Introducción

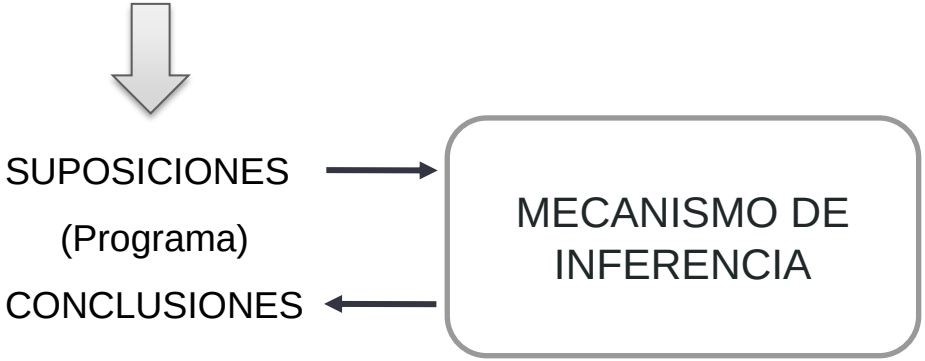
- Aunque los lenguajes imperativos son los más utilizados, otros paradigmas se adaptan mejor a la solución de ciertos problemas.
- Un ejemplo es Prolog, este lenguaje pertenece al paradigma lógico.
- Prolog fue creado en 1972 por Colmerauer y Roussel, y se adapta mejor para problemas donde es necesario describir el conocimiento, por ejemplo:
  - en aplicaciones que realizan computación simbólica;
  - comprensión del lenguaje natural;
  - en sistemas expertos.

Programación Lógica

- ¿Qué es?
  - **Formalismo** computacional que combina dos principios centrales:
    - Uso de **lógica** para expresar conocimiento
    - Uso de **inferencia** para manipular conocimiento
  - Objetivo general: inferir conclusiones a partir de algunas suposiciones.

Programación Lógica

- Estudiaremos formas para representar el conocimiento
- Pues es así que vamos a “programar”



Puntos a ponderar

- En principio, necesitamos apenas estudiar lógica
  - Para especificar lo que deseamos
- La máquina se preocupa en resolver
  - Como inferir conclusiones
- Es la belleza del paradigma
- Sin embargo, en la practica:
  - Políticas de implementación
  - Implementaciones ineficientes
  - Problemas intratables
- Requieren conocimiento del mecanismo de inferencia
  - O sea, ud. debe escribir programas de una cierta “forma” para obtener respuestas.
- Requieren primitivas no lógica

Puntos a ponderar

- Existe también otra cuestión importante
  - Aquello que expresamos en los programas no es el mundo real
    - Y si apenas parte de él
  - Pero el computador no puede considerar supuestos que no hayan sido definidos en los programas
    - Por lo tanto, es preciso cuidado al especificar los programas
      - Para no “dejar nada fuera”.

Introducción

- Existen varias implementaciones de Prolog, algunas de las más conocidas son:
- Win-Prolog;
- Ciao Prolog;
- YAP Prolog;
- SWI Prolog + SWI-Prolog-Editor (recomendado);
  - www.swi-prolog.org
  - https://sites.google.com/site/paradigmasdeprogramacion/software/editores/swi-prolog-editor
- SICStus Prolog

Sintaxis de Prolog

- Los datos representados en Prolog pueden ser de uno de los siguientes tipos:
- variables: comienzan con mayúsculas o guiones bajos (_), seguidos de cualquier carácter alfanumérico. Solo el guión bajo define una variable anónima.
- Ej: X, Y1, _Nombre, ...;
- átomos - son constantes, deben inicializarse con minúsculas seguidas de cualquier carácter alfanumérico o cualquier secuencia entre ' ' (comillas simples).
- Ej.: juan, 'Juan', '16', ...;
- enteros: cualquier secuencia numérica que no contenga un punto (.). Los caracteres ASCII entre “ ” (comillas dobles) se tratan como listas de números enteros. Ej.: 1, 6, -3, “a”,...;
- floats: números con un punto (.) y al menos un lugar decimal.
- Ej.: 5.3 (correcto), 7. (incorrecto);
- listas: secuencia ordenada de elementos entre [ ] y separados por comas. Ej: [a, b, c], [a | b, c].

Sintaxis de Prolog

- Clasifique los términos abajo como átomo variable, número, lista o invalido.

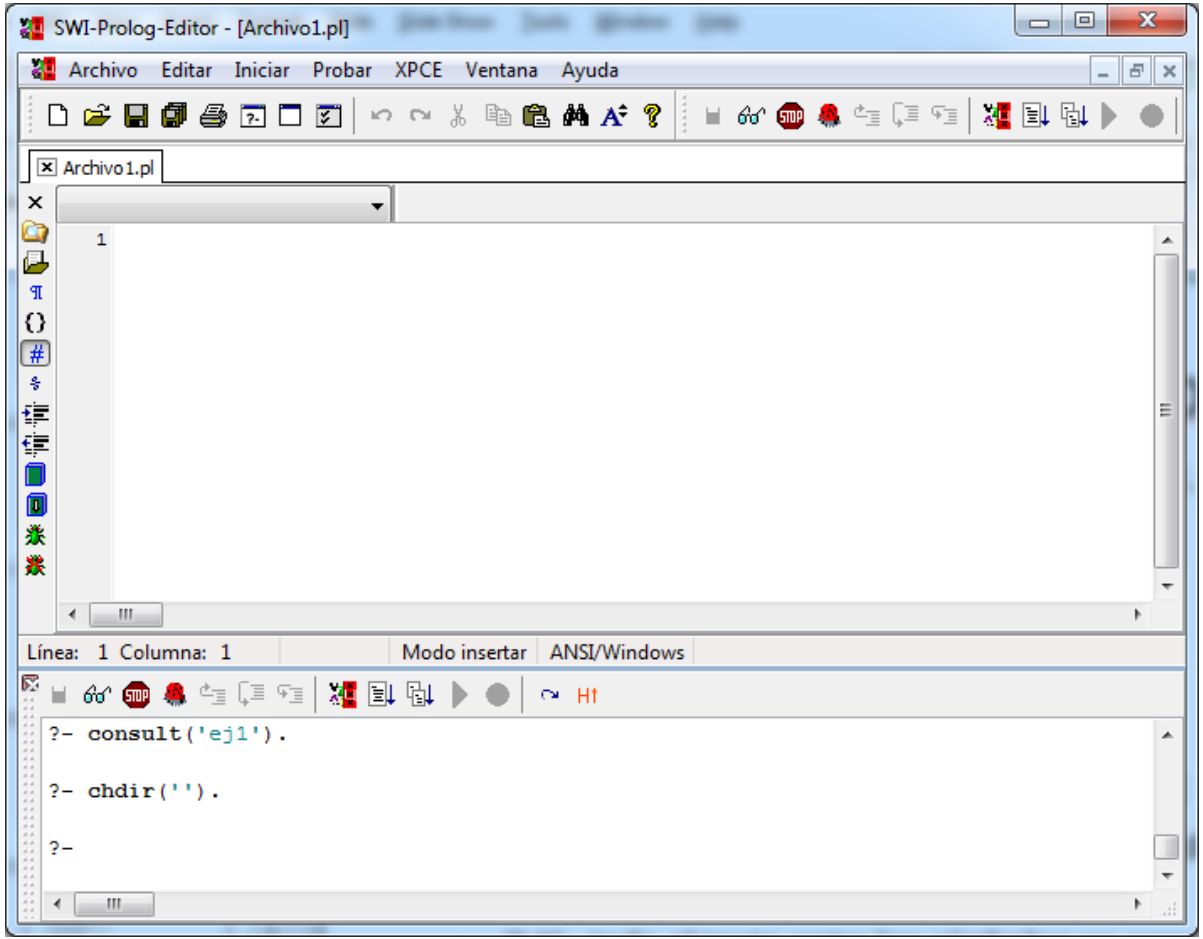
Término	Clasificación	Término	Clasificación
vINCENT		Footmassage	
23		65.	
variable23		23.0	
clases de lógica		_	
‘Juan’		‘clases de lógica’	
[1, [2, 3], 4]		“a”	
		[]	

Sintaxis de Prolog

- Escribir y leer comandos writey read sobre archivos estándar (monitor y teclado).
- Para escribir, simplemente use el comando write(+termino):
  - write('Impresión de prueba.'). (Correcto)
  - write(‘impresión de prueba). (Incorrecto)
  - write(X). (Correcto)
  - write(juan). (Correcto)
- Para leer, utilice el comando read(+var):
  - read(X). (Correcto)
  - read(x). (Incorrecto)
  - read(Juan). (Correcto)
  - read(juan). (Incorrecto)

Sintaxis de Prolog

- Algunos caracteres son especiales para imprimir, son:
  - nl, \n, \l - nueva línea.
  - \r - vuelve al principio de la línea;
  - \t - tabulador;
  - \% - imprime el símbolo %;
- Existen dos tipos de comentarios en Prolog, son:
  - %: todo el texto existente en la misma línea después del símbolo se considera un comentario;
  - /* */ - todo el texto entre los símbolos se considera un comentario.



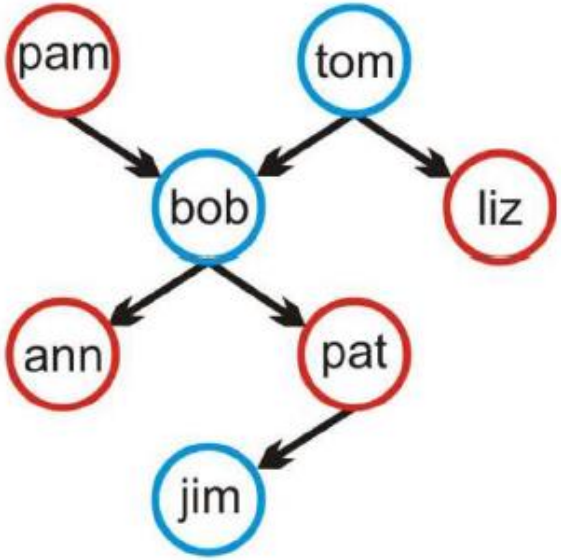
- Para tener el primer contacto con la herramienta, revisemos las respuestas del Ejercicio 1 usando los comandos a continuación en el prompt (2):
- atom(+término) - comprueba si el término es un átomo;
- var(+término) - comprueba si el término es una variable;
- número(+término) - comprueba si el término es un número (entero o float);
- is_list(+término) - comprueba si el término es una lista;
- Luego vamos a digitar siguiente comando en la base de conocimiento (1), cargar y ejecutar el código (3), finalmente, hacer la consulta en (4).
- start() :-
- write('Ingrese el valor de X:'), nl,
- read(X), nl,
- write(X), nl.

Hechos

- Un programa Prolog es una colección de hechos y reglas.
- Los hechos siempre son ciertos, pero las reglas necesitan ser evaluadas.
- Cómo crear un hecho en una base Prolog:
  - hombre(x). - significa que "x es un hombre";
  - padre(x, y). - significa que "x es el padre de y" o "y es el hijo de x";
- Es responsabilidad del programador definir correctamente los predicados.

Hechos- Ejercicio 2

- Define los hechos que describen la siguiente imagen:



Consultas

La cláusula proximo(Perú, Ecuador). es una consulta Prolog, porque "Perú" y “Ecuador” son variables.
Para responder consultas Prolog utiliza:

- matching : verifica si un cierto patrón está presente, para saber qué hechos y reglas se pueden usar;
- unificación - reemplaza el valor de las variables para determinar si la consulta es satisfecha por los hechos o reglas de la base (programa);
- resolución - comprueba si una consulta es una consecuencia lógica de los hechos y reglas de la base (programa);
- recursividad: utiliza reglas que se llaman a sí mismas para realizar pruebas;
- backtracking - para verificar todas las respuestas posibles.

Consultas

Ejemplo de las etapas en la consulta...

```
hombre(tom). % hecho
mujer(pam). % hecho
padre(pam, bob). % hecho
padre(tom, bob). % hecho
?- padre(tom, X). % ¿Tom es el padre de quién (X)?
X = bob ;
No
```

<– Unifica [X/bob]
<– padre(tom, bob). (matching)
<– busca otras soluciones (backtrack)
<– Ningún otro hecho satisface la consulta.

?- genitor(X, bob). %Quién (X) es/son lo(s) padres(es) de bob?
X = tom ;
X = pam ;
No

<– Unifica [X/tom]
<– padre(tom, bob). (matching)
<– busca otras soluciones (backtrack).
<– Unifica [X/pam]
<– padre(pam, bob). (matching)
<– busca otras soluciones (backtrack).
<– Ningun otro hecho satisface la consulta.

Consultas

Dada la base de conocimientos abajo, ¿cuáles son las respuestas para las consultas?

```
animal (oso).
animal (pez).
animal (pececito).
animal (lince).
animal (zorro).
animal (conejo).
animal (ciervo).
animal (mapache).
planta (algas).
planta (hierba).
comer (oso, pez).
comer (lince, ciervo).
comer (oso, zorro).
comer (oso, ciervo).
comer (pescado, pececito).
comer (pececitos, algas).
comer (mapache, pescado).
comer (zorro, conejo).
comer (conejo, hierba).
comer (ciervo, hierba).
comer (oso, mapache).
```

```
?- planta(X).
?- come(zorro,_).
?- come(_,_).
?- come(X, hierba).
```

Reglas

- Las reglas facilitan la ejecución de consultas y hacen que un programa sea mucho más expresivo.
- Una cláusula de Prolog es equivalente a una fórmula en lógica de primer orden, por lo que en Prolog existen los conectores:

:- (si), equivalente a la implicación;

, (e), equivalente a la conjunción;

; (o), equivalente a disyunción.

Ejemplo:

La fórmula: $A(x) \leftarrow B(x) \vee (C(x) \wedge D(x))$, se escribiría en Prolog como:

A(X) :- B(X); (C(x), D(x)).

Prolog no usa cuantificadores explícitamente, sin embargo, trata todas las reglas como si estuvieran universalmente cuantificadas y usa EU (eliminación de lo universal).

Reglas

Las consultas sobre las reglas se realizan de la misma forma que sobre los hechos.

Una regla se divide en conclusión (o cabeza) y condición, de la siguiente manera:

CONCLUSIÓN(+ARG) :- CONDICIÓN1(+ARG) CONECTIVO CONDICIÓN2(+ARG) ...

Usando matching, Prolog encuentra qué reglas se pueden usar para satisfacer una consulta. Cada vez que se produce un matching, la satisfacción de la regla se convierte en la meta actual.

Regla: descendiente(X,Y) :- progrenitor(Y,X)
Consulta: ?- descendiente(pam, bob)

Consultas

a)¿Cuál es la respuesta a las consultas?

?- descendiente(_, tom).

?- descendente(tom, X).

b) Describe las reglas para las siguientes relaciones:

madre(x,y), que significa “x es madre de y”;

abuelos(X,Z), que significa “x es el abuelo de y”.