

Universidade Federal de Juiz de Fora
Departamento de Ciência da Computação
DCC019 - Linguagem de Programação

Cifrando Mensagens em Prolog

Grupo

André Dias Nunes — MAT: 201665570C

Gabriel Di iorio Silva — MAT: 201765551AC

Professor

Leonardo Reis

Juiz de Fora

Maio de 2022

Sumário

1	Introdução	2
2	Banco de Dados	2
3	Cifra de César	3
3.1	Decifrador de César	4
4	Cifra de Vigenere	5
4.1	Primeiro predicado de Vigenere	6
4.2	Segundo predicado de Vigenere	7
4.3	Terceiro predicado de Vigenere	7
4.4	Quarto predicado de Vigenere	7
5	Instruções	8

1 Introdução

O presente Relatório Técnico tem como objetivo central descrever o desenvolvimento de um codificador, decodificador e decifrador utilizando os modelos da cifra de César e Vigenere.

Os capítulos a seguir representaram as etapas do desenvolvimento.

2 Banco de Dados

A etapa inicial deste trabalho, consiste em criar a tabela de símbolos que serão aceitos e associar com seus respectivos códigos únicos. Inicialmente pensamos em utilizar os valores da tabela ASCII, mas para facilitar nas transformações internas durante o programa, optamos por criar uma nova tabela, inicial com o espaço recebendo o valor 0, seguido por todas as letras minúsculas, letras maiúsculas e pontuações, com valores inteiros em ordem crescente.

O outro banco de dados é para salvar as palavras para que o programa possa reconhecer na etapa de Decifragem.

Para criar o banco de dados utilizamos a biblioteca `persistence`. Ao definirmos os predicados de manipulação e inserção, criasse automaticamente os arquivos `charCodes.journal` e `words.journal` que armazenam os logs das alterações.

Para a inserção de novas palavras para reconhecimento, basta utilizar o predicado `add_word("palavra")` e para remoção o predicado `del_word("palavra")`. Pode ser inserido caracteres maiúsculos, porem internamente a palavra sera salva em `lowercase`. Utilize aspas para passar a palavra.

3 Cifra de César

Para codificar e decodificar a cifra de César utilize o predicado `caesar(Key,"Decodificada","Codificada")`. O primeiro parâmetro deve ser passado o caractere que sera usado como chave. O segundo parâmetro deve ser enviado o texto traduzido que deve ser codificado. No terceiro parâmetro, deve ser inserido a frase no formato codificado para que seja traduzida. O parâmetro que não for enviado, será retornado com sua respectiva informação, como no exemplo abaixo:

```
?- caesar(j,"Estou testando o César!", A).  
A = PãézêjépãékyozjzjMÇãkâ,.  
  
?- caesar(j,B, "PãézêjépãékyozjzjMÇãkâ,").  
B = Estou testando o César!.  
  
?- caesar(Key, "Estou testando o César!", "PãézêjépãékyozjzjMÇãkâ,").  
Key = j .
```

Figura 1: Exemplo de execução do César

Todo o código possui comentários detalhados explicando as ações tomadas. Para o desenvolvimento da cifra de César, segue-se estas etapas:

- Inicialmente, é buscado o valor do code da Chave que foi passada. Se o predicado recebeu o texto codificado, então o transforma em lista, se foi o texto decodificado o transforma em lista.
- Então é utilizado o predicado `maplist`, enviando como parâmetros o predicado `cypherC` com o código da chave fixado, a lista de entrada e a lista de saída. O predicado `maplist` executa a cifra para cada elemento da lista de entrada.
- No predicado `cypherC` se foi enviado o code do char decodificado, então é calculado o code do char codificado utilizando a chave, ou vice versa. E no fim, converte de code para char e o retorna salvando na lista de saída.
- Ao executar o `cypherC` para todos os caracteres de entrada, converte a lista de saída para atom.

3.1 Decifrador de César

Para decifrar a cifra de César é utilizado o predicado `decypherCaesar("Texto codificado")`. Onde o único parâmetro é o texto codificado. Ao encontrar é exibido no console a Key da cifra e a Frase decodificada. Como no exemplo abaixo:

```
?- decypherCaesar("PãézêjépãékyozjzjMÇäkâ, ").  
  
Key: j  
Result: Estou testando o César!  
true .
```

Figura 2: Decifragem de César

Para o desenvolvimento do decifrador de César, segue-se estas etapas:

- Inicialmente é chamado o predicado `maxFreqChar` para calcular qual é o char mais frequente da entrada e é buscado seu código.
- Utilizando o predicado `listConcat`, é criada uma lista de códigos das letras mais frequentes da língua portuguesa em ordem decrescente de frequência. Utilizando o predicado `wordSave` é criada uma segunda lista com todas as palavras salvas que o programa reconhece.
- Então é chamado o predicado `decypherC` que calcula a chave necessária para que a letra mais frequente encontrada na entrada seja correspondente à letra mais frequente da língua portuguesa.
- Executa-se a cifra de César com a chave calculada e o texto de entrada e o texto obtido passa por um tratamento removendo pontuação, transformando em lower-case e é criada uma lista de palavras ao separar com o predicado `splitWords` que cria uma sublista ao encontrar um espaço e incrementa nesta nova lista.
- É verificado se a interseção entre a lista de palavras do texto obtido com as palavras salvas, e se houver se encerra, se não retorna tentado utilizar a próxima letra mais frequente.

4 Cifra de Vigenere

Para codificar e decodificar a cifra de Vigenere utilize o predicado `vigenere("Chave","Decodificada","Codificada")`. O primeiro parâmetro deve ser passado a palavra que sera usado como chave. O segundo parâmetro deve ser enviado o texto traduzido que deve ser codificado. No terceiro parâmetro, deve ser inserido a frase no formato codificado para que seja traduzida. Diferente do César, neste caso a palavra chave não é aceita como variável livre, somente os textos codificados ou decodificados. Como no exemplo abaixo:

```
?- vigenere("teste","Agora é hora do Próximo! Deu certo?",A).  
A = UmôAftúsâuAfsyutVüPããsô@eYkCthzxBõ_ .  
  
?- vigenere("teste",B,"UmôAftúsâuAfsyutVüPããsô@eYkCthzxBõ_").  
B = Agora é hora do Próximo! Deu certo? .
```

Figura 3: Exemplo de execução da cifra de Vigenere

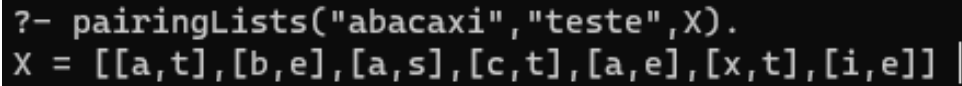
Para o desenvolvimento da cifra de Vigenere, segue-se estas etapas:

- Inicialmente, segue o mesmo padrão da cifra de César, com a diferença que agora a palavra chave é convertida em uma lista de codes utilizando o predicado `string2code`. A seguir é calculado o tamanho da chave e é criado um contador para percorrer os elementos da chave.
- O predicado `cypherV` pega o elemento da lista chave definido pelo contador, e executa o mesmo processo da cifra de César para calcular seu valor equivalente cifrado/decifrado.
- Se ainda não chegou no fim do texto de entrada, é verificado se o contador atingiu o fim da lista chave, se sim o valor do contador volta para o começo, se não é incrementado.
- Ao final converte o texto de saída para string.

4.1 Primeiro predicado de Vigenere

Para decifrar a cifra de Vigenere, foi solicitado o desenvolvimento de 4 predicados. Para o primeiro predicado é pedido que seja feita um predicado que receba duas listas e crie uma outra listas cada elemento sera um par de elementos das outras duas. Este predicado foi definido como `pairingLists("Lista1", "Lista2", "ListaResultado")`. Como o exemplo a seguir:

A imagem abaixo explica a lógica empregada no decifrador de César.



```
?- pairingLists("abacaxi", "teste", X).  
X = [[a,t],[b,e],[a,s],[c,t],[a,e],[x,t],[i,e]] |
```

Figura 4: pairingLists

Para o desenvolvimento deste predicado, segue-se estas etapas:

- Inicialmente verifica se o tamanho da Segunda lista é menor que a da primeira, se for, envia a segunda lista para o predicado `completList` para que ela seja concatenada consigo mesma até que atinja o tamanho da primeira lista.
- A seguir ambas as listas são enviadas a um predicado auxiliar, onde a head de ambas é inserida no na head da lista de saída na seguinte forma: `pairingListsAux([H1|T1], [H2|T2], [[H1,H2]|T])` até que ambas listas se esgotem.

4.2 Segundo predicado de Vigenere

Para o segundo é pedido um predicado que relaciona uma mensagem cifrada, um tamanho de chave, e uma palavra que sabidamente ocorre na mensagem decifrada e sua posição, com a chave. Definimos o predicado como `decypherVigenerePosition("Texto Cifrado", TamanhoDaChave, "Dica", PosiçãoDaDica, Chave)`, como exemplificado na figura 5.

Para o desenvolvimento deste predicado, segue-se estas etapas:

- Inicialmente pegamos o elemento do texto de entrada na posição da dica que foi passada e criamos um contador inicializando com a posição seguinte e através do predicado `getHintEncrypted` é obtida a palavra dica na sua versão cifrada.
- Utilizando o predicado anterior `pairingLists` é feito uma lista associando a dica cifrada e decifrada e é buscado utilizando a cifra de César o chave de cada associação. A seguir é duplicado a lista das chaves adquiridas, pois é sabido que a chave é menor que a dica, logo nesta lista de chaves em alguma posição esta a chave verdadeira.
- No predicado auxiliar, é selecionado os primeiros elementos da lista de chaves até que atinja o tamanho da chave, então é passado o resultado pela cifra de Vigenere e o texto resultante passa pelo mesmo tratamento do decifrador de César e é verificado se a dica foi encontrada nesse texto.
- Se foi encontrada, retorna a chave, se não, remove o primeiro elemento da lista de chaves e tenta novamente.

4.3 Terceiro predicado de Vigenere

Para o terceiro é pedido Um predicado que relaciona uma mensagem cifrada, um tamanho de chave e uma palavra que ocorre no texto com a mensagem decifrada. Definimos o predicado como `decypherVigenereHint("Texto Cifrado", TamanhoDaChave, "Dica", Chave)`. Exemplificado na figura 5.

A lógica aplicada é bem simples. Definimos um contador e supomos que a posição da dica seja a inicial e testamos utilizando o predicado anterior, se não tiver sucesso incrementamos o contador e testamos a posição seguinte até encontrar uma solução.

4.4 Quarto predicado de Vigenere

Para o terceiro é pedido um predicado que relaciona uma mensagem cifrada, uma lista de possiveis palavras que ocorre no texto e um tamanho de chave com a mensagem de-

cifrada.. Definimos o predicado como `decypherVigenereListHints("Texto Cifrado", TamanhoDaChave, ["Lista das Dicas"], Chave)`. Exemplificado na figura 5.

A lógica aplicada também é simples, é passado a cada elemento da lista de dicas para o predicado anterior, se não tiver sucesso tenta-se com a próxima dica.

Abaixo a imagem exemplificando os 3 últimos predicados solicitados.

```
Vigenere Key: [a,c,a,b,o,u]
Vigenere Encode: ÇkfiþøerapéCaijppónhovuCagpbÊBôájoé[aGfwoXfuuqf
Vigenere Decode: Chegando nos finalmentes do Próximo! Deu Certo?
true .

?- decypherVigenerePosition("ÇkfiþøerapéCaijppónhovuCagpbÊBôájoé[aGfwoXfuuqf", 6, "finalmentes", 14, Key).
Key = acabou .

?- decypherVigenereHint("ÇkfiþøerapéCaijppónhovuCagpbÊBôájoé[aGfwoXfuuqf", 6, "finalmentes", Texto).
Texto = Chegando nos finalmentes do Próximo! Deu Certo? .

?- decypherVigenereListHints("ÇkfiþøerapéCaijppónhovuCagpbÊBôájoé[aGfwoXfuuqf", 6, ["pizza","programação","finalmentes"], Texto).
Texto = Chegando nos finalmentes do Próximo! Deu Certo? .
```

Figura 5: Exemplo com os predicados pedidos

5 Instruções

Para popular o banco de dados utilize o comando: `swipl main.pl < seed.txt`

Para executar: `swipl main.pl`