

# Prolog

Eduardo Morales

INAOE

2012

# Contenido

- 1 Introducción
- 2 Antecedentes
- 3 Nociones de Lógica
- 4 Relación con Lógica
- 5 Sintaxis
- 6 Listas
- 7 Aritmética
- 8 Algunos Ejemplos
- 9 Control
- 10 Selectores y Acceso a Predicados y Programas
- 11 Todas las soluciones
- 12 Operadores
- 13 Estructuras Parciales
- 14 Definite Clause Grammar (DCG)

Introducción

Antecedentes

Nociones de  
Lógica

Relación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

Control

Selectores y  
Acceso a  
Predicados y  
Programas

Todas las  
soluciones

Operadores

Estructuras  
Parciales

# Objetivo General

Introducción

Antecedentes

Nociones de  
Lógica

Relación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

Control

Selectores y  
Acceso a  
Predicados y  
Programas

Todas las  
soluciones

Operadores

Estructuras  
Parciales

*Introducir los conceptos fundamentales de la programación lógica y aprender a programar en Prolog.*

# Temario

Introducción

Antecedentes

Nociones de  
Lógica

Relación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

Control

Selectores y  
Acceso a  
Predicados y  
Programas

Todas las  
soluciones

Operadores

Estructuras  
Parciales

## 1 Introducción a la programación lógica

- Historia
- Predicados, hechos, queries, variables, reglas
- Unificación y resolución, relación con lógica
- Estrategia seguida por Prolog

# Temario (cont.)

Introducción

Antecedentes

Nociones de  
Lógica

Relación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

Control

Selectores y  
Acceso a  
Predicados y  
Programas

Todas las  
soluciones

Operadores

Estructuras  
Parciales

- ② Sintaxis, listas, estructuras, aritmética
- ③ “Backtracking”, ! (cut), **not**
- ④ **=..** (univ), **arg**, **functor**
- ⑤ **bagof**, **setof**, **findall**
- ⑥ I/O

# Temario (cont.)

Introducción

Antecedentes

Nociones de  
Lógica

Relación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

Control

Selectores y  
Acceso a  
Predicados y  
Programas

Todas las  
soluciones

Operadores

Estructuras  
Parciales

- ⑦ Diferencias de listas, estructuras incompletas
- ⑧ Meta-predicados: *numbervars*, *ground*, *verify*
- ⑨ Definite Clause Grammar (**DCG**)

# Temario (cont.)

## 10 Aplicaciones:

- 1 Búsqueda (8–puzzle)
- 2 Matemáticas simbólicas (derivación, cripto–aritmética)
- 3 Propagación con de restricciones
- 4 Aprendizaje
- 5 Lenguaje natural
- 6 Programación Lógica Inductiva
- 7 Aprendizaje por refuerzo

Introducción

Antecedentes

Nociones de  
Lógica

Relación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

Control

Selectores y  
Acceso a  
Predicados y  
Programas

Todas las  
soluciones

Operadores

Estructuras  
Parciales

# Bibliografía

Introducción

Antecedentes

Nociones de  
LógicaRelación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

Control

Selectores y  
Acceso a  
Predicados y  
ProgramasTodas las  
soluciones

Operadores

Estructuras  
Parciales

- Bratko, I. “Prolog Programming for Artificial Intelligence”, 4a edición. Addison–Wesley.
- Clocksin, W.F. y Mellish, C.S. “Programming in Prolog”, 2a edición. Springer–Verlag.
- Sterling, L. y Shapiro, E. “The Art of Prolog: Advanced Programming Techniques”. The MIT Press, Cambridge, MA 1986.
- O ’Keefe, R. “The Craft of Prolog”. The MIT Press, Cambridge, MA.



# Introducción

Introducción

Antecedentes

Nociones de  
LógicaRelación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

Control

Selectores y  
Acceso a  
Predicados y  
ProgramasTodas las  
soluciones

Operadores

Estructuras  
Parciales

- Idea de programación simbólica:  
*Manejar símbolos, asignar símbolos a objetos  
y manipularlos*
- Contraste con otros lenguajes de programación que se concentran en el cálculo numérico (no implica que no puedan hacerse cosas numéricas / simbólicas con lenguajes simbólicos / numéricos, solo que es más difícil)

# Introducción

Introducción

Antecedentes

Nociones de  
LógicaRelación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

Control

Selectores y  
Acceso a  
Predicados y  
ProgramasTodas las  
soluciones

Operadores

Estructuras  
Parciales

- E.g., las reglas usadas en SE se pueden ver como manipulaciones simbólicas
- Lenguajes más usados en IA: Lisp y Prolog

# Introducción

## Introducción

### Antecedentes

### Nociones de Lógica

### Relación con Lógica

### Sintaxis

### Listas

### Aritmética

### Algunos Ejemplos

### Control

### Selectores y Acceso a Predicados y Programas

### Todas las soluciones

### Operadores

### Estructuras Parciales

- El concepto de lógica está ligado al del pensamiento científico
- Lógica provee de un lenguaje preciso para expresar en forma explícita las metas, conocimiento, suposiciones

# Introducción

Lógica provee los fundamentos para:

- 1 Deducir consecuencias a partir de premisas
- 2 Estudiar la verdad/falsedad de proposiciones a partir de la verdad/falsedad de otras proposiciones
- 3 Establecer consistencias de nuestras suposiciones
- 4 Verificar la validez de nuestros argumentos

Introducción

Antecedentes

Nociones de  
Lógica

Relación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

Control

Selectores y  
Acceso a  
Predicados y  
Programas

Todas las  
soluciones

Operadores

Estructuras  
Parciales

# Introducción

## Introducción

### Antecedentes

#### Nociones de Lógica

#### Relación con Lógica

#### Sintaxis

#### Listas

#### Aritmética

#### Algunos Ejemplos

#### Control

#### Selectores y Acceso a Predicados y Programas

#### Todas las soluciones

#### Operadores

#### Estructuras Parciales

- Las computadoras también son objeto de estudio científico.
- Al igual que la lógica requieren de una forma precisa y de proposiciones explícitas para expresar nuestras metas y suposiciones.
- Sin embargo, mientras la lógica se ha desarrollado con el pensamiento humano como única consideración externa, el desarrollo de la máquina ha estado gobernado desde el principio por restricciones de tecnología e ingeniería.

# Introducción

## Introducción

### Antecedentes

### Nociones de Lógica

### Relación con Lógica

### Sintaxis

### Listas

### Aritmética

### Algunos Ejemplos

### Control

### Selectores y Acceso a Predicados y Programas

### Todas las soluciones

### Operadores

### Estructuras Parciales

- Aunque las computadoras fueron diseñadas para ser usadas por el hombre, sus dificultades en su construcción ha sido tan dominante, que el lenguaje para expresar problemas y decirles como hacer las cosas fue diseñado desde la perspectiva de la ingeniería de la computadora.
- Esto empezó a solucionarse con lenguajes de más alto nivel, y las dificultades empezaron a ser en la inhabilidad del humano para programar.

# Introducción

## Introducción

## Antecedentes

## Nociones de Lógica

## Relación con Lógica

## Sintaxis

## Listas

## Aritmética

## Algunos Ejemplos

## Control

## Selectores y Acceso a Predicados y Programas

## Todas las soluciones

## Operadores

## Estructuras Parciales

- Idealmente deberíamos de poder expresarnos en forma más natural...
- ... la lógica (que ha estado en el proceso del pensamiento humano desde hace mucho tiempo) ha sido un buen candidato para ésto.

*“En lugar de hacer que el hombre se adapte/dependa de las máquinas, la máquina debería de hacer las instrucciones que son fáciles de dar por el hombre”*

# Introducción

Introducción

Antecedentes

Nociones de  
LógicaRelación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

Control

Selectores y  
Acceso a  
Predicados y  
ProgramasTodas las  
soluciones

Operadores

Estructuras  
Parciales

- En su forma más pura sugiere que solo se le de conocimiento y suposiciones del problema en forma de axiomas lógicos y se ejecuta tratando de probar metas.

Programa = conjunto de axiomas

Computación = prueba de metas



# Antecedentes

Introducción

Antecedentes

Nociones de  
Lógica

Relación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

Control

Selectores y  
Acceso a  
Predicados y  
Programas

Todas las  
soluciones

Operadores

Estructuras  
Parciales

- El 1<sup>er</sup> uso de la lógica como lenguaje de programación es una secuela del algoritmo de *unificación* de Robinson y su principio de *resolución* ('65)
- Kowalski: formula la interpretación procedural de lógica en cláusulas de Horn ('72).
- Colmerauer y su grupo en Marseille—Aix desarrollan un probador de teoremas experimental escrito en Fortran para procesamiento de lenguaje natural: Prolog (“programmation en logique”) ('73).

# Antecedentes

Introducción

**Antecedentes**

Nociones de  
Lógica

Relación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

Control

Selectores y  
Acceso a  
Predicados y  
Programas

Todas las  
soluciones

Operadores

Estructuras  
Parciales

- van Edman y Kowalski desarrollan la parte semántica formal.
- Poca aceptación en EUA después de intentos parecidos y fracasos con Micro-Planeer y Conniver (ineficientes, difíciles de controlar).
- Mediados de los 70's: D. Warren hace un compilador (Prolog-10) eficiente (con la mayor parte escrito en Prolog!).

# Antecedentes

Introducción

**Antecedentes**

Nociones de  
Lógica

Relación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

Control

Selectores y  
Acceso a  
Predicados y  
Programas

Todas las  
soluciones

Operadores

Estructuras  
Parciales

- Alrededor de 1980 todavía pocos ( $\approx 100$ ) programaban en Prolog.
- Japón anuncia su Proyecto de la Quinta Generación con la programación lógica como parte medular ('81).
- Notación “estandar”: la de Edimburgo.
- Idea general (idealizada): en lugar de programar, describir lo que es verdadero.

# Nociones de Lógica

Introducción

Antecedentes

Nociones de  
LógicaRelación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

Control

Selectores y  
Acceso a  
Predicados y  
ProgramasTodas las  
soluciones

Operadores

Estructuras  
Parciales

- Un programa lógico es un conjunto de axiomas o reglas que definen relaciones entre objetos y definen un conjunto de consecuencias (su sentido).
- El arte de programar en lógica consiste en construir programas concisos y elegantes que tengan el sentido deseado.
- En programación lógica existen tres elementos:
  - ① Hechos
  - ② Reglas
  - ③ *Queries*

# Nociones de Lógica

- Finalmente todo se puede ver como cláusulas: disjunción de un conjunto finito de literales ( $\neg$  = no).

$$\{A_1, A_2, \dots, A_n, \neg B_1, \dots, \neg B_m\}.$$

Notación equivalente:

$$A_1, A_2, \dots, A_n \leftarrow B_1, B_2, \dots, B_m.$$

De:  $A \leftarrow B \equiv A \vee \neg B$  y Ley de Morgan

$$A_1 \vee A_2 \vee \dots \vee A_n \leftarrow B_1 \wedge B_2 \wedge \dots \wedge B_m$$

# Nociones de Lógica

- Introducción
- Antecedentes
- Nociones de Lógica
- Relación con Lógica
- Sintaxis
- Listas
- Aritmética
- Algunos Ejemplos
- Control
- Selectores y Acceso a Predicados y Programas
- Todas las soluciones
- Operadores
- Estructuras Parciales

- Cláusulas de Horn (a lo más una literal positiva):

$$H \leftarrow B_1, \dots, B_m$$

$$A \leftarrow \quad \quad \quad (\text{hecho})$$

$$A \leftarrow B, C, \dots \quad \quad \quad (\text{regla})$$

$$\quad \quad \quad \leftarrow B, C, \dots \quad \quad \quad (\text{querie})$$

# Hechos

- Hechos: declaran relaciones entre objetos (cosas que son verdaderas). E.g.:

quiere(juan, maria).

valioso(oro).

papa(juan,juanito).

papa(rival,ana).

papa(juanito,patricia).

mama(maria,juana).

suma(2,3,5).

clase(prolog,eduardo).

papa(juan,juana).

papa(juanito,pepe).

mama(maria,juanito).

mama(maria,ana).

- A las relaciones también se les llama *predicados*.

Introducción

Antecedentes

Nociones de Lógica

Relación con Lógica

Sintaxis

Listas

Aritmética

Algunos Ejemplos

Control

Selectores y Acceso a Predicados y Programas

Todas las soluciones

Operadores

Estructuras Parciales

# Queries

Introducción

Antecedentes

Nociones de  
LógicaRelación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

Control

Selectores y  
Acceso a  
Predicados y  
ProgramasTodas las  
soluciones

Operadores

Estructuras  
Parciales

- Queries: forma de obtener información de un programa lógico.
- Un querie pregunta si se cumple alguna relación entre objetos.
- Un hecho dice que  $P$  es verdadero.  
Una querie pregunta si  $P$  es verdadero.



# Queries

Introducción

Antecedentes

Nociones de  
LógicaRelación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

Control

Selectores y  
Acceso a  
Predicados y  
ProgramasTodas las  
soluciones

Operadores

Estructuras  
Parciales

- Las preguntas también son “metas”.  
?- papa(juan,juana).  
yes  
?- suma(1,1,2).  
no (si no está definido)
- El contestar una pregunta es determinar si la pregunta es una consecuencia lógica del programa.

# Consecuencias Lógicas

Introducción

Antecedentes

Nociones de  
Lógica

Relación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

Control

Selectores y  
Acceso a  
Predicados y  
Programas

Todas las  
soluciones

Operadores

Estructuras  
Parciales

- Las consecuencias lógicas se obtienen aplicando reglas de inferencia.
- Preguntar usando variables es como construir muchas preguntas a la vez.
- Una pregunta con variables, pregunta si existe un valor para las variables que haga la pregunta una consecuencia lógica del programa.

# Variables

Introducción

Antecedentes

Nociones de  
LógicaRelación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

Control

Selectores y  
Acceso a  
Predicados y  
ProgramasTodas las  
soluciones

Operadores

Estructuras  
Parciales

- Si queremos preguntar quien es el papá de *ana*:  
?- papa(juan,ana).  
?- papa(juanito,ana).  
?- papa(maria,ana).  
...  
?- papa(rival,ana).
- Podemos hacerlo también como:  
?- papa(X,ana).  
X = rival
- Pregunta por instancias de *X* que hacen que se cumpla la relación.

# Variables

- Las variables en lógica (entidades sin especificar) son diferentes de las variables en programas convencionales (localización en memoria).
- Pueden existir varias soluciones (instancias)
  - ?- mama(maria,X).
  - X = juana;
  - X = juanito;
  - X = ana;
  - no
- Suponiendo que tenemos la tabla de “suma”:
  - ?- suma(X,Y,4).
  - X = 0, Y = 4; X = 1, Y = 3; X = 2, Y = 2;
  - X = 3, Y = 1; X = 4, Y = 0;
  - no
  - ?- suma(X,X,4).
  - X = 2

# Variables

Introducción

Antecedentes

Nociones de  
LógicaRelación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

Control

Selectores y  
Acceso a  
Predicados y  
ProgramasTodas las  
soluciones

Operadores

Estructuras  
Parciales

- Pueden existir variables en hechos. En lugar de tener que declarar algo que sucede para todos, e.g.:  
gusta(juan,mole).  
gusta(juana,mole).  
gusta(juanito,mole).  
...  
gusta(rival,mole).
- Podemos tener variables en los hechos (e.g., a todos les gusta el mole):  
gusta(X,mole).

# Variables

Introducción

Antecedentes

Nociones de  
LógicaRelación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

Control

Selectores y  
Acceso a  
Predicados y  
ProgramasTodas las  
soluciones

Operadores

Estructuras  
Parciales

```
mult(0,X,0).
```

```
?- mult(0,4,0).
```

```
yes
```

```
suma(0,X,X).
```

```
?- suma(0,2,2).
```

```
yes
```

```
?- suma(0,3,Y).
```

```
Y = 3.
```

# Varias queries

Introducción

Antecedentes

Nociones de  
LógicaRelación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

Control

Selectores y  
Acceso a  
Predicados y  
ProgramasTodas las  
soluciones

Operadores

Estructuras  
Parciales

- También podemos hacer varias preguntas a la vez:  
?- papa(juan,juanito), papa(juanito,pepe).  
yes  
?- papa(juan,X), papa(X,pepe).  
i.e., existe una persona que sea hijo de *juan* y papá de *pepe*?  
X = juanito

# Varias queries

Introducción

Antecedentes

Nociones de  
LógicaRelación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

Control

Selectores y  
Acceso a  
Predicados y  
ProgramasTodas las  
soluciones

Operadores

Estructuras  
Parciales

- ?- papa(juan,X), papa(X,Y).  
i.e., dame los nietos de *juan*.  
X = juanito, Y = pepe;  
X = juanito, Y = patricia;  
no
- Lo podemos restringir: ?- papa(juan,X), hombre(X).



# Reglas

Introducción

Antecedentes

Nociones de Lógica

Relación con Lógica

Sintaxis

Listas

Aritmética

Algunos Ejemplos

Control

Selectores y Acceso a Predicados y Programas

Todas las soluciones

Operadores

Estructuras Parciales

- Las reglas se pueden ver de dos formas: procedural y declarativa. Una regla se escribe con el siguiente formato:

$$A \leftarrow B_1, B_2, \dots, B_n$$

Donde:  $A$  = cabeza de la regla y  $B_1, B_2, \dots, B_n$  = cuerpo de la regla.

- Procedural*: son medios de expresar nuevas/complejas queries en terminos de más sencillas.
- Declarativa*: definen nuevas/complejas relaciones usando relaciones existentes. Interpreta a la regla como un axioma lógico.

# Reglas

Introducción

Antecedentes

Nociones de  
LógicaRelación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

Control

Selectores y  
Acceso a  
Predicados y  
ProgramasTodas las  
soluciones

Operadores

Estructuras  
Parciales

hijo(Y,X) :- papa(X,Y), hombre(Y).

hija(Y,X) :- papa(X,Y), mujer(Y).

abuelo(X,Z) :- papa(X,Y), papa(Y,X).

- Procedural: para contestar ?- hijo(X,juan). hay que contestar  
?- papa(juan,X), hombre(X).
- Declarativo: para toda  $X$  y  $Y$ ,  $Y$  es el hijo de  $X$  si  $X$  es el padre de  $Y$  y  $Y$  es hombre.

# Relación con Lógica

Introducción

Antecedentes

Nociones de  
LógicaRelación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

Control

Selectores y  
Acceso a  
Predicados y  
ProgramasTodas las  
soluciones

Operadores

Estructuras  
Parciales

- Como sabemos que un programa lógico dice lo que queremos que diga?
- El significado de un programa lógico  $P$  (o su modelo  $M(P)$ ), es el conjunto de todos los hechos que podemos deducir de  $P$ .
- Hay muchas reglas de inferencia

# Relación con Lógica

Introducción

Antecedentes

Nociones de  
LógicaRelación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

Control

Selectores y  
Acceso a  
Predicados y  
ProgramasTodas las  
soluciones

Operadores

Estructuras  
Parciales

- e.g., *modus ponens*, *modus tolens*, ...

$$P \rightarrow Q$$

$$P$$

---

$$Q$$

$$P \rightarrow Q$$

$$\neg Q$$

---

$$\neg P$$

# Relación con Lógica

Introducción

Antecedentes

Nociones de Lógica

Relación con Lógica

Sintaxis

Listas

Aritmética

Algunos Ejemplos

Control

Selectores y Acceso a Predicados y Programas

Todas las soluciones

Operadores

Estructuras Parciales

- La más estudiada es *resolución*: por ser *refutation complete* (deduce todo lo que se puede por refutación) y *correcta* o *sound* (todo lo que deduce es cierto).
- El corazón del modelo de computación de programas lógicos es el algoritmo de *unificación*.

- Ejemplo de resolución:

$$P \rightarrow Q$$

$$P \leftarrow$$

---

$$Q$$

# Relación con Lógica

Introducción

Antecedentes

Nociones de  
LógicaRelación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

Control

Selectores y  
Acceso a  
Predicados y  
ProgramasTodas las  
soluciones

Operadores

Estructuras  
Parciales

- Un unificador de dos términos es una substitución que hace los dos términos iguales.
- El unificador más general o *mg**u* es un unificador tal que si existe otro unificador éste es más específico (el *mg**u* es único).
- Dada una cláusula  $C$  con una literal  $L_1$ , del lado izquierdo y otra cláusula  $D$  con una literal  $L_2$  del lado derecho, tal que  $L_1$  y  $L_2$  tienen un  $mg$ *u*( $\theta$ ), podemos construir una nueva cláusula con la unión de  $C$  y  $D$  sin  $L_1$  y  $L_2$ :

$$((C - \{L_1\}) \wedge (D - \{L_2\}))\theta$$

# Relación con Lógica

Introducción

Antecedentes

Nociones de  
LógicaRelación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

Control

Selectores y  
Acceso a  
Predicados y  
ProgramasTodas las  
soluciones

Operadores

Estructuras  
Parciales

- En general hay que encontrar instanciaciones por medio de unificación.

hijo(X,Y) :- papa(Y,X).

papa(juan,juanito).

—————  $\Theta = [Y/\text{juan}, X/\text{juanito}]$

hijo(juanito,juan).

- alumno(X,proglA) :- estudia(X,prolog).  
es\_de\_univ(A,Univ) :- alumno(A,Materia),  
imparte(Univ,Materia).

—————  $\Theta = [B/X/A, \text{proglA}/\text{Materia}]$

es\_de\_univ(B,Univ) :- estudia(B,prolog),  
imparte(Univ,proglA).

# Relación con Lógica

Introducción

Antecedentes

Nociones de Lógica

Relación con Lógica

Sintaxis

Listas

Aritmética

Algunos Ejemplos

Control

Selectores y Acceso a Predicados y Programas

Todas las soluciones

Operadores

Estructuras Parciales

- Para demostrar que algo es consecuencia lógica (i.e.,  $\mathcal{T} \models \mathcal{A}$ ), se hace prueba por refutación (i.e., añadimos  $\neg \mathcal{A}$  a  $\mathcal{T}$  y demostramos que podemos llegar a una inconsistencia)  $\mathcal{A} \wedge \mathcal{T} \vdash \square$ .
- Cuando hacemos una querie, en realidad estamos buscando inconsistencias (es como agregar  $\neg \text{querie}$  a nuestra teoría y demostrar (con resolución) una inconsistencia).
- Cuando hay variables involucradas, obtenemos instanciaciones con valores particulares.



# Algoritmo de un interprete de programas lógicos.

Introducción

Antecedentes

Nociones de Lógica

Relación con Lógica

Sintaxis

Listas

Aritmética

Algunos Ejemplos

Control

Selectores y Acceso a Predicados y Programas

Todas las soluciones

Operadores

Estructuras Parciales

Input: dado un programa lógico  $P$  y una meta  $G$

Output:  $G\theta$ , si ésta es una instancia de  $G$  deducible de  $P$   
o fracaso

Inicializa: *resolvente* =  $G$

**while** *resolvente*  $\neq \{\square\}$  **do**

    escoge una meta  $A$  del *resolvente* y una cláusula

$A' \leftarrow B_1, B_2, \dots, B_n$ , tal que  $A$  y  $A'$  unifican con un *mg*  
     $\theta$

    Quita  $A$  de *resolvente* y añade  $B_1, B_2, \dots, B_n$

    Aplica  $\theta$  a *resolvente* y a  $G$

**end while**

# Ejemplo

- Introducción
- Antecedentes
- Nociones de Lógica
- Relación con Lógica
- Sintaxis
- Listas
- Aritmética
- Algunos Ejemplos
- Control
- Selectores y Acceso a Programas
- Todas las soluciones
- Operadores
- Estructuras Parciales

Teoria:

papa(juan,juanito). (b)  
 papa(juan,juana).  
 papa(juanito,pepe).  
 papa(juanito,patricia). (a)  
 abuelo(A,B):- papa(A,C), papa(C,B).

Querie:

?- abuelo(X,Y).  
 ?- papa(X,Z), papa(Z,Y).

Si tomamos la segunda cláusula con (a):

?- papa(X,juanito) [Y = patricia]

Tomando ésta con (b):

[X = juan, Y = patricia]

$G = \text{abuelo}(X,Y) \Theta_1 \Theta_2 = \text{abuelo}(\text{juan}, \text{patricia}).$

# Relación con lógica

Introducción

Antecedentes

Nociones de  
LógicaRelación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

Control

Selectores y  
Acceso a  
Predicados y  
ProgramasTodas las  
soluciones

Operadores

Estructuras  
Parciales

- El razonamiento con resolución es un proceso combinatorio. En general, hay muchas inferencias que se pueden hacer cada vez que se hace una.
- Hay diferentes formas de control (i.e., escoger cual hacer). Lo más fácil es hacer deducciones en un orden fijo (aunque sea arbitrario): ésto es lo que hace Prolog
- Existen otros (e.g., MRS) donde se puede expresar el control, sin embargo es más ineficiente.

# Relación con lógica

Introducción

Antecedentes

Nociones de  
LógicaRelación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

Control

Selectores y  
Acceso a  
Predicados y  
ProgramasTodas las  
soluciones

Operadores

Estructuras  
Parciales

## Estrategia de Prolog:

- Escoge la meta más a la izquierda
- Busca secuencial con backtracking
- Si hacemos el mismo ejemplo pero siguiendo la estrategia de Prolog:  
?- papa(X,Y), papa(Z,Y)  
la primera con (b):  
?- papa(juanito,Y) [X = juan]  
con la tercera:  
[Y = pepe]

# Ejemplo

- Otro ejemplo para ilustrar el caso más sencillo de backtracking:  
gusta(maria,comida).  
gusta(maria,vino).  
gusta(juan,vino).  
?- gusta(maria,X), gusta(juan,X).
- En otras palabras, actúa como un *stack* (*depth-first search*).

# Ejemplo

Introducción

Antecedentes

Nociones de  
LógicaRelación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

Control

Selectores y  
Acceso a  
Predicados y  
ProgramasTodas las  
soluciones

Operadores

Estructuras  
Parciales

- Esto provoca que:
  - 1 El orden de las reglas determina el orden en que las soluciones se encuentran
  - 2 El orden de las metas (en los cuerpos de las reglas) determina el árbol de búsqueda.

# Ejemplo

Introducción

Antecedentes

Nociones de  
LógicaRelación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

Control

Selectores y  
Acceso a  
Predicados y  
ProgramasTodas las  
soluciones

Operadores

Estructuras  
Parciales

hijo(X,Y) :- padre(Y,X), hombre(X).

hijo(X,Y) :- madre(Y,X), hombre(X).

abuelo(X,Z) :- padre(X,Y), padre(Y,Z).

abuelo(X,Z) :- padre(X,Y), madre(Y,Z).

abuelo(X,Z) :- madre(X,Y), padre(Y,Z).

abuelo(X,Z) :- madre(X,Y), madre(Y,Z).

Alternativa:

procreador(X,Y) :- madre(X,Y).

procreador(X,Y) :- padre(X,Y).

hijo(X,Y) :- procreador(Y,X), hombre(X).

abuelo(X,Z):- procreador(X,Y), procreador(Y,Z).

# Disjunción

- $P :- Q ; R.$   
 $P :- Q.$   
 $P :- R.$   
 $P :- Q, R ; S, T, U.$   
 $P :- (Q,R) ; (S,T,U).$
- Hacer *trace* de:  
 $\text{gusta}(\text{maria}, \text{comida}).$   
 $\text{gusta}(\text{maria}, \text{vino}).$   
 $\text{gusta}(\text{juan}, \text{vino}).$   
 $\text{gusta}(\text{juan}, \text{maria}).$   
 $?- \text{gusta}(\text{maria}, X), \text{gusta}(\text{juan}, X).$



# Sintaxis

- Introducción
- Antecedentes
- Nociones de Lógica
- Relación con Lógica
- Sintaxis**
- Listas
- Aritmética
- Algunos Ejemplos
- Control
- Selectores y Acceso a Programas
- Todas las soluciones
- Operadores
- Estructuras Parciales

términos  $\left\{ \begin{array}{l} \text{términos simples} \left\{ \begin{array}{l} \text{constantes} \left\{ \begin{array}{l} \text{números} \\ \text{átomos} \end{array} \right. \\ \text{variables} \end{array} \right. \\ \text{estructuras (términos compuestos)} \end{array} \right.$

# Sintáxis

- e.g., de números: 1, 2, 0 (en algunos casos hay subdivisiones en reales y enteros).
- e.g., de átomos: a, 'B', c\_d, ...
- e.g., de variables: A, \_a, \_+
- e.g., de estructuras: libro(autor,título,año).

# Sintaxis

- Existen diferentes formas de representar estructuras. Estas están definidas por su nombre y número de argumentos: *functor/arity*.  
`curso(prolog,lunes,11,14,eduardo,morales,ccc,8301).`  
 VS  
`curso(prolog,horario(lunes,11,14),  
 profesor(eduardo,morales),lugar(ccc,8301)).`  
`maestro(Mat,Maestro):-  
 curso(Mat,Horario,Maestro,Lugar).`  
`duracion(Mat,Tiempo):-  
 curso(Mat,horario(Dia,TiempoI,TiempoF),Prof,Lugar),  
 suma(TiempoI,Tiempo,TiempoF).`

# Sintaxis

- VS  
dia(prolog,lunes).  
tiempol(prolog,11).  
tiempoF(prolog,14).  
profesor\_nombre(prolog,eduardo).  
...

# Apareo vs. unificación

Introducción

Antecedentes

Nociones de  
Lógica

Relación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

Control

Selectores y  
Acceso a  
Predicados y  
Programas

Todas las  
soluciones

Operadores

Estructuras  
Parciales

- Se pueden tener estructuras dentro de los argumentos.

If S y T son Constantes y  $S = T$

If S = Variable, apareo e instancia S a T

If S y T son estructuras, S y T aparecen si  
S y T tienen el mismo nombre y sus argumentos  
aparean

# Ejemplo

Introducción

Antecedentes

Nociones de  
LógicaRelación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

Control

Selectores y  
Acceso a  
Predicados y  
ProgramasTodas las  
soluciones

Operadores

Estructuras  
Parciales

- E.g., de geometría:  
P1 = punto(1,1).  
P2 = punto(2,3).  
L = linea(P1,P2) = linea(punto(1,1),punto(2,3)).  
T = triangulo(P3,P4,P5) =  
triangulo(punto(4,2),punto(6,4),punto(7,1)).  
vertical(linea(punto(X,Y),punto(X,Z))).  
horizontal(linea(punto(X,Y),punto(Z,Y))).

# Ejemplo

- Podemos decir:  
?- vertical(linea(punto(1,8),punto(1,2))).  
yes  
?- vertical(linea(punto(2,3),X)).  
X = punto(2,\_)  
?- vertical(S), horizontal(S).  
S = linea(punto(X,Y),punto(X,Y))

# Apareo vs. unificación

Introducción

Antecedentes

Nociones de  
Lógica

Relación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

Control

Selectores y  
Acceso a  
Predicados y  
Programas

Todas las  
soluciones

Operadores

Estructuras  
Parciales

- Qué le falta para hacer unificación?
- “occur check”: e.g.  $X = f(X)$  o  $X = f(a, g(X), b)$



# Reglas Recursivas

Introducción

Antecedentes

Nociones de  
LógicaRelación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

Control

Selectores y  
Acceso a  
Predicados y  
ProgramasTodas las  
soluciones

Operadores

Estructuras  
Parciales

padre(juanito, pepe).

padre(pepe, luis).

padre(luis, carlosl).

ancestro(X,Y) :- padre(X,Y).

ancestro(X,Z) :- padre(X,Y), ancestro(Y,Z).

?- ancestro(juanito, carlosl).

Probar con diferentes ordenes de las reglas y del cuerpo de la segunda regla.

# Reglas Recursivas

Introducción

Antecedentes

Nociones de  
LógicaRelación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

Control

Selectores y  
Acceso a  
Predicados y  
ProgramasTodas las  
soluciones

Operadores

Estructuras  
Parciales

- Con todo tipo de pregunta:

1

2 :- A,B

2 :- A,B

1

- Solo si existe el ancestro

1

2 :- B,A

- Loop todo el tiempo

2 :- B,A

1

# Reglas Recursivas

- Problemas: hacer recursiones en donde se llama a sí mismo (sin ningún cambio) de nuevo: e.g.,  
P :- P.  
casado(X,Y) :- casado(Y,X).  
padre(X,Y) :- hijo(Y,X).  
hijo(A,B) :- padre(B,A).

# Reglas Recursivas

Introducción

Antecedentes

Nociones de  
LógicaRelación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

Control

Selectores y  
Acceso a  
Predicados y  
ProgramasTodas las  
soluciones

Operadores

Estructuras  
Parciales

- El orden óptimo varía con el uso, e.g.:
  - a)  $\text{abuelo}(X,Z) :- \text{padre}(X,Y), \text{padre}(Y,Z).$
  - vs.
  - b)  $\text{abuelo}(X,Z) :- \text{padre}(Y,Z), \text{padre}(X,Y).$
  - a) si ?-  $\text{abuelo}(\text{nombre}, \text{Var}).$
  - b) si ?-  $\text{abuelo}(\text{Var}, \text{Nombre}).$

## Otros parecidos

liga(a,b).                      liga(b,d).

liga(a,c).                      liga(d,e).

liga(c,d).                      liga(c,f).

conecta(Nodo1,Nodo2) :- liga(Nodo1,Nodo2).

conecta(Nodo1,Nodo2) :- liga(Nodo1,Nodo3),  
conecta(Nodo3,Nodo2).

sucesor(1,2).

sucesor(2,3).

sucesor(3,4).

...

menor(Num1,Num2) :- sucesor(Num1,Num2).

menor(Num1,Num2) :- sucesor(Num1,Num3),  
menor(Num3,Num2).

# Monkey and Banana

Introducción

Antecedentes

Nociones de  
Lógica

Relación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

Control

Selectores y  
Acceso a  
Predicados y  
Programas

Todas las  
soluciones

Operadores

Estructuras  
Parciales

- Acciones (movs.):
  - tomar el plátano
  - subir a la caja
  - empujar la caja
  - caminar

# Monkey and Banana

Introducción

Antecedentes

Nociones de  
Lógica

Relación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

Control

Selectores y  
Acceso a  
Predicados y  
Programas

Todas las  
soluciones

Operadores

Estructuras  
Parciales

- Descripción:  
edo(PosHorizMono,PosVertMono,PosCaja,TienePlátano).
  - PosHorizMono: puerta, ventana, centro
  - PosVertMono: piso, sobre\_caja
  - PosCaja: puerta, ventana, centro
  - TienePlátano: si, no

# Monkey and Banana

- Representar cambios de estado con 3 argumentos:  
`mov(estado1, accion, estado2).`
- **tomar platano:**  
`mov(edo(centro,sobre_caja,centro,no),toma,  
edo(centro,sobre_caja,centro,si)).`
- **subir:**  
`mov(edo(X, piso, X, Y), sube, edo(X, sobre_caja, X, Y)).`
- **empuja:**  
`mov(edo(X, piso, X, Y), empuja(X,Z), edo(Z, piso, Z, Y)).`
- **camina:**  
`mov(edo(X, piso, Y, Z), camina(X,W), edo(W, piso, Y,  
Z)).`



# Monkey and Banana

- Para ver si puede comer (hacer *trace* de):  
?- come(edo(puerta,piso,ventana,no)).  
come(edo(\_,-,-,si)).  
come(Edo1) :-  
    mov(Edo1, Acc, Edo2),  
    come(Edo2).

# Monkey and Banana

Introducción

Antecedentes

Nociones de  
Lógica

Relación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

Control

Selectores y  
Acceso a  
Predicados y  
Programas

Todas las  
soluciones

Operadores

Estructuras  
Parciales

- El orden de los movimientos, implica una preferencia de acciones. Cambiando el orden podríamos inclusive nunca llegar a la solución (e.g., si ponemos el último movimiento al principio, se la pasaría caminando!).

# Aritmética en Lógica

- Aunque Prolog tiene mecanimos para manejo de aritmética, estos pierden su sentido lógico, sin embargo, podemos mantenerlo con los axiomas de Peano.  
 $0, s(0), s(s(0)), \dots$   
`numero(0).`  
`numero(s(X)) :- numero(X).`  
`menor_igual(0,X) :- numero(X).`  
`menor_igual(s(X),s(Y)) :- menor_igual(X,Y).`  
`minimo(N1,N2,N1) :- menor_igual(N1,N2).`  
`minimo(N1,N2,N2) :- menor_igual(N2,N1).`  
 $\text{suma}(0,X,X) \text{ :- } \text{numero}(X). \quad (0 + X = X)$   
 $\text{suma}(s(X),Y,s(Z)) \text{ :- } \text{suma}(X,Y,Z). \quad (s(X) + Y = s(X + Y))$   

$$f(A_1, A_2, \dots, A_n) \Rightarrow p(A_1, A_2, \dots, A_n, A_{n+1})$$

# Aritmética en Lógica

Introducción

Antecedentes

Nociones de  
LógicaRelación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

Control

Selectores y  
Acceso a  
Predicados y  
ProgramasTodas las  
soluciones

Operadores

Estructuras  
Parciales

- Podemos decir:  $\text{suma}(\text{Num1}, \text{Num2}, X)$ ,  
 $\text{suma}(\text{Num1}, X, \text{Num2})$ ,  $\text{suma}(X, Y, \text{Num3})$ , etc. E.g.,  
 $\text{suma}(s(0), s(s(0)), X)$ ,  $\text{suma}(s(A), X, s(s(s(0))))$ ,  
 $\text{suma}(X, Y, s(s(s(0))))$ , etc.
- Ventajas: se le pueden dar diferentes usos al programa.
- Restricciones:  $\text{suma}(X, Y, Z) \text{ :- } \text{par}(X), \text{impar}(Y)$ .  
 $\text{mult}(0, X, 0)$ .  
 $\text{mult}(s(X), Y, Z) \text{ :- } \text{mult}(X, Y, W), \text{suma}(W, Y, Z)$ .

# Listas

- $[a,b,c] = .(a,.(b,.(c,[])))$
- lista vacia = []  
 lista en general:  $[H|T]$  ( $H$  = head,  $T$  = Tail)  
 $[H|T] = [a,b,c] \Rightarrow H = a, T = [b,c]$   
 $= [a|[b,c]]$   
 $= [a,b|[c]]$   
 $= [a,b,c|[]]$   
 $[a,b,c,d,e] = [a,b,c|T] \Rightarrow T = [d,e]$

# Listas

- Podemos definir lista como:

`es_lista([]).`

`es_lista([H|T]) :- es_lista(T).`

`anade(EI,Lista,[EI|Lista]).`

`elemento(X,[X|T]).`

`elemento(X,[_|T]):-`

`elemento(X,T). (usos)`

`quita(EI,[EI|L],L).`

`quita(EI,[H|T],[H|R]):-`

`quita(EI,T,R).`

Introducción

Antecedentes

Nociones de  
LógicaRelación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

Control

Selectores y  
Acceso a  
Predicados y  
ProgramasTodas las  
soluciones

Operadores

Estructuras  
Parciales

# Listas

- También para añadir un elemento en diferentes lugares:

`quita(a,L,[b,c,d])`

`elemento(EI,L) :- quita(EI,L,_).`

`anade(EI,Lista1,Lista2) :- quita(EI,Lista2,Lista1).`

`prefijo([],L).`

`prefijo([X|T],[X|R]) :- prefijo(T,R). (usos)`

`sufijo(L,L).`

`sufijo(L,[H|T]) :- sufijo(L,T). (usos)`

`sublista(L1,L2) :-`

`prefijo(L3,L2),`

`sufijo(L1,L3).`

Introducción

Antecedentes

Nociones de  
LógicaRelación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

Control

Selectores y  
Acceso a  
Predicados y  
ProgramasTodas las  
soluciones

Operadores

Estructuras  
Parciales

# Listas

Introducción

Antecedentes

Nociones de  
LógicaRelación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

Control

Selectores y  
Acceso a  
Predicados y  
ProgramasTodas las  
soluciones

Operadores

Estructuras  
Parciales

```
sublista(L1,L2):- prefijo(L1,L2).
```

```
sublista(L1,[H|T]):- sublista(L1,L2).
```

```
append([],L,L).
```

```
append([H|L1],L2,[H|L3]):-
```

```
    append(L1,L2,L3). (usos)
```

```
adyacentes(X,Y,L) :- append(.,[X,Y|_],L).
```

```
prefijo(L1,L2):- append(L1,X,L2).
```

```
sufijo(L1,L2):- append(X,L1,L2).
```

```
sublista(L1,L2) :-
```

```
    append(X,Y,L2),
```

```
    append(L1,Z,Y).
```



# Listas

```
sublista(L1,L2):-  
    append(X,Y,L2),  
    append(Z,L1,X).
```

```
member(X,L) :- append(Y,[X|Z],L).
```

```
member(X,L) :- sublista([X,L).
```

```
last(X,[X]).  
last(X,[_|T]):- last(X,T).
```

```
last(X,L) :- append(Y,[X],L).
```

```
long([],0).  
long([H|T],s(N)):- long(T,N).
```

# Listas

- Si queremos guardar las acciones que hace el mono podemos añadir un argumento extra:

```
come(edo(−,−,−,si),[]).  
come(Edo1,[Acc|Resto]) :-  
    mov(Edo1, Acc, Edo2),  
    come(Edo2,Resto).
```

# Permutación

Introducción

Antecedentes

Nociones de  
LógicaRelación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

Control

Selectores y  
Acceso a  
Predicados y  
ProgramasTodas las  
soluciones

Operadores

Estructuras  
Parciales

(A) : permuta el resto y añade un elemento

permuta([],[]).

permuta([E|L],Perm):-

permuta(L,LP),

anade(EI,LP,Perm).

(B) : quita 1 elemento y permuta el resto

permuta([],[]).

permuta(L,[E|R]):-

quita(EI,L,L2),

permuta(L2,R).

# Permutación

- Si preguntamos: `permuta(L,[a,b,c])`.  
(A) nos da las primeras 6 soluciones y despues se cicla  
(B) nos da solo la primera y despues se cicla
- Voltea una lista:  
`voltea([],[]).`  
`voltea([H|T],Vol):-`  
    `voltea(T,TVol),`  
    `append(TVol,[H],Vol).`

# Listas

- Sin embargo, ocupa mucha memoria. Un “truco” común en Prolog (y Lisp) es incluir un argumento extra para ir guardando los resultados parciales (un acumulador) y volverla *tail recursive*.

`voltea(L,LV) :- voltea(L,[],LV).`

`voltea([],L,L).`

`voltea([H|T],Acum,Vol):-  
 voltea(T,[H|Acum],LVol).`

# Palindrome

- Palindrome lo podemos definir asi:

```
palin([]).
```

```
palin([_]).
```

```
palin([H|T]):-
```

```
    append(EnMedio,[H],T),
```

```
    palin(EnMedio).
```

- Alternativamente:  
palin(L) :- voltea(L,L).

# Aplana

- Aplana:  
    `aplana([],[]).`  
    `aplana([H|T],Res) :-`  
        `aplana(H,HP),`  
        `aplana(T,TP),`  
        `append(HP,TP,Res).`  
    `aplana(X,[X]).`

# Aplana

- Podemos hacer una version con una lista intermedia para evitar el “append”

`aplana(List,ListPI) :- aplana(List,[],ListPI).`

`aplana([],L,L).`

`aplana([H|T],L1,L3) :-`

`aplana(H,L1,L2),`

`aplana(T,L2,L3).`

`aplana(X,S,[X|S]).`



# Aritmética

- “is”
- *Valor is Expresión* => evalúa expresión y unifica el resultado a valor
  - ?-  $X \text{ is } 3 + 5.$   
 $X = 8$
  - ?-  $8 \text{ is } 3 + 5.$   
yes
  - ?-  $3 + 5 \text{ is } 3 + 5.$   
no

# Aritmética

Introducción

Antecedentes

Nociones de  
LógicaRelación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

Control

Selectores y  
Acceso a  
Predicados y  
ProgramasTodas las  
soluciones

Operadores

Estructuras  
Parciales

- *Expresión* debe de evaluar a algo (si no falla)  $X$  is  $5 + Y$  solo si “Y” tiene un valor asignado
- “is” no es función de asignación  $X$  is  $X + 1$  falla!!  
 $\text{suma}(X,Y,Z) :- Z \text{ is } X + Y.$

Asumiendo que  $X$  y  $Y$  tiene valores asignados (luego veremos como podemos asegurar esto).

# Aritmética

- Otros operadores matemáticos:  
 $-$ ,  $*$ ,  $/$ ,  $mod$ ,  $>$ ,  $<$ ,  $>=$ ,  $=<$ ,  $==$ ,  $= \setminus \setminus =$
- Factorial: Idea:  $fact(n) = n * fact(n-1)$ .  
 $factorial(N, F) :-$   
 $N > 0,$   
 $N1 \text{ is } N - 1,$   
 $fact(N1, F1),$   
 $F \text{ is } N * F1.$   
 $factorial(0, 1).$

# Aritmética

- Podemos mejorarlo si añadimos uno (o dos) argumentos extras. Una forma manteniendo un contador hacia arriba (desde 0 hasta N), la otra manteniendo un contador hacia abajo (desde N hasta 0).

$\text{fact}(N,F) \text{ :- fact}(0,N,1,F).$

$\text{fact}(I,N,T,F) \text{ :-}$

$I < N,$

$I1 \text{ is } I + 1,$

$T1 \text{ is } T * I1,$

$\text{fact}(I1,N,T1,F).$

$\text{fact}(N,N,F,F).$

# Aritmética

- Alternativamente:  
 $\text{fact}(N,F) \text{ :- fact}(N,1,F).$

$\text{fact}(N,T,F) \text{ :-}$   
     $N > 0,$   
     $T1 \text{ is } T * N,$   
     $N1 \text{ is } N - 1,$   
     $\text{fact}(N1,T1,F).$   
 $\text{fact}(0,F,F).$

# Aritmética

- Un ejemplo parecido en donde se puede usar un argumento extra para guardar resultados intermedios: suma todos los elementos de una lista

sumalista([],0).

sumalista([H|T],Res) :-

sumalista(T,Res1),

Res is Res1 + H.

sumalista(L,R) :- sumalista(L,0,R).

sumalista([H|T],Inter,R) :-

Inter1 is Inter + H,

sumalista(T,Inter1,R).

sumalista([],R,R).

# Aritmética

Introducción

Antecedentes

Nociones de  
LógicaRelación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

Control

Selectores y  
Acceso a  
Predicados y  
ProgramasTodas las  
soluciones

Operadores

Estructuras  
Parciales

- Al usar “is” muchas veces depende del uso que le demos al programa. Por ejemplo si queremos obtener la longitud de una lista de elementos.
- Conocemos N y queremos construir listas (al revés marca error)

long([H|T],N) :-

    N > 0,

    N1 is N - 1,

    long(T,N1).

long([],0).

# Ejemplo

- Conocemos la lista y queremos conocer la longitud (al revés se cicla)

`long([],0).`

`long([_|T],N) :-  
 long(T,N1),  
 N is 1 + N1.`



## Ejemplo (cont).

Introducción

Antecedentes

Nociones de  
LógicaRelación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

Control

Selectores y  
Acceso a  
Predicados y  
ProgramasTodas las  
soluciones

Operadores

Estructuras  
Parciales

long([\_|T],N) :-  
    long(T,N1),  
    N = 1 + N1.

long([\_|T],N) :-  
    N is 1 + N1.  
    long(T,N1),

long([H|T],1 + N) :-  
    long(T,N1),

?- long([a,b,c],N), Long is N.

# Ejemplo

- Usando sucesor el código es más sencillo y se puede usar de las dos formas:

$\text{long}([H|T],s(N)) :-$   
     $\text{long}(T,N).$   
 $\text{long}([],0).$

Introducción

Antecedentes

Nociones de  
LógicaRelación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

Control

Selectores y  
Acceso a  
Predicados y  
ProgramasTodas las  
soluciones

Operadores

Estructuras  
Parciales

# Ordena Listas

Introducción

Antecedentes

Nociones de  
LógicaRelación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

Control

Selectores y  
Acceso a  
Predicados y  
ProgramasTodas las  
soluciones

Operadores

Estructuras  
Parciales

- Ordena Listas
- Una idea poco eficiente es hacer permutaciones y checar que esté en orden:

```
sort(L,LO) :-  
    permuta(L,LO),  
    orden(LO).
```

```
orden([A,B|T]) :-  
    A < B,  
    orden([B|T]).  
orden([_]).
```

# Ordena Listas

- Una solución un poco mejor es insertar el elemento en el lugar que le corresponde:

```
sort([H|T],LO) :-
    sort(T,TO),
    inserta(H,TO,LO).
sort([],[]).
```

```
inserta(H,[],[H]).
inserta(H,[F|T],[F|R]) :-
    H > F,
    inserta(H,T,R).
inserta(H,[F|R],[H,F|R]) :-
    H <= F.
```

# Ordena Listas

Quicksort: dividir en menores y mayores a un cierto elemento, ordenarlos y juntarlos todos.

```
qsort([H|T],LO) :-
    divide(H,T,Men,May),
    qsort(Men,MenO),
    qsort(May,MayO),
    append(MenO,[H|MayO],LO).
qsort([],[]).
```

```
divide(EI,[H|T],[H|Men],May) :-
    EI > H,
    divide(EI,T,Men,May).
divide(EI,[H|T],Men,[H|May]) :-
    EI < H,
    divide(EI,T,Men,May).
divide(_,[],[],[]).
```

Introducción

Antecedentes

Nociones de  
LógicaRelación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

Control

Selectores y  
Acceso a  
Predicados y  
ProgramasTodas las  
soluciones

Operadores

Estructuras  
Parciales

# Ordena Listas

- Mejora: tener una lista intermedia que sirva de acumulador

```
qsort(L,LO) :- qsort(L,[],LO).
```

```
qsort([H|T],Acum,LO) :-  
    divide(H,T,Men,May),  
    qsort(May,Acum,MayO),  
    qsort(Men,[H|MayO],LO).  
qsort([],L,L).
```

# Mini-aplicación: NDFA

- Representación: *trans*(*Edo1*, *Simbolo*, *Edo2*)., *trans\_vacio*(*Edo1*,*Edo2*)., *final*(*Edo*).  
*acepta*(*Edo*,[]) :-  
     *final*(*Edo*).  
*acepta*(*Edo*,[H|T]) :-  
     *trans*(*Edo*,H,NEdo),  
     *acepta*(NEdo,T).  
*acepta*(*Edo*,String) :-  
     *trans\_vacio*(*Edo*,NEdo),  
     *acepta*(NEdo,String).

# Mini-aplicación: NDFA

Introducción

Antecedentes

Nociones de  
LógicaRelación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

Control

Selectores y  
Acceso a  
Predicados y  
ProgramasTodas las  
soluciones

Operadores

Estructuras  
Parciales

$\text{trans}(s1,0,s1).$   $\text{trans}(s1,1,s2).$

$\text{trans}(s1,0,s4).$   $\text{trans}(s2,0,s2).$

$\text{trans}(s2,0,s3).$   $\text{trans}(s3,1,s5).$

$\text{trans\_vacio}(s4,s2).$   $\text{final}(s5).$

?-  $\text{ acepta}(s1,[0,1,0,1]).$

yes

?-  $\text{ acepta}(X,[0,0,1]).$

$X = s1;$   $X = s2;$   $X = s4$

?-  $\text{ acepta}(s1,[X,Y,Z]).$

$X = 1,$   $Y = 0,$   $Z = 1;$

$X = 0,$   $Y = 0,$   $Z = 1$

?-  $\text{ Sol} = [_,_,_], \text{ acepta}(s1, \text{ Sol}).$

$\text{ Sol} = [1,0,1];$   $\text{ Sol} = [0,0,1]$



# Mini-aplicación: NDFA

Introducción

Antecedentes

Nociones de  
LógicaRelación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

Control

Selectores y  
Acceso a  
Predicados y  
ProgramasTodas las  
soluciones

Operadores

Estructuras  
Parciales

- Si le ponemos un ciclo de transiciones vacías entonces podemos caer en loops infinitos si el string que se le da no es una solución (y en ciertos casos, aunque sea una solución): e.g., `trans_vacio(s2,s4)`. y `?- acepta(s1,[1])`.

# Control

Introducción

Antecedentes

Nociones de  
Lógica

Relación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

**Control**

Selectores y  
Acceso a  
Predicados y  
Programas

Todas las  
soluciones

Operadores

Estructuras  
Parciales

- Antes solo un recordatorio: las variables lógicas no permiten asignación destructiva. Una vez asignado un valor ese se le queda a menos que se le quite por medio de backtracking
- Toda la manipulación de datos se hace por medio de unificación
- El programador puede afectar la ejecución de un programa cambiando el orden de las cláusulas y de las metas

# Control

Introducción

Antecedentes

Nociones de  
Lógica

Relación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

**Control**

Selectores y  
Acceso a  
Predicados y  
Programas

Todas las  
soluciones

Operadores

Estructuras  
Parciales

- Prolog tiene un predicado especial llamado “CUT” (!) que puede afectar el comportamiento procedural de los programas
- Con el ! podemos prevenir Backtracking
- Función principal: reducir el espacio de búsqueda de soluciones cortandolo dinámicamente

# Control

Introducción

Antecedentes

Nociones de  
LógicaRelación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

Control

Selectores y  
Acceso a  
Predicados y  
ProgramasTodas las  
soluciones

Operadores

Estructuras  
Parciales

- Puede eliminar fácilmente el sentido declarativo de los programas!!,
- Para ver su uso más fácil, consideremos una función que tiene predicados mutuamente exclusivos. e.g., una función escalón:

$$f(X,0) :- X < 3.$$
$$f(X,2) :- X \geq 3, X < 6.$$
$$f(X,4) :- X \geq 6.$$

- Si preguntamos:  $?- f(1,Y), 2 < Y$ . Prolog a través del backtracking trata 2 soluciones que se ven que van a fallar

# Control

- Las 3 reglas son mutuamente exclusivas por lo que en el momento de que una jala, no tiene sentido en tratar con las otras 2. Esto puede hacerse con el !.

$f(X,0) :- X < 3, !.$

$f(X,2) :- X \geq 3, X < 6, !.$

$f(X,4) :- X \geq 6.$

# Control

Introducción

Antecedentes

Nociones de  
LógicaRelación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos**Control**Selectores y  
Acceso a  
Predicados y  
ProgramasTodas las  
soluciones

Operadores

Estructuras  
Parciales

- Todavía podemos reducir más el código: Idea: si la primera cláusula falla ya sabemos que  $X \geq 3$  (lo mismo para la primera condición de la tercera cláusula)

 $f(X,0) :- X < 3, !.$  $f(X,2) :- X < 6, !.$  $f(X,4).$ 

- En principio da el mismo resultado, pero le quitamos el sentido lógico (si le quitamos los cuts nos da varias soluciones)

# Control

Introducción

Antecedentes

Nociones de  
LógicaRelación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

Control

Selectores y  
Acceso a  
Predicados y  
ProgramasTodas las  
soluciones

Operadores

Estructuras  
Parciales

- Para entender más claramente la función del CUT:
- Meta Padre: la meta que apareó la cabeza de la cláusula que tiene el !
- Cuando se encuentra el ! como meta, ésta se cumple, pero “compromete” al sistema a todas las opciones hechas entre la meta padre y el ! (i.e., las otras posibles opciones entre la meta padre y el ! se eliminan)

$$H : -B_1, B_2, \dots, B_n, !, B_{n+1}, \dots, B_z.$$

# Control

- e.g., (poner árbol AND/OR)

$P :- Q, R.$

$P :- S, !, T.$

$P :- U, V.$

$A :- B, P, C.$

- Afecta el comportamiento de “P” pero no de “A”



# Control

- Usos:
  - decirle que ya encontró una solución correcta
  - decirle que si llegaste ahí, entonces falla (en conjunción con *fail*)

# Control

- Que hace ! ?
  - poda todas las cláusulas debajo de ésta
  - poda todas las posibles soluciones al lado izquierdo del cut
  - no afecta el lado derecho

# Control

Introducción

Antecedentes

Nociones de  
LógicaRelación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos**Control**Selectores y  
Acceso a  
Predicados y  
ProgramasTodas las  
soluciones

Operadores

Estructuras  
Parciales

- e.g.,  
 $\text{max}(X,Y,X) :- X \geq Y, !.$   
 $\text{max}(X,Y,Y) :- X < Y.$

De hecho Bratko en su libro quita la condición de la 2a cláusula, sin embargo esto solo funciona si preguntas por el Max, pero falla si preguntas: ?- max(4,2,2).

- Posible solución:  
 $\text{max}(X,Y,Z) :- X \geq Y, !, Z = X.$   
 $\text{max}(X,Y,Y).$

# Ejemplo

- Puede servir para salirse al encontrar la 1a. solución  

$$\text{member\_ck}(H, [H|_]) \text{ :- } !.$$

$$\text{member\_ck}(H, [_|T]) \text{ :-}$$

$$\text{member\_ck}(H, T).$$
- Añade un elemento, pero solo si no está en la lista:  

$$\text{anade}(E, L, L) \text{ :- member}(E, L), !.$$

$$\text{anade}(E, L, [E|L]).$$
- Es difícil hacer lo mismo sin el !

# Negación por falla

- **true** siempre se cumple
- **fail** siempre falla

Introducción

Antecedentes

Nociones de  
Lógica

Relación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

**Control**

Selectores y  
Acceso a  
Predicados y  
Programas

Todas las  
soluciones

Operadores

Estructuras  
Parciales

# Negación por falla

- En particular el **!** se puede usar en conjunción con **fail** para indicar que si ya llegaste ahí entonces falla, i.e., **!**, **fail**.
- E.g.,  
impuesto(X,N) :- desempleado(X), **!**, fail.  
impuesto(X,N) :- sueldo(X,Sx),  $Sx < 4 * \text{Min}$ , **!**, fail.  
impuesto(X,N) :- ...

# Negación por falla

- Por ejemplo para indicar *a menos que*:

`gusta(Maria,X) :- vibora(X), !, fail.`

`gusta(Maria,X) :- animal(X).`

`gusta(Maria,X) :-`

`( vibora(X), !, fail`

`; animal(X) ).`

Introducción

Antecedentes

Nociones de  
LógicaRelación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

Control

Selectores y  
Acceso a  
Predicados y  
ProgramasTodas las  
soluciones

Operadores

Estructuras  
Parciales

# Negación por falla

- Usando la misma idea:

diferente(X,X) :- !, fail.

diferente(X,Y).

diferente(X,Y) :-

( X = Y, !, fail

; true ).



# Negación por falla

- **not**: aunque los Prologs tienen definido NOT, lo podemos definir en términos de **!** y **fail**

`not(P) :- P, !, fail.`

`not(P).`

Aquí si cambiamos el orden cambia el sentido!!

# Negación por falla

- Generalmente **not** se define como  $\backslash +$   
 $\text{gusta}(\text{maria}, X) \text{ :- } \text{animal}(X), \backslash + \text{vibora}(X).$   
 $\text{diferente}(X, Y) \text{ :- } \backslash + X = Y.$

Introducción

Antecedentes

Nociones de  
LógicaRelación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos**Control**Selectores y  
Acceso a  
Predicados y  
ProgramasTodas las  
soluciones

Operadores

Estructuras  
Parciales

# Negación por falla

- Cuando se usa  $\backslash +$  el orden de las reglas es esencial. Lo que hace Prolog es *Negación por Falla* con *CWA* (i.e., si no lo puede probar, entonces asume que es falso)
- NO es negación lógica y puede tener problemas:  
`est_soltero(X) :-`  
`$\backslash +$  casado(X),`  
`estudiante(X).`  
`estudiante(juan).`  
`casado(pepe).`  
  
`?- est_soltero(X)`

# Negación por falla

- Así falla sin considerar a *juan*, sin embargo, si cambiamos el orden de las literales en la regla, si se cumple. Otros casos que ilustran esto:

?-  $\backslash + X = 1, X = 2$

$r(a).$

$q(b).$

$p(X) :- \backslash + r(X).$

?-  $q(X), p(X).$

$X = b.$

?-  $p(X), q(X).$

no

$p(s(X)) :- p(X).$

$q(a).$

?-  $\backslash + (p(X), q(X)).$

# Negación por falla

- Sin embargo, podemos usarlo en ciertos casos:  
`gusta(X,Y) :- animal(X), \+ vibora(X).`

`conj_disj(C1,C2) :-  
 \+ (member(X,C1), member(X,C2)).`

# CUTs

- Los CUTs se pueden clasificar en:
  - *Verdes*: no afectan el sentido declarativo (recortan posibles caminos que no dan soluciones nuevas)
  - *Rojos*: afectan el sentido declarativo (recortan condiciones explícitas)

# CUTs

Introducción

Antecedentes

Nociones de  
LógicaRelación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos**Control**Selectores y  
Acceso a  
Predicados y  
ProgramasTodas las  
soluciones

Operadores

Estructuras  
Parciales

- $p :- a, b. \implies (a \wedge b) \vee c$   
 $p :- c.$
- $p :- a, !, b. \implies (a \wedge b) \vee (\neg a \wedge c)$   
 $p :- c.$
- $p :- c. \implies c \vee (a \wedge b)$   
 $p :- a, !, b.$   
e.g., (de nuevo recordar la definición de MAX)

# CUTs

Introducción

Antecedentes

Nociones de  
LógicaRelación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

Control

Selectores y  
Acceso a  
Predicados y  
ProgramasTodas las  
soluciones

Operadores

Estructuras  
Parciales

- Los CUTs rojos eliminan pruebas y a veces son necesarios, pero hay que tener mucho cuidado porque esto afecta el sentido del programa. Un ejemplo útil, es definiendo If-Then-Else  
 $\text{ifthenelse}(P, Q, R) :- P, !, Q.$   
 $\text{ifthenelse}(P, Q, R) :- R.$
- Ponerlo “bien” sería añadir en la segunda regla:  $\backslash + \text{not } P, R$ , pero evaluar  $\backslash + P$  puede ser MUY costoso, y de hecho ya lo hicimos una vez (si no, no estaríamos en la segunda regla)



# If-Then-Else

Introducción

Antecedentes

Nociones de  
LógicaRelación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos**Control**Selectores y  
Acceso a  
Predicados y  
ProgramasTodas las  
soluciones

Operadores

Estructuras  
Parciales

- ifthenelse: viene definido en la mayoría de los Prologs como:  $P \rightarrow Q ; R$
- Notas (sobre los CUTs):
  - Si la solución está determinada NO hace falta
  - Un CUT se debe de poner en el punto exacto en donde se sabe que es la rama correcta del árbol de prueba
  - Hay que tener cuidado de no mover “cálculos” antes del CUT ya que pueden fallar y hacer que no se llegue al CUT

# CUTs

- Usos de CUTs rojos: (i) obtener la primera solución (ii) obtener un efecto if-then-else. E.g.,  
primera(Meta) :- call(Meta), !.

Introducción

Antecedentes

Nociones de  
LógicaRelación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos**Control**Selectores y  
Acceso a  
Predicados y  
ProgramasTodas las  
soluciones

Operadores

Estructuras  
Parciales

# Ejemplo

```
aplana(L,LA) :- aplana(L,[],LA).
```

```
aplana([],L,L) :- !.
```

```
aplana([H|T],L1,L) :-
```

```
    !,
```

```
    aplana(H,L1,L2),
```

```
    aplana(T,L2,L).
```

```
aplana(X,L,[X|L]).
```

Aquí los CUTs de las dos primeras cláusulas, son para darle sentido a la última!!. Deberíamos de tener en lugar de los CUTs, en la última:

$X = []$ ,  $X = [-|-]$ .

# Ejemplo

Introducción

Antecedentes

Nociones de  
LógicaRelación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos**Control**Selectores y  
Acceso a  
Predicados y  
ProgramasTodas las  
soluciones

Operadores

Estructuras  
Parciales

?- aplana([],X).

X = []

?- X = [\_], aplana([],X).

X = [[]]

Qué pasó? El ! debe ponerse tan pronto que se determine que el camino de prueba es el correcto: i.e.,  $\text{aplanan}([],L1,L)$   
 $\text{:- !, } L1 = L.$

# Selectores

Introducción

Antecedentes

Nociones de  
LógicaRelación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

Control

Selectores y  
Acceso a  
Predicados y  
ProgramasTodas las  
soluciones

Operadores

Estructuras  
Parciales`atom(X).``integer(X).``atomic(X) :- atom(X) ; integer(X).``var(X).``nonvar(X).``?- var(X), X = 2.``X = 2``?- X = 2, var(X).``no``?- atom(2).``no``?- atomic(2).``yes`

# Igualdades

Introducción

Antecedentes

Nociones de  
Lógica

Relación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

Control

Selectores y  
Acceso a  
Predicados y  
Programas

Todas las  
soluciones

Operadores

Estructuras  
Parciales

$X = Y$

$X \text{ is } \text{Expr1}$

$\text{Exp1} ::= \text{Exp2}$

$\text{Exp1} \neq \text{Exp2}$

$T1 == T2$

$T1 \neq T2$

# Acceso a Predicados

- Introducción
- Antecedentes
- Nociones de Lógica
- Relación con Lógica
- Sintaxis
- Listas
- Aritmética
- Algunos Ejemplos
- Control
- Selectores y Acceso a Predicados y Programas
- Todas las soluciones
- Operadores
- Estructuras Parciales

functor(Predicado,Nombre,Aridad).

Usos:

- 1 obtener nombre y número de args
- 2 construir un predicado

?- functor(papa(juan,juanito),F,N).

F = juan, N = 2

?- functor(Pred,foo,3).

Pred = foo(.,.,.).

# Acceso a Predicados

Introducción

Antecedentes

Nociones de  
LógicaRelación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

Control

Selectores y  
Acceso a  
Predicados y  
ProgramasTodas las  
soluciones

Operadores

Estructuras  
Parciales $\text{arg}(\text{N}, \text{Pred}, \text{Valor}).$ 

Usos:

- 1 seleccionar un argumento
- 2 instanciar un argumento

 $?- \text{arg}(2, \text{arbol}(6, 4, 8), X).$  $X = 4$  $?- \text{arg}(2, \text{arbol}(3, X, 7), 1).$  $X = 1$



# Substituye un elemento por otro

Introducción

Antecedentes

Nociones de  
LógicaRelación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

Control

Selectores y  
Acceso a  
Predicados y  
ProgramasTodas las  
soluciones

Operadores

Estructuras  
Parciales

```
subst(Old,New,Old,New).
```

```
subst(Old,New,Otro,Otro) :-
```

```
    atomic(Otro),
```

```
    Otro \ = Old.
```

```
subst(Old,New,Pred,NPred) :-
```

```
    functor(Pred,F,N),
```

```
    functor(NPred,F,N),
```

```
    subst_aux(N,Old,New,Pred,NPred).
```

```
subst_aux(N,Old,New,Pred,NPred) :-
```

```
    N > 0,
```

```
    arg(N,Pred,ArgN),
```

```
    subst(Old,New,ArgN,NArgN),
```

```
    arg(N,NPred,NArgN),
```

```
    N1 is N - 1,
```

```
    subst_aux(N1,Old,New,Pred,NPred).
```

# Ejemplo

El programa de arriba no toma en cuenta variables. Si tuvieramos variables, entonces hay que cambiar la primera por lo siguiente:

```
subst(Old,New,Var,Var) :- var(Var), !.  
subst(Old,New,Viejo,New):- atomic(Viejo), Old = Viejo, !.
```

- “univ”  $\Leftrightarrow$  “=..”
- $\text{Pred} =.. [F|\text{Args}]$
- E.g.,  
     $\text{padre}(a,b) =.. [\text{padre},a,b]$   
    ?-  $\text{Meta} =.. [\text{foo},a,b,c].$   
     $\text{Meta} = \text{foo}(a,b,c).$   
    ?-  $\text{foo}(a,b,c) =.. [P|A].$   
     $P = \text{foo}, A = [a,b,c]$

# Crear Predicados

Introducción

Antecedentes

Nociones de  
Lógica

Relación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

Control

Selectores y  
Acceso a  
Predicados y  
Programas

Todas las  
soluciones

Operadores

Estructuras  
Parciales

- Con functor y arg o con  $=..$ , podemos crear nuevos predicados.
- Podemos definir “univ” en términos de “functor” y “arg”.

# Functor, Arg y Univ

Si sabemos el predicado y queremos obtener la lista  $[F|Args]$ :

```
univ(Pred,[F|Args]) :-
    functor(Pred,F,N),
    args(1,N,Pred,Args).
```

```
args(I,N,Pred,[Arg1|Args]) :-
    I =< N,
    arg(I,Pred,Arg1),
    I1 is I + 1,
    args(I1,N,Pred,Args).
args(I,N,-,[]):- I > N.
```

# Functor, Arg y Univ

Sabemos la lista  $[F \mid \text{Args}]$  y queremos el predicado:

```
univ(Pred,[F|Args]) :-
    long(Args,N),
    functor(Pred,F,N),
    args(Args,Pred,1).
```

```
args([Arg1|Args],Pred,N) :-
    arg(N,Pred,Arg),
    N1 is N + 1,
    args(Args,Pred,N1).
args([],-,-).
```

```
long([],0).
long([H|T],N) :-
    long(T,N1),
    N is N1 + 1.
```

# Maplist

Maplist: tener un predicado que queremos aplicar a una lista de argumentos (asumiendo que el resultado siempre es el último argumento)

```
maplista(−,[ ],[ ]).
```

```
maplista(F,[L1|T1],[Res1|ResT]) :-
```

```
    junta(L1,[Res1],Args),
```

```
    Meta =.. [F|Args],
```

```
    call(Meta),
```

```
    maplista(F,T1,ResT).
```

e.g.,

```
?- maplista(junta,[[[a,b],[c,d]],[1,2],[3,4]],X).
```

```
X = [[a,b,c,d],[1,2,3,4]]
```

```
?- maplista(quita,[a,[b,c,a,d]],[1,[2,1,3,4]],X).
```

```
X = [[b,c,d],[2,3,4]]
```

Introducción

Antecedentes

Nociones de  
LógicaRelación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

Control

Selectores y  
Acceso a  
Predicados y  
ProgramasTodas las  
soluciones

Operadores

Estructuras  
Parciales

# Copia

Para copiar una estructura a otra:

```
copy(Pred,CPred) :-
    copy(Pred,CPred,[],-).
```

```
copy(Var1,Var2,Subst,Subst) :-
    var(Var1),
    v_member(Var1/Var2,Subst),
    !.
```

```
copy(Var1,Var2,Subst,[Var1/Var2|Subst]) :-
    var(Var1), !.
```

```
copy(Pred,CPred,Subst1,Subst2) :-
    functor(Pred,F,N),
    functor(CPred,F,N),
    copy_args(N,Pred,CPred,Subst1,Subst2).
```



# Copia (cont.)

Introducción

Antecedentes

Nociones de  
LógicaRelación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

Control

Selectores y  
Acceso a  
Predicados y  
ProgramasTodas las  
soluciones

Operadores

Estructuras  
Parciales

```

copy_args(0,_,_,S,S) :- !.
copy_args(N,Pred,CPred,S1,S3) :-
    arg(N,Pred,ArgN),
    copy(Arg1,CArg1,S1,S2),
    arg(N,CPred,CPredN),
    N1 is N - 1,
    copy_args(N1,Pred,CPred,S2,S3).

```

```

v_member(A/B,[V1/B|_]) :-
    A == V1, !.
v_member(X,[_|T]) :-
    v_member(X,T).

```

# Acceso y manipulación de programa

Introducción

Antecedentes

Nociones de  
LógicaRelación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

Control

Selectores y  
Acceso a  
Predicados y  
ProgramasTodas las  
soluciones

Operadores

Estructuras  
Parciales

clause(Head,Body):

- “Head” debe de tener el nombre de una cláusula y nos regresa el cuerpo de ésta.
- Los “hechos” (i.e., con cuerpo vacío) nos regresa “true”

?- clause(append(X,Y,Z),L).

X = [], Y=\_348, Z=\_348, L=true;

X=[\_780|\_773], Y=\_348, Z=[\_780|\_775], L=junta(\_773,\_348,\_775)

Esto nos permite acceder a cláusulas que tengamos definidas dentro de un programa

# Acceso y manipulación de programa

Introducción  
 Antecedentes  
 Nociones de Lógica  
 Relación con Lógica  
 Sintaxis  
 Listas  
 Aritmética  
 Algunos Ejemplos  
 Control  
 Selectores y Acceso a Predicados y Programas  
 Todas las soluciones  
 Operadores  
 Estructuras Parciales

Así como podemos acceder predicados con *clause/2*, también podemos añadir o quitar cláusulas con: `assert(X)` y `retract(X)`. e.g.,

?- entiendo.

no

?- `assert(entiendo)`.

yes

?- entiendo.

yes

?- `retract(entiendo)`.

yes

?- entiendo.

no

# Más ejemplos

- Introducción
- Antecedentes
- Nociones de Lógica
- Relación con Lógica
- Sintaxis
- Listas
- Aritmética
- Algunos Ejemplos
- Control
- Selectores y Acceso a Predicados y Programas
- Todas las soluciones
- Operadores
- Estructuras Parciales

a :- b, c.

d :- b, not c.

e :- c, f.

b.

c.

?- a.

yes

?- assert(f).

yes

?- e.

yes

?- retract(c).

yes

## Más ejemplos

Con assert y retract podemos añadir o quitar reglas

```
?- retract((append(.,.,.) :- append(.,.,.))).
```

```
?- assert((append([H|T],L,[H|R]) :- append(T,L,R))).
```

Variantes: asserta(X), assertz(X), abolish(F/N). e.g.,

```
?- asserta(foo(a,b)), assertz(foo(b,c)), asserta(foo(a,c)).
```

```
?- foo(X,Y).
```

```
?- functor(Pred,append,3), clause(Pred,Cuerpo),
   retract((Pred :- Cuerpo)).
```

Una forma en que podemos usar assert es para guardar soluciones:

```
lemma(P) :- P, asserta((P :- !)).
```

Introducción

Antecedentes

Nociones de  
LógicaRelación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

Control

Selectores y  
Acceso a  
Predicados y  
ProgramasTodas las  
soluciones

Operadores

Estructuras  
Parciales

# Torres de Hanoi

Introducción

Antecedentes

Nociones de  
LógicaRelación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

Control

Selectores y  
Acceso a  
Predicados y  
ProgramasTodas las  
soluciones

Operadores

Estructuras  
Parciales

```

hanoi(s(0),A,B,C,[A a B]).
hanoi(s(N),A,B,C,Movs) :-
    hanoi(N,A,C,B,Movs1),
    hanoi(N,C,B,A,Movs2),
    append(Movs1,[A a B | Movs2],Movs).

```

```

hanoi(1,A,B,C,[A a B]).
hanoi(N,A,B,C,Movs) :-
    N > 1,
    N1 is N - 1,
    lemma(hanoi(N1,A,C,B,Movs1)),
    hanoi(N1,C,B,A,Movs2),
    append(Movs1,[A a B | Movs2],Movs).

```

# Hanoi (cont.)

- Para probarlo, primero resolver en general y luego instanciar (para aprovechar los lemmas).

hanoi(N,Discos,Movs) :-  
    hanoi(N,A,B,C,Movs),  
    Discos = [A,B,C].

# Fobonacci

- Un ejemplo clásico es Fibonacci: el número  $N$  de Fibonacci (excepto los 2 primeros que es 1) es =  $f(N - 1) + f(N - 2)$

fibonacci(1,1).

fibonacci(2,1).

fibonacci(N,F) :-

$N > 2,$

$N1$  is  $N - 1,$

fibonacci( $N1$ , $F1$ ),

$N2$  is  $N - 2,$

fibonacci( $N2$ , $F2$ ),

$F$  is  $F1 + F2,$

asserta(fibonacci(N,F)).

Introducción

Antecedentes

Nociones de  
LógicaRelación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

Control

Selectores y  
Acceso a  
Predicados y  
ProgramasTodas las  
soluciones

Operadores

Estructuras  
Parciales



## Fibonacci (cont.)

- Otra forma de hacerlo es guardar los resultados al ir haciendolo.
- Aumentar 2 argumentos mas para tener estos resultados parciales (de  $N - 1$  y de  $N - 2$ ). Osea:  
 $\text{ffibo}(2, N, F - 1, F - 2, (F - 1 + F - 2))$

$\text{ffibo}(N, F) \text{ :- ffibo}(2, N, 1, 1, F).$

$\text{ffibo}(M, N, F1, F2, F2) \text{ :- } M \geq N.$

$\text{ffibo}(M, N, F1, F2, F) \text{ :-}$

$M < N,$

$M1 \text{ is } M + 1,$

$NF2 \text{ is } F1 + F2,$

$\text{ffibo}(M1, N, F2, NF2, F).$

# Más ejemplos

Introducción

Antecedentes

Nociones de  
LógicaRelación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

Control

Selectores y  
Acceso a  
Predicados y  
ProgramasTodas las  
soluciones

Operadores

Estructuras  
Parciales

- Una forma para copiar dos términos es usando *assert* y *retract*.

`copy(X,Y) :- asserta('$tmp'(X)), retract('$tmp'(Y)).`

- Un predicado útil es *repeat*:

`repeat.`

`repeat :- repeat.`

# Usos de *repeat*

- Por ejemplo para consultar “manualmente” un archivo:  
consulta(Archivo) :-  
    see(Archivo),  
    procesa,  
    seen.

# Alternativamente

```
consult(Archivo) :-  
    open(Stream, Archivo, r),  
    procesa(Stream),  
    close(Stream).
```

```
procesa :-  
    repeat,  
    read(Clausula),  
    procesa(Clausula), !.
```

```
procesa(X) :-  
    end_of_file(X), !.  
procesa(Clausula) :-  
    asserta(Clausula), fail.
```

# Variables Globales

Introducción

Antecedentes

Nociones de  
LógicaRelación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

Control

Selectores y  
Acceso a  
Predicados y  
ProgramasTodas las  
soluciones

Operadores

Estructuras  
Parciales

- Con assert podemos simular variables globales (aunque le quitan la parte declarativa a los programas, i.e., no usar mucho):

```
actualiza_var_global(Nombre,Valor) :-  
    nonvar(Nombre),  
    retract(var_global(Nombre,_)), !,  
    asserta(var_global(Nombre,Valor)).  
actualiza_var_global(Nombre,Valor) :-  
    nonvar(Nombre),  
    asserta(var_global(Nombre,Valor)).
```

# Name

Introducción

Antecedentes

Nociones de  
LógicaRelación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

Control

Selectores y  
Acceso a  
Predicados y  
ProgramasTodas las  
soluciones

Operadores

Estructuras  
Parciales

name(Atomo,ListaAscci)

?- name(abc,L).

L = [97,98,99]

Aplicación: *gensym* (crea un nuevo nombre)

gensym(Prefijo,Valor) :-

var(Valor),

atom(Prefijo),

obten\_valor(gensym(Prefijo),N),

N1 is N + 1,

actualiza\_var\_global(gensym(Prefijo),N1),

concatena(Prefijo,N1,Valor), !.

# Cont.

Introducción

Antecedentes

Nociones de  
LógicaRelación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

Control

Selectores y  
Acceso a  
Predicados y  
ProgramasTodas las  
soluciones

Operadores

Estructuras  
Parciales

```
obten_valor(X,N) :- var_global(X,N), !.  
obten_valor(X,0).
```

```
concatena(X,Y,XY) :-  
    name(X,LX),  
    name(Y,LY),  
    append(LX,LX,LXY),  
    name(XY,LXY).
```

# Todas las soluciones

- Introducción
- Antecedentes
- Nociones de Lógica
- Relación con Lógica
- Sintaxis
- Listas
- Aritmética
- Algunos Ejemplos
- Control
- Selectores y Acceso a Predicados y Programas
- Todas las soluciones**
- Operadores
- Estructuras Parciales

- A veces queremos obtener todas las soluciones.
- Esto permite convertir *backtracking* en un proceso iterativo y resuelve el problema de pasar información de una iteración a la otra por medio de *backtracking*.
- Predicados: findall, bagof, setof.



# findall(Var,Pred,Lista).

Introducción

Antecedentes

Nociones de  
LógicaRelación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

Control

Selectores y  
Acceso a  
Predicados y  
ProgramasTodas las  
soluciones

Operadores

Estructuras  
Parciales

```
clases(juan,prolog,miercoles).
```

```
clases(juan,cocina,jueves).
```

```
clases(juan,tenis,viernes).
```

```
clases(maria,ingles,lunes).
```

```
clases(maria,tenis,jueves).
```

```
clases(maria,algebra,miercoles).
```

```
?- findall(Dia,clases(_,_,Dia), Dias).
```

```
Dias = [miercoles,jueves,viernes,lunes,jueves,miercoels]
```

```
?- findall(Materia,clases(_,Materia,_), Materias).
```

```
?- findall(Nombre/Materia, clases(Nombre,Materia,_),L).
```

```
?- findall(Clase,clases(juan,Clase,_),L).
```

## Cont.

Si queremos simularlo podemos guardar una lista de soluciones

```
findall(X,Meta,Sols) :-
    ( call(Meta),
      assertz(lista(X)),
      fail
    ; assertz(lista(fondo)),
      colecta(Sols)
    ).
```

```
colecta(Sols) :-
    retract(lista(X)), !,
    ( X == fondo, !, Sols = []
    ; Sols = [X|Resto],
      colecta(Resto)
    ).
```

# Cont.

Introducción

Antecedentes

Nociones de  
LógicaRelación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

Control

Selectores y  
Acceso a  
Predicados y  
Programas**Todas las  
soluciones**

Operadores

Estructuras  
Parciales

El código tiene un pequeño “error”

```
?- findall(X,member(X,[alto,ancho,fondo,peso,color,marca]),L).
```

```
L = [alto,ancho]
```

```
?- listing(lista).
```

```
lista(peso).
```

```
lista(color).
```

```
lista(marca).
```

```
lista(fondo).
```

# Posible solución

```

findall(X,Meta,Sols) :-
    ( assertz('todas las sols'([])),
      call(Meta),
      assertz('todas las sols'({X})),
      fail
    ; encuentra todas'([],Sols)
    ).

encuentra todas'(Inter,Res) :-
    retract('todas las sols'(Sol1),
    !,
    encuentra todas'(Sol1, Inter, Res).

encuentra todas'([],Res,Res).
encuentra todas'({X},Inter,Res) :-
    encuentra todas'([X|Inter],Res).
  
```

## Otra posibilidad

La otra posibilidad es usar la base de datos interna de Prolog que nos da un número de referencia único

findall(X,Meta,Sols) :-

```
( recuerda('las sols',[],MarcaRef),
  call(Meta),
  recuerda('las sols',X,-),
  fail
; encuentra todas'(MarcaRef,[],L)
).
```

encuentra todas'(MarcaRef,Inter, Sol) :-

```
recorded('las sols', X, Ref), !,
erase(Ref),
( Ref = MarcaRef - > Inter = Sol
; encuentra todas'(MarcaRef,[X|Inter],Sol)
).
```

# Bagof

```
?- bagof(Materia, clases(N,Materia,D), Mats).
```

```
N=juan,
```

```
D=jueves,
```

```
Mats=[cocina];
```

```
N=juan,
```

```
D=miercoles,
```

```
Mats=[prolog];
```

```
...
```

```
?- bagof(Materia,Dia^clases(Nombre,Materia,Dia), Mats).
```

```
Nombre = juan,
```

```
Mats = [prolog,cocina,tenis];
```

```
Nombre = maria,
```

```
Mats = [ingles,tenis,algebra]
```

# setof

- *setof* quita duplicados y ordena
- Ejemplo:  
?- setof(Materia,Dia^Nombre^  
clases(Nombre,Materia,Dia),Mats).  
Mats = [algebra,cocina,ingles,prolog,tenis]
- Otro ejemplo:  
copy(A,B) :- setof(X,A^(A=X),[B]).

## bagof (cont.)

- Con *numera\_vars* se puede hacer lo siguiente:

variantes(A,B) :-

++ (numera\_vars(A,0,N),  
numera\_vars(B,0,N),  
A = B).

verifica(X) :- ++ X.

var(X) :- ++ X = a, ++ X = b.

(i.e., lo único que instancia a “a” y “b” al mismo tiempo es una variable)

- Otro ejemplo útil:

ground(Term) :- numera\_vars(Term,0,0).



# Operadores

Introducción

Antecedentes

Nociones de  
Lógica

Relación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

Control

Selectores y  
Acceso a  
Predicados y  
Programas

Todas las  
soluciones

**Operadores**

Estructuras  
Parciales

- $\text{:- op(Numero, Forma, Nombre)}.$
- Si tenemos:  $a * b + c * d \Rightarrow + (* (a,b), *(c,d))$
- $*$ ,  $+$  son operadores “infix”

# Operadores

Introducción

Antecedentes

Nociones de  
LógicaRelación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

Control

Selectores y  
Acceso a  
Predicados y  
ProgramasTodas las  
soluciones

Operadores

Estructuras  
Parciales

- La definición (representación real) se puede ver con *display*:

$H :- T \quad \Rightarrow \text{:- (H, T).}$

$f :- g, h, i \quad \Rightarrow \text{:- (f, '(g, '(h,i)))}$

$a :- b,c,d \text{ is } 3 + 4 \quad \Rightarrow \text{:- (a, '(b, '(c, is(d, +(3,4))))}$

$[1, 2, 3, 4] \quad \Rightarrow \text{. (1, . (2, . (3, . (4, [])))}$

- Todo la sintaxis en Prolog está definida por medio de operadores.
- La ventaja que tiene es que uno puede definir sus propios operadores!!

# Ejemplo

- Por ejemplo, si quiero escribir: *bloqueA sobre bloqueB* en lugar de `sobre(bloqueA,bloqueB)`, puedo definir un operador *sobre*
- La forma de hacerlo es por medio de (a veces llamados “directivos”): `:- op(Numero,Forma,Nombre).`  
`:- op(150,xfx,sobre).`  
`:- op(200,xfx,es_un).`

`bloqueA sobre bloqueB.`  
`juan es_un hombre.`

# Operadores

El número o la precedencia dependen de la implementación, aunque normalmente está dada como:

```
:- op(1200,xfx,':-').  
:- op(1200,fx,[':-','?']).  
:- op(1100,xfy,';').  
:- op(1000,xfy,',').  
:- op(500,yfx,['+','-']).  
:- op(400,yfx,['*','/']).
```

Introducción

Antecedentes

Nociones de  
LógicaRelación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

Control

Selectores y  
Acceso a  
Predicados y  
ProgramasTodas las  
soluciones**Operadores**Estructuras  
Parciales

# Operadores

Introducción

Antecedentes

Nociones de  
LógicaRelación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

Control

Selectores y  
Acceso a  
Predicados y  
ProgramasTodas las  
soluciones**Operadores**Estructuras  
Parciales

- Tipos de operadores:  
infix       $xfx, xfy, yfx$   
prefix     $fx, fy$   
postfix    $xf, yf$   
 $f$  = operador,  $X$  y  $Y$  = argumentos
- Precedencia de argumentos encerrados en paréntesis  
= 0
- $X$  = precedencia  $<$  que el operador  
 $Y$  = precedencia  $=<$  que el operador

# Operadores

Introducción

Antecedentes

Nociones de  
LógicaRelación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

Control

Selectores y  
Acceso a  
Predicados y  
ProgramasTodas las  
soluciones

Operadores

Estructuras  
Parciales

`:- op(500,yfx,+).`

$a + b + c \quad \Rightarrow \quad (a + b) + c$

`:- op(500,xfy,+).`

$a + b + c \quad \Rightarrow \quad a + (b + c)$

$\text{not not } p \quad \Rightarrow \quad fy$

$\text{not (not } p) \quad \Rightarrow \quad fx$

- `:-'` también puede servir para consultar un programa:  
`:- [archivo1, archivo2].`

# Ejemplos

Introducción

Antecedentes

Nociones de  
LógicaRelación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

Control

Selectores y  
Acceso a  
Predicados y  
ProgramasTodas las  
soluciones**Operadores**Estructuras  
Parciales

a sobre b y b sobre c y c sobre d.

$\text{:- op}(200, \text{xfx}, \text{sobre}).$

$\text{:- op}(300, \text{xfy}, \text{y}).$

$\text{y}(\text{sobre}(\text{a}, \text{b}), \text{y}(\text{sobre}(\text{b}, \text{c}), \text{sobre}(\text{c}, \text{d}))).$

la temperatura de la caldera es 200.

$\text{:- op}(100, \text{fx}, \text{la}).$

$\text{:- op}(299, \text{xfx}, \text{de}).$

$\text{:- op}(300, \text{xfx}, \text{es}).$

$\text{es}(\text{de}(\text{la}(\text{caldera}), \text{la}(\text{temperatura})), 200).$

# Más ejemplos

Introducción

:- op(100,xfx,en).

Antecedentes

:- op(100,xfx,y).

Nociones de  
Lógica

:- op(200,xfx,da).

Relación con  
Lógica

:- op(300,fx,junta).

Sintaxis

Elem en [Elem|\_].

Listas

Elem en [\_|T] :-

Aritmética

Elem en T.

Algunos  
Ejemplos

junta [] y L da L.

Control

junta [H|L1] y L2 da [H|L3] :-

Selectores y  
Acceso a  
Predicados y  
Programas

junta L1 y L2 da L3.

Todas las  
soluciones

?- a en [1,s,a,d].

Operadores

yes

Estructuras  
Parciales

?- junta [1,2] y [3,4] da L.

L=[1,2,3,4]



# Estructuras Parciales

Introducción

Antecedentes

Nociones de  
LógicaRelación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

Control

Selectores y  
Acceso a  
Predicados y  
ProgramasTodas las  
soluciones

Operadores

Estructuras  
Parciales

- Algo “común” en Prolog es trabajar con estructuras parcialmente definidas.
- Un ejemplo donde se ve esto, es en la definición de *append*  
 $\text{append}([], L, L).$   
 $\text{append}([H|T], L, [H|R]) :-$   
 $\quad \text{append}(T, L, R).$
- El problema de esta definición es que es muy ineficiente, sobre todo si la primera lista es grande (ya que tiene que recorrerla toda antes de acabar).
- Si supieramos donde acaba la lista ésto se podría solucionar.

# EStructuras Parciales

- Lo podemos solucionar si representamos a una lista como un par: la lista en sí, y el resto.
- Es común ponerlo como diferencia de listas:  $L1 - L2$  (pero puede usarse otra notación, e.g.,  $L1/L2$ )

$$\begin{aligned}
 [1,2,3] &= [1,2,3] - [] \\
 &= [1,2,3,4,5] - [4,5] \\
 &= [1,2,3|T] - T
 \end{aligned}$$

$$L1 = A1-Z1, L2 = A2-Z2, \text{append}(L1, L2, L3):$$

$$\text{append}(A1-Z1, Z1-Z2, A1-Z2).$$

$$?- \text{append}([1,2,3|T]-T, [4,5,6|R]-R, L3).$$

$$T = [4,5,6|R]$$

$$L3 = [1,2,3,4,5,6|R]-R$$

$$2. \text{append}([a,b,c|T1]-T1, [e,f,g|T2]-T2, Z) :-$$

# Estructuras Parciales

- Si queremos añadir un elemento al final de una lista:  
 $\text{anade}(\text{Elem}, \text{Lista} - [\text{Elem} | \text{T}], \text{Lista} - \text{T}).$

?-  $\text{anade}(a, [1, 2, 3 | \text{R}] - \text{R}, \text{X}).$

$\text{R} = [a | \text{T}],$

$\text{X} = [1, 2, 3, a | \text{T}] - \text{T}$

# Estructuras Parciales

- n general en todas las definiciones en donde usabamos *append* podemos cambiarlas para usar diferencia de listas (o estructuras incompletas).

- Por ejemplo para voltear una lista:

`voltea(L,LV) :- voltea2(L-[],LV-[]).`

`voltea2(A-Z,L-L) :- A == Z, !.`

`voltea2([H|T]-Z,RL-RZ) :-  
     voltea2(T-Z,RL-[H|RZ]).`

# Mas ejemplos

- Para quicksort:

qsort2([],Z-Z).

qsort2([H|T],A1-Z2) :-

divide(H,T,Men,May),

qsort2(Men,A1-[H|Z1]),

qsort2(May,Z1-Z2).

# Aumentar Listas Dinámicamente

- Una aplicación común, es ir aumentando una estructura parcial al tener nueva información.
- Por ejemplo: vamos a suponer que tenemos un diccionario que tiene una secuencia de pares (*Llave, Valor*)

Dict = [(juan,1432), (pepe,75), (maria, 3214)|R]

?- busca(pepe,Dict,N).

N = 75

?- busca(juan,Dict,1432).

yes

?- busca(pancrasio,Dict,2816).

Dict = [(juan,1432), (pepe,75), (maria, 3214), (pancrasio,2816)|R]

# Aumentar Listas Dinámicamente

- Introducción
- Antecedentes
- Nociones de Lógica
- Relación con Lógica
- Sintaxis
- Listas
- Aritmética
- Algunos Ejemplos
- Control
- Selectores y Acceso a Predicados y Programas
- Todas las soluciones
- Operadores
- Estructuras Parciales

```
busca(Llave,[(Llave,Valor)|Dict],Valor).
busca(Llave,[(Llave1,_)|Dict],Valor) :-
    Llave = Llave1,
    busca(Llave,Dict,Valor).
```

# Estructuras Parciales

Introducción

Antecedentes

Nociones de  
Lógica

Relación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

Control

Selectores y  
Acceso a  
Predicados y  
Programas

Todas las  
soluciones

Operadores

Estructuras  
Parciales

- Las estructuras de datos incompletas también pueden usarse en otras estructuras que no sean listas.
- En particular, el ejemplo de arriba se puede mejorar si en lugar de listas utilizamos un árbol binario.
- La idea es tener del lado izquierdo del árbol elementos menores y del derecho elementos mayores que el nodo raíz.



# Estructuras Parciales

- Antes veamos como podemos representar un árbol binario:

`arbol_binario(vacio).`

`arbol_binario(arbol(Nodo,lzq,Der)) :-  
 arbol_binario(lzq),  
 arbol_binario(Der).`

`elemento_arbol(N,arbol(N,_,_)).`

`elemento_arbol(N,arbol(_,lzq,_)) :-  
 elemento_arbol(N,lzq).`

`elemento_arbol(N,arbol(_,_,Der)) :-  
 elemento_arbol(N,Der).`

# Diccionario (cont.)

Introducción

Antecedentes

Nociones de  
LógicaRelación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

Control

Selectores y  
Acceso a  
Predicados y  
ProgramasTodas las  
soluciones

Operadores

Estructuras  
Parciales

- Regresando al diccionario, el árbol lo podemos representar como sigue:  
*dict(Llave, Valor, Izq, Der).*
- Asumimos que ahora las llaves son números y los valores pueden ser lo que sea.
- En caso de que las llaves no sean números se puede user @<, @>

## Diccionario (cont.)

$\text{busca}(\text{Llave}, \text{dict}(\text{Llave}, X, \text{Izq}, \text{Der}), \text{Valor}) :-$

$!, X = \text{Valor}.$

$\text{busca}(\text{Llave}, \text{dict}(\text{Llave1}, X, \text{Izq}, \text{Der}), \text{Valor}) :-$

$\text{Llave} < \text{Llave1},$

$\text{busca}(\text{Llave}, \text{Izq}, \text{Valor}).$

$\text{busca}(\text{Llave}, \text{dict}(\text{Llave1}, X, \text{Izq}, \text{Der}), \text{Valor}) :-$

$\text{Llave} > \text{Llave1},$

$\text{busca}(\text{Llave}, \text{Der}, \text{Valor}).$

$?- \text{busca}(2, L, b), \text{busca}(1, L, a), \text{busca}(3, L, c), \text{busca}(3, L, X).$

$L = \text{dict}(2, b, \text{dict}(1, a, -, -), \text{dict}(3, c, -, -)).$

$X = c$

En el programa de arriba necesitamos el CUT por si *Llave* no está instanciada (ya que si llega a  $< o >$  marcaría un error)

# Más de I/O

Introducción

Antecedentes

Nociones de  
LógicaRelación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

Control

Selectores y  
Acceso a  
Predicados y  
ProgramasTodas las  
soluciones

Operadores

Estructuras  
Parciales

```
read(X).  write(X).  
tab(N).   nl.  
get0(C).  get(C).  put(C).
```

```
?- consult(Archivo).  
:- [Archivo]
```

```
?- reconsult(Archivo).  
:- [-Archivo].
```

# Definite Clause Grammar (DCG)

Introducción

Antecedentes

Nociones de  
LógicaRelación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

Control

Selectores y  
Acceso a  
Predicados y  
ProgramasTodas las  
soluciones

Operadores

Estructuras  
Parciales

- Algo fácil de hacer en Prolog es escribir “parsers”.
- De hecho Prolog fué usado inicialmente para procesamiento de lenguaje natural.
- Prolog tiene la facilidad de traducir directamente sus cláusulas de Horn en DCGs.

# DCGs

- Introducción
- Antecedentes
- Nociones de Lógica
- Relación con Lógica
- Sintaxis
- Listas
- Aritmética
- Algunos Ejemplos
- Control
- Selectores y Acceso a Predicados y Programas
- Todas las soluciones
- Operadores
- Estructuras Parciales

DCG son una extensión de CFG:

- Un símbolo no terminal o terminal puede ser cualquier predicado de Prolog
- Se pueden incluir condiciones extras del lado derecho
- Permite tener “,” y “!”

# DCGs

Ejemplo de CFG (solo 1 símbolo del lado izq.)

oracion  $\rightarrow$  sujeto, predicado.

sujeto  $\rightarrow$  articulo, sustantivo.

predicado  $\rightarrow$  verbo, sujeto.

predicado  $\rightarrow$  verbo.

articulo  $\rightarrow$  [el].

articulo  $\rightarrow$  [la].

sustantivo  $\rightarrow$  [manzana].

sustantivo  $\rightarrow$  [hombre].

verbo  $\rightarrow$  [come]

Introducción

Antecedentes

Nociones de  
Lógica

Relación con  
Lógica

Sintaxis

Listas

Aritmética

Algunos  
Ejemplos

Control

Selectores y  
Acceso a  
Predicados y  
Programas

Todas las  
soluciones

Operadores

Estructuras  
Parciales

# DCGs

- La primera regla dice que una oración está compuesta por un sujeto seguida de un predicado.
- En Prolog podríamos “parsear” una oración, donde los símbolos terminales simplemente tienen lo que aceptan y los demás símbolos son listas con frases que aceptan.
- Una forma simple (pero ineficiente) de hacerlo es:



# DCGs (cont.)

oracion(Lista) :-  
    sujeto(Suj),  
    predicado(Pred),  
    append(Suj,Pred,Lista).

sujeto(Suj) :-  
    articulo(Art),  
    sustantivo(Sust),  
    append(Art,Sust,Suj).

articulo([el]).  
articulo([la]).

## DCGs (cont.)

- sto se puede mejorar, eliminando el “append” si cada parte de la oración nos dice cuanto ocupa:

?- oracion([el,hombre,come,la,manzana],[,]).

oracion(L1,L3)) :- sujeto(L1,L2), predicado(L2,L3).

sujeto(L1,L3) :- articulo(L1,L2), sustantivo(L2,L3).

predicado(L1,L3) :- verbo(L1,L2), sujeto(L2,L3).

predicado(L1,L2) :- verbo(L1,L2).

articulo([el|T],T).

articulo([la|T],T).

sustantivo([hombre|T],T).

sustantivo([manzana|T],T).

verbo([come|T],T).

## DCGs (cont.)

- Hay que notar que lo mismo se puede representar (casi igual) como diferencia de listas, i.e., `oracion(L1-L3) :- ...`
- Prolog tiene una forma de traducir la representación de arriba en notación de CFG + le dá facilidad de incluir argumentos y llamar a predicados de Prolog.
- Esto es, nosotros especificamos como un CFG y se traduce al código de arriba.
- Para hacer preguntas hay que recordar que se hizo la traducción, i.e.:  
?- `oracion([el,hombre,come,la,manzana],[,])`.

# Phrase

- Algunos Prolog (no todos) tienen “phrase”:  
?- phrase(oracion,[el,hombre,come,la,manzana]).  
?- phrase(sujeto,[la,manzana]).
- En la notación, se permiten incluir llamados entre corchetes.
- También podemos añadir más argumentos.

# TITULO

- Por ejemplo, si queremos guardar el árbol que genera el “parser”, podemos poner:

oracion(oracion(S,P)) -- > sujeto(S) predicado(P).

sujeto(suj(A,S)) -- > articulo(A), sustantivo(S).

predicado(pred(V,S)) -- > verbo(V), sujeto(S).

predicado(pred(V)) -- > verbo(V).

articulo(art(el)) -- > [el].

...

?- oracion(Arbol,[el,hombre,come,la,manzana],[]).

Arbol = oracion(suj(art(el),  
                  sust(hombre)),  
                  pred(verb(come),  
                      suj(art(la),  
                      sust(manzana)))).

# DCGs (cont.)

- Algo común es revisar por género y número:  
`oracion -- > sujeto(Num), predicado(Num).`  
`sujeto(N) -- > articulo(Num,Gen), sustantivo(Num,Gen).`  
`predicado(Num) -- > verbo(Num), sujeto(_).`  
`articulo(sing,fem) -- > [la].`  
`articulo(sing,masc) -- > [el].`  
`articulo(plural,fem) -- > [las].`  
`articulo(plural,masc) -- > [los].`  
`verbo(sing) -- > [come].`  
`verbo(plural) -- > [comen].`  
`sustantivo(sing,masc) -- > [hombre].`  
`sustantivo(sing,fem) -- > [manzana].`

# DCGs (cont.)

- Para “parsear” una expresión matemática y regresar su resultado:

$\text{expr}(Z) \text{ --- } > \text{term}(X), \text{“+”}, \text{expr}(Y), Z \text{ is } X + Y.$

$\text{expr}(Z) \text{ --- } > \text{term}(X), \text{“-”}, \text{expr}(Y), Z \text{ is } X - Y.$

$\text{expr}(Z) \text{ --- } > \text{term}(X).$

$\text{term}(Z) \text{ --- } > \text{numero}(X), \text{“*”}, \text{term}(Y), Z \text{ is } X * Y.$

$\text{term}(Z) \text{ --- } > \text{numero}(X), \text{“/”}, \text{term}(Y), Z \text{ is } X / Y.$

$\text{term}(Z) \text{ --- } > \text{numero}(Z).$

$\text{numero}(N) \text{ --- } > \text{“+”}, \text{numero}(N).$

$\text{numero}(N) \text{ --- } > \text{“-”}, \text{numero}(M), N \text{ is } - M.$

$\text{numero}(N) \text{ --- } > [X], 48 = < X, X = < 57, N \text{ is } X - 48.$

## DCGs (cont.)

- Alternativamente, se puede poner la expresion en un lista, todos los que están entre doble comillas, cambian a elementos en listas, e.g., “+” => [+], y la última de número cambia:

`numero(N) -- > [N], member(N,[0,1,2,3,4,5,6,7,8,9]).`

Con DCG podemos hacer más que con CFG (e.g.,  $a^n b^n c^n$ )

`s -- > [a], s(i).`

`s(l) -- > [a], s(i(l)).`

`s(l) -- > nb(l), nc(l).`

`nb(i(l)) -- > [b], nb(l).`

`nb(i) -- > [b].`

`nc(i(l)) -- > [c], nc(l).`

`nc(i) -- > [c].`