

Roteiro 5

Alexsandro Santos Soares

`prof.asoares@gmail.com`

Programação Lógica
Faculdade de Computação
Universidade Federal de Uberlândia

2 de abril de 2021

Este roteiro tem por finalidades:

- Familiarizá-lo com DCGs simples e com aquelas que utilizam argumentos e testes adicionais.

1 Exercícios envolvendo DCG

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 as e nada mais. Note que a string vazia ϵ pertence a $aPar$. Escreva uma DCG que gere $aPar$.
- Ex. 6** A linguagem formal $a^n b^{2m} c^{2m} d^n$ consiste de todas as strings da seguinte forma: um bloco contíguo de as seguido por um bloco contíguo de bs, seguido por um bloco contíguo de cs, seguido por um bloco contíguo de ds, 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 bs e cs. Por exemplo, ϵ , $abbccd$ e $aaabbbbccccddd$ pertencem a $a^n b^{2m} c^{2m} d^n$. Escreva uma DCG que gere esta linguagem.
- Ex. 7** 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. 8** 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. 9** 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. 10** Seja $a^n b^{2n}$ a linguagem formal que contém todas as strings da seguinte forma: um bloco contíguo de as de tamanho n seguido por um bloco contíguo de bs de tamanho $2n$, e nada mais. Por exemplo, abb , $aabbbb$ e $aaabbbbbbb$ 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. 11** 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. 12** 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. 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 entre zero e mil escritos por extenso em Português.

```
?- 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 Sugestões de leitura

- Leia os capítulos do livro de Eloi Favero referentes ao tema discutido nesta prática.
- Leia o wikilivro sobre Prolog em <http://pt.wikibooks.org/wiki/Prolog>.