Capítulo 4

Unificación y resolución

Una vez estudiado cómo transformar un conjunto de fórmulas en un conjunto de cláusulas, volvemos al problema central de la lógica. Estudiar cuando una deducción es correcta.

Nos situamos en un problema sencillo. Supongamos que queremos comprobar si $\forall x P(x) \vDash P(a)$. Esto claramente es cierto. Pero queremos encontrar un método más allá de nuestra intuición para poder concluir que esta implicación semántica es verdadera.

Sabemos que comprobar si $\forall x P(x) \vDash P(a)$ es equivalente a comprobar si $\{\forall x P(x), \neg P(a)\}$ es insatisfacible. Y si pasamos a forma clausular, esto es lo mismo que comprobar si el conjunto de cláusulas $\{P(x),\ \neg P(a)\}$ es insatisfacible.

Si quisiéramos aplicar un método análogo al método de resolución estudiado en el tema de lógica proposicional necesitaríamos, a partir de las cláusulas $P(x)$ y $\neg P(a)$ poder deducir la cláusula vacía. Esto requeriría poder "cambiar" la cláusula $P(x)$ por $P(a)$, y con las cláusulas $P(a)$ y $\neg P(a)$ deducir la cláusula vacía.

Dicho de otra forma, tenemos dos literales, $P(x)$ y $P(a)$, y queremos sustituir los símbolos de variable que intervienen por otros términos de forma que los dos literales sean iguales (los símbolos de constante, por ser constantes, no pueden sustituirse por nada).

El hecho de cambiar una variable por un término es lo que llamaremos *sustitución.* Encontrar la sustitución (o sustituciones) apropipada para hacer dos literales iguales se llama *unificación.*

4.1. Unificación

El objetivo de esta sección es, como ya hemos avanzado, dados dos o más literales, estudiar si es posible transformarlas de forma que sean iguales. Las transformaciones que van a estar permitidas son aquellas en las que una variable se sustituye por un término.

4.1.1. Sustituciones

Recordemos que un literal es una fórmula atómica o la negación de una fórmula atómica.

Definición 42. *Sea α un literal, x una variable con alguna ocurrencia en la fórmula α y t un término cualquiera. La* sustitución elemental *de x por t es la transformación de la fórmula α en otra fórmula en la que cada ocurrencia de la variable x es sustituida por el término t.*

Dicha sustitución la representaremos como $\sigma = (x|t)$, y la fórmula resultante de aplicar la sustitución σ, como $\sigma(\alpha)$

Ejemplo 4.1.1.

En la fórmula $R(f(x), a, g(h(x), y))$ realizamos la sustitución $(x|g(a, y))$. Nos queda entonces la fórmula $R(f(g(a, y)), a, g(h(g(a, y)), y))$.

En principio, al sustituir una variable x por un término, dicha variable puede aparecer en el término por el que sustituimos, así, podemos hacer la sustitución $\sigma = (x|f(x))$ y la fórmula α del ejemplo anterior, después de la sustituición quedaría

$$\sigma(\alpha) = R(f(f(x)), a, g(h(f(x)x), y)).$$

Obviamente, a la fórmula que resulta de aplicarle una sustitución elemental podemos aplicarle otra sustitución elemental. Si σ_0 es la primera sustitución que hacemos, y σ_1 la segunda, escribiremos la sustitución total como $\sigma_1\sigma_0$. Dicha sustitución la denominaremos *composición de* σ_0 *y* σ_1

Ejemplo 4.1.2.

Según el ejemplo anterior, el resultado de aplicar la sustitución $\sigma_0 = (x|g(a,y))$ *a la fórmula* $R(f(x),a,g(h(x),y))$ *es* $R(f(g(a,y)),a,g(h(g(a,y)),y))$.

Si ahora sustituimos la variable y *por* $g(a,h(a))$ *(es decir, aplicamos la sustitución* $\sigma_1 = (y|g(a,h(a)))$*), obtenemos la fórmula*

$$R(f(g(a,g(a,h(a)))),a,g(h(g(a,g(a,h(a)))),g(a,h(a))))$$

La sustitución realizada es entonces $\sigma = (y|g(a,h(a)))\ (x|g(a,y))$

Notemos que el orden en que tomamos las sustituciones elementales es importante. Puede comprobarse que no es lo mismo $\sigma = \sigma_1\sigma_0$ *que* $\tau = \sigma_0\sigma_1$.

Tenemos que:

$$\sigma(\alpha) = R(f(g(a,g(a,h(a)))),a,g(h(g(a,g(a,h(a)))),g(a,h(a))))$$

$$\tau(\alpha) = R(f(g(a,y)),a,g(h(g(a,y)),g(a,h(a))))$$

Comprueba que efectivamente éste es el resultado.

Si observamos el ejemplo anterior, vemos que el efecto de la sustitución σ es el de cambiar cada aparición de x por el término $g(a,g(a,h(a)))$, y cada aparición de y por $g(a,h(a))$. Dicha sustitución podemos representarla como

$$\sigma = (x|g(a,g(a,h(a)));\ y|g(a,h(a)))$$

Nótese que en este caso el orden en que lo escribamos no influye, es decir,

$$(x|g(a,g(a,h(a)));\ y|g(a,h(a))) = (y|g(a,h(a));\ x|g(a,g(a,h(a))))$$

La composición en orden inverso, es decir, la sustitución τ podríamos representarla ahora como

$$\tau = (x|g(a,y);\ y|g(a,h(a)))$$

que claramente no coincide con la sustitución σ.

En general, podemos dar la siguiente definición.

Definición 43. *Sea* α *un literal en un lenguaje de primer orden.*

Una sustitución en α *es la transformación que consiste en sustituir algunas (o todas) las variables que aparecen en* α *por términos del lenguaje.*

Si $x_1, x_2, \cdots, x_n$ *son las variables que vamos a sustituir, y* $t_1, t_2, \cdots, t_n$ *los términos por los que las sustituimos, entonces representaremos la sustitución como*

$$(x_1|t_1;\ x_2|t_2;\ \cdots\ x_n|t_n)$$

Ejercicio 4.1.1. *Comprueba que las sustituciones* $(x|y)\ (y|x)$, $(y|x)\ (x|y)$ *y* $(x|y;\ y|x)$ *son todas distintas, y expresa la última como composición de sustituciones elementales.*

4.1.2. Unificadores

A continuación pasamos a dar el concepto de *unificador*. Más adelante veremos como calcular un unificador para dos o más literales.

Definición 44. *Sean* $\alpha_1, \alpha_2, \cdots, \alpha_n$ *literales. Un unificador para tales fórmulas es una sustitución* σ *de forma que* $\sigma(\alpha_1) = \sigma(\alpha_2) = \cdots = \sigma(\alpha_n)$.

Un conjunto de literales se dice unificable si existe un unificador para ellas. Caso contrario, dicho conjunto se dice que no es unificable.

Ejemplo 4.1.3.

1. *Dadas las fórmulas $P(x)$ y $P(a)$, un unificador de ambas es $(x|a)$.*

   Por tanto, los literales $P(x)$ y $P(a)$ son unificables.

2. *Las fórmulas $P(x)$ y $P(f(x))$ no tienen unificador pues para cualquier sustitución $\sigma = (x|t)$ nos va a quedar que $\sigma(P(x)) = P(t)$ mientras que $\sigma(P(f(x))) = P(f(t))$, y estas fórmulas van a ser siempre distintas.*

   Tenemos entonces que los literales $P(x)$ y $P(f(x))$ no son unificables.

3. *Las fórmulas $Q(x, a)$ y $Q(y, b)$ no tienen unificador ya que no podemos sustituir las constantes por ningún término. Por tanto, para cualquier sustitución σ nos quedará que $\sigma(Q(x, a)) = Q(t_1, a)$ y $\sigma(Q(y, b)) = Q(t_2, b)$.*

   Los literales $Q(x, a)$ y $Q(y, b)$ no son unificables.

4. *Para las fórmulas $P(a, x, f(g(y)))$, $P(z, f(z), f(u))$ sí existen unificadores. Por ejemplo, es un unificador $(x|f(a);\ y|f(x);\ z|a;\ u|g(f(x)))$.*

   También $(x|f(a);\ y|a;\ z|a;\ u|g(a))$ es un unificador para estas dos fórmulas.

   Vemos entonces que los literales $P(a, x, f(g(y)))$ y $P(z, f(z), f(u))$ son unificables.

Nótese que si σ es un unificador para un conjunto de fórmulas $\alpha_1, \alpha_2, \cdots, \alpha_n$ y τ es una sustitución cualquiera, entonces $\tau\sigma$ es también un unificador para $\alpha_1, \alpha_2, \cdots, \alpha_n$.

Ejemplo 4.1.4.
Dadas las fórmulas $P(a, x, f(g(y)))$, $P(z, f(z), f(u))$, sabemos que

$$\sigma = (x|f(a);\ y|f(x);\ z|a;\ u|g(f(x)))$$

es un unificador para ambas. Sea ahora $\tau = (x|g(a))$. En tal caso, se tiene que

$$\tau\sigma = (x|f(a);\ y|f(g(a));\ z|a;\ u|g(f(g(a))))$$

que es también un unificador para $P(a, x, f(g(y)))$ y $P(z, f(z), f(u))$.
Sin embargo, $\sigma\tau$ no es unificador para las dos fórmulas, pues

$$\sigma\tau = (x|g(a);\ y|f(x);\ z|a;\ u|g(f(x)))$$

y se tiene que

$$\sigma\tau(P(a, x, f(g(y)))) = P(a, g(a), f(g(f(x))))$$
$$\sigma\tau(P(z, f(z), f(u)) = P(a, f(a), f(g(f(x))))$$

Vamos a analizar con un poco más de detalle este ejemplo.
Si queremos hacer una sustitución para que los literales $P(a, x, f(g(y)))$ y $P(z, f(z), f(u))$ sean iguales, entonces:

- z hay que sustituirla por a (para igualar la primera componente).
- x debe sustituirse por $f(z)$, pero como z se ha sustituido por a entonces x debe sustituirse por $f(a)$.
- Con respecto a u y a y, lo que tenemos es que si y la sustituimos por un término t entonces u hay que sustituirla por el término $g(t)$.

Si analizamos el ejemplo precedente, vemos que para lograr que las fórmulas $P(a, x, f(g(y)))$ y $P(z, f(z), f(u))$ sean iguales después de realizar sustituciones, entonces z debe sustituirse por a (para igualar la primera componente), x debe sustituirse por $f(z) = f(a)$, mientras que u debe sustituirse por lo que le demos a $g(y)$, independientemente del valor de y. Los unificadores que hemos dado para ambas fórmulas satisfacían estas condiciones, y sobre lo que teníamos libertad, que era el término que sustituye a y, ha tomado un valor en cada uno de los casos.
En el primer caso, sustituíamos y por $f(x)$ (y por tanto, u por $g(y) = f(g(x))$)
En el segundo caso, sustituíamos y por a
Y en el tercero, sustituíamos y por $f(g(a))$.
Vemos entonces que hay una condición, que podemos representar en la sustitución $(z|a;\ x|f(a);\ u|g(y))$ que debe satisfacer cualquier unificador.
Esta idea, vamos a formalizarla en la siguiente definición.

Definición 45. *Sean $\alpha_1, \alpha_2, \cdots, \alpha_n$ literales en un lenguaje de primer orden, y sea σ una sustitución.*

Se dice que σ es un **unificador principal** *para $\alpha_1, \alpha_2, \cdots, \alpha_n$ (también se llama* **unificador más general***) si:*

- σ es un unificador para $\alpha_1, \alpha_2, \cdots, \alpha_n$.

- Si τ es otro unificador para $\alpha_1, \alpha_2, \cdots, \alpha_n$, entonces existe una sustitución τ_1 de forma que $\tau = \tau_1\sigma$

Es decir, σ es un unificador principal si todos los unificadores pueden obtenerse a partir de σ.

Ejemplo 4.1.5.

1. *Consideramos nuevamente las fórmulas $P(a, x, f(g(y)))$ y $P(z, f(z), f(u))$. Entonces un unificador principal es*

$$(z|a;\ x|f(a);\ u|g(y)) \ = \ (x|f(a);\ z|a;\ u|g(y))$$

   La sustitución $(x|f(a);\ y|f(x);\ z|a;\ u|g(f(x)))$, que es un unificador para ambas fórmulas, se puede expresar como $(y|f(x))\ (x|f(a);\ z|a;\ u|g(y))$

   La sustitución $(x|f(a);\ y|a;\ z|a;\ u|g(a))$, que también es un unificador para ambas fórmulas, se puede expresar como $(y|a)\ (x|f(a);\ z|a;\ u|g(y))$

   Por último, la sustitución $(x|f(a);\ y|f(g(a));\ z|a;\ u|g(f(g(a))))$ puede expresarse como la composición $(y|f(g(a)))\ (x|f(a);\ z|a;\ u|g(y))$

2. *Un conjunto de literables unificable puede tener más de un unificador principal. Así, las sustituciones $\sigma_1 = (x|y)$, y $\sigma_2 = (y|x)$ son unificadores principales de las fórmulas $P(x, y)$ y $P(y, x)$*

Vamos a dar a continuación dos métodos para decidir si dado un conjunto de literales, éstos son unificables, y caso de serlo, encontrar un unificador principal.

Algoritmo de Unificación

Este algoritmo nos permite un cálculo automático de un unificador más general para un conjunto de literales.

Para poder hacer correr el algoritmo, es necesario definir el concepto de *conjunto de discordancia.*

Definición 46. *Dados dos literales construidos sobre el mismo símbolo de predicado, se define el conjunto de discordancia como el conjunto formado por los primeros términos (recorridos los literales de izquierda a derecha) en los que difieren. En el caso de que los literales sean iguales, el conjunto de discordancia es vacío.*

Para hacer el conjunto de discordancia de los literales $\lambda_1, \lambda_2, \cdots, \lambda_n$ con $n \geq 3$ se hace el conjunto de discordancia de λ_1 y λ_2. Si es vacío, el de λ_1 y λ_3, y así hasta que encontremos uno no vacío (si lo hubiera).

En el caso de que todos los literales coincidan totalmente, el conjunto de discordancia es el conjunto vacío.

Notemos que el conjunto de discordancia, o es vacío, o tiene dos términos. En función de los términos que contiene, el conjunto de discordancia puede ser de tres tipos:

- Tipo 1. Ninguno de los términos del conjunto de discordancia es un símbolo de variable.
- Tipo 2. Uno de los términos es un símbolo de variable, y en el otro término aparece este símbolo de variable.
- Tipo 3. Uno de los términos es un símbolo de variable, y éste no aparece en el otro término.

Ejemplo 4.1.6.

1. *Sean los literales $P(x)$ y $P(y)$. Los literales difieren en los términos que hemos marcado en negrita.*

$$P(\mathbf{x}), \quad P(\mathbf{y}).$$

Por tanto, el conjunto de discordancia es $\{x, y\}$

2. *Sean ahora los literales $\{Q(x, f(x, y))$ y $Q(x, f(y, y))$. Al igual que antes, marcamos en negrita las diferencias entre ambos literales.*

$$Q(x, f(\mathbf{x}, y)); \quad Q(x, f(\mathbf{y}, y)).$$

El conjunto de discordancia es $\{x, y\}$

3. *$\{R(g(x)), R(f(y))\}$ los literales difieren en los términos que ocupan la posición en negrita $R(\mathbf{g}(\mathbf{x})), R(\mathbf{f}(\mathbf{y}))$ Luego el conjunto de discordancia es $\{g(x), f(y)\}$*

4. *Los literales que consideramos ahora son $P(a, x, f(g(y)))$ y $P(z, f(z), f(u))$. Marcamos en negrita las diferencias:*

$$P(\mathbf{a}, \mathbf{x}, f(\mathbf{g}(\mathbf{y}))); \quad P(\mathbf{z}, \mathbf{f}(\mathbf{z}), f(\mathbf{u})).$$

El conjunto de discordancia es $\{a, z\}$ (los primeros términos que difieren).

5. *Sean los literales $P(a, f(x), g(a, f(b)))$, $P(a, f(y), g(x, x))$, $P(b, f(a), g(x, f(x)))$.*

El conjunto de discordancia de los dos primeros es $\{x, y\}$. Luego el conjunto de discordancia de los tres es $\{x, y\}$.

A continuación describimos el algoritmo:

Algoritmo de unificación:

Entrada: Conjunto W formado por dos o más literales.

Salida: Un unificador principal (si existe), o la respuesta *no son unificables.*

Descripción del algoritmo:

$W_0 := W$.

$\sigma := Id$.

Mientras $|W_k| \geq 2$

Tomamos D el conjunto de discordancia de W_k.

Si D es de Tipo 1 o Tipo 2:

- Devuelve: No son unificables.
- Fin.

Si D es tipo 3, de la forma $D = \{x_k, t_k\}$:

- $\tau := (x_k | t_k)$.
- $W_{k+1} := \{\tau(\alpha) : \alpha \in W_k\}$.
- $\sigma := \tau\sigma$.

Devuelve σ.

Fin

Ejemplo 4.1.7.

1. *Sean los literales $\lambda_1 = P(a, x, f(g(y)))$ y $\lambda_2 = P(z, f(z), f(u))$. Vamos a ver si son o no unificables siguiendo el algoritmo de unificación (sabemos que son unificables, pues en el ejemplo 4.1.3 dimos un unificador).*

Para seguir los pasos del algoritmo construimos la siguiente tabla:

D	τ	W_i	σ	$\lvert W_i\rvert$
		$\{P(a,x,f(g(y))),\ P(z,f(z),f(u))\}$	Id	2
$\{a,z\}$	$(z\lvert a)$	$\{P(a,x,f(g(y))),\ P(a,f(a),f(u))\}$	$(z\lvert a)$	2
$\{x,f(a)\}$	$(x\lvert f(a))$	$\{P(a,f(a),f(g(y))),\ P(a,f(a),f(u))\}$	$(x\lvert f(a))\,(z\lvert a)$	2
$\{g(y),u\}$	$(u\lvert g(y))$	$\{P(a,f(a),f(g(y)))\}$	$(u\lvert g(y))\,(x\lvert f(a))\,(z\lvert a)$	1

Vemos que los literales son unificables, y el unificador principal es $\sigma = (u|g(y))\,(x|f(a))\,(z|a) = (x|f(a);\ z|a;\ u|g(y))$.

La primera fila contiene los valores iniciales, y el cálculo del cardinal del conjunto W_i*, para ver si entramos en el bucle.*

Luego, cada una de las filas es cada una de las iteraciones del bucle.

2. *Sean ahora los literales* $\lambda_1 = P(x,y)$, $\lambda_2 = P(f(z),x)$ *y* $\lambda_3 = P(u,f(x))$. *Le aplicamos el algoritmo de unificación.*

D	τ	W_i	σ	$\lvert W_i\rvert$
		$\{P(x,y),\ P(f(z),x),\ P(u,f(x))\}$	Id	3
$\{x,f(z)\}$	$(x\lvert f(z))$	$\{P(f(z),y),\ P(f(z),f(z)),\ P(u,f(f(z)))\}$	$(x\lvert f(z))$	3
$\{y,f(z)\}$	$(y\lvert f(z))$	$\{P(f(z),f(z)),\ P(u,f(f(z)))\}$	$(y\lvert f(z))\,(x\lvert f(z))$	2
$\{f(z),u\}$	$(u\lvert f(z))$	$\{P(f(z),f(z)),\ P(f(z),f(f(z)))\}$	$(u\lvert f(z))\,(y\lvert f(z))\,(x\lvert f(z))$	2
$\{z,f(z)\}$				

Y como el último conjunto de discordancia es de Tipo 2, los literales no son unificables.

3. *Tomamos* $\lambda_1 = Q(x,g(y))$, $\lambda_2 = Q(f(y),g(f(b)))$. *Vemos si son unificables:*

D	τ	W_i	σ	$\lvert W_i\rvert$
		$\{Q(x,g(f(b))),\ Q(f(y),g(y))\}$	Id	2
$\{x,f(y)\}$	$(x\lvert f(y))$	$\{Q(f(y),g(f(b))),\ Q(f(y),g(y))\}$	$(x\lvert f(y))$	2
$\{f(b),y\}$	$(y\lvert f(b))$	$\{Q(f(f(b)),g(f(b)))\}$	$(y\lvert f(b))\,(x\lvert f(y))$	1

Y por tanto son unificables. Un unificador principal es $\sigma = (y|f(b))\,(x|f(y))$.

Notemos que $\sigma = (x|f(f(b));\ y|f(b))$ *y no* $(y|f(b);\ x|f(y))$.

Sistemas de ecuaciones en términos

A continuación vamos a dar otro método para decidir cuando un conjunto de fórmulas atómicas es o no unificable.

Supongamos que tenemos dos fórmulas atómicas α y β, que queremos ver si son o no unificables. Supongamos que $\alpha = R(t_1,t_2,\cdots,t_n)$ y $\beta = R(t'_1,t'_2,\cdots,t'_n)$. Obviamente, si las fórmulas empiezan por diferentes símbolos de predicado, entonces no son unificables.

Lo que tratamos es de encontrar una sustitución σ de forma que $\sigma(\alpha) = \sigma(\beta)$. En definitiva, lo que buscamos es una sustitución que transforme el término t_1 en lo mismo que el término t'_1; el término t_2 en lo mismo que el término t'_2, y así hasta el término t_n que debe ser transformado en lo mismo que el término t'_n.

Lo que tenemos podemos representarlo como n ecuaciones en términos como sigue:

$$\begin{array}{ccc} t_1 & = & t'_1 \\ t_2 & = & t'_2 \\ \cdots & \cdots & \cdots \\ t_n & = & t'_n \end{array}$$

Cada una de las expresiones $t_i = t'_i$ es una ecuación del sistema e_i. Un sistema de ecuaciones es por tanto un conjunto de ecuaciones $E = \{e_1, e_2, \cdots, e_n\}$

Una solución del sistema es una sustitución σ que transforme cada término t_i en lo mismo que el término t'_i.

Si un sistema tiene solución, entonces existe una sustitución σ de forma que cualquier solución del sistema τ se puede expresar como $\tau = \tau_1\sigma$ para alguna sustitución τ_1. Dicha solución se llamará *solución principal*

Dos sistemas de ecuaciones en términos son equivalentes si tienen las mismas soluciones.

Vamos a dar un método que permita, dado un sistema de ecuaciones en términos, decidir si tiene o no solución, y caso de tenerla, encontrarlas todas. Este método se basa en varios resultados que enunciamos a continuación. Por ser todos ellos muy intuitivos, los dejamos sin demostración.

Comenzamos con una definición.

Definición 47. *Dado un sistema de ecuaciones en términos $E = \{e_1, e_2, \cdots, e_n\}$, se dice que está en forma resuelta si:*

- *Cada ecuación del sistema es de la forma $x_i = t_i$, con x_i una variable.*
- *Las variables $x_1, x_2, \cdots, x_n$ son todas distintas.*
- *En los términos $t_1, t_2, \cdots, t_n$ no hay ninguna ocurrencia de las variables $x_1, x_2, \cdots, x_n$.*

En tal caso, $\sigma_E = (x_1|t_1;\ x_2|t_2;\ \cdots\ ; x_n|t_n)$ es una solución principal del sistema.

Vamos entonces a ver como transformar (si es posible) un sistema en otro sistema equivalente y que esté en forma resuelta. Para esto, vamos a ver algunos resultados que, bien permiten transformar un sistema en otro equivalente, bien concluir que el sistema no tiene solución.

1. Los sistemas $E \cup \{t = t\}$ y E son equivalentes. Es decir, podemos eliminar las ecuaciones en las que el término de la derecha y el de la izquierda coinciden.

2. Los sistemas $E \cup \{t_1 = t_2\}$ y $E \cup \{t_2 = t_1\}$ son equivalentes.

3. Los sistemas

$$E \cup \{f(t_1, \cdots, t_m) = f(t'_1, \cdots, t'_m)\} \text{ y } E \cup \{t_1 = t'_1, \cdots, t_m = t'_m\}$$

   son equivalentes.

4. Los sistemas $E \cup \{x = t\}$, donde x es una variable que no aparecen en el término t, y $\sigma(E) \cup \{x = t\}$ son equivalentes, donde σ es la sustitución $\sigma = (x|t)$.

   En este caso, $\sigma(E)$ es el sistema formado por las ecuaciones que resultan de realizar la sustitución σ en cada uno de los términos que intervienen en el sistema E.

5. Un sistema de la forma $E \cup \{x = t\}$ donde t es un término en el que interviene la variable x, y $t \neq x$ no tiene solución.

6. Un sistema de la forma $E \cup \{f(t_1, \cdots, t_n) = g(t'_1, \cdots, t'_m)\}$, con f y g símbolos de función distintos, no tiene solución.

7. Un sistema de la forma $E \cup \{f(t_1, \cdots, t_n) = a\}$, o de la forma $E \cup \{a = b\}$, donde a y b son constantes distintas, no tiene solución.

Utilizando los resultados que acabamos de enunciar, podemos, dado un sistema de ecuaciones en términos, bien transformarlo en un sistema en forma resuelta, bien concluir que no tiene solución. Veamos algunos ejemplos:

Ejemplo 4.1.8.

1. *Vamos a resolver el sistema:*

$$\left.\begin{array}{lcl} a & = & z \\ x & = & f(z) \\ f(g(y)) & = & f(u) \end{array}\right\}$$

El resultado 2 *permite transformarlo en*

$$\left.\begin{array}{lcl} z & = & a \\ x & = & f(z) \\ f(g(y)) & = & f(u) \end{array}\right\}$$

Ahora, por el resultado cuatro, y con la sustitución $(z|a)$ *transformamos el sistema en*

$$\left.\begin{array}{lcl} z & = & a \\ x & = & f(a) \\ f(g(y)) & = & f(u) \end{array}\right\}$$

Por el apartado 3 *llegamos a*

$$\left.\begin{array}{lcl} z & = & a \\ x & = & f(a) \\ g(y) & = & u \end{array}\right\}$$

Y por último, con el apartado segundo transformamos el sistema en

$$\left.\begin{array}{lcl} z & = & a \\ x & = & f(a) \\ u & = & g(y) \end{array}\right\}$$

que es un sistema en forma resuelta.

2. *Vamos a estudiar si el conjunto de fórmulas*

$$\{P(x,y),\ P(f(z),x),\ P(u,f(x))\}$$

es unificable o no.

Para esto, planteamos el sistema de ecuaciones en términos y tratamos de resolverlo. En negrita representamos la ecuación que vamos a utilizar en el paso siguiente.

$$\left.\begin{array}{lcl} \mathbf{x} & = & \mathbf{f(z)} \\ y & = & x \\ x & = & u \\ y & = & f(x) \end{array}\right\} \overset{(4)}{\Longrightarrow} \left.\begin{array}{lcl} x & = & f(z) \\ y & = & f(z) \\ \mathbf{f(z)} & = & \mathbf{u} \\ y & = & f(f(z)) \end{array}\right\} \overset{(2)}{\Longrightarrow} \left.\begin{array}{lcl} x & = & f(z) \\ \mathbf{y} & = & \mathbf{f(z)} \\ u & = & f(z) \\ y & = & f(f(z)) \end{array}\right\} \overset{(4)}{\Longrightarrow}$$

$$\left.\begin{array}{lcl} x & = & f(z) \\ y & = & f(z) \\ u & = & f(z) \\ \mathbf{f(z)} & = & \mathbf{f(f(z))} \end{array}\right\} \overset{(3)}{\Longrightarrow} \left.\begin{array}{lcl} x & = & f(z) \\ y & = & f(z) \\ u & = & f(z) \\ z & = & f(z) \end{array}\right\}$$

y al quedarmos al final la ecuación $z = f(z)$ *concluimos que el conjunto no es unificable (resultado 5).*

3. *Vamos por último a comprobar si el conjunto de dos fórmulas*

$$\{P(x,g(x),y,h(x,y),z,k(x,y,z));\ P(u,v,e(v),w,f(v,w),t)\}$$

es unificable o no. Para ello, vamos a plantear el sistema de ecuaciones en términos y vamos a tratar de llevarlo a forma resuelta.

$$\left\{\begin{array}{lcl} x & = & u \\ g(x) & = & v \\ y & = & e(v) \\ h(x,y) & = & w \\ z & = & f(v,w) \\ k(x,y,z) & = & t \end{array}\right. \xleftarrow{(4)} \qquad \left\{\begin{array}{lcl} x & = & u \\ g(u) & = & v \\ y & = & e(v) \\ h(u,y) & = & w \\ z & = & f(v,w) \\ k(u,y,z) & = & t \end{array}\right. \begin{array}{l} \\ \xleftarrow{(2)} \\ \\ \xleftarrow{(2)} \\ \\ \xleftarrow{(2)} \end{array}$$

$$\left\{\begin{array}{lcl} x & = & u \\ v & = & g(u) \\ y & = & e(v) \\ w & = & h(u,y) \\ z & = & f(v,w) \\ t & = & k(u,y,z) \end{array}\right. \xleftarrow{(4)} \qquad \left\{\begin{array}{lcl} x & = & u \\ v & = & g(u) \\ y & = & e(g(u)) \\ w & = & h(u,y) \\ z & = & f(g(u),w) \\ t & = & k(u,y,z) \end{array}\right. \xleftarrow{(4)}$$

$$\left\{\begin{array}{lcl} x & = & u \\ v & = & g(u) \\ y & = & e(g(u)) \\ w & = & h(u,e(g(u))) \\ z & = & f(g(u),w) \\ t & = & k(u,e(g(u)),z) \end{array}\right. \xleftarrow{(4)} \qquad \left\{\begin{array}{lcl} x & = & u \\ v & = & g(u) \\ y & = & e(g(u)) \\ w & = & h(u,e(g(u))) \\ z & = & f(g(u),h(u,e(g(u)))) \\ t & = & k(u,e(g(u)),z) \end{array}\right. \xleftarrow{(4)}$$

$$\left\{\begin{array}{lcl} x & = & u \\ v & = & g(u) \\ y & = & e(g(u)) \\ w & = & h(u,e(g(u))) \\ z & = & f(g(u),h(u,e(g(u)))) \\ t & = & k(u,e(g(u)),f(g(u),h(u,e(g(u))))) \end{array}\right.$$

Y hemos llegado a un sistema de ecuaciones, equivalente al de partida, y que está en forma resuelta.

4.2. Resolución

Como ya hemos comentado anteriormente, nuestro objetivo es, dado un conjunto de cláusulas que sea insatisfacible poder deducir la cláusula vacía. El punto de partida es el método de resolución que estudiamos en el tema dedicado a la lógica proposicional. Pero vimos que la aparición de variables hace que ese método haya que adaptarlo al contexto que tenemos.

El hecho de que en una cláusula las variables están cuantificadas universalmente significa que pueden tomar cualquier valor, es decir, podemos sustituirlas por cualquier término. En la última sección hemos estudiado cómo encontrar la sustituición adecuada para poder igualar dos literales.

Lo primero que necesitamos es trasladar el concepto de resolvente que teníamos en lenguaje proposicional.

Dadas dos cláusulas C_1 y C_2, una resolvente suya será una nueva cláusula C_3 que se deduzca de las dos dadas, es decir, que se verifique que $\{C_1, C_2\} \models C_3$.

4.2.1. Resolventes

Resolventes binarias

Antes de definir el concepto vamos a analizar algunos ejemplos sencillos.

Ejemplo 4.2.1.

Supongamos que tenemos las cláusulas $C_1 = \neg P(x) \vee Q(b)$ y $C_2 = P(a)$ y que ambas son ciertas.

La primera cláusula la podemos escribir como $\forall x(P(x) \to Q(b))$, es decir, que sea quien sea x la fórmula $P(x) \to Q(b)$ es cierta. En particular, será cierto $P(a) \to Q(b)$ (hemos sustituido la variable x

por el término a). Como además $P(a)$ es cierto podemos deducir que $Q(b)$ es cierto. Es decir, tenemos que

$$\{\neg P(x) \vee Q(b), P(a)\} \vDash Q(b)$$

Otra forma de llegar a la misma conclusión podría ser:

1. *Tomamos las cláusulas C_1 y C_2.*
2. *Elegimos los literales $L_1 = \neg P(x)$ y $L_2 = P(a)$ de la primera y segunda cláusulas respectivamente.*
3. *Comprobamos si L_1^C y L_2 son unificables. En este caso lo son, y un unificador principal es $\sigma = (x|a)$.*
4. *Hallamos $\sigma(C_1) = \neg P(a) \vee Q(b)$ y $\sigma(C_2) = P(a)$*
5. *Eliminamos de $\sigma(C_1)$ y $\sigma(C_2)$ los literales $\sigma(L_1)$ y $\sigma(L_2)$, y formamos una cláusula con los literales restantes (aquí procedemos de igual forma a como hacíamos en el tema segundo cuando calculábamos resolventes). La cláusula resultante es consecuencia lógica de las dos primeras.*

Vamos a repetir este proceso con las cláusulas

$$C_1 = P(x,b) \vee Q(x,a); \qquad C_2 = \neg P(a,z) \vee R(z)$$

1. *Tomamos el literal $L_1 = P(x,b)$ de la primera cláusula, y el literal $L_2 = \neg P(a,z)$ de la segunda.*
2. *Buscamos un unificador principal para L_1^C y L_2. Este unificador existe, y es $\sigma = (x|a;\ z|b)$.*
3. *Calculamos $\sigma(C_1) = P(a,b) \vee Q(a,a)$ y $\sigma(C_2) = \neg P(a,b) \vee R(b)$.*
4. *Eliminamos los literales $\sigma(L_1)$ y $\sigma(L_2)$ de las cláusulas $\sigma(C_1)$ y $\sigma(C_2)$. Nos queda la cláusula $C_3 = Q(a,a) \vee R(b)$*

Vamos a ver que $\{C_1, C_2\} \vDash C_3$.
Suponemos que tenemos una estructura $I = (\mathcal{E}, v)$ tal que $I(C_1) = I(C_2) = 1$.
Entonces $I(\forall x(P(x,b) \vee Q(x,a))) = 1$. Significa esto que $P(x,b) \vee Q(x,a)$ será cierto sea cual sea el valor de x. En particular, $I(P(a,b) \vee Q(a,a)) = 1$.
De la misma forma, como $I(C_2) = 1$, entonces $I(\neg P(a,b) \vee R(b)) = 1$.
Pueden ocurrir ahora dos cosas:

- $I(P(a,b)) = 1$.

  En este caso $I(\neg P(a,b)) = 0$, luego $I(R(b)) = 1$ (ya que $I(\neg P(a,b) \vee R(b)) = 1$). Por tanto, $I(Q(a,a) \vee R(b)) = I(C_3) = 1$.

- $I(P(a,b)) = 0$.

  En este caso $I(Q(a,a))$ debe valer uno, luego $I(C_3) = 1$.

Luego en cualquiera de los casos deducimos que $I(C_3) = 1$, es decir,

$$\{C_1, C_2\} \vDash C_3.$$

Notemos que la cláusula C_2 es equivalente a $\forall x(\neg P(a,x) \vee R(x))$. Pero ahora, cuando intentamos buscar un unificador para $P(x,b)$ y $P(a,x)$ vemos que no existe, luego no podríamos seguir el proceso anterior. Hay que tener en cuenta que

> las variables de dos cláusulas distintas son distintas, aunque tengan el mismo nombre. Caso de que coincidan los nombres es conveniente cambiar el nombre de las variables de alguna de ellas.

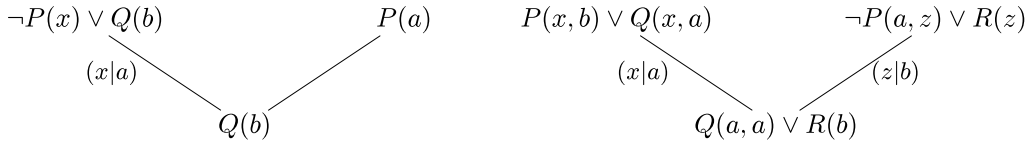
Definimos ya lo que es una resolvente binaria de dos cláusulas:

Definición 48. *Sean C_1 y C_2 dos cláusulas que no tienen variables comunes. Supongamos que $C_i = L_i \vee C_i'$, donde L_i, $i = 1, 2$ es un literal y $C_i, i = 1, 2$ es una cláusula (que podría ser la vacía), y que los literales L_1 y L_2^C tienen un unificador principal σ.*

Entonces la cláusula $\sigma(C_1') \vee \sigma(C_2')$ es una resolvente binaria de C_1 y C_2.

Por ejemplo, $Q(b)$ es una resolvente binaria de las cláusulas $\neg P(x) \vee Q(b)$ y $P(a)$, mientras que $Q(a, a) \vee R(b)$ es una resolvente binaria de $P(x, b) \vee Q(x, a)$ y $\neg P(a, z) \vee R(z)$.

Normalmente, para indicar que una cláusula se obtiene como resolvente binaria de otras dos usaremos la notación siguiente:

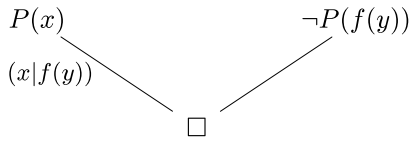


Una vez visto cómo obtener una resolvente, podemos afirmar que si de un conjunto de cláusulas podemos obtener la cláusula vacía haciendo resolventes, entonces el conjunto es insatisfacible.

Ejemplo 4.2.2.

1. *Vamos a comprobar que el conjunto de cláusulas $\{P(x), \neg P(f(x))\}$ es insatisfacible.*

   En principio podría parecer que no es posible obtener ninguna resolvente, pues $P(x)$ y $P(f(x))$ no son unificables. Sin embargo, hemos dicho que cuando tengamos variables que aparecen en dos cláusulas, podemos cambiar las de una de ellas. En este caso nos quedaría el conjunto de cláusulas como $\{P(x), \neg P(f(y))\}$, y ahora si podemos hacer resolventes:



2. *Tomamos el conjunto $\{P(x) \vee P(y), \neg P(z) \vee \neg P(u)\}$. Este conjunto también es insatisfacible. Sin embargo, si hacemos resolventes binarias nunca vamos a poder obtener la cláusula vacía.*

Resolventes

El último ejemplo nos muestra que el concepto de resolvente binaria no es suficiente para decidir si un conjunto de cláusulas es satisfacible o insatisfacible. Necesitamos por tanto extender el concepto de resolvente para que podamos contemplar casos como este (y otros algo más complicados).

Antes de definir una resolvente, necesitamos previamente el concepto de factor.

Definición 49. *Sea C una cláusula. Supongamos que en C hay dos o más literales que son unificables, y sea σ un unificador. En ese caso, diremos que $\sigma(C)$ es un* factor *de C.*

En el caso de que $\sigma(C)$ sea un factor de C se verifica que $C \vDash \sigma(C)$.

Ejemplo 4.2.3.
Sea $C = P(x) \vee P(f(a)) \vee Q(x, b)$ entonces $P(f(a)) \vee Q(f(a), b)$ es un factor de C (hemos unificado los dos primeros literales con la sustitución $(x|f(a))$).

Notemos que si queremos hallar un factor de una cláusula, y en dos o más literales tenemos variables repetidas, para obtener un unificador no podemos renombrar las variables (salvo que a todas las apariciones de la variable en la cláusula le demos el mismo nombre).

Es decir, de la cláusula $P(x) \vee P(f(x)) \vee Q(y, b)$ no podemos obtener ningún factor, ya que los literales $P(x)$ y $P(f(x))$ no son unificables.

Definición 50. *Sean C_1 y C_2 dos cláusulas. Se dice que C es una* resolvente *de C_1 y C_2 si C responde a alguna de las cuatro posibilidades siguientes:*

1. *C es una resolvente binaria de C_1 y C_2.*
2. *C es una resolvente binaria de C_1 y un factor de C_2.*
3. *C es una resolvente binaria de un factor de C_1 y de C_2.*
4. *C es una resolvente binaria de un factor de C_1 y un factor de C_2.*

Notemos que si C es una resolvente de C_1 y C_2 entonces $\{C_1, C_2\} \vDash C$.

Ejemplo 4.2.4.

1. *Consideramos las cláusulas $C_1 = P(x) \vee P(y)$ y $C_2 = \neg P(z) \vee \neg P(u)$. Entonces $P(x)$ es un factor de C_1, $\neg P(z)$ es un factor de C_2, y $\square$ es una resolvente binaria de $P(x)$ y $\neg P(z)$. Por tanto, $\square$ es una resolvente de C_1 y C_2.*

   Vemos así que el conjunto $\{P(x) \vee P(y), \neg P(z) \vee \neg P(u)\}$ es insatisfacible.

2. *Vamos ahora a obtener varias resolventes de las cláusulas*

$$C_1 = Q(x) \vee \neg R(x) \vee P(x,y) \vee P(f(z), f(z))$$
$$C_2 = \neg S(u) \vee \neg R(w) \vee \neg P(f(a), f(a)) \vee \neg P(f(w), f(w))$$

$$\begin{array}{l} \qquad\qquad\qquad\qquad \neg S(u) \vee \neg R(w) \vee \neg P(f(a), f(a)) \vee \neg P(f(w), f(w)) \\ Q(x) \vee \neg R(x) \vee P(x,y) \vee P(f(z), f(z)) \\ \qquad\quad {\scriptstyle (x \mid f(a);\, y \mid f(a))} \\ Q(f(a)) \vee \neg R(f(a)) \vee P(f(z), f(z)) \vee \neg S(u) \vee \neg R(w) \vee \neg P(f(w), f(w)) \end{array}$$

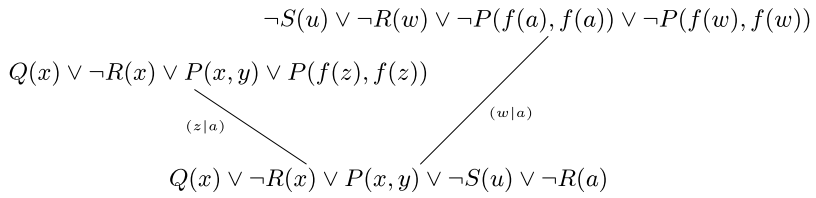
En este caso se ha obtenido una resovente binaria de C_1 y C_2.

$$\begin{array}{l} \qquad\qquad\qquad\qquad \neg S(u) \vee \neg R(w) \vee \neg P(f(a), f(a)) \vee \neg P(f(w), f(w)) \\ Q(x) \vee \neg R(x) \vee P(x,y) \vee P(f(z), f(z)) \\ \qquad\qquad\qquad\qquad\qquad\quad {\scriptstyle (w \mid z)} \\ Q(x) \vee \neg R(x) \vee P(x,y) \vee \neg S(u) \vee \neg R(z) \vee \neg P(f(a), f(a)) \end{array}$$

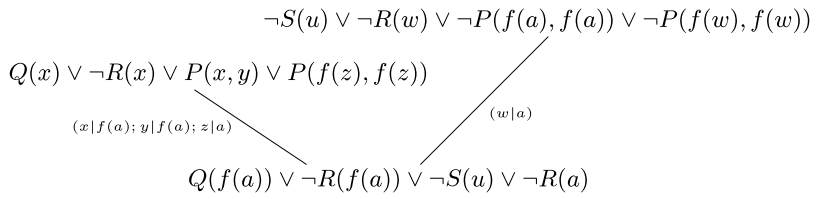
En este caso, también hemos obtenido una resolvente binaria de C_1 y C_2. Podemos conseguir dos más.

$$\begin{array}{l} \qquad\qquad\qquad\qquad \neg S(u) \vee \neg R(w) \vee \neg P(f(a), f(a)) \vee \neg P(f(w), f(w)) \\ Q(x) \vee \neg R(x) \vee P(x,y) \vee P(f(z), f(z)) \\ \qquad\quad {\scriptstyle (x \mid f(z);\, y \mid f(z))} \qquad\qquad {\scriptstyle (w \mid z)} \\ Q(f(z)) \vee \neg R(f(z)) \vee \neg S(u) \vee \neg R(z) \vee \neg P(f(a), f(a)) \end{array}$$

Aquí hemos hecho una resolvente binaria de un factor de C_1 y C_2. Podríamos hacer otra más.



Aquí, una resolvente binaria de C_1 y un factor de C_2. Se puede hacer otra más.



Por último, lo que hemos hecho es una resolvente binaria de un factor de C_1 y un factor de C_2.

4.2.2. Deducciones y principio de resolución.

Sea Γ un conjunto de cláusulas, y C una cláusula. Una *deducción* de C a partir de Γ es una sucesión finita de cláusulas $C_1, C_2, \cdots C_n$ donde $C_n = C$, y para $i < n$, C_i es una cláusula, que bien es un elemento de Γ, bien es una resolvente de dos cláusulas que la preceden.

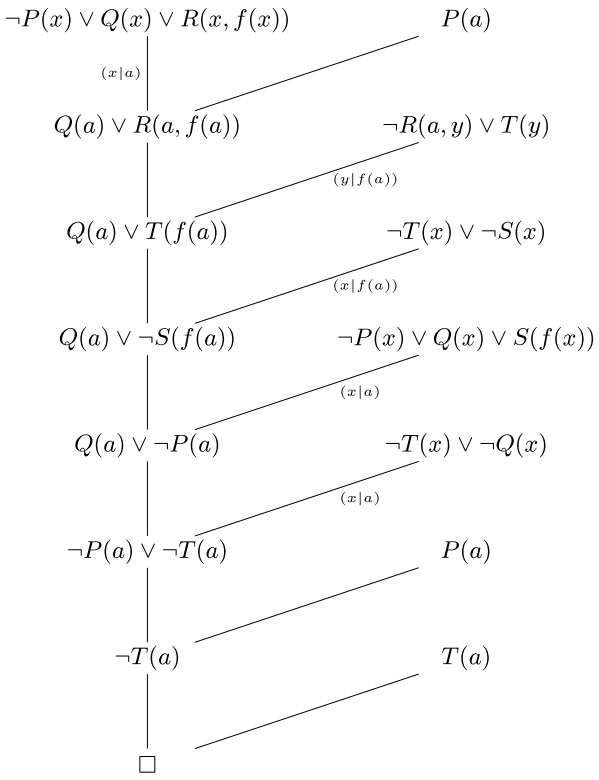
Ejemplo 4.2.5. *En el ejemplo 3.6.17 estuvimos viendo que para comprobar si*

$$\left\{ \begin{array}{l} \forall x(P(x) \wedge \neg Q(x) \rightarrow \exists y(R(x,y) \wedge S(y))) \\ \exists y \forall x((R(y,x) \rightarrow T(x)) \wedge T(y) \wedge P(y) \end{array} \right\} \models \exists x(T(x) \wedge (Q(x) \vee S(x)))$$

bastaba con comprobar si el conjunto de cláusulas

$$\Gamma = \left\{ \begin{array}{lll} \neg P(x) \vee Q(x) \vee R(x, f(x)); & \neg R(a,y) \vee T(y); & T(a); \quad P(a) \\ \neg P(x) \vee Q(x) \vee S(f(x)); & \neg T(x) \vee \neg Q(x); & \neg T(x) \vee \neg S(x) \end{array} \right\}$$

es insatisfacible. Ahora vamos a ver que este conjunto es insatisfacible, para lo cual vamos a dar una deducción de la cláusula vacía.



Si consideramos las cláusulas:

$$C_1 = \neg P(x) \vee Q(x) \vee R(x, f(x)); \; C_2 = P(a); \; C_3 = Q(a) \vee R(a, f(a)); \; C_4 = \neg R(a, y) \vee T(y);$$
$$C_5 = Q(a) \vee T(f(a)); \; C_6 = \neg T(x) \vee \neg S(x); \; C_7 = Q(a) \vee \neg S(f(a)); \; C_8 = \neg P(x) \vee Q(x) \vee S(f(x));$$
$$C_9 = Q(a) \vee \neg P(a); \; C_{10} = \neg T(x) \vee \neg Q(x); \; C_{11} = \neg P(a) \vee \neg T(a); \; C_{12} = \neg T(a); \; C_{13} = T(a); \; C_{14} = \square$$

entonces:

1. C_1 *y* C_2 *son elementos de* Γ.
2. C_3 *es resolvente de* C_1 *y* C_2.
3. C_4 *es un elemento de* Γ.
4. C_5 *es una resolvente de* C_3 *y* C_4.
5. C_6 *es un elemento de* Γ.
6. C_7 *es una resolvente de* C_5 *y* C_6.
7. C_8 *es un elemento de* Γ.
8. C_9 *es una resolvente de* C_7 *y* C_8.
9. C_{10} *es un elemento de* Γ.
10. C_{11} *es una resolvente de* C_9 *y* C_{10}.
11. C_{12} *es una resolvente de* C_2 *y* C_{11}.
12. C_{13} *es un elemento de* Γ.
13. C_{14} *es una resolvente de* C_{12} *y* C_{13}.

Por tanto $C_{14} = \square$ *se ha deducido a partir de* Γ.

Una vez visto esto podemos dar el teorema principal de este capítulo:

Teorema 4.2.1. *(Completitud del principio de resolución)*

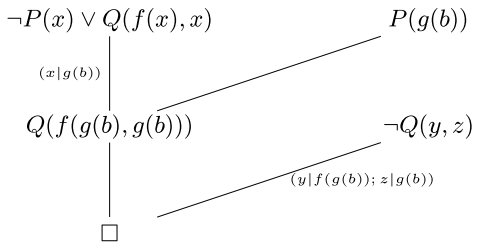
Sea Γ *un conjunto de cláusulas. Entonces* Γ *es insatisfacible si, y sólo si, existe una deducción de* $\square$ *a partir de* Γ.

Ejemplo 4.2.6.

1. *Vamos a comprobar que el conjunto de cláusulas*

$$\Gamma = \{\neg P(x) \vee Q(f(x), x), \quad P(g(b)), \quad \neg Q(y, z)\}$$

es insatisfacible. Para esto, vamos a obtener una deducción de la cláusula vacía.



2. *Vamos a comprobar que*

$$\{\forall x Q(x) \vee \exists y P(y),\ \neg \forall y Q(y)\} \vDash \exists x P(x)$$

Demostrar eso es lo mismo que probar que el conjunto de fórmulas

$$\{\forall x Q(x) \vee \exists y P(y),\ \neg \forall y Q(y),\ \neg \exists x P(x)\}$$

es insatisfacible.

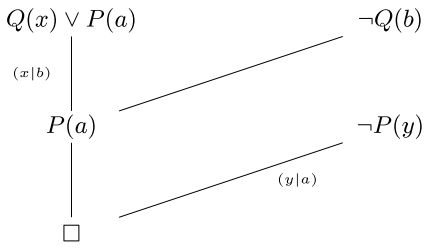
Para esto, hallamos la forma clausular de cada una de las fórmulas:

$$\left|\begin{array}{c} \forall x Q(x) \vee \exists y P(y) \\ \exists y \forall x (Q(x) \vee P(y)) \\ \forall x (Q(x) \vee P(a)) \end{array}\right| \left|\begin{array}{c} \neg \forall y Q(y) \\ \exists y \neg Q(y) \\ \neg Q(b) \end{array}\right| \left|\begin{array}{c} \neg \exists x P(x) \\ \forall x \neg P(x) \\ \forall y \neg P(y) \end{array}\right|$$

Luego debemos ver que el conjunto de cláusulas

$$\{Q(x) \vee P(a),\ \neg Q(b),\ \neg P(y)\}$$

es insatisfacible.



4.2.3. Por qué no utilizar el algoritmo de Davis-Putnam

En lo que sigue, y hasta la siguiente sección, se va a tratar de dar una justificiación de porqué no puede usarse en este contexto el algoritmo de Davis-Putnam. Salvo para algunos comentarios, el contenido de la sección no se usará más adelante, así que puede saltarse sin perder la continuidad.

Cuando estudiamos la lógica proposicional vimos cuatro métodos para resolver el problema de la implicación semántica: tablas de verdad, ecuaciones en $\mathbb{Z}_2$, algoritmo de Davis-Putnam y resolución.

Las tablas de verdad no podemos trasladarlas a la lógica de predicados, pues dado un conjunto de fórmulas no podemos calcular todas las posibles interpretaciones.

Tampoco podemos plantear ecuaciones en $\mathbb{Z}_2$, ya que cuando tenemos alguna fórmula de la forma $\forall x\alpha$ o $\exists x\alpha$ no tenemos una expresión sencilla del valor de verdad de esa fórmula en función del valor de verdad de α.

Vamos a explicar aquí que pasaría con el algoritmo de Davis-Putnam. Para eso, vamos a analizar algunos ejemplos:

1. Comenzamos con $\forall x P(x) \vDash P(a)$.

   Sabemos que comprobar eso es lo mismo que comprobar que el conjunto de cláusulas $\{P(x),\ \neg P(a)\}$ es insatisfacible.

   Puesto que la cláusula $P(x)$ es en realidad $\forall x P(x)$, lo que vamos a hacer es sustituir x por todos los posibles términos que tenemos en el lenguaje (que no contengan símbolos de variable). En nuestro caso, el único término es a.

   Tendríamos entonces el conjunto $\{P(a),\ \neg P(a)\}$. A este conjunto le podemos aplicar el algoritmo de Davis-Putnman y vemos que es insatisfacible.

2. Supongamos ahora que nuestro conjunto de cláusulas es $\{Q(x) \vee P(a),\ \neg Q(b),\ \neg P(y)\}$. En este caso, nuestro lenguaje tiene dos términos (aparte de los símbolos de variable) a y b. Si sustituimos las variables que tenemos por estos términos lo que tenemos es:

   $$\{Q(a) \vee P(a),\ Q(b) \vee P(a),\ \neg Q(b),\ \neg P(a),\ \neg P(b)\}$$

   Aplicamos el algoritmo de Davis-Putnam a este conjunto:

   $$\begin{array}{c} \{Q(a) \vee P(a),\ Q(b) \vee P(a),\ \neg Q(b),\ \neg P(a),\ \neg P(b)\} \\ \Big| \quad \lambda = \neg P(a) \\ \{Q(a),\ Q(b),\ \neg Q(b),\ \neg P(b)\} \\ \Big| \quad \lambda = Q(b) \\ \{Q(a),\ \square,\ \neg P(b)\} \end{array}$$

A la vista de estos ejemplos podría pensarse en utilizar el algoritmo de Davis-Putnam a la lógica de predicados. Dado un conjunto de cláusulas Σ, los pasos a seguir serían:

- Tomamos el conjunto formado por todos los términos del lenguaje en los que no aparecen símbolos de variable. Llamemos H a este conjunto (este conjunto se conoce como el Universo de Herbrand. En los ejemplos anteriores sería $\{a\}$ para el primero, y $\{a, b\}$ para el segundo).
- En cada una de las cláusulas de Σ sustituimos las variables por los términos que tenemos en H. Esto nos da un conjunto que llamaremos S (a este conjunto se le conoce como el sistema de Herbrand, y cada uno de sus elementos se dice que es una instancia básica).
- Estudiamos por el algoritmo de Davis-Putnam si este conjunto es satisfacible o insatisfacible.

Esto es lo que hemos hecho en los dos ejemplos anteriores. Pero vamos a ve que ocurre en el siguiente caso.

3. Supongamos que queremos ver si $\{\forall x(P(x) \to P(f(x)));\ P(a)\} \vDash \exists x P(f(f(x)))$.

   Transformamos el problema en ver si el siguiente conjunto de cláusulas

   $$\{\neg P(x) \vee P(f(x));\ P(a); \neg P(f(f(x)))\}$$

   es insatisfacible.

   Y ahora, el conjunto de todos los términos es infinito. De hecho, este conjunto (universo de Herbrand) es

   $$H = \{a,\ f(a),\ f(f(a)),\ f(f(f(a))),\ \cdots\}$$

   Y por tanto, el conjunto de todas las cláusulas (sistema de Herbrand) es también infinito.

   $$S = \{\neg P(a) \vee P(f(a));\ \neg P(f(a)) \vee P(f(f(a)));\ \neg P(f(f(a))) \vee P(f(f(f(a))));\cdots;\ P(a);\ \neg P(f(f(a)));\ \neg P(f(f(f(a))));\ \cdots\}$$

   Si aplicáramos ahora el "algoritmo" de Davis-Putnam a este conjunto, tendríamos:

   $$\{\neg P(a) \vee P(f(a));\ \neg P(f(a)) \vee P(f(f(a)));\ \neg P(f(f(a))) \vee P(f(f(f(a))));\cdots;\ P(a);\ \neg P(f(f(a)));\ \neg P(f(f(f(a))));\ \cdots\}$$
   $$\Big|\ \lambda = P(a)$$
   $$\{P(f(a));\ \neg P(f(a)) \vee P(f(f(a)));\ \neg P(f(f(a))) \vee P(f(f(f(a))));\cdots;\ \neg P(f(f(a)));\ \neg P(f(f(f(a))));\ \cdots\}$$
   $$\Big|\ \lambda = P(f(a))$$
   $$\{P(f(f(a)));\ \neg P(f(f(a))) \vee P(f(f(f(a))));\cdots;\ \neg P(f(f(a)));\ \neg P(f(f(f(a))));\ \cdots\}$$
   $$\Big|\ \lambda = P(f(f(a)))$$
   $$\{P(f(f(f(a))));\cdots;\ \square;\ \neg P(f(f(f(a))));\ \cdots\}$$

   Y al obtener un conjunto que contiene a la cláusula vacía, el conjunto es insatisfacible.

   Para llegar a esta conclusión no habia sido necesario trabajar con todo el conjunto S. Si hubiéramos tomado el siguiente subconjunto suyo:

   $$\{\neg P(a) \vee P(f(a));\ \neg P(f(a)) \vee P(f(f(a))); P(a);\ \neg P(f(f(a)))\}$$

   y le hubiéramos aplicado el algoritmo de Davis-Putnam llegaríamos también a la cláusula vacía.

El problema que encotramos cuando queremos trasladar el algoritmo de Davis-Putnam a la lógica de predicados es que el conjunto al que habría que aplicarselo es al conjunto de todas las instancias básicas (*el sistema de Herbrand*), y este conjunto, cuando en el conjunto de cláusulas haya algún símbolo de función, es infinito (en tal caso, ya no podríamos hablar de algoritmo).

Hay un teorema (*teorema de Herbrand*) que afirma que un conjunto de cláusulas es insatisfacible si, y sólo si, existe un conjunto finito de instancias básicas que es insatisfacible. El problema es que, a priori, a partir del sistema de Herbrand no sabemos cuál puede ser ese conjunto finito.

Una forma de encontrarlo es mediante resolución. Por ejemplo, si tenemos el conjunto de cláusulas $\{\neg P(x) \vee P(f(x));\ P(a); \neg P(f(f(x)))\}$, podemos hacer:

$$\begin{array}{ll} P(a) & \neg P(x) \vee P(f(x)) \\ \Big| & \\ P(f(a)) & \neg P(x) \vee P(f(x)) \\ \Big| & \\ P(f(f(a))) & \neg P(f(f(x))) \\ \Big| & \\ \square & \end{array}$$

Y de aquí podemos extraer el conjunto finito de instancias básicas que es insatisfacible.

Tenemos por una parte $P(a)$ (pues es la cláusula que hemos usado al principio), $\neg P(a) \vee P(f(a))$ (que es la instancia básica que se obtiene de hacer la sustitución $(x|a)$ en la cláusula $\neg P(x) \vee P(f(x))$), $\neg P(f(a)) \vee P(f(f(a)))$ (que es la instancia básica que se obtiene de hacer la sustitución $(x|f(a))$ en la cláusula $\neg P(x) \vee P(f(x))$ y $\neg P(f(f(a)))$ (que es la instancia básica que se obtiene de hacer la sustitución $(x|a)$ en la cláusula $\neg P(f(f(x)))$).

Vemos que el método de resolución nos dice cómo encontrar el conjunto de instancias básicas que es insatisfacible. Pero si ya sabemos que es insatisfacible, ¿para qué vamos a aplicar el algoritmo de Davis-Putnam para ver que es insatisfacible)?.

En resumen, si quisiéramos aplicar el algoritmo de Davis-Putnam, en principio tendríamos un conjunto infinito de cláusulas al que aplicarselo (el sistema de Herbrand). Por el teorema de Herbrand, de ese conjunto podríamos extraer un subconjunto finito (caso de ser insatisfacible). Una forma de saber que conjunto extraer es mediante resolución. Pero en tal caso, ya no tiene sentido usar el algoritmo de Davis-Putnam.

4.3. Estrategias de gestión

Podría parecer que con el Teorema de Completitud del Principio de Resolución está resuelto el problema de la insatisfacibilidad de un conjunto de cláusulas, y por tanto el problema de la implicación semántica. Sin embargo, esto está muy lejos de ser cierto.

Sabemos que si un conjunto de cláusulas es insatisfacible, entonces existe una deducción (que contiene un número finito de pasos) de la cláusula vacía. Pero no existe ningún algoritmo que nos dé la respuesta a la pregunta de si un conjunto es o no insatisfacible, aunque, si es insatisfacible, hay algoritmos que nos conducen a la cláusula vacía en un número finito de pasos (otro tema es que eso puede ser computacionalmente muy costoso). La dificultad está en que si el conjunto de cláusulas es satisfacible no hay ningún algoritmo que nos dé la respuesta en un finito de pasos.

Lo que vamos a ver a continuación son distintas estrategias para, dado un conjunto de cláusulas, obtener, si es insatisfacible, una deducción de la cláusula vacía.

En el tema dedicado a la lógica proposicional ya dimos un avance de estas estrategias.

4.3.1. Estrategia de Saturación

Recordemos que el problema que tenemos en este momento es el de determinar si un conjunto de cláusulas es satisfacible o insatisfatisfacible. La búsqueda de una deducción de la cláusula vacía (también llamada una *refutación*) o llegar a la conclusión de que ésta no existe puede abordarse calculando todas las posibles resolventes a partir del conjunto de partida. Es la *estrategia de saturación*. El procedimiento puede ejecutarse de una forma algorítmica que describimos a continuación:

Llamamos $S_0 = S$.

Para cada i

> Calculamos S_{i+1} como el conjunto que se obtiene de S_i al añadir todas las resolventes que puedan calcularse usando cláusulas de S_i.
>
> Si S_{i+1} contiene a la cláusula vacía, entonces el conjunto de partida es *insatisfacible*;
>
> si $S_{i+1} = S_i$, entonces el conjunto de partida es *satisfacible*;
>
> en otro caso incrementar i y volver a ejecutar el bucle.

Este algoritmo nos proporciona una cadena de conjuntos de cláusulas, es decir, una secuencia de conjuntos en el que cada uno está contenido en el siguiente:

$$S_0 \subseteq S_1 \subseteq \cdots \subseteq S_i \subseteq S_{i+1} \subseteq \ldots$$

y de forma que todos tienen el mismo carácter de satisfacibilidad (es decir, uno es insatisfacible si, y sólo si, lo es otro cualquiera).

Cuando en uno de los eslabones aparece la cláusula vacía tenemos asegurada la insatisfacibilidad; cuando la cadena se estabiliza (es decir $S_i = S_{i+1}$, y por tanto todos los siguientes vuelven a ser iguales a S_i puesto que no aparecen nuevas resolventes) entonces ya se tiene que no es posible obtener una deducción de la cláusula vacía, ya que se han explorado todas las posibilidades.

Hay dos problemas que presenta este método: la imposibilidad de detener el proceso en un número finito de pasos y que el esfuerzo de cálculo sea inabordable.

Ejemplo 4.3.1.

1. *Consideremos el conjunto*

$$S = \{\neg P(x) \vee Q(f(x)),\ P(a),\ \neg P(y) \vee \neg Q(y)\}$$

Inicializamos $S_0 = S$ y calculamos las posibles resolventes entre sus cláusulas. Las numeramos para obtener una más fácil referencia:

$C_1 = \neg P(x) \vee Q(f(x))$

$C_2 = P(a)$

$C_3 = \neg P(x) \vee \neg Q(x)$

entonces podemos obtener las siguientes nuevas cláusulas:

$C_4 = R(C_1, C_2) = Q(f(a))$ *con la sustituición* $(x|a)$*;*

$C_5 = R(C_1, C_3) = \neg P(x) \vee \neg P(f(x))$ *con la sustitución* $(y|f(x))$*;*

$C_6 = R(C_2, C_3) = \neg Q(a)$ *con la sustitución* $(x|a)$*.*

El conjunto S_1 consta de las seis cláusulas que tenemos hasta el momento:

$$S_1 = \{\neg P(x) \vee Q(f(x)),\ P(a),\ \neg P(y) \vee \neg Q(y),\ Q(f(a)),\ \neg P(x) \vee \neg P(f(x)),\ \neg Q(a)\}$$

Como no contiene a la cláusula vacía ni coincide con el anterior. Proseguimos calculando S_2:

A partir de la C_1 no es posible calcular nuevas resolventes;

$C_7 = R(C_2, C_5) = R(C_3, C_4) = \neg P(f(a))$*;*

tampoco hay más resolventes entre el resto de cláusulas.

$$S_2 = \left\{ \begin{array}{llll} \neg P(x) \vee Q(f(x)); & P(a); & \neg P(y) \vee \neg Q(y); & Q(f(a)); \\ \neg P(x) \vee \neg P(f(x)); & \neg Q(a); & \neg P(f(a)) & \end{array} \right\}$$

De nuevo examinamos si aparece la cláusula vacía, lo que no ocurre, y $S_2 \neq S_1$, así que debemos continuar calculando S_3: como sólo hay una cláusula nueva entonces las nuevas resolventes tendrían que serlo de C_7, pero ésta no admite ninguna resolvente con el resto. Así que

$$S_3 = S_2$$

y el algoritmo en este caso acaba emitiendo la respuesta: el conjunto es satisfacible.

2. *Sea ahora $S = \{A(b),\ \neg M(y) \vee P(b, y),\ \neg P(x, z),\ M(a),\ C(a)\}$ y comencemos a ejecutar el algoritmo. Para eso, nombramos las cláusulas que pertenecen a S como C_1, C_2, C_3, C_4 y C_5.*

$C_6 = R(C_2, C_3) = \neg M(y)$

$C_7 = R(C_2, C_4) = P(b, a)$

Así que S_1 consta de las siete cláusulas descritas, es decir, es el conjunto

$$\{A(b),\ \neg M(y) \vee P(b,y),\ \neg P(x,z),\ M(a),\ C(a),\ \neg M(y),\ P(b,a)\}$$

que no contiene a la cláusula vacía y tampoco coincide con el anterior.

Calculamos entonces S_2 y encontramos $C_8 = R(C_4, C_6) = \square$ con lo que el algoritmo acaba con la respuesta el conjunto es insatisfacible.

Para reconstruir la deducción que nos lleva a la cláusula vacía recorremos el proceso en sentido inverso:

$\square = R(C_4, C_6)$

$C_6 = R(C_2, C_3)$

C_2 y C_3 son cláusulas del conjunto de partida.

y la deducción sería:

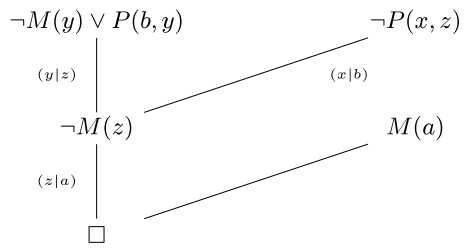
a) *C_2 está en S;*

b) *C_3 está en S;*

c) *C_6 es resolvente de las dos anteriores;*

d) *$\square$ es resolvente de dos anteriores.*

o dibujada en forma de árbol:



Los dos ejemplos anteriores son representativos de la situación habitual, cuando la cantidad de cláusulas que aparecen en los sucesivos conjuntos S_i pueden ser enormes. Se propone como ejercicio realizar los primeros pasos de la estrategia de saturación para el siguiente conjunto de cláusulas:

$$\Gamma = \{\neg E(x,y) \vee \neg E(x,z) \vee E(z,y),\ \neg E(u,v) \vee E(v,u),\ E(a,b),\ E(b,c),\ \neg E(a,c)\ \}$$

Por último, tomamos el conjunto de cláusulas

$$\Gamma = \{\neg P(x) \vee P(f(x));\ P(a)\}$$

Partimos de $S_0 = \Gamma$, y vamos construyendo los distintos conjuntos S_i:

$$S_1 = \{\neg P(x) \vee P(f(x));\ P(a);\ P(f(a))\}$$
$$S_2 = \{\neg P(x) \vee P(f(x));\ P(a);\ P(f(a));\ P(f(f(a)))\}$$
$$S_3 = \{\neg P(x) \vee P(f(x));\ P(a);\ P(f(a));\ P(f(f(a)));\ P(f(f(f(a))))\}$$

Y podemos ver como obtenemos una sucesión

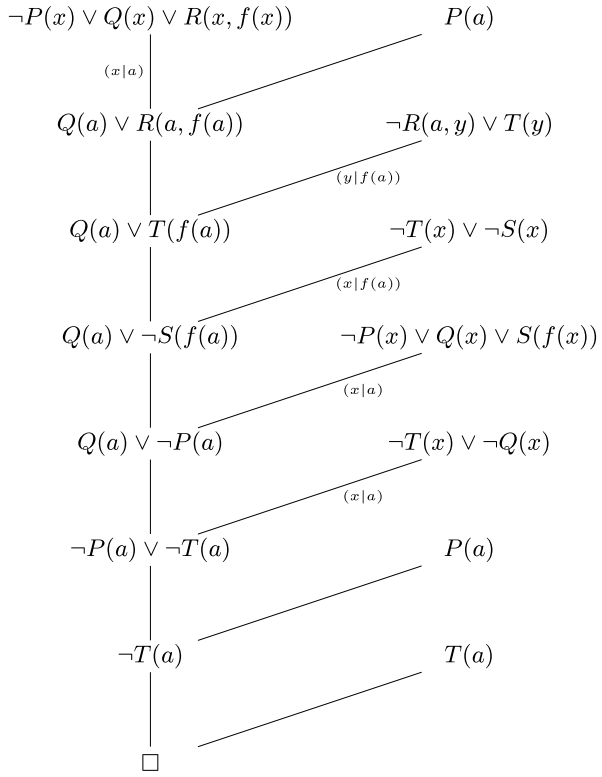
$$S_0 \subsetneq S_1 \subsetneq S_2 \subsetneq S_3 \subsetneq \cdots \subsetneq S_i \subsetneq S_{i+1} \subsetneq \cdots$$

y en la que nunca nos aparecerá la cláusula vacía.

Por tanto, el algoritmo en este caso no terminaría. Si en algún caso no llegamos a ninguna de las dos condiciones de parada, ¿cuándo terminamos?. Si no nos ha salido la cláusula vacía, ¿es porque no se puede obtener, o porque no hemos hecho las iteraciones necesarias para conseguirla?.

4.3.2. Deducciones lineales

Hasta ahora hemos escrito y dibujado varios ejemplos de deducciones. Si observamos el esquema de la deducción:



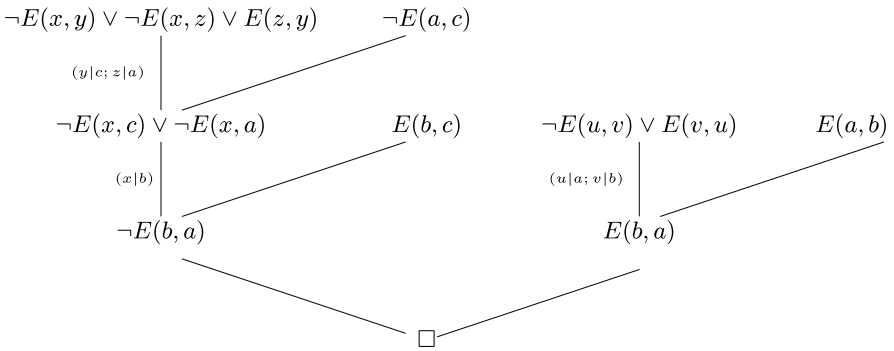
notamos que cada nueva resolvente se ha calculado utilizando justo la que se ha obtenido en el paso anterior. En el dibujo aparece una línea vertical que nos lleva desde la cláusula de partida a la cláusula vacía. Es lo que se llama una *deducción lineal*, como ya vimos anteriormente.

Veamos un ejemplo de una deducción que no sea lineal.

Ejemplo 4.3.2. *Sea el conjunto de cláusulas*

$$\Gamma = \{\neg E(x,y) \vee \neg E(x,z) \vee E(z,y),\ \neg E(u,v) \vee E(v,u),\ E(a,b),\ E(b,c),\ \neg E(a,c)\}$$

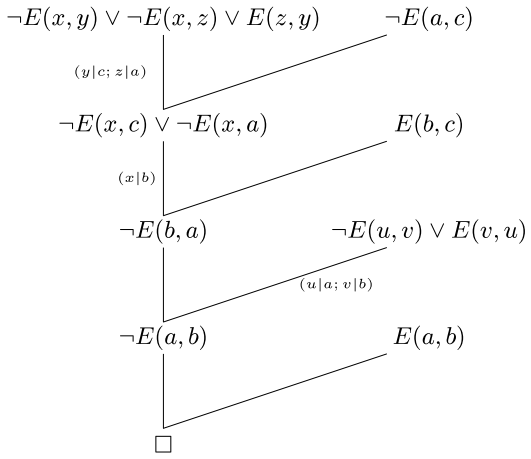
y la siguiente deducción de la cláusula vacía:



Sin embargo, a partir de cada deducción se puede obtener una que sea. Para esto, nos fijamos en la rama de la izquierda, que ha terminado en la cláusula $\neg E(b,a)$ *antes de iniciar la otra rama.*

Luego, la cláusula $\neg E(b,a)$ la hemos unido con el literal $E(b,a)$. Si recorremos la rama de la derecha hacia arriba, vemos que este literal proviene del literal $E(v,u)$ que formaba parte de la cláusula $\neg E(u,v) \vee E(v,u)$.

Calculamos entonces una resolvente de la cláusula $\neg E(b,a)$ con la cláusula $\neg E(u,v) \vee E(v,u)$. Y con esto, tenemos el siguiente árbol de



Esto es un resultado general, se tiene:

Teorema 4.3.1. *Si para un conjunto de cláusulas existe una deducción de la cláusula vacía, entonces existe una deducción lineal de la cláusula vacía.*

Como consecuencia de este teorema, a partir de ahora nos limitaremos a usar deducciones lineales (o refutaciones lineales).

Sin embargo, esto aún deja muchas opciones. En una deducción lineal a la hora de calcular una resolvente una de las cláusulas ya está elegida (la que hemos obtenido en el paso anterior), pero no la otra.

Por tanto, lo que tenemos que ver es cómo elegir la primera cláusula (la que llamaremos raíz), y cómo elegir cada que vamos añadiendo en la deducción.

Vamos a suponer que tenemos fijada la cláusula raíz, y vamos a centrarnos en las cláusulas que pueden ir entrando para calcular resolventes. Las posibilidades aquí son inmensas: podemos no imponer ninguna restricción a este conjunto (es decir, todas las cláusulas que nos vayan saliendo pueden ser usadas para calcular resolvente); podemos elegir siempre una cláusula unit; podemos elegir la cláusula con menos literales; podemos ordenar las cláusulas y los literales elegirlas de acuerdo con ese orden; podemos prefijar el conjunto que vamos a usar desde el principio, etc. Tenemos así estrategia unit, estrategias ordenadas, estrategias input, etc.

De los casos que hemos mencionado, el único completo es el primero (en el que no ponemos restricción). Para los otros, al limitar el conjunto de cláusulas que usamos puede que estemos con un conjunto de cláusulas insatisfacible y la cláusula vacía sólo pueda obtenerse usando alguna de las cláusulas que hemos quitado.

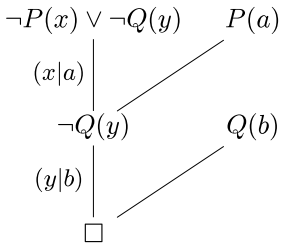
Vamos a analizar brevemente algunas de estas estrategias antes de dedicarnos con más detalle a las deducciones lineales-input.

Comenzamos con la estrategia unit. En ella, para hacer una resolvente vamos a usar una cláusula que tenga sólo un literal.

Partimos del conjunto de cláusulas

$$\Gamma = \{\neg P(x) \vee Q(x),\ \neg P(x) \vee \neg Q(y),\ P(a),\ Q(b)\,\}.$$

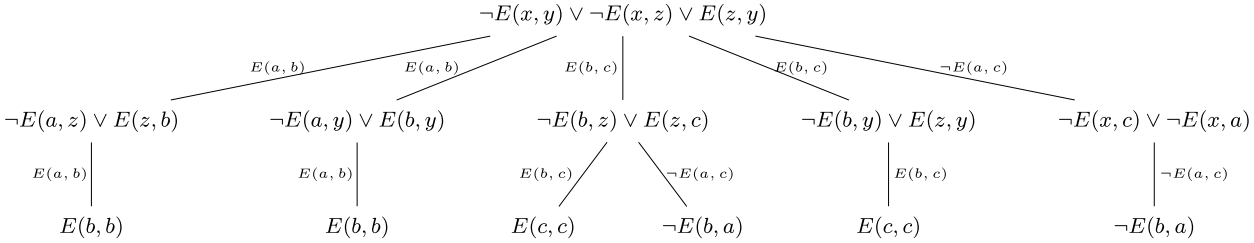
Si tomamos como raíz la cláusula $\neg P(x) \vee \neg Q(y)$ fácilmente podemos llegar a la cláusula vacía usando únicamente cláusulas unit.



Pero veamos el siguiente ejemplo. Partimos del conjunto de cláusulas

$$\Gamma = \{\neg E(x,y) \vee \neg E(x,z) \vee E(z,y),\ \neg E(u,v) \vee E(v,u),\ E(a,b),\ E(b,c),\ \neg E(a,c)\,\}$$

que ya vimos en el ejemplo 4.3.2 que es insatisfacible. Vamos a dibujar el árbol con todas las deducciones lineales-unit con raíz la cláusula $\neg E(x,y) \vee \neg E(x,z) \vee E(z,y)$.



En la raíz del árbol está la cláusula $\neg E(x,y) \vee \neg E(x,z) \vee E(z,y)$.

El nodo $\neg E(a,z) \vee E(z,b)$, que cuelga del raíz por la rama que hemos etiquetado con $E(a,b)$ significa que la cláusula $\neg E(a,z) \vee E(z,b)$ es una resolvente de $\neg E(x,y) \vee \neg E(x,z) \vee E(z,y)$ y $E(a,b)$. En este caso se ha obtenido con la sustitución $(x|a; y|b)$.

Es decir, cada nodo del árbol (salvo el nodo raíz) es una resolvente de la cláusula que aparece en el nodo padre y una cláusula unit.

Notemos que si quisiéramos seguir con esta estrategia no podríamos continuar, pues no puede calcularse ninguna resolvente con las cláusulas que están en las hojas del árbol y las cláusulas unit que tenemos.

Por tanto, en este caso, la estrategia de usar únicamente cláusulas unit no nos da la respuesta adecuada.

Vamos a modificar ligeramente esta estrategia. En lugar de elegir únicamente cláusulas unit, permitimos hacer resolventes con la cláusula más pequeña con que sea posible.

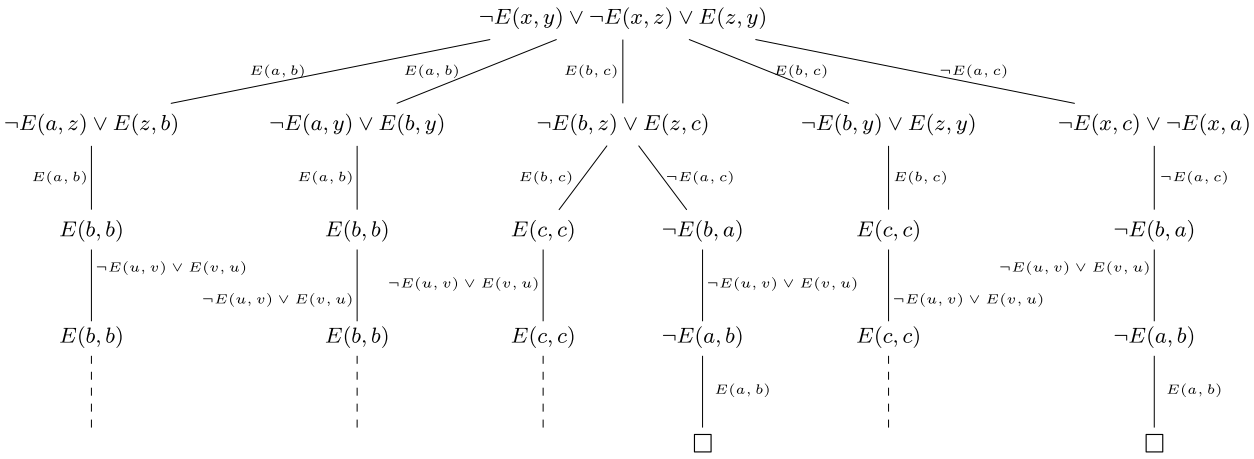
Entonces, una vez llegados a la cláusula $E(b,b)$, y puesto que no hay ninguna cláusula unit con la que se pueda calcular una resolvente, probamos con cláusulas de dos literales. La única que tenemos es $\neg E(u,v) \vee E(v,u)$. En este caso, sí podemos hacer una resolvente, y nos vuelve a salir la cláusula $E(b,b)$.

Esto último nos introduciría en una rama infinita del árbol de deducciones, pues de cada nodo $E(b,b)$ colgaría el nodo $E(b,b)$.

Igual ocurriría con $E(c,c)$.

Pero para las hojas $\neg E(b,a)$, la resolvente con $\neg E(u,v) \vee E(v,u)$ es $\neg E(a,b)$. Y como entre las cláusulas unit tenemos $E(a,b)$, llegaríamos a la cláusula vacía.

Para ver esto más claro, vamos a dibujar el árbol de deducciones lineales con raíz $\neg E(x,y) \vee \neg E(x,z) \vee E(z,y)$, y con preferencia de la cláusula más corta.



A la vista de este ejemplo, lo que habría que hacer es una exploración del árbol de deducciones lineales, y ver si en ese árbol se encuentra la cláusula vacía.

Puesto que el árbol puede ser muy grande, nos limitamos a explorar algunos subárboles. En este ejemplo, hemos tomado el subárbol formado por las deducciones lineales con raíz la cláusula $\neg E(x,y) \vee \neg E(x,z) \vee E(z,y)$, y donde las resolventes se eligen tomando la cláusula más pequeña.

La exploración de un árbol puede hacerse primero en profundidad (con retroceso) o primero en anchura. Cada una de las dos técnicas tiene sus ventajas con respecto a la otra.

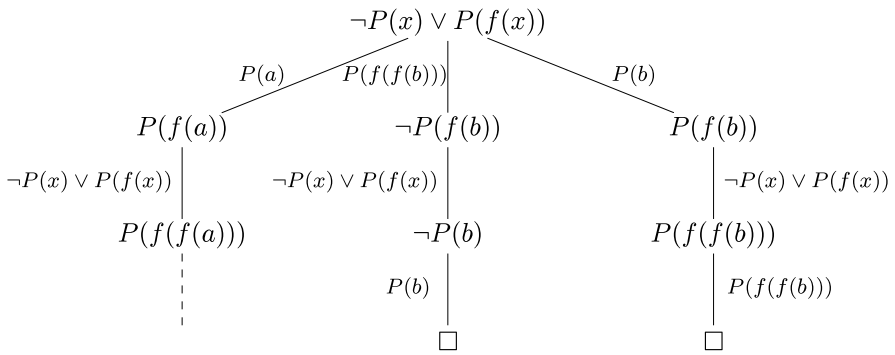
La exploración primero en profundidad requiere menos uso de memoria y es más fácil de programar. Pero corre el riesgo de perderse en una rama infinita, lo cual nos haría imposible el acceso a otras ramas donde posiblemente se encuentre la cláusula vacía.

La exploración primero en anchura nos va a encontrar la cláusula vacía caso de que se encuentre entre los nodos del árbol. Pero requiere un gasto muy grande de memoria.

En el ejemplo que tenemos más arriba, si elegimos exploración primero en profundidad y comenzamos por la rama de la izquierda, entramos en una rama infinita.

Una solución en este caso sería una vuelta atrás cuando se encuentre un nodo repetido. En tal caso, al llegar por segunda vez a la cláusula $E(b,b)$ volvería hacia atrás y exploraría otra rama. Sin embargo, esta solución no nos vale en el siguiente ejemplo.

Sea el conjunto de cláusulas $\{\neg P(x) \vee P(f(x)),\ P(a),\ \neg P(f(f(b))),\ P(b),\ \neg P(f(b))\}$. Vamos a dibujar el árbol de deducciones lineales con raíz la cláusula $\neg P(x) \vee P(f(x))$ y con preferencia de la cláusula más pequeña.



Vemos que si exploramos en primer lugar la rama de la izquierda entramos en una rama infinita. Y además, en ningún momento se repite una cláusula luego no acabaríamos nunca. Sin embargo, si comenzamos con cualquiera de las otras dos ramas en seguida llegamos a la cláusula vacía. De haber hecho una exploración primero en anchura también habríamos encontrado pronto la cláusula vacía.

Para hacer una exploración primero en profundidad con retroceso vemos que es importante el orden en que vayamos seleccionando las ramas a explorar. Este orden se puede prefijar de antemano dando un

orden en las cláusulas que vamos a ir usando. Ya al tomar la estrategia de elección de la cláusula más pequeña estamos en cierto modo ordenando las cláusulas.

Vamos a hacer un pequeño resumen, con los ejemplos que hemos visto, de las distintas situaciones que se nos han presentado.

1. En primer lugar consideramos el conjunto de cláusulas

$$\{\neg E(x,y) \vee \neg E(x,z) \vee E(z,y),\ \neg E(u,v) \vee E(v,u),\ E(a,b),\ E(b,c),\ \neg E(a,c)\}$$

y vamos a tomar deducciones lineales con raíz la cláusula $\neg E(x,y) \vee \neg E(x,z) \vee E(z,y)$. Haremos exploración primero en profundidad.

- Si adoptamos la estrategia de usar cláusulas unit, en ningún caso llegaremos a la cláusula vacía (y el conjunto es insatisfacible).
- Supongamos que ordenamos las cláusulas y vamos seleccionando la rama a explorar siguiendo el orden que hemos dado.
  - Si el orden de las cláusulas es $E(a,b),\ E(b,c),\ \neg E(a,c),\ \neg E(u,v) \vee E(v,u),\ \neg E(x,y) \vee \neg E(x,z) \vee E(z,y)$ entraremos en una rama infinita. Ahora bien, como en esa rama las cláusulas que se van obteniendo se repiten podemos añadir una condición de vuelta atrás cuando lleguemos a una situación anterior. En tal caso, después de explorar varias ramas llegaremos a la cláusula vacía.
  - Si el orden de las cláusulas es $\neg E(a,c),\ E(a,b),\ E(b,c),\ \neg E(u,v) \vee E(v,u),\ \neg E(x,y) \vee \neg E(x,z) \vee E(z,y)$ entonces encontraremos en la primera exploración la cláusula vacía.

2. Ahora consideramos el conjunto de cláusulas

$$\{\neg P(x) \vee P(f(x)),\ P(a),\ \neg P(f(f(b))),\ P(b),\ \neg P(f(b))\}$$

y las deduccines lineales con raíz $\neg P(x) \vee P(f(x))$.

- Si tomamos la estrategia de usar sólo cláusulas unit no encontraremos la cláusula vacía.
- Si tomamos la estrategia de usar la cláusula más pequeña, y empezamos por $P(a)$ entraremos en una rama infinita y en la que no se repiten dos cláusulas. Por tanto, no llegaremos a la cláusula vacía.
- Si adoptamos la estrategia de la cláusula más pequeña, y comenzamos por $P(b)$ o $\neg P(f(f(b)))$ encontraremos rápidamente la cláusula vacía.

Deducciones lineales-input

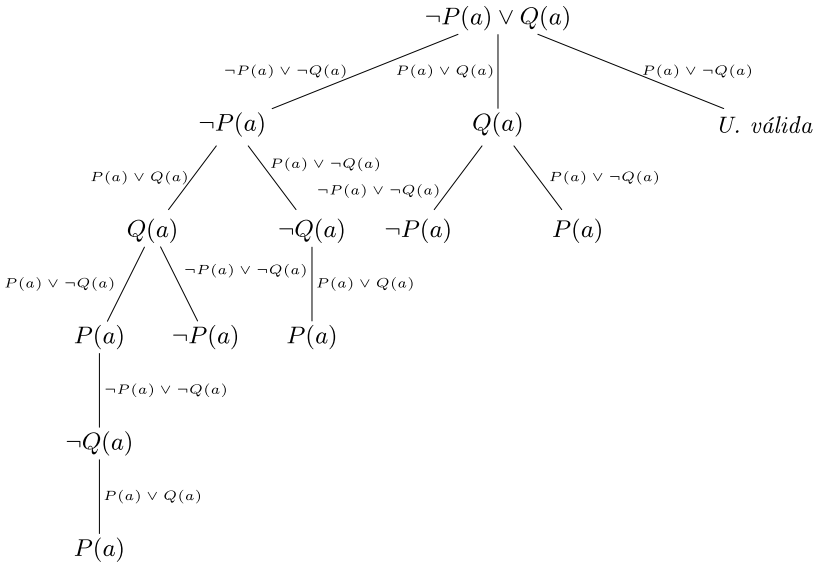
En los ejemplos que hemos visto las cláusulas que íbamos incorporando a la deducción eran siempre del conjunto inicial de cláusulas. En tal caso, hablaremos de una deducción input. La otra posibilidad es poder usar las cláusulas que vamos obteniendo en la deducción.

La estrategia lineal-input no es completa. Ya lo vimos en el tema de lógica proposicional. Vamos a ver aquí otro ejemplo.

Ejemplo 4.3.3. *Consideremos el conjunto*

$$\Gamma = \{\neg P(a) \vee Q(a),\ \neg P(a) \vee \neg Q(a),\ P(a) \vee Q(a),\ P(a) \vee \neg Q(a)\}.$$

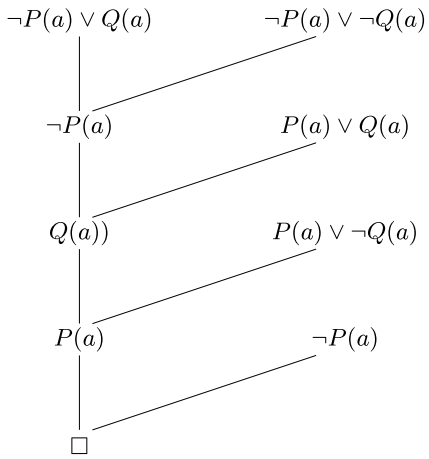
Vamos a tomar como raíz la cláusula $P(a) \vee Q(a)$ y vamos a formar el árbol de todas las deducciones lineales-input con esta raíz.



Y vemos como cualquier rama que pase por $P(a)$, como cualquier rama que pase por $\neg P(a)$ puede prolongarse hasta el infinito, y nunca llega a la cláusula vacía.

Si hubiéramos tomado como raíz otra de las cláusulas, nos saldría un árbol de deducciones semejante al que hemos obtenido aquí.

Sin embargo, este conjunto es insatisfacible, como podemos ver en el siguiente diagrama:



en el que tenemos una deducción lineal de la cláusula vacía que no es input, pues para la última resolvente que hemos obtenido, la cláusula vacía, no se ha empleado ninguna del conjunto Γ, sino la obtenida en el paso anterior ($P(a)$) y la obtenida en el primer paso ($\neg P(a)$).

Conjuntos de Horn

Como ya sabemos, la estrategia lineal-input no es completa. Sin embargo, hay un caso en el que sí lo es. Este caso ya fue estudiado en la lógica proposicional (ver definición 51 y teorema 2.6.4).

Repetimos aquí esa definición y teorema.

Definición 51. *Sea $\mathcal{L}$ un lenguaje de primer orden.*

1. *Un literal se dice positivo si es una fórmula atómica.*
2. *Un literal se dice negativo si es el negado de una fórmula atómica.*

3. *Una cláusula se dice* negativa *si todos los literales que aparecen en ella son literales negativos.*

4. *Una cláusula se dice* cláusula de Horn *si tiene exactamente un literal positivo.*

5. *Un conjunto de cláusulas es un* conjunto de Horn *si tiene exactamente una cláusula negativa, y el resto de las cláusulas son cláusulas de Horn.*

Ejemplo 4.3.4.
El conjunto de cláusulas

$$\{C(b);\ D(x) \vee \neg M(x);\ \neg D(x) \vee M(x);\ \neg C(y) \vee CC(f(y), y);$$
$$\neg C(y) \vee \neg CC(x, y) \vee M(x);\ \neg M(x)\neg D(x) \vee \neg CC(x, y)\neg C(y)\}$$

es un conjunto de Horn.

Entre las cláusulas de Horn podemos distinguir dos tipos: si tienen algún literal negativo se llaman *reglas* y si sólo contienen al literal positivo se llaman *hechos.*
Una cláusula negativa también se denomina *objetivo.*

Ejemplo 4.3.5. *La siguiente es una lista de cláusulas de Horn:*

$C(b)$ *es un hecho;*

$D(x) \vee \neg M(x)$ *es una regla;*

$\neg D(x) \vee M(x)$ *es una regla;*

$\neg C(y) \vee CC(f(y), y)$ *es una regla;*

$\neg C(y) \vee \neg CC(x, y) \vee M(x)$ *es una regla.*

$\neg M(x) \vee \neg D(x) \vee \neg CC(x, y) \vee \neg C(y)$ *es un objetivo.*

En el lenguaje de probramación PROLOG, un conjunto de reglas y hechos (es decir, un conjunto de cláusulas de Horn) forman un programa.
El siguiente teorema es el que nos justifica el estudio de los conjuntos de Horn.

Teorema 4.3.2. *Sea* Γ *un conjunto de Horn insatisfacible. Entonces existe una deducción lineal-input de la cláusula vacía con raíz la cláusula negativa.*

Es decir, la estrategia de reducirnos a las deducciones lineales-input con raíz la cláusula objetivo (negativa) es completa. Así, en estas circunstancias, será suficiente explorar el árbol de las deducciones lineales-input con raíz el objetivo para determinar si el conjunto es o no insatisfacible.
La observación que hicimos después del teorema 2.6.4 vale exactamente igual en este contexto. En particular, dado un conjunto de Horn, todas las resolventes que se obtienen en una deducción lineal-input con raíz la cláusula negativa son siempre negativas.

Ejemplo 4.3.6.
Sea Γ *el siguiente conjunto de cláusulas:*

$$\{C(b);\ D(x) \vee \neg M(x);\ \neg D(x) \vee M(x);\ \neg C(y) \vee CC(f(y), y);$$
$$\neg C(y) \vee \neg CC(x, y) \vee M(x);\ \neg M(x)\neg D(x) \vee \neg CC(x, y)\neg C(y)\}$$

Este conjunto es un conjunto de Horn. Tomamos la cláusula negativa $\neg M(x)\neg D(x) \vee \neg CC(x, y)\neg C(y)$. *Con esta cláusula podemos calcular 5 resolventes, que son:*

1. $\neg M(x) \vee \neg D(x) \vee \neg CC(x, b)$

2. $\neg M(x) \vee \neg CC(x, y) \vee \neg C(y)$

3. $\neg D(x) \vee \neg CC(x, y) \vee \neg C(y)$

4. $\neg M(x)\neg D(x) \vee \neg C(y)$

5. $\neg D(x) \vee \neg CC(x,y) \vee \neg C(y)$

Esto nos daría 5 ramas en el árbol de deducciones lineales-input con raíz la cláusula negativa.

A partir de cada uno de estos nodos, que de nuevo son objetivos, se vuelven a abrir ramas al resolver con las cláusulas de Horn.

Tenemos que señalar en este momento que este proceso se puede convertir en automático. Sólo es necesario determinar el orden en que se examinan los objetivos parciales (cada uno de los literales que aparecen en el objetivo) y las cláusulas con un literal positivo.

Nos preguntamos ahora qué problemas de implicación semántica van a producir conjuntos de Horn cuando sean transformados a insatisfacibilidad de un conjunto de cláusulas. Podemos dar una sencilla respuesta sin más examinar los distintos tipos de cláusulas.

Un hecho representa a una afirmación sobre la veracidad de un predicado en un contexto: $C(b)$, es decir, C es cierto para la constante b, o $C(x)$ que recordemos que es la manera en la que escribimos $\forall x C(x)$, es decir, C es cierto para todo elemento del dominio.

Para aproximarnos al significado de una regla, digamos $\neg C(y) \vee \neg CC(x,y) \vee M(x)$ observamos que podemos, usando las leyes de Morgan, transformarla en la fórmula lógicamente equivalente $\neg(C(y) \wedge CC(x,y)) \vee M(x)$ que a su vez puede cambiarse por $(C(y) \wedge CC(x,y)) \rightarrow M(x)$. Ahora, recordemos que hay que incluir el cierre universal y leemos $\forall x \forall y[(C(y) \wedge CC(x,y)) \rightarrow M(x)]$. Es decir, si ocurre $C(y)$ y ocurre $CC(x,y)$ entonces ocurre $M(x)$.

Por último fijemos nuestra atención en la cláusula negativa y pensemos que proviene de incluir la negación de la consecuencia en el conjunto de premisas; entonces el objetivo $\neg M(x) \neg D(x) \vee \neg CC(x,y) \vee \neg C(y)$ que, como antes, representa a su cierre universal, proviene de la negación de la fórmula

$$\exists x \exists y[M(x) \wedge D(x) \wedge CC(x,y) \wedge C(y)]$$

que es una pregunta de la forma ¿existen elementos para los que son ciertos simultáneamente todos los predicados?

Ejemplo 4.3.7. *Vamos a mostrar un ejemplo en el que trataremos de determinar si alguna frase es consecuencia lógica del siguiente conjunto de premisas:*

1. *Adán es una persona*
2. *Eva es una persona*
3. *Eva es la madre de Caín*
4. *Eva es la madre de Abel*
5. *Todo hijo de una persona es una persona*

Observamos que cada una de estas premisas está enunciada como un hecho o como una regla. Para su traducción a un lenguaje de primer orden consideramos el lenguaje dado por $\mathcal{C} = \{e, c, a\}$, $\mathcal{R} = \{P, M\}$ y la estructura:

- *Dominio: Seres vivos.*
- *Asignación de constantes: $a =$ Adán, $c =$ Caín, $e =$ Eva.*
- *Asignación de predicados: $P(x) \equiv x$ es persona, $M(x,y) \equiv x$ es madre de y.*

En cuyo caso, las premisas que tenemos podríamos decirlas:

1. $P(a)$ *(hecho)*
2. $P(e)$ *(hecho)*
3. $M(e,c)$ *(hecho)*
4. $M(e,a)$ *(hecho)*

5. $\forall x \forall y(M(x,y) \wedge P(x) \rightarrow P(y)$ *(regla)*

y las correspondientes cláusulas (de Horn) que proporcionan (lo que se llamaría un programa en lenguaje PROLOG) son:

1. $P(a)$ *(hecho)*
2. $P(e)$ *(hecho)*
3. $M(e,c)$ *(hecho)*
4. $M(e,a)$ *(hecho)*
5. $P(y) \vee \neg M(x,y) \vee \neg P(x)$ *(regla).*

Este conjunto de premisas puede utilizarse para determinar si una serie de preguntas tienen respuesta afirmativa. Estas preguntas darán lugar al objetivo. Para este programa podemos formular preguntas como:

¿Es Eva una persona?

¿Es Caín una persona?

¿Hay alguna persona?

¿Es Caín hijo?

¿Tiene madre Abel?

¿Hay alguien que sea hijo de Eva?

¿Tiene madre Eva?

que se traducen en el lenguaje de primer orden que estamos usando por las fórmulas:

1. $P(e)$,
2. $P(c)$,
3. $\exists x P(x)$,
4. $\exists x M(x,c)$,
5. $\exists x M(x,a)$
6. $\exists y M(e,y)$

Cada una de estas preguntas da lugar a un problema de implicación semántica, y este se convierte en un problema de probar la insatisfacibilidad de un conjunto de Horn; además, las cláusulas de Horn son en todos los casos las mismas y solo varía la cláusula objetivo de la que partimos.