Escola de Ciências e Tecnologia 

Departamento de Informática

Licenciatura em Engenharia Informática

Unidade curricular Programação III

Ano letivo 2020/2021

**Relatório**

**Trabalho Prático dde Programação III**

Docentes

Professor Salvador Abreu

Professor Pedro Patinho

Discentes

Ludgero Teixeira, nº41348

José Santos, nº43017

Évora, janeiro de 2021



**Descrição do trabalho**

O trabalho 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 se optar pela Programação Lógica, tem que se implementar um programa

através de um predicado ambíguo/4 que recebe, no primeiro argumento, o código e devolve, no segundo argumento, a mensagem codificada encontrada e nos dois restantes argumentos duas das suas possíveis interpretações.

Se se escolher a Programação Funcional, tem que se desenvolver uma

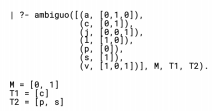
função ambíguo que recebe como argumento o código e devolve um triplo contendo a mensagem codificada encontrada e duas das suas possíveis interpretações.

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).

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).



Acima constam dois exemplos do programa e output desejado, sendo que: caso se

opte pela programação lógica, Prolog (imagem da esquerda) ou pela programação funcional, Ocaml (imagem da direita).

2



**Escolhas e resoluções**

Para a resolução deste programa, decidimos usar a linguagem de programação funcional Ocaml.

A abordagem utilizada para a resolução do mesmo foi a seguinte: começámos por calcular todos os arranjos possíveis, dentro dos limites, usando a lista de códigos dada. De seguida, tendo os arranjos, calculámos se duas mensagens distintas, produzem o mesmo código.

Para certificar que recebemos a menor mensagem possível, comparámos todos os arranjos, com a lista dos códigos. Se o programa não encontrar ambiguidade, comparámos a lista que contém todos os arranjos, com a lista de arranjos de 2, e por aí adiante.

Ao encontrar esta ambiguidade, recebemos um tuplo do tipo **(['p';'s'],[0;1])**, sendo que, com este, percorremos uma última vez a lista de todos os arranjos, até encontrarmos qual o tuplo que tem o código igual a este, tendo cuidado para não encontrar o mesmo.

Por fim, arrumámos as mensagens para que, estas, ficassem idênticas ao resultado esperado.

Os outputs gerados pelos exemplos dados no enunciado foram os seguintes:

• Primeiro exemplo - ([0;1], ['c'], ['p'; 's'])

• Segundo exemplo - ([0;1;0;0;1;1;0;0;1;1;1;1;1;0;1;1;1;0], ['l'; 'c'; 'f'], ['j'; 'a'; 'b'; 'r'])

Esta breve explicação do funcionamento do programa, encontra-se mais detalhada no código sucintamente comentado.

3



**Limites de funcionamento**

Tendo em conta a lógica utilizada para a resolução do problema, o programa

reproduz todos os outputs esperados, especificados no enunciado do trabalho. No entanto, devido à mesma, existem certas limitações no que toca ao funcionamento do programa. Embora para arranjos até seis caracteres o programa corra como esperado, para arranjos de sete caracteres, guardados na mesma lista, o programa acaba por entrar em overflow. Esta limitação poderia, contudo, ter sido evitada, se, em vez dos arranjos serem guardados, fossem calculados sempre que fosse preciso, o que iria causar um grande impacto temporal na execução do programa.

Esta limitação faz com que, por muito pequena que seja a ambiguidade encontrada, como por exemplo [(‘a’;[0]);(‘b’;[0])] -> ([0],[‘b’],[‘a’]), o programa calcula primeiro todos os arranjos, e só depois é que compara com a lista de códigos.

Em suma, apesar do método utilizado não ser o mais eficiente, acreditamos que o presente programa resolve todos os exemplos dentro das limitações supramencionadas. Deste modo, a resolução deste trabalho, possibilitou a obtenção de novos conhecimentos e um aprofundamento da linguagem Ocaml.

4