Tarea Independiente 20/10/2025

David Núñez Franco

October 23, 2025

Inventario de Conceptos Claves

- Control de ejecución: por recursión, backtracking o metapredicados
- Ejecución de un .pl (de arriba hacia abajo, de izquierda a derecha)
- Datos: por medio de estructuras simbólicas (ASTs)
- Términos como ASTs, operadores como símbolos
- Simbólico por default, evaluación explícita: predicado is
- Algoritmo de Unificación: solución de ecuaciones de ASTs
- Listas y ASTs
- Reglas y patrones recursivos y de backtracking comunes

Ejercicio 0

```
En cada caso, resuelva la ecuación unificando, y encuentre el unificador o indique por

a) f([X, h(W) | [h(Y)]]) = f([b, h(h(a)), h(W)])

R/ Unifica. Igualando listas:

X = b
h(W) = h(h(a)) -> W = h(a)
h(Y) = h(W) -> Y = W -> Y = h(a) *NOTA: DONDE W = h(a), se sustituye*

Unificador general: {X/b, W/h(a), Y/h(a)}

b) [X | [ f(a), f(f(X))] ] = [f(X), X, f(f(f(X)))].

R/ No unifica:
```

X = f(X) -> no existe unificador finito, ya que X aparece dentro del término que deber

Ejercicio 1

Explique en palabras precisas qué funciones prestan las siguientes clausuras:

erste([F | _], F).
R/ erste(L, F): F el primero de una lista L no vacia o genera listas L que empiecen co

a) zumindestzwo([_, _ | _]).
R/ Verdadero si la lista tiene al menos dos elementos. No importa cuáles sean ni cuánt

b) voranstellen(X, R, [X | R]).
R/ Inserta X al inicio de la lista L, produciendo [X | R]; o al revés, separa el prime

c)
anhaengen([], A, A).
anhaengen([F | R], B, [F | S]) :- anhaengen(R, B, S).

R/ Concatena listas: anhaengen([1,2],[3,4], R) -> R = [1,2,3,4].

Ejercicio 2

Escriba indexof(E, L, I) que retorne en I la posición de E en la lista L (numerando po
Ejemplo
?- indexof(666, [a, b, 666, c, [666], 666], I).
I=2;
I=5;
false
?-

Solución

```
1        % indexof(Element, List, Index)
2        indexof(E, [E | _], 0). % el elemento E esta al inicio,
            index = 0
3        indexof(E, [_ | R], I) :- % descarta el primer elemento, y
            busca en el resto
4        indexof(E, R, I1),
5        I is I1 + 1.
```

```
?- indexof(666, [a,b,666,c,[666],666], I).
I = 2 ;
I = 5 ;
false.
```

Listing 1: output

Ejercicio 3

Escriba cartesian(A, B, AxB) que recursivamente calcule en AxB una lista de listas de
Ejemplo
?- cartesian([1,2], [a,b,c], R).
R = [[1,a], [1,b], [1,c], [2,a], [2,b], [2,c]]
?-

Solución

```
% cartesian(A, B, AxB)
% calcula el producto cartesiano de dos listas A y B
% devuelve en AxB una lista de pares [a,b]

% Caso base: si la primera lista esta vacia, el resultado lo esta
cartesian([], _, []).
% Caso recursivo: genera todos los pares con el primer elemento A,
% luego combina con los pares generados del resto As
cartesian([A | As], B, R) :-
pair_with(A, B, Pairs),
cartesian(As, B, Rest),
append(Pairs, Rest, R). % une ambas listas

% pair_with(A, B, Pairs) genera una lista de pares [A, b]

% Caso base, sin elementos en B no hay pares
pair_with(_, [], []).
% Caso recursivo: crea el par [A, B] y continua con el resto
pair_with(A, [B | Bs], [[A, B] | R]) :-
pair_with(A, Bs, R).
```

```
?- cartesian([1,2], [a,b,c], R).
R = [[1,a], [1,b], [1,c], [2,a], [2,b], [2,c]].
```

Listing 2: output