PEP 2 (2019-2) - Pregunta 1/2

10/2019

RUT: _____ **Profesor: _____**

Instrucciones

- Cada pregunta debe responderse en hojas independientes. Puede usar una hoja por pregunta.
- Dispone de 90 minutos para responder la prueba.

Pregunta 1 (30 pts)

Un polinomio disperso corresponde a una representación en la que solo se almacenan aquellos términos cuyos coeficientes son distintos de 0. A modo de ejemplo, para el polinomio

$$f(x) = 10x^0 + 0x + 3x^2 + 0x^3 + 0x^4 + 0x^5 - 8x^6$$

Solo se almacena

$$f(x) = 10x^0 + 3x^2 - 8x^6$$

Luego, bajo la mirada del paradigma lógico se solicita realizar una implementación en pseudo-prolog en base a los siguientes requerimientos

a) Especificar una representación de polinomio disperso de acuerdo a los recursos de SWI-Prolog (**5 pts**).

Lista de pares de la forma [T|Ts], donde T es una lista [Coef,Exp] con Coef y Exp valores numéricos.

.

b) Predicado que permita consultar por la evaluación de un polinomio. Esto es, su resultado al sustituir la variable *x* por algún valor (**5 pts**).

```
%cláusulas y hechos
evaluar([],_,0).
evaluar([[C,E]|Ts],X,R):-Raux is C*X**E, evaluar(Ts,X,Rant),R is Rant + Raux.
```

c) Predicado que permita consultar por la ponderación de un polinomio. Esto es, el resultado del polinomio tras multiplicar cada uno de sus coeficientes por un valor (**10 pts**).

```
%clausulas
ponderarTerm([C,E],P,[NC,E]):-NC is P*C.
```

```
ponderar([T],P,[NT]):-ponderarTerm(T,P,NT).
ponderar([T|Ts],P,[NT|R]):-ponderarTerm(T,P,NT),ponderar(Ts,P,R).
```

d) Documentar con claridad código indicando Dominios, Predicados, Metas (primarias y secundarias), cláusulas (hechos, reglas) (**5 pts**).

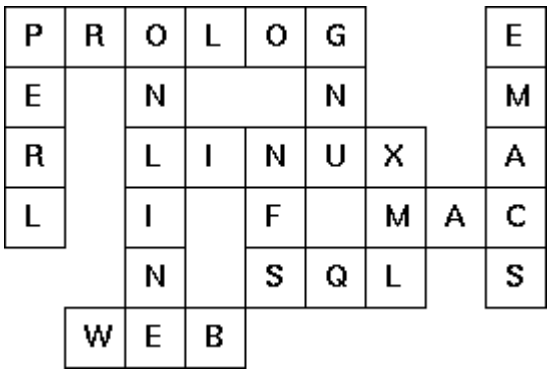
```
%Domains
%Polinomio: [Termino|Terminos]
%X:Número
%Resultado:Número
%Termino:[Coef|Exp]
%Coef:Número
%Exp:Número
%Ponderador: Termino

%Predicados
%evaluar(Polinomio,X,Resultado).
%ponderarTerm(Termino,Ponderador,Termino).
%ponderar(Polinomio,Ponderardor,Polinomio).

%Metas
%principales
%Evaluar: Evalúa un polinomio al reemplazar X por un valor particular
%Ponderar: Ponderar los coeficientes de un polinomio
%secundarias
%ponderarTerm:Pondera el coeficiente de un término específico
```

Pregunta 2 (30 pts)

Considere el siguiente crucigrama:



Luego, bajo la mirada del paradigma lógico se solicita realizar una implementación en pseudo-prolog de:

1) Representación y base de conocimiento que refleje los elementos del crucigrama (5 pts.)
2) Predicado que permita consultar a partir de la estructura vacía (o parcialmente llena) del crucigrama anterior, si disponer una letra en alguno de los espacios del mismo es válido de acuerdo a las palabras válidas que están registradas en el mismo (**10 pts.**).
3) Predicado que permita verificar si una palabra del crucigrama es palíndromo (Que se lee de igual forma de izquierda a derecha y de derecha a izquierda) (**10 pts**).
4) Documentar con claridad código indicando Dominios, Predicados, Metas (primarias y secundarias), cláusulas (hechos, reglas) (**5 pts**).

Rúbrica de Evaluación (Pregunta 2)

La completitud de cada ítem se evalúa (y pondera) con la escala [0, 0.25, 0.5, 0.75, 1]

Pregunta 1: Representación y base de conocimiento
1a) La representación es clara y concisa (2 pto)
1b) La base de conocimiento cubre el caso descrito en el crucigrama de ejemplo (2 pto)
1c) La base de conocimiento es coherente (1 pto)

Pregunta 2: Poner letra en el crucigrama
2a) Descripción de la solución (2 pto)
2b) Predicados auxiliares (metas secundarias) (2 pto)
2c) Casos base (3 ptos)
2d) Descomposiciones Recursivas (3 ptos)

Pregunta 3: Verificar Palíndromo
3a) Descripción de la solución (2 pto)
3b) Predicados auxiliares (metas secundarias) (2 pto)
3c) Casos base (3 ptos)
3d) Descomposiciones Recursivas (3 ptos)

Pregunta 4: Documentación del código
4a) Indica Dominios (1 pto)
4b) Indica Predicados (1 pto)
4c) Indica Metas (1 pto)
4d) Indica Hechos (1 pto)
4e) Indica Reglas (1 pto)

Pauta Pregunta 2 (solo como referencia, parcialmente probada en SWI-Prolog)

```
% Representación TDA Palabra
% palabra(I, D, L, X, Y, P)
% I: ID de la palabra
% D: Dirección de la palabra (horizontal o vertical)
% L: Largo de la palabra en caracteres
% X: Posición en eje X de la primera letra
% Y: Posición en eje Y de la primera letra
% P: Lista con los caracteres de la palabra

% Base de conocimiento (hechos)
palabra(1, horizontal, 6, 1, 1, [p, r, o, l, o, g]).
palabra(2, vertical, 4, 1, 1, [p, e, r, l]).
palabra(3, vertical, 6, 3, 1, [o, n, l, i, n, e]).
palabra(4, vertical, 3, 6, 1, [g, n, u]).
palabra(5, vertical, 5, 9, 1, [e, m, a, c, s]).
palabra(6, horizontal, 5, 3, 3, [l, i, n, u, x]).
palabra(7, horizontal, 3, 7, 4, [m, a, c]).
palabra(8, horizontal, 3, 5, 5, [s, q, l]).
palabra(9, horizontal, 3, 2, 6, [w, e, b]).
palabra(10, vertical, 3, 5, 3, [n, f, s]).
palabra(11, vertical, 3, 7, 3, [x, m, l]).

% Reglas

% Predicado pertenece(L, E)
% Dominio: Lista X Átomo
% Meta secundaria
% Verifica si un átomo E pertenece a una lista L
pertenece([E|Xs], E).
pertenece([X|Xs], E):- pertenece(Xs, X).
```

```
% Predicado verificarLetra(L, P)
% Dominio: Lista X Átomo
% Meta secundaria
% Verifica si una letra L pertenece a una palabra P
verificarLetra(L, P):-
      palabra(I, _, _, _, _, W), pertenece(W, L), P is I.

% Predicado obtenerPosicion(L, N, E)
% Dominio: Lista X Entero X Átomo
% Meta secundaria
% Obtiene el elemento E en la posición N de una lista L
obtenerPosicion([X|Xs], 1, E):- E is X.
obtenerPosicion([X|Xs], N, E):- Ns is N-1, obtenerPosicion(Xs, Ns).


% Predicado ponerLetra(L, X, Y)
% Meta primaria
% Verifica si una letra L es válida en una posición (X,Y)
ponerLetra(L, X, Y):-
      verificarLetra(L, P),
      palabra(P, _, Ln, Px, Py, Ls),
      X2 is Px+X,
      Y2 is Py+Y,
      ((obtenerPosicion(Ls, X2, E), L=E);
      obtenerPosicion(Ls, Y2, E), L=E))).

% Predicado reversar(L, Z, T)
% Dominio: Lista X Lista X Lista
% Meta secundaria
% Reversa una lista L y la retorna en Z
reversar([], X, X).
reversar([X|Xs], Z, T):- reversar(Xs, [X|Z], T).

% Predicado palindromo(ID)
% Dominio: Entero
% Meta primaria
% Dado el ID de una palabra, verificar si es palíndromo
palindromo(ID):-
      palabra(ID, _, _, _, _, Ls),
      reversar(Ls, [], Ls).
```