Teoria dos Autômatos e Linguagens Formais

aula 20: Gramáticas

1 Introdução

As nossas primeiras ideias sobre os padrões recursivos apareceram quando nós fizemos a seguinte observação sobre a linguagem PARBAL

- *As palavras de* PARBAL *podem ter uma outra palavra de* PARBAL *em seu interior*

Por exemplo,

$$\begin{array}{c} (\ ) \\ (\ ) \\ (\ ) \end{array} \quad \Rightarrow \quad \begin{array}{c} (\ . \ . \ . \ . \ . \ . \ ) \\ (\ . \ . \ . \ ) \\ (\ ) \end{array} \quad \Rightarrow \quad (\ (\ (\ ) \ ) \ )$$

Hoje nós vamos levar essa ideia adiante para obter o conceito de gramática.

2 Gramáticas

Nós podemos codificar o padrão recursivo da linguagem PARBAL da seguinte maneira[1]

$$x \quad = \quad (\ x \ )$$

Quer dizer, a ideia é que dos dois lados o x significa *"uma palavra de* PARBAL".

E daí, o que a equação está dizendo é

- *um palavra de* PARBAL *pode ter uma outra palavra de* PARBAL *em seu interior*

Mas, nós também podemos interpretar a equação de maneira ligeiramente diferente.

Quer dizer, nós podemos interpretar a equação como uma *regra de substituição*

- toda vez que a gente encontra o x, nós podemos substituí-lo pelo bloco (x)

$$x \quad \rightarrow \quad (\ x \ )$$

E agora, a regra de substituição se torna um mecanismo para construir palavras de PARBAL.

Quer dizer, começando com a variável x e aplicando a regra uma vez, nós obtemos a palavra

$$(\ x \ )$$

[1] Note como essa equação é semelhante ao código de um procedimento recursivo

```
proc F()
 {    ....
      F();
      ....
 }
```

Em seguida, aplicando a regra outra vez, nós obtemos a palavra

$$(\; (\; x \;) \;)$$

e assim por diante.

Mas, a regra $x \to (\; x \;)$ não é ainda um mecanismo completo gerador de palavras de PARBAL.

Nós precisamos de uma regra para eliminar o x da expressão

$$x \quad \to \quad \texttt{e}$$

E nós precisamos de uma regra que nos permitar produzir padrões repetitivos

$$x \quad \to \quad x \, x$$

Uma *gramática* é basicamente uma coleção de regras de substituição que gera as palavras de uma linguagem

$$\begin{aligned} x \quad &\to \quad (\; x \;) \\ x \quad &\to \quad x \, x \\ x \quad &\to \quad \texttt{e} \end{aligned}$$

Em geral, as gramáticas podem ter diversas variáveis diferentes, e os livros utilizam a letra S para indicar onde tudo começa.

Abaixo nós temos a gramática da linguagem PARBAL.

1. $S \quad \to \quad (\; S \;)$
2. $S \quad \to \quad S \, S$
3. $S \quad \to \quad \texttt{e}$

E abaixo nós temos a *derivação* (produção, geração, construção, etc.) de uma palavra de PARBAL utilizando as regras da gramática

$$\texttt{S} \overset{(2)}{\Longrightarrow} \texttt{S S} \overset{(1)}{\Longrightarrow} \texttt{(S)S} \overset{(3)}{\Longrightarrow} \texttt{()S} \overset{(1)}{\Longrightarrow} \texttt{()(S)} \overset{(1)}{\Longrightarrow} \texttt{()((S))} \overset{(3)}{\Longrightarrow} \texttt{()(())}$$

A seguir, nós veremos outros exemplos de gramáticas.

Exemplos

a. A linguagem $\texttt{a}^\texttt{n}\texttt{b}^\texttt{n}$, e suas variantes

Abaixo nós temos a gramática da linguagem $\texttt{a}^\texttt{n}\texttt{b}^\texttt{n}$

1. $S \quad \to \quad \texttt{a} \; S \; \texttt{b}$
2. $S \quad \to \quad \texttt{e}$

que pode ser apresentada de maneira ainda mais sucinta como

$$S \quad \rightarrow \quad \texttt{a}\ S\ \texttt{b} \ \mid\ \texttt{e}$$

(isto é, a variável S pode ser substituída por $\texttt{a}S\texttt{b}$ ou $\texttt{e}$)

A primeira variante da linguagem $\texttt{a}^{\texttt{n}}\texttt{b}^{\texttt{n}}$ que nós vamos considerar é a linguagem

$$\texttt{a}^{\texttt{n}}\texttt{b}^{\texttt{2n}}$$

que pode ser gerada pela gramática

$$S \quad \rightarrow \quad \texttt{a}\ S\ \texttt{bb} \ \mid\ \texttt{e}$$

A seguir, nós consideramos a variante

$$\texttt{a}^{\texttt{n}}\texttt{b}^{\texttt{m}} \quad \text{, onde } \texttt{n} = \texttt{2m} \text{ ou } \texttt{m} = \texttt{2n}$$

Isto é, aqui nós temos o conjunto de palavras da forma $\texttt{a..ab..b}$, onde um dos blocos tem o dobro do tamanho do outro.

Uma ideia seria ter uma gramática para cada caso

$$S_1 \quad \rightarrow \quad \texttt{aa}\ S_1\ \texttt{b} \ \mid\ \texttt{e} \qquad\qquad S_2 \quad \rightarrow \quad \texttt{a}\ S_2\ \texttt{bb} \ \mid\ \texttt{e}$$

E a seguir nós adicionamos a regra

$$S \quad \rightarrow \quad S_1 \ \mid\ S_2$$

(Em certo sentido, aqui nós temos uma gramática definida como a união de duas gramáticas mais simples.)

A próxima variante que nós vamos considerar é a seguinte

$$\texttt{a}^{\texttt{n}}\texttt{b}^{\texttt{m}} \quad \text{, onde } \frac{\texttt{n}}{2} \ \leq\ \texttt{m} \ \leq\ \texttt{2n}$$

Uma primeira ideia para obter a gramática que gera essa linguagem seria aplicar alternadamente as regras

$$S_1 \quad \rightarrow \quad \texttt{aa}\ S_1\ \texttt{b} \ \mid\ \texttt{e} \qquad\qquad S_2 \quad \rightarrow \quad \texttt{a}\ S_2\ \texttt{bb} \ \mid\ \texttt{e}$$

E uma maneira simples de conseguir isso é

$$S \quad \rightarrow \quad S_1 \ \mid\ S_2 \ \mid\ \texttt{e}$$
$$S_1 \quad \rightarrow \quad \texttt{aa}\ S\ \texttt{b}$$
$$S_2 \quad \rightarrow \quad \texttt{a}\ S\ \texttt{bb}$$

Por exemplo, a palavra $\texttt{aaabbb}$ seria derivada da seguinte maneira

$$\texttt{S} \implies \texttt{S}_1 \implies \texttt{a S bb} \implies \texttt{a}\,\texttt{S}_2\,\texttt{bb} \implies \texttt{aaa S bbb} \implies \texttt{aaabbb}$$

Mas, essa gramática não é capaz de produzir, por exemplo, as palavras $\texttt{ab}$ e $\texttt{aabb}$.

Abaixo nós temos uma solução simples para esse problema

$$S \quad \rightarrow \quad S_1 \ \mid\ S_2 \ \mid\ S_3 \ \mid\ \texttt{e}$$

$$
\begin{array}{lcl}
S_1 & \rightarrow & \texttt{aa}\ S\ \texttt{b} \\
S_2 & \rightarrow & \texttt{a}\ S\ \texttt{bb} \\
S_3 & \rightarrow & \texttt{a}\ S\ \texttt{b}
\end{array}
$$

Finalmente, a última variante que nós vamos considerar é a seguinte

$$\texttt{a}^{\texttt{n}}\texttt{b}^{\texttt{m}} \quad \text{, onde } \texttt{m} > \texttt{n}$$

Aqui nós podemos raciocinar de duas maneiras.

A primeira delas consiste em ver as palavras dessa linguagem como a concatenação

$$\left(\texttt{a}^{\texttt{n}}\texttt{b}^{\texttt{n}}\right)\texttt{b}^{\texttt{k}} \quad \text{, onde } \texttt{k} > \texttt{0}$$

E daí nós obtemos a gramática

$$
\begin{array}{lcl}
S & \rightarrow & S_1\ B \\
S_1 & \rightarrow & \texttt{a}\ S\ \texttt{b}\ \mid\ \texttt{e} \\
B & \rightarrow & \texttt{b}\ B\ \mid\ \texttt{b}
\end{array}
$$

A segunda maneira consiste em ver as palavras da linguagem como

$$\texttt{a}^{\texttt{n}}\left(\texttt{b}^{\texttt{k}}\right)\texttt{b}^{\texttt{n}} \quad \text{, onde } \texttt{k} > \texttt{0}$$

E daí nós obtemos a gramática

$$
\begin{array}{lcl}
S & \rightarrow & \texttt{a}\ S\ \texttt{b}\ \mid\ B \\
B & \rightarrow & \texttt{b}\ B\ \mid\ \texttt{b}
\end{array}
$$

◇

b. PAL, BAL e DOB

Nós já conhecemos o padrão recursivo da linguagem PAL:

- existem 4 blocos básicos: `aa`, `bb`, `a`, `b`
- e as palavras mais complexas são obtidas inserindo um bloco no meio de outro bloco

As regras abaixo implementam esse procedimento de construção recursiva

$$
\begin{array}{lcl}
S & \rightarrow & \texttt{a}\ S\ \texttt{a}\ \mid\ \texttt{b}\ S\ \texttt{b} \\
S & \rightarrow & \texttt{aa}\ \mid\ \texttt{bb}\ \mid\ \texttt{a}\ \mid\ \texttt{b}\ \mid\ \texttt{e}
\end{array}
$$

A linguagem BAL é semelhante, mas

- ela possui apenas 2 blocos básicos: `ab` e `ba`
- e ela também envolve o padrão repetitivo

Isso nos leva à seguinte gramática

$$
\begin{array}{lcl}
S & \rightarrow & \texttt{a}\ S\ \texttt{b}\ \mid\ \texttt{b}\ S\ \texttt{a}\ \mid\ S\ S \\
S & \rightarrow & \texttt{ab}\ \mid\ \texttt{ba}\ \mid\ \texttt{e}
\end{array}
$$

Mas, pensando de maneira diferente, nós também podemos chegar à seguinte gramática

$$\begin{array}{lcl} S & \to & \texttt{a}B \\ S & \to & \texttt{b}A \\ S & \to & \texttt{e} \end{array} \qquad \begin{array}{lcl} A & \to & \texttt{a} \\ A & \to & \texttt{a}S \\ A & \to & BAA \end{array} \qquad \begin{array}{lcl} B & \to & \texttt{b} \\ B & \to & \texttt{b}S \\ B & \to & ABB \end{array}$$

(*Você consegue ver que essa gramática também gera a linguagem* BAL*?*)

(*Você consegue explicar porque?*)

A seguir, nós vamos considerar a seguinte linguagem

DOB = palavras onde um dos símbolos aparece o dobro de vezes que o outro

Para obter a gramática que gera a linguagem DOB, nós começamos quebrando o problema

$$\text{DOB} \quad = \quad \text{DOB-a} \ \cup \ \text{DOB-b}$$

onde DOB-a (DOB-b) é o conjunto das palavras ondo o número de a's (b's) é o dobro do número de b's (a's).

Daí, a ideia é que nós vamos ter a regra

$$S \quad \to \quad S_1 \ \mid \ S_2$$

onde a variável S_1 produz as palavras de DOB-a e S_2 produz as palavras de DOB-b.

Agora basta construir a gramática de DOB-a — pois a gramática de DOB-b é análoga.

Vejamos.

As palavras mais simples de DOB-a são

$$\texttt{a a b}, \quad \texttt{a b a}, \quad \texttt{b a a}, \quad \text{(além da palavra vazia e)}$$

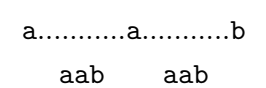
e isso já nos dá os blocos básicos da construção recursiva.

Em seguida, nós observamos que existem dois lugares onde um bloco pode ser inserido dentro de outro bloco

Isso corresponde às seguintes regras de substituição

$$S \quad \to \quad \texttt{a}\,S\,\texttt{ab} \ \mid \ \texttt{aa}\,S\,\texttt{b}$$

Mas, nós também podemos inserir dois blocos no interior de um único bloco



o que corresponde à regra

$$S \quad \to \quad \texttt{a}\,S\,\texttt{a}\,S\,\texttt{b}$$

Finalmente, nós observamos que a linguagem DOB-a também envolve o padrão repetitivo, que é implementado pela regra

$$S \quad \to \quad S\,S$$

Abaixo nós temos a gramática completa de DOB-a

$$\begin{aligned} S &\to \mathtt{a}\,S\,\mathtt{a}\,S\,\mathtt{b} \mid \mathtt{a}\,S\,\mathtt{b}\,S\,\mathtt{a} \mid \mathtt{b}\,S\,\mathtt{a}\,S\,\mathtt{a} \\ S &\to S\,S \\ S &\to \mathtt{aab} \mid \mathtt{aba} \mid \mathtt{baa} \mid \mathtt{e} \end{aligned}$$

É possível reduzir o número de regras da gramática da seguinte maneira

$$\begin{aligned} S &\to \mathtt{a}\,S\,\mathtt{a}\,S\,\mathtt{b}\,S \mid \mathtt{a}\,S\,\mathtt{b}\,S\,\mathtt{a}\,S \mid \mathtt{b}\,S\,\mathtt{a}\,S\,\mathtt{a}\,S \\ S &\to \mathtt{e} \end{aligned}$$

(*Você consegue ver?*)

◇

c. $\mathtt{a}^{n}\,\mathtt{b}\,\mathtt{a}^{n+m}\,\mathtt{b}\,\mathtt{a}^{m}$

Como nós vimos na aula 19, nós podemos decompor as palavras dessa linguagem da seguinte maneira

$$\left(\mathtt{a}^{n}\mathtt{b}\mathtt{a}^{n}\right)\left(\mathtt{a}^{m}\mathtt{b}\mathtt{a}^{m}\right)$$

isto é, uma concatenação de duas palavras da linguagem $\mathtt{a}^{n}\mathtt{b}\mathtt{a}^{n}$.

Logo, a primeira regra da gramática é

$$S \quad \to \quad T\,T$$

E a seguir, nós escrevemos as regras que permitem derivar uma palavra de $\mathtt{a}^{n}\mathtt{b}\mathtt{a}^{n}$ a partir da variável T.

As regras são as seguintes

$$\begin{aligned} T &\to \mathtt{a}\,T\,\mathtt{a} \\ T &\to \mathtt{b} \end{aligned}$$

A seguir, nós consideramos a seguinte variante

$$\mathtt{a}^{n}\,\mathtt{b}\,\mathtt{a}^{m}\,\mathtt{b}\,\mathtt{a}^{m+n}$$

Mais uma vez, nós lembramos que as palavras dessa linguagem podem ser decompostas como

$$\mathtt{a}^{n}\mathtt{b}\left(\mathtt{a}^{m}\mathtt{b}\mathtt{a}^{m}\right)\mathtt{a}^{n}$$

isto é, uma palavra da linguagem $\mathtt{a}^{n}\mathtt{b}\mathtt{a}^{n}$ dentro da outra.

Logo, a ideia é realizar a construção recursiva da palavra mais externa e, no momento em que a construção é encerrada, nós iniciamos a construção da palavra mais interna.

Abaixo nós temos as regras da gramática

$$S \quad \to \quad \mathtt{a}\,S\,\mathtt{a}$$

$$
\begin{array}{lcl}
S & \to & \mathtt{b}\,T \\
T & \to & \mathtt{a}\,T\,\mathtt{a} \\
T & \to & \mathtt{b}
\end{array}
$$

$\diamond$

d. Uma linguagem regular

Considere a linguagem

- palavras que possuem um bloco de a's de tamanho ímpar

Nós sabemos que isso é uma linguagem regular.

Logo, nós podemos escrever uma expressão regular que descreve essas palavras

$$\Big((\mathtt{a} \cup \mathtt{b})^*\mathtt{b} \cup \mathtt{e}\Big)\ \mathtt{a}(\mathtt{aa})^*\ \Big(\mathtt{b}(\mathtt{a} \cup \mathtt{b})^* \cup \mathtt{e}\Big)$$

E agora não é difícil construir a gramática.

O primeiro passo é decompor a expressão regular em 3 partes

$$\Big(\underbrace{(\mathtt{a} \cup \mathtt{b})^*\mathtt{b} \cup \mathtt{e}}\Big)\ \underbrace{\mathtt{a}(\mathtt{aa})^*}\ \Big(\underbrace{\mathtt{b}(\mathtt{a} \cup \mathtt{b})^* \cup \mathtt{e}}\Big)$$

E daí nós podemos ter variáveis T_1, T_2 e T_3 que produzem palavras descritas por cada parte.

As regras de T_1 são

$$
\begin{array}{lcl}
T_1 & \to & Q\,\mathtt{b} \mid \mathtt{e} \\
Q & \to & \mathtt{a}\,Q \mid \mathtt{b}\,Q \mid \mathtt{e}
\end{array}
$$

Quer dizer, a variável Q produz uma sequência qualquer de a's e b's, e essa sequência está separada do bloco ímpar de a's (na parte 2) por um símbolo b.

Abaixo nós temos as regras de T_2

$$T_2 \to \mathtt{aa}\,T_2 \mid \mathtt{a}$$

E abaixo nós temos as regras de T_3

$$T_3 \to \mathtt{b}\,Q \mid \mathtt{e}$$

onde as regras de Q já foram dadas.

A seguir, nós vamos considerar a linguagem

- palavras que possuem dois blocos de a's do mesmo tamanho

Qual é a ideia aqui?

Bom, nós precisamos gerar os dois blocos de a's que vão ter o mesmo tamanho simultaneamente — para garantir que eles vão ter o mesmo tamanho.

Isso pode ser feito com as regras

$$
\begin{array}{lcl}
S & \to & \mathtt{a}\,S_1\,\mathtt{a} \\
S_1 & \to & \mathtt{a}\,S_1\,\mathtt{a}
\end{array}
$$

Daí, quando a construção dos blocos termina, nós podemos colocar qualquer coisa entre eles

(menos um `a` ou a palavra vazia).

No caso mais simples, existe apenas um `b` entre os dois blocos, o que corresponde à regra

$$S_1 \quad \rightarrow \quad \texttt{b}$$

E, no caso geral, pode haver qualquer coisa (separada dos blocos de `a`'s por um símbolo `b`), o que corresponde à regra

$$S_1 \quad \rightarrow \quad \texttt{b}\, Q\, \texttt{b}$$

Finalmente, nós lembramos que pode aparecer qualquer coisa antes do primeiro bloco e depois do segundo bloco.

Logo, nós modificamos a primeira regra para

$$S \quad \rightarrow \quad T_1\, \texttt{a}\, S_1\, \texttt{a}\, T_3$$

onde T_1 e T_3 são variáveis do exemplo anterior.

Abaixo nós temos a gramática completa da linguagem

$$\begin{aligned}
S &\rightarrow T_1\, \texttt{a}\, S_1\, \texttt{a}\, T_3 \\
S_1 &\rightarrow \texttt{a}\, S_1\, \texttt{a} \mid \texttt{b}\, Q\, \texttt{b} \mid \texttt{b} \\
T_1 &\rightarrow Q\, \texttt{b} \mid \texttt{e} \\
T_3 &\rightarrow \texttt{b}\, Q \mid \texttt{e} \\
Q &\rightarrow \texttt{a}\, Q \mid \texttt{b}\, Q \mid \texttt{e}
\end{aligned}$$

◇