

# REPRESENTACION DEL CONOCIMIENTO

## PREMISA FUNDAMENTAL DE IA:

Para que un sistema informático demuestre un comportamiento inteligente en la solución de problemas, debe poseer *gran cantidad de conocimiento* y un potente *mecanismo de razonamiento*.



IMPORTANCIA DE UNA ADECUADA  
REPRESENTACION DEL CONOCIMIENTO

- **CONOCIMIENTO DEL MUNDO (en IA)**

Es la habilidad para construir un modelo de los objetos, sus vinculaciones y de las acciones que pueden realizar.

- **REPRESENTACION DEL CONOCIMIENTO**

Es la *expresión mediante algún lenguaje*, de un modelo que exprese el conocimiento sobre el mundo.

# DISTINTOS PARADIGMAS:

- *DECLARATIVO*

*Descripción del estado del mundo*

- *PROCEDIMENTAL*

*Expresión de las transformaciones de estados*

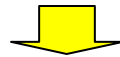
- *ORIENTADO A OBJETOS*

*Descripción de los objetos existentes*

# ELEMENTOS BASICOS QUE INTERVIENEN EN EL DISEÑO DE UN SISTEMA BASADO EN EL CONOCIMIENTO (KBS)

- ◆ Lenguaje formal para expresar conocimiento
- ◆ Forma de efectuar razonamientos

## *COMPONENTE MEDULAR DE UN KBS (Agente)*



### **BASE DE CONOCIMIENTOS (KB)**



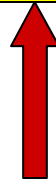
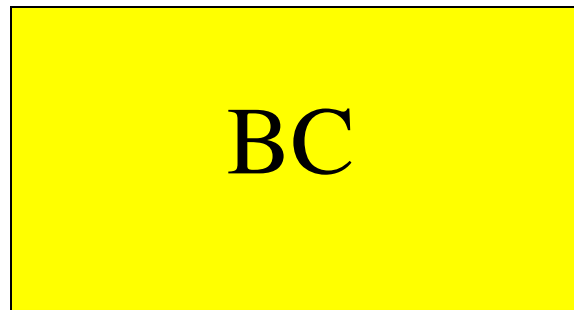
*Es un conjunto de representaciones de hechos  
acerca del mundo*



*Conjunto de sentencias del lenguaje para la  
representación del conocimiento*

UNA KB DEBE PERMITIR CON EFICIENCIA:

AÑADIR Y MODIFICAR  
SENTENCIAS



PREGUNTAS

RESPUESTAS

MECANISMO DE INFERENCIAS

# PROPIEDADES DE UN BUEN FORMALISMO DE REPRESENTACION:

- ADECUACION REPRESENTACIONAL
- ADECUACION INFERENCIAL
- EFICIENCIA INFERENCIAL
- EFICIENCIA EN LA ADQUISICION-MODIFICACION

Rich & Knight

# DISTINTOS FORMALISMOS

❖ FORMALISMOS LOGICOS

❖ SISTEMAS DE PRODUCCION

❖ FORMALISMOS ESTRUCTURADOS:

- *REDES SEMANTICAS*
- *FRAMES*
- *OBJETOS*



◆ *LENGUAJE DE REPRESENTACION*

↗ SINTAXIS: posibles sentencias del lenguaje

↘ SEMANTICA: conexión entre sentencias y el mundo

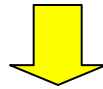
◆ *MECANISMO DE INFERENCIA*

- *Generar* nuevas sentencias que derivan de BC
- Dada una sentencia, puedo *contestar si es consecuencia* de la BC

# FORMALISMOS LOGICOS

Constituyen sistemas formales en los cuales:

- ♦ SINTAXIS Y SEMANTICA ESTA BIEN DEFINIDA
- ♦ HAY UNA TEORIA DE LA DEMOSTRACION
  - Completa y Consistente



## LA LOGICA DE 1<sup>er</sup> ORDEN

Es la base de la mayoría de los esquemas de representación

# FORMALISMOS LOGICOS

- Conocimiento es representado mediante un conjunto de fórmulas bien formadas (fbfs) en algún sistema lógico (proposicional - predicados - multivaluada...)
- Los mecanismos de inferencia son los métodos deductivos del sistema lógico (Resolución en predicados)

## DISTINTOS SISTEMAS LOGICOS:

- ◆ LOGICA PROPOSICIONAL
- ◆ LOGICA DE PREDICADOS
- ◆ LOGICAS NO-CLASICAS
  - MULTIVALUADAS (Fuzzy Logic)
  - MODALES

***OBJETIVO: ESTABLECER LA VALIDEZ DE  
DISTINTOS RAZONAMIENTOS -  
OBTENER CONCLUSIONES DE UN CONJUNTO  
DE FORMULAS***

REPRESENTACION DEL  
CONOCIMIENTO

FORMALISMOS LOGICOS

# Lógica proposicional

# LOGICA PROPOSICIONAL

## ♦ LENGUAJE

- Sintaxis: fbfs
- Semántica: asignación de valores a las variables

## ♦ SISTEMA FORMAL

- Lenguaje
- Axiomas
- Reglas de inferencia

- *COMPLETO Y CONSISTENTE*

- *EL PROCESO DE DEMOSTRACION NO ES EFECTIVO*

# Introducción Informal

- **Proposición:** Una oración afirmativa de la cual podemos decir que es **verdadera** o **falsa** (pero no ambas!!)
- Ejemplos de Proposiciones:
  - Ayer llovió en Rosario.
  - El sol gira alrededor de la tierra.
  - $2 \cdot 3 = 3 + 3$
  - 3 es primo.
  - El sucesor de 3 es primo.



## más proposiciones...

- Si ayer llovió en Rosario, entonces el intendente se mojó.
- El sol gira alrededor de la tierra o la tierra gira alrededor del sol.
- $2 \cdot 3 = 6$  y 6 es impar
- 3 no es primo.
- Hay un número natural que es par y es primo.
- Todo entero par mayor que cuatro es la suma de dos números primos.

# ejemplos de oraciones que no son proposiciones...

- ¿Ayer llovió en Rosario?
- ¿Por qué es importante saber si el sol gira alrededor de la tierra?
- Parece que no hay primos que sean pares.
- Averigüen si la tierra gira alrededor del sol o si el sol gira alrededor de la tierra.
- $2 \cdot n = n + n$
- $x - y = y - x$

# Sintaxis

## Alfabeto PROPOSICIONAL

$\Sigma_{\text{PROP}}$  que consiste de:

- i) variables proposicionales  $p_0, p_1, p_2, \dots$
- ii) conectivos  $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$
- iii) símbolos auxiliares:  $(, )$

Notación : llamaremos  $\mathbf{C}$  al conjunto  $\{\wedge, \vee, \rightarrow, \leftrightarrow\}$

# Sintaxis

## Fórmulas proposicionales PROP

PROP es el conjunto definido **inductivamente** por :

- i)  $p_i \in \text{PROP}$  para todo  $i \in \mathbb{N}$  (fórmulas atómicas - AT)
- ii) Si  $\alpha \in \text{PROP}$  y  $\beta \in \text{PROP}$  entonces
  - $(\alpha \wedge \beta) \in \text{PROP}$
  - $(\alpha \vee \beta) \in \text{PROP}$
  - $(\alpha \rightarrow \beta) \in \text{PROP}$
  - $(\alpha \leftrightarrow \beta) \in \text{PROP}$
- iii) Si  $\alpha \in \text{PROP}$  entonces  $(\neg \alpha) \in \text{PROP}$

# PROP (cont.)

- Ejemplos de objetos de PROP:
  - $p_0$
  - $(p_1 \rightarrow p_3)$
  - $((p_1 \rightarrow p_2) \vee (p_3 \wedge (\neg p_5)))$

# Traducción al lenguaje lógico

- Las oraciones simples se traducen como letras de proposición (elementos de P)

## – Ejemplos:

- Ayer llovió en Rosario  $\rightarrow p_0$ .
- El intendente se mojó  $\rightarrow p_1$ .
- El sol gira alrededor de la tierra  $\rightarrow p_2$ .
- $2 \cdot 3 = 6 \rightarrow p_3$
- 6 es impar  $\rightarrow p_4$ .
- El sucesor de 3 es primo  $\rightarrow p_5$ .

# Traducción al lenguaje Lógico

- Las oraciones compuestas se traducen usando los conectivos

## – Ejemplos:

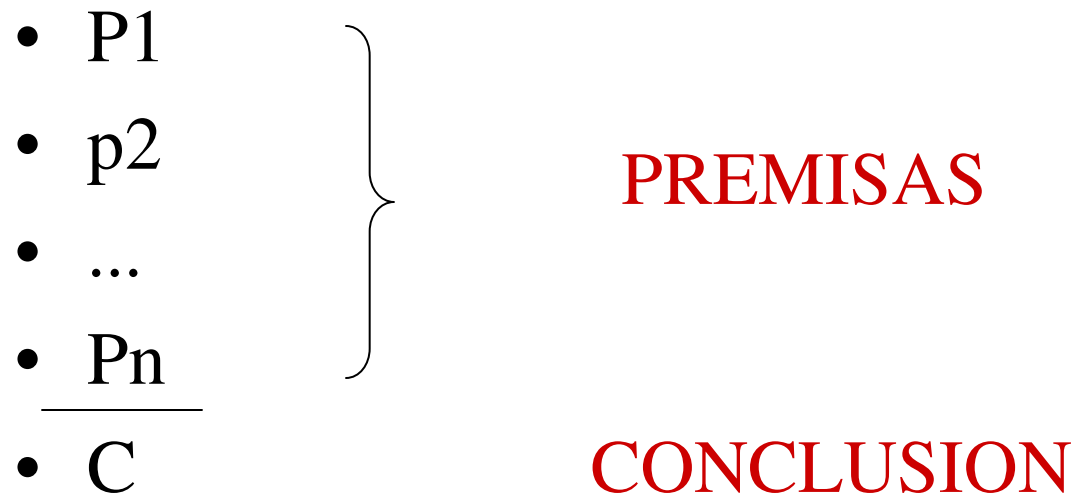
- Si ayer llovió en Rosario, entonces el intendente se mojó  $\rightarrow (p_0 \rightarrow p_1)$ .
- $2 \cdot 3 = 6$  y 6 es impar  $\rightarrow (p_3 \wedge p_4)$ .
- 6 no es impar  $\rightarrow (\neg p_4)$ .

# Traducción al lenguaje Lógico

- Algunas oraciones no tienen una buena traducción a PROP:
  - Hay aves que no vuelan.  $p_0$
  - Todo entero par mayor que cuatro es la suma de dos números primos.  $p_1$



# Razonamientos



## *EJEMPLO*

- Rex es un perro
- Si Rex es un perro entonces tiene cuatro patas /  $\therefore$  Rex tiene 4 patas.

# Razonamiento

Si continúa la lluvia el río aumentará.

Si el río aumenta entonces el puente será arrastrado.

Si la continuación de la lluvia hace que el puente sea arrastrado entonces un solo camino no será suficiente para la ciudad.

O bien un solo camino es suficiente para la ciudad, o los ingenieros han cometido un error.

Por lo tanto los Ingenieros han cometido un error.

**ES VALIDO ????**

# Justificación de la validez del razonamiento?

Dos maneras diferentes de justificar

- Justificar que la veracidad de las hipótesis implica la veracidad de la conclusión

(Justificación semántica  $\Gamma \models \beta$ )

- Dar una prueba matemática, que llegue a la conclusión a partir de las hipótesis, a través de pasos debidamente justificados.

(Justificación sintáctica  $\Gamma \vdash \beta$ )

# Justificación Semántica

- Consiste en verificar que la fórmula de PROLOG que codifica el razonamiento es una **tautología**

$$\models \{ p_1 \wedge p_2 \wedge p_3 \dots \wedge p_n \rightarrow C \}$$

## EJEMPLO DE PROLOG

$$\models \{ ((R_p \rightarrow 4p) \wedge R_p) \rightarrow 4p \}$$

# Justificación Sintáctica

Dar una **prueba matemática**, que:

- llegue a la conclusión a partir de las hipótesis,
- esté constituida de pasos debidamente justificados

p1  
p2  
Pn

PREMISAS

d1  
dr

CONCLUSIONES

INTERMEDIAS

C

CONCLUSION

# Reglas de Inferencia

- ✓ Pertenecen a las especificaciones del Sistema Lógico Formal, o sea al **Metalinguaje**.
- ✓ Son reglas sintácticas que **me permiten deducir** a partir de ciertas formas proposicionales, otras formas proposicionales.
- ✓ La **prueba** consiste en un encadenamiento de pasos de reglas de inferencia que nos permite llegar a la conclusión.

## EJEMPLOS DE REGLAS:

- MODUS PONENS:  $A \rightarrow B, A / \therefore B$
- MODUS TOLLENS:  $A \rightarrow B, \neg B / \therefore \neg A$
- SILOGISMO DISYUNTIVO:  $A \vee B, \neg A / \therefore B$

# Razonamiento (ejemplo)

1-  $C \rightarrow R$

2-  $R \rightarrow P$

3-  $(C \rightarrow P) \rightarrow \neg S$

4-  $S \vee E / \therefore E$

5-  $C \rightarrow P$

1y2 por S.H.

6-  $\neg S$

3y5 por M.P.

7-  $E$

4y6 por S.D.

**LUEGO EL RAZONAMIENTO ES  
VALIDO**

# Del conjunto de hipótesis $\Gamma$ se deduce $\alpha$ ?

$\Gamma \models \alpha$  ?

- Tablas de verdad
- equivalencia lógicas
- existe un método que siempre responde SI o NO

$\Gamma \vdash \alpha$  ?

- Prueba formal
- requiere ingenio

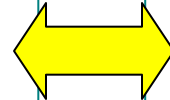
**Estas dos formas de responder la pregunta inicial son equivalentes?**



# Teorema de completitud

$$\Gamma \models \alpha$$

- Tablas de verdad
  - equivalencia lógicas
- existe un método que  
siempre responde  
SI o NO



$$\Gamma \vdash \alpha$$

- Prueba formal
- requiere ingenio

✓ El teorema nos autoriza a combinar ambas técnicas y utilizar equivalencias semánticas y pruebas.  
(que es lo que usualmente hacemos en matemáticas)

# Lógica de predicados

# LOGICA PREDICADOS (1er orden)

## ♦ LENGUAJE

- Sintaxis: fórmulas bien formadas (FORM)
- Semántica: Interpretación - valoración

## ♦ SISTEMA FORMAL

- Lenguaje
- Axiomas
- Reglas de inferencia (se agregan a las para manejar cuantificadores)

- *COMPLETO Y CONSISTENTE*
- *EL PROCESO DE DEMOSTRACION NO ES EFECTIVO*

Todo perro es un mamífero y **Rex es un perro,**  
**luego Rex es un mamífero..**

$$\frac{\forall x (Perro(x) \rightarrow Mamífero(x)) \quad Perro(Rex)}{Mamífero(Rex)} \quad \frac{\forall x. P(x)}{P(Rex)}$$

- La corrección de este razonamiento depende de la relación entre los sujetos de las proposiciones.
- **Lógica proposicional NO es suficientemente expresiva para captar esta relación**

# Por qué lógica de predicados ?

- Lógica proposicional : bajo poder expresivo
- Muchas expresiones usuales no son representables

- « Rex es un perro »

En proposicional:

**p** (una prop. **atómica**)

En predicados:

**Sujeto:** Rex

**Propiedad:** Ser Perro

**Perro(Rex)**

# Como Traducir ???

Por ejemplo la oración

Rex es un perro

puede analizarse de una de las siguientes maneras:

- Es (Rex, perro)
- Es-perro (Rex)
- Es-Rex (Perro)

según la propiedad o relación que se identifique, y según los individuos del universo de quienes se hable.

# Lenguaje de lógica de predicados

- *símbolos para denotar **objetos***
  - **sb. de constante** (ej.  $Rex$ ,  $2$ ,  $\pi$ )
  - **sb. de variable** (ej.  $x$ ,  $y$ ,  $z$ )
  - **sb. de función** (ej.  $+$ ,  $*$ ,  $Padre$ ) etc que permiten crear nuevos nombres de objetos
- *símbolos de **propiedades y de relaciones***
- *conectivos*
- *cuantificadores*

# Ejemplos de traducción

- Si algunos perros son mamíferos, luego todos son mamíferos

$$(\exists x) (P(x) \wedge M(x)) \rightarrow \forall x (P(x) \rightarrow R(x))$$

- Todo número es par o impar

$$(\forall x) (N(x) \rightarrow P(x) \vee I(x))$$

$$(\forall x) (N(x) \rightarrow P(x) \vee \neg P(x))$$

- Ningún número es a la vez par e impar

$$\neg(\exists x) (P(x) \wedge I(x))$$



# Ejemplos de traducción

- Toda ave tiene alas y plumas

$$(\forall x) (Av(x) \rightarrow Al(x) \wedge Pl(x))$$

Existen aves que no vuelan

$$(\exists x) (Av(x) \wedge \neg V(x))$$

- Para todo número natural hay otro natural que es mayor que el.

$$(\forall x) (N(x) \rightarrow (\exists y) (N(y) \wedge y > x))$$

***Cuidado con el orden de los cuantificadores !!!***

$$(\exists y) (\forall x) (N(x) \rightarrow (N(y) \wedge y > x))$$

# Universo de discurso

- Si algunos trenes se retrasan entonces todos se retrasan

*y sólo hablamos de trenes*

$$(\exists x) R(x) \rightarrow (\forall x) R(x)$$

- Todo número es par o impar

*y sólo hablamos de naturales*

$$(\forall x) (P(x) \vee I(x))$$

# Alfabeto de un lenguaje de primer orden

Un **alfabeto** para un lenguaje de primer orden, consiste de los siguientes símbolos:

- Símbolos de **relación**:  $P_1, P_2, \dots, P_n, =$
- Símbolos de **función**:  $f_1, f_2, \dots, f_m$
- Símbolos de **constantes**:  $\underline{c}_i$  tal que  $i \in I$  y  $|I| = k$
- Variables:  $x_1, x_2, x_3, \dots$
- Conectivos :  $\rightarrow, \leftrightarrow, \neg, \wedge, \vee$
- Cuantificadores:  $\forall, \exists$
- Auxiliares :  $(, )$

# Términos

El conjunto **TERM** de los *términos de un lenguaje de primer orden* se define inductivamente por:

- i)  $x_i \in \text{TERM}$  ( $i \in \mathbb{N}$ )
- ii)  $\underline{c}_i \in \text{TERM}$  ( $i \in I$ )
- iii) si  $t_1 \in \text{TERM}, \dots, t_{a_i} \in \text{TERM}$   
entonces  $f_i(t_1, \dots, t_{a_i}) \in \text{TERM}$

✓ *Los términos son las expresiones que representarán a los objetos de mi dominio*

# Fórmulas bien formadas (FORM)

El conjunto **FORM** de las *fórmulas de un lenguaje de primer orden* se define inductivamente por:

i) Si  $t_1 \in \mathbf{TERM}$ , ...  $t_{r_i} \in \mathbf{TERM}$  entonces

$$P_j(t_1, \dots, t_{r_i}) \in \mathbf{FORM}$$

ii) Si  $\alpha \in \mathbf{FORM}$  y  $\beta \in \mathbf{FORM}$  entonces

$$(\alpha \in \beta) \in \mathbf{FORM} \text{ donde } \in \in \{\rightarrow, \leftrightarrow, \wedge, \vee\}$$

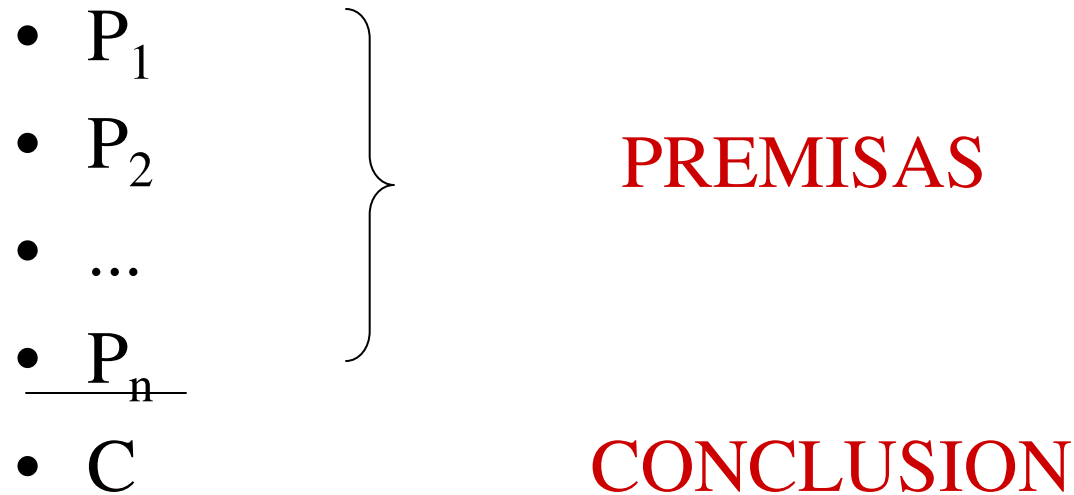
iii) Si  $\alpha \in \mathbf{FORM}$  entonces  $(\neg \alpha) \in \mathbf{FORM}$

iv) Si  $\alpha \in \mathbf{FORM}$  entonces  $((\forall x_i) \alpha) \in \mathbf{FORM}$   
y  $((\exists x_i) \alpha) \in \mathbf{FORM}$

## Ejemplos (menos formales)

- $\text{Padre}(x, y) \rightarrow \text{Hijo}(y, x)$
- $\text{Padre}(x, y) \wedge \text{Padre}(y, z) \rightarrow \text{Abuelo}(x, z)$
- $(\exists x) ( \text{Mamífero}(x) \wedge \text{Pelos}(x) )$

# Razonamientos en Lógica de 1er orden



\* *LAS PREMISAS Y LA CONCLUSION PERTENECEN A FORM*

# Justificación de la validez del razonamiento?

Una **sola** manera de justificar

- Dar una prueba matemática, que llegue a la conclusión a partir de las hipótesis, a través de pasos debidamente justificados.

(Justificación sintáctica  $\Gamma \vdash \beta$  )

(No existe justificación semántica - no siempre tienen sentido las tablas de verdad)



# Reglas de Inferencia

**Reglas de Inferencia del cálculo proposicional**

**+**

**Reglas específicas para el manejo de los cuantificadores**

- Ejemplificación universal (EU)
- Generalización universal (GU)
- Ejemplificación existencial (EE)
- Generalización existencial (GU) ...

# Razonamientos en Lógica de 1er orden

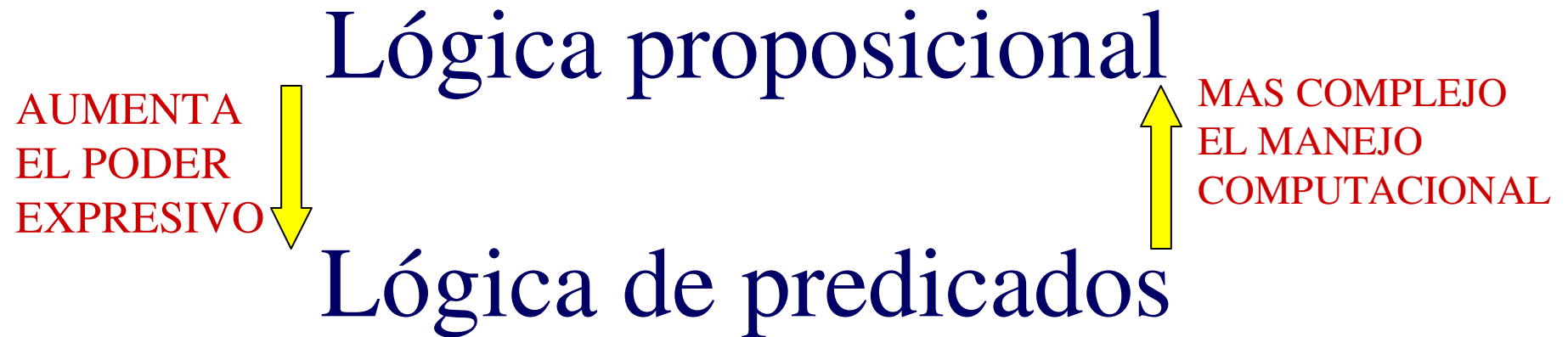
- Todos los Ovejeros Alemanes son perros y todos los perros son mamíferos. Luego, todos los Ovejeros Alemanes son mamíferos.

$$(\forall x) (Oa(x) \rightarrow P(x))$$

$$(\forall x) (P(x) \rightarrow M(x)) / \therefore (\forall x) Oa(x) \rightarrow M(x)$$

- Todos los perros caminan al menos que alguno esté lastimado. Algunos perros no caminan. Luego, hay algún perro lastimado.

Es válido???



## **PROBLEMAS PARA AUTOMATIZACION:**

- Que regla de inferencia aplicar
- A que fórmulas aplicarlas

# **Demostración por Resolución**

## **(Robinson 1965)**

- **SOLUCIONA:**
  - Selección de las RI
  - Generación de algunas proposiciones sin interés
- **OPERA CON SENTENCIAS EN LA FORMA CLAUSAL**
  - Forma genérica:
$$A_1 \vee A_2 \vee \dots A_k \vee \neg A_j \vee \dots \vee \neg A_n$$
donde  $A_i$  es una fórmula atómica

Algoritmo: fbf  conjunto de cláusulas

**Llevar a forma normal prenex**

$(Q_1x_1) \dots (Q_nx_n) \quad (M)$   
Prefijo de cuantificadores      Matriz libre de cuantificadores

- Expresar la fórmula utilizando los conectivos  $\neg$ ,  $\wedge$  y  $\vee$
- Trabajar la fórmula de modo que el  $\neg$  este delante de fórmulas atómicas
- Normalizar variables
- Llevar cuantificadores adelante

# Algoritmo: fbf conjunto de cláusulas

- A partir de la forma Prenex (cuantificadores adelante).
- Eliminar cuantificadores Existenciales  
(utilizando constantes / funciones de Skolem)
  - $(\exists y)$  presidente (y)    P: cte de Skolem  
    presidente (P)
  - $(\forall x) (\exists y)$  padre (y,x)    P2: función padre  
     $(\forall x)$  padre (P2(x), x)    (función de Skolem)

# Algoritmo: fbf conjunto de cláusulas

- Eliminar cuantificadores Universales.

- Llevarlo a una forma normal conjuntiva

$$(A_1 \vee A_2 \vee \dots A_k) \wedge \dots \wedge (A_1 \vee \neg A_2 \vee \dots \neg A_k)$$

cláusula

cláusula

$$\left\{ \begin{array}{l} (A_1 \vee A_2 \vee \dots A_k) \\ \dots \\ (A_1 \vee \neg A_2 \vee \dots \neg A_k) \end{array} \right.$$

- Normalizar las variables de las distintas cláusulas.

Forma clausal

# Paso a forma clausal (ejemplo)

- $(\forall x) (\text{usuario-comp}(x) \rightarrow ((\exists y) \text{clave}(y) \wedge \text{posee}(x,y)))$
- $(\forall x) (\neg \text{usuario-comp}(x) \vee ((\exists y) \text{clave}(y) \wedge \text{posee}(x,y)))$
- $(\forall x) (\exists y) (\neg \text{usuario-comp}(x) \vee (\text{clave}(y) \wedge \text{posee}(x,y)))$     **forma normal Prenex**
- $(\forall x) (\neg \text{usuario-comp}(x) \vee (\text{clave}(P(x)) \wedge \text{posee}(x, P(x))))$



## Paso a forma clausal (cont.)

- $(\forall x) (\neg \text{usuario-comp}(x) \vee (\text{clave}(P(x)) \wedge \text{posee}(x, P(x))))$
- $(\neg \text{usuario-comp}(x) \vee (\text{clave}(P(x)) \wedge \text{posee}(x, P(x))))$
- $(\neg \text{usuario-comp}(x) \vee \text{clave}(P(x)) \wedge (\neg \text{usuario-comp}(x) \vee \text{posee}(x, P(x))))$

**Cláusulas**

$(\neg \text{usuario-comp}(x) \vee \text{clave}(P(x)))$

$(\neg \text{usuario-comp}(x) \vee \text{posee}(x, P(x)))$

# Paso a forma clausal

## Otro Ejemplo (Rich)

*Todo romano que conoce a Marco o bien odia a César o bien piensa que cualquiera que odie a otro está loco.*

- $(\forall x) (((\text{romano}(x) \wedge \text{conoce}(x, \text{Marco})) \rightarrow (\text{odia}(x, \text{Cesar}) \vee (\forall y)(\exists z)(\text{odia}(y, z) \rightarrow \text{cree\_loco}(x, y))))$

**Forma Cláusul ???**

# Resolución

- Trabaja con razonamientos en **forma clausal**
- **Opera por refutación**
  - Agrego  $\neg C$  al conjunto de las premisas en forma clausal y trato de llegar a la cláusula vacía  $\emptyset$  (contradicción:  $A \wedge \neg A$ ).
- Es un proceso iterativo simple en el cual se utiliza **una única Regla de Inferencia**
  - resolución  $A \vee B, \neg A \vee C \quad / \quad B \vee C$

# Algoritmo: Resolución de proposiciones P I- C

- Convertir todas las proposiciones de P a **forma clausal**
- Negar C y añadir al conjunto de cláusulas
- Hasta que se encuentre una contradicción o no se pueda seguir avanzando repetir:
  - Seleccionar dos cláusulas (padres)
  - Resolverlas ( $A \vee B, \neg A \vee C$  /  $B \vee C$ , resolvente)
  - Si la resolvente es  $\emptyset$ , se ha encontrado una contradicción, si no lo es, agregarla al conjunto de cláusulas.

# Resolución en Proposiciones

## Razonamiento

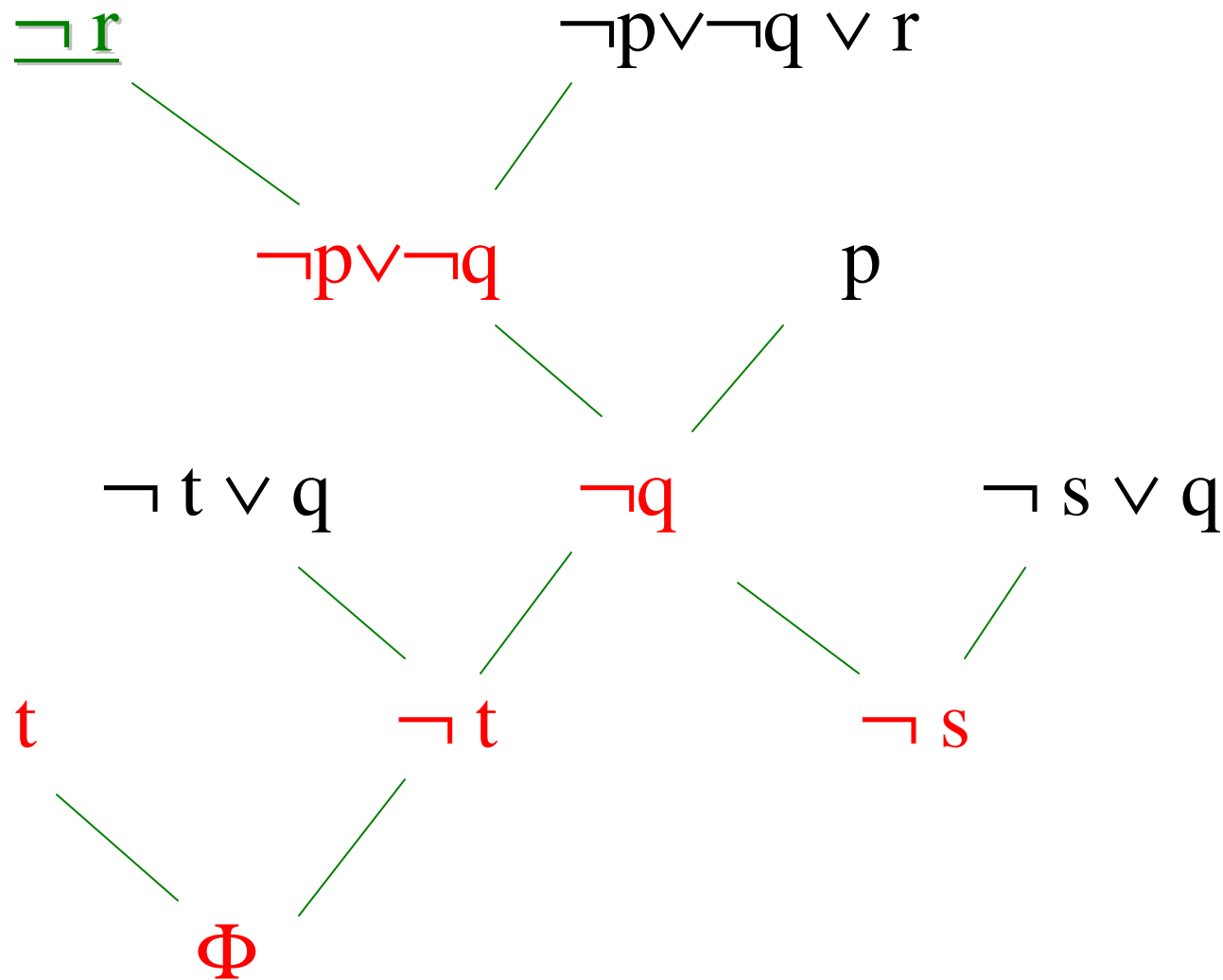
- $p$
- $(p \wedge q) \rightarrow r$
- $(s \vee t) \rightarrow q$
- $t / \therefore r$

## Prueba por resolución

## Forma cláusal

$p$   
 $\neg p \vee \neg q \vee r$   
 $\neg s \vee q$   
 $\neg t \vee q$   
 $t$   
.....  
 $\neg r$   
 $\neg p \vee \neg q$   
 $\neg q$   
 $\neg t$   
 $\Phi$

# Resolución en Proposiciones (ejemplo)



# Resolución

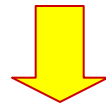
## Observaciones

- Si existe una contradicción se la encontrará en algún momento
- La conclusión negada debe estar involucrada en la contradicción que estamos buscando (si no el conjunto de premisas ya era inconsistente)
- Si no existe contradicción, puede que el proceso nunca termine

# Resolución en Predicados

- Las bases del Método son las mismas que para proposiciones

- Situación más compleja



Para resolver dos cláusulas debo encontrar  
sustitución adecuada de variables



**ALGORITMO DE UNIFICACION**



# Algoritmo de Unificación

*Idea:* ver si existe una sustitución que haga concordar a dos fórmulas

*Ejemplos:*

$\left. \begin{array}{l} \text{ama}(x, y) \\ \text{ama}(\text{Marco}, z) \end{array} \right\}$

Sustituciones que unifican  
(Marco/x, Paula/y, Paula/z)  
(Marco/x, z/y)



ES MAS GENERAL

• ***SE BUSCA ENCONTRAR LAS MINIMAS SUSTITUCIONES QUE UNIFIQUEN***

# Algoritmo de Unificación (idea)

*1- Ver si los predicados coinciden, si no falla*

*2- Comprobar si los argumentos de a pares son unificables, devolver sustitución, si alguno no lo es, falla la unificación. Proceso recursivo:*

- las ctes unifican si son iguales, sino falla
- una variable  $x$  unifica con:
  - otra variable  $S: [y/x]$
  - una cte  $k$   $S: [k/x]$
  - una función que no tenga ninguna instancia de la variable  $S: [f(y)/x]$

*Devuelve  $S_k \dots S_1$  o falla*

# Algoritmo: Resolución en Predicados

- Convertir todas las fórmulas a **forma cláusal**.
- **Negar C y agregar** al conjunto de cláusulas.
- Hasta que se encuentre una contradicción o se realizó cantidad de esfuerzo predeterminado:
  - *Seleccionar dos cláusulas padres*
  - *Resolverlas* ( $A1 \vee B, \neg A2 \vee C$  , donde A1 y A2 son unificables mediante [S ], la resolvente será  $(B \vee C) [S]$  , resolvente)
- Si la resolvente es  $\emptyset$ , se ha encontrado una contradicción, si no lo es, agregarla al conjunto de cláusulas.

# Resolución en Predicados (ejemplo)

- Razonamiento

$(\forall x) (Perro(x) \rightarrow Mamífero(x))$

$Perro(Rex) / \therefore$

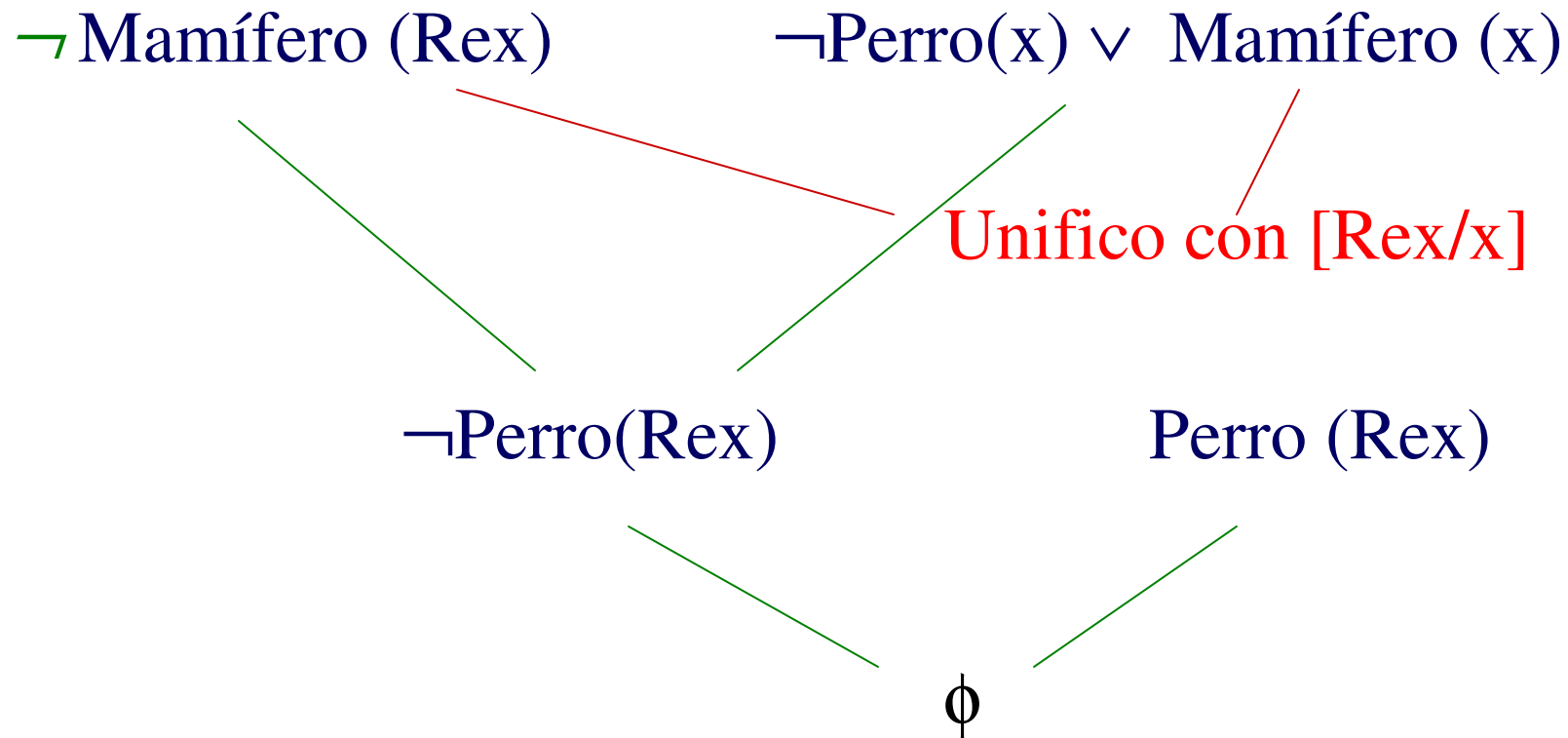
$Mamífero(Rex)$

- Forma clausal

$\neg Perro(x) \vee Mamífero(x)$

$Perro(Rex) / \therefore Mamífero(Rex)$

# Resolución en Predicados (ejemplo)



- *Cuando unifico debo aplicar la sustitución a TODA la cláusula*

# Algoritmo: Resolución en Predicados

## Observaciones:

- Si la selección de padres se hace de forma sistémica, siguiendo ciertas reglas, el procedimiento encontrará la contradicción, si esta existe.
- Hay estrategias de selección para mejorar la complejidad temporal

# Completitud de la Resolución

\* Es completa en cuanto a la refutación ↓

\* Si un conjunto de sentencias no se puede satisfacer, mediante la resolución se obtendrá una contradicción.

# Completitud – Conceptos para la demostración

## \* Universo de Herbrand $H_s$ :

- \* Es el conjunto de todos los términos de base que se pueden generar a partir de las constantes de S y de los símbolos de funciones (si hay).

$$S: P(x, f(x)) \wedge Q(x, A) \rightarrow R(x, B)$$

$$H_s = \{ A, B, f(A), f(B), f(f(A)), f(f(B)), \dots \}$$

## \* Saturación:

- \* Si S es un conjunto de cláusulas y P es un conjunto de términos de base  $P(S)$  es el conjunto de todas las cláusulas que se obtienen con todas las sustituciones de las variables por los términos de base de P.

## \* Base de Herbrand $H_s(S)$

- \* Es la saturación de un conjunto de cláusulas S respecto a su universo de Herbrand.

$$H_s(S) = \{ P(A, f(A)) \wedge Q(A, A) \rightarrow R(A, B),$$

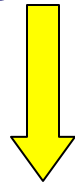
$$P(B, f(B)) \wedge Q(B, A) \rightarrow R(B, B),$$

$$P(f(A), f(f(A)) \wedge Q(f(A), A) \rightarrow R(f(A), B), \dots \} \quad \#H_s(S) = \infty$$



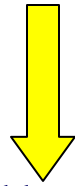
# Completitud - Estructura de la demostración (Robinson)

- \* Dado un conjunto  $S$  en forma clausal que no es satisfactible



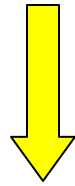
TEOREMA DE HERBRAND

- \* Algún  $S'$  de casos específicos de base no es satisfactible



TEOREMA DE RESOLUCION  
DE BASE

- \* La resolución puede llegar a una contradicción en  $S'$



PREMISA DE TRANSFERENCIA

- \* Hay una demostración de resolución de la contradicción de  $S$

# Para Trabajar: ejemplo N&R

- \* **Juan tiene un perro.**
- \* **Todos los que tienen un perro aman a los animales.**
- \* **Nadie que ame a los animales los mata.**
- \* **Juan o Curiosidad mató al gato, que se llama Tuna.**
- \* **Mató Curiosidad al gato???**

Escribir en lenguaje lógico

Pasar a forma clausal

Usar resolución para probarlo

# Para Trabajar:

- \* Frodo era un Hobbit.**
- \* Sam era un Hobbit.**
- \* Todos los Hobbits vivían en la Comarca**
- \* Todos los que vivían en la Comarca vivían en la Tierra Media.**
- \* Todos los que vivían en la tierra Media eran leales a Sauron o lo odiaban.**
- \* Todos los seres son leales a alguien.**
- \* Uno sólo intenta destruir a alguien a quien no es leal.**
- \* Frodo intentó destruir a Saurón.**

# Para Trabajar:

Escribir en lenguaje lógico

Pasar a forma clausal

Usar resolución para probar


➤ **Odia Frodo a Sauron???**

➤ **Alguien que vive en la Comarca odia a Sauron???**

# Resolución

- \* Nos acercamos a la automatización del cálculo de predicados.
- \* **Problema:** falta una estructura de control adecuada que me indique que cláusulas deben resolverse.

# **PROLOG:** Una implementación de programación lógica

- Utiliza un proceso de control para decidir que par de cláusulas deben resolverse.
- Reduce el poder expresivo de la lógica de 1<sup>er</sup> orden:
  - Cláusulas  Cláusulas de Horn:  
tienen a lo sumo 1 literal positivo
    - $A_1 \vee \neg A_2 \vee \dots \vee \neg A_n$
    - o su forma equivalente:  $A_1 \leftarrow (A_2 \wedge \dots \wedge A_n)$
    - en Prolog:  $A_1 :- (A_2, \dots, A_n)$

# CONTROL EN PROLOG

Se aplica el Principio de Resolución:

- Se lo implementa como búsqueda en un árbol y/o.
- Estrategia de control:
  - Búsqueda en profundidad, de izquierda a derecha y con backtracking.

# CONTROL EN PROLOG

- ✓ Es una implementación particular de la lógica automatizada.
- ✓ Modelo estandar: única estrategia de control
  - Búsqueda backward, en profundidad y con backtrack
  - No es muy eficiente para implementar otras estrategias de control (búsqueda a lo ancho, forward)



# LOGICA DE PREDICADOS + RESOLUCION

- Dada la BC y una fórmula  $\alpha$  podemos probar que

»  $BC \mid - \alpha$

**Podemos contestar**  
**preguntas como**  $\left\{ \begin{array}{l} \text{-perro (Rex) ?} \\ \text{- X / perro (X) ?} \end{array} \right.$

- Pero **no podemos** obtener todas las conclusiones ( $\beta$ ) que se derivan de una base

»  $\beta ? / BC \mid - \beta$

# LOGICA DE PREDICADOS COMO FORMALISMO DE REPRESENTACION

## ❖ *VENTAJAS:*

- ✓ Es un formalismo bien establecido con una sintaxis y semántica bien definida y que maneja fácilmente aspectos cuantificación.
- ✓ Se establece un sistema de inferencias completo (se puede extender al método de resolución).

## ❖ *LIMITACIONES:*

- ✓ Existen límites en el poder expresivo:
  - posibilidades, incertidumbre,
- ✓ Problemas en la implementación de razonamientos no-monotonos.