Roteiro 5

Alexsandro Santos Soares
prof.asoares@gmail.com

Programação Lógica
Faculdade de Computação
Universidade Federal de Uberlândia

15 de janeiro de 2022

Este roteiro tem por finalidades:

- Familiarizá-lo com DCGs simples e com aquelas que utilizam argumentos e testes adicionais.
- Ilustrar o uso de predicados pré-construídos para imprimir termos na tela.

1 Exercícios envolvendo DCGs

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

Começamos com alguns exercícios práticos:

Ex. 1 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. 2 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. 3 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. 4 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. 5 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. 6 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. 7 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. 8 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. 9 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.

2 Gramáticas de cláusulas definidas com argumentos e testes extras

Primeiro alguns exercícios de fixação:

Ex. 10 Rastreie alguns exemplos de DCG que utilizem argumentos extras para tratar a distinção sujeito/objeto, de DCG que produza análise sintática e de DCG que utilize testes extras para separar o léxico das regras. Certifique-se de que você compreenda totalmente o modo com o qual todas as três DCGs funcionam. Use os slides da aula teórica dessa semana.

Ex. 11 Realize alguns rastreamentos para a DCG para a linguagem $a^n b^n c^n$. Experimente casos onde os três blocos de as, bs e cs sejam de fato do mesmo tamanho, assim como casos onde isto não ocorre.

Agora alguns exercícios para treinar a técnica.

Ex. 12 A linguagem formal $a^n b^{2m} c^{2m} d^n$ consiste de todas as strings da seguinte forma: um bloco contíguo de `a`s seguido por um bloco contíguo de `b`s, seguido por um bloco contíguo de `c`s, seguido por um bloco contíguo de `d`s, tal que os blocos `a` e `d` são exatamente do mesmo tamanho, e os blocos `b` e `c` são exatamente do mesmo tamanho e, além disto, consistem de um número par de `b`s e `c`s. Por exemplo, ϵ, *abbccd* e *aaabbbbccccddd* pertecem a $a^n b^{2m} c^{2m} d^n$. Escreva uma DCG que gere esta linguagem.

Ex. 13 Abaixo encontra-se a nossa DCG básica.

```
s --> sn, sv.

sn --> det, n.

sv --> v, sn.
sv --> v.

det --> [o].
det --> [a].

n --> [mulher].
n --> [homem].

v --> [bate].
```

Suponha que adicionemos o nome "homens", que é plural, e o verbo "batem". Então, gostaríamos de uma DCG que diga que "O homem bate" está correto, "Os homens batem" está correto, "O homem batem" não está correto e que "Os homens bate" também não está correto. Altere a DCG tal que ela corretamente trate estas sentenças. Use um argumento extra para lidar com a distinção singular/plural.

Ex. 14 Traduza a seguinte regra DCG em um formato padrão de regras do Prolog:

```
cangu(V,R,Q) --> ru(V,R), salta(Q,Q), {marsupial(V,R,Q)}.
```

Finalmente, alguns exercícios de programação.

Ex. 15 Primeiro, reúna todas as coisas que aprendeu sobre DCGs para Português em uma única DCG. Em particular, nessa semana vimos como usar argumentos extras para lidar com a distinção sujeito/objeto, e nos exercícios anteriores você usou argumentos adicionais para lidar com a distinção singular/plural. Escreva uma DCG que trate ambos. Além disto, escreva a DCG de tal forma que ela produza árvores sintáticas e faça uso de um léxico separado.

Ex. 16 Escreva uma DCG que reconheça numerais cardinais escritos por extenso em Português, entre zero e mil.

```
?- cardinal([zero], []).
true

?- cardinal([vinte,e,um], []).
true

?- cardinal([novecentos,e,trinta,e,sete], []).
true

?- cardinal([setecentos, e, setenta, e, sete], []).
true.

?- cardinal([mil], []).
true.

?- cardinal([cem,onze], []).
false.
```

Ex. 17 Modifique o reconhecedor do exercício anterior para que ele também produza os dígitos do número reconhecido em uma lista.

```
?- cardinal(N, [zero], []).
N = [0] .

?- cardinal(N, [vinte,e,um], []).
N = [2, 1] .

?- cardinal(N, [novecentos,e,trinta,e,sete], []).
N = [9, 3, 7] .

?- cardinal(N, [novecentos,e,trinta], []).
N = [9, 3, 0] .

?- cardinal([7,7,7], Extenso, []).
Extenso = [setecentos, e, setenta, e, sete] .

?- cardinal([1,0,0,0], Extenso, []).
Extenso = [mil].
```

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 Exercícios

Ex. 18 Quais das seguintes consultas tem sucesso e quais falham?

```
?- 12 is 2*6.

?- 14 =\= 2*6.

?- 14 = 2*7.

?- 14 == 2*7.
```

```
?- 14 \== 2*7.

?- 14 =:= 2*7.

?- [1,2,3|[d,e]] == [1,2,3,d,e].

?- 2+3 == 3+2.

?- 2+3 =:= 3+2.

?- 7-2 =\= 9-2.

?- p == 'p'.

?- p =\= 'p'.

?- vicente == VAR.

?- vicente=VAR, VAR==vicente.
```

5 Sugestões de leitura

- Luiz Arthur Pagani. (2004). *Analisador gramatical em Prolog para gramáticas de estrutura sintagmática*. Revista Virtual de Estudos da Linguagem -ReVEL. Ano 2, n. 3. Disponível em:

  http://www.revel.inf.br/files/artigos/revel_3_analisador_gramatical.pdf

- 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