UNIVERSIDADE DE ÉVORA

Programação III - Trabalho Prático - Relatório

Trabalho realizado por:

Rui Roque nº 42720

Tomás Dias nº 42784

Implementação

Predicados utilizados

Para a realização do trabalho prático foi utilizado o **Prolog**. O tempo e a exploração que esta linguagem obteve durante as aulas, e consequente habituação com a mesma, foram as principais razões para a sua escolha.

Foram utilizados os seguintes predicados para a criação do programa:

- **atualiza/3:** Este predicado recebe o código do símbolo (**C**) e a mensagem (**M**) e devolve uma lista (**R**) com a sequência de dígitos da mensagem que não está contida no código do símbolo.

- **contido/3:** Este predicado recebe o código do símbolo (**C**) e a mensagem (**M**) e se o código do símbolo estiver contido na mensagem devolve uma lista (**R**) com o código do símbolo, caso contrário devolve uma lista (**R**) vazia.

- **verifica_ambiguidade/3:** Este predicado recebe o código com os símbolos e a sua respetiva sequência de bits (**L**) e a mensagem (**M**) e verifica se no código existem sequências de bits cujo a sua junção seja igual à mensagem e, por sua vez, se verifique a ambiguidade. É utilizado o predicado **contido** para esta verificação. Em caso afirmativo, devolve uma lista (**T2**) em que estão contidos os símbolos que formam uma das interpretações da mensagem. Caso contrário, devolve uma lista (**T2**) vazia.

- **ambiguo/4:** Este predicado recebe o código com os símbolos e a sua respetiva sequência de bits (**L**) e devolve a mensagem codificada encontrada (**M**) e duas das suas possíveis interpretações (**T1**) e (**T2**). É utilizado o predicado **verifica_ambiguidade**.

Funcionamento do programa

Primeiramente, no predicado **ambiguo,** é verificado se a sequência de bits do primeiro símbolo, ou seja, a mensagem, é ambígua. Para esta verificação, é chamado o predicado **verifica_ambiguidade**.

No predicado **verifica_ambiguidade**, um par (símbolo e sequência de bits) é analisado em cada iteração utilizando o predicado **contido**.

No predicado **contido**, é verificado se a sequência de bits da iteração atual está contida na mensagem. Caso esteja contida, novamente no predicado **verifica_ambiguidade**, é chamado o predicado **atualiza**.

No predicado **atualiza**, a mensagem passa a conter a sequência de bits não comum á sequência de bits da iteração atual.

Ex: Sendo M = [1,0,0,1] e C = [1,0] e estando C⊂M, o novo M = [0,1].

Novamente em **verifica_ambiguidade**, é executado um novo ciclo repetindo o procedimento descrito para a nova mensagem.

Quando no predicado **verifica_ambiguidade** a iteração atual é no último par do código e o predicado contido não é chamado, o programa volta para o predicado **ambiguo** e é retornado uma lista com os símbolos que compõem a mensagem (no caso de não existir ambiguidade, a lista é vazia).

No predicado **ambiguo**, é novamente analisado cada par do código e se o tamanho da sequência de bits deste (sendo este necessariamente ambíguo) for inferior ao tamanho da sequência de bits do par anterior considerado ambíguo, a interpretação do código ambíguo é substituída pelo par analisado.

Por fim, é apresentada a mensagem ambígua com as respetivas duas interpretações possíveis.

Outputs de testes realizados

```
| ?- 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]
```

```
| ?- 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).

M = []
T1 = []
T2 = []
```

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

M = [0,1,0]
T1 = [a]
T2 = [l,p]
```

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

M = [0,1,0]
T1 = [a]
T2 = [l,p,l]
```

```
| ?- 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])], M, T1, T2).

M = [0,1,0,0]
T1 = [l]
T2 = [j,r]
```

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

M = [0,1,0]
T1 = [j]
T2 = [f,r]
```

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

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

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

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