$

y como sabíamos que:

$(\overline{t_0\sigma}, \overline{Y_1\sigma}) \in \Phi(R_1)$

$(\overline{Y_1\sigma}, \overline{Y_2\sigma}) \in \Phi(R_2)$

$\vdots$

$(\overline{Y_{n-1}\sigma}, \overline{t_1\sigma}) \in \Phi(R_n)$

aplicando la definición 2.2 (composición de relaciones binarias) concluimos que:

$$(\overline{t_0\sigma}, \overline{t_1\sigma}) \in \Phi(Q_1)$$

y el lema queda demostrado. ◇

Lema 7.4 *Sea Γ un programa cadena y $\mathcal{P} = \langle D, C, V, I, S\rangle$ un POE asociado a Γ (o viceversa). Sea Φ cualquier función asociada a $\mathcal{P}$, sean t_0 y t_1 términos sin variables y sea r un símbolo de predicado en Γ. Si $\Gamma \models r(t_0, t_1)$ entonces $(\overline{t_0}, \overline{t_1}) \in \Phi(R)$.*

Demostración Si $\Gamma \models r(t_0, t_1)$ y t_0, t_1 son términos sin variables, entonces por la observación 5.38, existe una refutación SLD para $\Gamma \cup \{\longleftarrow r(t_0, t_1)\}$ con respuesta calculada ε. Por lo tanto, por el lema 7.3, se cumple que $(\overline{t_0}, \overline{t_1}) = (\overline{t_0\varepsilon}, \overline{t_1\varepsilon}) \in \Phi(R)$. ◇

Ahora estamos en condiciones de enunciar el resultado principal de este capítulo.

Teorema 7.5 *Sea Γ un programa cadena y $\mathcal{P} = \langle D, C, V, I, S\rangle$ un POE asociado a Γ (o viceversa). Sea Ω la mínima función asociada a $\mathcal{P}$, sean t_0 y t_1 términos sin variables y sea r un símbolo de predicado en Γ. Entonces $\Gamma \models r(t_0, t_1)$ si y sólo si $(\overline{t_0}, \overline{t_1}) \in \Omega(R)$.*

Demostración Definamos la función $\Omega' : V \cup C \longrightarrow Pot(D \times D)$ como:

$$\Omega'(R) = \{(\overline{t_0}, \overline{t_1}) \mid \Gamma \models r(t_0, t_1) \text{ donde } t_0 \text{ y } t_1 \text{ son términos sin variables}\}$$

Es claro que el teorema queda demostrado si probamos que Ω' coincide con Ω.

Por el lema 7.4 y la parte (3) de la observación 7.2 es inmediato que si Φ es cualquier función asociada a $\mathcal{P}$, entonces:

$$\Omega'(R) \subseteq \Phi(R) \text{ para todo } R \in V$$

Por lo tanto basta mostrar que Ω' es una función asociada, ya que entonces sería la mínima de tales funciones.

1. Supongamos que $P_0 \supseteq P_1; P_2; \cdots; P_m$ está en I. Vamos a demostrar que $\Omega'(P_0) \supseteq \Omega'(P_1); \Omega'(P_2); \cdots; \Omega'(P_m)$:

   Por la parte (1) de la observación 7.2 tenemos que la cláusula

   $$p_0(X_0, X_m) \longleftarrow p_1(X_0, X_1), \ldots, p_m(X_{m-1}, X_m)$$

   está en Γ. Entonces

   $$\Gamma \models \forall(p_0(X_0, X_m) \longleftarrow p_1(X_0, X_1), \ldots, p_m(X_{m-1}, X_m)) \qquad (7.1)$$

   Sea $(\overline{t_0}, \overline{t_m}) \in \Omega'(P_1); \Omega'(P_2); \cdots; \Omega'(P_m)$. Por la definición 2.2 existen $\overline{t_1}, \overline{t_2}, \ldots, \overline{t_{m-1}}$ en D tales que:

   $$(\overline{t_0}, \overline{t_1}) \in \Omega'(P_1)$$
   $$(\overline{t_1}, \overline{t_2}) \in \Omega'(P_2)$$
   $$\vdots$$
   $$(\overline{t_{m-1}}, \overline{t_m}) \in \Omega'(P_m)$$

   Entonces por definición de Ω' tenemos que:

   $$\Gamma \models p_1(t_0, t_1)$$
   $$\Gamma \models p_2(t_1, t_2)$$
   $$\vdots$$
   $$\Gamma \models p_m(t_{m-1}, t_m)$$

   Que junto con (7.1) implican que $\Gamma \models p_0(t_0, t_m)$. Nuevamente por definición de Ω' concluimos que $(\overline{t_0}, \overline{t_m}) \in \Omega'(P_0)$. Por lo tanto $\Omega'(P_0) \supseteq \Omega'(P_1); \Omega'(P_2); \cdots; \Omega'(P_m)$.

2. Sea Q un comando en C. Vamos a demostrar que $\Omega'(Q) = Q$:

   $\boxed{\subseteq}$ Si $(\overline{t_0}, \overline{t_1}) \in \Omega'(Q)$ entonces $\Gamma \models q(t_0, t_1)$ donde t_0 y $t_1$son términos sin variables. Entonces por la observación 5.38, existe una refutación SLD para $\Gamma \cup \{\longleftarrow q(t_0, t_1)\}$ con respuesta calculada ε. Pero como Q es un comando entonces el símbolo de predicado q sólo aparece en la cabeza de una cláusula en Γ si ésta es unitaria.. Entonces existe una cláusula $q(s_0, s_1) \longleftarrow$ en Γ y una sustitución θ tal que $q(t_0, t_1) = q(s_0\theta, s_1\theta)$. Por la parte (2) de la observación 7.2 concluimos que $(\overline{t_0}, \overline{t_1}) \in Q$ y por lo tanto $\Omega'(Q) \subseteq Q$.

   $\boxed{\supseteq}$ Si $(\overline{t_0}, \overline{t_1}) \in Q$ entonces, por la parte (2) de la observación 7.2, t_0 y $t_1$son términos sin variables y existe una cláusula unitaria $q(s_0, s_1) \longleftarrow$ en Γ y una sustitución θ tal que $q(t_0, t_1) = q(s_0\theta, s_1\theta)$. Entonces $\Gamma \models \forall(q(s_0, s_1))$ y en particular $\Gamma \models q(t_0, t_1)$. Concluimos que $(\overline{t_0}, \overline{t_1}) \in \Omega'(Q)$ y por lo tanto $Q \subseteq \Omega'(Q)$.

Por lo tanto Ω' es una función asociada. Por lo tanto Ω' es la mínima de las funciones asociadas. Por lo tanto $\Omega' = \Omega$ y el teorema queda demostrado.$\diamond$

Conclusiones

En la primera parte de la tesis demostramos los siguientes dos teoremas:

- Sea $\mathcal{P} = \langle D, C, V, I, S \rangle$ un programa orientado a estados y $\mathcal{GR}(\mathcal{P}) = \left\langle N, T, \overline{S}, P \right\rangle$ la gramática adjunta a $\mathcal{P}$. Entonces existe una mínima función asociada a $\mathcal{P}$, la cual denotamos con Ω, y las relaciones derivadas de $\mathcal{P}$ están caracterizadas de la siguiente manera:

  para toda etiqueta R en V se tiene que,

  $$\Omega(R) = \bigcup \left\{ Q_1; Q_2; \cdots; Q_n \mid \overline{Q}_1 \overline{Q}_2 \cdots \overline{Q}_n \in L[\overline{R}],\ n \geq 1 \right\}$$

  donde $Q_1, Q_2, \ldots, Q_n$ son comandos de $\mathcal{P}$, $\overline{Q}_1, \overline{Q}_2, \cdots, \overline{Q}_n$ son símbolos terminales de $\mathcal{GR}(\mathcal{P})$ y $L[\overline{R}]$ es el lenguaje generado a partir del símbolo $\overline{R}$.

- Sea $\mathcal{G} = \langle N, T, S, P \rangle$ una gramática libre de contexto y $\mathcal{PR}(\mathcal{G}) = \left\langle D, C, V, I, \widehat{S} \right\rangle$ el programa asociado a $\mathcal{G}$. Entonces la relación principal de $\mathcal{PR}(\mathcal{G})$ es igual a la representación cociente del lenguaje generado por la gramática, es decir:

  $$\Omega(\widehat{S}) = \mathcal{Q}(L(\mathcal{G}))$$

Estos resultados caracterizan a la relación que existe entre las gramáticas libres de contexto y los programas orientados a estados. Es importante recordar que a distintos programas orientados a estados puede corresponderles la misma gramática adjunta. De modo que las gramáticas formales no recuperan el significado de los programas orientados a estados. En cambio, un programa asociado a una gramática libre de contexto si recupera el significado de ésta. De hecho, como se vio en la demostración de teorema 2.18, $\mathcal{G}$ y $\mathcal{GR}(\mathcal{PR}(\mathcal{G}))$ resultan ser la misma gramática. Por el contrario, $\mathcal{P}$ y $\mathcal{PR}(\mathcal{GR}(\mathcal{P}))$ en general no son el mismo programa. El primer resultado no había sido publicado antes, mientras que el segundo aparece sin demostración en [Ros93].

En la segunda parte de la tesis mostramos que:

- Si Γ es un programa cadena y $G = \longleftarrow p_1(t_0, X_1), \ldots, p_m(X_{m-1}, X_m)$ es una meta tal que t_0 es un término sin variables. Entonces el analizador por cartas $A(\Gamma, G)$ es tal que:

1. Todas las aristas en $C(\Gamma)$ representan consecuencias lógicas de $\Gamma \cup \{G\}$.
2. Si $\Gamma \models p_1(t_0,t_1) \wedge \ldots \wedge p_m(t_{m-1},t_m)$ entonces después de un número finito de aplicaciones de las reglas (i),(ii) y (RF) la carta $C(\Gamma)$ contiene las aristas (pasivas):

$$\langle t_0, t_1, p_1 \longleftarrow \rangle, \langle t_1, t_2, p_2 \longleftarrow \rangle, \ldots, \langle t_{m-1}, t_m, p_m \longleftarrow \rangle$$

En [Ros92], [Mel91] y [yCRS92] aparecen ejemplos del uso de analizadores por cartas para inferir consecuencias lógicas de programas cadena, y en [Ros93] se da una demostración de validez, pero la completud no había sido demostrada antes.

Por último la tercera parte de la tesis conecta los programas orientados a estados con los programas cadena a través del siguiente teorema:

- Sea Γ un programa cadena y $\mathcal{P} = \langle D, C, V, I, S \rangle$ un programa orientado a estados asociado a Γ (o viceversa). Sea Ω la mínima función asociada a $\mathcal{P}$, sean t_0 y t_1 términos sin variables y sea r un símbolo de predicado en Γ. Entonces $\Gamma \models r(t_0,t_1)$ si y sólo si $(\overline{t_0}, \overline{t_1}) \in \Omega(R)$.

Que hasta donde sabemos tampoco había sido demostrado. Aquí cabe mencionar que el restringir nuestro trabajo a los programas cadena no es una limitación computacional seria, ya que los programas cadena también están en la misma jerarquía que las máquinas de Turing. Mas aun, existen algoritmos que permiten transformar un programa lógico arbitrario en un programa cadena lógicamente equivalente (véase [Ros92]).

Por último sólo resta agregar que el trabajo de esta tesis puede continuarse para abarcar el caso general de utilizar cualquier analizador sintáctico como sistema formal para programas cadena y posiblemente una clase más amplia de programas lógicos que incluya fórmulas con variables.

Bibliografía

[AU72] Alfred V. Aho and Jeffrey D. Ullman. *The Theory of Parsing, Translation, and Compiling*, volume 1. Prentice Hall, 1972.

[End87] Herbert B. Enderton. *Una Introducción Matemática a la Lógica.* Universidad Nacional Autónoma de México, 1987.

[GM89] Gerald Gazdar and Chris Mellish. *Natural Language Processing in Prolog. An Introduction to Computational Linguistics.* Addison-Wesley, 1989.

[Gur92] Elisa Viso Gurovich. *Notas de Autómatas y Lenguajes Formales.* Facultad de Ciencias. Universidad Nacional Autónoma de México, 1992.

[HU69] John E. Hopcroft and Jeffrey D. Ullman. *Formal Languages and their Relation to Automata.* Addison-Wesley, 1969.

[JLM86] J. Jaffar, J-L. Lassez, and M.J. Maher. Some issues and trends in the semantics of logic programming. In *Proceedings of the Third International Conference on Logic Programming*, pages 223–241, London, 1986.

[Kow74] Robert Kowalski. Predicate logic as programming language. In *Proc. IFIP 74*, pages 569–574, 1974.

[Lin90] Peter Linz. *An Introduction to Formal Languages and Automata.* D.C. Heath and Company, 1990.

[Llo87] J.W. Lloyd. *Foundations of Logic Programming.* Springer-Verlag, 2nd edition, 1987.

[Mel91] Rafael Ramirez Melendez. *Programas Lógicos y Lenguajes Libres de Contexto.* Tesis de Licenciatura. Universidad Nacional Autónoma de México, 1991.

[Men64] Elliott Mendelson. *Mathematical Logic.* D. Van Nostrand Company, 1964.

[Rob65] J.A. Robinson. A machine-oriented logic based on the resolution principle. *J. ACM*, 12:23–41, 1965.

[Ros92] David A. Rosenblueth. Chart parsers as proof procedures for fixed-mode logic programs. In *Proceedings of the International Conference on Fifth Generation Computer Systems*, pages 1125–1132, Tokio, Japón, 1992.

[Ros93] David A. Rosenblueth. An execution mechanism for nondeterministic, state-oriented programs based on a chart parser. *Information Processing Letters*, 45:211–217, 1993.

[Sal73] Arto Salomaa. *Formal Languages*. Academic Press, 1973.

[yCRS92] Julio C. Peralta y Carlos R. Silva. *Una conexión entre las Gramáticas Formales y las Bases de Datos Deductivas*. Tesis de Licenciatura. Universidad Nacional Autónoma de México, 1992.