+++ draft = false date = 2025-05-9 title = 'Práctica 4: Paradigma Lógico' summary = 'Edgar Rafael Garcia Ceseña' +++

Introducción

El paradigma lógico es un estilo de programación basado en la lógica matemática. En lugar de dar instrucciones paso a paso, el programador define hechos y reglas, y el programa resuelve problemas haciendo inferencias lógicas. Un ejemplo típico es Prolog, donde defines relaciones y el lenguaje encuentra respuestas a partir de ellas. Es útil en inteligencia artificial y sistemas expertos.

¿Qué es Prolog y cuáles son sus características principales?
Prolog (Programming in Logic) es el lenguaje más representativo del paradigma lógico. Sus características incluyen: uso de hechos y reglas, evaluación mediante un motor de inferencia, soporte nativo para recursividad, unificación de variables y backtracking automático. Es especialmente útil para problemas de inteligencia artificial, sistemas expertos y procesamiento de lenguaje natural.

¿Cómo funciona la ejecución de programas en Prolog?
La ejecución sigue un modelo de búsqueda con backtracking: el intérprete intenta satisfacer los objetivos buscando hechos que coincidan o aplicando reglas que permitan derivar nuevos hechos. Cuando encuentra un camino que satisface todas las condiciones, devuelve la solución; si falla, retrocede (backtrack) para explorar alternativas.

¿Qué ventajas ofrece el paradigma lógico?
Entre sus principales ventajas destacan la capacidad para manejar conocimiento declarativo, la facilidad para representar relaciones complejas, la implementación natural de algoritmos de búsqueda, y su adecuación para problemas donde las relaciones entre datos son más importantes que los cálculos numéricos.

En esta práctica se explicarán programas básicos del paradigma lógico en Prolog.

Desarrollo

Programas:

En esta sesión se analizaron 5 programas fundamentales para comprender este paradigma. A continuación se muestran los programas con su explicación.

1. Torres de Hanoi

```
move(1,X,Y,_) :-   write('Move top disk from '), write(X), write(' to '),
                    write(Y), nl.
move(N,X,Y,Z) :-   N>1,   M is N-1,   move(M,X,Z,Y),   move(1,X,Y,_),
move(M,Z,Y,X).
```

Explicación:
Este código utiliza recursividad para resolver el problema clásico:

- Si hay un solo disco, lo mueve directamente al destino
- Para N discos: mueve N-1 discos a la torre auxiliar, mueve el disco base al destino, y luego mueve los N-1 discos al destino

2. Mínimo y Máximo

```
find_max(X, Y, X) :- X >= Y, !.
find_max(X, Y, Y) :- X < Y.
find_min(X, Y, X) :- X =< Y, !.
find_min(X, Y, Y) :- X > Y.
```

Características:

- `find_max/3`: Devuelve el mayor de dos números
- `find_min/3`: Devuelve el menor de dos números
- El corte (`!`) evite backtracking innecesario

3. Segmento de Recta

```
vertical(seg(point(X,_),point(X,_))).
horizontal(seg(point(_,Y),point(_,Y))).
oblique(seg(point(X1,Y1),point(X2,Y2))) :-
    X1 \== X2, Y1 \== Y2.
```

Clasificación:

- `vertical`: Misma coordenada X
- `horizontal`: Misma coordenada Y
- `oblique`: Diferencias en X e Y

4. El Mono y el Plátano

```
move(state(middle,onbox,middle,hasnot), grasp, state(middle,onbox,middle,has)).
move(state(P,onfloor,P,H), climb, state(P,onbox,P,H)).
move(state(P1,onfloor,P1,H), drag(P1,P2), state(P2,onfloor,P2,H)).
move(state(P1,onfloor,B,H), walk(P1,P2), state(P2,onfloor,B,H)).
canget(state(_,_,_,has)).
canget(State1) :- move(State1,_,State2), canget(State2).
```

Lógica del problema:

- Modela los estados posibles (posición, postura, caja, plátano)
- Define acciones: `grasp`, `climb`, `drag`, `walk`
- Usa recursión para buscar soluciones válidas

5. Lista Enlazada

```
add_front(L,E,NList) :- NList = node(E,L).
add_back(nil, E, NList) :- NList = node(E,nil).
add_back(node(Head,Tail), E, NList) :-
    add_back(Tail, E, NewTail),
    NList = node(Head,NewTail).
```

Operaciones:

- `add_front`: Inserta al inicio (O(1))
- `add_back`: Inserta al final (O(n))
- Estructura: `node(Elemento, Siguiente)`

Conclusión

Esta práctica mostró las ventajas del paradigma lógico usando Prolog. Los ejercicios demostraron su enfoque declarativo para resolver problemas mediante reglas y recursividad, destacando su utilidad para modelar relaciones complejas. Aunque requiere un cambio de mentalidad, Prolog resulta poderoso para problemas donde las relaciones entre datos son más importantes que los pasos de solución.

Links

REPOSITORIO

PAGINA