Conocimiento y Razonamiento Automatizado
Temas 2 y 3
Grado en Ingeniería Informática

José Enrique Morais San Miguel

18 de marzo de 2021

Universidad de Alcalá

Ejemplo

¿De las fórmulas

$$\forall X \forall Y p(X) \rightarrow (r(X) \wedge s(X, Y)),$$

$$\forall X \exists Y \forall Z \neg s(X, Y) \vee t(Y, Z) \text{ y}$$

$$\forall Y \forall Z \neg t(Y, Z) \vee v(Y),$$

se deduce $\forall X p(X) \rightarrow v(X)$? Usar resolución para responder a la pregunta.

Skolenizando cada una de las fórmulas, se obtienen las cláusulas:

1.- $\neg p(X) \vee r(X)$
2.- $\neg p(X_1) \vee s(X_1, Y_1)$
3.- $\neg s(X_2, f(X_2)) \vee t(f(X_2), Z_2)$
4.- $\neg t(Y_3, Z_3) \vee v(Y_3)$
5.- $p(a)$
6.- $\neg v(a)$

Ejemplo

1.- $\neg p(X) \vee r(X)$
2.- $\neg p(X_1) \vee s(X_1, Y_1)$
3.- $\neg s(X_2, f(X_2)) \vee t(f(X_2), Z_2)$
4.- $\neg t(Y_3, Z_3) \vee v(Y_3)$
5.- $p(a)$
6.- $\neg v(a)$

7.- $r(a)$	$\{X \leftarrow a\}$	$1,5$
8.- $\neg p(X_1) \vee t(f(X_1), Z_2)$	$\{X_2 \leftarrow X_1, Y_1 \leftarrow f(X_1)\}$	$2,3$
9.- $s(a, Y_1)$	$\{X_1 \leftarrow a\}$	$2,5$
10.- $\neg s(X_2, f(X_2)) \vee v(f(X_2))$	$\{Y_3 \leftarrow f(X_2), Z_3 \leftarrow, Z_2\}$	$3,4$
11.- $t(f(a), Z_2)$	$\{X_2 \leftarrow a, Y_1 \leftarrow f(a)\}$	$3,9$
12.- $\neg p(X_1) \vee v(f(X_1))$	$\{X_2 \leftarrow X_1, Y_1 \leftarrow f(X_1)\}$	$2,10$
13.- $\neg t(a, Z_3)$	$\{Y_3 \leftarrow a\}$	$4,6$
~~14.- $v(f(X_1)) \vee \neg p(X_1)$~~	~~$\{Y_3 \leftarrow f(X_1), Z_3 \leftarrow Z_2\}$~~	~~$4,8$~~
14.- $v(f(a)$	$\{Y_3 \leftarrow f(a)\}$	$4,11$
~~15.- $t(f(a), Z_2)$~~	~~$\{X_1 \leftarrow a\}$~~	~~$5,8$~~
~~15.- $v(f(a))$~~	~~$\{X_1 \leftarrow a\}$~~	~~$5,12$~~

Ejemplo. Problema 3. PEI de 31 de marzo de 2016

¿De las fórmulas

$$\forall X \forall Y (p(X,Y) \rightarrow r(X)),$$

$$\forall X \exists Y (\neg p(X,Y) \longrightarrow \neg s(X,Y)) \text{ y}$$

$$\exists X \forall Y s(X,Y),$$

se deduce $\forall X \exists Y p(X,Y)$? Usar resolución para responder a la pregunta.

Skolenizando cada una de las fórmulas, se obtienen las cláusulas:

1.- $\neg p(X,Y) \vee r(X)$
2.- $p(X_1, f(X_1)) \vee \neg s(X_1, f(X_1))$
3.- $s(a, Y_2)$
4.- $\neg p(b, Y_3)$

Problema 3. PEI de 31 de marzo de 2016

1.- $\neg p(X, Y) \vee r(X)$
2.- $p(X_1, f(X_1)) \vee \neg s(X_1, f(X_1))$
3.- $s(a, Y_2)$
4.- $\neg p(b, Y_3)$

5.- $r(X_1) \vee \neg s(X_1, f(X_1))$	$\{X \leftarrow X_1, Y \leftarrow f(X_1)\}$	1,2
6.- $p(a, f(a))$	$\{X_1 \leftarrow a, Y_2 \leftarrow f(a)\}$	2,3
7.- $\neg s(b, f(b))$	$\{X_1 \leftarrow b, Y_3 \leftarrow f(b)\}$	2,4
8.- $r(a)$	$\{X \leftarrow a, Y \leftarrow f(a)\}$	1,6
9.- ~~$r(a)$~~	~~$\{X_1 \leftarrow a, Y_2 \leftarrow f(a)\}$~~	~~3,5~~

Ejemplo. Problema 3. PEI de 21 de marzo de 2019

¿De las fórmulas

$$\forall X \forall Y\, p(X,Y) \longrightarrow q(X,Y),$$

$$\forall X \exists Y\, r(X,Y) \longrightarrow \boxed{s(X,Y)}\ ^{a}\ \text{y}$$

$$\exists X \forall Z\, (\neg r(X,Z) \vee s(X,Z)) \longrightarrow p(X,Z),$$

se deduce $\exists X \exists Y\, p(X,Y) \vee q(X,Y)$? Usar resolución para responder a la pregunta.

---

[a] $s(Y,X)$ en el examen.

1.- $\neg p(X,Y) \vee q(X,Y)$

2.- $\neg r(X_1, f(X_1)) \vee \boxed{s(X_1, f(X_1))}$

3.- $r(a,Z) \vee p(a,Z)$

4.- $\neg s(a,Z_1) \vee p(a,Z_1)$

5.- $\neg p(X_2,Y_2)$

6.- $\neg q(X_3,Y_3)$

Ejemplo Problema 3. PEI de 21 de marzo de 2019

1.- $\neg p(X, Y) \vee q(X, Y)$
2.- $\neg r(X_1, f(X_1)) \vee s(X_1, f(X_1))$
3.- $r(a, Z) \vee p(a, Z)$
4.- $\neg s(a, Z_1) \vee p(a, Z_1)$
5.- $\neg p(X_2, Y_2)$
6.- $\neg q(X_3, Y_3)$

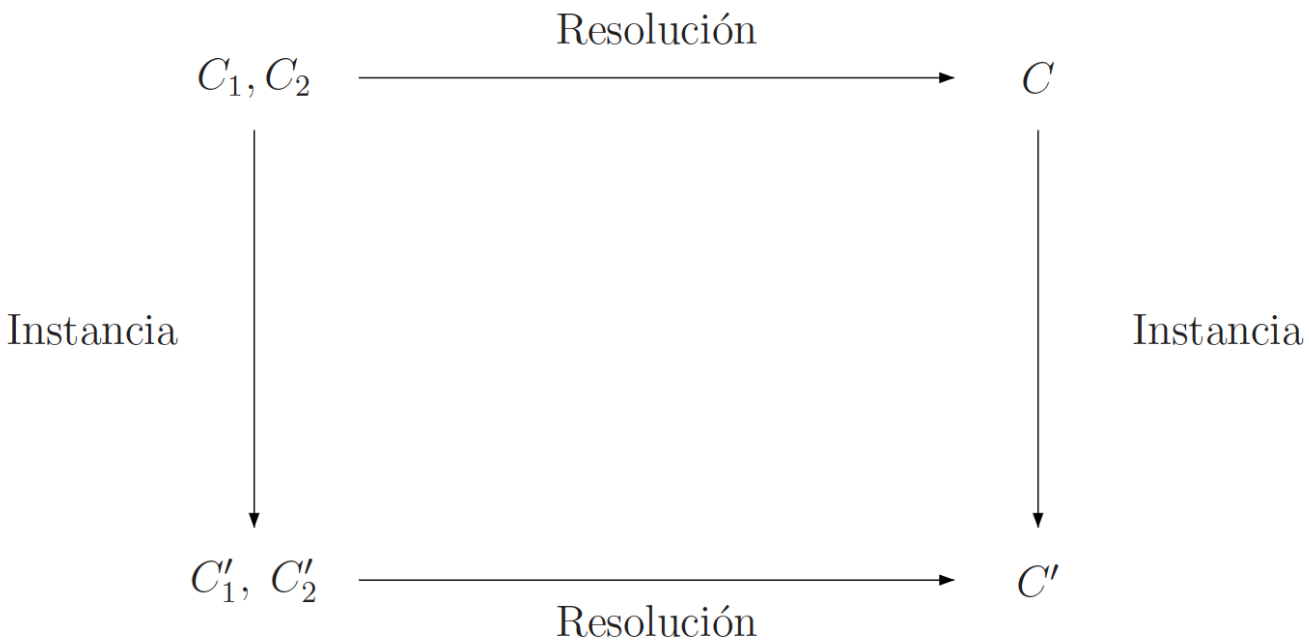
7.- $\neg p(X, Y)$	$\{X_3 \leftarrow X, Y_3 \leftarrow Y\}$	1,6
8.- $r(a, Y)$	$\{X \leftarrow a, Z \leftarrow Y\}$	3,7
9.- $s(a, f(a))$	$\{X_1 \leftarrow a, Y \leftarrow f(a)\}$	2,8
10.- $p(a, f(a))$	$\{Z_1 \leftarrow f(a)\}$	4,9
11.- $\square$	$\{X_2 \leftarrow a, Y_2 \leftarrow f(a)\}$	5,10

Resolución:

Teorema (Lema de levantamiento)

Supongamos que C'_1 y C'_2 son dos instancias básicas de C_1 y C_2 respectivamente y sea C' una resolvente básica de C'_1 y C'_2. Entonces, existe una resolvente C de C_1 y C_2 para la cual C' es instancia básica.

La situación descrita por el Teorema puede visualizarse así:



Resolución

Usando el lema del levantamiento, se puede probar que

Teorema

Sean C_1 y C_2 dos clásulas que permiten resolución. Entonces, C_1 y C_2 son simultáneamente satisfacibles si, y sólo si, $Res(C_1, C_2)$ es satisfacible para cualquier resolución.

Demostración.- Supongamos, en primer lugar, que C_1 y C_2 son simultáneamente satisfacibles, que son contradictorias para una umg σ y que $C = Res(C_1, C_2)$ para la citada unificación. Entonces, existe una interpretación $\mathscr{I}$ tal que $\nu_{\mathscr{I}}(C_i) = T$.En consecuencia, existen instancias base $C_i' = C_i\lambda_i$ tal que $\nu_{\mathscr{I}}(C_i') = T$. Puesto que σ es una umg, hay sustituciones θ_i tales que $\lambda_i = \sigma\theta_i$. Entonces, $C_i' = C_i\lambda = C_i(\sigma\theta_i) = (C_i\sigma)\theta_i$, lo que demuestra que $C_i\sigma$ es satisfacible para la misma interpretación.
Supongamos que las cláusulas de C_1 y C_2 que sirven para resolver se unifican en l y l^c respectivamente. Sólo uno de entre l o l^c puede ser satisfacible en la interpretación $\mathscr{I}$. Supongamos, sin restricción de generalidad, que $\nu_{\mathscr{I}}(l) = T$. Puesto que $C_2\sigma$ es satisfacible, debe existir un literal $l' \in C_2$ distinto de l^c y tal que $\nu_{\mathscr{I}}(l'\sigma) = T$. Por la construcción de la resolvente, $l' \in C$ y, por lo tanto, $\nu_{\mathscr{I}}(C) = T$.
Ahora, supongamos que $Res(C_1, C_2)$ es satisfacible para cualquier resolución. Por reducción al absurdo, asumamos que C_1 y C_2 no son simultáneamente satisfacibles. Entonces, tienen que existir instancias base C_1' y C_2' de C_1 y C_2 mutuamente insatisfacibles y generatrices. La resolvente C' de C_1' y C_2' es insatisfacible. Por el lema del levantamiento, existe una cláusula, C que es una resolvente de C_1 y C_2 tal que C' es una instancia básica suya. En consecuencia, C es insatisfacible contradiciendo la hipótesis de que toda resolvente de C_1 y C_2 es satisfacible.

Robustez y completitud de la resolución

Una vez establecido el resultado anterior y usando los argumentos de su demostración, la prueba de la robustez y completitud de la resolución es la misma que la del caso de la resolución en el Cálculo Proposicional, salvo que cuando se define nodo de fallo en un árbol semántico para el conjunto de clásulas S, se define cambiando falsificando de una cláusula por falsificación de alguna *instancia base* de una cláusula. Del mismo modo, rama cerrada quiere decir rama que falsifica una instancia base de S.

Teorema (Robustez)

Si la resolución deriva la cláusula vacía, el conjunto de cláusulas es insatisfacible

Teorema (Completitud)

Si el conjunto de cláusulas es insatisfacible, el proceso de resolución puede derivar la cláusula vacía $\square$.

Selective Linear Definite clause resolution

Los axiomas en cualquier teoría suelen tener la forma

$$A = \forall X_1 \cdots \forall X_m (B_1 \wedge \ldots \wedge B_m \longrightarrow B),$$

donde los B_i y B son átomos. En forma clausal, el axioma es $A = \neg B_1 \vee \ldots \vee \neg B_n \vee B$. Para probar que una fórmula $G = G_1 \wedge \ldots \wedge G_k$, con los G_i átomos, es consecuencia lógica de un conjunto de axiomas, se añade la negación de G al conjunto de axiomas y se trata de construir una refutación por resolución. Típicamente $\neg G$ se denomina *cláusula objetivo*.
Como presumiblemente los axiomas se han escogido para que sean satisfacibles, carece de sentido realizar resolución entre los axiomas por lo que la resolución se realiza entre la cláusula objetivo con alguno de los axiomas. Como $\neg G$ es una disyunción de literales negativos sólo puede resolverse con el único literal positivo del axioma. Se obtiene, por lo tanto, una cláusula que sólo contiene literales negativos. Si entre los axiomas se encuentran axiomas sin antecedentes, esto es, axiomas cuya forma clausal es un átomo positivo, eventualmente, podríamos producir resolventes que cada vez contendrían menos literales y, finalmente, la cláusula vacía.
Si escribimos los axiomas usando $\leftarrow$, podemos interpretar la fórmula

$$A = \forall X_1 \cdots \forall X_n (B \leftarrow B_1 \wedge \ldots \wedge B_m),$$

como un procedimiento: para computar B, computar $B_1, \ldots, B_n$. Esta es la razón por la que se suelen escribir de esta forma los axiomas en Programación Lógica.

Definiciones básicas

Definición (Cláusulas de Horn)

Se dice **cláusula de Horn** *a toda fórmula del tipo* $A \leftarrow B_1, \ldots, B_n$ *con, a lo más, un literal positivo. Si existe ese literal, A, se le dice* **cabeza** *y a los literales* B_i, $1 \leq i \leq n$, *se les dice* **cuerpo** *de la cláusula. En particular, llamaremos* **hechos** *a las cláusulas del tipo* $A \leftarrow$ *y* **cláusulas objetivo** *a las del tipo* $\leftarrow B_1, \ldots, B_n$.

Definición

Un conjunto de cláusulas cuya cabeza tiene la misma letra de predicado se llamará **procedimiento**. *Un conjunto de procedimientos se dice* **programa lógico**.

Ejemplo

En el siguiente programa lógico las cláusulas C_3 *a* C_{14} *forman una base de datos.*

$$C_1 = p(X,Y) \leftarrow q(X,Z), r(Z,Y)$$
$$C_2 = p(X,Y) \leftarrow s(X,Z), q(Z,W), r(W,Y)$$

$C_3 = s(a,b)$	$C_4 = s(b,a)$	$C_5 = s(c,d)$
$C_6 = s(d,c)$	$C_7 = q(b,c)$	$C_8 = q(c,b)$
$C_9 = r(a,e)$	$C_{10} = r(b,e)$	$C_{11} = r(c,f)$
$C_{12} = r(d,f)$	$C_{13} = r(a,g)$	$C_{14} = r(b,g)$

Definiciones básicas

Definición

Dado un programa lógico P y una cláusula objetivo $\leftarrow G$, llamaremos **respuesta correcta** *a toda sustitución θ tal que $P \vdash \forall(\neg G\theta)$.*

Ejemplo

Sea P el conjunto de axiomas de la aritmética en $\mathbb{N}$ y $G = \neg(X = Y + 13)$. Entonces, $\theta = \{Y \leftarrow X - 13\}$ es una respuesta correcta pues

$$P \models \forall(X = X - 13 + 13)$$

Obsérvese, por lo tanto, que una respuesta correcta no tiene que ser necesariamente una sustitución básica.

Ejemplo

$$C_1 = p(X,Y) \leftarrow q(X,Z), r(Z,Y)$$
$$C_2 = p(X,Y) \leftarrow s(X,Z), q(Z,W), r(W,Y)$$

$C_3 = s(a,b)$	$C_4 = s(b,a)$	$C_5 = s(c,d)$
$C_6 = s(d,c)$	$C_7 = q(b,c)$	$C_8 = q(c,b)$
$C_9 = r(a,e)$	$C_{10} = r(b,e)$	$C_{11} = r(c,f)$
$C_{12} = r(d,f)$	$C_{13} = r(a,g)$	$C_{14} = r(b,g)$

Pongamos que queremos construir una refutación de la cláusula objetivo $\leftarrow p(d,Y)$ con el programa dado en el ejemplo

- P1: $\leftarrow p(d,Y_1)$, C_2, $\{X \leftarrow d, Y \leftarrow Y_1\}$, $\leftarrow s(d,Z), q(Z,W), r(W,Y_1)$.
- P2: $\leftarrow s(d,Z)$, C_6, $\{Z \leftarrow c\}$, $\leftarrow q(c,W), r(W,Y_1)$.
- P3: $\leftarrow q(c,W)$, C_8, $\{W \leftarrow b\}$, $\leftarrow r(b,Y_1)$.
- P4: $\leftarrow r(b,Y_1)$, C_{10}, $\{Y_1 \leftarrow e\}$, $\square$.

La composición de las unificaciones es $\{X \leftarrow d, Z \leftarrow c, W \leftarrow b, Y \leftarrow e\}$, por lo que la sustitución $\{Y \leftarrow e\}$ es una respuesta correcta, es decir, $P \models p(d,e)$. En consecuencia, $P \models \leftarrow p(d,Y) \equiv \neg p(d,Y)$.

Regla de computación

En el ejemplo inmediatamente anterior hemos ido escogiendo literales de las diferentes cláusulas objetivo tomando siempre la situada más a la izquierda. Naturalmente, no es esta la única forma de hacerlo. Si no se fija una regla para escoger estas cláusulas, el algoritmo consecuente es no determinista.

Definición (Regla de computación)

Se dice regla de computación a una función que, cuando se aplica a una cláusula objetivo, escoge uno y sólo uno de los átomos de la misma[a].

[a] Por ejemplo, Prolog elige siempre el átomo que se encuentra más a la izquierda.

Derivación SLD

Una vez introducidas las ideas anteriores, estamos en condiciones de definir lo que se entiende por derivación SLD.

Definición

Sea P un programa, R una regla de computación y G una cláusula objetivo. Una derivación por SLD-resolución es una secuencia de pasos entre cláusulas objetivo y cláusulas del programa. La primera cláusula objetivo es $G_0 = G$ y, supuesta construida G_i, se construye G_{i+1} seleccionando $A_k \in G_i$ (mediante R), escogiendo $C_i \in P$ tal que su cabeza unifica con A_k gracias a una u.m.g. θ_i y resolviendo:

$$G_i = \leftarrow A_1, \ldots, A_{k-1}, A_k, A_{k+1}, \ldots, A_n$$

$$C_i = A \leftarrow B_1, \ldots, B_l$$

$$A_k\theta_i = A\theta_i$$

$$G_{i+1} = \leftarrow (A_1, \ldots, A_{k-1}, B_1, \ldots, B_l, A_{k+1}, \ldots, A_n)\theta_i.$$

Derivación SLD

Se distinguen los siguientes tipos de derivación:

- *Infinita*: Se produce cuando en cualquier objetivo G_i de la secuencia, el átomo A_i escogido por la regla de computación unifica con (una variante) una cláusula del programa P.
- *Finita*: la derivación termina en un número finito de pasos. Atendiendo al último objetivo de la secuencia, G_n, las derivaciones pueden ser:
  - De éxito, si $G_n = \square$, es decir, una derivación de éxito es una refutación.
  - De fallo, en cualquier otro caso.

Robustez y completitud

Teorema (Robustez)

Sea P un programa, R una regla de computación y G una cláusula objetivo. Supongamos que existe una refutación SLD de G. Sea $\theta = \theta_1 \cdots \theta_l$ la sucesión de unificadores y sea σ la restricción de θ a las variables de G. Entonces, σ es una respuesta corrrecta para G.

La refutación SLD no es completa para conjuntos de cláusulas en general. El conjunto de cláusulas

$$\{p \vee q, \neg p \vee q, p \vee \neg q, \neg p \vee \neg q\}$$

es insatisfacible, pero no hay forma de derivar la clásula vacía si se exige que una de las cláusulas con la que resolver sea una de las dadas. Sin embargo, sí es completa para cláusulas de Horn.

Teorema (Completitud)

Sea P un programa, R una regla de computación y G una cláusula objetivo. Sea σ una respuesta correcta. Entonces, existe una refutación SLD de G a partir de P tal que σ es la restricción de la sucesión de unificadores $\theta = \theta_1 \cdots \theta_n$ a las variables de G.

Árboles SLD

En la Definición de derivación SLD, no se especifica en qué orden elegir las cláusulas del programa con la que hacer resolución con el átomo elegido por la regla de computación. Cualquier forma de especificar cómo hacer esta elección se dice *regla de ordenación*.

Definición (Árbol SLD)

Sea un programa P y G una cláusula objetivo. Un árbol SLD, para unas ciertas reglas de computación y de ordenación, es un árbol definido por:

1. *El nodo raíz de es el objetivo inicial.*
2. *Si $G_i \equiv \leftarrow A_1 \wedge \ldots \wedge A_k \wedge \ldots \wedge A_n$ es un nodo y A_k es el átomo seleccionado por la regla de computación, para cada cláusula $C_j \equiv A \leftarrow B_1 \wedge \ldots \wedge B_m$ de P cuya cabeza A unifica con A_k, escogidas en el orden dado por la regla de ordenación, con umg θ, el resolvente*

$$G_{ij} \equiv \leftarrow (A_1 \wedge \ldots A_{k-1} \wedge B_1 \wedge \ldots \wedge B_m \wedge A_{k+1} \wedge \ldots \wedge A_n)\theta$$

es un nodo del árbol hijo de G_i.

Árboles SLD

Nota

Obsérvese que las hojas del árbol se corresponden con cláusulas que no pueden resolverse con ninguna cláusula del programa. Estas hojas o bien son la cláusula vacía o nodos de fallo. Por otro lado, cada rama del árbol es una derivación SLD. Las ramas que se corresponden con derivaciones infinitas se dicen ramas infinitas, las que se corresponden con refutaciones se dicen ramas de éxito y las que se corresponden con derivaciones de fallo, se dicen ramas de fallo. Como veremos a continuación, los árboles dependen de las reglas de computación y también de la regla de ordenación.

Ejemplo

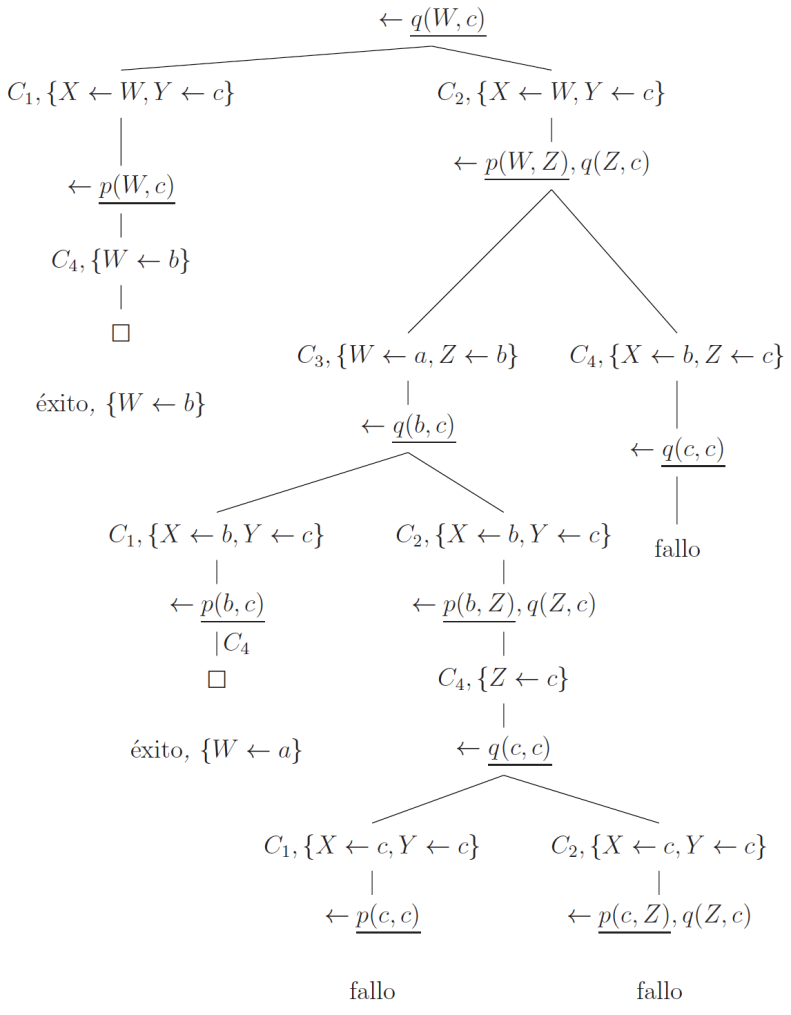
Consideremos el siguiente programa lógico:

- $C_1 = q(X, Y) \leftarrow p(X, Y)$
- $C_2 = q(X, Y) \leftarrow p(X, Z), q(Z, Y)$
- $C_3 = p(a, b)$
- $C_4 = p(b, c)$

y tomemos como objetivo $\leftarrow q(X, c)$.
Primero construimos el árbol SLD que se corresponde con la regla de computación que elige el átomo más a la izquierda y la regla de ordenación que elige la primera cláusula posible, esto es la que está más arriba. Es decir, usamos las reglas de computación y de ordenación de Prolog.

Ejemplo

- $C_1 = q(X, Y) \leftarrow p(X, Y)$
- $C_2 = q(X, Y) \leftarrow p(X, Z), q(Z, Y)$
- $C_3 = p(a, b)$
- $C_4 = p(b, c)$



Ejemplo

Nota

Cada arista del árbol está etiquetada con la cláusula del programa con la que hacemos resolución y la correspondiente umg. Además, subrayamos el átomo de la clásula objetivo escogido. Por otro lado, no completamos el árbol: en la rama más a la derecha se repite parte del árbol ya construido, por lo que la marcamos como rama de fallo.

En la siguiente diapositiva, construimos el árbol cambiando la regla de computación: elegimos el átomo en la cláusula objetivo más a la derecha

Ejemplo

- $C_1 = q(X, Y) \leftarrow p(X, Y)$
- $C_2 = q(X, Y) \leftarrow p(X, Z), q(Z, Y)$
- $C_3 = p(a, b)$
- $C_4 = p(b, c)$