**Teoria dos Autômatos e Linguagens Formais**

**aula 22: Raciocínio recursivo com gramáticas**

## 1 Introdução

Consider a seguinte gramática

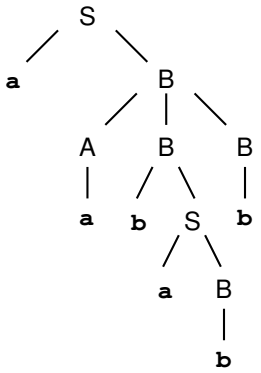
$S \rightarrow \texttt{a}B \mid \texttt{b}A \mid \texttt{e}$ $\qquad$ $B \rightarrow \texttt{b} \mid \texttt{b}S \mid ABB$

$A \rightarrow \texttt{a} \mid \texttt{a}S \mid BAA$

*Que palavras são geradas por essa gramática?*

A melhor maneira de descobrir isso consiste em examinar alguns exemplos do que se pode fazer com as suas regras.

Abaixo nós temos um desses exemplos

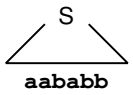



O esquema acima é chamado de *árvore de derivação*.

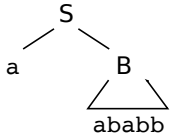
Em uma árvore de derivação, cada nó interno e seu conjunto de filhos corresponde à aplicação de uma regra da gramática. E o conjunto de folhas da árvore, percorrido da esquerda para a direita, corresponde à palavra que foi produzida nessa derivação: `aababb`.

As árvores de derivação oferecem uma maneira conveniente de ver como as diversas partes da palavra final são produzidas a partir de símbolos diferentes que aparecem na derivação.

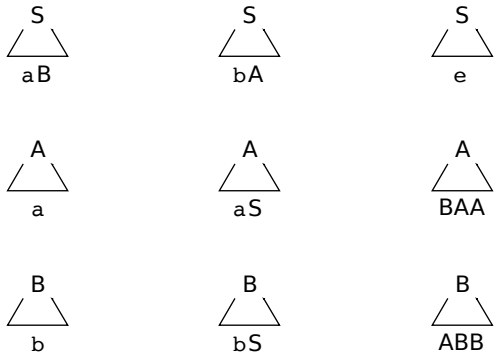
Por exemplo, nós podemos representar o fato de que a palavra final inteira é obtida a partir do símbolo inicial $S$ como



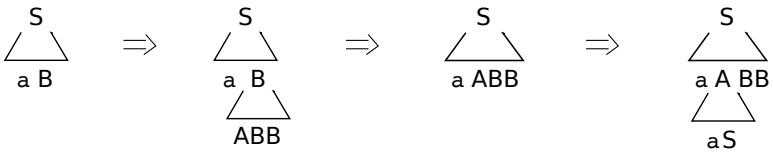

E para ver a parte da palavra que é produzida pelo símbolo $B$ que aparece na parte de cima da árvore, nós fazemos



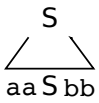
Esse esquema de representação também pode ser utilizado para apresentar as regras da gramática de maneira gráfica




A partir desse ponto de vista, a produção de palavras com as regras da gramática pode ser vista como um jogo de quebra-cabeças




E, brincando com as regras da gramática dessa maneira, nós chegamos eventualmente à seguinte observação:




que corresponde à regra $S \to \texttt{aa}\, S\, \texttt{bb}$.

Nesse ponto, nós começamos a desconfiar que a gramática que nós vimos no início corresponde a mais uma maneira de gerar a linguagem BAL.

Esse fato é comprovado com mais duas observações adicionais:

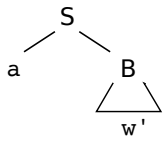
1. todas as regras da gramática de BAL que nós vimos na aula passada podem ser obtidas dessa maneira — portanto, essa gramática pode gerar todas as palavras de BAL.
2. as regras dessa gramática sempre produzem palavras com a mesma quantidade de $\texttt{a}, \texttt{A}$'s e $\texttt{b}, \texttt{B}$'s — portanto, essa gramática não gera palavras que não estão em BAL.

## 2 Raciocínio recursivo com gramáticas

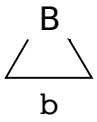
Agora que sabemos o que a gramática faz, é fácil descobrir a lógica por trás das suas regras.

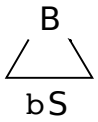
Considere a seguinte derivação

Sabendo que a palavra $\texttt{w} = \texttt{a}\,\texttt{w}'$ obtida nessa derivação é uma palavra de BAL, isso nos dá a dica de que o símbolo $B$ é responsável por produzir palavras que possuem exatamente um símbolo b a mais do que a's.

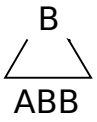



De fato, isso segue imediatamente das regras




E, assumindo que o símbolo $A$ produz palavras com exatamente um símbolo `a` a mais do que `b`'s, nós vemos que isso também é verdade para a regra




A seguir, nós vamos ver outros exemplos que exploram esse ponto de vista recursivo sobre as gramáticas.

**Exemplos**

a. Uma linguagem regular

Considere a linguagem

$$\mathtt{a}^*\big(\mathtt{aba} \cup \mathtt{ba}\big)^*(\mathtt{aa}^*\mathtt{b})^*$$

Note que qualquer palavra `w` dessa linguagem pode ser decomposta como

$$\mathtt{w} \;=\; \mathtt{w}'\,\mathtt{w}''\,\mathtt{w}'''$$

onde

$$\mathtt{w}' \in \mathtt{a}^* \qquad \mathtt{w}'' \in \big(\mathtt{aba} \cup \mathtt{ba}\big)^* \qquad \mathtt{w}''' \in (\mathtt{aa}^*\mathtt{b})^*$$

Agora é fácil definir as regras da gramática, utilizando símbolos intermediários para produzir cada uma das 3 partes da palavra:

$S \;\to\; R\,T\,U$
$R \;\to\; RR \mid \mathtt{a} \mid \mathtt{e}$
$T \;\to\; TT \mid \mathtt{aba} \mid \mathtt{ba} \mid \mathtt{e}$
$U \;\to\; UU \mid \mathtt{a}R\mathtt{b} \mid \mathtt{e}$

◇

b. PARBAL

Nós obtemos uma gramática para a linguagem PARbal a partir de três observações simples:

- A concatenação de duas palavras de PARbal também é uma palavra de PARbal, o que corresponde à regra

  $S \;\to\; S\,S$

- Ao colocar uma palavra de PARbal entre parênteses nós obtemos uma outra palavra de PARbal, o que corresponde à regra

  $S \;\to\; (\,S\,)$

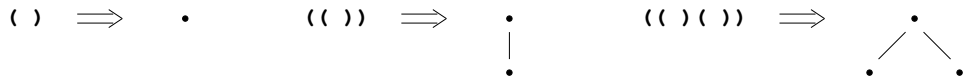
- A palavra vazia e é uma palavra de PARbal, o que corresponde à regra

  $S \;\to\; \mathsf{e}$

◇

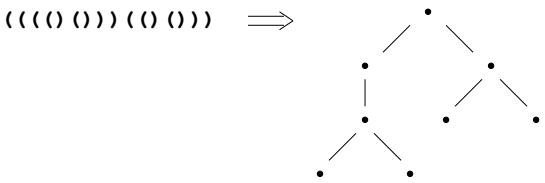
c. ARVBIN

Essa é a linguagem das árvores binárias codificadas na forma de expressões com parênteses balanceados. Por exemplo,




Note que, ARVbin é o subconjunto de PARbal onde todo par de parênteses contém 0, 1 ou 2 termos em seu interior.

No vocabulário das árvores binárias, o nó que fica no topo é chamado de *raiz* da árvore, e os nós que ficam logo abaixo são chamados de *filhos* da raiz. A observação importante, no entanto, é que os filhos da raiz podem ser eles mesmos raizes de outras árvores binárias. Veja o exemplo abaixo:




Essa observação nos dá imediatamente as regras da gramática que gera a linguagem ARVbin, analisando os quatro casos possíveis:

- árvore cuja raiz possui dois filhos:

  $S \;\to\; (\,S\,S\,)$

- árvore cuja raiz possui apenas um filho:

  $S \;\to\; (\,S\,)$

- árvore que possui apenas a raiz:

  $S \;\to\; (\,)$

- árvore vazia:

  $S \;\to\; \mathsf{e}$

**Nota:** Observe que a segunda e a terceira regras são desnecessárias, uma vez que elas podem ser simuladas por aplicações sucessivas da primeira e da última regras.

◇

d. EXPR

```
E  -->  ( E )  |  E + E  |  E - E  |  E * E  |  E / E  |  N
N  -->  N D  |  D
D  -->  0  |  1  |  ...  |  9
```

*Solução alternativa:*

```
E  -->  ( E )  |  E + T  |  E - T  |  T
T  -->  T * F  |  T / F  |  F
F  -->  ( E )  |  N
N  -->  N D  |  D
D  -->  0  |  1  |  ...  |  9
```

◇

e. REG

Nesse exemplo nós vamos ver que o conjunto de todas as expressões regulares pode ser gerado por uma gramática. Isso é fácil de ver raciocinando recursivamente, pois

- se $\alpha$ e $\beta$ são expressões regulares, então $\alpha\beta$ também é uma expressão regular, o que corresponde à regra

  $S \rightarrow SS$

- se $\alpha$ e $\beta$ são expressões regulares, então $(\alpha \cup \beta)$ também é uma expressão regular, o que corresponde à regra

  $S \rightarrow (S \cup S)$

- se $\alpha$ é uma expressão regular, então $(\alpha)^*$ também é uma expressão regular, o que corresponde à regra

  $S \rightarrow (S)^*$

- os símbolos `a`, `b` e a palavra vazia `e` são expressões regulares triviais, o que corresponde à regra também é uma expressão regular, o que corresponde à regra

  $S \rightarrow$ `a` | `b` | `e`

◇

f. LOOP

( . . . )

◇

g. FRAGC

```
<compound stmt> --> { <stmt list> }
```

```
<stmt list> --> <stmt> <stmt list> | epsilon
<stmt> --> <compound stmt>
<stmt> --> if ( <expr> ) <stmt>
<stmt> --> if ( <expr> ) <stmt> else <stmt>
<stmt> --> while ( <expr> ) <stmt>
<stmt> --> do <stmt> while ( <expr> ) ;
<stmt> --> for ( <stmt> <expr> ; <expr> ) <stmt>
<stmt> --> case <expr> : <stmt>
<stmt> --> switch ( <expr> ) <stmt>
<stmt> --> break ; | continue ;
<stmt> --> return <expr> ; | goto <id> ;
```

◇