Tema 7
Programación Lógica Inductiva

Gonzalo A. Aranda-Corral

Ciencias de la Computación e Inteligencia Artificial
Universidad de Huelva

diciembre 2020

Índice

Definiciones

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

Ejemplo:

```
Temperatura es alta
Cuello es recto
Clase es A
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.

Definiciones

- 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 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 (

Definiciones

- Una fórmula bien formada 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 2N filas. 2^N
- Las tablas de verdad de los conectivos lógicos son las siguientes:

P	Q	P ∨ Q	P ∧ Q	¬ P	P→Q	P↔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

Definiciones

- 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.
- 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 decidible si se puede demostrar si es válida o no en un número finito de pasos.
- Se dice que un sistema lógico es consistente si no permite demostrar simultáneamente una fórmula y su contraria.
- La lógica proposicional es decidible, porque existe un algoritmo (la tabla de verdad) que permite demostrar la validad de cualquier fórmula en un número finito de pasos

Definiciones

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

Definiciones

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)
- $a \vee -a = T$ (Tercio excluso)
- $a \wedge -a = F$ (No contradicción)
- $a \wedge a = a$; $a \vee a = a$; $a \vee (a \wedge b) = a$; $a \wedge (a \vee b) = a$;

Definiciones

- 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**.

Definiciones

Algoritmo para transformar una fórmula lógica en forma normal conjuntiva.

I. Sustituir toda implicación ($a \implies b$) por ($-a \vee b$).
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 = $\{A \implies B, A\}$ $$\frac{(A \implies B) \qquad A}{B}$$

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

Inferencia

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.

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

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

Deducción.

- **Resolución**:

$$\frac{\neg p \vee q \vee r \ldots \qquad p \vee s \vee t \ldots}{q \vee r \vee \ldots \vee s \vee t \vee \ldots}$$

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 ($BC \vdash Q$)?

Inferencia

- 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 resolucion:

```
function Resolucion(BC, objetivo) returns boolean
input:
        BC // Base de conocimientos

objetivo // expresion que se pretende demostrar
        BC <-- BC /\ -objetivo

repeat
        c1, c2 <-- EscogerClausulas(BC)
        resolvente <-- ReglaDeResolucion(c1,c2)
        BC <-- BC /\ resolvente
until resolvente = vacia
return True
```

Inferencia

Demostración por resolucion:

- para evitar bucles infinitos habría que controlar que no se repitieran c1 y c2 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:

```
p
q /\ r
p /\ q --> s
r /\ s --> t
```

Objetivo: demostrar que t es cierto

Inferencia

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

```
p
q
r
—p \/ —q \/ s
—r \/ —s \/ t
```

2º paso: añadir la negación del objetivo

```
(6) —t
```

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

```
(7) —r \/ —s
```

Inferencia

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

```
(8) —s
```

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

```
(9) —p \/ —q
```

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

```
(10) —p
```

7º paso: aplicar la regla de resolución

```
(—p ), ( p )
```

¡CONTRADICCIÓN! Lo que demuestra t por refutación

Definiciones

- 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

Definiciones

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

Definiciones

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 (nº 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)

Definiciones

Átomo (equivale a las proposiciones)

- Igualdad o identidad:
  - Permite incorporar la condición de que un término sea igual a otro
  - Ejemplo: x = y, Padre(Juan) = Pedro, 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

Definiciones

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

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

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

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

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

Inferencia

- Sustitución: es una secuencia finita de asociaciones entre variables y términos. θ = V1/ t1 , V2/ t2 , . . . , Vn/ tn
- Aplicación de una sustitución θ a una clausula C: consiste en sustituir cada instancia de la variable Vi en la clausula C por el término ti asociado en la sustitución. Se denota Cθ.
- Unificación: se dice que dos átomos P1 y P2 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).

Inferencia

- Regla de resolución:
  - $\neg p(x) \vee \alpha(x)$
  - $p(y) \vee \beta(y) \theta = UnificadorMasGeneral(p(x)/p(y))$
  - $(\alpha(x) \vee \beta(y))\theta$

- Ejemplo:
  - $\neg Hombre(x) \vee Mortal(x)$
  - $Hombre(Socrates)$
  - $\theta = x/Socrates$
  - $Mortal(Socrates)$

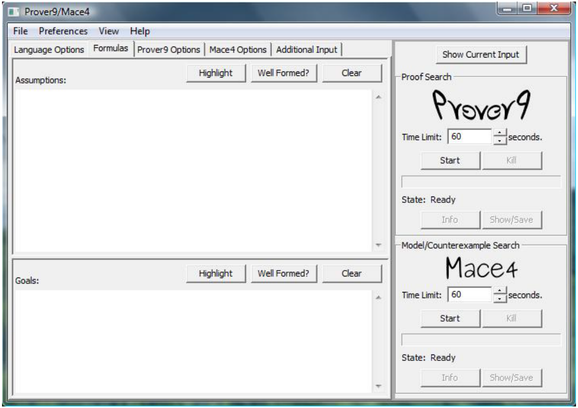
Inferencia

- Eliminación universal:
  - $\forall x : a(x)$
  - $\theta(x/Constante, a(x))$
  - Ejemplo $\forall x : Gusta(x, Helado)$
  - $Gusta(Juan, Helado)$
- Introducción existencial:
  - $a(Constante)$
  - $\exists x : a(x)$

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 ditribuye de manera gratuita en:

Prover9


Prover9/Mace4
File Preferences View Help
Language Options
Formulas
Prover9 Options
Mace4 Options
Additional Input
Assumptions:
Highlight
Well Formed?
Clear
Goals:
Highlight
Well Formed?
Clear
Show Current Input
Proof Search
Prover9
Time Limit: 60 seconds.
Start
Kill
State: Ready
Info
Show/Save
Model/Counterexample Search
Mace4
Time Limit: 60 seconds.
Start
Kill
State: Ready
Info
Show/Save

Prover9

La sintaxis de Prover9 es la siguiente:

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

Prover9

Ejemplo de entrada de Prover9:

Hipotesis

```
exists x ( Perro(x) & Posee(Juan,x) ).
all x ( (exists y (Perro(y) & Posee(x,y)) ) ->
              AmaALosAnimales(x) ).
all x ( AmaALosAnimales(x) -> (all y (Animal(y)
                 -> -Mata(x,y) ) ) ).
Gato(Tuna) & (Mata(Juan,Tuna) | Mata(Curiosidad,Tuna)).
all x ( Gato(x) -> Animal(x) ).
all x ( Perro(x) -> Animal(x) ).
```

Objetivos

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

Prover9

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)].
```

niega el objetivo.

Propiedades

- La lógica de primer orden es completa[1].
  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

Propiedades

- 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)

Propiedades

- Una cláusula de Horn es una disyunción de literales en la que hay a lo sumo un único literal positivo.
  $\neg p1 \vee \neg p2 \vee \neg p3 \vee \cdots \vee \neg pn \vee q$
- Aplicando leyes de DeMorgan, esto equivale a
  $\neg(p1 \wedge p2 \wedge p3 \wedge \cdots \wedge pn) \vee q$
- Por la definición de implicación, esto equivale a
  $(p1 \wedge p2 \wedge p3 \wedge \cdots \wedge pn) \implies q$
- 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 Resoultion for Definite Clauses).

(SWI-)Prolog

- (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.
  $q \leftarrow (p1 \wedge p2 \wedge p3 \wedge ... \wedge pn)$
- La implicación se denota por ":-", la conjunción por una coma y las cláusulas deben terminar en punto.
  $q\ :- p1, p2, p3, ..., pn.$
- Las cláusulas que no tienen cuerpo se denominan **hechos**. q.
- Las cláusulas que sí tienen cuerpo se denominan **reglas**.

Prolog

- 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], $[E|L]$)

Prolog

- 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 *is* $3 + 8$, $J =:= K$, . . .)

Índice

Definición

- La Programación Lógica Inductiva es la aplicación de técnicas de aprendizaje inductivo sobre la lógica de primer orden.
- El término en ingles 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 clausulas auxilliares que describan el conocimiento del dominio* (es, decir, los predicados que pueden ser utilizados en las cláusulas a construir).

Definición

- La Programación Lógica Inductiva es la aplicación de técnicas de aprendizaje inductivo sobre la lógica de primer orden.
- El término en ingles 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 clausulas auxilliares que describan el conocimiento del dominio* (es, decir, los predicados que pueden ser utilizados en las cláusulas a construir).

Definición

- La Programación Lógica Inductiva es la aplicación de técnicas de aprendizaje inductivo sobre la lógica de primer orden.
- El término en ingles 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 clausulas auxilliares que describan el conocimiento del dominio* (es, decir, los predicados que pueden ser utilizados en las cláusulas a construir).

Definición

- 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 ($P \nvdash e$).

- 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 ($P \vdash e$).

Definición

- 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 ($P \nvdash e$).
- 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 ($P \vdash e$).

Definición

- 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 ($P \nvdash e$).
- 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 ($P \vdash e$).

Definición

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.

Definición

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.

Definición

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.

Definición

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^-).

Definición

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.

- 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

FOIL

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

```
function  FOIL returns listaDeReglas
input  Ep // Conjunto de ejemplos positivos
input  En // Conjunto de ejemplos negativos
input  Domain // Conjunto de predicados del dominio
input  Pred // Predicado objetivo

begin
        listaDeReglas := Vacia
        while NoVacio(Ep)
                regla := ReglaVacia(Pred)
                while Cubre(regla,En)
                        regla := AnadirLiteral(regla,Domain)
                endwhile
                Ep := EliminarEjemplosCubiertos(Ep,regla)
                listaDeReglas := AnadirRegla(listaDeReglas,regla)
        endwhile
        return listaDeReglas
end
```

FOIL

- General scheme of a rule generation algorithm by specification:

```
function  FOIL returns listOfRules
input  Ep // Set of positive examples
input  En // Set of negative examples
input  Domain // Set of domain predicates
input  Pred // Target predicate

begin
        listOfRules := EmptySet
        while Not Empty(Ep)
                rule := EmptyRule(Pred)
                while Satisfy(rule,En)
                        rule := AddLiteral(rule,Domain)
                endwhile
                Ep := RemoveCoveredPositives(Ep,rule)
                listOfRules := AddRegla(listOfRules,rule)
        endwhile
        return listOfRules
end
```

FOIL

- 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

- 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 hechos[2]

```
nieta(victor, sharon).
padre(sharon, bob).
padre(tom, bob).
mujer(sharon).
padre(bob, victor).
```

[2] La 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

- FOIL parte de la regla más general:
  `nieta(X, Y) :- .`

- Para añadir términos a la regla, FOIL considera los siguientes candidatos:
  - `q(V1, V2, ..., Vr)`, donde q es alguno de los predicados del dominio y Vi 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

- FOIL parte de la regla más general:
  `nieta(X, Y) :- .`

- Para añadir términos a la regla, FOIL considera los siguientes candidatos:
  - `q(V1, V2, ..., Vr)`, donde q es alguno de los predicados del dominio y Vi 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

- FOIL parte de la regla más general:
  `nieta(X, Y) :- .`

- Para añadir términos a la regla, FOIL considera los siguientes candidatos:
  - `q(V1, V2, ..., Vr)`, donde q es alguno de los predicados del dominio y Vi 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

- FOIL parte de la regla más general:
  `nieta(X, Y) :- .`

- Para añadir términos a la regla, FOIL considera los siguientes candidatos:
  - `q(V1, V2, ..., Vr)`, donde q es alguno de los predicados del dominio y Vi 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

- FOIL parte de la regla más general:
  `nieta(X, Y) :- .`

- Para añadir términos a la regla, FOIL considera los siguientes candidatos:
  - `q(V1, V2, ..., Vr)`, donde q es alguno de los predicados del dominio y Vi 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))

```
R = {}
E = Ejemplos
mientras Ep in E :
        Regla = CrearRegla(P(X1,...,Xn),{});
        mientras cubre(Regla, En):
                Literales = GenerarLiterales(BC,Regla)
                L = Mejor(Literales)
                Regla.add(L)
        R.add(Regla)
        E = QuitarCubiertos(E,Regla)
devolver R
```

FOIL

- Siguiendo el ejemplo, los candidatos serían los siguientes:

```
mujer(X),
mujer(Y),
padre(X,Y),
padre(Y,X),
padre(X, Z),
padre(Z,X),
padre(Y,Z),
padre(Z,Y),
X = Y,
X \= Y.
```

FOIL

- 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.

FOIL

- 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.

FOIL

- 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.

FOIL

- 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.

FOIL

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'.

FOIL

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'.

FOIL

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'.

FOIL

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'.

FOIL

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'.

FOIL

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, que hacen cierta a la regla
- n_R el número de sustituciones negativas y
- t es el número de ejemplos positivos cubiertos por R que permanecen cubiertos por R'.

Nosotros
$t = p_R$

Como es una de no influye demasiado

FOIL

Ejemplo:

- Conjunto de ejemplos positivos:

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

- Conjunto de ejemplos negativos:

```
nieta(victor, victor). nieta(victor, bob). nieta(victor, tom).
nieta(bob, victor). nieta(bob, bob). nieta(bob, tom). nieta(bob, sharon).
nieta(tom, victor). nieta(tom, bob). nieta(tom, tom). nieta(tom, sharon).
nieta(sharon, victor). nieta(sharon, bob). nieta(sharon, tom). nieta(sharon, sharon).
```

- Dominio:

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

FOIL

- Estado inicial:

$$nieta(X, Y).$$

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

FOIL

- Estado inicial:

$$nieta(X, Y).$$

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

FOIL

- Estado inicial:

$$nieta(X, Y).$$

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

FOIL

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 /= 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	4	0	0
nieta(X,Y):- mujer(Y), padre(X,Y).	0	4	0	0
nieta(X,Y):- mujer(Y), padre(Y,X).	0	4	0	0
nieta(X,Y):- mujer(Y), padre(X,Z).	0	3	0	0
nieta(X,Y):- mujer(Y), padre(Z,X).	1	2	1	0.41
nieta(X,Y):- mujer(Y), padre(Y,Z).	1	3	1	0.0
nieta(X,Y):- mujer(Y), padre(Z,Y).	0	0	0	0
nieta(X,Y):- mujer(Y), X = Y.	0	4	0	0
nieta(X,Y):- mujer(Y), X /= Y.	0	0	4	0

FOIL

- Segunda iteración:

para todos $p_R = 1$ y $n_R = 3$				
Candidata	P	N	t	ganancia
nieta(X,Y):- mujer(Y), mujer(X).	0	4	0	0
nieta(X,Y):- mujer(Y), padre(X,Y).	0	4	0	0
nieta(X,Y):- mujer(Y), padre(Y,X).	0	4	0	0
nieta(X,Y):- mujer(Y), padre(X,Z).	0	3	0	0
nieta(X,Y):- mujer(Y), padre(Z,X).	1	2	1	0.41
nieta(X,Y):- mujer(Y), padre(Y,Z).	1	3	1	0.0
nieta(X,Y):- mujer(Y), padre(Z,Y).	0	0	0	0
nieta(X,Y):- mujer(Y), X = Y.	0	4	0	0
nieta(X,Y):- mujer(Y), X /= Y.	0	0	4	0

FOIL

- Segunda iteración:

para todos $p_R = 1$ y $n_R = 3$				
Candidata	P	N	t	ganancia
nieta(X,Y):- mujer(Y), mujer(X).	0	4	0	0
nieta(X,Y):- mujer(Y), padre(X,Y).	0	4	0	0
nieta(X,Y):- mujer(Y), padre(Y,X).	0	4	0	0
nieta(X,Y):- mujer(Y), padre(X,Z).	0	3	0	0
nieta(X,Y):- mujer(Y), padre(Z,X).	1	2	1	0.41
nieta(X,Y):- mujer(Y), padre(Y,Z).	1	3	1	0.0
nieta(X,Y):- mujer(Y), padre(Z,Y).	0	0	0	0
nieta(X,Y):- mujer(Y), X = Y.	0	4	0	0
nieta(X,Y):- mujer(Y), X /= Y.	0	0	4	0

FOIL

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=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

FOIL

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=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

FOIL

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=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

FOIL

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=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

FOIL

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=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

FOIL

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=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=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

FOIL

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=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:

$$\frac{\neg p \vee q \vee r \ldots \qquad p \vee s \vee t \ldots}{q \vee r \vee \ldots \vee s \vee t \vee}$$

- 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.

$$\frac{\neg p \vee q \vee \neg r \ldots \qquad p \vee \neg s \vee \neg t \ldots}{q \vee \neg r \vee \ldots \vee \neg s \vee \neg t \vee \ldots} \qquad \frac{p \wedge r \wedge \ldots \rightarrow q \qquad s \wedge t \ldots \rightarrow p}{r \wedge \ldots \wedge s \wedge t \ldots \rightarrow q}$$

- Consiste en considerar la inducción como la operación inversa de la deducción.

- Regla de resolución en lógica proposicional:

$$\frac{\neg p \vee q \vee r \ldots \qquad p \vee s \vee t \ldots}{q \vee r \vee \ldots \vee s \vee t \vee}$$

- 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.

$$\frac{\neg p \vee q \vee \neg r \ldots \qquad p \vee \neg s \vee \neg t \ldots}{q \vee \neg r \vee \ldots \vee \neg s \vee \neg t \vee \ldots} \qquad \frac{p \wedge r \wedge \ldots \rightarrow q \qquad s \wedge t \ldots \rightarrow p}{r \wedge \ldots \wedge s \wedge t \ldots \rightarrow q}$$

Resolución inversa en lógica proposicional

- Consiste en considerar la inducción como la operación inversa de la deducción.

- Regla de resolución en lógica proposicional:

$$\frac{\neg p \vee q \vee r \ldots \qquad p \vee s \vee t \ldots}{q \vee r \vee \ldots \vee s \vee t \vee}$$

- 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.

$$\frac{\neg p \vee q \vee \neg r \ldots \qquad p \vee \neg s \vee \neg t \ldots}{q \vee \neg r \vee \ldots \vee \neg s \vee \neg t \vee \ldots} \qquad \frac{p \wedge r \wedge \ldots \rightarrow q \qquad s \wedge t \ldots \rightarrow p}{r \wedge \ldots \wedge s \wedge t \ldots \rightarrow q}$$

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

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

- Si los literales iguales son negativos:
  - $C_1 = h1 \lor \neg r1 \lor \neg r2$
  - $C_2 = h2 \lor \neg q1 \lor \neg q2 \lor \neg r1 \lor \neg r2$
  - $O(C_1, C_2) = h2 \lor \neg q1 \lor \neg q2 \lor \neg h1$
- Si los literales iguales incluyen al literal positivo:
  - $C_1 = h1 \lor \neg r1 \lor \neg p1$
  - $C_2 = h1 \lor \neg r1 \lor \neg q1 \lor \neg q2$
  - $O(C_1, C_2) = p1 \lor \neg q1 \lor \neg q2$
- 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

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

- Si los literales iguales son negativos:
  - $C_1 = h1 \lor \neg r1 \lor \neg r2$
  - $C_2 = h2 \lor \neg q1 \lor \neg q2 \lor \neg r1 \lor \neg r2$
  - $O(C_1, C_2) = h2 \lor \neg q1 \lor \neg q2 \lor \neg h1$
- Si los literales iguales incluyen al literal positivo:
  - $C_1 = h1 \lor \neg r1 \lor \neg p1$
  - $C_2 = h1 \lor \neg r1 \lor \neg q1 \lor \neg q2$
  - $O(C_1, C_2) = p1 \lor \neg q1 \lor \neg q2$
- 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:
  - $C_1 = h1 \lor \neg r1 \lor \neg r2$
  - $C_2 = h2 \lor \neg q1 \lor \neg q2 \lor \neg r1 \lor \neg r2$
  - $O(C_1, C_2) = h2 \lor \neg q1 \lor \neg q2 \lor \neg h1$
- Si los literales iguales incluyen al literal positivo:
  - $C_1 = h1 \lor \neg r1 \lor \neg p1$
  - $C_2 = h1 \lor \neg r1 \lor \neg q1 \lor \neg q2$
  - $O(C_1, C_2) = p1 \lor \neg q1 \lor \neg q2$
- 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^+)

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^+)

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^+)

Resolución inversa en lógica de primer orden

- Regla de resolución en lógica de primer orden:

$$\frac{\neg p1 \vee q1 \vee r1 \ldots \qquad p2 \vee s2 \vee t2 \ldots}{(q1 \vee r1 \vee \ldots \vee s2 \vee t2 \vee \ldots.)\theta} \; \theta \equiv \textit{unificador} \text{ tal que } (p_1\theta = p_2\theta)$$

- 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.

$$\frac{\neg p1 \vee q1 \vee \neg r1 \ldots \qquad p2 \vee \neg s2 \vee \neg t2 \ldots}{(q1 \vee \neg r1 \vee \ldots \vee \neg s2 \vee \neg t2 \vee \ldots.)\theta} \; \theta \equiv \textit{unificador} \text{ tal que } (p_1\theta = p_2\theta)$$

Resolución inversa en lógica de primer orden

- Regla de resolución en lógica de primer orden:

$$\frac{\neg p1 \vee q1 \vee r1 ... \qquad p2 \vee s2 \vee t2 ...}{(q1 \vee r1 \vee ... \vee s2 \vee t2 \vee ....)\theta} \quad \theta \equiv \textit{unificador} \text{ tal que } (p_1\theta = p_2\theta)$$

- 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.

$$\frac{\neg p1 \vee q1 \vee \neg r1 ... \qquad p2 \vee \neg s2 \vee \neg t2 ...}{(q1 \vee \neg r1 \vee ... \vee \neg s2 \vee \neg t2 \vee ....)\theta} \quad \theta \equiv \textit{unificador} \text{ tal que } (p_1\theta = p_2\theta)$$

Resolución inversa en lógica de primer orden

- Regla de resolución en lógica de primer orden:

$$\frac{\neg p1 \vee q1 \vee r1 ... \qquad p2 \vee s2 \vee t2...}{(q1 \vee r1 \vee ... \vee s2 \vee t2 \vee ....)\theta} \; \theta \equiv \textit{unificador} \text{ tal que } (p_1\theta = p_2\theta)$$

- 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.

$$\frac{\neg p1 \vee q1 \vee \neg r1 ... \qquad p2 \vee \neg s2 \vee \neg t2...}{(q1 \vee \neg r1 \vee ... \vee \neg s2 \vee \neg t2 \vee ....)\theta} \; \theta \equiv \textit{unificador} \text{ tal que } (p_1\theta = p_2\theta)$$

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.
    - $C_1 = p1 \vee \neg r11 \vee \neg r12$
    - $C_2 = q1 \vee \neg q2 \vee \neg q3 \vee \neg r21 \vee \neg r22$

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.

  - $C_1 = p1 \vee \neg r11 \vee \neg r12$
  - $C_2 = q1 \vee \neg q2 \vee \neg q3 \vee \neg r21 \vee \neg r22$

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 (${r_1}_i\theta = {r_2}_i\theta$).

- La sustitución se puede descomponer en dos partes (θ_1 y θ_2), referidas a las variables de C_1 y C_2, respectivamente.

  - $C_1 = p1 \vee \neg r11 \vee \neg r12$
  - $C_2 = q1 \vee \neg q2 \vee \neg q3 \vee \neg r21 \vee \neg r22$

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.

  - $C_1 = p1 \vee \neg r11 \vee \neg r12$
  - $C_2 = q1 \vee \neg q2 \vee \neg q3 \vee \neg r21 \vee \neg r22$

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.
    - $C_1 = p1 \vee \neg r11 \vee \neg r12$
    - $C_2 = q1 \vee \neg q2 \vee \neg q3 \vee \neg r21 \vee \neg r22$

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.
    - $C_1 = p1 \vee \neg r11 \vee \neg r12$
    - $C_2 = q1 \vee \neg q2 \vee \neg q3 \vee \neg r21 \vee \neg r22$

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.
    - $C_1 = p1 \vee \neg r11 \vee \neg r12$
    - $C_2 = q1 \vee \neg q2 \vee \neg q3 \vee \neg r21 \vee \neg r22$

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 Buntime, 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 ([Sammut y Banerji, 86]) y CIGOL ([Muggleton y Buntime, 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

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

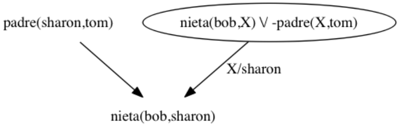
- De nuevo, pueden existir muchas formas de aplicar este operador.
- Las herramientas MARVIN ([Sammut y Banerji, 86]) y CIGOL ([Muggleton y Buntime, 88]) se basan en este algoritmo.

Ejemplo

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

- entonces $C_3 = nieta(bob, X) \vee \neg padre(X, tom)$

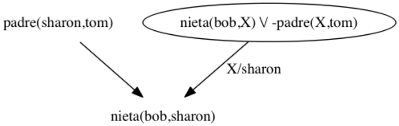
$$(nieta(bob, X) :- \; padre(X, tom))$$



Ejemplo

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

$$(nieta(bob, X) :- \ padre(X, tom))$$

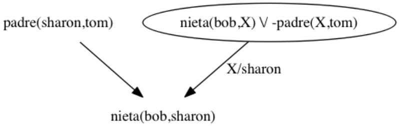


Ejemplo

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

- entonces $C_3 = nieta(bob, X) \vee \neg padre(X, tom)$

$$(nieta(bob, X) :- \; padre(X, tom))$$

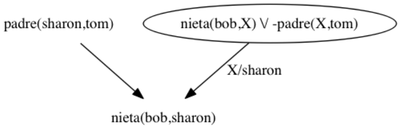


Ejemplo

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

- entonces $C_3 = nieta(bob, X) \vee \neg padre(X, tom)$

$$(nieta(bob, X) :- \; padre(X, tom))$$

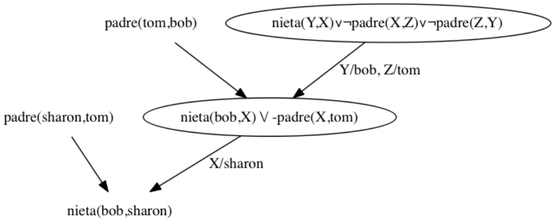


Ejemplo

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

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

$$(nieta(Y, X) : -padre(X, Z), padre(Z, Y))$$

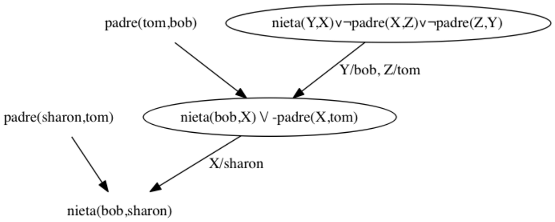


Ejemplo

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

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

$$(nieta(Y, X) : -padre(X, Z), padre(Z, Y))$$

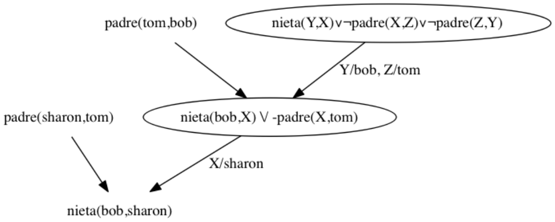


Ejemplo

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

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

$$(nieta(Y, X) : -padre(X, Z), padre(Z, Y))$$

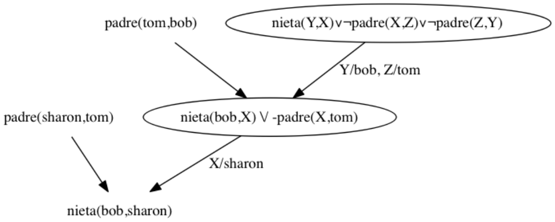


Ejemplo

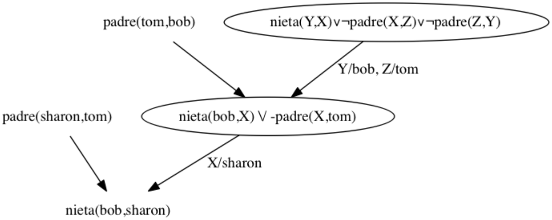
- Si $C_4 = padre(tom, bob)$
- $C_3 = nieta(bob, X) \vee \neg padre(X, tom)$
- y $\theta = \{Y/bob, Z/tom\}$
- entonces

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

$$(nieta(Y, X) : -padre(X, Z), padre(Z, Y))$$



Ejemplo



$$(nieta(Y, X) : -padre(X, Z), padre(Z, Y))$$

Reglas

Absorción:

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

Identificación:

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

Intra-construcción:

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

Inter-construcción:

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

Algoritmo

Algoritmo

¿ Alguna idea ?