

# Fundamentos da Teoria da Computação

Professor: Newton José Vieira

Monitor: Reinaldo Fortes

2014-01

DCC/ICE/UFMG

## Trabalho Prático 02 (TP02) - Algoritmo CYK

### - O trabalho é Individual.

- O padrão de entrada e saída deve ser respeitado **exatamente** como determinado no enunciado.
  - Deve ser usada uma das linguagens: C, C++ ou Java.
  - A entrega do código fonte e executável deverá ser feita através do Moodle até o dia **08/05/2014 até 23:55**.
- Bom trabalho!**

## 1 Descrição

Seja  $G = (V, \Sigma, R, S)$  uma gramática livre do contexto (GLC) em que  $V$  é o conjunto de variáveis,  $\Sigma$  o alfabeto,  $R$  o conjunto de regras e  $S$  o símbolo de partida.

O objetivo deste trabalho prático é a implementação do algoritmo CYK, de ordem polinomial, capaz de reconhecer palavras geradas por uma GLC na forma normal de Chomsky (FNC). O nome desse algoritmo, que data de 1965, tem as iniciais de seus criadores *J. Cocke, D. H. Younger e T. Kasami*.

O algoritmo CYK, a partir de uma GLC  $G$  na FNC e uma palavra  $w$ , realiza uma análise ascendente. Ele inicia com a palavra  $w$ , que se deseja avaliar, e a cada passo tenta deduzir qual regra da gramática leva à geração da palavra no passo seguinte, obtendo, ao final, uma matriz triangular ( $M$ ). Caso o símbolo de partida da gramática esteja na célula no topo da matriz, a palavra pertence à linguagem gerada pela gramática, caso contrário, ela não pertence.

O algoritmo possui variadas aplicações práticas, tais como em *análise sintática*, em *bioinformática*, na verificação de *alinhamento de sequências*, em *linguística*, na verificação de estruturas de sentenças e palavras em *linguagem natural*, entre outras.

O pseudo-código do algoritmo, que pode ser usado como base para implementação, é apresentado no Algoritmo 1. Contudo, este trabalho também compreende um pequeno esforço da parte do aluno em pesquisar a respeito do método em outras referências ou na Web. Algumas referências sugeridas:

- Hopcroft, J.E., Motwani, R., Ullman, J.D. *Introduction to Automata Theory, Languages and Computation*, 2nd ed., Addison-Wesley, 2001 (Seção 7.4.4, páginas 298 - 302);
- Menezes, P.B. *Linguagens Formais e Autômatos*, 2a ed., Sagra Luzzatto, 2000 (Seção 3.9.2, páginas 122 - 124);
- Wikipedia: [http://en.wikipedia.org/wiki/CYK\\_algorithm](http://en.wikipedia.org/wiki/CYK_algorithm).

### 1.1 Ideia principal

Em termos informais, este algoritmo considera todas as subpalavras de  $w$  e define  $T[i, j, k]$  para ser *verdadeiro* se a subpalavra a partir de  $i$  de comprimento  $j$  pode ser gerada a partir da variável  $V_k$ . Uma vez consideradas subpalavras de comprimento 1, ele passa para subpalavras de comprimento 2, e assim por diante. Para subpalavras de comprimento maior que 1, ele considera todas as possíveis partições da subpalavra em duas partes, e verifica se há alguma produção  $A \rightarrow BC \in R$  tal que  $B$  corresponde à primeira parte e  $C$  corresponde à segunda parte. Se assim for, ele registra  $A$  como combinando toda a subpalavra. Quando esse processo for concluído,  $w$  é gerada pela gramática se o símbolo de partida foi registrado para a palavra inteira.

A matriz booleana  $T$  é uma representação da matriz triangular  $M$ , que é preenchida de baixo para cima com símbolos variáveis. Como exemplo, seja a gramática definida por:  $G = \{\{S, A\}, \{a, b\}, R, S\}$ , com  $R = \{S \rightarrow AA, S \rightarrow AS, S \rightarrow b, A \rightarrow AS, A \rightarrow SA, A \rightarrow a\}$ . Deve-se verificar se  $G$  gera a palavra *abaab*.

A Figura 1(A) apresenta o conteúdo da matriz após o primeiro passo do algoritmo. Ao término do primeiro passo, a linha inferior da matriz é preenchida com todas as variáveis que geram cada símbolo da palavra. Na Figura 1, a palavra a ser verificada é apresentada logo abaixo da matriz  $M$ .

---

**Algoritmo 1:** Algoritmo CYK.

---

```

input : GLC  $G$ , palavra  $w[1\dots n]$ 
output: SIM, se  $w \in L(G)$  ou NAO, se  $w \notin L(G)$ 

1 Crie uma matriz booleana  $T[n, n, |V|]$  com todas as células em Falso;
2 for  $i = 1$  TO  $n$  do
3   for each  $V_j \rightarrow a_i \in R$  do
4     |  $T[i, 1, j] \leftarrow Verdadeiro$ ;
5   end
6 end

7 for  $i = 2$  TO  $n$  do
8   for  $j = 1$  TO  $n-i+1$  do
9     for  $k = 1$  TO  $i-1$  do
10    for each  $V_A \rightarrow V_B V_C \in R$  do
11      | if  $T[j, k, B] \wedge T[j+k, i-k, C]$  then
12        | |  $T[j, i, A] \leftarrow Verdadeiro$ ;
13      end
14    end
15  end
16 end

17 end

18 for each  $x \in S$  do
19   if  $T[1, n, x]$  then
20     | Retorne SIM;
21   end
22 end

23 Retorne NAO;

```

---

Nos passos seguintes, para preencher as linhas superiores, uma variável  $A$  será colocada na matriz  $M$  se existem duas outras variáveis  $B$  e  $C$  em que:

- $A \rightarrow BC$  é uma regra;
- $B$  está à esquerda de  $C$  e ambos estão “por baixo” e à direita de  $A$ .

As Figuras 1(B) e 1(C), apresentam o segundo e o quinto (e último) passos do algoritmo, respectivamente. Podemos concluir que  $G$  gera a palavra *abaab*, já que o símbolo de partida  $S$  encontra-se no topo da matriz  $M$ .

(A)	(B)	(C)

Figura 1: Matriz  $M$  construída a partir do primeiro, segundo e quinto (e último) passos do algoritmo CYK.

## 2 Representação

Para simplificar um pouco o formato dos dados de entrada consideramos que: o conjunto de terminais consiste das letras minúsculas do nosso alfabeto, ou seja,  $\Sigma = \{a, b, \dots, z\}$ ; o conjunto de variáveis consiste das letras maiúsculas do nosso alfabeto, ou seja,  $V = \{A, B, \dots, Z\}$ ; e o símbolo  $S$  é o símbolo de partida da gramática.

## 2.1 Entrada

Dadas as convenções anteriores, a entrada do programa será constituída de:

- Uma linha contendo a palavra de entrada. Ela deve ter no máximo 50 símbolos.
- Uma linha contendo um inteiro positivo,  $r$ , que representa o número de regras da gramática.
- $r$  linhas contendo as regras da gramática. Cada regra deve ter o formato  $X \rightarrow a_1 a_2 \dots a_n$ , em que  $X \in V$  e  $a_i \in (V \cup \Sigma)$ , com cada  $a_i$  separado de  $a_{i+1}$  por um único espaço.

## 2.2 Saída

O programa deve retornar apenas o resultado da aplicação do algoritmo dizendo:

- SIM, se a gramática gera a palavra.
- NAO, se a gramática não gera a palavra.

## 2.3 Exemplos de Entrada e Saída

Entrada	Saída
abaab 6 $S \rightarrow A A$ $S \rightarrow A S$ $S \rightarrow b$ $A \rightarrow A S$ $A \rightarrow S A$ $A \rightarrow a$	SIM

Entrada	Saída
abbabba 7 $S \rightarrow S F$ $S \rightarrow a$ $A \rightarrow C C$ $A \rightarrow S S$ $A \rightarrow C S$ $C \rightarrow b$ $F \rightarrow A S$	SIM

Entrada	Saída
aaabbabaaaabba 8 $S \rightarrow S F$ $S \rightarrow a$ $A \rightarrow C G$ $A \rightarrow S S$ $A \rightarrow C S$ $C \rightarrow b$ $F \rightarrow A S$ $G \rightarrow C A$	NAO