Roteiro 7

Alexsandro Santos Soares
prof.asoares@gmail.com

Programação Lógica
Faculdade de Computação
Universidade Federal de Uberlândia

10 de julho de 2022

Este roteiro tem por finalidades:

- Praticar o uso de predicados para manipulação de arquivos.
- Familiarizá-lo com DCGs simples.
- Ilustrar o uso de predicados pré-construídos para imprimir termos na tela.

Ao fazer os exercícios não use qualquer predicado pré definido ou de alguma biblioteca que resolva diretamente o problema pedido.

1 Exercícios envolvendo arquivos

Ex. 1 Escreva um programa que leia um arquivo texto palavra por palavra e insira todas as palavras lidas juntamente com suas frequências na memória do Prolog. Use um predicado dinâmico `palavra/2` para armazenar as palavras, com o primeiro argumento é uma palavra e o segundo é a frequência desta palavra no arquivo. O predicado principal será chamado de `histograma/1` e recebe como argumento o nome do arquivo. O programa deve ignorar sinais de pontuação e espaços em branco.

Como exemplo, suponha que o conteúdo abaixo pertença ao arquivo `arq1.txt`:

```
tinha suspirado, tinha beijado o papel devotamente! Era a primeira vez que
lhe escreviam aquelas sentimentalidades, e o seu orgulho dilatava-se ao
calor amoroso que saía delas, como um corpo ressequido que se estira num
banho tépido; sentia um acréscimo de estima por si mesma, e parecia-lhe que
entrava enfim numa existência superiormente interessante, onde cada hora
tinha o seu encanto diferente, cada passo condizia a um êxtase, e a alma
se cobria de um luxo radioso de sensações!
```

Então, as consultas abaixo deveriam ser verdadeiras para ele:

```
?- histograma('arq1.txt').
true

?- palavra(a, Freq).
Freq=3

?- palavra(acréscimo, Freq).
Freq=1

?- palavra(cada, Freq).
Freq=2

?- palavra(tinha, Freq).
Freq=3

?- palavra(um, Freq).
Freq=4

?- palavra(devotamente, Freq).
Freq=1
```

Ex. 2 Escreva um predicado `copia_arq/2` que copia o conteúdo de um arquivo para outro. O primeiro argumento é o nome do arquivo fonte e o segundo é o nome do arquivo de destino.

Após a consulta abaixo deverão existir dois arquivos: `arq1.txt`, o arquivo fonte; e o arquivo `cópia.txt` que é uma cópia exata de `arq1.txt`.

```
?- copia_arq('arq1.txt', 'cópia.txt').
true
```

Ex. 3 Escreva um predicado `idênticos/2` que compara o conteúdo de dois arquivos e decide se eles são idênticos. Como exemplo, considere `arq1.txt` e `cópia.txt` do exercício anterior. A consulta a seguir deve ser verdadeira, pois um arquivo é uma cópia exata do outro.

```
?- idênticos('arq1.txt', 'cópia.txt').
true
```

Se os arquivos diferirem em algum caracter, o predicado deve falhar. Lembre-se de sempre fechar os arquivos após abrí-los para qualquer que seja o uso.

2 Exercícios envolvendo DCGs

O propósito desta parte é ajudá-lo a se familiarizar com DCGs, diferenças de listas e a relação entre ambas.

Começamos com alguns exercícios práticos:

Ex. 4 Primeiro digite o reconhecedor simples baseado em `append`, discutido em sala de aula, e depois execute alguns rastreamentos. Como você descobrirá, não exageramos ao dizer que a performance da gramática baseada em `append` era muito pobre. Mesmo para sentenças

simples como *A mulher chuta o homem*, você verá que o rastreamento é muito longo e muito difícil de seguir.

Ex. 5 Depois, digite o segundo reconhecedor, aquele baseado em listas de diferenças, e execute mais rastreamentos. Como você verá, existe um ganho dramático em eficiência. Além disso, mesmo se você acha a ideia de listas de diferenças um pouco difícil de seguir, você verá que os rastreamentos são *muito* simples de entender, especialmente quando comparados aos monstros produzidos pela implementação baseada em `append`.

Ex. 6 Na sequência, digite a DCG discutida na aula. Digite `listing` para ver o resultado da tradução feita pelo Prolog das regras DCGs. Como o seu sistema traduz regras da forma `Det --> [o]`?

Ex. 7 Agora execute alguns rastreamentos. Exceto pelos nomes das variáveis, os rastreamentos que você observará aqui deveriam ser muito similares àqueles observados quando executava o rastreador baseado em listas de diferenças.

E agora é hora de escrever algumas DCGs:

Ex. 8 A linguagem formal *aPar* é muito simples: ela consiste em todas as strings contendo um número par de `a`s e nada mais. Note que a string vazia ϵ pertence a *aPar*. Escreva uma DCG que gere *aPar*.

Ex. 9 A linguagem que os lógicos chamam de *lógica proposicional sobre os símbolos proposicionais p, q e r* pode ser definida pela seguinte gramática livre de contexto

$$\begin{aligned}
prop &\rightarrow p \\
prop &\rightarrow q \\
prop &\rightarrow r \\
prop &\rightarrow \neg prop \\
prop &\rightarrow (prop \wedge prop) \\
prop &\rightarrow (prop \vee prop) \\
prop &\rightarrow (prop \Rightarrow prop)
\end{aligned}$$

Escreva uma DCG que gere esta linguagem. De fato, como ainda não aprendemos sobre operadores Prolog, você terá que fazer algumas concessões que parecerão bastante desajeitadas. Por exemplo, ao invés de reconhecer

$$\neg(p \Rightarrow q)$$

você terá que reconhecer coisas como

```
[não, '(', p, implica, q, ')']
```

Use `ou` para $\vee$, e `e` para $\wedge$.

Ex. 10 Dada a seguinte DCG:

```
s --> foo,bar,wiggle.
foo --> [chu].
foo --> foo,foo.
```

```
bar --> mar,zar.
mar --> me,my.
me --> [eu].
my --> [sou].
zar --> blar,car.
blar --> [um].
car --> [trem].
wiggle --> [tchu].
wiggle --> wiggle,wiggle.
```

Escreva as regras Prolog comuns que correspondam a estas regras DCGs. Quais são as primeiras três respostas que o Prolog dá à consulta `s(X,[])`?

Ex. 11 A linguagem formal $a^n b^n - \{\epsilon\}$ consiste de todas as strings $a^n b^n$, exceto a string vazia. Escreva uma DCG que gere esta linguagem.

Ex. 12 Seja $a^n b^{2n}$ a linguagem formal que contém todas as strings da seguinte forma: um bloco contíguo de `a`s de tamanho n seguido por um bloco contíguo de `b`s de tamanho $2n$, e nada mais. Por exemplo, *abb*, *aabbbb* e *aaabbbbbb* pertencem a $a^n b^{2n}$, assim como a string vazia. Escreva uma DCG que gere esta linguagem.

3 Predicados para impressão de termos

Nesta sessão prática, pretendemos introduzir alguns predicados pré-construídos para imprimir termos na tela.

O primeiro predicado a ser visto é `display/1`, que recebe um termo e o imprime na tela.

```
?- display(ama(vicente,maria)).
ama(vicente,maria)
true.

?- display('julio come um grande sanduíche').
julio come um grande sanduíche
true.
```

Mais estritatemente falando, `display` imprime a representação interna do Prolog para termos.

```
?- display(2+3+4).
+(+(2,3),4)
true.
```

De fato, esta propriedade de `display` torna-o muito útil para aprender como operadores funcionam em Prolog. Assim, antes de aprender mais como escrever coisas na tela, tente as seguintes consultas. Assegure-se de compreender porque o Prolog responde da forma que ele faz.

```
?- display([a,b,c]).
?- display(3 is 4 + 5 / 3).
?- display(3 is (4 + 5) / 3).
?- display((a:-b,c,d)).
?- display(a:-b,c,d).
```

Assim, `display` é bom para olhar na representação interna dos termos na notação de operadores, mas normalmente nós provavelmente preferiríamos imprimir na notação mais amigável. Especialmente na impressão de listas, seria muito melhor ter `[a,b,c]` do que `'[|]'(a'[|]'(b'[|]'(c,[])))`. Isto é o que faz o predicado pré-construído `write/1`. Ele recebe um termo e o imprime na tela usando a notação mais amigável.

```
?- write(2+3+4).
2+3+4
true

?- write(+(2,3)).
2+3
true

?- write([a,b,c]).
[a, b, c]
true

?- write('[|]'(a,'[|]'(b,[]))).
[a, b]
true
```

E aqui está o que acontece quando o termo a ser escrito contém variáveis.

```
?- write(X).
_G204
true

?- X = a, write(X).
a
X = a.
```

O exemplo seguinte mostra o que acontece quando você coloca dois comandos write, um após o outro.

```
?- write(a), write(b).
ab
true
```

Prolog apenas executa um após o outro sem colocar qualquer espaço entre a saída dos diferentes comandos write. Naturalmente, você pode dizer ao Prolog para imprimir espaços pedindo para escrever o termo `' '`:

```
?- write(a), write(' '), write(b).
a b
```

E se você quiser mais que um espaço, por exemplo cinco espaços, diga ao Prolog para escrever `' '`:

```
?- write(a), write('     '), write(b).
a     b
```

Uma outra forma de imprimir espaços é pelo uso do predicado `tab/1`. Este predicado recebe um número como argumento e então imprime tantos espaços quanto especificado por este número.

```
?- write(a), tab(5), write(b).
a     b
```

Um outro predicado útil para formatação é `nl`. Este predicado diz ao Prolog para saltar uma linha:

```
?- write(a), nl, write(b).
a
b
```

4 Sugestões de leitura

- Luiz Gustavo Almeida Martins. *Lógica para Computação.*

  http://www.facom.ufu.br/~gustavo/Logica/Logica.html

- Luiz A. M. Palazzo. *Introdução à programação Prolog*

  http://puig.pro.br/Logica/palazzo.pdf

- Eloi L. Favero. *Programação em Prolog: uma abordagem prática*

  http://www3.ufpa.br/favero

- Wikilivro sobre Prolog em

  http://pt.wikibooks.org/wiki/Prolog

- Patrick Blackburn, Johan Bos and Kristina Striegnitz. *Learn Prolog Now!*

  http://www.learnprolognow.org