Teoria da Computação

Modelos de Computação para a Engenharia e Ciência da Computação

Flávio Luis de Mello

Mello, Flávio Luis de
Teoria da Computação: Modelos de Computação para a Engenharia e Ciência da Computação / Flávio Luis de Mello. – Rio de Janeiro, RJ: Ed. do Autor, 2024.
Impresso,PDF

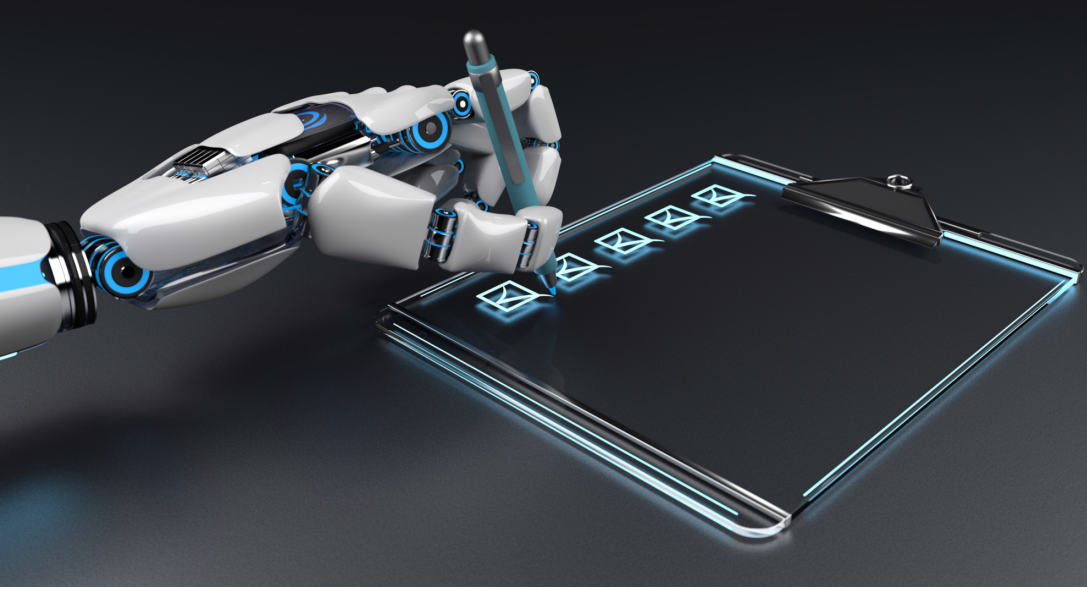
ISBN 978-65-01-05427-8

1. Algoritmos. 2. Recursão Primitiva. 3. Sistemas Formais. 4. Gramáticas. 5.Autômatos. 6. Maquina de Turing. I. Título.

CDD-004.07

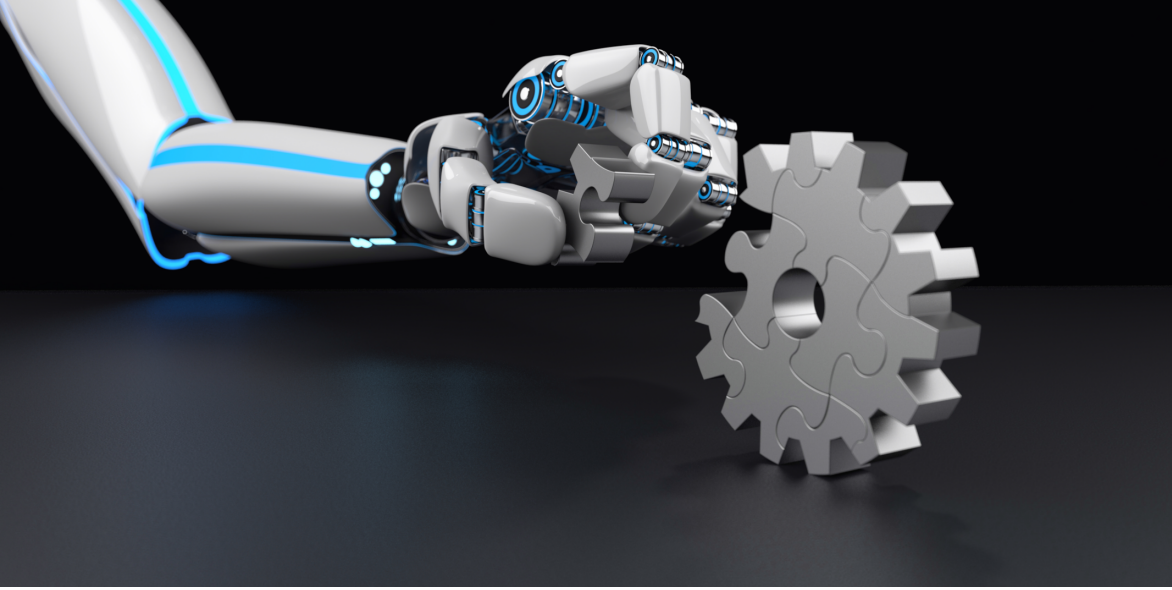
Primeira Edição, Junho 2024

Ilustrações: Depositphotos (standard license OD-3533016) IDs: 273814828, 230191678, 273811870, 230191828, 449211536, 273810600, 312108250, 449211536, 273809496, 710494544, 273809284, 640277714, 407529128, 616916078, 118993206, 273812456, 255433646, 209926176, 273809582, 255430884

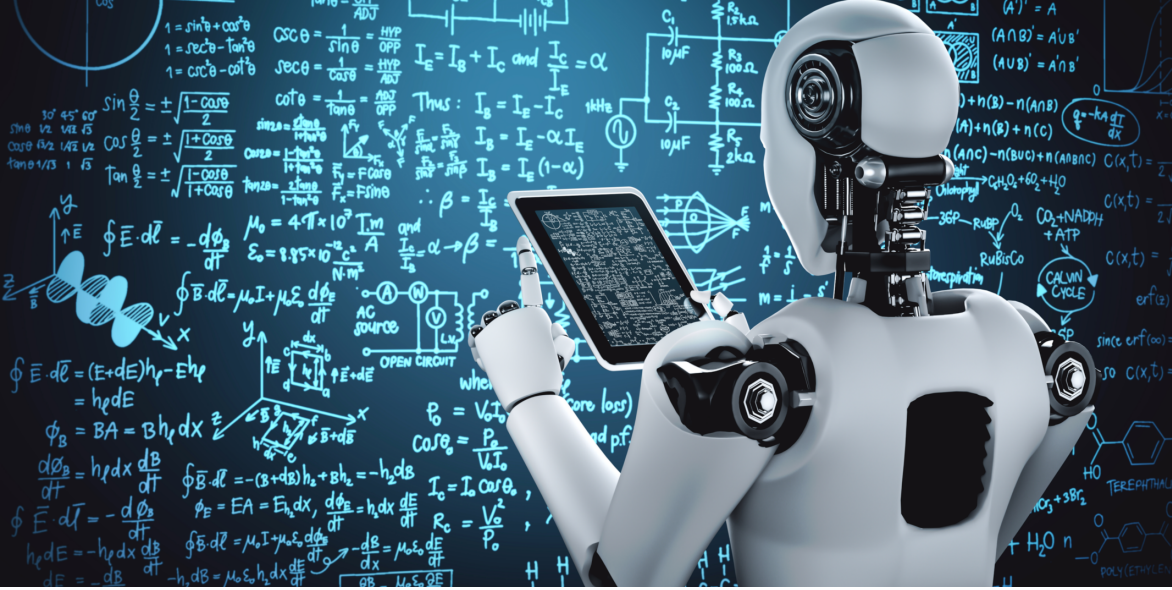


Sumário

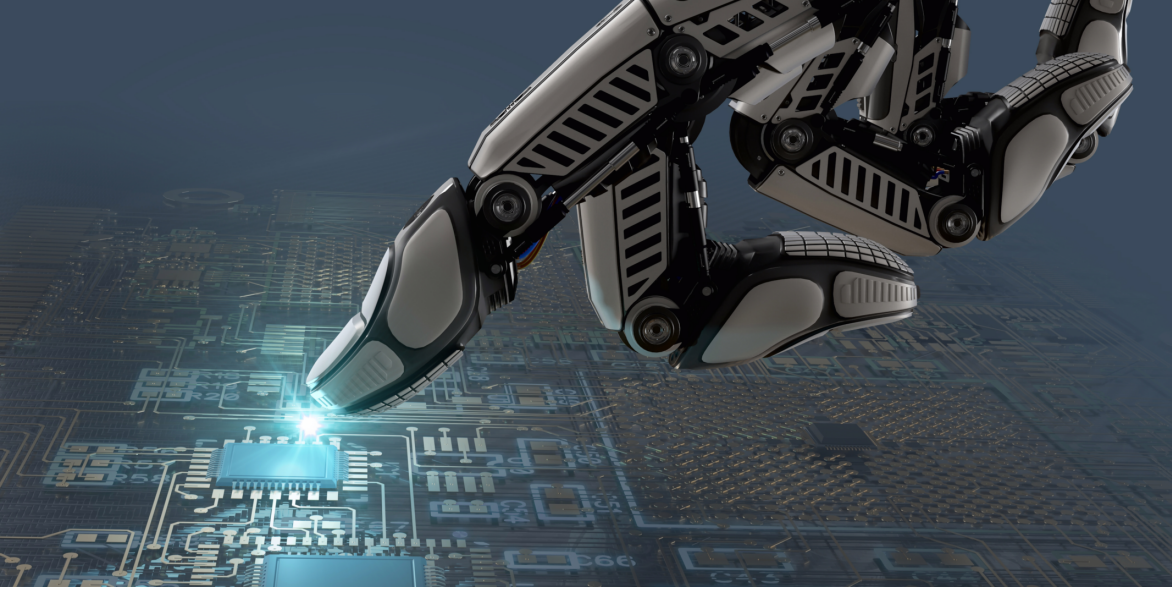
Lista de Figuras



Lista de Tabelas



Lista de Algoritmos



9. Sistemas Formais

A Informática é a ciência do artificial por excelência e, mais ainda, é a ciência do formal: não existe na história da humanidade nenhum profissional que tenha se comportado de modo mais formal. Os computadores seguem fielmente regras, e não admitem exceções; é preciso especificar, codificar, digitar, depurar e, mesmo assim, o programa pode não funcionar simplesmente porque foi trocado um '0' (zero) por 'O' , ou 'a' por 'A'. Depois de semanas buscando erros lógicos, alguém de fora, por trás dos ombros do programador diz: **"Claro que não funciona: você usou '1' e não 'l'."**, e então descobre-se que o engano era de natureza apenas formal. O grande objetivo dos sistemas amigáveis é a eliminação dos erros formais.

Explicar ou definir formal é uma tarefa difícil e, se for buscado o seu significado em um dicionário, encontra-se que:

(0) Formal - do latim formale - relativo a forma.

(1) Forma - do latim forma - Configuração exterior dos corpos; disposição das partes de um corpo, aparência; feitio de um objeto; modelo; norma.

(2) Forma - do latim forma - molde dentro do qual ou sobre o qual se forma qualquer coisa que toma o feitio desse molde.

Depois desta consulta, descobre-se que "formal" é definido através de "forma" e o problema é saber qual a escolha a ser feita: se (1) ou (2), notando-se que há diferença fonética, ainda que não gráfica entre

elas: em (1) a letra "o" é pronunciada aberta e em (2) fechada.

Então pode-se usar (1) e (2) para melhorar o entendimento de formal. Pois o próprio conceito definido em (1) deve se encaixar com a "forma" (com "o" fechado) de um grupo social.

A matemática é considerada como a mais formal das ciências sendo a linguagem formal utilizada pelas outras, pois todos os resultados são baseados em regras e apresentados por fórmulas. No entanto os formalistas são apenas um grupo dentre os matemáticos, tendo existido sérias controvérsias com respeito à validade do enfoque formal. Na realidade, a maioria dos matemáticos desenvolve seus resultados dentro de um espírito informal e intuitivo, mais geométrico que algébrico; e quando algébrico, se analisado de forma mais rigorosa, pouco formal. De qualquer modo, a mais formal das correntes em matemática tem feito e faz concessões ao informal. No que se segue será apresentado que o conceito de formal a ser adotado é extremamente exigente; pode-se dizer que o formal de que se necessita é também mecânico, completamente dissociado de qualquer intuição ou fatores de natureza cognitiva.

9.1 Sistemas Formais

Nesta seção será definido o que se entende por um sistema formal. Para isto é preciso esclarecer o que significa alfabeto e palavra, pois o conceito de formal - e, particularmente sistemas formais - depende da definição e conhecimentos básicos sobre representação gráfica de símbolos e a fixação de critérios particulares de sua aceitação ou definição (recomenda-se ao leitor uma releitura da Seção 2.7). Dá-se o nome de letra a todo sinal gráfico satisfazendo os seguintes critérios:

1. As letras devem possuir uma estrutura espacial que facilite sua reprodução e reconhecimento. Por exemplo: $\square, \triangle, |, *$. Como contra-exemplo o leitor pode imaginar figuras complicadas como rubricas pessoais, tais como $\flat\natural$.
2. As letras devem possuir uma estrutura que impossibilite decomposições horizontais. Assim, " ||" não seria uma escolha apropriada, pois é composta horizontalmente de dois sinais iguais ("|" e "|"); no entanto, "–" e "==" seriam escolhas adequadas - apesar de poderem ser decompostas verticalmente.
3. Para a construção de alguns sistemas formais, existe a necessidade de um suprimento infinito de letras, assim deve-se exigir

que elas possam ser produzidas de modo uniforme.
Os elemento de $\Sigma_1 = \{*, |\}$ e $\Sigma_2 = \{\square, \triangle, \odot\}$ satisfazem os critérios (a), (b) e (c) recém descritos. *Alfabetos* são conjuntos recursivos de letras.

Expressões, *palavras*[1] ou *cadeias* são seqüências de letras justapostas horizontalmente, tendo seus limites claramente identificados por interespaço separador com mesma função que o espaço em branco na escrita convencional. Assim, $\square\triangle$ e $\square \odot \square\triangle$ são expressões no alfabeto Σ_2.

Alguns autores definem uma cadeia u em um alfabeto Σ como uma função $u : \{1 \cdots n\} \to \Sigma$, onde n é o comprimento de u. Assim, uma cadeia u pode ser escrita como $a_1 a_2 \cdots a_n$, onde a_i é a i-ésima letra. A cadeia de comprimento 0 é chamada **cadeia nula** e denotada por Λ.

A justaposição de duas cadeias é chamada concatenação. Sejam $u : \{1 \cdots m\} \to \Sigma_1$ e $v : \{1 \cdots n\} \to \Sigma_2$, então $u \frown v$ (ou simplesmente uv) é definida como:

$$u \frown v : \{1 \cdots m+n\} \to \Sigma_1 \cup \Sigma_2$$
$$u \frown v = \begin{cases} u(i) & se\ 1 \leq i \leq m \\ v(i-m) & se\ m+1 \leq i \leq m+n \end{cases}$$

O produto de dois alfabetos é dado pela combinação de pares feita entre seus elementos. Por exemplo, se $\Sigma_1 = \{*, |\}$ e $\Sigma_2 = \{\square, \triangle, \odot\}$, tem-se que $\Sigma_1 \times \Sigma_2 = \{*\square, *\triangle, *\odot\ |\square, |\triangle, |\odot\}$, e ainda que $\Sigma_1 \times \Sigma_2 \neq \Sigma_2 \times \Sigma_1$.

A exponenciação de um alfabeto é dada por:

$$\Sigma^0 = \{\Lambda\}$$
$$\Sigma^1 = \Sigma$$
$$\Sigma^n = \Sigma^{n-1} \times \Sigma$$

observe que Σ^n é o conjunto de todas as cadeias de símbolos de Σ com comprimento n.

[1] Alguns autores mais rigorosos consideram que palavras são expressões que respeitam determinados critérios explícitos para sua formação. Desta forma pode-se explicitar que o critério de formação para as palavras em Σ_1 seja: *as únicas palavras são as expressões onde não apareçam mais que duas ocorrências sucessivas de '*' e não menos que duas de "|" em seqüência.* Utilizando-se este critério, conclui-se (informalmente) que, $**||*$ é uma palavra, e que $**|*$ não é, pois possui menos que duas ocorrências sucessivas de "|".

Por fim, se Σ for um alfabeto, denota-se por Σ^* o conjunto de todas as cadeias de Σ, incluindo a cadeia nula, isto é, $\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \cdots \cup \Sigma^n \cup \cdots$. Segue-se que uma linguagem em Σ é qualquer subconjunto de Σ^*. Desta forma, se Σ é um alfabeto, então qualquer conjunto de expressões em Σ é uma *linguagem* em Σ. Assim, por exemplo, $\mathcal{L} = \{\Box, \Box\triangle, \Box\odot, \triangle\}$ é uma linguagem em Σ.

Definição 9.1 — Sistema Formal. Um sistema formal $\mathcal{F}$ é uma quádrupla $< \Sigma, \mathcal{L}, \mathcal{A}, \mathcal{R} >$, onde:
Σ – é um alfabeto.
$\mathcal{L}$ – é um conjunto recursivo em Σ, chamado de linguagem do sistema formal.
$\mathcal{A}$ – é um subconjunto recursivo de $\mathcal{L}$, chamado de *axiomas*
$\mathcal{R}$ – é um conjunto recursivo de relações em $\mathcal{L}$

■ **Exemplo 9.1** Seja um sistema formal, onde o alfabeto, as palavras, os axiomas e as relações estejam definidas a seguir:

$$\Sigma = \{|, *\},\ \mathcal{L} = \{\Sigma^*\},\ \mathcal{A} = \{|, *\},\ \mathcal{R} = \{r_1, r_2\},\ \textit{onde}:$$
$$r_1 = \{< x|, x* > | x \in \Sigma^*\}$$
$$r_2 = \{\{< x|*, x* | > | x \in \Sigma^*\} \cup \{< x| **, x* || > | x \in \Sigma^*\} \cup \{< x*, x|| > | x \in \Sigma^*\}\}$$

■

Definição 9.2 — Dedução. Seja $\mathcal{F} =< \Sigma, \mathcal{L}, \mathcal{A}, \mathcal{R} >$ um sistema formal, $\Gamma \subseteq \mathcal{L}$. Uma dedução de α a partir de Γ em $\mathcal{F}$ é uma seqüência $\alpha_1, \alpha_2, \cdots, \alpha_n$ de palavras de $\mathcal{L}$, tal que:

1. α_n é α; e
2. Para todo $j, 1 \leq j < n, \alpha_j \in \Gamma \cup \mathcal{A}$, ou existem $\alpha_{j_1}, \cdots, \alpha_{j_k}$, $j_i \in \{1, \cdots, j-1\}, 1 \leq i \leq k$ tais que $< \alpha_{j_1}, \cdots, \alpha_{j_k}, \alpha_j > \in r$ com $r \in \mathcal{R}$.

Se existir uma dedução de α a partir de Γ diz-se que α é dedutível a partir de Γ em $\mathcal{F}$. Isto é denotado por $\Gamma \vdash_{\mathcal{F}} \alpha$.

■ **Exemplo 9.2** No sistema formal do exemplo 9.1 uma dedução de $*|$

é:

$$\begin{array}{ll} | & (\in \mathcal{A}) \\ * & (<|,*>\in r_1) \\ || & (<*,||>\in r_2) \\ |* & (<||,|*>\in r_1) \\ *| & (<|*,*|>\in r_2) \end{array}$$

portanto a seqüência $*, ||, |*, *|$ é uma dedução de $*|$ onde $\Gamma = \emptyset$, assim $\emptyset \vdash *|$. ∎

Existem várias considerações que devem ser feitas com respeito às componentes de um sistema formal.

Σ – Os alfabetos dos sistemas formais podem conter vários tipos de letras, assim em geral devem ser especificados tais tipos de modo não ambíguo. Os tipos de letras correspondem a modos de construir as palavras da linguagem $\mathcal{L}$. Assim na definição de Σ pode-se ter:

$$\Sigma = \Sigma_1 \cup \Sigma_2 \cup \cdots \Sigma_k$$

Por exemplo, nos dialetos formalizáveis oriundos de simplificações das linguagens naturais tem-se letras para representar as diversas categorias gramaticais. É frequente se incluir no alfabeto letras que são auxiliares (variáveis sintáticas) para representar subclasses da linguagem $\mathcal{L}$.

$\mathcal{L}$ – A definição da linguagem pode já necessitar de um outro sistema formal, ou notações informais para sua especificação detalhada. Note que a exigência de ser a linguagem um conjunto recursivo não é suficiente quando o sistema formal visa aplicações. As operações que são utilizadas para os critérios de construção dos componentes da linguagem devem ser de baixa complexidade.

$\mathcal{A}$ – Existem várias maneiras de se especificar os axiomas de um sistema formal. Se o conjunto for finito basta uma simples enumeração, no caso de um conjunto infinito pode-se utilizar um outro sistema formal para sua especificação. É frequente na literatura a utilização de *axiomas esquemas* que são bastante difíceis de serem efetivos nas aplicações computacionais.

$\mathcal{R}$ – A definição das regras de inferência é a parte mais complicada de um sistema formal. As regras de inferência devem ser relações recursivas definidas na linguagem $\mathcal{L}$ podendo, portanto, ser bastante complexas para definir e para serem efetivamente utilizadas. Nos exemplos que serão apresentados se discutirá a

complexidade de tais regras.

Estas observações devem ser levadas em conta quando se deseja ter um sistema formal que possa ser efetivamente utilizado. O desenvolvimento de critérios para a escolha de um sistema formal entre alternativas de formalização é uma necessidade pouco estudada. Deve-se acrescentar que os sistemas formais foram inventados com o objetivo de se *mecanizar* a solução de problemas, principalmente na matemática tradicional, e que o *agente* executante das deduções era inicialmente imaginado como um ser humano treinado como matemático. Hoje deve-se dirigir os esforços na construção de sistemas formais cujos *agentes* são de outra natureza, muitas vezes uma máquina ou sistema de programação.

Na especificação de regras de inferência escreve-se:

$$\frac{\alpha_1, \alpha_2, \cdots, \alpha_l}{\beta},\ ou\ \alpha_1, \alpha_2, \cdots, \alpha_l \rightarrow \beta$$

significando que $< \alpha_1, \alpha_2, \cdots, \alpha_l, \beta > \in \mathcal{R}$.

No caso em que $\Gamma \subseteq \mathcal{L} = \emptyset$, diz-se que α é um teorema em $\mathcal{F}$. O conjunto de teoremas de um sistema formal $\mathcal{F}$ é denotado por $\mathcal{T}_{\mathcal{F}}$, ou simplesmente por $\mathcal{T}$, quando é claro a que sistema formal pertencem os teoremas. O conjunto de teoremas de um sistema formal inclui os axiomas, ou seja, todo axioma é um teorema. Em geral, o conjunto de teoremas é maior que o conjunto de axiomas, mas é claro que as regras de um sistema formal podem não ser aplicáveis, e neste caso o conjunto de teoremas é composto apenas dos axiomas. O conjunto de teoremas pode ser igual à linguagem, e neste caso dizemos que o sistema formal é inconsistente[2]. A figura 9.1 mostra a relação entre axiomas, teoremas e linguagem.

Existem três tipos básicos de sistemas formais: geradores, reconhecedores e transdutores. Os geradores são sistemas formais cujo objetivo é produzir cadeias, somente a partir dos axiomas. Os reconhecedores admitem outras cadeias como entradas e verificam se tais cadeias são adequadas, sem produzir cadeias de saída. Por fim, os transdutores transformam cadeias de entrada em cadeias de saída.

A seguir serão apresentadas algumas variações dos tipos básicos, que serão importantes para o entendimento de questões a serem discutidas nos próximos capítulos.

[2]Existem várias concepções da noção de inconsistência. Esta, em particular, é chamada de inconsistência absoluta [44]

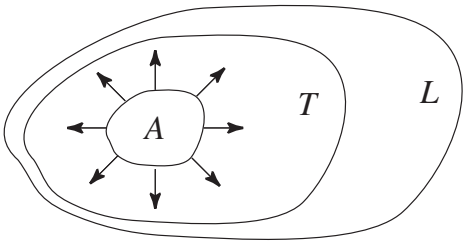


Figura 9.1: Derivação de Teoremas.

9.2 Sistemas de Produções de Post

Definição 9.3 — Sistema de Produção de Post. Um sistema de produção de Post (SPP) $\mathcal{S}$ é um sistema formal $< \Sigma, \mathcal{L}, \mathcal{A}, \mathcal{R} >$, onde:

Σ É um alfabeto consistindo de dois subconjuntos disjuntos N e T, chamados de alfabeto não terminal e terminal, respectivamente, onde $N = \{\boxed{\mathrm{i}} | i > 0\}$

$\mathcal{L}$ É um subconjunto Σ^*

$\mathcal{A}$ É um sub-conjunto de Σ^*, e é dito ser o conjunto de palavras de partida.

$\mathcal{R}$ É um conjunto de relações binárias em $\mathcal{L}$, que são chamadas de regras de produção. Cada regra é da forma:

$$x_0\boxed{i_1}x_1\boxed{i_2}\cdots x_{n-1}\boxed{i_n}x_n \rightarrow y_0\boxed{j_1}y_1\boxed{j_2}\cdots y_{k-1}\boxed{j_k}y_k$$

,onde $i_1, i_2, \cdots, i_n \in \mathbb{N}$, *e* $j_1, j_2, \cdots, j_k \in \{i_1, i_2, \cdots, i_n\}$ e $x_0, x_1, \cdots, x_n, y_0, \cdots, y_k \in (\Sigma - N)^*$

Definição 9.4 Seja $\mathcal{S} =< \Sigma, \mathcal{L}, \mathcal{A}, \mathcal{R} >$ um SPP e $\alpha \in \mathcal{L}$ uma palavra.

(a) Diz-se que uma regra:

$$x_0\boxed{i_1}x_1\boxed{i_2}\cdots x_{n-1}\boxed{i_n}x_n \rightarrow y_0\boxed{j_1}y_1\boxed{j_2}\cdots y_{k-1}\boxed{j_k}y_k$$

é aplicável a α se e somente se:

$$\alpha = x_0 z_{i_1} x_1 z_{i_2} \cdots x_{n-1} z_{i_n} x_n$$

,onde $z_{i_1}, z_{i_2}, \cdots, z_{i_n} \in \Sigma^*$ e se $i_t = i_s$ então $z_{i_t} = z_{i_s}$

(b) Se uma regra:

$$x_0 \boxed{i_1} x_1 \boxed{i_2} \cdots x_{n-1} \boxed{i_n} x_n \rightarrow y_0 \boxed{j_1} y_1 \boxed{j_2} \cdots y_{k-1} \boxed{j_k} y_k$$

é aplicável a uma palavra $\alpha = x_0 z_{i_1} x_1 z_{i_2} \cdots x_{n-1} z_{i_n} x_n$ então o resultado da aplicação é a palavra

$$y_0 z_{j_1} y_1 z_{j_2} y_2 \cdots y_{k-1} z_{j_k} y_k$$

(c) Diz-se $\alpha \Rightarrow_{\mathcal{S}} \beta$ se β foi obtida a partir da palavra α pela aplicação de uma regra $r \in \mathcal{R}$.

(d) Diz-se que β é derivável em $\mathcal{S}$ a partir de α, o que denota-se por $\alpha \Rightarrow^*_{\mathcal{S}} \beta$, se e somente se $\beta = \alpha$ ou existem $\beta_1, \beta_2, \cdots, \beta_l$ tais que $\beta_l = \beta$ e para todo $i,\ 1 \leq i < l\ \beta_i \Rightarrow_{\mathcal{S}} \beta_{i+1}$

(e) A linguagem gerada por $\mathcal{S}$ é definida por:

$$\mathcal{L}(\mathcal{S}) = \{x \in (\Sigma - N)^* | \alpha \Rightarrow^*_{\mathcal{S}} x, \alpha \in \mathcal{A}\}$$

Intuitivamente as caixas $\boxed{i_m}$ capturam as palavras z_{i_m} quando a regra é aplicada e reproduzem as palavras capturadas por estas caixas na cadeia do lado direito da regra.

■ **Exemplo 9.3** A regra $aa\boxed{1}b\boxed{2}a \rightarrow a\boxed{2}ba\boxed{1}$ aplicada à palavra $aaabbaa$ coloca ab na caixa numerada com 1 e a na caixa numerada por 2 e gera a palavra $aabaab$. Note que outra solução seria colocar a na caixa 1 e ba na caixa 2 e o resultado neste caso seria $ababaa$. ■

■ **Exemplo 9.4** Seja S o SPP no qual

$$\begin{array}{rcl} \Sigma & = & \{\square, \diamond, \underline{suc}, S, , \} \\ N & = & \{S, \underline{suc}, , \} \\ \mathcal{A} & = & \{\underline{suc}x | x \in (\Sigma - N)^*\} \\ \mathcal{R} & = & \left\{ \begin{array}{l} \underline{suc}\boxed{1} \rightarrow S\boxed{1}, \\ S, \boxed{1} \rightarrow \square\boxed{1} \\ S\boxed{1}\square, \boxed{2} \rightarrow \boxed{1}\diamond\boxed{2} \\ S\boxed{1}\diamond, \boxed{2} \rightarrow S\boxed{1}, \square\boxed{2} \end{array} \right. \end{array}$$

A seguir serão exibidas algumas derivações em S:

Axioma	Geração	Axioma	Geração
$\underline{suc}$	$\underline{suc} \Rightarrow S$, $S, \Rightarrow \square$	$\underline{suc}\diamond$	$\underline{suc}\diamond \Rightarrow S\diamond$, $S\diamond, \Rightarrow S, \square$ $S, \square \Rightarrow \square\square$
$\underline{suc}\square$	$\underline{suc}\square \Rightarrow S\square$, $S\square, \Rightarrow \diamond$	$\underline{suc}\square\square$	$\underline{suc}\square\square \Rightarrow S\square\square$, $S\square\square, \Rightarrow \square\diamond$

Pode-se interpretar este SPP como gerando a partir de um axioma $\underline{suc}$, onde n é a representação diádica de um natural no alfabeto $\{\square, \diamond\}$, a representação $n+1$. ∎

As regras de inferência de um SPP podem ser bastante complexas. As operações básicas são de concatenação, casamento de padrões e substituição. Tais operações não fazem parte do sistema formal e são de difícil execução, mesmo quando o agente é um ser humano bem treinado. O leitor familiarizado com os processos de análise sintática para a construção de compiladores pode apreciar tais dificuldades. A complexidade da linguagem $\mathcal{L}$ é um fator preponderante tanto para o uso do casamento de padrões como para a construção das regras de inferência. O leitor interessado pode consultar Book e Otto [8] que trata dos chamados sistemas de reescrita.

9.3 Linguagens Formais e Gramáticas

O aprendizado de uma língua qualquer envolve o estudo da estrutura das sentenças. A grosso modo esta estrutura é chamada de gramática para a língua, e sua apresentação usual é como um conjunto de critérios chamados de regras gramaticais que definem a corretude das sentenças. A rigor gramáticas são sistemas formais geradores de linguagens, portanto deve-se esperar que uma gramática construa a língua.

Em geral, uma linguagem natural tende a ser muito extensa e complexa. Duas das atividades dos linguistas são definir com precisão as sentenças válidas de linguagens e encontrar formas estruturadas de representá-las. Entretanto, há uma dificuldade para definir completamente estas linguagens. Considere a seguinte sentença do português: *O menino louco pintava o lindo quadro.*

Pode-se dizer que a sentença é constituída de sujeito “O menino louco” - e predicado - “pintava o lindo quadro”. Estes elementos podem ser mais analisados. O predicado é constituído de verbo “pin-

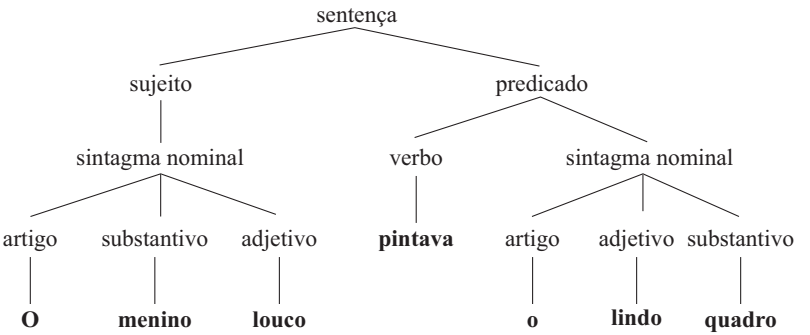


Figura 9.2: Árvore de Derivação Sintática de uma sentença em Português.

tava" e objeto "lindo quadro". A análise desta sentença é apresentada na Figura 9.2 e descreve uma representação conhecida como árvore de derivação sintática. Quando uma sentença produz uma árvore de derivação válida, diz-se que esta sentença é válida. Contudo, para uma árvore de derivação ser válida é preciso que ela seja especificada, uma tarefa pouco trivial quando se trata de uma linguagem natural. Uma tentativa incompleta de formalização das regras do Português pode ser descrita como se segue:

$< sentenca > \rightarrow < sujeito >< predicado >$
$< sujeito > \rightarrow < sintagma\ nominal >$
$< predicado > \rightarrow < verbo >< sintagma\ nominal >$
$< sintagma\ nominal > \rightarrow < artigo >< substantivo >< adjetivo >$
$< sintagma\ nominal > \rightarrow < artigo >< adjetivo >< substantivo >$
$< verbo > \rightarrow pintava$
$< artigo > \rightarrow o$
$< substantivo > \rightarrow menino$
$< substantivo > \rightarrow quadro$
$< adjetivo > \rightarrow louco$
$< adjetivo > \rightarrow lindo$

Estas regras indicam que os elementos à esquerda da seta podem ser transformados nos elementos à direita da seta. Os símbolos $< s >$ indicam categorias gramaticais genéricas, chamados *símbolos não-terminais*, e os demais símbolos são palavras do Português, chamados *símbolos terminais*.

As regras podem estabelecer que a sentença dada é gramaticalmente correta, bastando constatar que esta é gerável a partir daquelas. Além disso, estas regras podem ser utilizadas para construir outras sentenças corretas do Português como "O menino lindo pintava o quadro louco". Além disso, também podem ser produzidas outras sentenças sem valor semântico como "O quadro louco pintava o lindo menino".

Até o presente momento, não existe um conjunto de regras gramaticais e de palavras que possibilitem a formalização de uma linguagem natural, dada a explosão combinatorial que palavras e regras de uma linguagem natural produz. Entretanto, o mesmo conceito funciona para linguagens de programação pois estas podem ser definidas por uma sintaxe rígida e uma semântica bem determinada, possibilitando o processamento computacional.

Para especificar a sintaxe de uma linguagem de programação de modo sistemático é preciso uma gramática que possibilite:

- *Geração*: um método sistemático de especificação para a construção de programas corretos;
- *Reconhecimento*: um método de análise de um programa a fim de verificar se ele está sintaticamente correto.

A definição seguinte torna precisa a noção de gramática como um sistema formal, isto é, uma Gramática de Chomsky[3].

Definição 9.5 — Gramática de Chomsky. Uma gramática $\mathcal{G}$ é um sistema formal $< \Sigma, \mathcal{L}, \mathcal{A}, \mathcal{R} >$, onde:

Σ É um alfabeto consistindo de dois subconjuntos disjuntos N e T, chamados de alfabeto não terminal e terminal, respectivamente.

$\mathcal{L}$ É o conjunto Σ^*

$\mathcal{A}$ É o conjunto unitário $\{S\}$, sendo $S \in N$. S é dito ser o símbolo de partida e possui o nome da categorial sintática principal.

$\mathcal{R}$ É um conjunto de relações binárias em $\mathcal{L}$, chamadas de regras de produção.

Por convenção, os símbolos não terminais são representados por

[3] Avram Noam Chomsky (1928-) é responsável pela gramática gerativa transformacional, abordagem que revolucionou os estudos no domínio da linguística teórica. É também o autor de trabalhos fundamentais sobre as propriedades matemáticas das linguagens formais, sendo o seu nome associado à chamada Hierarquia de Chomsky.

letras maiúsculas e os símbolos terminais, por letras minúsculas. Seja a gramática $G1$ com $N = \{S, A\}$, $T = \{a, b\}$, $\mathcal{A} = \{S\}$ e $\mathcal{R} = \{S \to aA, A \to b\}$. Neste caso, $S \to aA \to ab$ que é a única cadeia gerada por $G1$.

Definição 9.6 — Derivação. Seja $\mathcal{G} =< \Sigma, \mathcal{L}, \mathcal{A}, \mathcal{R} >$ uma gramática, $x, y \in \mathcal{L}$. Diz-se que:

(a) y é diretamente derivável em $\mathcal{G}$ a partir de x, o que é denotado por $x \Rightarrow_{\mathcal{G}} y$ se e somente se

$$x = x_1 \alpha x_2 \ e \ y = x_1 \beta x_2 \ e \ \alpha \to \beta \in \mathcal{R}$$

(b) y é derivável em $\mathcal{G}$ a partir de x, o que é denotado por $x \Rightarrow^*_{\mathcal{G}} y$ se e somente se $y = x$ ou existem $\beta_1, \beta_2, \cdots, \beta_l$ tais que $\beta_l = \beta$ e para todo i, $1 \leq i < l$ $\beta_i \Rightarrow_{\mathcal{G}} \beta_{i+1}$

Uma gramática pode ser considerada como um SPP, cujas regras de produção da forma $\alpha \to \beta$ são representadas como $\boxed{1}\alpha\boxed{2} \to \boxed{1}\beta\boxed{2}$. Seja $G2$ com $N = \{A, B\}$, $T = \{a, b, c\}$, $\mathcal{A} = \{A\}$ e $\mathcal{R} = \{A \to aB, B \to bB, B \to c\}$. A sentença $x = abbbc$ é uma produção de $G2$ porque ela é derivável da seguinte forma: $A \overset{1}{\Rightarrow} aB \overset{2}{\Rightarrow} abB \overset{2}{\Rightarrow} abbB \overset{2}{\Rightarrow} abbbB \overset{3}{\Rightarrow} abbbc$.

Definição 9.7 — Linguagem Formal. Seja $\mathcal{G} =< \Sigma, \mathcal{L}, \mathcal{A}, \mathcal{R} >$ uma gramática. A linguagem formal gerada por $\mathcal{G}$, denotada por $L(\mathcal{G})$ é o conjunto de teoremas de $\mathcal{G}$ contendo apenas símbolos terminais de $\mathcal{G}$. Assim $L(\mathcal{G}) = \{x \in T^* | S \Rightarrow^*_{\mathcal{G}} x\}$.

■ **Exemplo 9.5** Seja a gramática $G2$ definida anteriormente. Pode-se verificar que $L(G2) = \{x \in T^* | x = ab^n c, n \geq 0\}$. ■

■ **Exemplo 9.6** Seja a gramática $G3$ com $N = \{S\}$, $T = \{a, b, c\}$, $\mathcal{A} = \{S\}$ e $\mathcal{R} = \{S \to aSa, S \to bSb, S \to c\}$. Então é fácil verificar que:

$$L(\mathcal{G}) = \{c, aca, aacaa, bcb, bacab, bbcbb, \cdots\}$$

■

■ **Exemplo 9.7** Seja a gramática $G4$ com $N = \{E, T, F\}$, $T = \{+, *, (,), x\}$, $\mathcal{A} = \{E\}$ e $\mathcal{R} = \{E \to E + T, E \to T, T \to T * F, T \to F, F \to (E), F \to x\}$. Um exemplo de cadeia derivável da gramática é a expressão conforme descrito na derivação $E \overset{2}{\Rightarrow} T \overset{3}{\Rightarrow} T + F \overset{5}{\Rightarrow} T * (E) \overset{5}{\Rightarrow} F*(E) \overset{1}{\Rightarrow} F*(E+T) \overset{2}{\Rightarrow} F*(T+T) \overset{4}{\Rightarrow} F*(F+F) \overset{6}{\Rightarrow} x*(x+x)$.

■

Quando se trata de linguagem de programação, um impedimento importante é a impossibilidade de uma cadeia $w \in L(G)$ possuir mais de um entendimento. Assim, linguagens de programação não devem ser ambíguas, ou ao menos, devem permitir que eventuais ambiguidades possam ser facilmente evitadas. O mais notório caso de ambiguidade é conhecido como o *else pendente* (dangling else).

Seja um gramática $\mathcal{G}$ com o seguinte conjunto de regras:

$BLK \rightarrow$ if a then BLK else BLK

$BLK \rightarrow$ if a then BLK

$BLK \rightarrow$ b

A gramática $\mathcal{G}$ é ambígua uma vez que a cadeia "if a then if a then b else b" pode ser interpretada como "if a then (if a then b else b)" ou como "if a then (if a then b) else b". Alguns compiladores não aceitam este tipo de construção, e outros, quando aceitam, escolhem a interpretação que associa o *else* ao *if* mais próximo, isto é, "if a then (if a then b else b)".

Em alguns casos, é possível perceber esta ambiguidade como algo inerente à gramática e nestas situações, pode-se reescrever a mesma a fim de eliminar estar ambiguidades. Seja a gramática $G5$ com $N = \{E, T, F, A\}$, $T = \{+, *, 0, 1, 2, \ldots, 9\}$, $\mathcal{A} = \{E\}$ e $\mathcal{R}$ dado por:

$E \rightarrow E + T$

$E \rightarrow T$

$T \rightarrow T * F$

$T \rightarrow F$

$F \rightarrow 0 \ldots 9$

que é utilizada para gerar expressões aritméticas com $+$ e $*$ para números inteiros. Este tipo de construção é bastante empregada em linguagens de programação. Além disto, seja também a gramática $G6$ com $N = \{E, A\}$, $T = \{+, *, 0, 1, 2, \ldots, 9\}$, $\mathcal{A} = \{E\}$ e $\mathcal{R}$ dado por:

$E \rightarrow E + E$

$E \rightarrow E * E$

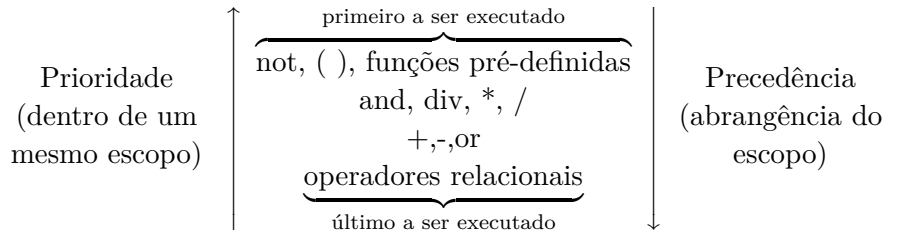
$E \rightarrow F$

$F \rightarrow 0 \ldots 9$

que não só é equivalente a $G5$, como também possui uma quantidade menor de produções e símbolos. Entretanto, $G6$ é ambígua enquanto que $G5$, não. Por exemplo, a cadeia $2+4*6$ possui dois entendimentos na gramática $G6$, "2+(4*6)" e "(2+4)*6". Já na gramática $G5$ só há

o entendimento "2+(4*6)".

Para dirimir a ambiguidade é preciso compreender as propriedades de precedência e de associatividade. A precedência permite decidir qual o correto entendimento das expressões, isto é, quanto maior a precedência de um operador, maior será o escopo contemplado por este mesmo operador. Já a associatividade permite decidir o correto entendimento de operações contendo operadores de mesma precedência. Em geral, a relação de prioridade e precedência entre operadores segue o esquema a seguir.



Por exemplo, na gramática $G5$ o operador $+$ tem maior precedência que $*$ porque está definido mais próximo das regras de produção que se iniciam por axiomas. Além disso, $*$ tem maior prioridade que $+$ pois é resolvido primeiro que $+$. O exemplo a seguir demostra a precedência de operadores da gramática $G5$.

■ **Exemplo 9.8** Precedência de operadores da gramática $G5$:

$$2 + 4 * 6 = +(2, *(4, 6))$$
$$2 * 4 + 6 * 8 = +(*(2, 4), *(6, 8))$$
$$2 + 4 + 6 = +(+(2, 4), 6)$$

■

Assim, o truque é aumentar a precedência dos operadores, colocando suas regras cada vez mais distantes do axioma $\mathcal{A}$. No exemplo a seguir, a primeira gramática é inicialmente ambígua. Na segunda gramática, a precedência é resolvida pela associatividade que é colocada em um nível mais baixo da árvore sintática. Contudo, este procedimento delega a quem monta a expressão a responsabilidade de determinar a precedência entre as operações utilizando a associatividade pois os operadores de $+$ e $*$ têm a mesma precedência. Por fim, a última gramática resolve este problema colocando o operador $*$ com maior precedência, isto é, colocando o operador $*$ e um nível mais baixo da árvore sintática que o operador $+$.

1^a Gramática	2^a Gramática	3^a Gramática
	Ambiguidade removida com associatividade,	
	+ e * tem a mesma	Precedência
Gramática ambígua	precedência	convencional
$E \to E + E$	$E \to E + T$	$E \to E + T$
$E \to E * E$	$E \to E * T$	$E \to T$
$E \to (E)$	$E \to T$	$T = T * F$
$E \to a$	$T \to (E)$	$T \to F$
	$T \to a$	$F \to (E)$
		$F \to a$

9.4 Hierarquia de Chomsky

Formas normais impõem restrições às formações de regras de uma gramática a fim de assegurar que determinadas propriedades sejam alcançadas. Existem alguns tipos de abordagens para essa construção das restrições, sendo duas de maior destaque: a forma normal de Chomsky e a de Greibach. Nesta seção será usada a abordagem de Chomsky. Sob esta ótica, uma gramática $\mathcal{G} =< \Sigma, \mathcal{L}, \mathcal{A}, \mathcal{R} >$ pode ser classificada pela forma de suas regras de produção em quatro tipos:

Tipo 0 Nenhuma restrição é imposta na forma das regras de produção. Estas gramáticas são chamadas de Turing reconhecíveis. A definição 9.5 refere-se a este tipo.

Tipo 1 Se $\alpha \to \beta \in \mathcal{R}$ tem-se que $|\alpha| \leq |\beta|$, onde $|\alpha|$ e $|\beta|$ representam os comprimentos de α e β, respectivamente, e excetuando a regra $\alpha \to \Lambda$. Gramáticas do tipo 1 são também chamadas *sensíveis ao contexto* pois há restrição de que as regras sejam da forma $\alpha_1 A \alpha_2 \to \alpha_1 \beta \alpha_2$ com α_1, α_2 e $\beta \in \Sigma^*$, $\beta \neq \Lambda$ e $A \in N$. Esta restrição é equivalente à anterior, e pode ser lida como "A pode ser substituído por β no contexto de α_1 e α_2". As gramáticas dos exemplos a seguir são do tipo 1.

■ **Exemplo 9.9** Seja a gramática $\mathcal{G}$ com $N = \{S\}$, $T = \{0, 1\}$ e $\mathcal{R} = \{S \to 0S1, S \to 01\}$. Então é fácil verificar que $L(\mathcal{G}) = \{0^n 1^n | n \geq 1\}$. ■

■ **Exemplo 9.10** Seja a gramática $\mathcal{G}$ com $N = \{S, B, C\}$ e $T =$

Teoria da Computação

Modelos de Computação para a Engenharia e Ciência da Computação

A maioria esmagadora das pessoas toma conhecimento da computação através de suas técnicas e tecnologias, raramente tem acesso à, ou mesmo compreendem a, ciência que está envolvida. Muito daquilo que se conhece hoje como computação moderna se baseia nas ideias, conceitos e modelos que aqui são apresentados.

Esta obra destina-se ao estudo de nível universitário, com parte dos capítulos voltados à graduação, e os demais, à pós-graduação. Primeiro estuda-se o conceito de computação e algoritmos. Em seguida são abordados os paradigmas relacionados com funções primitivas recursivas, linguagens minimais, maquinas de registros, algoritmos com rótulos, propriedades dos algoritmos, inferências por lógica simbólica, sistemas formais, gramáticas, autômatos e por fim, máquinas de Turing.

O livro trata com cuidado e critério rigoroso a apresentação de teoremas e lemas os quais, via de regra, estão acompanhados de suas respectivas demonstrações, didaticamente experimentadas em sala de aula. A técnica de ensino compreende uma apresentação formal de conceitos e modelos de computação, acompanhada de 193 exemplos que auxiliam o leitor a sedimentar o entendimento. O material também é ricamente complementado com 231 exercícios que auxiliam na revisão e amadurecimento do conhecimento discente.

Flávio Luis de Mello é professor Titular da Universidade Federal do Rio de Janeiro (UFRJ), chefe do Laboratório de Inteligência de Máquina e Modelos de Computação. Graduado em Computação, com mestrado e doutorado em Sistemas de Computação. Trabalha na área desde 1998, tendo atuado em projetos para as Forças Armadas, ABIN, setor elétrico, Petrobras, Intel, Nvidia, entre outros.

ISBN: 978-65-01-05427-8

9 786501 054278

COMPRE AQUI