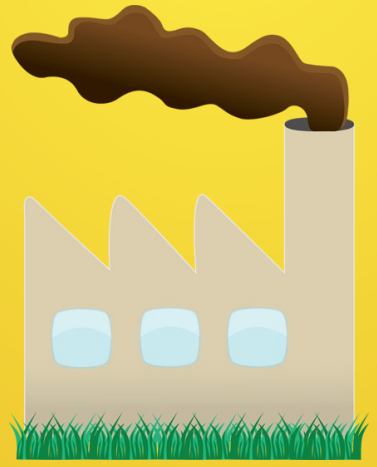
Prolog...

Luis Casillas
DCC, UdeG, CUCEI
Diciembre, 2015

Origen

- Su implementación se basa en el *Teorema de Resolución de Robinson (1965)*. Es un procedimiento sintético para probar teoremas.
- Prolog surge en Francia a principios de los años 70, como un medio para Lenguaje Natural. *Colmerauer, Kanoui et al. (1973)*



Semántica

- El fracaso de Prolog en el proceso de Lenguaje Natural (LN), *es la causa de su éxito posterior*.
- Al eliminar las fuentes de ambigüedad propias del LN, *sólo queda la parte lógica*.
- Utiliza un esquema basado en **cálculo de predicados de primer orden**, el cual provee: *formalismo, flexibilidad y robustez*.



Implementación

- Basado en **Programación de la Lógica**, Prolog ha hecho importantes contribuciones a la IA.
- El **Cálculo de Predicados** es un esquema central para *Representar Conocimiento*.
- La **Semántica Declarativa** de Prolog permite modelar relaciones entre entidades.

Niveles SBC

Newell (1982)

Bondades$_1$

- El cálculo de predicados de primer orden es un metalenguaje propuesto por David Hilbert 1920~. Cuenta con gran *poder representacional*.
- Así, Prolog puede expresar relaciones generales entre entidades.
- *“Todas las mujeres son inteligentes”* en lugar de las proposiciones aisladas: *“Lunia es inteligente”*, *“Martha es inteligente”*, *“María es inteligente”* y así...

Bondades$_2$

- Prolog utiliza **conocimiento declarativo**.
- Posee una máquina de unificación que se encarga del ***pareamiento* dinámico de patrones**.
- Cuenta con sistema de búsqueda **primero profundidad** de **izquierda a derecha**.
- Estos aspectos favorecen la **expresión y gestión de relaciones IA** de una forma más directa y natural. Asimismo tiene un gran potencial de resolución.

Sintaxis

- Aunque existen varios dialectos de Prolog, algunos aspectos son consistentes:

Español	Cálculo de Predicados	Prolog
y	**∧**	**,**
o	**∨**	**;**
Sólo si	**¬**	**:-**
no	**~**	not

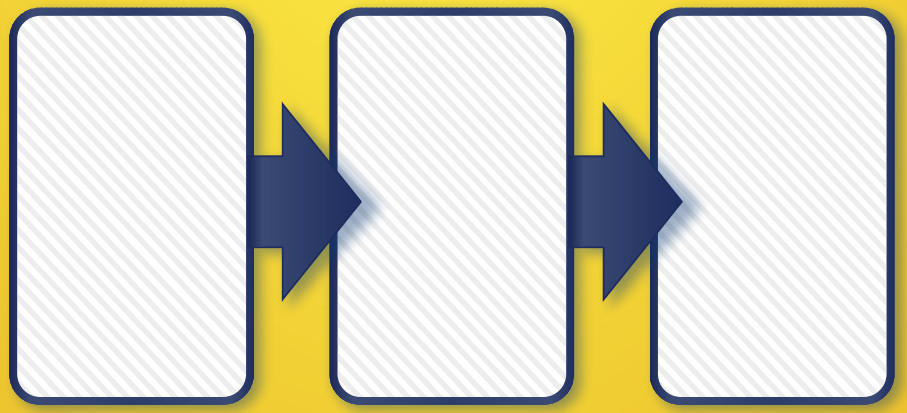
Términos

- Todo Prolog es definido por términos, tanto:
  - Los elementos constructivos del programa
  - Los datos y otras estructuras que el programa usa
- Clases de término:
  - Constantes: enteros, números reales y átomos
  - Variables
  - Términos compuestos




Unificación

- En Prolog el “Pareamiento de Patrones” es llevado adelante por medio del principio de **Unificación**.
- Dos términos se **unifican** si existe alguna manera de vincular sus variables para hacerlos idénticos.
- Por ejemplo, los términos: **`agrada(juan, Alguien)`** y **`agrada(juan, lucía)`** se unifican si la variable **`Alguien`** es vinculada a **`lucía`**.



Base de Datos Prolog

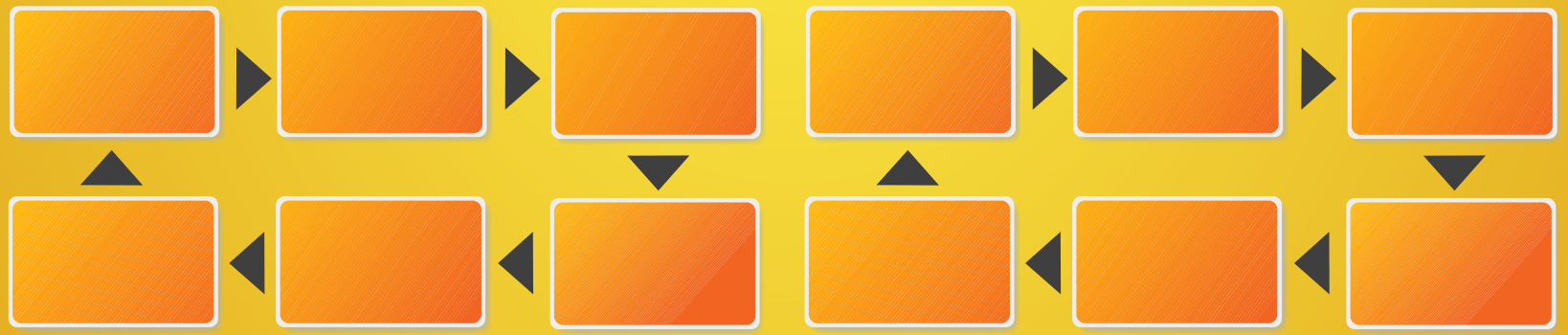
- Puede incluso ser entendida como una **Base de Conocimiento**, considerando la semántica de los contenidos.
- Es un ensamble de **hechos** y **reglas** de inferencia. Este ensamble constituye ***per se*** un programa Prolog.
- *La BD cambia gradualmente mientras el programa es realizado*. Y en la medida que cambia la BD, cambia también la manera de responder del propio programa. **Se auto-ajusta**.

Reglas

- Indican la forma es que se probará algo. Para probar la cabeza (conclusión), probar el cuerpo (condición):

```
Cabeza:-Condición.
```

- Una regla constituye una generalización que modela a un grupo de casos específicos, que de otro modo demandarían una lista de hechos aislados. La regla les agrupa.
- Así, un programa es un conjunto de cláusulas. Las cláusulas pueden ser hechos o reglas, terminando en punto “ . ”



Reglas: Enfoque Declarativo

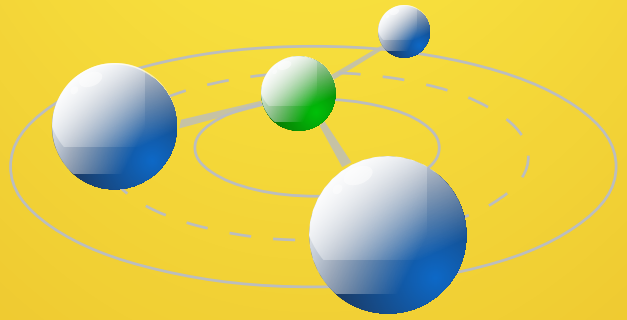
∀x∀y(agrada(X,Y)∧agrada(Y,X)→amigos(X,Y))

- Una regla es una afirmación lógica.
  - Para todo enlace de **X** y **Y** con elementos del universo de discurso (**U**), si **agrada(X,Y)** y **agrada(Y,X),** entonces **amigos(X,Y).**
- Se trata sólo de una afirmación, un formulismo que **NO** indica **cómo** hacer algo.

Reglas: Enfoque Procedimental

```
amigos(X,Y):-agrada(X,Y),agrada(Y,X).
```

- Para probar **`amigos(X,Y)`** es preciso **encontrar** algún **`X`** y algún **`Y`** para los cuales se pueda verificar que **`agrada(X,Y)`** y **`agrada(Y,X)`**
- Un programa en Prolog especifica procedimientos de prueba para las consultas. Básicamente se trata de **búsquedas en profundidad de izquierda a derecha**.



Uso 1

En Prolog los nombres para **Predicados** y **Variables** vinculadas, son secuencias de caracteres alfanuméricos.

<nom_predicado>::=<letra_min>[<sec_cv>]

<nom_variable>::=<letra_may>[<sec_cv>]

<sec_cv>::=(<cv>|<cv><sec_cv>)

<cv>::=(<letra_min>|<letra_may>|<dig>|_)

<letra_min>::=(a|b|c|...|z)

<letra_may>::=(A|B|C|...|Z)

<dig>::=(0|1|2|3|4|5|6|7|8|9)

Uso$_2$

Los predicados son expresiones que pueden ser evaluadas como Verdaderas o Falsas. Normalmente reciben términos como argumentos, aunque podrían no tener argumentos. Un término es una Variable o un objeto concreto del U.

<predicado>::=(<nom_predicado>['('<lista_terms>')']|<compn>)

<lista_terms>::=(<term>|<term>,<lista_terms>)

<term>::=(<nom_variable>|<objeto_U>|<termCompuesto>)

<objeto_U>::=(<letra_min>[<sec_cv>]|<número>)

<termCompuesto>::=(<nom_predicado>'('<lista_terms>')'

<número>::=(<entero>|<flotante>)

<compn>::=<term><opr_cmp><term>

<opr_cmp>::=(=|\==|<|>|>=|=<)

Uso 3

El conocimiento declarativo va presentándose a Prolog como una sucesión de Hechos y Reglas. Los **hechos** describen la situación actual de los objetos concretos del U que se representa. Las **reglas** describen de manera general las relaciones entre objetos del U.

<hecho>::=<nom_predicado>'('<lista_objsU>')'.

<lista_objsU>::=(<objeto_U>|<objeto_U>,<lista_objsU>)

<regla>::=<predicado_conc>:-<prueba_log>.

<predicado_conc>::=<nom_predicado>'('<lista_pars>')'

<lista_pars>::=(<nom_variable>|<nom_variable><lista_pars>)

<prueba_log>::=(<predicado>|<asignación>|!|fail|

not'('<prueba_log>')'|'('<prueba_log>')'|

<prueba_log><opr_log_bin><prueba_log>)

<opr_log_bin>::=(,|;)

<asignación>::=<nom_variable> is <objeto_U>

Uso 4

Resta modelar la realidad. Una forma de hacerlo es la **Conceptualización**:

U: **Personas**

R: **agrada(X,Y)**: X siente agrado por Y

amigos(X,Y): X y Y son amigos

amistad.pl

```
%Hechos:

agrada(juan, lucía).

agrada(rodolfo, telma).

agrada(esmeralda, pablo).

agrada(lucía, rodolfo).

agrada(pablo, esmeralda).

%Regla:

amigos(X,Y):-agrada(X,Y),agrada(Y,X).
```

Uso 5

amistad.pl (~/swi-prologs

Abrir ▾ Guardar

amistad.pl ×

```
%Hechos:
agrada(juan, lucía).
agrada(rodolfo, telma).
agrada(esmeralda, pablo).
agrada(lucía, rodolfo).
agrada(pablo, esmeralda).
%Regla:
amigos(X,Y):-agrada(X,Y),
             agrada(Y,X).
```

Prolog ▾ Anchura de la pe

zaratustra@Zaratustra-Athlon: ~/swi-prologs

```
zaratustra@Zaratustra-Athlon:~/swi-prologs$ swipl
Welcome to SWI-Prolog (Multi-threaded, 64 bits, Version 7.2.3)
Copyright (c) 1990-2015 University of Amsterdam, VU Amsterdam
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to redistribute it under certain conditions.
Please visit http://www.swi-prolog.org for details.

For help, use ?- help(Topic). or ?- apropos(Word).

?- [amistad].
true.

?- agrada(rodolfo,telma).
true.

?- agrada(telma,rodolfo).
false.

?- agrada(Q1,Q2).
Q1 = juan,
Q2 = lucía ;
Q1 = rodolfo,
Q2 = telma ;
Q1 = esmeralda,
Q2 = pablo ;
Q1 = lucía,
Q2 = rodolfo ;
Q1 = pablo,
Q2 = esmeralda.

?- amigos(Q1,Q2).
Q1 = esmeralda,
Q2 = pablo ;
Q1 = pablo,
Q2 = esmeralda.

?-
```

Gracias!

Dudas... Comentarios... ?

luis.casillas@cucei.udg.mx