

Tema 7

Programación Lógica Inductiva

Gonzalo A. Aranda-Corral

Ciencias de la Computación e Inteligencia Artificial
Universidad de Huelva

noviembre 2022

1 Lógica

- Proposicional
- Lógica de Primer Orden
- Prover9

2 Programación Lógica Inductiva

- Introducción
- Generación de reglas por especificación
- Generación de reglas por resolución inversa

Proposición: afirmación simple, que puede tomar los valores cierto o falso

Ejemplo:

```
1 Temperatura es alta
2 Cuello_es_recto
3 Clase_es_A
4 Celda_1_3_es_4
```

- Las proposiciones expresan un conocimiento concreto.
- No permiten expresar conceptos genéricos como “hay un 4 en alguna celda”.
- Para este tipo de enunciados es necesaria la lógica de primer orden.

- La lógica proposicional trabaja con fórmulas bien formadas. (FBF)
- Las FBF son expresiones lógicas que se construyen combinando proposiciones simples y conectivas lógicas.
- Una FBF es:
 - Una proposición (p)
 - La negación de una FBF ($\neg p$)
 - La conjunción de dos FBF ($p \wedge q$)
 - La disyunción de dos FBF ($p \vee q$)
 - La implicación entre dos FBF ($p \implies q$)
 - La doble implicación entre dos FBF ($p \iff q$)

Definiciones

- Una FBF puede tomar los valores cierto o falso, en función de los valores de las proposiciones simples que la formen.
- Para representar el significado de una fórmula se utilizan tablas de verdad, en la que se enumeran todas las combinaciones posibles de las proposiciones simples y el valor de la fórmula para cada combinación.
- Para una fórmula formada por N proposiciones, el tamaño de la tabla de verdad es de 2^N filas.
- Las tablas de verdad de los conectivos lógicos son las siguientes:

P	Q	$P \vee Q$	$P \wedge Q$	$\neg P$	$P \rightarrow Q$	$P \leftrightarrow Q$
V	V	V	V	F	V	V
V	F	V	F	F	F	F
F	V	V	F	V	V	F
F	F	F	F	V	V	V

- Una fórmula es decidable si se puede demostrar si es válida o no en un número finito de pasos.
- Una fórmula es satisfactible si su valor es cierto para alguna combinación de valores. Si una fórmula es siempre falsa se dice que es insatisfactible.
- Una fórmula es **válida** si su valor es cierto para todas las combinaciones de valores de las proposiciones que la forman. Una fórmula válida se conoce también como tautología.
- Se dice que un sistema lógico es consistente si no permite demostrar simultáneamente una fórmula y su contraria.
- La lógica proposicional es decidable, porque existe un algoritmo (la tabla de verdad) que permite demostrar la validez de cualquier fórmula en un número finito de pasos

- Dos expresiones lógicas son equivalentes si tienen la misma tabla de verdad.
- Algunas expresiones equivalentes:
 - $\neg(a \vee b) = \neg a \wedge \neg b$ (Ley de Morgan)
 - $\neg(a \wedge b) = \neg a \vee \neg b$ (Ley de Morgan)
 - $a \rightarrow b = \neg a \vee b$ (Implicación)
 - $a \leftrightarrow b = (a \wedge b) \vee (\neg a \wedge \neg b)$ (Doble implicación)
 - $\neg(\neg a) = a$ (Doble negación)
 - $(a \vee b) = (b \vee a)$ (Propiedad conmutativa de la disyunción)
 - $(a \wedge b) = (b \wedge a)$ (Propiedad conmutativa de la conjunción)

Algunas expresiones equivalentes

- $a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$ (Propiedad distributiva)
- $a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$ (Propiedad distributiva)
- $a \vee F = a$ (Elemento neutro)
- $a \vee T = T$ (Elemento absorbente)
- $a \wedge T = a$ (Elemento neutro)
- $a \wedge F = F$ (Elemento absorbente)
- $a \vee \neg a = T$ (Tercio excluso)
- $a \wedge \neg a = F$ (No contradicción)
- $a \wedge a = a; a \vee a = a; a \vee (a \wedge b) = a; a \wedge (a \vee b) = a;$

- Aplicando las transformaciones anteriores es posible convertir cualquier fórmula lógica en una forma normal conjuntiva. (FNC)
- Una FNC tiene las siguientes características:
 - No contiene implicaciones.
 - No contiene dobles implicaciones.
 - Las negaciones aparecen aplicadas sólo a proposiciones simples.
 - Las expresiones están formadas por **conjunción de disyunciones**.

Algoritmo para transformar una fórmula lógica en forma normal conjuntiva.

- I. Sustituir toda implicación ($a \implies b$) por $(\neg a \vee b)$.
- II. Sustituir toda doble implicación ($a \leftrightarrow b$) por $(a \wedge b) \vee (\neg a \wedge \neg b)$.
- III. Sustituir las expresiones de tipo $\neg(a \vee b)$ por $(\neg a \wedge \neg b)$
- IV. Sustituir las expresiones de tipo $\neg(a \wedge b)$ por $(\neg a \vee \neg b)$
- V. Aplicar la propiedad distributiva para expresar la fórmula como conjunción de disyunciones.

Inferencia

El razonamiento en lógica consiste en la obtención de nuevo conocimiento a partir del que está representado en nuestra base de conocimiento.

Existen varias estrategias para la obtención de nuevo conocimiento. Básicamente podemos destacar:

- **Deducción** o razonamiento hacia delante: consiste en obtener nuevas afirmaciones a partir de los que ya tenemos, utilizando los mecanismos de inferencia, añadirlas a la base de conocimiento y continuar con la obtención de nuevas afirmaciones. (Búsqueda a ciegas).
- **Demostración** o razonamiento hacia atrás: consiste en explicitar el conocimiento que deseamos obtener y comprobar si es consecuencia lógica de (está justificado por) nuestra base de conocimiento. (Búsqueda guiada).

Deducción.

- **Modus Ponens:** si existe una implicación y el antecedente es verdadero, entonces, el consecuente es verdadero.
Intuitivamente: si se da la causa, se da el efecto.

$$BC = \{A \implies B, A\} \quad \frac{(A \implies B) \quad A}{B}$$

pasa a $BC = \{A \implies B, A, B\}$

Deducción.

- **Modus Tollens**: si existe una implicación y el consecuente es falso, entonces el antecedente es falso. Intuitivamente: si no se da el efecto, es imposible que se de la causa.

$$\text{BC} = \{A \implies B, -B\} \quad \frac{A \implies B \quad -B}{-A}$$

pasa a $\text{BC} = \{A \implies B, -B, -A\}$

Dedución.

- **Resolución:**

$$\frac{\neg p \vee q \vee r... \quad p \vee s \vee t...}{q \vee r \vee ... \vee s \vee t \vee ...}$$

A partir de la regla de resolución se puede deducir la regla Modus Ponens y la regla Modus Tollens.

Demostración

Dada una cierta base de conocimientos BC , que se asume como cierta,

¿se puede **demostrar** que una cierta expresión Q es cierta ($BC \vdash Q$)?

- Considerando que BC se puede expresar en forma normal conjuntiva, podemos considerar la base de conocimientos completa como una única expresión formada por la conjunción de todas sus cláusulas.
- En tal caso, lo que nos preguntamos es si $BC \implies Q$ es cierto.
- O, teniendo en cuenta que $a \implies b \equiv \neg(a \wedge \neg b)$, si $BC \wedge \neg Q$ es falso.
- Esto se conoce como demostración por **refutación** y consiste en añadir la cláusula $\neg Q$ a la base de conocimientos y, utilizando la resolución como mecanismo de deducción, llegar a una contradicción.

Demostración por resolución:

```
1 function Resolucion(BC, objetivo) returns boolean
2 input:
3     BC // Base de conocimientos
4
5 objetivo // expresion que se pretende demostrar
6     BC ← BC /\ -objetivo
7
8 repeat
9     c1, c2 ← EscogerClausulas(BC)
10    resolvente ← ReglaDeResolucion(c1,c2)
11    if ClausulaVacía pertenece a Resolvente:
12        return False // Inconsistente
13    BC ← BC /\ resolvente
14 until resolvente = vacía
15
16 return True // Consistente
```

Demostración por resolución:

- para evitar bucles infinitos habría que controlar que no se repitieran c_1 y c_2 y que no se insertan cláusulas duplicadas.
- para que el algoritmo sea más rápido se utiliza la heurística “preferencia por la unidad” porque genera cláusulas cada vez más cortas.

Demostración

- La lógica proposicional es completa, en el sentido de que si una expresión es válida entonces existe algún mecanismo que demuestra que lo es.
- Por ejemplo, el estudio de la tabla de verdad es un mecanismo completo, ya que permite demostrar la validez de cualquier expresión.
- El problema es que su complejidad es de orden exponencial (2^N , siendo N el número de proposiciones diferentes).
- La verificación en lógica proposicional es un problema NP-Completo.

Demostración

- La demostración por refutación es un método **completo**, ya que si lo que se trata de demostrar es cierto, entonces el mecanismo de resolución encontrará la contradicción.
- Por el contrario, si lo que se trata de demostrar es falso, el mecanismo no encuentra la contradicción y no para.
- Es decir, nunca podremos saber si el algoritmo no para porque aún no ha encontrado la contradicción o porque ésta no existe. Se dice, por tanto, que el método de resolución es semidecidible.

Base de conocimientos:

```
1 p
2 q /\ r
3 p /\ q -> s
4 r /\ s -> t
```

Objetivo: demostrar que t es cierto

1er paso: Transformar la base de conocimientos a forma normal conjuntiva

```
1 p
2 q
3 r
4 -p \ / -q \ / s
5 -r \ / -s \ / t
```

2º paso: añadir la negación del objetivo

```
1 (6) -t
```

3er paso: aplicar la regla de resolución $(-r \ -s \ t), (-t)$

```
1 (7) -r \ / -s
```

4º paso: aplicar la regla de resolución $(\neg r \vee \neg s), (r)$

1 (8) $\neg s$

5º paso: aplicar la regla de resolución $(\neg p \vee \neg q \vee s), (\neg s)$

1 (9) $\neg p \vee \neg q$

6º paso: aplicar la regla de resolución $(\neg p \vee \neg q), (q)$

1 (10) $\neg p$

7º paso: aplicar la regla de resolución

1 $(\neg p), (p)$

¡CONTRADICCIÓN! Lo que demuestra t por refutación


- La lógica de primer orden considera objetos y sus propiedades y relaciones.
- Por ejemplo, para el Sudoku los objetos podrían ser las celdas (o mejor las filas y columnas) y los posibles valores:
 - F0, F1, F2, F3, F4, F5, F6, F7, F8.
 - C0, C1, C2, C3, C4, C5, C6, C7, C8.
 - No1, No2, No3, No4, No5, No6, No7, No8, No9
- Las propiedades podrían ser:
 - Fila/1, para indicar que un objeto es una fila (Fila(F0)).
 - Columna/1, para indicar que un objeto es una columna

Término (forma de expresar objetos)

■ Constantes

- Permiten identificar los objetos del dominio
- Se denotan mediante cadenas (comienzan en mayúscula) o números
- Ejemplo: Juan, María, F1, C8, 27, 524.8

■ Variables

- Permite referenciar cualquier objeto del dominio
- Se denotan mediante cadenas que comienzan en minúscula 

Funciones

- Permiten obtener una referencia a un objeto en función de los valores de otros.
- Se denotan mediante un nombre y un conjunto de argumentos, que tienen que ser términos
- Ejemplo: Padre(Juan), Suma(5, 8)
- Las constantes se pueden considerar funciones de 0 argumentos

Átomo (equivale a las proposiciones)

■ Predicado:

- Permite describir una relación entre objetos o una propiedad del objeto
- Se caracterizan por su nombre y su aridad (n° de argumentos)
- El nombre es una cadena que comienza en mayúscula
- Los argumentos deben ser términos
- Un predicado puede ser cierto o falso
- Ejemplo: Fila(F1), Celda(F3,C4), Contiene(F6,C8,No3)

Átomo (equivale a las proposiciones)

■ Igualdad o identidad:

- Permite incorporar la condición de que un término sea igual a otro
- Ejemplo: $x = y$, $\text{Padre}(\text{Juan}) = \text{Pedro}$, $\text{Suma}(x, 4) = y$
- Se puede considerar como un predicado binario
- Requiere un tratamiento especial para la inferencia
- Su utilización supone una extensión de la lógica de primer orden

Fórmulas bien formadas

- Átomo
- La negación de una fórmula bien formada ($\neg p$)
- La conjunción de dos fórmulas bien formadas ($p \wedge q$)
- La disyunción de dos fórmulas bien formadas ($p \vee q$)
- La implicación entre dos fórmulas bien formadas ($p \implies q$)
- La doble implicación entre dos fórmulas bien formadas ($p \equiv q$)
- El **cuantificador universal** aplicado a una fórmula bien formada ($\forall x : P$)
- El **cuantificador existencial** aplicado a una fórmula bien formada ($\exists x : P$)

Equivalencias asociadas con los cuantificadores

- $\forall x : \neg P \equiv \neg(\exists x : P)$
- $\neg(\forall x : P) \equiv \exists x : \neg P$
- $\forall x : P \equiv \neg(\exists x : \neg P)$
- $\neg(\exists x : \neg P) \equiv \forall x : P$
- $\forall x : P \wedge Q \equiv (\forall x : P) \wedge (\forall x : Q)$
- $\exists x : P \vee Q \equiv (\exists x : P) \vee (\exists x : Q)$

Equivalencias asociadas con los cuantificadores

- $\forall x : \neg P \equiv \neg(\exists x : P)$
- $\neg(\forall x : P) \equiv \exists x : \neg P$
- $\forall x : P \equiv \neg(\exists x : \neg P)$
- $\neg(\forall x : \neg P) \equiv \exists x : P$
- $\forall x : P \wedge Q \equiv (\forall x : P) \wedge (\forall x : Q)$
- $\exists x : P \vee Q \equiv (\exists x : P) \vee (\exists x : Q)$

Equivalencias asociadas con los cuantificadores

- $\forall x : \neg P \equiv \neg(\exists x : P)$
- $\neg(\forall x : P) \equiv \exists x : \neg P$
- $\forall x : P \equiv \neg(\exists x : \neg P)$
- $\neg(\forall x : \neg P) \equiv \exists x : P$
- $\forall x : P \wedge Q \equiv (\forall x : P) \wedge (\forall x : Q)$
- $\exists x : P \vee Q \equiv (\exists x : P) \vee (\exists x : Q)$

Equivalencias asociadas con los cuantificadores

- $\forall x : \neg P \equiv \neg(\exists x : P)$
- $\neg(\forall x : P) \equiv \exists x : \neg P$
- $\forall x : P \equiv \neg(\exists x : \neg P)$
- $\neg(\forall x : \neg P) \equiv \exists x : P$
- $\forall x : P \wedge Q \equiv (\forall x : P) \wedge (\forall x : Q)$
- $\exists x : P \vee Q \equiv (\exists x : P) \vee (\exists x : Q)$

Equivalencias asociadas con los cuantificadores

- $\forall x : \neg P \equiv \neg(\exists x : P)$
- $\neg(\forall x : P) \equiv \exists x : \neg P$
- $\forall x : P \equiv \neg(\exists x : \neg P)$
- $\neg(\forall x : \neg P) \equiv \exists x : P$
- $\forall x : P \wedge Q \equiv (\forall x : P) \wedge (\forall x : Q)$
- $\exists x : P \vee Q \equiv (\exists x : P) \vee (\exists x : Q)$

Equivalencias asociadas con los cuantificadores

- $\forall x : \neg P \equiv \neg(\exists x : P)$
- $\neg(\forall x : P) \equiv \exists x : \neg P$
- $\forall x : P \equiv \neg(\exists x : \neg P)$
- $\neg(\forall x : \neg P) \equiv \exists x : P$
- $\forall x : P \wedge Q \equiv (\forall x : P) \wedge (\forall x : Q)$
- $\exists x : P \vee Q \equiv (\exists x : P) \vee (\exists x : Q)$

Forma Normal Conjuntiva

Transformación a forma normal conjuntiva

- I. Eliminar las implicaciones
- II. Reducir el ámbito de las negaciones para que sólo afecten a átomos
- III. Asociar variables distintas a cada cuantificador
- IV. Mover los cuantificadores al comienzo, preservando el orden
- V. Eliminar los cuantificadores existenciales (Skolemización)
- VI. Convertir la fórmula en conjunción de disyunciones (propiedad distributiva de \vee y \wedge)
- VII. Generar una cláusula a partir de cada disyunción

Transformación a forma normal conjuntiva

Ejemplo: $\exists x : (\forall y : (p(x, y) \vee q(x, y)) \implies r(y))$

I. $\exists x : (\forall y : \neg(p(x, y) \vee q(x, y)) \vee r(y))$

II. $\exists x : (\forall y : (\neg p(x, y) \wedge \neg q(x, y)) \vee r(y))$

III. $\exists x : \forall y : (\neg p(x, y) \wedge \neg q(x, y)) \vee r(y)$

IV. $\forall y : (\neg p(SK, y) \wedge \neg q(SK, y)) \vee r(y)$

V. $\forall y : (\neg p(SK, y) \vee r(y)) \wedge (\neg q(SK, y) \vee r(y))$

VI. $[\neg p(SK, y), r(y)], [\neg q(SK, y), r(y)]$

- Sustitución: es una secuencia finita de asociaciones entre variables y términos. $\theta = V_1/t_1, V_2/t_2, \dots, V_n/t_n$
- Aplicación de una sustitución θ a una clausula C : consiste en sustituir cada instancia de la variable V_i en la clausula C por el término t_i asociado en la sustitución. Se denota $C\theta$.
- Unificación: se dice que dos átomos P_1 y P_2 unifican si existe una sustitución θ tal que $P_1\theta \equiv P_2\theta$.
- Unificador más general: es la sustitución θ que permite unificar los átomos y cuyos términos son lo más generales posibles (una variable es más general que una constante, ya que permite referenciar a cualquier objeto).

■ Regla de resolución:

- $\neg p(x) \vee \alpha(x)$
- $p(y) \vee \beta(y) \theta = \text{UnificadorMasGeneral}(p(x)/p(y))$
- $(\alpha(x) \vee \beta(y))\theta$

■ Ejemplo:

- $\neg \text{Hombre}(x) \vee \text{Mortal}(x)$
- $\text{Hombre}(\text{Socrates})$
- $\theta = x/\text{Socrates}$
- $\text{Mortal}(\text{Socrates})$

■ Eliminación universal:

- $\forall x : a(x)$
- $\theta(x / \textit{Constante}, a(x))$
- Ejemplo $\forall x : \textit{Gusta}(x, \textit{Helado})$
- $\textit{Gusta}(\textit{Juan}, \textit{Helado})$

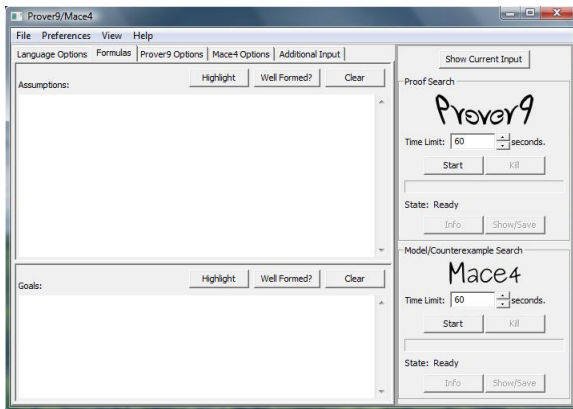
■ Introducción existencial:

- $a(\textit{Constante})$
- $\exists x : a(x)$

Prover9

- Prover9 es una herramienta desarrollada por William McCune para automatizar la demostración de teoremas en lógica de primer orden con igualdad.
- Se distribuye junto a la herramienta Mace4, que está dedicada a encontrar contraejemplos para los teoremas que no se puedan demostrar.
- Prover9 es el sucesor de Otter 33, otro demostrador de teoremas desarrollado por el mismo autor.
- Se distribuye de manera gratuita en:

Prover9



La sintaxis de Prover9 es la siguiente:

- Variables: identificadores que comiencen en minúscula
- Constantes: identificadores que comiencen en mayúscula
- Conjunción: $\&$
- Disyunción: $|$
- Negación: $-$
- Implicación: \rightarrow
- Doble implicación: \leftrightarrow
- Cuantificador universal: `all x`
- Cuantificador existencial: `exists x`

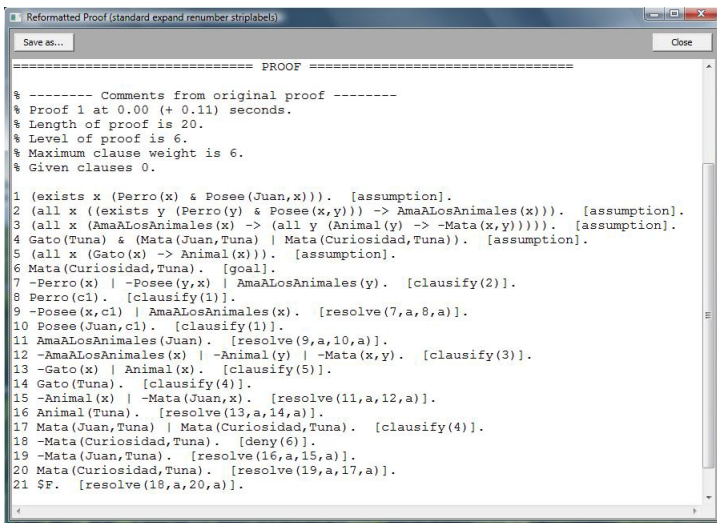
Ejemplo de entrada de Prover9:

Hipotesis

```
1 exists x ( Perro(x) & Posee(Juan,x) ).
2 all x ( ( exists y ( Perro(y) & Posee(x,y)) ) ->
3           AmaALosAnimales(x) ).
4 all x ( AmaALosAnimales(x) -> ( all y ( Animal(y)
5           -> -Mata(x,y) ) ) ).
6 Gato(Tuna) & ( Mata(Juan,Tuna) | Mata(Curiosidad,Tuna) ).
7 all x ( Gato(x) -> Animal(x) ).
8 all x ( Perro(x) -> Animal(x) ).
```

Objetivos

```
1 Mata(Curiosidad,Tuna).
```



```
Reformatted Proof (standard expand renumber striplabels)

Save as... Close

===== PROOF =====

% ----- Comments from original proof -----
% Proof 1 at 0.00 (+ 0.11) seconds.
% Length of proof is 20.
% Level of proof is 6.
% Maximum clause weight is 6.
% Given clauses 0.

1 (exists x (Perro(x) & Posee(Juan,x))). [assumption].
2 (all x ((exists y (Perro(y) & Posee(x,y))) -> AmaALosAnimales(x))). [assumption].
3 (all x (AmaALosAnimales(x) -> (all y (Animal(y) -> -Mata(x,y)))). [assumption].
4 Gato(Tuna) & (Mata(Juan,Tuna) | Mata(Curiosidad,Tuna)). [assumption].
5 (all x (Gato(x) -> Animal(x))). [assumption].
6 Mata(Curiosidad,Tuna). [goal].
7 -Perro(x) | -Posee(y,x) | AmaALosAnimales(y). [clausify(2)].
8 Perro(c1). [clausify(1)].
9 -Posee(x,c1) | AmaALosAnimales(x). [resolve(7,a,8,a)].
10 Posee(Juan,c1). [clausify(1)].
11 AmaALosAnimales(Juan). [resolve(9,a,10,a)].
12 -AmaALosAnimales(x) | -Animal(y) | -Mata(x,y). [clausify(3)].
13 -Gato(x) | Animal(x). [clausify(5)].
14 Gato(Tuna). [clausify(4)].
15 -Animal(x) | -Mata(Juan,x). [resolve(11,a,12,a)].
16 Animal(Tuna). [resolve(13,a,14,a)].
17 Mata(Juan,Tuna) | Mata(Curiosidad,Tuna). [clausify(4)].
18 -Mata(Curiosidad,Tuna). [deny(6)].
19 -Mata(Juan,Tuna). [resolve(16,a,15,a)].
20 Mata(Curiosidad,Tuna). [resolve(19,a,17,a)].
21 $F. [resolve(18,a,20,a)].
```

- La lógica de primer orden es completa¹.
Dado un conjunto de fórmulas bien formadas (BC) y una fórmula bien formada (Q) que sea consecuencia lógica de BC ($BC \models Q$), entonces se puede demostrar Q a partir de BC ($BC \vdash Q$) por medio de la demostración por refutación.
- Sin embargo, si Q no es consecuencia lógica de BC ($BC \not\models Q$), entonces la demostración podría no terminar nunca. Por tanto, la lógica de primer orden es **semidecidible**.
- Para conseguir un procedimiento de inferencia que demuestre cualquier fórmula en **tiempo lineal** es necesario introducir **restricciones**: Cláusulas de Horn.

¹ Siempre que sólo sea LPO

- También existen las lógicas de orden superior: Lógicas cuyos predicados admiten como argumentos otros predicados o funciones, o incluso cuantificadores.
- Hay muchos sistemas deductivos (de LPO) que son “adecuados” (Todo lo que se puede demostrar es cierto en todos los modelos) y “completos” (Todo lo que es verdad en todos los modelos se puede probar)

- Una cláusula de Horn es una disyunción de literales en la que hay a lo sumo un único literal positivo.

$$\neg p_1 \vee \neg p_2 \vee \neg p_3 \vee \dots \vee \neg p_n \vee q$$

- Aplicando leyes de DeMorgan, esto equivale a

$$\neg(p_1 \wedge p_2 \wedge p_3 \wedge \dots \wedge p_n) \vee q$$

- Por la definición de implicación, esto equivale a

$$(p_1 \wedge p_2 \wedge p_3 \wedge \dots \wedge p_n) \implies q$$

- Se denomina cabeza al literal positivo.
- Se denomina cuerpo al conjunto de literales negativos.
- La lógica basada en cláusulas de Horn permite definir un algoritmo de inferencia que es **completo, decidible y orden lineal** (System Linear Resolution for Definite Clauses).

- (SWI-)Prolog es un lenguaje de programación lógica basado en el uso de cláusulas de Horn.
- La sintaxis de Prolog es diferente a la aceptada comúnmente para la lógica de primer orden.
- Las cláusulas se escriben “al revés”: en primer lugar la cabeza y a continuación el cuerpo.

$$q \leftarrow (p1 \wedge p2 \wedge p3 \wedge \dots \wedge pn)$$

- La implicación se denota por “:-”, la conjunción por una coma y las cláusulas deben terminar en punto.

$$q : - p1, p2, p3, \dots, pn.$$

- Las cláusulas que no tienen cuerpo se denominan **hechos**. q .
- Las cláusulas que sí tienen cuerpo se denominan **reglas**.

- Variables: comienzan con una letra mayúscula o un subrayado. (por ejemplo: X, Fila, ...)
- Constantes: comienzan con una letra minúscula o entre comillas simples (por ejemplo: juan, 'Juan')
- Predicados: comienzan con letra minúscula (por ejemplo: padre(juan, pedro))
- Los términos pueden ser variables, constantes, funciones y predicados. (por ejemplo: nodo(nodo(1,2) , nodo(3,4)))
- Los términos también pueden ser listas: (por ejemplo: [0], [0, 1], [E|L])

- El lenguaje contiene las funciones y constantes matemáticas más comunes. (por ejemplo: `sin()`, `cos()`, `tan()`, `asin()`, `acos()`, `atan()`, `sqrt()`, `exp()`, `log()`, `log10()`, `random()`, `pi`, `e`, ...)
- El lenguaje contiene un gran número de predicados predefinidos. (por ejemplo: `member/2`, `append/2`, `last/2`, `reverse/2`)
- Los átomos pueden ser predicados y operadores booleanos (comparaciones entre términos). (por ejemplo $X > 3$, $Y \text{ is } 3 + 8$, $J ::= K$, ...)

1 Lógica

- Proposicional
- Lógica de Primer Orden
- Prover9

2 Programación Lógica Inductiva

- Introducción
- Generación de reglas por especificación
- Generación de reglas por resolución inversa

- La Programación Lógica Inductiva es la aplicación de técnicas de aprendizaje inductivo sobre la lógica de primer orden.
- El término en inglés es Inductive Logic Programming (ILP).
- *La Programación Lógica Inductiva consiste en construir de forma automática las cláusulas lógicas que describan un cierto predicado, en base a un conjunto de ejemplos positivos y negativos de dicho predicado y a un conjunto de cláusulas auxiliares que describan el conocimiento del dominio (es, decir, los predicados que pueden ser utilizados en las cláusulas a construir).*

- La Programación Lógica Inductiva es la aplicación de técnicas de aprendizaje inductivo sobre la lógica de primer orden.
- El término en inglés es Inductive Logic Programming (ILP).
- *La Programación Lógica Inductiva consiste en construir de forma automática las cláusulas lógicas que describan un cierto predicado, en base a un conjunto de ejemplos positivos y negativos de dicho predicado y a un conjunto de cláusulas auxiliares que describan el conocimiento del dominio (es, decir, los predicados que pueden ser utilizados en las cláusulas a construir).*

- La Programación Lógica Inductiva es la aplicación de técnicas de aprendizaje inductivo sobre la lógica de primer orden.
- El término en inglés es Inductive Logic Programming (ILP).
- *La Programación Lógica Inductiva consiste en construir de forma automática las cláusulas lógicas que describan un cierto predicado, en base a un conjunto de ejemplos positivos y negativos de dicho predicado y a un conjunto de cláusulas auxiliares que describan el conocimiento del dominio (es, decir, los predicados que pueden ser utilizados en las cláusulas a construir).*

- Programa lógico: es un conjunto de cláusulas de Horn.
- Un programa lógico, P , se dice consistente respecto a un conjunto de ejemplos negativos, ϵ^- , si no existe ningún ejemplo $e (\in \epsilon^-)$ que pueda ser deducido a partir del programa ($P \not\models e$).
- Un programa lógico, P , se dice completo respecto a un conjunto de ejemplos positivos, ϵ^+ , si todos los ejemplos $e (\in \epsilon^+)$ pueden ser deducidos a partir del programa ($P \models e$).

- Programa lógico: es un conjunto de cláusulas de Horn.
- Un programa lógico, \mathbf{P} , se dice consistente respecto a un conjunto de ejemplos negativos, ϵ^- , si no existe ningún ejemplo $\mathbf{e} (\in \epsilon^-)$ que pueda ser deducido a partir del programa ($\mathbf{P} \not\models \mathbf{e}$).
- Un programa lógico, \mathbf{P} , se dice completo respecto a un conjunto de ejemplos positivos, ϵ^+ , si todos los ejemplos $\mathbf{e} (\in \epsilon^+)$ pueden ser deducidos a partir del programa ($\mathbf{P} \models \mathbf{e}$).

- Programa lógico: es un conjunto de cláusulas de Horn.
- Un programa lógico, \mathbf{P} , se dice consistente respecto a un conjunto de ejemplos negativos, ϵ^- , si no existe ningún ejemplo $\mathbf{e} (\in \epsilon^-)$ que pueda ser deducido a partir del programa ($\mathbf{P} \not\models \mathbf{e}$).
- Un programa lógico, \mathbf{P} , se dice completo respecto a un conjunto de ejemplos positivos, ϵ^+ , si todos los ejemplos $\mathbf{e} (\in \epsilon^+)$ pueden ser deducidos a partir del programa ($\mathbf{P} \models \mathbf{e}$).

Componentes de un problema de Programación Lógica Inductiva:

- Un conjunto de ejemplos positivos, ϵ^+ .
- Un conjunto de ejemplos negativos, ϵ^- .
- Un programa lógico consistente, T , a partir del cual no se pueda deducir al menos uno de los ejemplos positivos. T representa el conocimiento de dominio.

Componentes de un problema de Programación Lógica Inductiva:

- Un conjunto de ejemplos positivos, ϵ^+ .
- Un conjunto de ejemplos negativos, ϵ^- .
- Un programa lógico consistente, T , a partir del cual no se pueda deducir al menos uno de los ejemplos positivos. T representa el conocimiento de dominio.

Componentes de un problema de Programación Lógica Inductiva:

- Un conjunto de ejemplos positivos, ϵ^+ .
- Un conjunto de ejemplos negativos, ϵ^- .
- Un programa lógico consistente, T , a partir del cual no se pueda deducir al menos uno de los ejemplos positivos. T representa el conocimiento de dominio.

Objetivo de la Programación Lógica Inductiva:

- Encontrar un programa lógico H tal que $T \cup H$ sea completo (respecto de ϵ^+) y consistente (respecto de ϵ^-).

Objetivo de la Programación Lógica Inductiva:

- Encontrar un programa lógico H tal que $T \cup H$ sea completo (respecto de ϵ^+) y consistente (respecto de ϵ^-).

- La solución de un problema de Programación Lógica Inductiva puede ser un programa lógico H formado por cláusulas de Horn que describen el predicado al que se refieren los ejemplos (positivos y negativos) del problema.
- Una de las estrategias para obtener estas cláusulas consiste en proponer un cláusula lo más general posible (que, por tanto, no será consistente con el conjunto de ejemplos negativos) e ir especificándola (añadiendo términos al cuerpo) hasta hallar la consistencia.
- **Esta es la estrategia seguida en el algoritmo FOIL.**
- FOIL fue desarrollado por J.R. Quinlan y mejorado en colaboración con R.M. Cameron-Jones.
- Sobre la base de FOIL, se han propuesto numerosos algoritmos.

- La solución de un problema de Programación Lógica Inductiva puede ser un programa lógico H formado por cláusulas de Horn que describen el predicado al que se refieren los ejemplos (positivos y negativos) del problema.
- Una de las estrategias para obtener estas cláusulas consiste en proponer un cláusula lo más general posible (que, por tanto, no será consistente con el conjunto de ejemplos negativos) e ir especificándola (añadiendo términos al cuerpo) hasta hallar la consistencia.
- Esta es la estrategia seguida en el algoritmo FOIL.
- FOIL fue desarrollado por J.R. Quinlan y mejorado en colaboración con R.M. Cameron-Jones.
- Sobre la base de FOIL se han propuesto numerosos algoritmos.

- La solución de un problema de Programación Lógica Inductiva puede ser un programa lógico H formado por cláusulas de Horn que describen el predicado al que se refieren los ejemplos (positivos y negativos) del problema.
- Una de las estrategias para obtener estas cláusulas consiste en proponer un cláusula lo más general posible (que, por tanto, no será consistente con el conjunto de ejemplos negativos) e ir especificándola (añadiendo términos al cuerpo) hasta hallar la consistencia.
- **Esta es la estrategia seguida en el algoritmo FOIL.**

- FOIL fue desarrollado por J.R. Quinlan y mejorado en colaboración con R.M. Cameron-Jones.

- Sobre la base de FOIL, se han propuesto numerosos algoritmos.

- La solución de un problema de Programación Lógica Inductiva puede ser un programa lógico H formado por cláusulas de Horn que describen el predicado al que se refieren los ejemplos (positivos y negativos) del problema.
- Una de las estrategias para obtener estas cláusulas consiste en proponer un cláusula lo más general posible (que, por tanto, no será consistente con el conjunto de ejemplos negativos) e ir especificándola (añadiendo términos al cuerpo) hasta hallar la consistencia.
- **Esta es la estrategia seguida en el algoritmo FOIL.**
- FOIL fue desarrollado por J.R. Quinlan y mejorado en colaboración con R.M. Cameron-Jones.

- La solución de un problema de Programación Lógica Inductiva puede ser un programa lógico H formado por cláusulas de Horn que describen el predicado al que se refieren los ejemplos (positivos y negativos) del problema.
- Una de las estrategias para obtener estas cláusulas consiste en proponer un cláusula lo más general posible (que, por tanto, no será consistente con el conjunto de ejemplos negativos) e ir especificándola (añadiendo términos al cuerpo) hasta hallar la consistencia.
- **Esta es la estrategia seguida en el algoritmo FOIL.**
- FOIL fue desarrollado por J.R. Quinlan y mejorado en colaboración con R.M. Cameron-Jones.
- Sobre la base de FOIL se han propuesto numerosos algoritmos.

- Esquema general de un algoritmo de generación de reglas por especificación:

```

1 function FOIL returns listaDeReglas
2 input Ep // Conjunto de ejemplos positivos
3 input En // Conjunto de ejemplos negativos
4 input Domain // Conjunto de predicados del dominio
5 input Pred // Predicado objetivo
6
7 begin
8     listaDeReglas := Vacía
9     while NoVacío(Ep)
10         regla := ReglaVacía(Pred)
11         while Cubre(regla, En)
12             regla := AnadirLiteral(regla, Domain)
13         endwhile
14         Ep := EliminarEjemplosCubiertos(Ep, regla)
15         listaDeReglas := AnadirRegla(listaDeReglas, regla)
16     endwhile
17     return listaDeReglas
18 end

```

- General scheme of a rule generation algorithm by specification:

```

1 function FOIL returns listOfRules
2 input  Ep // Set of positive examples
3 input  En // Set of negative examples
4 input  Domain // Set of domain predicates
5 input  Pred // Target predicate
6
7 begin
8     listOfRules := EmptySet
9     while Not Empty(Ep)
10         rule := EmptyRule(Pred)
11         while Satisfy(rule, En)
12             rule := AddLiteral(rule, Domain)
13         endwhile
14         Ep := RemoveCoveredPositives(Ep, rule)
15         listOfRules := AddRegla(listOfRules, rule)
16     endwhile
17     return listOfRules
18 end

```

- FOIL se basa en la suposición de mundo cerrado.
(aunque es posible definir el algoritmo sin asumir esta suposición).
- Esto quiere decir que no es necesario enumerar los ejemplos negativos. Cualquier instancia del predicado objetivo que esté formada por las constantes del conocimiento de dominio y que no se encuentre entre los ejemplos positivos se considera negativa.

- FOIL se basa en la suposición de mundo cerrado.
(aunque es posible definir el algoritmo sin asumir esta suposición).
- Esto quiere decir que no es necesario enumerar los ejemplos negativos. Cualquier instancia del predicado objetivo que esté formada por las constantes del conocimiento de dominio y que no se encuentre entre los ejemplos positivos se considera negativa.

Por ejemplo, consideremos el siguiente conjunto de hechos²

```
1 nieta(victor, sharon).  
2 padre(sharon, bob).  
3 padre(tom, bob).  
4 mujer(sharon).  
5 padre(bob, victor).
```

²La hipótesis de mundo cerrado significa que cualquier término `nieta(X, Y)`, donde las variables se sustituyan por las constantes del dominio (`victor`, `sharon`, `bob` y `tom`) será falsa excepto las que se especifiquen explícitamente como ciertas (`nieta(victor, sharon)`).

- FOIL parte de la regla más general:

$\text{nieta}(X, Y) :- .$

- Para añadir términos a la regla, FOIL considera los siguientes candidatos:

- $q(V_1, V_2, \dots, V_r)$, donde q es alguno de los predicados del dominio y V_i son variables donde al menos una debe estar ya presente en la regla
- $X = Y$, donde X e Y son variables que ya están presentes en la regla.
- La negación de los dos tipos de términos anteriores. (Si nos basamos en cláusulas de Horn, la negación de los predicados no estaría permitida)

- FOIL parte de la regla más general:

$\text{nieta}(X, Y) \text{ :- } .$

- Para añadir términos a la regla, FOIL considera los siguientes candidatos:

- $q(V_1, V_2, \dots, V_r)$, donde q es alguno de los predicados del dominio y V_i son variables donde al menos una debe estar ya presente en la regla
- $X = Y$, donde X e Y son variables que ya están presentes en la regla.
- La negación de los dos tipos de términos anteriores. (Si nos basamos en cláusulas de Horn, la negación de los predicados no estaría permitida)

- FOIL parte de la regla más general:

$\text{nieta}(X, Y) :- .$

- Para añadir términos a la regla, FOIL considera los siguientes candidatos:
 - $q(V_1, V_2, \dots, V_r)$, donde q es alguno de los predicados del dominio y V_i son variables donde al menos una debe estar ya presente en la regla
 - $X = Y$, donde X e Y son variables que ya están presentes en la regla.
 - La negación de los dos tipos de términos anteriores. (Si nos basamos en cláusulas de Horn, la negación de los predicados no estaría permitida)

- FOIL parte de la regla más general:

$\text{nieta}(X, Y) :- .$

- Para añadir términos a la regla, FOIL considera los siguientes candidatos:
 - $q(V_1, V_2, \dots, V_r)$, donde q es alguno de los predicados del dominio y V_i son variables donde al menos una debe estar ya presente en la regla
 - $X = Y$, donde X e Y son variables que ya están presentes en la regla.
 - La negación de los dos tipos de términos anteriores. (Si nos basamos en cláusulas de Horn, la negación de los predicados no estaría permitida)

- FOIL parte de la regla más general:

$\text{nieta}(X, Y) :- .$

- Para añadir términos a la regla, FOIL considera los siguientes candidatos:
 - $q(V_1, V_2, \dots, V_r)$, donde q es alguno de los predicados del dominio y V_i son variables donde al menos una debe estar ya presente en la regla
 - $X = Y$, donde X e Y son variables que ya están presentes en la regla.
 - La negación de los dos tipos de términos anteriores. (Si nos basamos en cláusulas de Horn, la negación de los predicados no estaría permitida)

FOIL(Ejemplos, Conocimiento-base, Predicado-objetivo(=P))

```
1 R = {}  
2 E = Ejemplos  
3 mientras Ep in E :  
4     Regla = CrearRegla(P(X1,...,Xn), {});  
5     mientras cubre(Regla, En):  
6         Literales = GenerarLiterales(BC, Regla)  
7         L = Mejor(Literales)  
8         Regla.add(L)  
9     R.add(Regla)  
10    E = QuitarCubiertos(E, Regla)  
11 devolver R
```

- Siguiendo el ejemplo, los candidatos serían los siguientes:

```
1  mujer(X) ,  
2  mujer(Y) ,  
3  padre(X,Y) ,  
4  padre(Y,X) ,  
5  padre(X, Z) ,  
6  padre(Z,X) ,  
7  padre(Y,Z) ,  
8  padre(Z,Y) ,  
9  X = Y ,  
10 X \= Y.
```


- Para seleccionar el término candidato, FOIL estudia todas las sustituciones posibles de las variables de la regla.
- La sustitución $X/victor, Y/sharon$ es correcta.
- La sustitución $X/bob, Y/tom$ es una evidencia negativa de la regla.
- Con 2 variables y 4 constantes existen 16 sustituciones posibles.

- Para seleccionar el término candidato, FOIL estudia todas las sustituciones posibles de las variables de la regla.
- La sustitución $X/victor, Y/sharon$ es correcta.
- La sustitución $X/bob, Y/tom$ es una evidencia negativa de la regla.
- Con 2 variables y 4 constantes existen 16 sustituciones posibles.

- Para seleccionar el término candidato, FOIL estudia todas las sustituciones posibles de las variables de la regla.
- La sustitución $X/victor, Y/sharon$ es correcta.
- La sustitución $X/bob, Y/tom$ es una evidencia negativa de la regla.
- Con 2 variables y 4 constantes existen 16 sustituciones posibles.

- Para seleccionar el término candidato, FOIL estudia todas las sustituciones posibles de las variables de la regla.
- La sustitución $X/victor, Y/sharon$ es correcta.
- La sustitución $X/bob, Y/tom$ es una evidencia negativa de la regla.
- Con 2 variables y 4 constantes existen 16 sustituciones posibles.

El término seleccionado es...

- el que tenga mayor ganancia de información:

$$ganancia = t \cdot (\log_2(\frac{p_{R'}}{p_{R'} + n_{R'}}) - \log_2(\frac{p_R}{p_R + n_R}))$$

where:

- R representa la regla inicial,
- R' la regla al añadir el término,
- p_R es el número de sustituciones positivas,
- n_R el número de sustituciones negativas y
- t es el número de ejemplos positivos cubiertos por R que permanecen cubiertos por R'.

El término seleccionado es...

- el que tenga mayor ganancia de información:

$$\text{ganancia} = t \cdot (\log_2(\frac{p_{R'}}{p_{R'} + n_{R'}}) - \log_2(\frac{p_R}{p_R + n_R}))$$

where:

- R representa la regla inicial,
- R' la regla al añadir el término,
- p_R es el número de sustituciones positivas,
- n_R el número de sustituciones negativas y
- t es el número de ejemplos positivos cubiertos por R que permanecen cubiertos por R'.

El término seleccionado es...

- el que tenga mayor ganancia de información:

$$ganancia = t \cdot (\log_2(\frac{p_{R'}}{p_{R'} + n_{R'}}) - \log_2(\frac{p_R}{p_R + n_R}))$$

where:

- R representa la regla inicial,
- R' la regla al añadir el término,
- p_R es el número de sustituciones positivas,
- n_R el número de sustituciones negativas y
- t es el número de ejemplos positivos cubiertos por R que permanecen cubiertos por R'.

El término seleccionado es...

- el que tenga mayor ganancia de información:

$$\text{ganancia} = t \cdot (\log_2(\frac{p_{R'}}{p_{R'} + n_{R'}}) - \log_2(\frac{p_R}{p_R + n_R}))$$

where:

- R representa la regla inicial,
- R' la regla al añadir el término,
- p_R es el número de sustituciones positivas,
- n_R el número de sustituciones negativas y
- t es el número de ejemplos positivos cubiertos por R que permanecen cubiertos por R'.

El término seleccionado es...

- el que tenga mayor ganancia de información:

$$ganancia = t \cdot (\log_2(\frac{p_{R'}}{p_{R'} + n_{R'}}) - \log_2(\frac{p_R}{p_R + n_R}))$$

where:

- R representa la regla inicial,
- R' la regla al añadir el término,
- p_R es el número de sustituciones positivas,
- n_R el número de sustituciones negativas y
- t es el número de ejemplos positivos cubiertos por R que permanecen cubiertos por R'.

El término seleccionado es...

- el que tenga mayor ganancia de información:

$$ganancia = t \cdot (\log_2(\frac{p_{R'}}{p_{R'} + n_{R'}}) - \log_2(\frac{p_R}{p_R + n_R}))$$

where:

- R representa la regla inicial,
- R' la regla al añadir el término,
- p_R es el número de sustituciones positivas,
- n_R el número de sustituciones negativas y
- t es el número de ejemplos positivos cubiertos por R que permanecen cubiertos por R'.

Ejemplo:

■ Conjunto de ejemplos positivos:

```
1 nieta(victor , sharon)
```

■ Conjunto de ejemplos negativos:

```
1 nieta(victor , victor). nieta(victor , bob). nieta(victor , tom).  
2 nieta(bob , victor). nieta(bob , bob). nieta(bob , tom). nieta(bob , sharon).  
3 nieta(tom , victor). nieta(tom , bob). nieta(tom , tom). nieta(tom , sharon).  
4 nieta(sharon , victor). nieta(sharon , bob). nieta(sharon , tom). nieta(sharon , sharon).
```

■ Dominio:

```
1 padre(sharon , bob). padre(tom , bob). padre(bob , victor ).  
2 mujer(sharon ).
```

- Estado inicial:

nieta(X, Y).

- $p_R = 1$
- $n_R = 15$

- Estado inicial:

nieta(X, Y).

- $p_R = 1$

- $n_R = 15$

- Estado inicial:

nieta(X, Y).

- $p_R = 1$
- $n_R = 15$

Primera iteración:

para todos $p_R = 1$ y $n_R = 15$				
Candidata	$p_{R'}$	$n_{R'}$	t	ganancia
nieta(X,Y):- mujer(X).	0	4	0	0
nieta(X,Y):- mujer(Y).	1	3	1	2.0
nieta(X,Y):- padre(X,Y).	0	3	0	0
nieta(X,Y):- padre(Y,X).	0	3	0	0
nieta(X,Y):- padre(X,Z).	0	12	0	0
nieta(X,Y):- padre(Z,X).	1	11	1	0.41
nieta(X,Y):- padre(Y,Z).	1	11	1	0.41
nieta(X,Y):- padre(Z,Y).	0	12	0	0
nieta(X,Y):- X = Y.	0	4	0	0
nieta(X,Y):- X \neq Y.	1	11	1	0.41

■ Segunda iteración:

para todos $p_R = 1$ y $n_R = 3$				
Candidata	P	N	t	ganancia
nieta(X,Y):- mujer(Y), mujer(X).	0	1	0	0
nieta(X,Y):- mujer(Y), padre(X,Y).	0	0	0	0
nieta(X,Y):- mujer(Y), padre(Y,X).	0	1	0	0
nieta(X,Y):- mujer(Y), padre(X,Z).	0	3	0	0
nieta(X,Y):- mujer(Y), padre(Z,X).	1	1	1	1
nieta(X,Y):- mujer(Y), padre(Y,Z).	1	3	1	0
nieta(X,Y):- mujer(Y), padre(Z,Y).	0	0	0	0
nieta(X,Y):- mujer(Y), $X = Y$.	0	1	0	0
nieta(X,Y):- mujer(Y), $X \neq Y$.	1	2	1	0.41

- **Tercera iteración:**
nieta(X,Y) :- mujer(Y), padre(Y,Z), padre(Z,X).
- En esta regla $p_{R'} = 1$ y $n_{R'} = 0$
- Al no haber ejemplos negativos cubiertos, paramos de aprender.

- Tercera iteración:
`nieta(X,Y) :- mujer(Y), padre(Y,Z), padre(Z,X).`
- En esta regla $p_{R'} = 1$ y $n_{R'} = 0$
- Al no haber ejemplos negativos cubiertos, paramos de aprender.

- Tercera iteración:
`nieta(X,Y) :- mujer(Y), padre(Y,Z), padre(Z,X).`
- En esta regla $p_{R'} = 1$ y $n_{R'} = 0$
- Al no haber ejemplos negativos cubiertos, paramos de aprender.

Mejoras propuestas en FOIL:

- Posibilidad de incorporar constantes:
 - Los literales estudiados sólo utilizan variables.
 - Muchos problemas se resuelven con casos base para ciertos valores
 - La solución es añadir literales “ $V=\text{constante}$ ” entre las opciones de FOIL
 - Es necesario incorporar restricciones a las constantes (indicar cuales pueden ser utilizadas en estos literales)
- Posibilidad de generar reglas recursivas:
 - Para generarlas es necesario considerar entre los literales al predicado de la cabeza

Mejoras propuestas en FOIL:

- Posibilidad de incorporar constantes:
 - Los literales estudiados sólo utilizan variables.
 - Muchos problemas se resuelven con casos base para ciertos valores
 - La solución es añadir literales “ $V=\text{constante}$ ” entre las opciones de FOIL
 - Es necesario incorporar restricciones a las constantes (indicar cuales pueden ser utilizadas en estos literales)
- Posibilidad de generar reglas recursivas:
 - Para generarlas es necesario considerar entre los literales al predicado de la cabeza

Mejoras propuestas en FOIL:

- Posibilidad de incorporar constantes:
 - Los literales estudiados sólo utilizan variables.
 - Muchos problemas se resuelven con casos base para ciertos valores
 - La solución es añadir literales “ $V=\text{constante}$ ” entre las opciones de FOIL
 - Es necesario incorporar restricciones a las constantes (indicar cuales pueden ser utilizadas en estos literales)
- Posibilidad de generar reglas recursivas:
 - Para generarlas es necesario considerar entre los literales al predicado de la cabeza

Mejoras propuestas en FOIL:

- Posibilidad de incorporar constantes:
 - Los literales estudiados sólo utilizan variables.
 - Muchos problemas se resuelven con casos base para ciertos valores
 - La solución es añadir literales “ $V=\text{constante}$ ” entre las opciones de FOIL
 - Es necesario incorporar restricciones a las constantes (indicar cuales pueden ser utilizadas en estos literales)
- Posibilidad de generar reglas recursivas:
 - Para generarlas es necesario considerar entre los literales al predicado de la cabeza

Mejoras propuestas en FOIL:

- Posibilidad de incorporar constantes:
 - Los literales estudiados sólo utilizan variables.
 - Muchos problemas se resuelven con casos base para ciertos valores
 - La solución es añadir literales “ $V=\text{constante}$ ” entre las opciones de FOIL
 - Es necesario incorporar restricciones a las constantes (indicar cuales pueden ser utilizadas en estos literales)
- Posibilidad de generar reglas recursivas:
 - Para generarlas es necesario considerar entre los literales al predicado de la cabeza

Mejoras propuestas en FOIL:

- Posibilidad de incorporar constantes:
 - Los literales estudiados sólo utilizan variables.
 - Muchos problemas se resuelven con casos base para ciertos valores
 - La solución es añadir literales “ $V=\text{constante}$ ” entre las opciones de FOIL
 - Es necesario incorporar restricciones a las constantes (indicar cuales pueden ser utilizadas en estos literales)
- Posibilidad de generar reglas recursivas:
 - Para generarlas es necesario considerar entre los literales al predicado de la cabeza

Mejoras propuestas en FOIL:

- Posibilidad de incorporar constantes:
 - Los literales estudiados sólo utilizan variables.
 - Muchos problemas se resuelven con casos base para ciertos valores
 - La solución es añadir literales “ $V=\text{constante}$ ” entre las opciones de FOIL
 - Es necesario incorporar restricciones a las constantes (indicar cuales pueden ser utilizadas en estos literales)
- Posibilidad de generar reglas recursivas:
 - Para generarlas es necesario considerar entre los literales al **predicado de la cabeza**

Mejoras propuestas en FOIL:

- Posibilidad de incorporar constantes:
 - Los literales estudiados sólo utilizan variables.
 - Muchos problemas se resuelven con casos base para ciertos valores
 - La solución es añadir literales “ $V=\text{constante}$ ” entre las opciones de FOIL
 - Es necesario incorporar restricciones a las constantes (indicar cuales pueden ser utilizadas en estos literales)
- Posibilidad de generar reglas recursivas:
 - Para generarlas es necesario considerar entre los literales al **predicado de la cabeza**

Resolución inversa en lógica proposicional

- Consiste en considerar la inducción como la operación inversa de la deducción.
- Regla de resolución en lógica proposicional:

$$\frac{\neg p \vee q \vee r \dots \quad p \vee s \vee t \dots}{q \vee r \vee \dots \vee s \vee t \vee \dots}$$

- Si nos limitamos a cláusulas de Horn, toda cláusula debe tener un único literal positivo.

El resultado de la regla de resolución entre cláusulas de Horn es otra cláusula de Horn.

$$\frac{\neg p \vee q \vee \neg r \dots \quad p \vee \neg s \vee \neg t \dots}{q \vee \neg r \vee \dots \vee \neg s \vee \neg t \vee \dots} \quad \frac{p \wedge r \wedge \dots \rightarrow q \quad s \wedge t \dots \rightarrow p}{r \wedge \dots \wedge s \wedge t \dots \rightarrow q}$$

Resolución inversa en lógica proposicional

- Consiste en considerar la inducción como la operación inversa de la deducción.
- Regla de resolución en lógica proposicional:

$$\frac{\neg p \vee q \vee r \dots \quad p \vee s \vee t \dots}{q \vee r \vee \dots \vee s \vee t \vee \dots}$$

- Si nos limitamos a cláusulas de Horn, toda cláusula debe tener un único literal positivo.

El resultado de la regla de resolución entre cláusulas de Horn es otra cláusula de Horn.

$$\frac{\neg p \vee q \vee \neg r \dots \quad p \vee \neg s \vee \neg t \dots}{q \vee \neg r \vee \dots \vee \neg s \vee \neg t \vee \dots} \quad \frac{p \wedge r \wedge \dots \rightarrow q \quad s \wedge t \dots \rightarrow p}{r \wedge \dots \wedge s \wedge t \dots \rightarrow q}$$

Resolución inversa en lógica proposicional

- Consiste en considerar la inducción como la operación inversa de la deducción.
- Regla de resolución en lógica proposicional:

$$\frac{\neg p \vee q \vee r \dots \quad p \vee s \vee t \dots}{q \vee r \vee \dots \vee s \vee t \vee \dots}$$

- Si nos limitamos a cláusulas de Horn, toda cláusula debe tener un único literal positivo.

El resultado de la regla de resolución entre cláusulas de Horn es otra cláusula de Horn.

$$\frac{\neg p \vee q \vee \neg r \dots \quad p \vee \neg s \vee \neg t \dots}{q \vee \neg r \vee \dots \vee \neg s \vee \neg t \vee \dots} \quad \frac{p \wedge r \wedge \dots \rightarrow q \quad s \wedge t \dots \rightarrow p}{r \wedge \dots \wedge s \wedge t \dots \rightarrow q}$$

Operador de resolución inversa $O(C_1, C_2)$

- Dadas C_1 y C_2 , el operador debe generar una cláusula $O(C_1, C_2)$ tal que

$$O(C_1, C_2) \wedge C_1 \vdash C_2$$

- Para aplicar el operador de resolución inversa hay que buscar los literales comunes en C_1 y C_2 y los literales diferentes en C_1 y C_2 .

Operador de resolución inversa $O(C_1, C_2)$

- Dadas C_1 y C_2 , el operador debe generar una cláusula $O(C_1, C_2)$ tal que

$$O(C_1, C_2) \wedge C_1 \vdash C_2$$

- Para aplicar el operador de resolución inversa hay que buscar los literales comunes en C_1 y C_2 y los literales diferentes en C_1 y C_2 .

Resolución inversa en lógica proposicional

Operador de resolución inversa $O(C_1, C_2)$

- Si los literales iguales son negativos:

- $C_1 = h1 \vee \neg r1 \vee \neg r2$

- $C_2 = h2 \vee \neg q1 \vee \neg q2 \vee \neg r1 \vee \neg r2$

- $O(C_1, C_2) = h2 \vee \neg q1 \vee \neg q2 \vee \neg h1$

- Si los literales iguales incluyen al literal positivo:

- $C_1 = h1 \vee \neg r1 \vee \neg p1$

- $C_2 = h1 \vee \neg r1 \vee \neg q1 \vee \neg q2$

- $O(C_1, C_2) = p1 \vee \neg q1 \vee \neg q2$

- Para que el resultado sea una cláusula de horn, debe haber como

Resolución inversa en lógica proposicional

Operador de resolución inversa $O(C_1, C_2)$

- Si los literales iguales son negativos:

- $C_1 = h1 \vee \neg r1 \vee \neg r2$

- $C_2 = h2 \vee \neg q1 \vee \neg q2 \vee \neg r1 \vee \neg r2$

- $O(C_1, C_2) = h2 \vee \neg q1 \vee \neg q2 \vee \neg h1$

- Si los literales iguales incluyen al literal positivo:

- $C_1 = h1 \vee \neg r1 \vee \neg p1$

- $C_2 = h1 \vee \neg r1 \vee \neg q1 \vee \neg q2$

- $O(C_1, C_2) = p1 \vee \neg q1 \vee \neg q2$

- Para que el resultado sea una cláusula de horn, debe haber como

Resolución inversa en lógica proposicional

Operador de resolución inversa $O(C_1, C_2)$

- Si los literales iguales son negativos:

- $C_1 = h1 \vee \neg r1 \vee \neg r2$

- $C_2 = h2 \vee \neg q1 \vee \neg q2 \vee \neg r1 \vee \neg r2$

- $O(C_1, C_2) = h2 \vee \neg q1 \vee \neg q2 \vee \neg h1$

- Si los literales iguales incluyen al literal positivo:

- $C_1 = h1 \vee \neg r1 \vee \neg p1$

- $C_2 = h1 \vee \neg r1 \vee \neg q1 \vee \neg q2$

- $O(C_1, C_2) = p1 \vee \neg q1 \vee \neg q2$

- Para que el resultado sea una cláusula de horn, debe haber como

Resolución inversa en lógica proposicional

Operador de resolución inversa $O(C_1, C_2)$

- Si los literales iguales son negativos:

- $C_1 = h1 \vee \neg r1 \vee \neg r2$

- $C_2 = h2 \vee \neg q1 \vee \neg q2 \vee \neg r1 \vee \neg r2$

- $O(C_1, C_2) = h2 \vee \neg q1 \vee \neg q2 \vee \neg h1$

- Si los literales iguales incluyen al literal positivo:

- $C_1 = h1 \vee \neg r1 \vee \neg p1$

- $C_2 = h1 \vee \neg r1 \vee \neg q1 \vee \neg q2$

- $O(C_1, C_2) = p1 \vee \neg q1 \vee \neg q2$

- Para que el resultado sea una cláusula de horn, debe haber como

Resolución inversa en lógica proposicional

Operador de resolución inversa $O(C_1, C_2)$

- Si los literales iguales son negativos:

- $C_1 = h1 \vee \neg r1 \vee \neg r2$

- $C_2 = h2 \vee \neg q1 \vee \neg q2 \vee \neg r1 \vee \neg r2$

- $O(C_1, C_2) = h2 \vee \neg q1 \vee \neg q2 \vee \neg h1$

- Si los literales iguales incluyen al literal positivo:

- $C_1 = h1 \vee \neg r1 \vee \neg p1$

- $C_2 = h1 \vee \neg r1 \vee \neg q1 \vee \neg q2$

- $O(C_1, C_2) = p1 \vee \neg q1 \vee \neg q2$

- Para que el resultado sea una cláusula de horn, debe haber como

Resolución inversa en lógica proposicional

Operador de resolución inversa $O(C_1, C_2)$

- Si los literales iguales son negativos:

- $C_1 = h1 \vee \neg r1 \vee \neg r2$

- $C_2 = h2 \vee \neg q1 \vee \neg q2 \vee \neg r1 \vee \neg r2$

- $O(C_1, C_2) = h2 \vee \neg q1 \vee \neg q2 \vee \neg h1$

- Si los literales iguales incluyen al literal positivo:

- $C_1 = h1 \vee \neg r1 \vee \neg p1$

- $C_2 = h1 \vee \neg r1 \vee \neg q1 \vee \neg q2$

- $O(C_1, C_2) = p1 \vee \neg q1 \vee \neg q2$

- Para que el resultado sea una cláusula de horn, debe haber como

Resolución inversa en lógica proposicional

Operador de resolución inversa $O(C_1, C_2)$

- Si los literales iguales son negativos:

- $C_1 = h1 \vee \neg r1 \vee \neg r2$

- $C_2 = h2 \vee \neg q1 \vee \neg q2 \vee \neg r1 \vee \neg r2$

- $O(C_1, C_2) = h2 \vee \neg q1 \vee \neg q2 \vee \neg h1$

- Si los literales iguales incluyen al literal positivo:

- $C_1 = h1 \vee \neg r1 \vee \neg p1$

- $C_2 = h1 \vee \neg r1 \vee \neg q1 \vee \neg q2$

- $O(C_1, C_2) = p1 \vee \neg q1 \vee \neg q2$

- Para que el resultado sea una cláusula de horn, debe haber como

Resolución inversa en lógica proposicional

Operador de resolución inversa $O(C_1, C_2)$

- Si los literales iguales son negativos:

- $C_1 = h1 \vee \neg r1 \vee \neg r2$

- $C_2 = h2 \vee \neg q1 \vee \neg q2 \vee \neg r1 \vee \neg r2$

- $O(C_1, C_2) = h2 \vee \neg q1 \vee \neg q2 \vee \neg h1$

- Si los literales iguales incluyen al literal positivo:

- $C_1 = h1 \vee \neg r1 \vee \neg p1$

- $C_2 = h1 \vee \neg r1 \vee \neg q1 \vee \neg q2$

- $O(C_1, C_2) = p1 \vee \neg q1 \vee \neg q2$

- Para que el resultado sea una cláusula de horn, debe haber como

Resolución inversa en lógica proposicional

Operador de resolución inversa $O(C_1, C_2)$

- Si los literales iguales son negativos:

- $C_1 = h1 \vee \neg r1 \vee \neg r2$

- $C_2 = h2 \vee \neg q1 \vee \neg q2 \vee \neg r1 \vee \neg r2$

- $O(C_1, C_2) = h2 \vee \neg q1 \vee \neg q2 \vee \neg h1$

- Si los literales iguales incluyen al literal positivo:

- $C_1 = h1 \vee \neg r1 \vee \neg p1$

- $C_2 = h1 \vee \neg r1 \vee \neg q1 \vee \neg q2$

- $O(C_1, C_2) = p1 \vee \neg q1 \vee \neg q2$

- Para que el resultado sea una cláusula de horn, debe haber como

Resolución inversa en lógica proposicional

- Dado un conjunto de cláusulas C_i , pueden existir muchas formas de aplicar el operador de implicación inversa.
- Los algoritmos basados en esta estrategia deben seleccionar la mejor de las opciones
- (la que mantenga la consistencia en ϵ^- y cubra más ejemplos en ϵ^+)

- Dado un conjunto de cláusulas C_i , pueden existir muchas formas de aplicar el operador de implicación inversa.
- Los algoritmos basados en esta estrategia deben seleccionar la mejor de las opciones
- (la que mantenga la consistencia en ϵ^- y cubra más ejemplos en ϵ^+)

- Dado un conjunto de cláusulas C_i , pueden existir muchas formas de aplicar el operador de implicación inversa.
- Los algoritmos basados en esta estrategia deben seleccionar la mejor de las opciones
- (la que mantenga la consistencia en ϵ^- y cubra más ejemplos en ϵ^+)

Resolución inversa en lógica de primer orden

- Regla de resolución en lógica de primer orden:

$$\frac{\neg p1 \vee q1 \vee r1 \dots \quad p2 \vee s2 \vee t2 \dots}{(q1 \vee r1 \vee \dots \vee s2 \vee t2 \vee \dots)\theta} \quad \theta \equiv \text{unificador tal que } (p1\theta = p2\theta)$$

- Si nos limitamos a cláusulas de Horn, toda cláusula debe tener un único literal positivo.
- El resultado de la regla de resolución entre cláusulas de Horn es otra cláusula de Horn.

$$\frac{\neg p1 \vee q1 \vee \neg r1 \dots \quad p2 \vee \neg s2 \vee \neg t2 \dots}{(q1 \vee \neg r1 \vee \dots \vee \neg s2 \vee \neg t2 \vee \dots)\theta} \quad \theta \equiv \text{unificador tal que } (p1\theta = p2\theta)$$

Resolución inversa en lógica de primer orden

- Regla de resolución en lógica de primer orden:

$$\frac{\neg p1 \vee q1 \vee r1 \dots \quad p2 \vee s2 \vee t2 \dots}{(q1 \vee r1 \vee \dots \vee s2 \vee t2 \vee \dots)\theta} \quad \theta \equiv \text{unificador tal que } (p1\theta = p2\theta)$$

- Si nos limitamos a cláusulas de Horn, toda cláusula debe tener un único literal positivo.
- El resultado de la regla de resolución entre cláusulas de Horn es otra cláusula de Horn.

$$\frac{\neg p1 \vee q1 \vee \neg r1 \dots \quad p2 \vee \neg s2 \vee \neg t2 \dots}{(q1 \vee \neg r1 \vee \dots \vee \neg s2 \vee \neg t2 \vee \dots)\theta} \quad \theta \equiv \text{unificador tal que } (p1\theta = p2\theta)$$

Resolución inversa en lógica de primer orden

- Regla de resolución en lógica de primer orden:

$$\frac{\neg p1 \vee q1 \vee r1 \dots \quad p2 \vee s2 \vee t2 \dots}{(q1 \vee r1 \vee \dots \vee s2 \vee t2 \vee \dots)\theta} \quad \theta \equiv \text{unificador tal que } (p1\theta = p2\theta)$$

- Si nos limitamos a cláusulas de Horn, toda cláusula debe tener un único literal positivo.
- El resultado de la regla de resolución entre cláusulas de Horn es otra cláusula de Horn.

$$\frac{\neg p1 \vee q1 \vee \neg r1 \dots \quad p2 \vee \neg s2 \vee \neg t2 \dots}{(q1 \vee \neg r1 \vee \dots \vee \neg s2 \vee \neg t2 \vee \dots)\theta} \quad \theta \equiv \text{unificador tal que } (p1\theta = p2\theta)$$

Resolución inversa en lógica de primer orden

Operador de resolución inversa $O(C_1, C_2)$.

Dadas C_1 y C_2 ,

- el operador debe generar una cláusula $O(C_1, C_2)$ tal que

$$O(C_1, C_2) \wedge C_1 \vdash C_2$$

- Para aplicar el operador de resolución inversa hay que buscar los literales comunes en C_1 y C_2 (r_i) y los literales diferentes en C_1 (p_i) y C_2 (q_i).
- Hay que buscar la sustitución θ tal que los literales comunes unifiquen ($r_{1i}\theta = r_{2i}\theta$).
- La sustitución se puede descomponer en dos partes (θ_1 y θ_2), referidas a las variables de C_1 y C_2 , respectivamente.

$$C_1 = p_1 \vee \neg r_{11} \vee \neg r_{12}$$

$$C_2 = q_1 \vee \neg r_{21} \vee \neg r_{22}$$

Resolución inversa en lógica de primer orden

Operador de resolución inversa $O(C_1, C_2)$.

Dadas C_1 y C_2 ,

- el operador debe generar una cláusula $O(C_1, C_2)$ tal que

$$O(C_1, C_2) \wedge C_1 \vdash C_2$$

- Para aplicar el operador de resolución inversa hay que buscar los literales comunes en C_1 y C_2 (r_i) y los literales diferentes en C_1 (p_i) y C_2 (q_i).
- Hay que buscar la sustitución θ tal que los literales comunes unifiquen ($r_{1i}\theta = r_{2i}\theta$).
- La sustitución se puede descomponer en dos partes (θ_1 y θ_2), referidas a las variables de C_1 y C_2 , respectivamente.

$$C_1 = p_1 \vee \neg r_{11} \vee \neg r_{12}$$

$$C_2 = q_1 \vee \neg r_{21} \vee \neg r_{22}$$

Resolución inversa en lógica de primer orden

Operador de resolución inversa $O(C_1, C_2)$.

Dadas C_1 y C_2 ,

- el operador debe generar una cláusula $O(C_1, C_2)$ tal que

$$O(C_1, C_2) \wedge C_1 \vdash C_2$$

- Para aplicar el operador de resolución inversa hay que buscar los literales comunes en C_1 y C_2 (r_i) y los literales diferentes en C_1 (p_i) y C_2 (q_i).
- Hay que buscar la sustitución θ tal que los literales comunes unifiquen ($r_{1i}\theta = r_{2i}\theta$).
- La sustitución se puede descomponer en dos partes (θ_1 y θ_2), referidas a las variables de C_1 y C_2 , respectivamente.

$$C_1 = p_1 \vee \neg r_{11} \vee \neg r_{12}$$

$$C_2 = q_1 \vee \neg r_{21} \vee \neg r_{22}$$

Resolución inversa en lógica de primer orden

Operador de resolución inversa $O(C_1, C_2)$.

Dadas C_1 y C_2 ,

- el operador debe generar una cláusula $O(C_1, C_2)$ tal que

$$O(C_1, C_2) \wedge C_1 \vdash C_2$$

- Para aplicar el operador de resolución inversa hay que buscar los literales comunes en C_1 y C_2 (r_i) y los literales diferentes en C_1 (p_i) y C_2 (q_i).
- Hay que buscar la sustitución θ tal que los literales comunes unifiquen ($r_{1i}\theta = r_{2i}\theta$).
- La sustitución se puede descomponer en dos partes (θ_1 y θ_2), referidas a las variables de C_1 y C_2 , respectivamente.

$$\bullet C_1 = p_1 \vee \neg r_{11} \vee \neg r_{12}$$

$$\bullet C_2 = q_1 \vee \neg q_2 \vee \neg q_3 \vee \neg r_{21} \vee \neg r_{22}$$

Resolución inversa en lógica de primer orden

Operador de resolución inversa $O(C_1, C_2)$.

Dadas C_1 y C_2 ,

- el operador debe generar una cláusula $O(C_1, C_2)$ tal que

$$O(C_1, C_2) \wedge C_1 \vdash C_2$$

- Para aplicar el operador de resolución inversa hay que buscar los literales comunes en C_1 y C_2 (r_i) y los literales diferentes en C_1 (p_i) y C_2 (q_i).
- Hay que buscar la sustitución θ tal que los literales comunes unifiquen ($r_{1i}\theta = r_{2i}\theta$).
- La sustitución se puede descomponer en dos partes (θ_1 y θ_2), referidas a las variables de C_1 y C_2 , respectivamente.

$$\bullet C_1 = p_1 \vee \neg r_{11} \vee \neg r_{12}$$

$$\bullet C_2 = q_1 \vee \neg q_2 \vee \neg q_3 \vee \neg r_{21} \vee \neg r_{22}$$

Resolución inversa en lógica de primer orden

Operador de resolución inversa $O(C_1, C_2)$.

Dadas C_1 y C_2 ,

- el operador debe generar una cláusula $O(C_1, C_2)$ tal que

$$O(C_1, C_2) \wedge C_1 \vdash C_2$$

- Para aplicar el operador de resolución inversa hay que buscar los literales comunes en C_1 y C_2 (r_i) y los literales diferentes en C_1 (p_i) y C_2 (q_i).
- Hay que buscar la sustitución θ tal que los literales comunes unifiquen ($r_{1i}\theta = r_{2i}\theta$).
- La sustitución se puede descomponer en dos partes (θ_1 y θ_2), referidas a las variables de C_1 y C_2 , respectivamente.

$$\bullet C_1 = p_1 \vee \neg r_{11} \vee \neg r_{12}$$

$$\bullet C_2 = q_1 \vee \neg q_2 \vee \neg q_3 \vee \neg r_{21} \vee \neg r_{22}$$

Resolución inversa en lógica de primer orden

Operador de resolución inversa $O(C_1, C_2)$.

Dadas C_1 y C_2 ,

- el operador debe generar una cláusula $O(C_1, C_2)$ tal que

$$O(C_1, C_2) \wedge C_1 \vdash C_2$$

- Para aplicar el operador de resolución inversa hay que buscar los literales comunes en C_1 y C_2 (r_i) y los literales diferentes en C_1 (p_i) y C_2 (q_i).
- Hay que buscar la sustitución θ tal que los literales comunes unifiquen ($r_{1i}\theta = r_{2i}\theta$).
- La sustitución se puede descomponer en dos partes (θ_1 y θ_2), referidas a las variables de C_1 y C_2 , respectivamente.

$$\bullet C_1 = p_1 \vee \neg r_{11} \vee \neg r_{12}$$

$$\bullet C_2 = q_1 \vee \neg q_2 \vee \neg q_3 \vee \neg r_{21} \vee \neg r_{22}$$

Resolución inversa en lógica de primer orden

En logica de primer orden, el operador de resolución inversa es:

- Si $(C_1 = L_1 \vee R)$ y $(C_2 = L_2 \vee R)$ y $\theta = \theta_1\theta_2$ entonces

$$C = \{(C_2 - R\theta_1)\theta_2^{-1}\} \cup \{\neg L_1\theta_1\theta_2^{-1}\}$$

- De nuevo, pueden existir muchas formas de aplicar este operador.
- Las herramientas MARVIN ([Sammur y Banerji, 86]) y CIGOL ([Muggleton y Buntine, 88]) se basan en este algoritmo.

Resolución inversa en lógica de primer orden

En logica de primer orden, el operador de resolución inversa es:

- Si $(C_1 = L_1 \vee R)$ y $(C_2 = L_2 \vee R)$ y $\theta = \theta_1\theta_2$ entonces

$$C = \{(C_2 - R\theta_1)\theta_2^{-1}\} \cup \{\neg L_1\theta_1\theta_2^{-1}\}$$

- De nuevo, pueden existir muchas formas de aplicar este operador.
- Las herramientas MARVIN ([Sammur y Banerji, 86]) y CIGOL ([Muggleton y Buntine, 88]) se basan en este algoritmo.

En logica de primer orden, el operador de resolución inversa es:

- Si $(C_1 = L_1 \vee R)$ y $(C_2 = L_2 \vee R)$ y $\theta = \theta_1\theta_2$ entonces

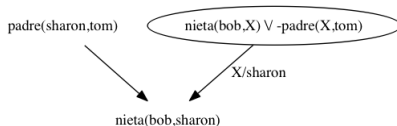
$$C = \{(C_2 - R\theta_1)\theta_2^{-1}\} \cup \{\neg L_1\theta_1\theta_2^{-1}\}$$

- De nuevo, pueden existir muchas formas de aplicar este operador.
- Las herramientas MARVIN ([Sammut y Banerji, 86]) y CIGOL ([Muggleton y Buntine, 88]) se basan en este algoritmo.

Ejemplo

- Si $C_1 = \text{padre}(\text{sharon}, \text{tom})$,
- $C_2 = \text{nieta}(\text{bob}, \text{sharon})$
- y $\theta = \{X/\text{sharon}\}$
- entonces $C_3 = \text{nieta}(\text{bob}, X) \vee \neg \text{padre}(X, \text{tom})$

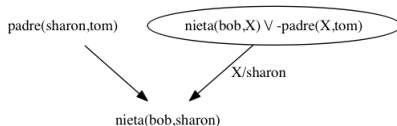
$(\text{nieta}(\text{bob}, X) : \neg \text{padre}(X, \text{tom}))$



Ejemplo

- Si $C_1 = \text{padre}(\text{sharon}, \text{tom})$,
- $C_2 = \text{nieta}(\text{bob}, \text{sharon})$
- y $\theta = \{X/\text{sharon}\}$
- entonces $C_3 = \text{nieta}(\text{bob}, X) \vee \neg \text{padre}(X, \text{tom})$

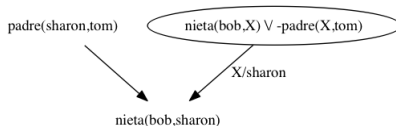
$(\text{nieta}(\text{bob}, X) : \neg \text{padre}(X, \text{tom}))$



Ejemplo

- Si $C_1 = \text{padre}(\text{sharon}, \text{tom})$,
- $C_2 = \text{nieta}(\text{bob}, \text{sharon})$
- y $\theta = \{X/\text{sharon}\}$
- entonces $C_3 = \text{nieta}(\text{bob}, X) \vee \neg \text{padre}(X, \text{tom})$

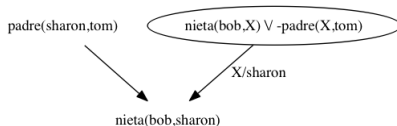
$(\text{nieta}(\text{bob}, X) : \neg \text{padre}(X, \text{tom}))$



Ejemplo

- Si $C_1 = \text{padre}(\text{sharon}, \text{tom})$,
- $C_2 = \text{nieta}(\text{bob}, \text{sharon})$
- y $\theta = \{X/\text{sharon}\}$
- entonces $C_3 = \text{nieta}(\text{bob}, X) \vee \neg \text{padre}(X, \text{tom})$

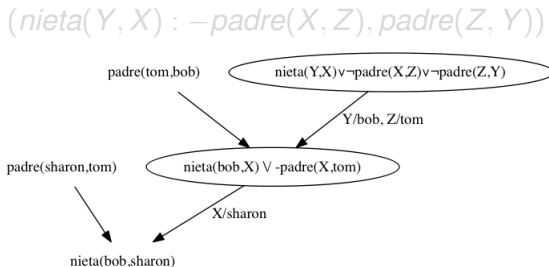
$$(\text{nieta}(\text{bob}, X) : \neg \text{padre}(X, \text{tom}))$$



Ejemplo

- Si $C_4 = \text{padre}(\text{tom}, \text{bob})$
- $C_3 = \text{nieta}(\text{bob}, X) \vee \neg \text{padre}(X, \text{tom})$
- y $\theta = \{Y/\text{bob}, Z/\text{tom}\}$
- entonces

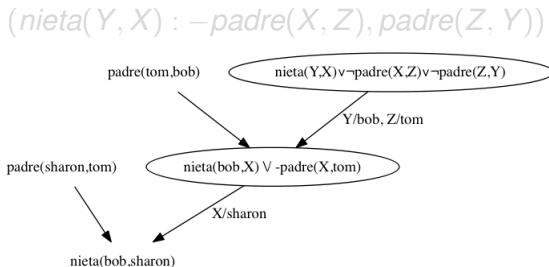
$$C_5 = \text{nieta}(Y, X) \vee \neg \text{padre}(X, Z) \vee \neg \text{padre}(Z, Y)$$



Ejemplo

- Si $C_4 = \text{padre}(\text{tom}, \text{bob})$
- $C_3 = \text{nieta}(\text{bob}, X) \vee \neg \text{padre}(X, \text{tom})$
- y $\theta = \{Y/\text{bob}, Z/\text{tom}\}$
- entonces

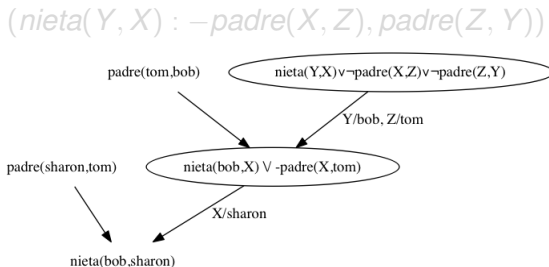
$$C_5 = \text{nieta}(Y, X) \vee \neg \text{padre}(X, Z) \vee \neg \text{padre}(Z, Y)$$



Ejemplo

- Si $C_4 = \text{padre}(\text{tom}, \text{bob})$
- $C_3 = \text{nieta}(\text{bob}, X) \vee \neg \text{padre}(X, \text{tom})$
- y $\theta = \{Y/\text{bob}, Z/\text{tom}\}$
- entonces

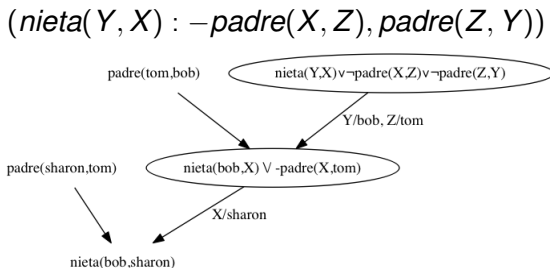
$$C_5 = \text{nieta}(Y, X) \vee \neg \text{padre}(X, Z) \vee \neg \text{padre}(Z, Y)$$



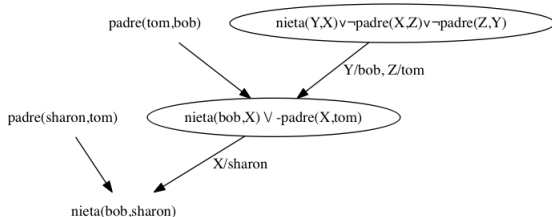
Ejemplo

- Si $C_4 = \text{padre}(\text{tom}, \text{bob})$
- $C_3 = \text{nieta}(\text{bob}, X) \vee \neg \text{padre}(X, \text{tom})$
- y $\theta = \{Y/\text{bob}, Z/\text{tom}\}$
- entonces

$$C_5 = \text{nieta}(Y, X) \vee \neg \text{padre}(X, Z) \vee \neg \text{padre}(Z, Y)$$



Ejemplo



$$(nieta(Y, X) : \neg padre(X, Z), padre(Z, Y))$$

Absorción:

$$\frac{q \leftarrow A \quad p \leftarrow A, B}{q \leftarrow A \quad p \leftarrow q, B}$$

Identificación:

$$\frac{q \leftarrow A, B \quad p \leftarrow A, q}{q \leftarrow B \quad p \leftarrow A, q}$$

Intra-construcción:

$$\frac{p \leftarrow A, B \quad p \leftarrow A, C}{q \leftarrow B \quad p \leftarrow A, q \quad q \leftarrow C}$$

Inter-construcción:

$$\frac{p \leftarrow A, B \quad q \leftarrow A, C}{p \leftarrow r, B \quad r \leftarrow A \quad q \leftarrow r, C}$$

¿ Alguna idea ?