

Cuaderno Didáctico

Lógica de Predicados

Jose Emilio Labra Gayo

Daniel Fernández Lanvin

Escuela Universitaria de Ingeniería Técnica en Informática de
Oviedo (E.U.I.T.I.O.)



Universidad de Oviedo



Tabla de Contenidos

Introducción	1
Lenguaje Formal de la Lógica de Predicados	2
Otros Ordenes.....	6
Interpretación	7
Consecuencia y Equivalencia Lógicas.....	14
Formas Normales.....	18
Algoritmo de Resolución	23
Introducción	23
Resolución Proposicional	23
Resolución General	27
Unificación.....	30
Algoritmo de Resolución General.....	35
Estrategias de resolución	44
Prueba de Teoremas por Resolución	49
Bibliografía.....	51
Índice Alfabético	53
Información de Contacto	56



Introducción

En las últimas décadas, ha aumentado considerablemente el interés de la informática por la aplicación de la lógica a la programación. De hecho, ha aparecido un nuevo paradigma de programación (la programación declarativa) cuyos fundamentos teóricos se basan en los desarrollos de la **lógica de predicados** que pretendían alcanzar sistemas de demostración automática de teoremas. En este cuaderno didáctico se describirán los conceptos básicos de esta disciplina aplicables a la informática, en concordancia con el programa de la asignatura de Lógica presente en el primer año de los planes de estudio de Ingeniero Técnico en Informática de Sistemas e Ingeniero Técnico en Informática de Gestión impartidos en la Escuela de Ingeniería Técnica en Informática de Oviedo (EUITIO).

Para su lectura no se necesita ningún conocimiento previo ya que se van definiendo todos los términos que aparecen. No obstante sería recomendable cierta familiarización con la lógica de proposiciones y la nomenclatura matemática.

El orden de exposición se divide en dos bloques:

- Un primer bloque contiene los conceptos clásicos de la lógica de predicados con la definición sintáctica del lenguaje (fórmulas bien formadas), la aproximación semántica (concepto de interpretación y de equivalencia lógica) .

- A continuación, se expone el algoritmo de resolución, para lo cual se describen los conceptos de sustitución y unificación.



Lenguaje Formal de la Lógica de Predicados

El instrumento fundamental de comunicación humana es el Lenguaje, formado por frases de tipo interrogativo, imperativo y **declarativo**. Estas últimas constituyen el elemento básico de descripción del conocimiento.

La **lógica** es la disciplina que estudia los métodos de formalización del conocimiento humano. En lógica se estudian, por tanto, métodos de formalización de frases declarativas. Para ello existen dos niveles de abstracción según el grado de detalle que se quiera formalizar: Lógica proposicional y lógica de predicados.

La **lógica proposicional o lógica de enunciados** toma como elemento básico las frases declarativas simples o proposiciones que son aquellos elementos de una frase que constituyen por sí solos una unidad de comunicación de conocimientos y pueden ser considerados **Verdaderos** o **Falsos**.

La **lógica de predicados** estudia las frases declarativas con mayor grado de detalle, considerando la estructura interna de las proposiciones. Se tomarán como elemento básico los **objetos** y las **relaciones** entre dichos objetos. Es decir, se distingue:

- **Qué se afirma** (predicado o relación)
- **De quién se afirma** (objeto)

A continuación se definirá el lenguaje formal que se utilizará en la lógica de predicados. Se define el conjunto de símbolos que aparecerán en las distintas formalizaciones (alfabeto). Seguidamente se definen las reglas sintácticas de construcción de fórmulas correctas.

Definición 1: El **alfabeto** de la lógica de predicados estará formado por los siguientes conjuntos de símbolos:

- Conjunto de símbolos de **Variables (VAR)**: Está formado por las últimas letras del alfabeto minúsculas. También se utilizan subíndices, por ejemplo: $x, y, z, x_1, y_1, z_1, \dots, x_n, y_n, z_n \in VAR$.
- Conjunto de símbolos de **Constantes (CONS)**: Primeras letras del alfabeto minúsculas (con subíndices), por ejemplo: $a, b, c, a_1, b_1, c_1, \dots, a_n, b_n, c_n \in CONS$.
- Conjunto de letras de **función (FUNC)**: Está formado por las letras f, g, h, \dots . También se pueden incluir subíndices para diferenciar distintas funciones: $f_1, g_1, h_1, \dots, f_n, g_n, h_n \in FUNC$.

En algunos casos se indica la aridad¹ mediante un superíndice. Así por ejemplo $f^2 \in FUNC$ será una función con dos argumentos.

¹La **aridad** de una función o de un predicado se define como el número de argumentos que tiene.

- Conjunto de letras de **Predicado** ($PRED$): Se representan mediante letras mayúsculas, $P, Q, R, K \in PRED$

Como en el caso de las funciones, se puede representar la aridad mediante un superíndice, por ejemplo: $P^3 \in PRED$ será un predicado con tres argumentos.

Símbolos de **conectivas**: Serán:

- \neg = Negación
- \wedge = conectiva 'y'
- \vee = conectiva 'o'
- \rightarrow = implicación
- \leftrightarrow = doble implicación o equivalencia

Cuantificadores: Serán los símbolos:

- \forall = Cuantificador universal
- \exists = Cuantificador existencial

Signos de **puntuación**: Paréntesis () y coma.

Definición 2: Un **término** es una cadena de símbolos utilizada para representar objetos siguiendo las siguientes reglas:

- Toda variable o constante individual es un término.
- Si t_1, t_2, \dots, t_n son términos y f^n es una función de aridad n entonces $f^n(t_1, t_2, \dots, t_n)$ es un término.
- Todos los términos posibles se generan aplicando únicamente las dos reglas anteriores.

Definición 3: Un **átomo**² es una cadena de símbolos de la forma:

$$P^n(t_1, t_2, \dots, t_n)$$

Donde P^n es un predicado de aridad n y t_1, t_2, \dots, t_n son términos.

²Obsérvese que un término representará un objeto, mientras que un átomo tomará un valor Verdadero o Falso según la interpretación.



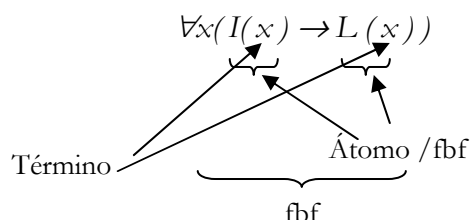
Definición 4: Dado un alfabeto de lógica de predicados, se puede definir el conjunto de fórmulas bien formadas (fbf) cuyos elementos siguen las reglas:

- Todo átomo es una fórmula bien formada (se denominará **fórmula atómica**)
- Si A y B son fórmulas bien formadas entonces:

$$\left. \begin{array}{l} (A) \\ (B) \\ \neg A \\ \neg B \\ A \wedge B \\ A \vee B \\ A \rightarrow B \\ A \leftrightarrow B \end{array} \right\} \text{ Son fórmulas bien formadas}$$

- Si A es una fórmula bien formada y x una variable³ entonces $\forall x(A)$ y $\exists x(A)$ serán fórmulas bien formadas.

Ejemplo 1: La frase "*Todos los estudiantes de informática son listos*" podría formalizarse empleando los predicados $I(x)$ ="x estudia informática" y $L(x)$ ="x es listo" como:



Definición 5: En una expresión del tipo $\forall xA$, $\exists xA$, la variable x es conocida como **variable de cuantificación**, a la fórmula A como **ámbito** o **recorrido** de la cuantificación.

Definición 6: En una fórmula bien formada A , una variable está ligada si está en el ámbito de un cuantificador que la tiene como variable de cuantificación.. Una variable está libre si no está ligada.

³Más adelante se modificará esta regla imponiendo la condición de que x sea una **variable libre** en \mathcal{A}



Ejemplo 2: En la expresión: $\forall x \underbrace{P(x, f(x, y))}_{\text{ámbito}}$

- La variable x aparece en el ámbito del cuantificador universal que además la tiene como variable de cuantificación, por tanto, la variable x está ligada.

- **La variable y** también está en el ámbito del cuantificador universal, pero éste no la tiene como variable de cuantificación, por lo tanto, es una variable libre.

Definición 7: Se dice que una fórmula bien formada es una **fórmula cerrada** si **todas** sus variables están **ligadas**.

Con las definiciones anteriores podrían aparecer problemas si una variable aparece cuantificada dos veces. Por ejemplo, en la fórmula $\exists x (P(x) \rightarrow \forall x Q(x))$, la x está sometida a dos cuantificaciones. Para evitar confusiones. En la definición de fórmula bien formada, se establece una restricción en el tercer punto que quedaría:

*“Si A es una fórmula bien formada y **x una variable libre en A** entonces $\forall x A$ y $\exists x A$ serán fórmulas bien formadas.”*



Otros Ordenes

Como ya se ha indicado, existen diversos órdenes dentro del cálculo de predicados.

Se denomina lógica de **predicados de Orden Cero** a la lógica de predicados en la que se trabaja con predicados de aridad cero (serán proposiciones Verdaderas o Falsas, de ahí que también se le conozca como lógica de proposiciones o enunciados). Puesto que no se utilizan constantes, variables, funciones ni cuantificadores su estudio es sencillo. Sin embargo, existen estructuras deductivas que la lógica de proposiciones no puede formalizar de forma adecuada, por ejemplo, la deducción:

"Todos los informáticos son listos, Juan es informático, luego Juan es listo"

En lógica de predicados de orden cero se formalizaría mediante tres proposiciones p, q y r independientes y la fórmula resultante " $p \wedge q \rightarrow r$ " no sería válida.

En **lógica de predicados de Primer Orden** se permite la cuantificación de variables. De esa forma, el razonamiento anterior se formalizaría :

$$(\forall x(\text{Informático}(x) \rightarrow \text{Listo}(x)) \wedge \text{Informático}(\text{Juan})) \rightarrow \text{Listo}(\text{Juan})$$

En este caso, sí se puede comprobar la validez de esa fórmula (se estudiarán métodos en los siguientes apartados).

Sin embargo, se impone la limitación de no poder cuantificar más que variables, con lo cual, existe un amplio conjunto de frases que se quedan fuera del ámbito de la lógica de primer orden.

Ejemplo 3: Una frase de la forma: "*Existe una función f tal que para cualquier x existe un y de forma que $f(x) = y$* " se debería formalizar incluyendo un cuantificador existencial sobre las funciones :

$$\exists f \forall x \exists y \text{ Igual}(f(x), y)$$

Ejemplo 4: La frase "*Algunas relaciones entre pares de alumnos de la clase son simétricas⁴*" se formalizaría: $\exists R \forall x \forall y (R(x, y) \rightarrow R(y, x))$

En los ejemplos anteriores aparecen fórmulas con cuantificadores aplicados a variables de predicado y función, suponiendo la existencia de dominios predicados y de funciones. La lógica que permite ese tipo de cuantificación se conoce como **lógica de predicados de Segundo Orden**.

El problema de identificar fórmulas válidas está resuelto⁵ en lógica proposicional (Orden Cero), es parcialmente resoluble en lógica de predicados de Primer Orden e irresoluble en lógica de Orden Superior.

⁴Se recuerda que una relación R es simétrica si xRy implica yRx para todo x e y .



Interpretación

Una interpretación es un mecanismo que permitirá asignar un valor Verdadero (**V**) o Falso (**F**) a una fórmula. Para ello es necesario dar un significado a cada uno de los símbolos que aparecen en la fórmula. Las conectivas y los cuantificadores tienen un significado fijo, mientras que el significado de las constantes, símbolos de función y símbolos de predicado puede variar.

En una interpretación, se define un **dominio** de discurso sobre el cual varían las variables, se asignan valores a las constantes y se definen los símbolos de función y de predicados de forma particular. Se necesita, por tanto, asignar un valor concreto a cada símbolo de la fórmula en un dominio determinado.

Definición 8: Una interpretación de una fórmula F consiste en:

-Un conjunto no vacío D , llamado **Dominio** de la interpretación

-Asociar a cada letra de constante $c \in F$ un elemento del dominio $c^I \in D$.

-Asignar a cada letra de función f de aridad n una aplicación f^I de la forma $D \times D \times \dots \times D \rightarrow D$.

-Asignar a cada letra de predicado P de aridad n una aplicación P^I de la forma $D \times D \times \dots \times D \rightarrow \{\mathbf{V}, \mathbf{F}\}$

⁵Un problema está resuelto si siempre se puede encontrar solución. Es parcialmente resoluble si se encuentra solución para algunos casos y no se encuentra para otros, y es irresoluble si no se encuentra solución en ningún caso.

Ejemplo 5: Sea $F = \forall x(P(x) \wedge Q(x) \rightarrow R(f(x))) \wedge Q(a)$ y las interpretaciones siguientes:

Interpretación	I_1	I_2
Dominio:	Números naturales	Personas
Constantes	$a^{I_1} = 2$	$a^{I_2} = \text{juan}$
Funciones	$f^{I_1}(x) = x^2$	$f^{I_2}(x) = \text{madre de } x$
Predicados:	$P^{I_1}(x) = x \text{ es un número impar}$ $Q^{I_1}(x) = x > 0$ $R^{I_1}(x) = x \text{ es múltiplo de } 9$	$P^{I_2}(x) = x \text{ juega al póker}$ $Q^{I_2}(x) = x \text{ estudia informática}$ $R^{I_2}(x) = x \text{ es terco}$

Bajo la interpretación I_1 la fórmula equivaldría a:

"El cuadrado de todo número impar mayor que cero es múltiplo de 9 y el 2 es mayor que cero"

Mientras que bajo la interpretación I_2 sería:

"Todas las madres de los estudiantes de informática que juegan al póker son tercas y juan estudia informática".

Definición 9: Dada una interpretación I se define el valor de una fórmula F bajo I , denotado por $V_I(F)$ ⁶ de la siguiente forma⁷:

- Si F es un predicado de n argumentos de la forma $P(a_0, a_1, a_2, \dots, a_n)$ entonces $V_I(F) = P(a_0^I, a_1^I, \dots, a_n^I)$ donde cada elemento a_i^I es el resultado de aplicar la interpretación I al argumento a_i .

- Si F es de la forma $\neg G$ entonces $V_I(F) = \begin{cases} \mathbf{V} & \text{si } V_I(G) = \mathbf{F} \\ \mathbf{F} & \text{si } V_I(G) = \mathbf{V} \end{cases}$

⁶En esta definición se supone que la fórmula F no tiene variables libres. Si F tuviese, por ejemplo, x_0, x_1, \dots, x_n variables libres, la valoración de la fórmula se dejaría en función de los posibles valores que puedan tomar x_0, x_1, \dots, x_n en el dominio. La expresión correcta sería $V_I(F)[x_0, x_1, \dots, x_n]$. No se ha utilizado esa notación para no complicar las expresiones resultantes.

⁷Obsérvese que $V_I(F)$ es el valor de verdad de F , que puede ser Verdadero o Falso.



$$\text{-Si } F \text{ es de la forma } G \wedge H \text{ entonces } V_I(F) = \begin{cases} \mathbf{V} & \text{si } V_I(G) = V_I(H) = \mathbf{V} \\ \mathbf{F} & \text{encaso contrario} \end{cases}$$

$$\text{-Si } F \text{ es de la forma } G \vee H \text{ entonces } V_I(F) = \begin{cases} \mathbf{F} & \text{si } V_I(G) = V_I(H) = \mathbf{F} \\ \mathbf{V} & \text{encaso contrario} \end{cases}$$

$$\text{-Si } F \text{ es de la forma } G \rightarrow H \text{ entonces } V_I(F) = \begin{cases} \mathbf{F} & \text{si } V_I(G) = \mathbf{V} \text{ y } V_I(H) = \mathbf{F} \\ \mathbf{V} & \text{encaso contrario} \end{cases}$$

$$\text{-Si } F \text{ es de la forma } G \leftrightarrow H \text{ entonces } V_I(F) = \begin{cases} \mathbf{V} & \text{si } V_I(G) = V_I(H) \\ \mathbf{F} & \text{encaso contrario} \end{cases}$$

-Si F es de la forma $\forall x G(x)$ entonces

$$V_I(F) = \begin{cases} \mathbf{V} & \text{si } V_I(G(d)) = \mathbf{V} \text{ para todo } d \in D \\ \mathbf{F} & \text{encaso contrario} \end{cases}$$

-Si F es de la forma $\exists x G(x)$ entonces:

$$V_I(F) = \begin{cases} \mathbf{V} & \text{si } V_I(G(d)) = \mathbf{V} \text{ para algún } d \in D \\ \mathbf{F} & \text{encaso contrario} \end{cases}$$

(En las dos posibilidades anteriores, $G(d)$ se forma sustituyendo todas las apariciones de x por d en la fórmula $G(x)$)

Es importante remarcar la naturaleza no constructiva de la definición. Si el dominio es infinito, para calcular el valor de la fórmula se deberían chequear infinitos elementos, lo cual es **inviabile**. Nótese la diferencia con respecto a la lógica de proposiciones o lógica de orden 0, en donde SI era posible acotar todas las posibles interpretaciones desde la premisa que cada letra proposicional podía adoptar sólo dos valores (V y F). La definición no aporta ningún algoritmo para calcular $V_I(F)$, de hecho, en la mayoría de los casos, no existe tal algoritmo.

Ejemplo 6: Dada la fórmula $F = \forall x \exists y (P(x, y) \wedge Q(f(x))) \rightarrow \neg Q(g(a, b, f(y)))$, una interpretación I para la fórmula sería:

$$\text{Dominio: } D = \{1, 2, 3\}$$

$$\text{Constantes: } a^I = 1, b^I = 3$$

$$\text{Funciones: } f^I(x) = 4 - x$$

$$g^I(x, y, z) = [(x + y + z) \text{ MOD } 3] + 1$$



Lógica de Predicados E.U.I.T.I.O.



Predicados: $P^I(x, y) = "x \leq y"$

$Q^I(x) = "x = 2 \text{ OR } x = 3"$

Para calcular el valor de la fórmula para esa interpretación se chequeará si para todo x existe un y que cumpla $(P(x, y) \wedge Q(f(x))) \rightarrow \neg Q(g(a, b, f(y)))$.

Se comienza por $x=1$:

- a) Se busca un y , escogiendo, por ejemplo, $y=1$. Como se puede ver en el siguiente esquema, el valor que se obtiene será **F**:

$$\begin{array}{ccccc}
 & 1 & 1 & & 1 \\
 & \underbrace{} & & \underbrace{} & & 1 & 3 & \underbrace{} \\
 & \mathbf{V} & & 3 & & & & 3 \\
 & & & \underbrace{} & & & \underbrace{} & \\
 & & & \mathbf{V} & & & 2 & \\
 & \underbrace{} & & & & \underbrace{} & & \\
 & \mathbf{V} & & & & \mathbf{F} & & \\
 & \underbrace{} & & & & & & \\
 & & & & & \mathbf{F} & &
 \end{array}$$

- b) Puesto que para $y=1$ no se cumple, se busca un nuevo valor de y . Para $y=2$ se tiene:

$$\begin{array}{ccccc}
 & 1 & 2 & & 1 \\
 & \underbrace{} & & \underbrace{} & & 1 & 3 & \underbrace{} \\
 & \mathbf{V} & & 3 & & & & 2 \\
 & & & \underbrace{} & & & \underbrace{} & \\
 & & & \mathbf{V} & & & 1 & \\
 & \underbrace{} & & & & \underbrace{} & & \\
 & \mathbf{V} & & & & \mathbf{V} & & \\
 & \underbrace{} & & & & & & \\
 & & & & & \mathbf{V} & &
 \end{array}$$

Al obtener valor **V** no es necesario chequear más valores de y (ya se ha encontrado uno) y se sigue buscando un nuevo valor de x .



c) Para $x=2$:

a. $y=1$:

$$(P(x, y) \wedge Q(f(x))) \rightarrow \neg Q(g(a, b, f(y)))$$

2	1	2	1	3	1
⏟		⏟			⏟
F		2			3
		⏟		⏟	
		V		2	
⏟			⏟		
F			F		
⏟					
V					

(ya se ha encontrado un valor de y , se chequea el último valor de x)

d) Para $x=3$:

a. $y=1$:

$$(P(x, y) \wedge Q(f(x))) \rightarrow \neg Q(g(a, b, f(y)))$$

3	1	3	1	3	1
⏟		⏟			⏟
F		1			3
		⏟		⏟	
		F		2	
⏟			⏟		
F			F		
⏟					
V					

Como se ha encontrado un valor de y para todos los x que hacía **V** la fórmula, el valor de la fórmula bajo esa interpretación $V_I(F) = \mathbf{V}$

Ejemplo 7: Dada la fórmula $F = \forall x \exists y P(x, f(y)) \wedge Q(a)$ y la interpretación I :

Dominio: $D = \{1, 2, 3\}$

Constantes: $a^I = 3$

Funciones:

x	$f^I(x)$
1	2
2	3
3	1

Predicados: $P^I = \{(1,3), (2,3)\}$, $Q^I = \{2,3\}$

Se puede comprobar que, en este caso, para $x=3$ no existe ningún valor de y que haga **V** la fórmula. Por tanto, $V_I(F) = \mathbf{F}$.

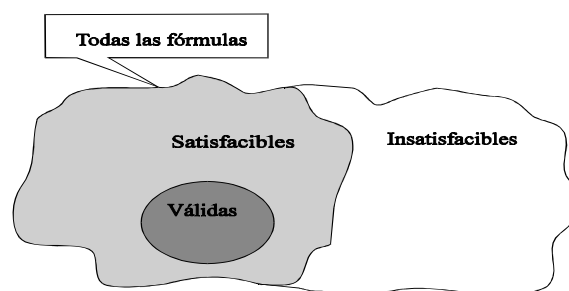
Definición 10: Una interpretación I de una fórmula F es un **modelo** para F si $V_I(F) = \mathbf{V}$.

Definición 11: Una fórmula F es **válida** si y sólo si toda interpretación I es un modelo de F .

Obsérvese que al existir infinitas interpretaciones, no será posible chequear si una fórmula es o no válida mediante tablas de verdad.

Definición 12: Una fórmula F es **satisfacible** si existe alguna interpretación I que sea un modelo de F .

Definición 13: Una fórmula F es **insatisfacible o contradicción** si no existe ninguna interpretación I que sea un modelo de F .





Lógica de Predicados E.U.I.T.I.O.



Una fórmula puede ser: satisfacible o insatisfacible. Un tipo especial de fórmula satisfacible, es aquella que toma siempre valor **V** (es válida). Por tanto, las fórmulas válidas son un subconjunto de las satisfacibles.

Teorema 1: Una fórmula F es **válida** si y sólo si su negación $\neg F$ es **insatisfacible**.

Dem: Si F es válida quiere decir que $V_I(F) = \mathbf{V}$ para toda Interpretación I . Por tanto, $V_I(\neg F) = \mathbf{F}$ para toda interpretación, y no existe ninguna interpretación tal que $V_I(F) = \mathbf{V}$.



Consecuencia y Equivalencia Lógicas

Definición 14: Sea C un conjunto de fórmulas $\{P_1, P_2, \dots, P_n\}$ y sea Q una fórmula. Se dice que Q es **consecuencia lógica** del conjunto C de premisas (se denotará $C \Rightarrow Q$) si toda interpretación que es un modelo de C es también un modelo de Q .

Es decir, si para toda interpretación I se cumple que si $V_I(P_1) = V_I(P_2) = \dots = V_I(P_n) = \mathbf{V}$ entonces $V_I(Q) = \mathbf{V}$ (Intuitivamente, se podría considerar cada interpretación como un "posible mundo". De esa forma, decir que Q es consecuencia lógica de unas premisas es equivalente a pensar que Q toma valor \mathbf{V} en cualquier mundo en el que las premisas tomen valor \mathbf{V}).

Una estructura de la forma $\{P_1, P_2, \dots, P_n\} \Rightarrow Q$ se denomina **razonamiento**. Donde $\{P_1, P_2, \dots, P_n\}$ es el conjunto de premisas y Q , la conclusión.

Se dice que un razonamiento es **correcto** si la conclusión es consecuencia lógica de las premisas.

Teorema 2: $P_1, P_2, \dots, P_n \Rightarrow Q$ es correcto si y sólo si $P_1 \wedge P_2 \wedge \dots \wedge P_n \rightarrow Q$ es válida

Dem: La demostración se dividirá en dos partes:

1.- Si $P_1, P_2, \dots, P_n \Rightarrow Q$ entonces $P_1 \wedge P_2 \wedge \dots \wedge P_n \rightarrow Q$ es válida.

$P_1, P_2, \dots, P_n \rightarrow Q$ no es válida cuando existe una interpretación I que cumple que $V_I(P_1 \wedge P_2 \wedge \dots \wedge P_n) = \mathbf{V}$ mientras que $V_I(Q) = \mathbf{F}$. Es decir en caso de que $V_I(P_1) = V_I(P_2) = \dots = V_I(P_n) = \mathbf{V}$ y $V_I(Q) = \mathbf{F}$. Es decir cuando no se cumple que $P_1, P_2, \dots, P_n \Rightarrow Q$.

2.- Si $P_1 \wedge P_2 \wedge \dots \wedge P_n \rightarrow Q$ es válida entonces $P_1, P_2, \dots, P_n \Rightarrow Q$

Si $P_1 \wedge P_2 \wedge \dots \wedge P_n \rightarrow Q$ es válida entonces toda interpretación para la cual $V_I(P_1) = V_I(P_2) = \dots = V_I(P_n) = \mathbf{V}$ obliga a que $V_I(Q) = \mathbf{V}$, por tanto $P_1, P_2, \dots, P_n \Rightarrow Q$



Definición 15: Se dice que dos fórmulas A y B son **equivalentes lógicamente** (se denota por $A \equiv B$ ó $A \Leftrightarrow B$) si para toda interpretación I , se cumple que $V_I(A) = V_I(B)$

A continuación se indican una serie de fórmulas válidas o leyes del cálculo de predicados.

Se supone que A , B y C son fórmulas bien formadas y x una variable.

Supresión de Implicación: $A \rightarrow B \equiv \neg A \vee B$

Supresión de Doble Implicación: $A \leftrightarrow B \equiv (A \rightarrow B) \wedge (B \rightarrow A)$

Absorción	$A \wedge (B \vee A) \equiv A$ $A \wedge \mathbf{F} \equiv \mathbf{F}$	$A \vee (B \wedge A) \equiv A$ $A \vee \mathbf{V} \equiv \mathbf{V}$
Elemento neutro	$A \wedge \mathbf{V} \equiv A$	$A \vee \mathbf{F} \equiv A$
El. Complementario	$A \wedge \neg A \equiv \mathbf{F}$	$A \vee \neg A \equiv \mathbf{V}$
Idempotencia	$A \wedge A \equiv A$	$A \vee A \equiv A$
Prop. Commutativa	$A \vee B \equiv B \vee A$	$A \wedge B \equiv B \wedge A$
Prop. Asociativa	$A \wedge (B \wedge C) \equiv (A \wedge B) \wedge C$	$A \vee (B \vee C) \equiv (A \vee B) \vee C$
Prop. Distributiva	$A \vee (B \wedge C) \equiv (A \vee B) \wedge (A \vee C)$	$A \wedge (B \vee C) \equiv (A \wedge B) \vee (A \wedge C)$
Leyes de De Morgan	$\neg(A \vee B) \equiv \neg A \wedge \neg B$	$\neg(A \wedge B) \equiv \neg A \vee \neg B$
Doble Negación (Involución)	$\neg\neg A \equiv A$	

Leyes de De Morgan con cuantificadores:

$$\neg \exists x A(x) \equiv \forall x \neg A(x)$$

$$\neg \forall x A(x) \equiv \exists x \neg A(x)$$

Ejemplo 8: La frase "No existen aves que no sepan volar" se formalizaría: $\neg \exists x (Ave(x) \wedge \neg Vuela(x))$

Aplicando las leyes anteriores sería equivalente a:

$$\forall x \neg (Ave(x) \wedge \neg Vuela(x)) \equiv \forall x (\neg Ave(x) \vee Vuela(x)) \equiv \forall x (Ave(x) \rightarrow Vuela(x))$$

que podría leerse como: "Todas las aves saben volar".

Intercambio de cuantificadores:

$$\forall x \forall y A(x, y) \equiv \forall y \forall x A(x, y)$$

$$\exists x \exists y A(x, y) \equiv \exists y \exists x A(x, y)$$

$$\exists x \forall y A(x, y) \Rightarrow \forall y \exists x A(x, y)$$

Nota: La ley anterior se cumple únicamente en el sentido dado, ya que $\forall y \exists x A(x, y) \not\Rightarrow \exists x \forall y A(x, y)$.

Si se Considera la interpretación: *Dominio=Personas*, $A(x,y) = "x \text{ está casado con } y"$.

La ley anterior implicaría la frase:

"Si todas las personas (y) están casadas con alguien (x) entonces existe alguien (x) que está casado con todas las personas (y)"

Gran distributividad

$$\exists x (A(x) \vee B(x)) \equiv (\exists x A(x) \vee \exists x B(x)) \quad \forall x (A(x) \wedge B(x)) \equiv (\forall x A(x) \wedge \forall x B(x))$$

Nota: No se cumplen cambiando la conectiva " \vee " por la conectiva " \wedge ", es decir:

$\exists x (A(x) \wedge B(x)) \not\equiv (\exists x A(x) \wedge \exists x B(x))$, por ejemplo, para la interpretación $A(x) = x \text{ es alumno}$ y $B(x) = x \text{ es un borracho}$. No es lo mismo: "Existen alumnos borrachos" que "Existen alumnos y existen borrachos".

De la misma forma: $\forall x (A(x) \vee B(x)) \not\equiv (\forall x A(x) \vee \forall x B(x))$, ejemplo:

Dominio: estudiantes. $A(x) = x \text{ está alegre}$. $B(x) = x \text{ está triste}$.

La frase "Todos los estudiantes están alegres o tristes" no es equivalente a "Todos los estudiantes están alegres o todos están tristes".



Gran distributividad restringida:

$$B \vee \exists x A(x) \equiv \exists x (B \vee A(x))$$

$$B \vee \forall x A(x) \equiv \forall x (B \vee A(x))$$

$$B \wedge \exists x A(x) \equiv \exists x (B \wedge A(x))$$

$$B \wedge \forall x A(x) \equiv \forall x (B \wedge A(x))$$

(donde x no aparece en B)



Formas Normales

Definición 16: Se dice que una fórmula F está en **Forma Normal Prenexa(FNP)** si es de la forma

$$Q_1x_1Q_2x_2\cdots Q_nx_nM$$

donde $Q_1Q_2\cdots Q_n$ son cuantificadores (\forall, \exists) y M es una fórmula sin cuantificadores.

La secuencia $Q_1x_1Q_2x_2\cdots Q_nx_n$ se denomina **prefijo** de F , mientras que M será la **matriz** de la fórmula.

A continuación se indica mediante un teorema que cualquier fórmula cerrada puede ser transformada en una fórmula equivalente en Forma Normal Prenexa. Si la fórmula no fuese cerrada (tuviese variables libres) se aplicaría el **cierre existencial** consistente en añadir al principio de la fórmula un cuantificador existencial por cada variable libre.

Teorema 3: Para cualquier fórmula cerrada de cálculo de predicados se puede encontrar una fórmula equivalente en Forma Normal Prenexa.

Dem: La demostración consiste en indicar cómo se transformaría la fórmula original utilizando las reglas de equivalencia de la sección anterior, obteniendo la fórmula en Forma Normal Prenexa.

Los pasos a seguir serían:

- Renombrar variables con el mismo nombre y distinto cuantificador.
- Eliminar la doble implicación (\leftrightarrow), utilizando $F \leftrightarrow G \equiv (F \rightarrow G) \wedge (G \rightarrow F)$
- Eliminar implicación: $F \rightarrow G \equiv \neg F \vee G$
- Aplicar leyes de De Morgan con y sin cuantificadores de forma que las negaciones sólo afecten a fórmulas atómicas.
- Pasar los cuantificadores al principio de la fórmula aplicando las leyes de cuantificadores.

A continuación se definirán la Forma Normal Conjuntiva y la Forma Normal Disyuntiva. Para ello, se utilizará la siguiente notación:



Lógica de Predicados E.U.I.T.I.O.



$$\bigwedge_{i=1}^n X_i \text{ es igual a la fórmula } X_1 \wedge X_2 \wedge \dots \wedge X_n \quad \bigvee_{i=1}^n X_i \text{ es igual a la fórmula } X_1 \vee X_2 \vee \dots \vee X_n$$

Definición 17: Un **literal** es una fórmula atómica $p(t_1, t_2, \dots, t_n)$ (también llamado **literal positivo**) o una fórmula atómica negada $\neg p(t_1, t_2, \dots, t_n)$ (llamado **literal negativo**)

Definición 18: Una fórmula F está en **Forma Normal Conjuntiva (FNC)** si es una conjunción finita de la forma $\bigwedge_{i=1}^n \left(\bigvee_{j=1}^{m_i} L_{ij} \right)$, donde cada L_{ij} es un literal.

Definición 19: Una fórmula F está en **Forma Normal Disyuntiva (FND)** si es una disyunción finita de la forma $\bigvee_{i=1}^n \left(\bigwedge_{j=1}^{m_i} L_{ij} \right)$, donde cada L_{ij} es un literal.

Teorema 4: Para cualquier fórmula cerrada se puede encontrar una fórmula equivalente en Forma Normal Conjuntiva (disyuntiva).

Dem: Cualquier fórmula se puede transformar en una fórmula en Forma Normal Conjuntiva (disyuntiva) equivalente mediante las leyes de De Morgan y las propiedades distributivas.

Definición 20: Una fórmula está en **Forma Normal Conjuntiva (disyuntiva) Prenexa**, se escribe FNCP (FNDP), si está en Forma Normal Conjuntiva (disyuntiva) y en Forma Normal Prenexa.

Teorema 5: Cualquier fórmula cerrada se puede transformar en una fórmula lógicamente equivalente a ella en FNCP.

Dem: Se transforma la fórmula en FNP mediante el teorema 3 obteniendo la fórmula en FNP y, mediante el teorema 4 se transforma la matriz a FNC

Definición 21: Una fórmula cerrada está en **Forma Normal de Skolem (FNS)** si está en FNCP y todos los cuantificadores son universales.

Para convertir una fórmula a Forma Normal de Skolem es necesario suprimir los cuantificadores existenciales. La fórmula obtenida no será lógicamente equivalente a la anterior, sino que será equisatisfacible.

A continuación se muestra el algoritmo de skolemización para suprimir los cuantificadores existenciales de una fórmula cerrada.

Algoritmo (skolemización)

- 1.- Se busca el primer cuantificador existencial comenzando por la izquierda.
- 2.- Si no se encontró, finalizar.
- 3.- Si el cuantificador existencial está al principio (de la forma $\exists x F(x)$), se suprime la variable cuantificada por una nueva constante (quedaría $F(a)$).

Volver al paso 1.

- 4.- Si existen n cuantificadores universales antes del cuantificador existencial de la forma: $\forall x_1 \forall x_2 \cdots \forall x_n \exists y F(x_1, x_2, \cdots, x_n, y)$ suprimir la variable cuantificada existencialmente (y) por una nueva función de las variables universales anteriores (x_1, x_2, \cdots, x_n). Quedando de la forma: $\forall x_1 \forall x_2 \cdots \forall x_n F(x_1, x_2, \cdots, x_n, f(x_1, x_2, \cdots, x_n))$.

Volver al paso 1.

Teorema 6: Dada cualquier fórmula en lógica de predicados se puede encontrar una fórmula *equisatisfacible* a ella en Forma Normal de Skolem.

Dem: Mediante los teoremas anteriores, se puede obtener una fórmula equivalente en Forma Normal Conjuntiva Prenexa (FNCP). Para obtener la fórmula en FNS es necesario suprimir los cuantificadores existenciales según los pasos 3 y 4 del algoritmo anterior.

Si se realiza el paso 3, una fórmula de la forma $\exists x F(x)$ es satisfacible cuando existe un valor constante del dominio que satisface F . Puesto que la nueva constante a no tiene valor asignado, si se le asocia el valor que satisface F , entonces $F(a)$ será satisfacible.

Si se realiza el paso 4, tenemos un valor que satisface F dependiendo de los valores x_1, x_2, \cdots, x_n . Si escogemos una función f , que asigne ese valor a $f(x_1, x_2, \cdots, x_n)$. Entonces, la fórmula $\forall x_1 \forall x_2 \cdots \forall x_n F(x_1, x_2, \cdots, x_n, f(x_1, x_2, \cdots, x_n))$ también será satisfacible.

La demostración en el otro sentido es evidente.

Obsérvese la diferencia entre $\forall x \exists y C(x, y)$ y $\exists y \forall x C(x, y)$. Si se toma la siguiente interpretación: " $C(x, y)$ = x está casado con y " se tiene:

$\forall x \exists y C(x, y)$ = "Todos están casados con alguien" que sería satisfacible con $C(x, f(x))$ donde se habría escogido la función $f(x)$ como la persona que está casada con x .

$\exists y \forall x C(x, y)$ = "Existe alguien que está casado con todos". En este caso sería equisatisfacible con $C(x, a)$. Escogiendo la constante a como la persona que está casada con todos.



Definición 22: Una fórmula está en **Forma Clausal** si está en FNS y se suprimen los cuantificadores universales y las conjunciones por comas⁸.

Ejemplo 9: Formalizar y pasar a Forma Clausal la frase: *"Todo el que estudie historia o filosofía, aprenderá algo interesante y conocerá cualquier personaje griego"*

Para formalizar la frase, se toma:

- $H(x)$ = x estudia historia - $F(x)$ = x estudia filosofía - $I(x)$ = x es interesante
- $A(x,y)$ = x aprende y - $C(x,y)$ = x conoce y - $G(x)$ = x es un personaje griego

Quedaría la fórmula:

$$\forall x((H(x) \vee F(x)) \rightarrow (\exists y(I(y) \wedge A(x, y)) \wedge \forall y(G(y) \rightarrow C(x, y))))$$

Para transformar la fórmula a Forma Clausal se seguirá, los pasos generales:

1.- Renombrar variables comunes:

$$\forall x((H(x) \vee F(x)) \rightarrow (\exists y(I(y) \wedge A(x, y)) \wedge \forall z(G(z) \rightarrow C(x, z))))$$

1.- Eliminar \rightarrow

$$\forall x(\neg(H(x) \vee F(x)) \vee (\exists y(I(y) \wedge A(x, y)) \wedge \forall z(\neg G(z) \vee C(x, z))))$$

2.- Introducir negaciones.

$$\forall x((\neg H(x) \wedge \neg F(x)) \vee (\exists y(I(y) \wedge A(x, y)) \wedge \forall z(\neg G(z) \vee C(x, z))))$$

3.- Pasar cuantificadores al principio de la fórmula. Se obtiene FNP.

$$\forall x \exists y \forall z((\neg H(x) \wedge \neg F(x)) \vee ((I(y) \wedge A(x, y)) \wedge (\neg G(z) \vee C(x, z))))$$

4.- Poner la matriz en FNC, aplicando propiedades distributivas. Se obtiene FNCP.

$$\forall x \exists y \forall z \left\{ \begin{array}{l} (\neg H(x) \vee I(y)) \wedge \\ (\neg H(x) \vee A(x, y)) \wedge \\ (\neg H(x) \vee \neg G(z) \vee C(x, z)) \wedge \\ (\neg F(x) \vee I(y)) \wedge \\ (\neg F(x) \vee A(x, y)) \wedge \\ (\neg F(x) \vee \neg G(z) \vee C(x, z)) \end{array} \right\}$$

⁸La Forma Clausal es equivalente a la FNS, simplemente se reescribe como un conjunto de cláusulas.



5.- Quitar cuantificadores existenciales. Se obtiene FNS

$$\forall x \forall z \left\{ \begin{array}{l} (\neg H(x) \vee I(f(x))) \wedge \\ (\neg H(x) \vee A(x, f(x))) \wedge \\ (\neg H(x) \vee \neg G(z) \vee C(x, z)) \wedge \\ (\neg F(x) \vee I(f(x))) \wedge \\ (\neg F(x) \vee A(x, f(x))) \wedge \\ (\neg F(x) \vee \neg G(z) \vee C(x, z)) \end{array} \right\}$$

6.- Suprimir cuantificadores universales y reescribir como un conjunto de cláusulas, obteniendo, así Forma Clausal.

$$\left\{ \begin{array}{l} \neg H(x) \vee I(f(x)), \neg H(x) \vee A(x, f(x)), \neg H(x) \vee \neg G(z) \vee C(x, z), \\ \neg F(x) \vee I(f(x)), \neg F(x) \vee A(x, f(x)), \neg F(x) \vee \neg G(z) \vee C(x, z) \end{array} \right\}$$

Definición 23: Una cláusula Horn es una cláusula que tiene como mucho un literal positivo, se denota de la forma: $C \leftarrow P_1, \dots, P_n$ donde el literal positivo C se conoce como la cabeza de la cláusula y los literales negativos como el **cuerpo** de la cláusula.

Si $n = 0$, se denomina **Hecho**, será " C "

Una cláusula Horn sin literal positivo se conoce como **Objetivo**, será de la forma " $\leftarrow P_1, \dots, P_n$ "

Definición 24: Se denomina cláusula vacía y se denota por \square a la cláusula que no tiene literales. Por definición, la cláusula vacía es insatisfacible o contradicción



Algoritmo de Resolución

Introducción

El método de resolución es un algoritmo fácilmente mecanizable propuesto por J.A. Robinson en 1965. Si un conjunto de cláusulas es insatisfacible, el algoritmo lo detecta y para (teorema de Completud). Si el conjunto es satisfacible podría detectarlo o no parar, por lo cual se dice que es un algoritmo parcialmente completo.

La presentación se dividirá en dos partes, en la primera parte se presenta el algoritmo de resolución para lógica de proposiciones demostrando su consistencia y completud.

En la segunda parte se presentará el algoritmo de resolución general para lógica de predicados, el cual incluye un paso previo de unificación de cláusulas mediante sustituciones.

El algoritmo se basa en una única regla de inferencia sencilla y, a la vez de gran potencia: la **regla de resolución**. Puesto que se utiliza una sola regla, es fácil de analizar e implementar.

Resolución Proposicional

Intuitivamente, la idea de la resolución proposicional es simple:

Si se sabe que se cumple: "P ó Q" y que se cumple "no P ó R" entonces se puede deducir que se cumplirá "Q ó R".

Ejemplo 21: Si se tiene: "Gana o Pierde o Empata" y "Si Gana entonces da una Fiesta o Va de Viaje". Se puede deducir que: "O Pierde o Empata o da una Fiesta o va de Viaje".

Formalizando, la primera frase sería: $G \vee P \vee E$ y la segunda: $G \rightarrow F \vee V \equiv \neg G \vee F \vee V$

La regla de resolución inferirá: $P \vee E \vee F \vee V$

Definición 34: Dadas dos cláusulas C_1 y C_2 tales que exista un literal l de forma que $l \in C_1$ y $\neg l \in C_2$, se denomina **resolvente de C_1 y C_2 respecto a l** a: $R_l(C_1, C_2) = (C_1 - \{l\}) \cup (C_2 - \{\neg l\})$. Se dice que C_1 y C_2 son **cláusulas resolubles**.

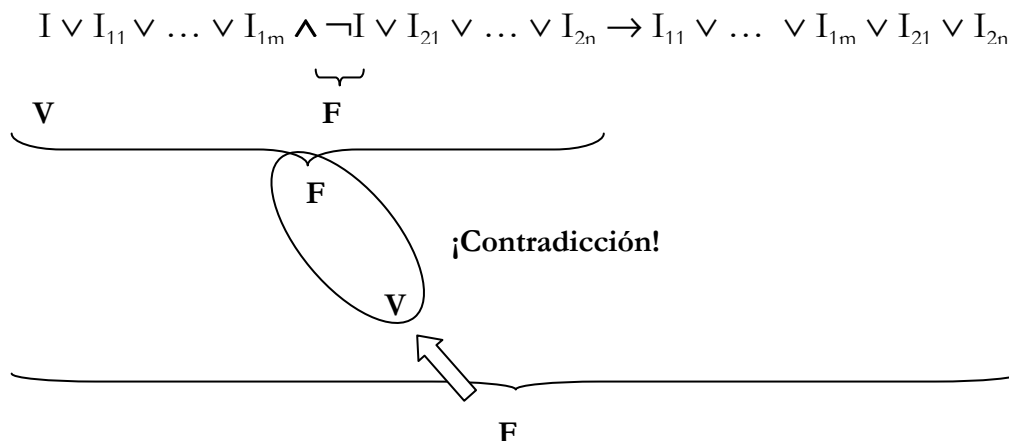
Teorema 6 (Consistencia de la regla de resolución): El resolvente de dos cláusulas es consecuencia lógica de ellas. Es decir $\{C_1, C_2\} \Rightarrow R(C_1, C_2)$

Dem: Sean $C_1 = l \vee l_{11} \vee l_{12} \vee \dots \vee l_{1m}$ y $C_2 = \neg l \vee l_{21} \vee l_{22} \vee \dots \vee l_{2n}$ dos cláusulas resolubles.

El resolvente de C_1 y C_2 respecto a l será $R_l(C_1, C_2) = l_{11} \vee l_{12} \vee \dots \vee l_{1m} \vee l_{21} \vee l_{22} \vee \dots \vee l_{2n}$



Por el teorema 2, probar que $\{C_1, C_2\} \Rightarrow R(C_1, C_2)$ es equivalente a probar que $C_1 \wedge C_2 \rightarrow R(C_1, C_2)$ es válida. Supóngase que existe una interpretación que la hace Falsa, la asignación de valores será:



Puesto que se llega a una contradicción, la fórmula no puede ser **Falsa** y será siempre verdadera, es decir, la fórmula es **Válida**.

Teorema 7: Si el resolvente de 2 cláusulas C_1 y C_2 es la cláusula vacía, entonces esas dos cláusulas son insatisfacibles.

Dem: Según el teorema anterior, el resolvente de 2 cláusulas es consecuencia lógica de ellas, por tanto, la fórmula: $C_1 \wedge C_2 \rightarrow \square$ es válida. Puesto que \square tiene valor Falso, para que la implicación sea válida, $C_1 \wedge C_2$ tienen que tener valor Falso. Es decir, son insatisfacibles.

El algoritmo de resolución chequeará si un conjunto de cláusulas es insatisfacible, para ello seleccionará pares de cláusulas resolubles y calculará su resolvente. Si se obtiene la cláusula vacía, el conjunto es insatisfacible. En caso contrario, el resolvente se añade al conjunto de cláusulas y se seguirán buscando nuevos pares de cláusulas.

Algoritmo de resolución proposicional

Entrada: Un conjunto de cláusulas C

Salida: Detecta si C es insatisfacible

- 1.- Buscar dos cláusulas $C_1, C_2 \in C$ tales que exista un literal l que cumple que $l \in C_1$ y $\neg l \in C_2$
- 2.- Si se encuentran:
 - 3.- Calcular $R_l(C_1, C_2)$ y añadirlo al conjunto C
 - 4.- Si $R_l(C_1, C_2) = \square$ entonces **SALIR** indicando que C es **insatisfacible**.
 - 5.- Si no, Añadir $R_l(C_1, C_2)$ a C y Volver a 1
- 3.- Si no se encuentran: **SALIR** indicando que C **no es insatisfacible**.

Ejemplo 22: Sea C el siguiente conjunto de cláusulas $\{p, \neg p \vee q, \neg r, \neg p \vee \neg q \vee r\}$, se puede demostrar que C es insatisfacible por resolución. Para ello:

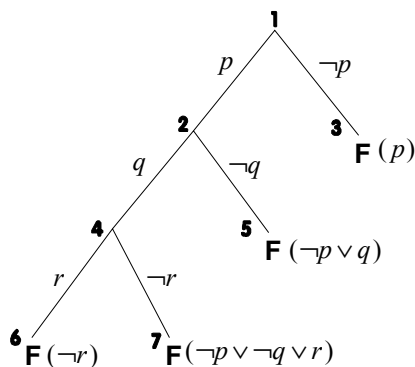


- Se resuelve la tercera cláusula ($\neg r$) con la cuarta ($\neg p \vee \neg q \vee r$), obteniendo $\neg p \vee \neg q$.
- Se resuelve ahora la cláusula anterior con la segunda cláusula ($\neg p \vee q$) obteniendo: $\neg p$
- Se resuelve ahora la cláusula anterior con la primera y se llega a la cláusula vacía \square

Puesto que se llega a la cláusula vacía, C es insatisfacible.

A continuación se presentarán las ideas generales necesarias para la demostración del teorema de completud del algoritmo de resolución proposicional basándose en el concepto de árbol semántico definido en la sección anterior. Se puede profundizar más en [Ben-Ari 93]. Además, existe otra forma de demostrar el teorema por inducción en el número de literales, véase [Sperschneider 91]

Ejemplo 23: Sea el conjunto de cláusulas $C = \{p, \neg p \vee q, \neg r, \neg p \vee \neg q \vee r\}$, para construir el árbol semántico para C se recuerda que un conjunto de cláusulas equivale a una fórmula en Forma Normal Conjuntiva, en este caso, $p \wedge (\neg p \vee q) \wedge (\neg r) \wedge (\neg p \vee \neg q \vee r)$. En la siguiente figura se muestra el árbol semántico correspondiente marcando la cláusula falsificada en los nodos de fallo. El árbol semántico será:



Lema 1: Si un conjunto de cláusulas es insatisfacible, entonces el árbol semántico es finito y está limitado por nodos de fallo, se denomina, en ese caso, **árbol de fallo**.

Lema 2: Cada nodo de fallo n falsifica al menos a una de las cláusulas del conjunto que será la **cláusula asociada a n** .

Lema 3: La cláusula C asociada a un nodo de fallo n contiene un subconjunto de los complementos de los literales que aparecen en la rama que va desde la raíz del árbol semántico hasta n .

Dem: Puesto que la cláusula C es falsificada en el nodo n , todos sus literales deben tener asignado un valor en la interpretación parcial correspondiente a n . Además, el valor de esos literales debe ser **F** (puesto que C es una disyunción) con lo cual, el valor asignado debe ser el complementario.

Definición 35: Se denomina **nodo de inferencia** a un nodo del árbol semántico cuyos dos hijos son nodos de fallo.

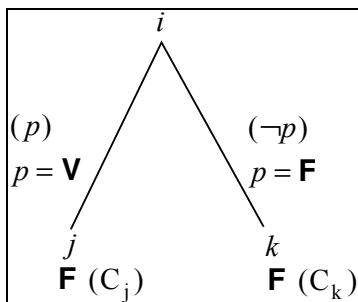
Lema 4: En un árbol de fallo, salvo que sólo tenga un nodo, debe existir al menos un nodo de inferencia.

Dem: Puesto que el árbol de fallo es finito y las ramas se desarrollan de dos en dos, necesariamente tendremos un último nodo desarrollado con dos hijos.



Lema 5: Un nodo de inferencia i indica un paso de resolución de las cláusulas asociadas a sus dos hijos. El resolvente de dichas cláusulas es falsificado por el nodo i y, ocasionalmente, por alguno de sus antecesores.

Dem: En un nodo de inferencia i cualquiera, se tendrá un esquema como el que sigue:



- Puesto que el nodo i no falsificó C_j y lo único que cambia en el nodo j respecto a i es el valor de p , la cláusula C_j debe contener el literal $\neg p$ (complementado para que sea Falso).

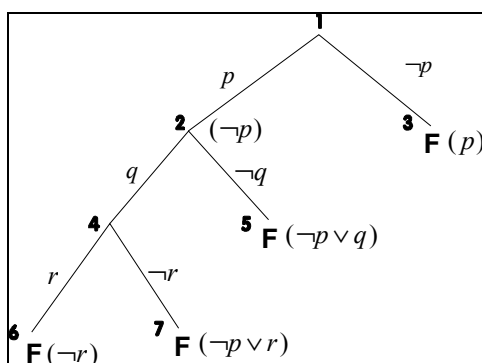
- Por la misma razón anterior, la cláusula C_k debe contener el literal p (sin complementar para que sea Falso)

Por tanto C_j y C_k son resolubles respecto a p . El esquema será:

$$\left. \begin{array}{l} C_j = \neg p \vee \text{resto}_j \\ C_k = p \vee \text{resto}_k \end{array} \right\} R_p(C_j, C_k) = \text{resto}_j \vee \text{resto}_k$$

En el nodo j , C_j toma valor Falso, por tanto resto_j tomará también valor Falso, como resto_j no contiene el literal p también tomarán valor Falso en el nodo i . De la misma forma, resto_k tomará valor Falso en el nodo i . Por tanto, $R_p(C_j, C_k) = \text{resto}_j \vee \text{resto}_k$ tomará valor Falso en el nodo i , es decir, el nodo i , es un nodo de fallo para el resolvente de C_j y C_k

En ocasiones, puede ocurrir que el resolvente sea falsificado también por alguno de los padres del nodo de inferencia, como ejemplo, considérese el conjunto de cláusulas $\{p, \neg p \vee q, \neg r, \neg p \vee r\}$, el árbol semántico, junto con los resolventes sería:



El resolvente de los nodos 6 y 7 es $(\neg p)$ que falsifica al nodo 4 pero también falsifica a su antecesor, el nodo 2.



Teorema 8 (Compleitud del Algoritmo de Resolución Proposicional): Si un conjunto de cláusulas es insatisfacible entonces, aplicando el algoritmo de resolución, se alcanza la cláusula vacía.

Dem: Dado un conjunto de cláusulas insatisfacibles, el árbol semántico será un árbol de fallo (Lema 1), además existirá al menos un nodo de inferencia (Lema 4) con el cual se puede formar un resolvente de sus dos hijos (Lema 5) y añadirlo al conjunto. Si se construye el árbol semántico para el nuevo conjunto, el nuevo árbol semántico será más pequeño que el anterior (puesto que se encuentra un nodo de fallo antes) y, como el conjunto es insatisfacible, existirá de nuevo otro nodo de inferencia, ..., repitiendo el proceso se llegará a un árbol semántico con un solo nodo que corresponderá a la cláusula vacía.

Resolución General

Como ya se ha indicado, el algoritmo de resolución general para lógica de predicados de primer orden es una generalización del algoritmo de resolución proposicional que incluye un paso previo de unificación de cláusulas mediante sustituciones. Es necesario, por tanto, definir qué es una sustitución para definir a continuación qué es un unificador de cláusulas. Una vez definidos ambos conceptos se presenta el método de resolución general demostrando su consistencia y completud.

Sustitución

Una sustitución será un mecanismo que nos permitirá transformar fórmulas. El interés principal de las sustituciones radica, en esta exposición, en su **aplicación a la unificación** de un conjunto de expresiones, haciendo que las expresiones resulten sintácticamente idénticas.

Definición 36: Una **sustitución** σ es un conjunto finito de la forma $\{v_1 / t_1, v_2 / t_2, \dots, v_n / t_n\}$ donde cada v_i es una variable, cada t_i es un término distinto de v_i y las variables v_1, v_2, \dots, v_n son distintas entre sí.

Una **sustitución** es **básica** si los términos t_i no contienen variables.

Definición 37: Una **expresión** es un término, un literal o una conjunción o disyunción de literales. Una **expresión simple** es un término o un átomo.

Definición 38: Sea $\sigma = \{v_1 / t_1, v_2 / t_2, \dots, v_n / t_n\}$ y E una expresión, entonces se define la **instancia** de E por σ , denotado por $\sigma(E)$, como la expresión que resulta de substituir simultáneamente cada aparición de la variable v_i en E por el término t_i ($i = 1, 2, \dots, n$).

Si $\sigma(E)$ no contiene variables entonces $\sigma(E)$ es una **instancia básica** de E .

Ejemplo 24: Sea $E = P(x, y, f(a))$ y $\sigma = \{x / b, y / f(x)\}$ entonces $\sigma(E) = P(b, f(x), f(a))$.



Definición 39: Sean $\sigma_1 = \{x_1 / t_1, x_2 / t_2, \dots, x_m / t_m\}$ y $\sigma_2 = \{y_1 / s_1, y_2 / s_2, \dots, y_n / s_n\}$ dos sustituciones, entonces la **composición de sustituciones** $\sigma_1 \circ \sigma_2$ es la sustitución:

$\{x_1 / \sigma_2(t_1), x_2 / \sigma_2(t_2), \dots, x_m / \sigma_2(t_m)\} \cup \{y_j / s_j\}$ Donde $j = 1 \dots n$,
 $y_j \notin \{x_1, x_2, \dots, x_m\}$ y se eliminan los elementos $x_i / \sigma_2(t_i)$ que cumplen $x_i = \sigma_2(t_i)$.

Ejemplo 25: Sea $\sigma_1 = \{x / f(y), y / z\}$ y $\sigma_2 = \{x / a, y / b, z / y\}$ entonces

$$\sigma_1 \circ \sigma_2 = \{x / f(b), z / y\}^9$$

Definición 40: La sustitución definida por el conjunto vacío se denomina **sustitución vacía** ó sustitución identidad y se denota por ε .

Propiedades de la Composición de Sustituciones

- **Asociativa:** $\sigma_1 \circ (\sigma_2 \circ \sigma_3) = (\sigma_1 \circ \sigma_2) \circ \sigma_3$ para todas las sustituciones $\sigma_1, \sigma_2, \sigma_3$

Esta propiedad permite omitir los paréntesis en la composición de n sustituciones

- **Elemento Neutro:** La sustitución vacía cumple $\sigma \circ \varepsilon = \varepsilon \circ \sigma = \sigma$ para toda sustitución σ .

- **No cumple la propiedad conmutativa.**

- $\sigma_2(\sigma_1(E)) = \sigma_1 \circ \sigma_2(E)$ para todas las sustituciones σ_1, σ_2 y las expresiones E .

Ejemplo 26: Sea $\sigma_1 = \{x / f(y), y / z\}$ y $\sigma_2 = \{x / a, z / b\}$.

$$\sigma_1 \circ \sigma_2 = \{x / f(y), y / b, z / b\}.$$

Sea $E = P(x, y, g(z))$, entonces:

$$\sigma_1(E) = P(f(y), z, g(z)) \text{ y } \sigma_2(\sigma_1(E)) = P(f(y), b, g(b))$$

$$\text{De la misma forma: } \sigma_1 \circ \sigma_2(E) = P(f(y), b, g(b))$$

Definición 41: Dadas dos expresiones, E y F , se dice que son **variantes** si existen dos sustituciones σ_1 y σ_2 tales que $E = \sigma_1(F)$ y $F = \sigma_2(E)$. También se dice que E es una variante de F o que F es una variante de E .

Ejemplo 27: $P(f(x, y), g(z), a)$ es una variante de $P(f(y, x), g(u), a)$. Sin embargo,

⁹Obsérvese que no se incluye el par y / y .



Lógica de Predicados E.U.I.T.I.O.



$P(x, x)$ no es una variante de $P(x, y)$.

Definición 42: Sea E una expresión y V el conjunto de variables que aparecen en E . Una **substitución de renombramiento** para E es una substitución de la forma $\{x_1 / y_1, \dots, x_n / y_n\}$ tal que $\{x_1, \dots, x_n\} \subseteq V$ e y_j son variables distintas entre sí que no pertenecen a V



Unificación

El concepto de unificación se debe a Herbrand que presentó, en su tesis de 1930, un algoritmo no determinista para calcular el unificador de dos términos. Sin embargo, sería Robinson, en 1965, quien lo volviera a rediseñar para aplicarlo al algoritmo de prueba de teoremas por resolución. El algoritmo de Robinson requería tiempo y espacio exponenciales y, desde entonces, se ha realizado un gran esfuerzo para mejorar su eficacia. Para un tratamiento detallado del algoritmo de unificación y sus aplicaciones a otros campos, véase [Knight 89].

Definición 43: Dado un conjunto finito de expresiones simples $C = \{E_1, E_2, \dots, E_n\}$ y una sustitución ω . Se dice que ω es un **unificador** para C si después de aplicar la sustitución, el conjunto es unitario. Es decir, si: $\omega(E_1) = \omega(E_2) = \dots = \omega(E_n)$

Ejemplo 28: Sea $C = \{P(f(x), z), P(y, a)\}$, entonces $\omega = \{x/a, y/f(a), z/a\}$ es un unificador de C , ya que $\omega(E_1) = \omega(E_2) = P(f(a), a)$

Sin embargo, el conjunto $\{P(f(x), a), P(y, f(w))\}$ no es unificable porque los segundos argumentos a y $f(w)$ no unifican.

Definición 44: Se dice que un unificador ω para un conjunto finito de expresiones simples es un **unificador más general** (umg) si para cualquier otro unificador ω' existe una sustitución σ que cumple $\omega' = \omega \circ \sigma$. De forma intuitiva, el unificador más general puede considerarse como el unificador más simple de todos los posibles unificadores.

Ejemplo 29: Sea $C = \{P(f(x), g(u, z)), P(y, g(v, a))\}$, se puede comprobar que $\omega = \{y/f(x), u/v, z/a\}$ es un unificador más general que cualquier otro unificador. Por ejemplo el unificador $\omega' = \{y/f(a), x/a, u/v, z/a\}$ puede ser obtenido a partir de ω aplicando la sustitución $\sigma = \{x/a\}$. Es decir, se cumple que $\omega' = \omega \circ \sigma$.

En el ejemplo anterior, se puede obtener otro umg: $\omega_2 = \{y/f(x), v/u, z/a\}$. Esto indica que un unificador no está únicamente determinado. Sin embargo se puede observar que ω y ω_2 están relacionados por dos sustituciones de renombramiento $\{v/u\}$ y $\{u/v\}$, ya que: $\omega = \omega_2 \circ \{u/v\}$ y $\omega_2 = \omega \circ \{v/u\}$, y por tanto, las cláusulas que resulten de aplicar dos umg distintos serán variantes.

Teorema 9: Si ω_1 y ω_2 son dos umg de un conjunto de expresiones C , entonces las expresiones $\omega_1(C)$ y $\omega_2(C)$ son variantes.



A continuación se presentará el **algoritmo de unificación**, que toma como entrada un conjunto finito de expresiones y devuelve el umg si las expresiones son unificables. En caso contrario informa que las expresiones no son unificables.

Intuitivamente se comporta de la siguiente forma. Supóngase que se quieren unificar dos expresiones, se toman dos punteros comenzando por la izquierda de cada una de las expresiones. Los punteros se mueven juntos hacia la derecha hasta que se encuentran dos símbolos diferentes. En ese momento, se intentan unificar las dos subexpresiones encontradas. Si se pudieron unificar, el proceso continúa con las dos expresiones tras aplicarles la nueva substitución, si no, las expresiones no son unificables y se acaba. Cuando los punteros alcancen el final de las dos expresiones, la composición de todas las substituciones será el umg.

Para la descripción del algoritmo es necesario definir qué es el conjunto de discrepancias.

Definición 45: Dado un conjunto de expresiones simples $E = \{E_1, E_2, \dots, E_n\}$, se define el **Conjunto de Discrepancias** o desacuerdos de E y se denota por $D(E)$, como el conjunto obtenido de la siguiente forma:

- Si E es unitario, entonces $D(E) = \emptyset$
- Si no:
 - Si las expresiones de E son todas "similares"¹⁰, es decir, E es de la forma:

$$E = \{f(t_{11}, t_{12}, \dots, t_{1n}), f(t_{21}, t_{22}, \dots, t_{2n}), \dots, f(t_{m1}, t_{m2}, \dots, t_{mn})\}$$

Se define $E_j = \{t_{1j}, t_{2j}, \dots, t_{mj}\}$ y $D(E) = D(E_j)$ para el menor j tal que $D(E_j) \neq \emptyset$

Si no $D(E) = E$.

Ejemplo 30: Sea $E = \{P(f(x), h(y), a), P(f(x), z, a), P(f(x), h(y), b)\}$, puesto que las tres expresiones de E son similares, se buscan el primer conjunto E_j cuyas discrepancias sean distintas del conjunto vacío. Se tiene:

¹⁰Se consideran expresiones similares, las que están formadas por una misma letra (de predicado o función) seguida de un mismo número n de argumentos, donde $n \geq 0$



$E_1 = \{f(x)\}$, que al ser unitario, sus discrepancias son igual al conjunto vacío.

$E_2 = \{h(y), z\}$, se buscan las discrepancias de E_2 , que, como no es unitario, ni las expresiones similares, quedará $D(E_2) = E_2 = \{h(y), z\}$

El menor j que cumple que $D(E_j) \neq \emptyset$ es 2 y, por tanto, $D(E) = D(E_2) = \{h(y), z\}$

Algoritmo de Unificación

1.- Inicialización: $k := 0$, $\sigma_0 := \varepsilon$

2.- Si $\sigma_k(E)$ consta de una sola fórmula atómica (* Ha sido unificado por σ_k *)

entonces: Salir y Devolver σ_k que será el umg de E

sino: $D_k := D(\sigma_k(E))$

3.- Si en D_k existe una variable v y un término t en el que no aparece la variable v entonces:

$$\sigma_{k+1} = \sigma_k \circ \{v/t\}$$

$$k = k + 1$$

Volver al paso 2.

sino: Salir y Devolver que el conjunto E no es unificable.

El algoritmo de unificación presentado contiene una indeterminación en el paso 3, ya que pueden existir varias posibilidades a la hora de seleccionar v y t . Sin embargo, la aplicación de cualquier umg producido por el algoritmo obtiene expresiones que sólo difieren en el nombre de las variables. Está claro que el algoritmo termina, ya que se tiene un número finito de variables y cada aplicación del paso 3 elimina una variable.



Ejemplo 31: Sea

$$E = \{P(f(a), g(x)), P(y, y)\}.$$

Paso 1: $k = 0$, $\sigma_0 = \varepsilon$.

Paso 2: $D_0 = \{f(a), y\}$

Paso 3: Selecciona $v = y$ y $t = f(a)$. $\sigma_1 = \{y / f(a)\}$, $k = 1$

Paso 2: $\sigma_1(E) = \{P(f(a), g(x)), P(f(a), f(a))\}$, $D_1 = \{g(x), f(a)\}$

Paso 3: No encuentra variable en D_1 sale e indica que NO son unificables.

Ejemplo 32: Sea $E = \{P(a, x, h(g(z))), P(z, h(y), h(y))\}$

Paso 1: $k = 0$, $\sigma_0 = \varepsilon$.

Paso 2: $D_0 = \{a, z\}$

Paso 3: Selecciona $v = z$, $t = a$. $\sigma_1 = \{z / a\}$, $k = 1$

Paso 2: $\sigma_1(E) = \{P(a, x, h(g(a))), P(a, h(y), h(y))\}$, $D_1 = \{x, h(y)\}$

Paso 3: Selecciona: $v = x$, $t = h(y)$. $\sigma_2 = \{x / h(y)\}$. $k = 2$

Paso 2: $\sigma_2(E) = \{P(a, h(y), h(g(a))), P(a, h(y), h(y))\}$. $D_2 = \{y, g(a)\}$

Paso 3: Selecciona $v = y$, $t = g(a)$. $\sigma_3 = \{y / g(a)\}$, $k = 3$.

Paso 2: $\sigma_3(E) = \{P(a, h(g(a)), h(g(a)))\}$ es unitario. Salir y devolver σ_3 que será el umg de E .

En el paso 3 del algoritmo de unificación, se establece el chequeo de que el término no contenga la variable. Ese chequeo se conoce como "chequeo de ocurrencias"(occur check). A continuación se da un ejemplo de su uso.



Ejemplo 33: Sea $E = \{P(x, x), P(y, f(y))\}$.

Paso 1: $k = 0$, $\sigma_0 = \varepsilon$.

Paso 2: $D_0 = \{x, y\}$

Paso 3: Selecciona $v \equiv x$, $t \equiv y$. $\sigma_1 = \{z / y\}$, $k = 1$

Paso 2: $\sigma_1(E) = \{P(y, y), P(y, f(y))\}$, $D_1 = \{y, f(y)\}$

Paso 3: Puesto que y aparece en $f(y)$, el conjunto NO es unificable.

Robinson demostró en lo que se conoce como **teorema de unificación** que el algoritmo de unificación para y devuelve un umg si el conjunto de expresiones es unificable o para e indica que el conjunto no es unificable si éste es el caso.

El algoritmo de unificación puede llegar a ser muy ineficiente. En el peor de los casos, su tiempo de ejecución puede llegar a ser una función exponencial de la longitud de la entrada.

Ejemplo 34: Sea $E = \{P(x_1, \dots, x_n), P(f(x_0, x_0), f(x_1, x_1), \dots, f(x_{n-1}, x_{n-1}))\}$.



Algoritmo de Resolución General

En lógica de Primer Orden el algoritmo de resolución requiere un paso previo de unificación de los literales por los que se resuelve. A continuación se exponen las definiciones del algoritmo de resolución general con unificación y se demostrará que es consistente y parcialmente completo.

Definición 46: Dadas dos cláusulas C_1 y C_2 sin variables comunes, tales que $l_1 \in C_1$ y $\neg l_2 \in C_2$ son dos literales que pueden ser unificados por un umg ω . Se dice entonces que C_1 y C_2 son **cláusulas resolubles** y se define el **resolvente de C_1 y C_2 respecto a l_1 y l_2** como la cláusula:

$$R_{l_1, l_2}(C_1, C_2) = \{\omega(C_1) - \omega(l_1)\} \cup \{\omega(C_2) - \omega(\neg l_2)\}$$

Ejemplo 35: Considérense las cláusulas

$$C_1 = P(f(x), g(y)) \vee Q(x, y) \vee \neg R(x) \text{ y } C_2 = \neg P(f(f(a)), g(z)) \vee Q(f(a), z)$$

Tomando los literales $l_1 = P(f(x), g(y))$ y $l_2 = P(f(f(a)), g(z))$ se encuentra un umg

$$\omega = \{x / f(a), z / y\}$$

Entonces, el resolvente de C_1 y C_2 respecto a l_1 y l_2 será:

$$R_{l_1, l_2}(C_1, C_2) = Q(f(a), y) \vee \neg R(f(a)) \vee Q(f(f(a)), y)$$

Ejemplo 36: Sean las cláusulas: $C_1 = P(x)$ y $C_2 = \neg P(a)$. Unificando los dos únicos literales, se obtiene el umg $\omega = \{x / a\}$ y el resolvente $R(C_1, C_2) = \square$ será la cláusula vacía.

Se demostrará que, si el resolvente de dos cláusulas de un conjunto es la cláusula vacía, entonces ese conjunto es insatisfacible. En el ejemplo, está claro que el conjunto $\{P(x), \neg P(a)\}$ es insatisfacible ya que representa la fórmula $\forall x P(x) \wedge \neg P(a)$.

Para que el método sea completo (ver pág. 42), las cláusulas no deben tener variables en común. Esto se logra manteniendo las cláusulas en su forma original y, a la hora de utilizarlas, substituyendo todas sus variables por nombres de variable nuevos, es decir, se aplica una substitución de renombramiento para obtener una cláusula variante. Se recuerda que las variables en una cláusula están cuantificadas implícitamente de forma universal luego el renombramiento de variables no cambia la satisfacibilidad del conjunto.

Ejemplo 37: Sea el conjunto $\{P(f(x)), \neg P(x)\}$.



Lógica de Predicados E.U.I.T.I.O.



Las cláusulas $C_1 = P(f(x))$ y $C_2 = \neg P(x)$ no pueden ser resueltas ya que no unifican los literales debido al chequeo de ocurrencias. Sin embargo, si se renombran sus variables, se tiene:

$$C_1^v = P(f(x_1)) \text{ y } C_2^v = \neg P(x_2)$$

Utilizando el umg $\omega = \{x_2 / f(x_1)\}$, queda: $R(C_1^v, C_2^v) = \square$

Puesto que se obtiene la cláusula vacía, el conjunto $\{P(f(x_1)), P(x_2)\}$ es insatisfacible, y el conjunto original: $\{P(f(x)), P(x)\}$ también será insatisfacible. Este resultado es correcto, ya que, el conjunto anterior, teniendo en cuenta la cuantificación universal de las variables, sería:

$$\forall x P(f(x)) \wedge \forall x \neg P(x)$$

Ese conjunto, es, evidentemente, insatisfacible¹¹.

Utilizando la resolución se puede comprobar que si un conjunto de cláusulas es insatisfacible (teorema de completud, pág 42) se podrán ir generando resolventes de una cláusula con otra hasta que se llegue a la cláusula vacía. A continuación se describe el algoritmo general de resolución:

Algoritmo de resolución general

Entrada: Un conjunto de cláusulas C

Salida: Detecta si C es insatisfacible

- 1.- Buscar dos cláusulas $C_1, C_2 \in C$ resolubles
- 2.- Si se encuentran:
 - 3.- Calcular $R(C_1, C_2)$ y añadirlo al conjunto C
 - 4.- Si $R(C_1, C_2) = \square$ entonces **SALIR** indicando que C es **insatisfacible**.
 - 5.- Si no, Añadir $R(C_1, C_2)$ a C y Volver a 1
- 3.- Si no se encuentran: **SALIR** indicando que C **no es insatisfacible**.

¹¹El proceso de renombramiento, no debe confundirse con el chequeo de ocurrencias. El cual previene la unificación **dentro** de una cláusula: $P(f(x)) \vee \neg P(x)$. Esa cláusula representa la fórmula $\forall x (P(f(x)) \vee \neg P(x))$ que no es válida.

Ejemplo 38: A continuación se describe el comportamiento del algoritmo con el conjunto:

$$\left\{ \begin{array}{l} \neg P(x) \vee Q(x) \vee R(x, f(x)), \neg P(x) \vee Q(x) \vee S(f(x)), \\ T(a), P(a), \neg R(a, y) \vee T(y), \neg T(x) \vee \neg Q(x), \neg T(x) \vee \neg S(x) \end{array} \right\}$$

En las líneas 1-7 se escriben de nuevo las cláusulas del conjunto. A continuación (líneas 8 a 15) aparecen las distintas Cláusulas C (resolvente) junto con el umg utilizado y los números de cláusulas resolubles.

1	$\neg P(x) \vee Q(x) \vee R(x, f(x))$		
2	$\neg P(x) \vee Q(x) \vee S(f(x))$		
3	$T(a)$		
4	$P(a)$		
5	$\neg R(a, y) \vee T(y)$		
6	$\neg T(x) \vee \neg Q(x)$		
7	$\neg T(x) \vee \neg S(x)$		
8	$\neg Q(a)$	$\{x/a\}$	3-6
9	$Q(a) \vee S(f(a))$	$\{x/a\}$	2-4
10	$S(f(a))$	ε	8-9
11	$Q(a) \vee R(a, f(a))$	$\{x/a\}$	1-4
12	$R(a, f(a))$	ε	8-11
13	$T(f(a))$	$\{y/f(a)\}$	5-12
14	$\neg S(f(a))$	$\{x/f(a)\}$	7-13
15	\square	ε	10-14

Puesto que se obtiene la cláusula vacía el algoritmo termina en el paso 3 indicando que el conjunto de cláusulas es insatisfacible

El siguiente paso consistirá en demostrar que el sistema de razonamiento que utiliza el principio de resolución es consistente y parcialmente completo¹².

¹²Normalmente se dice que es refutacionalmente completo. Es decir, completo para refutaciones o derivaciones de la cláusula vacía.



Teorema 10 (Consistencia del principio de resolución): Si C es el resolvente de C_1 y C_2 entonces C es consecuencia lógica de C_1 y C_2 .

Dem: Puesto que una sustitución significa particularizar las variables de una expresión. La instancia de cualquier expresión E será consecuencia lógica de E , es decir, para cualquier sustitución σ se cumple que $E \Rightarrow \sigma(E)$

En particular el umg de l_1 y l_2 es una sustitución ω y también se cumplirá que

$$\begin{aligned} C_1 &\Rightarrow \omega(C_1) \\ C_2 &\Rightarrow \omega(C_2). \end{aligned} \quad [1]$$

Ahora resta probar que $\omega(C_1) \wedge \omega(C_2) \Rightarrow C$

Se tiene: $C_1 = l_1 \vee RC_1$ y que $C_2 = \neg l_2 \vee RC_2$ (Recuérdese que $l_1 \in C_1$ y que $\neg l_2 \in C_2$)

donde RC_1 y RC_2 representan el resto de literales de C_1 y C_2 .

Si ω es un umg de l_1 y l_2 entonces $\omega(l_2) = \omega(l_1)$

$$\omega(C_1) = \omega(l_1) \vee \omega(RC_1)$$

$$\omega(C_2) = \neg \omega(l_2) \vee \omega(RC_2) = \neg \omega(l_1) \vee \omega(RC_2)$$

Por otra parte,

$$C = R_{l_1, l_2}(C_1, C_2) = \{\omega(C_1) - \omega(l_1)\} \cup \{\omega(C_2) - \omega(\neg l_2)\} = \omega(RC_1) \vee \omega(RC_2)$$

Supóngase que el resolvente no es consecuencia lógica de C_1 y C_2 , entonces, se tendrían los siguientes valores de verdad:

$$\begin{array}{ccccccc} \underbrace{\{\omega(l_1) \vee \omega(RC_1)\}}_{?} & \underbrace{\wedge}_{\text{F}} & \underbrace{\{\neg \omega(l_2) \vee \omega(RC_2)\}}_{?} & \underbrace{\rightarrow}_{\text{F}} & \underbrace{\omega(RC_1) \vee \omega(RC_2)}_{\text{F}} \\ \underbrace{\hspace{10em}}_{\text{V}} & & \underbrace{\hspace{10em}}_{\text{F}} & & \\ \underbrace{\hspace{15em}}_{\text{F}} & & & & \end{array}$$



$\omega(l_1)$ no puede tomar valor **F**, ya que la expresión del primer corchete sería falsa (tiene que ser **V**). De la misma forma, tampoco puede tomar valor **V** pues haría falsa la expresión del segundo corchete. Por tanto, la fórmula anterior no puede tomar valor **F** y será válida. Se cumple que:

$\omega(C_1) \wedge \omega(C_2) \Rightarrow C$ y, uniendo con [¡Error! Marcador no definido.] se tiene:

$$\left. \begin{array}{l} C_1 \Rightarrow \omega(C_1) \\ C_2 \Rightarrow \omega(C_2) \end{array} \right\} \Rightarrow C$$

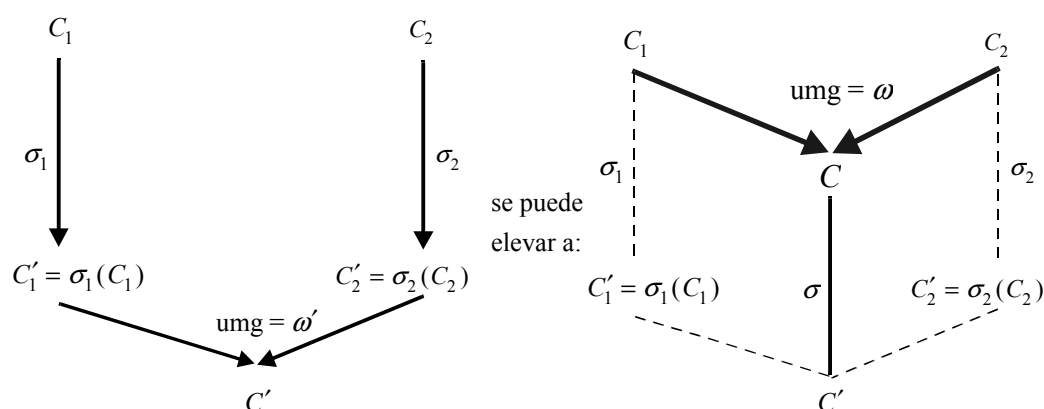
Falta demostrar que el algoritmo de resolución es parcialmente completo en el sentido de que, cuando el conjunto de cláusulas es insatisfacible, encuentra la cláusula vacía.

Los pasos a seguir en la demostración son:

- Utilizando el teorema de Herbrand, si un conjunto de cláusulas es insatisfacible entonces existe un subconjunto finito de cláusulas básicas insatisfacible.

- Puesto que son cláusulas básicas, se pueden considerar como proposiciones con valor **V/F**. Se ha demostrado que al aplicar el algoritmo de resolución sobre un conjunto de proposiciones insatisfacibles se obtiene la cláusula vacía (teorema de completud proposicional).

Teorema 11 (Lema de elevación): Si C_1 y C_2 son cláusulas sin variables comunes y σ_1 y σ_2 dos sustituciones tales que $C'_1 = \sigma_1(C_1)$ y $C'_2 = \sigma_2(C_2)$ no tienen variables en común y son resolubles mediante un umg ω' en una cláusula C' entonces existe (puede *elevarse*) una cláusula C y dos sustituciones ω y σ tales que C es el resolvente de C_1 y C_2 mediante un umg ω y $C' = \sigma(C)$ (C' es una instancia de C). Véase el esquema de la figura:





- Una vez obtenida la cláusula vacía sobre el conjunto de cláusulas básicas se extrapola el resultado al conjunto original de cláusulas, para lo cual se emplea el lema de elevación.

Dem: Supóngase que:

$$C_1 = \{l_1 \vee l_2 \vee \dots \vee l_m\} \cup RC_1$$

$$C_2 = \{\neg l_{m+1} \vee \neg l_{m+2} \vee \dots \vee \neg l_{m+n}\} \cup RC_2$$

Donde l_i serán los literales que unificarán, RC_1 y RC_2 representan el resto de literales de cada cláusula y, puesto que tiene que existir al menos un literal en cada cláusula $m > 0$ y $n > 0$.

Se cumplirá que:

$$C'_1 = \sigma_1(C_1) = \{\sigma_1(l_1) \vee \dots \vee \sigma_1(l_m)\} \cup \sigma_1(RC_1)$$

$$C'_2 = \sigma_2(C_2) = \{\sigma_2(\neg l_{m+1}) \vee \dots \vee \sigma_2(\neg l_{m+n})\} \cup \sigma_2(RC_2)$$

ω' es un umg de los literales: $\{\sigma_1(l_1), \dots, \sigma_1(l_m), \sigma_2(l_{m+1}), \dots, \sigma_2(l_{m+n})\}$ y el resolvente será:

$$C' = \omega'(\sigma_1(RC_1)) \cup \omega'(\sigma_2(RC_2)) \quad [2]$$

Puesto que σ_1 y σ_2 no tienen variables en común y $\sigma_1(C_1)$ y $\sigma_2(C_2)$ tampoco, la substitución $\omega' \circ (\sigma_1 \cup \sigma_2)$ será un unificador de $\{l_1, \dots, l_m, l_{m+1}, \dots, l_{m+n}\}$ ya que:

$$\begin{aligned} &(\sigma_1 \cup \sigma_2)\{l_1, \dots, l_m, l_{m+1}, \dots, l_{m+n}\} = \\ &\{\sigma_1 \cup \sigma_2(l_1), \dots, \sigma_1 \cup \sigma_2(l_m), \sigma_1 \cup \sigma_2(l_{m+1}), \dots, \sigma_1 \cup \sigma_2(l_{m+n})\} = \\ &\{\sigma_1(l_1), \dots, \sigma_1(l_m), \sigma_2(l_{m+1}), \dots, \sigma_2(l_{m+n})\} \end{aligned}$$

Y ese conjunto era unificable por ω' .

Si existe un unificador, existirá un umg, sea ω el umg de $\{l_1, \dots, l_m, l_{m+1}, \dots, l_{m+n}\}$, entonces, por la definición de umg cumplirá que, para cualquier otro unificador (en este caso $\omega' \circ (\sigma_1 \cup \sigma_2)$) existe una substitución σ tal que

$$\omega' \circ (\sigma_1 \cup \sigma_2) = \sigma \circ \omega \quad [3]$$

Sea $C = \omega(RC_1) \cup \omega(RC_2)$ el resolvente de C_1 y C_2 respecto a ω , entonces:

$$\sigma(C) =$$

$$\sigma(\omega(RC_1) \cup \omega(RC_2)) =$$

$$\sigma \circ \omega(RC_1 \cup RC_2) = \quad (\text{aplicando [2]})$$



$$\omega' \circ (\sigma_1 \cup \sigma_2)(RC_1 \cup RC_2) =$$

$$\omega' \circ (\sigma_1 \cup \sigma_2)(RC_1) \cup \omega' \circ (\sigma_1 \cup \sigma_2)(RC_2) =$$

(Puesto que C_1 y C_2 no tienen variables en común, la substitución σ_1 sólo afecta a C_1 y no tiene efecto sobre RC_2 (de igual forma, σ_2 no afecta a RC_1), por tanto:
 $\sigma_1 \cup \sigma_2(RC_1) = \sigma_1(RC_1)$ y $\sigma_1 \cup \sigma_2(RC_2) = RC_2$)

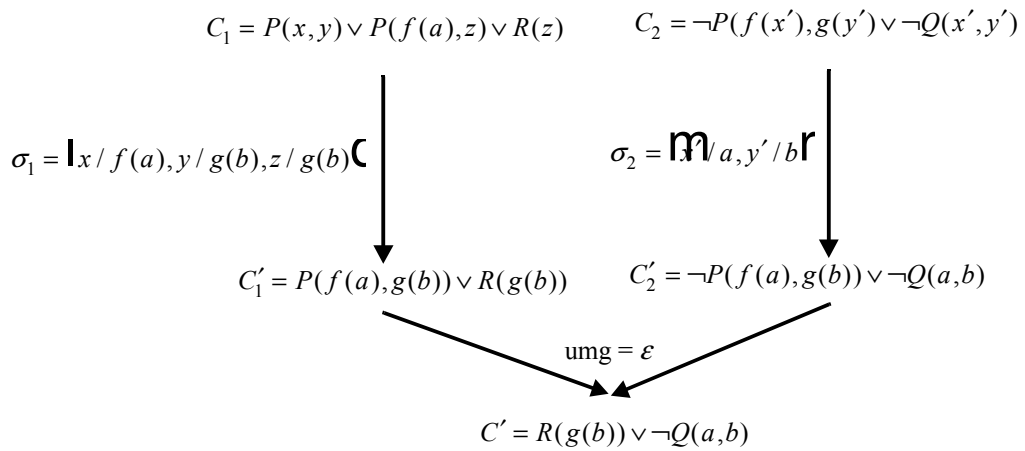
$$= \omega'(\sigma_1(RC_1)) \cup \omega'(RC_2) =$$

(aplicando [¡Error! Marcador no definido.])

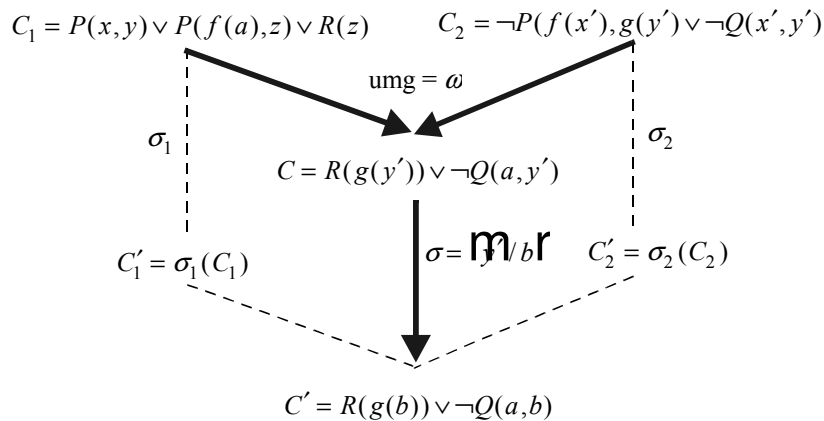
$$= C'$$

Por tanto $\sigma(C) = C'$, es decir, C' es una instancia de C

Ejemplo 39: Sea $C_1 = P(x, y) \vee P(f(a), Z) \vee R(z)$ y $C_2 = \neg P(f(x'), g(y')) \vee \neg Q(x', y')$ y las substituciones: $\sigma_1 = \{x / f(a), z / f(y)\}$ y $\sigma_2 = \{x' / a, y' / b\}$. Se obtiene el esquema:



Siguiendo el lema de elevación, existirá una cláusula C resolvente de C_1 y C_2 con $\text{umg } \omega = \{x / f(a), y / g(y'), z / g(y'), x' / a\}$ que seguirá el esquema:



Con el lema de elevación queda demostrado el teorema de completud:

Teorema 12 (Compleitud del principio de resolución general): Si un conjunto de cláusulas es insatisfacible, entonces, aplicando el algoritmo de resolución general se obtendrá la cláusula vacía.



Dem: Sea C el conjunto de cláusulas insatisfacibles, entonces se tiene la siguiente cadena de equivalencias:

C es insatisfacible

\Updownarrow (por el teorema de Herbrand)

Existe un subconjunto finito de cláusulas básicas de C insatisfacible

\Downarrow (Compleitud de resolución proposicional)

Existe una derivación de un conjunto de cláusulas básicas de C a la cláusula vacía.

\Downarrow (lema de elevación)

Existe una derivación del conjunto C de cláusulas a la cláusula vacía.

NOTA: en la última implicación, para poder aplicar el lema de elevación, se supone que las cláusulas de C no tienen variables en común.

Si no se utilizan variantes, el algoritmo no es completo, así, por ejemplo, el conjunto: $\{P(x), \neg P(f(x))\}$ es insatisfacible, sin embargo, $P(x)$ y $P(f(x))$ no son unificables a no ser que se utilicen variantes



Estrategias de resolución

El método de resolución es un algoritmo no determinista ya que pueden encontrarse múltiples formas de alcanzar la cláusula vacía en un conjunto insatisfacible. Muchas veces, siguiendo un determinado camino se alcanzará la cláusula vacía con muchos menos pasos de resolución que por otro camino.

Durante el desarrollo del algoritmo es necesario responder las siguientes preguntas: ¿Qué dos cláusulas se seleccionan y sobre qué literales se realiza la resolución?

Las distintas estrategias de resolución tratan de responder a ambas preguntas de forma que se mantenga la completud (si el conjunto es insatisfacible, alcanzar la cláusula vacía) y que se obtenga un comportamiento eficiente.

Una de las desventajas de la utilización de la reglas de resolución sin ninguna restricción consiste en que se pueden seleccionar cláusulas cuyo resolvente no sea útil en el camino de búsqueda de la cláusula vacía. Se observa que muchas veces los resolventes son redundantes o no aportan ninguna utilidad para la búsqueda. A continuación se mencionan una serie de estrategias que servirán para eliminar el trabajo inútil.

Estrategias de Borrado

Una estrategia de borrado será una técnica en la cual se eliminan una serie de cláusulas antes de que sean utilizadas. Si dichas cláusulas no van a aportar nada para la búsqueda de la cláusula vacía, su eliminación permitirá un ahorro computacional.

Eliminación de cláusulas con literales puros

Definición 47: Un literal es **puro** si y sólo si no existe un literal complementario con el que unifique en el conjunto de cláusulas.

Una cláusula que contenga un literal *puro* es inútil en la búsqueda de la cláusula vacía, puesto que el literal *puro* no podrá ser eliminado nunca mediante resolución. Por tanto, una estrategia de borrado consiste en la eliminación de cláusulas con literales puros.

Ejemplo 40: El conjunto $C = \{P(a), \neg P(x) \vee P(f(x)), \neg P(a) \vee Q(a), \neg Q(a)\}$ es insatisfacible, sin embargo, para demostrarlo, se puede ignorar la segunda cláusula, puesto que ambas contienen el literal puro $P(f(x))$.

Eliminación de tautologías **Definición 48:** Una **tautología** es una cláusula que contiene el mismo literal en su forma directa e inversa.

Ejemplo 41: La cláusula $p \vee \neg q \vee r \vee \neg p$ es una *tautología*.

La presencia o ausencia de tautologías en un conjunto de cláusulas no afecta la condición de satisfacibilidad del conjunto. Un conjunto de cláusulas permanecerá satisfacible independientemente de que se le añadan tautologías. De la misma forma, un conjunto de cláusulas insatisfacible seguirá siendo insatisfacible aunque se eliminen todas sus



tautologías. Es posible, por tanto, eliminar las tautologías de un conjunto de cláusulas para que no intervengan en el proceso de búsqueda sin alterar la satisfacibilidad del conjunto.

Obsérvese que los literales de una cláusula deben ser exactamente complementarios para poder aplicar la eliminación de tautologías. No se pueden eliminar literales no idénticos simplemente porque sean unificables.

Ejemplo 42: El conjunto $\{\neg P(a) \vee P(x), P(a), \neg P(b)\}$ es insatisfacible. Sin embargo, si se elimina la primera cláusula el conjunto resultante sería satisfacible.

Eliminación de Subsunciones **Definición 49:** Una cláusula C **subsume** a una cláusula D si existe una sustitución σ tal que $\sigma(C) \subseteq D$.

Ejemplo 43: En el conjunto $\{P(x) \vee \neg Q(y), P(a) \vee \neg Q(v) \vee R(w), \neg P(b), Q(a)\}$. La cláusula $P(x) \vee \neg Q(y)$ subsume a $P(a) \vee \neg Q(v) \vee R(w)$ puesto que la cláusula resultante de aplicar la sustitución $\{x/a, y/v\}$ a la primera cláusula, está incluida en la segunda cláusula. De esa forma, se puede eliminar la segunda cláusula, sin alterar la insatisfacibilidad del conjunto.

Se puede demostrar que si una cláusula D de un conjunto de cláusulas es subsumida por otra, entonces el conjunto de cláusulas tras eliminar D es satisfacible si y sólo si el conjunto original de cláusulas lo era. Las cláusulas subsumidas pueden ser, por tanto, eliminadas sin modificar la condición de satisfacibilidad del conjunto.

Es necesario observar que, durante el desarrollo del proceso de resolución, se pueden generar resolventes de cláusulas que sean *tautologías* o *cláusulas subsumidas*. Las estrategias de borrado deberán chequear el conjunto de cláusulas original así como los distintos resolventes generados en cada resolución.

Resolución Unitaria

Definición 50: Un resolvente **unitario** es un resolvente en el cual al menos uno de sus padres es una cláusula unitaria (con un sólo literal).

Una **estrategia de resolución unitaria** es una aplicación del algoritmo de resolución en la cual todos los resolventes son unitarios.

Ejemplo 44: Sea $C = \{p \vee q, \neg p \vee r, \neg q \vee r, \neg r\}$. A continuación se aplicará la estrategia de resolución unitaria, para ello, se seleccionan siempre dos cláusulas resolubles tales que una de ellas tenga un literal.

- | | |
|-------------------------|---------------------------|
| 1.- $p \vee q$ | 7.- q $R_p(1,5)$ |
| 2.- $\neg p \vee r$ | 8.- p $R_q(1,6)$ |
| 3.- $\neg q \vee r$ | 9.- r $R_q(3,7)$ |
| 4.- $\neg r$ | 10.- \square $R_q(6,7)$ |
| 5.- $\neg p$ $R_r(2,4)$ | |
| 6.- $\neg q$ $R_r(3,4)$ | |

Obsérvese que los resolventes generados son un subconjunto de los que se podrían generar mediante la resolución sin restricciones. Por ejemplo, las cláusulas 1 y 2 podrían

haberse seleccionado para obtener $q \vee r$. Sin embargo ni esa cláusula ni sus descendientes podrán ser generados porque ninguna de las cláusulas que la generan es unitaria.



Los procedimientos de resolución basados en resolución unitaria son sencillos de implementar y, normalmente, bastante eficientes. Obsérvese que si una cláusula es resuelta con una cláusula unitaria, su resolvente tiene menos literales que la cláusula original. De esa forma los procedimientos siguen una búsqueda directa hacia la cláusula vacía ganando en eficiencia.

Desafortunadamente, los procedimientos de inferencia basados en resolución unitaria no son, en general, completos. Por ejemplo, el conjunto $C = \{p \vee q, \neg p \vee q, p \vee \neg q, \neg p \vee \neg q\}$ es insatisfacible, sin embargo, la resolución unitaria no encontrará la cláusula vacía porque ninguna de las cláusulas es unitaria.

Por otro lado, restringiendo el formato de cláusulas a cláusulas Horn (cláusulas con un literal positivo como máximo) se puede demostrar que si un conjunto de cláusulas Horn es insatisfacible, entonces se llegará a la cláusula vacía aplicando la estrategia de resolución unitaria.

Resolución de Entrada

Definición 51: Un **resolvente de entrada** es un resolvente en el cual al menos uno de sus padres es una cláusula del conjunto original de entrada.

Una **estrategia de resolución de entrada** es una aplicación del algoritmo de resolución en la cual todos los resolventes son de entrada.

Ejemplo 45: Sea $C = \{p \vee q, \neg p \vee r, \neg q \vee r, \neg r\}$. A continuación se aplicará la estrategia de resolución de entrada, para ello, se seleccionan siempre dos cláusulas resolubles tales que una de ellas pertenezca al conjunto inicial de cláusulas:

- | | |
|---------------------------|--------------------------|
| 1.- $p \vee q$ | 7.- $\neg p$ $R_r(2,4)$ |
| 2.- $\neg p \vee r$ | 8.- r $R_p(2,6)$ |
| 3.- $\neg q \vee r$ | 9.- \square $R_r(4,8)$ |
| 4.- $\neg r$ | |
| 5.- $q \vee r$ | |
| $R_p(1,2)$ | |
| 6.- $p \vee r$ $R_q(1,3)$ | |

Se puede demostrar que la resolución unitaria y la resolución de entrada tienen el mismo poder de inferencia en el sentido de que si con una estrategia se puede alcanzar la cláusula vacía, con la otra también.

Una consecuencia de lo anterior es que la resolución de entrada es completa para cláusulas Horn, pero incompleta en general. Como contraejemplo, se puede tomar el del apartado anterior.

Resolución Lineal

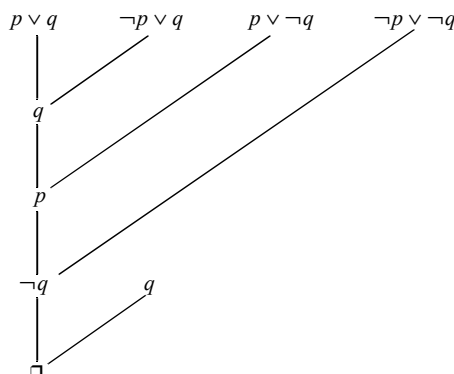
La resolución lineal (también conocida como resolución con filtrado de antepasados) es una generalización de la resolución de entrada en la cual al menos una de cada par de cláusulas a resolver pertenece al conjunto inicial de entrada o es un antepasado del otro padre.



Definición 52: Sea S un conjunto de cláusulas y C un elemento de ese conjunto. Entonces una **derivación por resolución lineal de C_n desde S con cláusula inicial C** es una secuencia de cláusulas de la forma $C_0 = C, C_1, \dots, C_n$ tales que para cada $i=0,1,\dots,n-1$:

- (1) C_{i+1} es un resolvente de C_i (llamada *cláusula central*) y de otra cláusula B_i (llamada *cláusula lateral*), y
- (2) Cada B_i o pertenece a S (sería una cláusula de entrada) o es un antepasado de C_i .

La resolución lineal toma su nombre del aspecto lineal que presentan las inferencias realizadas. Una resolución lineal comienza con una cláusula del conjunto inicial (llamada **cláusula cabeza**) y produce una cadena lineal de resoluciones como la que se muestra en la figura para el conjunto de cláusulas $C = \{p \vee q, \neg p \vee q, p \vee \neg q, \neg p \vee \neg q\}$. Obsérvese que cada resolvente, después del primero, se obtiene del resolvente anterior y de alguna otra cláusula del conjunto inicial o de sus antepasados.



La resolución lineal evita muchas resoluciones inútiles centrándose en cada paso en los antepasados de una cláusula y en los elementos del conjunto inicial.

Se puede demostrar que la resolución lineal es completa. Además, no es necesario probar con todas las cláusulas del conjunto inicial como cláusulas cabeza ya que si un conjunto de cláusulas S es satisfacible y $S \cup \{C\}$ es insatisfacible, entonces se encuentra la cláusula vacía mediante resolución lineal tomando C como cláusula cabeza. Por tanto, si se sabe que un conjunto de cláusulas S es satisfacible, no es necesario probar con los elementos de S como cláusulas cabeza.

Resolución Ordenada

La resolución *ordenada* o *selectiva* es una estrategia de resolución muy restrictiva en la cual cada cláusula se toma como un conjunto de literales ordenados. La resolución sólo se realiza con el primer literal de cada cláusula. Los literales del resolvente mantienen el orden de las cláusulas padre con los literales del padre positivo (la cláusula que contenía el literal por el que se resuelve afirmado) seguidos de los literales del padre negativo (la cláusula que contenía el literal por el que se resuelve negado).



Ejemplo 46: Sea $C = \{p \vee q, \neg p \vee r, \neg q \vee r, \neg r\}$. A continuación se aplicará la estrategia de resolución ordenada (se han ordenado los literales de cada cláusula por orden alfabético):

- | | | |
|---------------------|----------------|------------|
| 1.- $p \vee q$ | 5.- $q \vee r$ | $R_p(1,2)$ |
| 2.- $\neg p \vee r$ | | |
| 3.- $\neg q \vee r$ | 6.- r | $R_q(3,5)$ |
| 4.- $\neg r$ | 7.- \square | $R_r(4,6)$ |

La cláusula 5 es el único resolvente ordenado entre las cláusulas 1 y 4. Las cláusulas 1 y 3 no resuelven puesto que sus literales complementarios no son los primeros. Por la misma razón tampoco resuelven las cláusulas 2 y 4 ni las cláusulas 3 y 4. Una vez generada la cláusula 5, resuelve con la cláusula 3 para producir la cláusula 6, la cual resuelve con la cláusula 4 para producir la cláusula vacía.

La resolución ordenada es la más eficiente (en el ejemplo, se obtuvo la cláusula vacía en el tercer paso de resolución). Desafortunadamente, la resolución ordenada no es completa. Sin embargo, se ha demostrado que la resolución ordenada sí es completa para cláusulas Horn.

Tras este breve repaso de las principales estrategias de resolución, cabe reseñar que los principales sistemas de demostración automática utilizan una combinación de las dos últimas estrategias restringidas a conjuntos de cláusulas Horn. En particular, los sistemas de Programación Lógica utilizan la resolución lineal ordenada conocida como resolución SLD (*Selective Linear resolution for Definite Clauses*) que se definirá en la siguiente sección.



Prueba de Teoremas por Resolución

Una de las primeras aplicaciones de los ordenadores para la **computación simbólica**¹³ fue la demostración automática de teoremas.

Una **teoría** está formada por un conjunto de consecuencias lógicas (teoremas) que se siguen a partir de un conjunto inicial de fórmulas (axiomas) mediante reglas de inferencia. La axiomatización de las teorías matemáticas fue estudiada de forma intensiva en las décadas que precedieron la aparición de los ordenadores. Se consideró que los ordenadores serían capaces de tomar como entrada un conjunto de axiomas y una fórmula y chequear si la fórmula era consecuencia lógica de los axiomas.

El cálculo de predicados es el lenguaje formal utilizado universalmente para expresar las distintas teorías. En 1900, Hilbert consideraba que se podría llegar a axiomatizar toda la teoría matemática, de forma que probar nuevos teoremas sería una cuestión mecanizable.

Sin embargo, en 1931, Gödel demostró que cualquier sistema suficientemente general como para contener la teoría elemental de los números naturales no es completo. Es decir contiene fórmulas cuya validez o insatisfacibilidad no puede ser demostrada.

No existe, por tanto, un procedimiento general para decidir si una fórmula es consecuencia lógica de un conjunto de axiomas utilizando el cálculo de predicados, de ahí que **no todos** los teoremas puedan ser descubiertos. Sin embargo, existen algoritmos de decisión para subconjuntos limitados como el cálculo proposicional o teorías geométricas sencillas, donde se ha aplicado el ordenador como herramienta capaz de probar nuevos teoremas.

Ejemplo 47: A continuación se muestran las cláusulas resultantes de una posible axiomatización de la teoría de números naturales con las operaciones suma y producto. Para ello se utiliza la constante 0, la función $s(x) = "x+1"$ y los predicados: $\text{suma}(x, y, z) = "z \text{ es igual a } x + y"$ y $\text{producto}(x, y, z) = "z \text{ es igual a } x \cdot y"$

1.- $\text{suma}(x, 0, x)$

2.- $\neg \text{suma}(x, y, z) \vee \text{suma}(x, s(y), s(z)) \quad x + (y + 1) = (x + y) + 1$

3.- $\text{producto}(x, 0, 0)$

4.- $\neg \text{producto}(x, y, v) \vee \neg \text{suma}(x, v, z) \vee \text{producto}(x, s(y), z) \quad x * (y + 1) = x * y + x$

¹³La computación simbólica utiliza el ordenador para manipular símbolos, se opone a la computación numérica que utiliza el ordenador como máquina de cómputo de cantidades numéricas. Un ejemplo característico consistiría en la diferencia entre el cálculo de la función derivada (la solución es una fórmula) respecto al cálculo de la derivada en un punto determinado (la solución es una cantidad)



Lógica de Predicados E.U.I.T.I.O.



A partir de los axiomas anteriores, se podrían intentar demostrar algunos teoremas sencillos de la teoría de números. Por ejemplo, para saber si existe un número X cuyo producto por sí mismo sea igual a uno. Se plantearía la conclusión: su producto se intentaría concluir:

$$\exists x(\text{producto}(x, x, s(0)))$$

Para comprobarlo, se añadiría la negación de esa conclusión al conjunto de cláusulas y, si al aplicar resolución se obtiene la cláusula vacía, se deduce que existe tal número.



Bibliografía



- [Ben-Ari, 93] M. Ben-Ari
Mathematical Logic for Computer Science
Prentice Hall Intl. (1993)
- [Cuenca 85] José Cuenca
Lógica Informática
Alianza Informática (1985)
- [Genesereth, 87] M. R. Genesereth, N.J. Nilsson
Logical foundations of Artificial Intelligence
Morgan Kaufmann Publishers, Inc. (1987)
- [ISO 93] Comité ISO Internacional
PROLOG. Part 1, General Committee Draft (Borrador Normativa ISO).
Referencia: ISO/IEC JTC1 SC22 WG17. N110. Marzo 1993
- [Knight 89] Kevin Knight
Unification: A Multidisciplinary Survey
ACM Computing Surveys, Vol. 21 No. 1, Marzo 1989. pp. 92-124
- [Kowalski 79] R. Kowalski
Logic for Problem Solving
North Holland. Traducción española: Lógica, Programación e Inteligencia Artificial. (1979)
- [Kumar 92] Subrata Kumar Das
Deductive Databases and Logic Programming
Addison-Wesley (1992)
- [Lloyd 87] John Wylie Lloyd
Foundations of Logic Programming.
Springer Verlag . 2nd. Ed. (1987)
- [Maier, 88] David Maier, David S. Warren
Computing with Logic
The Benjamin/Cummings Publishing Company, Inc. (1988)
- [Nilsson 90] Ulf Nilsson, Jan Matuszynski
Logic, Programming and Prolog
John Wiley & Sons (1990)

[Sperschneider, 91] V. Sperschneider, G. Antoniou
Logic: A foundation for Computer Science
Addison-Wesley Publishing Company (1991)

[Sterling 86] Leon Sterling, Ehud Shapiro
The Art of Prolog
The MIT Press. 2nd. Ed. 94 (1986)

Índice Alfabético

<p>"chequeo de ocurrencias" 33</p> <p>alfabeto 2</p> <p>Algoritmo de resolución general 36</p> <p>Algoritmo de resolución proposicional 24</p> <p>algoritmo de unificación 31</p> <p>Algoritmo de Unificación 32</p> <p>ámbito 4</p> <p>aridad 2</p> <p>átomo 3</p> <p>cabeza de una cláusula 22</p> <p>chequeo de ocurrencias 36</p> <p>cierre existencial 18</p> <p>cláusula cabeza 47</p> <p>cláusula Horn 22</p> <p>cláusula vacía 36</p> <p>cláusula vacía 22</p> <p>cláusula variante 35</p> <p>cláusulas resolubles 23</p> <p>completo 35</p> <p>Completo 39</p> <p>Completud del Algoritmo de Resolución Proposicional 27</p> <p>completud del principio de resolución 42</p>	<p>composición de substituciones 28</p> <p>computación simbólica 49</p> <p>conectivas 3</p> <p>Conjunto de Desacuerdos 31</p> <p>Conjunto de Discrepancias 31</p> <p>Conjunto de letras de función 2</p> <p>Conjunto de letras de Predicado 3</p> <p>Conjunto de símbolos de Constantes 2</p> <p>Conjunto de símbolos de Variables 2</p> <p>consecuencia lógica 14</p> <p>contradicción 12</p> <p>correcto 14</p> <p>Cuantificadores 3</p> <p>cuerpo de la cláusula 22</p> <p>demostración automática 49</p> <p>dominio 7</p> <p>equivalencia lógica 15</p> <p>Estrategias de resolución 44</p> <p>expresión 27</p> <p>expresión simple 27</p> <p>Expresiones similares 31</p> <p>FNC 19</p> <p>FNCP 19</p>
---	---

	Lógica de Predicados E.U.I.T.I.O.		
FND	19	literal negativo	19
FNDP	19	literal positivo	19
FNP	18	literal puro	44
FNS	19	lógica	2
Forma Clausal	21	lógica de enunciados	2
Forma Normal Conjuntiva	19	lógica de predicados	2
Forma Normal Conjuntiva Prenexa	19	lógica de predicados de orden	6
		cero	
Forma Normal de Skolem	19	lógica de predicados de Primer	6
		Orden	
Forma Normal Disyuntiva	19	lógica de predicados de	6
		Segundo Orden	
Forma Normal Disyuntiva Prenexa	19	lógica proposicional	2
Forma Normal Prenexa	18	matriz	18
fórmula atómica	4	modelo	12
fórmula cerrada	5	nodo de inferencia	25
fórmula insatisfacible	12	Objetivo	22
fórmula satisfacible	12	occur check	33
fórmula válida	12	prefijo	18
fórmulas bien formadas	4	programación lógica	1
Gödel	49	razonamiento	14
Hecho	22	recorrido	4
Herbrand	30	regla de resolución	23
Hilbert	49	Resolución proposicional	23
instancia	27	resolvente	23, 35
instancia básica	27	resolvente de entrada	46
Interpretación	7, 8, 14	Robinson	30
lema de elevación	40	Signos de puntuación	3
literal	19		

	Lógica de Predicados E.U.I.T.I.O.	
---	--	---

substitución	27	término	3
substitución básica	27	umg	30
substitución de renombramiento	29, 30, 35	unificación	30
substitución vacía	28	unificador	30
subsunción	45	unificador más general	30
tautología	44	valor de una fórmula	8
Teorema de completud proposicional	39	variable de cuantificación	4
teorema de Herbrand	39	variable libre	4
teorema de unificación	34	variable ligada	4
teoría	49	variante	30
		variantes	28



Información de Contacto

Escuela Universitaria de Ingeniería
Técnica en Informática (E.U.I.T.O.)

www.eutio.uniovi.es

Jose Emilio Labra Gayo

labra@lsi.uniovi.es

Daniel Fernández Lanvin

dflanvin@uniovi.es