# Lógica de predicados

## Razonamientos

# LOGICA - INTRODUCCION

- **OBJETIVO**

  Uno de los fundamentales objetivos ha sido el estudio de las DEDUCCIONES, RAZONAMIENTOS O ARGUMENTOS

  ⇓

**RAZONAMIENTOS EN LA LOGICA DE 1er ORDEN**

# Razonamiento

Todos los hombres son mortales,

Sócrates es hombre,

luego Sócrates es mortal

$$\frac{\begin{array}{l}(\forall x)\ (H(x)\ M(x)) \\ H(socrates)\end{array}}{M(socrates)}$$

**ES VALIDO, COMO PROBARLO ????**

**Definición:** **Razonamiento válido**

Un razonamiento es válido si la conclusión es consecuencia semántica de las premisas

$\Gamma = \{\varphi_1, \varphi_2, \ldots, \varphi_n\}$: $\Gamma \models C$

sii para toda interpretación I:

si $I \models \varphi_i \ \forall \varphi_i \in \Gamma$, entonces $I \models C$

( sii $\models \varphi_1 \wedge \varphi_2 \wedge \ldots \wedge \varphi_n \rightarrow C$)

# Justificación de la validez del razonamiento?

– Probar la consecuencia semántica ($\Gamma \models C$)

Para lo cual habría que probar que

$\models \varphi_1 \wedge \varphi_2 \wedge \ldots \wedge \varphi_n \rightarrow C$

No hay un método semántico similar a las tablas de verdad !

– 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 de la validez del razonamiento

## Justificación sintáctica

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

**Esta justificación coincide con la definición**

**semántica de validez ???**

# Razonamientos con predicados de 1er orden

- Nos restringiremos a predicados monádicos (de un argumento) por ser más simples de tratar.

Luego utilizaremos una restricción de FORM: $FORM_1$

# Justificación Sintáctica

Dar una prueba matemática, que:

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

| | |
|---|---|
| p1 | |
| p2 | PREMISAS |
| Pn | |
| d1 | CONCLUSIONES |
| dr | INTERMEDIAS |
| C | CONCLUSION |

# Sistema formal

**Para especificar un sistema formal se requiere:**

1. Un alfabeto de símbolos.
2. Un conjunto de cadenas finitas de dichos símbolos, llamadas fórmulas bien formadas (fbf). ***(lenguaje formal)***
3. Un conjunto de fórmulas bien formadas, llamadas ***axiomas.***
4. Un conjunto finito de ***«reglas de inferencia o de deducción»***

# Justificación Sintáctica

## El método de la deducción

**Este método se sustenta en la noción de deducción dada para un sistema formal. Aquí utilizaremos el siguiente sistema formal para la lógica proposicional:**

- ✓ **Lenguaje formal: FORM1**
- ✓ **Reglas de Inferencia (Log Proposicional + reglas para el uso de cuantificadores).**

# El sistema formal que utilizaremos

1. Alfabeto de símbolos {alfabeto de FORM}
2. Conjunto de fbfs FORM restringido a predicados
3. Axiomas no usamos
4. Regla de inferencia:

   las reglas que utilizamos en lógica proposicional mas las siguientes reglas para el uso de los cuantificadores.

# Reglas de Inferencia

✓1) *Modus ponens* (M.P.): $A \rightarrow B, A$ /:. $B$

✓2) *Modus tollens* (M.T.): $A \rightarrow B, \neg B$ /:. $\neg A$

✓3) *Conjunción* (Conj.): $A, B$ /:. $A \wedge B$

✓4) *Simplificación* (Simplif.): $A \wedge B$ /:. $A$

✓5) *Adición* (Ad.): $A$ /:. $A \vee B$

✓6) *Silogismo disyuntivo* (S.D.): $A \vee B, \neg A$ /:. $B$

✓*7) Silogismo hipotético* (S.H.): $A \rightarrow B, B \rightarrow C$ /:. $A \rightarrow C$

*8) Dilema constructivo* (D.C.):

✓$A \rightarrow B, C \rightarrow D, A \vee C$ /:. $B \vee D$

✓*9) Dilema destructivo* (D.D.):

✓$A \rightarrow B, C \rightarrow D, \neg B \vee \neg D$ /:. $\neg A \vee \neg C$

# Reglas de Inferencia
## se agregan a las ya utilizadas

- ✓ $(\forall x)\ F(x) \vee (\forall x)\ G(x)\ \ /\therefore (\forall x)\ (F(x) \vee G(x))$
- ✓ $(\exists x)\ (F(x) \wedge G(x))\ \ /\therefore (\exists x)\ F(x) \wedge (\exists x)\ G(x)$
- ✓ *Ejemplificación universal (EU)*
  $(\forall x)\ F(x)$, $t \in$ TERM $/\therefore F(T)$
- ✓ *Generalización universal (GU)*
  $F(x)$, x variable $/\therefore (\forall x)\ F(x)$
- ✓ *Ejemplificación existencial (EE)*
  $(\exists x)\ F(x)$, $a \in$ TERMc, no utilizado $/\therefore F(a)$
- ✓ *Generalización existencial (GE)*
  $F(t)$, $t \in$ TERMc $/\therefore (\exists x)\ F(x) \neg C$

## *10)* Remplazo de equivalentes (R.E.):

pueden remplazarse unos por otros los siguientes pares de formas equivalentes:

Doble negación (D.N.): $\neg\neg$ **A eq A**

Conmutatividad (Conmut.) ($\wedge$ , $\vee$ , $\leftrightarrow$*)*

*Asociatividad (Asoc.)* ($\wedge$ , $\vee$ , $\leftrightarrow$*)*

*Distributividad (Distrib.)*

$$\mathbf{A \wedge (B \vee C)\ eq\ (A \wedge B) \vee (A \wedge C)}$$

$$\mathbf{A \vee (B \wedge C)\ eq\ (A \vee B) \wedge (A \vee C)}$$

§ De Morgan (De M.):

$$\mathbf{\neg(A \vee B)\ eq\ \neg A \wedge \neg B}$$

$$\mathbf{\neg(A \wedge B)\ eq\ \neg A \vee \neg B}$$

*10)* **Remplazo de equivalentes (R.E.):**

***Definición del condicional (Def $\rightarrow$):***

$$\mathbf{A \rightarrow B \;\; eq \; \neg A \vee B}$$

*Trasposición (Trasp.):* $A \rightarrow B$ eq $\neg B \rightarrow \neg A$

*Definición del bicondicional* (Def.$\leftrightarrow$)

$$\mathbf{A \leftrightarrow B \; eq \; (A \rightarrow B) \wedge (B \rightarrow A)}$$

$$\mathbf{A \leftrightarrow B \; eq \; (A \wedge B) \vee (\neg A \wedge \neg B)}$$

***Exportación (Exp.):***

$$\mathbf{(A \wedge B) \rightarrow C \; eq \; A \rightarrow (B \rightarrow C)}$$

***Idempotencia (ldemp.):***

$$\mathbf{A \; eq \;\; A \wedge A}$$

$$\mathbf{A \; eq \;\; A \vee A}$$

## Remplazo de equivalentes (R.E.):

Se agregan a las equivalencias ya utilizadas

Intercambio de cuantificadores (IC)

$(\forall x)\ F(x)$ eq $\neg\ (\exists x)\ \neg\ F(x)$

$(\exists\ x)\ F(x)$ eq $\neg\ (\forall x)\ \neg\ F(x)$

Distributividad de cuantificadores (DC)

$(\forall x)\ (F(x) \wedge G(x))$ eq $(\forall x)\ F(x) \wedge (\forall x)\ G(x)$

$(\exists x)\ (F(x) \vee G(x))$ eq $(\exists x)\ F(x) \vee (\exists x)\ G(x)$

# Razonamientos- Ejemplos

✓ $(\forall x)\ (H(x) \rightarrow M(x))$
✓ H(socrates) /∴ M(socrates)
✓ H(socrates) → M(socrates) de 1 por EU
* M(socrates) de 2 y 3 por MP

$(\forall x)\ (F(x) \wedge G(x)) \rightarrow \neg H(a))$

$\neg(\exists x)\ \neg G(x)$

$(\forall x)\ F(x)$ /∴ $\neg H(a)$

**Probar !**

# Razonamientos- Ejemplos

- Todo hombre es mamífero y todo mamífero es vertebrado. Por lo tanto todo hombre es vertebrado.

1. $(\forall x)\ (H(x) \rightarrow M(x))$
2. $(\forall x)\ (M(x) \rightarrow V(x))\ /\therefore\ (\forall x)\ (H(x) \rightarrow V(x))$
3. $H(x) \rightarrow M(x)$ de 1 por EU
4. $M(x) \rightarrow V(x)$ de 2 por EU
5. $H(x) \rightarrow V(x)$ de 3y4 por SH
6. $(\forall x)\ (H(x) \rightarrow V(x))$ de 5 por GU

# Razonamientos- Ejemplos

Los perros son vertebrados y mamíferos. Algunos perros son guardianes.
Luego, algunos vertebrados son guardianes.

**Es válido ?**

Todas las aves que no están lastimadas , vuelan.
Algunas aves no vuelan, luego, algunas aves están lastimadas.

**Es válido ?**

# El sistema formal que utilizamos

1. Es un sistema completo y consistente.
2. Las reglas de inferencia son sobreabundantes
3. Este método deductivo NO prueba que un razonamiento no es válido
4. Para probar invalidez debo probar que:

$$\not\models \varphi_1 \wedge \varphi_2 \wedge \ldots \wedge \varphi_n \rightarrow C$$

o sea encontrar una interpretación I /

$$I \models \varphi_i \quad \forall \varphi_i \in \Gamma \text{ y } \quad I \not\models C$$

# Ejemplo: Razonamiento no valido

$(\forall x)\ (M(x) \rightarrow G(x))$

$(\forall x)\ (I(x) \rightarrow \neg M(x))\ / \therefore\ (\forall x)\ (I(x) \rightarrow \neg G(x))$

Sea la interpretación I/

DI es N

<u>M(x)</u> es ¨x es múltiplo de 6 ¨

<u>G(x)</u> es ¨x es múltiplo de 3 ¨

<u>I(x)</u> es ¨x es impar¨

$\forall x \in N$ si x es múltiplo de 6 entonces x es múltiplo de 3 V

$\forall x \in N$ si x es impar, entonces x no es múltiplo de 6 V

$\forall x \in N$ si x es impar, entonces x no es múltiplo de 3 F

# Lógica de predicados

## Automatización

# Lógica proposicional

AUMENTA EL PODER EXPRESIVO

MAS COMPLEJO EL MANEJO COMPUTACIONAL

# Lógica de predicados

**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:

$$A_1 \vee A_2 \vee \ldots A_k \vee \neg A_j \vee \ldots \vee \neg A_n$$

donde $A_i$ es una fórmula atómica

# Algoritmo: fbf ➡ conjunto de cláusulas

**Llevar a forma normal prenex**

$$(Q_1x_1)\ldots(Q_nx_n) \quad (M)$$

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

http://delta.cs.cinvestav.mx/~schapa/red/logica/node29.html

# 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

$$\underbrace{(A_1 \vee A_2 \vee \ldots A_k)}_{\text{cláusula}} \wedge \ldots \wedge \underbrace{(A_1 \vee \neg A_2 \vee \ldots \neg A_k)}_{\text{cláusula}}$$

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

- Normalizar las variables de las distintas cláusulas.

Forma clausal

# Paso a forma clausal (ejemplo)

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

# Paso a forma clausal (cont.)

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

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

**Cláusulas**

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

# Resolución

- **Trabaja con razonamientos en forma cláusal**

- **Opera por refutación**
  - Agrego $\neg$ C al conjunto de las premisas en forma clausal y trato de llegar a la cláusula vacía $\varnothing$ (contradicción: $A \wedge \neg A$).

- Es un proceso iterativo simple en el cual se utiliza **una única Regla de Inferencia**
  - resolución $A \vee B, \neg A \vee C \;\; / \;\; B \vee C$

# Algoritmo: Resolución de proposiciones P l- C

- **Convertir todas las proposiciones de P a forma cláusal**
- **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

$p$

$\neg p \vee \neg q \vee r$

$\neg s \vee q$

$\neg t \vee q$

$t$

$\neg \mathbf{r}$

$\neg p \vee \neg q$

$\neg q$

$\neg t$

$\Phi$

# Resolución en Proposiciones (ejemplo)

$\neg r$ $\qquad\qquad$ $\neg p \vee \neg q \vee r$

$\neg p \vee \neg q$ $\qquad\qquad$ $p$

$\neg t \vee q$ $\qquad\qquad$ $\neg q$ $\qquad\qquad$ $\neg s \vee q$

$t$ $\qquad\qquad$ $\neg t$ $\qquad\qquad$ $\neg s$

$\Phi$

# 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:***

Sustituciones que unifican

ama (x , y)
ama (Marco, z)

(Marco/x, Paula/y, Paula/z)

(Marco/x, z/y)

ES MAS GENERAL

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

# 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 cláusal

$$\neg Perro(x) \vee Mamífero\ (x)$$
$$Perro\ (Rex)\ /\therefore\ Mamífero\ (Rex)$$

# Resolución en Predicados (ejemplo)

$\neg$ Mamífero (Rex) $\qquad\qquad$ $\neg$Perro(x) $\vee$ Mamífero (x)

Unifico con [Rex/x]

$\neg$Perro(Rex) $\qquad\qquad$ Perro (Rex)

$\phi$

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

# 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.

# 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 1er orden:
  - Cláusulas $\Rightarrow$ Cláusulas de Horn:

    tienen a lo sumo 1 literal positivo

    * $A_1 \vee \neg A_2 \vee \ldots \vee \neg A_n$
    * o su forma equivalente: $A_1 \leftarrow (A_2 \wedge \ldots \wedge A_n)$
    * en Prolog: $A_1$ :- ($A_2$ ,... ,$A_n$ )

# 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 α podemos probar que
  - » BC $\vdash$ α

**Podemos contestar preguntas como**

- -perro (Rex) ?
- - X / perro (X) ?

- Pero **no podemos** obtener todas las conclusiones ( β) que se derivan de una base
  - » β ? / BC $\vdash$ β

# Un ejemplo en PROLOG

Base sobre el mundo del Señor de los Anillos

```
hobbit(frodo).
hobbit(sam).
intenta_a(frodo,sauron).
vivian_c(X):-hobbit(X).
tierra_m(X):-vivian_c(X).
odia(X,sauron):-
  tierra_m(X),no_leal(X,sauron).
no_leal(X,Y):- intenta_a(X,Y).
```

# Un ejemplo en PROLOG

Consultas

?- odia(frodo,sauron).

Yes

?- intenta_a(X,Y).

X = frodo

Y = sauron ,

no

?- hobbit(X).

X = frodo ,

X = sam ,

no

# Una versión de PROLOG

- AMZI PROLOG
- Se puede bajar una versión libre y tiene un buen tutorial
  - » http://www.amzi.com