Lógica y Programación
Trabajo práctico Prolog

9 de Noviembre de 2020

1 Introducción

Corría el año 2000 y los teléfonos celulares revolucionaban el mundo. Al igual que hoy en día, existían miles de modelos que introducían numerosas novedades en cuanto a diseño y funcionalidad. De entre esta temprana generación de teléfonos celulares se destaca nuestro héroe, el Nokia 1100.

El 1100 nunca fue un teléfono con unas capacidades espectaculares pero sí que cumplía con lo que por entonces se pedía a un teléfono móvil. Un teléfono con función linterna, con hasta 36 tonos para elegir, y el famoso compositor de tonos de Nokia para almacenar hasta 7 tonos creados por nosotros, y el Snake II, el mítico juego de la serpiente. También ofrecía otras funciones como cronómetro, calculadora o una función chat. Pero por sobre todo destacaba su diseño robusto y su carcasa "indestructible", fuente de innumerables proezas y que le valió llevó en 2007 el galardón de "teléfono más vendido de la historia".

Como no podía ser menos entre sus coetáneos, también incluía la famosa función de teclado predictivo. Esto implicaba la incorporación de un diccionario de palabras y un algoritmo que iba restringiendo las posibles palabras que puedan ser resultado de la presión de una serie de teclas del aparato.

El objetivo del Trabajo Práctico es representar en Prolog el comportamiento predictivo de los mensajes de texto del Nokia 1100. Para ello se definen dos predicados que brindan la información necesaria del teléfono celular:

1. Se define el mapeo entre teclas del celular y los caracteres de la siguiente manera:

```
teclado([
            (1, [1]),   (2, [2,a,b,c]),   (3, [3,d,e,f]),
      (4, [4,g,h,i]),   (5, [5,j,k,l]),   (6, [6,m,n,o]),
    (7, [7,p,q,r,s]),   (8, [8,t,u,v]), (9, [9,w,x,y,z]),
          (*, [*,+]),       (0, [0,-]),         (#, [#])
])
```

   Es decir, con una lista de tuplas `(D,Xs)` donde `D` es el dígito y `Xs` es la lista de caracteres que puede representar dicho dígito. Los caracteres se

representan por medio de un átomo o un número entero. Notar que el carácter " " (espacio) está representado por el átomo "`-`".

2. El diccionario de palabras se representa de la siguiente manera:

```
diccionario([
  [l,a], [l,a], [c,a,s,a], [a], [d,e], [r,e,j,a], [t,i,e,n,e],
  [c,a,s,a,m,i,e,n,t,o], [d,e,l], [a,n,t,e,s]
])
```

   Es decir, con una lista de listas de caracteres que representan las palabras conocidas. Puede asumirse que ninguna palabra del diccionario contiene el átomo "`-`".

2 Implementación

Se pide definir los siguientes predicados, respetando la instanciación pedida de tal manera que no se devuelvan soluciones repetidas:

1. `teclasNecesarias(+Palabra, -ListaDigitos)` donde `Palabra` es una lista de caracteres y tiene éxito si `ListaDigitos` es la lista de dígitos que deben presionarse para obtenerla. Ejemplo:

```
?- teclasNecesarias([c,a,s,a], Ds).
Ds = [2,2,7,2] ;
false.
```

2. `palabraPosible(+ListaDigitos, ?Palabra)` donde `ListaDigitos` es una lista de teclas presionadas y tiene éxito si `Palabra` es una palabra del diccionario, y con las teclas presionadas se obtiene esa palabra o un prefijo de la misma. Ejemplo:

```
?- palabraPosible([2], P).
P = [a] ;
P = [a,n,t,e,s] ;
P = [c,a,s,a] ;
P = [c,a,s,a,m,i,e,n,t,o] ;
false.
```

3. `todasLasPalabrasPosible(+ListaDigitos, ?Palabras)` donde `ListaDigitos` es una lista de teclas presionadas y tiene éxito si `Palabras` es la lista de palabras del diccionarios tal que las teclas presionadas generan una lista de caracteres que puede ser prefijo de las mismas.

   Nota: tener en cuenta que la solución debe ser vista como un conjunto. O sea, soluciones con las mismas palabras pero en distinto orden deben ser consideradas como iguales (no deben devolverse repetidos). Ejemplo:

```
?- todasLasPalabrasPosible([2], Ps).
Ps = [[a], [a,n,t,e,s], [c,a,s,a], [c,a,s,a,m,i,e,n,t,o]].

?- todasLasPalabrasPosible([2], [[c,a,s,a], [a], [a,n,t,e,s],
    [c,a,s,a,m,i,e,n,t,o]]).
true.
```

4. `oracionPosible(+ListaDigitos, -Oracion)` donde `ListaDigitos` es una lista de teclas presionadas que puede incluir `0` (que mapea al espacio) y tiene éxito si se formó una `Oracion` correcta. Una oración es correcta si cada una de las secuencias de teclas entre los `0` puede formar una palabra del diccionario. Ejemplo:

```
?- oracionPosible([2,0,3], O).
O = [a,-,d,e] ;
O = [a,-,d,e,l] ;
O = [a,n,t,e,s,-,d,e] ;
O = [a,n,t,e,s,-,d,e,l] ;
O = [c,a,s,a,-,d,e] ;
O = [c,a,s,a,-,d,e,l] ;
O = [c,a,s,a,m,i,e,n,t,o,-,d,e] ;
O = [c,a,s,a,m,i,e,n,t,o,-,d,e,l] ;
false.
```

3 Pautas de entrega

La entrega se realiza por mail a la lista lds-doc-lyp@listas.unq.edu.ar. Se debe incluir el código fuente junto con un pequeño informe en donde además de los predicados definidos se comente la idea de implementación en cada uno de los predicados. **Fecha de entrega 22-11-2020. Recuperatorio 29-11-2020.**