Informe TP Prolog

Alejandro Rossi

Noviembre, 2020

Abstract

Informe de programa codificado en Prolog, perteneciente al Trabajo práctico de la materia Lógica y Pogramación de la UNQ, carrera Licenciatura en Informática.

1 Punto 1: teclasNecesarias

Para esta solución creé dos funciones auxiliares, dándole una estructura final similar a la de una función de lenguajes mas "tradicionales", ya que quedó divido en responsabilidades. Dichas funciones están descritas en 1.1 y 1.2, siendo la función principal descrita en 1.3.

1.1 listaDeDigito((Digito, X))

En esta función se le pasa una tupla, conteniendo un `Dígito` y una `X`, que será el elemento con el cual unificará en teclado.

Con la ayuda de `teclado(Teclado)`, que es parte de la base de conocimiento y la función nativa de Prolog `member/2`, permite encontrar el listado de teclas perteneciente a `Digito`. También sirve para verificar si un listado dado (pasado en `X`) es un teclado perteneciente al dígito, es decir:

- `listaDeDigito((2,X))` retorna `[2,a,b,c]`
- `listaDeDigito(2, [2,a,b,c])` retorna `true`
- `listaDeDigito((2,[2,a,c]))` retorna `false`

Para dicha parte de la tarea me inspiré en ejercicios hechos en las clases prácticas donde veíamos la utilización de la función `member`.

1.2 letraEsDeDigito(Letra, Digito)

Para esta segunda función auxiliar, decidí resolver otra pequeña parte de la problemática, que sería ver si una letra pertenece al listado de determinado dígito, es decir, recibe la letra a verificar, el Dígito y utiliza `member/2` sobre el listado encontrado con la función descrita en **1.1**.

1.3 teclasNecesarias(+Palabra, -ListaDigitos)

En la solución pedida en el punto uno, utilizo ambas funciones, definidas en **1.1** y en **1.2**, es decir que puedo verificar por cada dígito del listado pasado por parámetro, si la letra corresponde a éste. En cuanto al caso base, considero que ambas listas son del mismo largo, por lo cual deben ser ambas vacías.

2 Punto 2: palabraPosible

En el segundo punto también realicé dos funciones auxiliares, una a modo de verificación de palabra valida y otra que fue vista en clases. Esta función es explicada en el punto **2.3**.

2.1 palabraDeDiccionario(Palabra)

Parecido a lo hecho previamente, en esta funcion verificamos que la palabra pasada sea válida, es decir forme parte del diccionario, por lo cual hago uso de `member/2`.

2.2 prefijo(Lista1,Lista2)

Esta función fue implementada en la clase practica, por lo cual reutilicé el funcionamiento.

2.3 palabraPosible(Ds, Palabra)

Esta función recibe una lista de dígitos y una palabra, utilizando las funciones auxiliares y la del punto anterior, logra verificar no solo que la palabra pertenezca al diccionario, sino que los dígitos son prefijos de la palabra y que a la vez puede ser formada por esos dígitos.

3 Punto 3: todasLasPalabrasPosibles

Este punto, según mi esfuerzo y tiempo invertido, fue, en mi opinión, el mas difícil del trabajo. Lo resolví con el uso de dos funciones auxiliares, una de ellas para eliminar elementos de una lista y evitar resultados repetidos, y la otra una función de permutación.

3.1 del(Elem,Lista,Res)

Esta función es reutilizada de lo hecho en clase, elimina un elemento de una lista.

3.2 perm(List1,List2)

Esta función se encarga de permutar la List1 pasada como parámetro, dejando en List2 la permutación, además utiliza la función anterior.

3.3 todasLasPalabrasPosibles(Ds, Ls)

En esta función se utiliza la permutación para poder obtener los distintos resultados, y también se reutiliza el método del punto 2 `palabraPosible`.

Otra función utilizada es una nativa de Prolog: `findall/3` que permite buscar todos los resultados que unifican con `palabraPosible` junto a `perm`.

4 Punto 4: oracionPosible

En el ultimo punto del trabajo práctico, inicialmente tenía un total de 4 funciones pero no lograba que funcionen de manera correcta en `oracionPosible` así que reduje las funciones auxiliares a 2, básicamente dividiendo la lógica, una encargada de hacer reemplazos y otra de la validación.

4.1 oracionBienSeparada(List)

Esta función es la encargada de realizar la validación de caracteres, fue particularmente complicada ya que debía controlar que haya "0" separando los elementos de la lista, siendo su aparición previa o póstuma a la del resto de elementos.

Para lograrlo, hago uso de la función nativa `dif/2` que retorna true, sólo si los términos pasados como parámetro son diferentes. La lógica básica que tiene es que considera los escenarios `0,OtroDigito` y `OtroDigito,0` al recorrer recursivamente la lista de dígitos.

4.2 espacioOPalabra(Lista1, Lista2)

Como mencioné previamente, en este método delego la responsabilidad del reemplazo de caracteres, si son un "0" a un "-" y sino, reutilizo `palabraPosible` para obtener el reemplazo correspondiente.

Inicialmente tenia los reemplazos separados en dos funciones distintas, y luego intentaba aplanar la lista para obtener todo junto. Pero no solo no lograba un funcionamiento correcto, sino que sentía que hacía pasos de mas y que el código era poco intuitivo/entendible.

Luego de leer la documentación de Prolog, decidí hacer uso del condicional `booleano -> then; else.`. Con esto logré hacer ambos reemplazos en la misma iteración del mismo bloque de código y a la vez con el uso de `append/3`, guardar el resultado en una lista de respuesta.

Como output obtengo una lista de palabras separadas por espacios, obtenida en una sola iteración, sin necesidad de aplanar o volver a recorrer para validar.

4.3 oracionPosible(Xs,Ys)

Gracias al trabajo extra en las dos funciones previas, esta función logra quedar simplificada ya que es solamente quien llama las subtareas, validando los parámetros y obteniendo la respuesta. Es decir que el verdadero trabajo de esta función está en **4.1** y **4.2**.

5 Opiniones y conclusión final

Como inicio de conclusión puedo comenzar con una valoración personal sobre la dificultad del mismo, la cual considero que fue bastante alta. Si bien al observar el código final puedo ver que en términos generales es simple, la creación del mismo no lo fue, pero lo difícil radicaba principalmente en el cambio de "paradigmas tradicionales" al paradigma lógico, esto provocó en una curvatura de avance muy inclinada, ocasionando que el inicio del trabajo haya obtenido un porcentaje de avance prácticamente insignificante en relación al tiempo invertido.

Sin ahondar mucho en detalles sobre cada uno de los puntos, el resto de las complicaciones fueron inherentes a las que se esperan de una instancia de evaluación.

En cuanto a mi valoración del enunciado, me hubiese gustado haber tenido más ejemplos en cada punto, para poder entender mejor lo que se pedía.