Práctica #4: Introducción al Paradigma Lógico con Prolog

Montes Solis Kimberly

✦ Instalación del Entorno de Desarrollo e Introducción a Prolog

Para comenzar con Prolog se recomienda usar **SWI-Prolog**, un entorno gratuito y de código abierto.

🌀 Se puede descargar desde: https://www.swi-prolog.org/download/stable

Una vez instalado, iniciamos la consola de Prolog y comenzamos a escribir hechos, reglas y consultas.

✦ Programación en Prolog

⤷ Hechos

Un hecho representa una afirmación básica:

```
cat(tom).
loves_to_eat(jorge, pasta).
lazy(juan).
```

⤷ Reglas

Las reglas se definen con `:-` y permiten inferir información:

```
happy(lili) :- dances(lili).
goToPlay(ryan) :- isClosed(school), free(ryan).
```

⤷ Consultas

Se realizan para verificar hechos o inferir conocimiento:

```
?- cat(tom).
?- happy(lili).
```

✦ Aplicaciones con Prolog

⋆.˚ Base de Conocimientos

Prolog permite modelar relaciones complejas entre objetos:

```
parent(pam, bob).
male(bob).
female(pam).
mother(X, Y) :- parent(X, Y), female(X).
```

También se pueden definir relaciones más complejas como:

```
brother(X,Y) :- parent(Z,X), parent(Z,Y), male(X), male(Y), X \== Y.
```

⋆.˚ Recursión y Árbol Genealógico

```
predecessor(X, Z) :- parent(X, Z).
predecessor(X, Z) :- parent(X, Y), predecessor(Y, Z).
```

⋆.˚ Operaciones Matemáticas

```
calc :- X is 100 + 200, write(X).
```

⋆.˚ Bucles y Decisiones

```
count_to_10(10) :- write(10).
count_to_10(X) :- write(X), Y is X + 1, count_to_10(Y).
```

```
gte(X,Y) :- X > Y, write('X is greater').
gte(X,Y) :- X =:= Y, write('X and Y are same').
```

✦ Listas y Manipulación

Prolog ofrece potentes mecanismos para manipular listas:

```
list_member(X, [X|_]).
list_member(X, [_|TAIL]) :- list_member(X, TAIL).
```

✦ Entrada/Salida y Archivos

Puedes manipular entradas/salidas y trabajar con archivos usando predicados como `read/1`, `write/1`, etc.

✦ Aplicaciones del Paradigma Lógico y Prolog

El paradigma lógico, sobre el que se basa Prolog, se enfoca en la representación del conocimiento mediante hechos y reglas, permitiendo inferencias automáticas. Esta capacidad lo hace ideal en múltiples áreas de aplicación.

⋆.˚ Inteligencia Artificial

Prolog se emplea en la creación de sistemas inteligentes como chatbots, motores de inferencia o agentes inteligentes.

Ejemplo: Diagnóstico Médico

```
symptom(jose, fever).
symptom(jose, cough).
disease(jose, flu) :- symptom(jose, fever), symptom(jose, cough).
```

⋆.˚ Sistemas Expertos

Ideal para construir sistemas que emulan la toma de decisiones de expertos humanos.

Ejemplo: Asistente Legal

```
contract_type(written).
enforceable(X) :- contract_type(X), X == written.
```

⋆.˚ Procesamiento de Lenguaje Natural (PLN)

Útil para crear analizadores gramaticales, traductores automáticos y sistemas de comprensión del lenguaje.

Ejemplo: Reglas Gramaticales

```
sentence --> noun_phrase, verb_phrase.
noun_phrase --> [the], noun.
verb_phrase --> verb, noun_phrase.
noun --> [cat]; [dog].
verb --> [chased]; [saw].
```

⋆.˚ Juegos y Resolución de Problemas

Prolog se utiliza en la creación de motores lógicos para resolver rompecabezas, juegos de estrategia, y más.

Ejemplo: Movimiento del Rey (ajedrez)

```
legal_move(king, X1-Y1, X2-Y2) :-
  DX is abs(X1 - X2), DY is abs(Y1 - Y2),
  DX =< 1, DY =< 1.
```

⋆.˚ Robótica y Planificación

Prolog es usado en la planificación de tareas y rutas en entornos dinámicos.

Ejemplo: Planificación de Ruta

```
connected(a, b).
connected(b, c).
path(X, Y) :- connected(X, Y).
path(X, Y) :- connected(X, Z), path(Z, Y).
```

⋆◇ Conclusión

Prolog es un lenguaje poderoso para la lógica simbólica, relaciones y estructuras de conocimiento. Su capacidad para realizar inferencias lo hace ideal en áreas como **IA, sistemas expertos y árboles genealógicos**. La sintaxis declarativa y la potencia del motor lógico permiten resolver problemas complejos de manera elegante.