Códigos Ambíguos

Trabalho Prático de Programação III

Introdução

Por questões de segurança, o Departamento de Informática do Banco da Antártida desenvolveu um sistema de códigos para guardar a informação relativa aos seus clientes. Neste sistema altamente sofisticado, a cada símbolo (letra, número, ...) é associada uma *sequência de bits* (0s e 1s) e cada string é codificado pela concatenação das sequências correspondentes aos seus caracteres.

Tudo parecia estar a correr bem até ao dia em que a D. Java apareceu a reclamar porque a conta do barbeiro do Sr. Pascal lhe tinha sido debitada. O gerente do banco foi investigar e descobriu que o código que estava a ser utilizado era o seguinte.

a	c	j	l	p	s	v
010	01	001	10	0	1	101

De facto, o nome **pascal** é representado pelo código **001010101010**, e infelizmente...este código também corresponde ao nome **java**.

Tarefa

A sua tarefa consiste em escrever um programa que receba um código ambíguo e devolva a mensagem codificada **mais curta** (caso haja várias, a primeira por ordem lexicográfica) que pode ser interpretada de forma ambígua e duas das suas possíveis interpretações.

Se optar pela Programação Lógica, implemente o programa através de um predicado **ambiguo/4** que receba no primeiro argumento o código e devolva no segundo argumento a

triplo contendo a mensagem codificada encontrada e duas das suas possíveis interpretações.

Os Dados

O código é fornecido como uma lista de pares cujo primeiro elemento é o símbolo e o segundo elemento é o código desse símbolo (lista de 0s e 1s).

Os Resultados

A mensagem codificada deve ser representada como uma lista de 0s e 1s; as mensagens descodificadas devem ser representadas como listas de átomos (Prolog) ou caracteres (OCaml).

1º Exemplo (Prog. em Lógica)

Segue-se um exemplo ilustrativo do programa pretendido.

```
| ?- ambiguo([(a, [0,1,0]),
              (c, [0,1]),
              (j, [0,0,1]),
              (l, [1,0]),
              (p, [0]),
              (s, [1]),
              (v, [1,0,1])], M,
           T1, T2).

M = [0, 1]
T1 = [c]
T2 = [p, s]
```

Outro exemplo que o seu programa deverá ser capaz de resolver é o seguinte.

```
| ?- ambiguo([(a, [0,1,1,0]),
              (b,
     [0,1,1,1,1,1]),
              (c,
     [1,1,0,0,1,1,1,1]),
              (f,
     [1,0,1,1,1,0]),
              (j, [0,1,0]),
              (l, [0,1,0,0]),
              (r,
     [0,1,1,1,0])], M, T1,
     T2).
```

pretendido:

```
# ambiguo [('a', [0;1;0]);
           ('c', [0;1]);
           ('j', [0;0;1]);
           ('l', [1;0]);
           ('p', [0]);
           ('s', [1]);
           ('v', [1;0;1])] ;;

int list * char list * char
list = ([0; 1], ['c'], ['p';
's'])
```

Condições Gerais

- Grupos de com 1 ou 2 alunos
- Entrega
  - 2021.01.17
  - Moodle com 2 ficheiros
    - zip (ou tar.gz) com todo o código source Prolog ou OCaml
    - pdf com o relatório
- Discussão
  - 2010.01.28-29

Relatório

Deverá entregar um relatório com o máximo de 3 páginas A4 , em formato PDF, e no qual aborda as seguintes questões:

- Discussão das escolhas que realizou (linguagem, organização do código, etc...)
- Limites de funcionamento do programa.