Linguagens Formais e Autômatos

Aula 6: Linguagens Regulares

Prof. Dr. Rodrigo Xavier de Almeida Leão

Cientista de Dados e Big Data

Tipos de Gramáticas	Regras	Exemplos de Linguagens geradas por estas gramáticas
Regulares	$A \rightarrow b$, $A \rightarrow \epsilon$ ou $A \rightarrow aB$ $A, B \in V^*$, $a, b \in T^*$	$a^n b^m, n, m > 0$

Uma das aplicações mais básicas e importantes relacionadas a uma linguagem formal é a possibilidade de reconhecer-se mecânica ou automaticamente as palavras que pertencem a ela. Este é o papel do reconhecedor ou analisador sintático. Um reconhecedor para uma linguagem formal $L \subseteq \Sigma^*$ é um procedimento que ao ler

qualquer palavra $\omega \in \Sigma^*$ indica se $\omega \in L$ ou se $\omega \notin L$. Na realidade, o analisador sintático faz um pouco além disso, mas isto é assunto de outra unidade.

Uma das primeiras tarefas no reconhecimento de uma linguagem, seja ela natural ou não, trata da identificação das palavras ou unidades básicas que fazem parte dela. Já sabemos que uma palavra é uma cadeia de caracteres ou símbolos. A exigência básica é que estes sejam reconhecíveis. Isto é, para reconhecer a linguagem deve haver um mecanismo básico ou primitivo, capaz de ler cada símbolo individualmente e reconhecer se eles são diferentes de outros símbolos do mesmo alfabeto ou qualquer outro alfabeto que o inclua.

Por exemplo, no alfabeto binário $\{0,1\}$, este mecanismo básico deve ser capaz de distinguir o 0 do 1, além de distinguir 0 ou 1 de a, por exemplo, um símbolo que não pertence a $\{0,1\}$.

Todo alfabeto deve dispor deste mecanismo de identificação de seus elementos. Dado um símbolo s e um alfabeto Σ, saber se $s \in \Sigma$ e saber se $s \neq s_2 \in \Sigma$ deve ser um procedimento automático.

Como se faz para reconhecer uma palavra? A resposta imediata é verificando-a símbolo a símbolo. Por exemplo, sabemos que $12324A563$ não é um numeral decimal, dado que a palavra que o representa não é formada somente com dígitos decimais. Cada símbolo lido em $12324A563$ deve ser um dígito. Esta palavra é curta, e, portanto, não parece ser necessária nenhuma disciplina na sua leitura para encontrar o A que foge à regra de escrita de numerais decimais. No entanto, uma palavra com 10^100 caracteres justapostos

decimais. No entanto, uma palavra com 10^100 caracteres justapostos necessita de alguma disciplina de leitura. Como ter certeza que não há caracteres que não pertencem a $\{0,1,2,3,4,5,6,7,8,9\}$? Somente verificando sistematicamente. Se a linguagem fosse dos números pares até 10^{100} teríamos que verificar também se o último dígito está em $\{0,2,4,6,8\}$. Cada linguagem formal traz muitas possibilidades.

Para tornar o problema mais interessante ainda, lembramos que as linguagens mais usadas no dia a dia e em computação são infinitas. Portanto, um procedimento de verificação é essencial neste caso, dado que não há como especificar ou verificar pertinência que não seja via procedimento computacional.

Portanto, um procedimento de verificação é essencial neste caso, dado que não há como especificar ou verificar pertinência que não seja via procedimento computacional.

Voltando ao reconhecimento de uma linguagem formal, façamos um paralelo com a forma com que nós, seres humanos, processamos um texto em linguagem natural. Ao lermos um texto, uma tarefa que

nos é exigida imediatamente é a segmentação do texto em palavras. Somente depois de reconhecermos as palavras é que as frases ou as orações são processadas - em processamento de linguagem natural por meio de programas de computador ou aplicativos, essa também é a primeira tarefa. Em linguagens de programação esta etapa recebe

é a primeira tarefa. Em linguagens de programação esta etapa recebe o nome de análise léxica e é exatamente isso, percorre-se o texto agrupando os símbolos em agregados (*clusters*) que denominamos, na terminologia de compiladores, de *itens léxicos*. Em termos de especificação da linguagem esses são os símbolos do nosso alfabeto. Exemplos de itens léxicos na linguagem de programação C são os identificadores, nomes de variáveis, nomes de funções e palavras reservadas tais como *"main"*, *"function"*, *"while"* etc. (AHO et al., 2006)

Uma curiosidade com relação à formação dos itens léxicos é o fato da leitura/escrita de um texto se dar em diferentes direções nas linguagens naturais. Mandarim e japonês são lidos/escritos de cima para baixo e em seguida da esquerda para a direita. Hebraico e árabe são escritos e lidos da direita para a esquerda, enquanto a maioria das línguas ocidentais é escrita da esquerda para a direita. No entanto, alguns nomes próprios que podem ser escritos em todas estas linguagens são os mesmos nomes, estejam eles na vertical, escritos da direita para a esquerda ou escritos da esquerda para a direita. Por

alguns nomes próprios que podem ser escritos em todas estas linguagens são os mesmos nomes, estejam eles na vertical, escritos da direita para a esquerda ou escritos da esquerda para a direita. Por exemplo, Saul em hebraico é escrito לואש(lwu'ahS), lido da direita para esquerda é praticamente o que vemos em grego como $\Sigma\alpha\upsilon\lambda$, da esquerda para a direita. A direção da escrita/leitura não altera todas as características da palavra, até mesmo quando o alfabeto utilizado muda (NAVEH, 2012). Observe como os símbolos em Saul se relacionam, mesmo estando em alfabetos distintos. Veremos nesta seção que a direção de leitura/escrita em uma linguagem formal não a altera como conjunto de palavras.

O tipo de gramática mais simples é a gramática regular. Uma gramática regular possibilita uma análise bem simples da sua linguagem gerada. Melhor ainda, esta análise é feita de forma sistemática a partir da própria gramática. Lembramos que na Unidade 1 definimos uma gramática regular como uma gramática G = (V,T,P,S), que só possui regras da forma $A \to b$, $A \to \epsilon$ ou $A \to aB$, com $A, B \in V$ e $a, b \in T$. Observe que A e B aqui representam não terminais quaisquer de V, podendo ser iguais entre si ou diferentes. Da mesma forma a e b representam terminais quaisquer do conjunto T. (HOPCROFT; ULLMAN, 1969) (GARCIA, 2017).

Assimile

Uma gramática G = (V, T, P, S) é regular se, e somente se, só possui regras da forma $A \to b$, $A \to \epsilon$ ou $A \to aB$, com $A, B \in V$ e $a, b \in T$.

Observe em particular a linguagem $L = \{aab, bba\}$ sobre o alfabeto $\Sigma = \{a, b\}$. Esta linguagem pode ser gerada pela seguinte gramática regular:

$$S \to aA_1,$$
$$S \to bB_1,$$
$$A_1 \to aA_2,$$
$$A_2 \to b,$$
$$B_1 \to bB_2,$$
$$B_2 \to a$$

Observe que a primeira cadeia pode ser gerada pela derivação:

$$S \Rightarrow aA_1 \Rightarrow aaA_2 \Rightarrow aab$$

Enquanto que a segunda cadeia pode ser gerada pela derivação:

$$S \Rightarrow bB_1 \Rightarrow bbB_2 \Rightarrow bba$$

Lembramos que na Unidade 1 definimos que uma linguagem é regular se, e somente se, existir uma gramática regular que a gere. Assim, dada uma gramática G que não é regular, é possível que a linguagem $L(G)$ seja regular, bastando que, para isso, exista uma gramática regular G_2 tal que $L(G_2) = L(G)$.

Considere a gramática:

$$S \to A_1 b\,,$$

$$S \to B_1 a\,,$$

$$A_1 \to A_2 a\,,$$

$$A_2 \to a\,,$$

$$B_1 \to B_2 b\,,$$

$$B_2 \to b$$

Deve estar claro que esta gramática não é regular, entretanto, temos que $L(G) = \{aab, bba\}$ e vimos anteriormente que esta linguagem é gerada por uma gramática regular. Portanto, $L(G)$ é regular.

A gramática do exemplo anterior é um exemplo de gramática linear à esquerda, onde as regras podem ser da forma $A \to b$, $A \to \epsilon$ ou $A \to Ba$, com $A, B \in V$ e $a, b \in T$. A linguagem gerada por gramáticas deste tipo é regular.

Da mesma forma, gramáticas lineares à direita são aquelas em que as regras são da forma $A \to b$, $A \to \epsilon$ ou $A \to aB$, com $A, B \in V$ e $a, b \in T$. Estas gramáticas são o que definimos como gramáticas regulares. Alguns autores consideram ambos os tipos (tanto as lineares à esquerda quanto as lineares à direita) gramáticas regulares, por exemplo, Menezes (2000).

Regras simples da forma $A \to B$ não acrescentam poder a essas gramáticas. Estas regras podem ser substituídas para obtermos uma gramática regular equivalente. Considere o exemplo:

$S \to A$,

$S \to a$,

$A \to aB$,

$A \to B$,

$B \to b$.

Para esta gramática observamos que $S \Rightarrow^* A$, $S \Rightarrow^* B$, $A \Rightarrow^* B$,

portanto, podemos substituir o lado direito das regras simples da forma $C \to D$ pelo lado direito das regras cujo lado esquerdo é D. Neste caso obtemos a seguinte gramática regular equivalente:

$$S \to aB\,,$$
$$S \to b\,,$$
$$S \to a\,,$$
$$A \to aB\,,$$
$$A \to b\,,$$
$$B \to b\,.$$

Esta mesma gramática pode ser apresentada de forma mais concisa quando juntamos as regras que têm o mesmo lado esquerdo:

$$S \to aB \mid b \mid a,$$

$$A \to aB \mid b,$$

$$B \to b.$$

Portanto, a linguagem gerada por uma gramática apenas com regras regulares e regras simples é regular. Este resultado é importante porque nos permite apresentar um resultado central para esta seção, a saber: se L_1 e L_2 são linguagens regulares, então $L_1 \cup L_2$ também é regular.

Considere a gramática G_1 :

$$S_1 \rightarrow bA_1 ,$$

$$A_1 \rightarrow a .$$

e a gramática G_2 :

$$S_2 \rightarrow aA_2 \mid bB_2 ,$$

$$A_2 \rightarrow a ,$$

$$B_2 \rightarrow bB_2 \mid b .$$

Podemos construir a nova gramática G_3 tal que $L(G_3) = L(G_1) \cup L(G_2)$:

$S \to S_1 \mid S_2$,

$S_1 \to bA_1$,

$A_1 \to a$,

$S_2 \to aA_2 \mid bB_2$

$A_2 \to a$,

$B_2 \to bB_2 \mid b$.

Observe que G_3 é uma gramática apenas com regras regulares e simples, portanto, gera uma linguagem regular.

Voltemos a considerar a gramática G_2 :

$S_2 \to aA_2$,

$S_2 \to bB_2$,

$A_2 \to a$,

$B_2 \to bB_2$,

$B_2 \to b$.

Queremos agora encontrar uma gramática que gere a linguagem $L(G_2)^*$. Uma vez que G_2 só tem regras da forma $A \to b$ e $A \to aB$, e que o símbolo inicial não aparece do lado direito, essa construção será

bem fácil. Basta colocar o símbolo inicial nas regras da forma $A \to b$ e acrescentar a regra $S_2 \to \epsilon$, obtendo a gramática:

$$S_2 \to \epsilon$$
$$S_2 \to aA_2,$$
$$S_2 \to bB_2,$$
$$A_2 \to aS_2,$$
$$B_2 \to bB_2,$$
$$B_2 \to bS_2.$$