Conceitos Básicos de Linguagens

Bibliografia

1. *Linguagens Formais: Teoria, Modelagem e Implementação*
M.V.M. Ramos, J.J. Neto e I.S. Vega
Bookman, 2009

Roteiro

Símbolo, cadeia e alfabeto

Os **símbolos**, também denominados **palavras** ou **átomos**, são representações gráficas, indivisíveis, empregadas na construção de **cadeias**. Estas são formadas através da justaposição de um número finito de símbolos, obtidos de algum conjunto finito não-vazio, denominado **alfabeto**.

Símbolo, cadeia e alfabeto

Cada símbolo é considerado como uma unidade atômica, não importando a sua particular representação visual. São exemplos de símbolos: $a, abc, begin, if, 5, 1024, 2.017e4$. Perceba-se que não há uma definição formal para "símbolo". Deve-se intuir o seu significado como entidade abstrata, e dessa forma aceitá-lo como base para a teoria que será desenvolvida. Pode-se dizer que se trata de um conceito primitivo.

Convenções

Ao longo deste texto será adotada a seguinte convenção para denotar símbolos, cadeias e alfabetos:

- Símbolos: letras minúsculas do início do alfabeto romano: $(a, b, c...)$.
- Cadeias: letras minúsculas do final do alfabeto romano $(r, s, x, w...)$, ou letras minúsculas do alfabeto grego $(\alpha, \beta, \gamma...)$.
- Alfabetos: letras maiúsculas do alfabeto grego $(\Sigma, \Gamma, \Delta...)$.

Exemplo

Exemplo 1.1
Como exemplo de alfabeto podemos mencionar o conjunto Σ dos dígitos hexadecimais, em que cada elemento (dígito) desse conjunto corresponde a um determinado símbolo:

$$\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, e, f\}$$

Naturalmente, as cadeias que podem ser construídas a partir dos símbolos desse alfabeto correspondem aos numerais hexadecimais: $123, a0b56, fe5dc, b, abc, 55efff$ ...

Comprimento de uma cadeia

O **comprimento** de uma cadeia é um número natural que designa a quantidade de símbolos que a compõem. O comprimento de uma cadeia α é denotado por $|\alpha|$.

Exemplo 1.2
Considerem-se as cadeias $\alpha = 1, \beta = 469, \chi = bce60$ e $\phi = df$, também construídas sobre o alfabeto Σ do Exemplo 1.1. Então, $|\alpha| = 1, |\beta| = 3, |\chi| = 5$ e $|\phi| = 2$.

Cadeia unitária

Dá-se o nome de **cadeia elementar** (ou **unitária**) a qualquer cadeia formada por um único símbolo, como é o caso da cadeia α do Exemplo 1.2. Naturalmente, toda cadeia unitária tem comprimento 1.

Cadeia vazia

O conceito de **cadeia vazia** é especialmente importante na teoria das linguagens formais. Denota-se por ε a cadeia formada por uma quantidade nula de símbolos, isto é, a cadeia que não contém nenhum símbolo. Formalmente,

$$|\varepsilon| = 0$$

Concatenação de cadeias

Duas cadeias, sejam elas elementares ou não, podem ser anexadas, formando uma só cadeia, através da operação de **concatenação**. Essa operação fornece como resultado uma nova cadeia, formada pela justaposição ordenada dos símbolos que compõem os seus operandos separadamente. Observe-se que a operação de concatenação entre uma cadeia e um símbolo é realizada através da concatenação da cadeia em questão com a cadeia elementar correspondente ao símbolo.
Denota-se a concatenação de duas cadeias α e β como $\alpha \cdot \beta$ ou, simplesmente, $\alpha\beta$.

Exemplos

Exemplo 1.3
Considere o alfabeto $\Sigma = \{a, b, c, d\}$, e as cadeias $\alpha = abc$, $\beta = dbaca$ e $\sigma = a$.
A concatenação da cadeia α com a cadeia β é assim obtida:
$\alpha \cdot \beta = \alpha\beta = abcdbaca$, e $|\alpha\beta| = |\alpha| + |\beta| = 3 + 5 = 8$
Da mesma forma, obtém-se a concatenação de β com α:
$\beta \cdot \alpha = \beta\alpha = dbacaabc$, e, $|\beta\alpha| = |\beta| + |\alpha| = 5 + 3 = 8$
Note-se que, neste exemplo, $\alpha\beta \neq \beta\alpha$.
A concatenação da cadeia α com a cadeia elementar σ é dada por:
$\alpha \cdot \sigma = \alpha\sigma = abca$, e $|\alpha\sigma| = |\alpha| + |\sigma| = 3 + 1 = 4$
Finalmente, a concatenação da cadeia elementar σ com a cadeia β é obtida como:
$\sigma \cdot \beta = \sigma\beta = adbaca$, e $|\sigma\beta| = |\sigma| + |\beta| = 1 + 5 = 6$

Propriedades da concatenação de cadeias

Como se pode perceber, a operação de concatenação, embora associativa, não é comutativa. Dadas três cadeias α, β, γ quaisquer, pode-se sempre afirmar que $(\alpha\beta)\gamma = \alpha(\beta\gamma)$.
Por outro lado, dependendo dos particulares α e β considerados, pode ser que ou $\alpha\beta \neq \beta\alpha$ ou $\alpha\beta = \beta\alpha$ (por exemplo, se α e/ou β forem cadeias vazias ou, ainda, se $\alpha = \beta$).
No caso da cadeia vazia ε (elemento neutro em relação ao operador de concatenação) são válidas as seguintes relações:

1. $\alpha\varepsilon = \varepsilon\alpha = \alpha$
2. $|\alpha\varepsilon| = |\varepsilon\alpha| = |\alpha|$

Prefixos e sufixos

Diz-se que uma cadeia α é um **prefixo** de outra cadeia β se for possível escrever β como $\alpha\gamma$. A cadeia α é dita **sufixo** de β se β puder ser escrita como $\gamma\alpha$. Em ambos os casos, admite-se a possibilidade de $\gamma = \varepsilon$. Nos casos em que $\gamma \neq \varepsilon$, diz-se que α é, respectivamente, **prefixo próprio** ou **sufixo próprio** da cadeia β. Note que a cadeia vazia ε pode ser considerada simultaneamente prefixo ou sufixo de qualquer cadeia.

Subcadeias

Dadas quatro cadeias α, β, γ e δ, uma cadeia α é chamada **subcadeia** de uma cadeia β sempre que $\beta = \gamma\alpha\delta$. Note-se que, se γ ou δ ou ambos forem vazios, a definição também se aplica. Note-se também que prefixos e sufixos são casos particulares de subcadeias.

Cadeia reversa

Uma cadeia α é dita o **reverso** de uma cadeia β, denotando-se o fato por $\alpha = \beta^R$, se α contiver os mesmos símbolos que β, porém justapostos no sentido inverso, ou seja:

$$\text{se } \alpha = \sigma_1\sigma_2...\sigma_{n-1}\sigma_n \text{ então } \beta = \sigma_n\sigma_{n-1}...\sigma_2\sigma_1$$

Por definição, $\varepsilon^R = \varepsilon$.

Exemplo 1.4
Considerem-se as cadeias $\alpha = 123abc$ e $\beta = d$. Então, $\alpha^R = cba321$ e $\beta^R = d$.

Repetição de símbolos

Finalmente, convenciona-se que σ^i representa a cadeia formada por "i" símbolos σ concatenados. Por definição, $\sigma^0 = \varepsilon$.

Exemplo 1.5
Considere-se o símbolo a. Então:

- $a^0 = \varepsilon$;
- $a^1 = a$;
- $a^2 = aa$;
- $a^3 = aaa$;
- etc.

Linguagem formal

Uma **linguagem formal** é um conjunto, finito ou infinito, de cadeias de comprimento finito, formadas pela concatenação de elementos de um alfabeto finito e não-vazio. Além das operações previamente definidas para conjuntos, como união, diferença, intersecção etc., outras operações, tais como a concatenação e os fechamentos, também são fundamentais para a definição e o estudo das linguagens formais.

Cadeia vazia e conjunto vazio

Convém notar a distinção que há entre os seguintes conceitos:

- cadeia vazia ε;
- conjunto vazio $\emptyset$;
- conjunto que contém apenas a cadeia vazia $\{\varepsilon\}$;
- conjunto que contém apenas o conjunto vazio $\{\emptyset\}$.

O primeiro deles, ε, denota a cadeia vazia, ou seja, uma cadeia de comprimento zero, ao passo que os demais são casos particulares de conjuntos: $\emptyset$ denota uma linguagem vazia, ou seja, uma linguagem que não contém nenhuma cadeia, $\{\varepsilon\}$ denota uma linguagem que contém uma única cadeia (a cadeia vazia), e $\{\emptyset\}$ denota um conjunto que contém um único elemento, o conjunto vazio. Observe-se que $|\varepsilon| = |\emptyset| = 0$ e $|\{\varepsilon\}| = |\{\emptyset\}| = 1$.

Alfabetos, linguagens e cadeias

Note-se a diferença conceitual que há entre alfabetos, linguagens e cadeias. Alfabetos são conjuntos, finitos e não-vazios, de símbolos, através de cuja concatenação são obtidas as **cadeias**. Linguagens, por sua vez, são conjuntos, finitos (eventualmente vazios) ou infinitos, de cadeias. Uma cadeia é também denominada **sentença** de uma linguagem, ou simplesmente sentença, no caso de ela pertencer à linguagem em questão. Linguagens são, portanto, coleções de sentenças sobre um dado alfabeto.

Símbolos, alfabeto, cadeias, linguagem

A Figura 1 ilustra a relação entre os conceitos de símbolo, alfabeto, cadeia e linguagem.

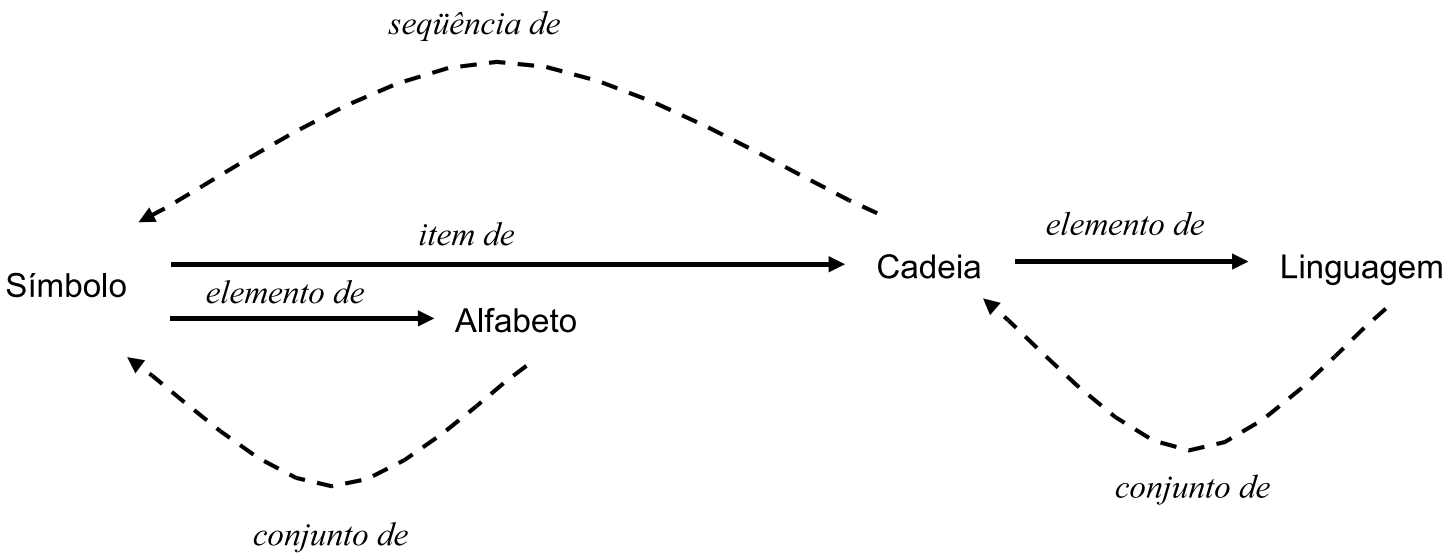
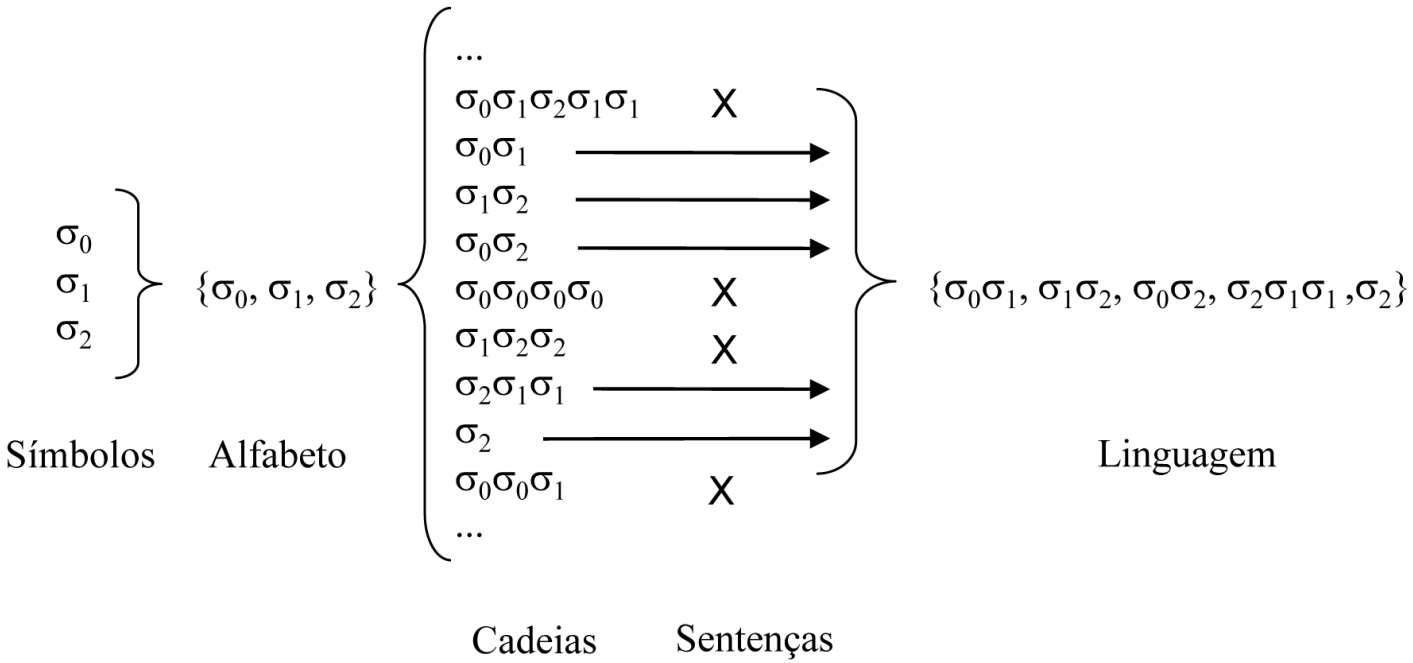


Figura 1: Símbolo, alfabeto, cadeia e linguagem

Símbolos, alfabeto, cadeias, linguagem

Outra maneira de associar significados aos termos “símbolo”, “alfabeto”, “cadeia” e “linguagem” é apresentada na Figura 2, que também ilustra o conceito de “sentença”.



Exemplo

Exemplo 2.1
O símbolo a é elemento do alfabeto $\{a\}$ e também um item da cadeia aaa, que por sua vez é elemento da linguagem $\{aaa\}$. Por outro lado, a linguagem $\{aaa\}$ é um conjunto que contém a cadeia aaa, a cadeia aaa é uma seqüência de símbolos a e o alfabeto $\{a\}$ contém o símbolo a. A Figura 3 ilustra esses conceitos, conforme a Figura 1.

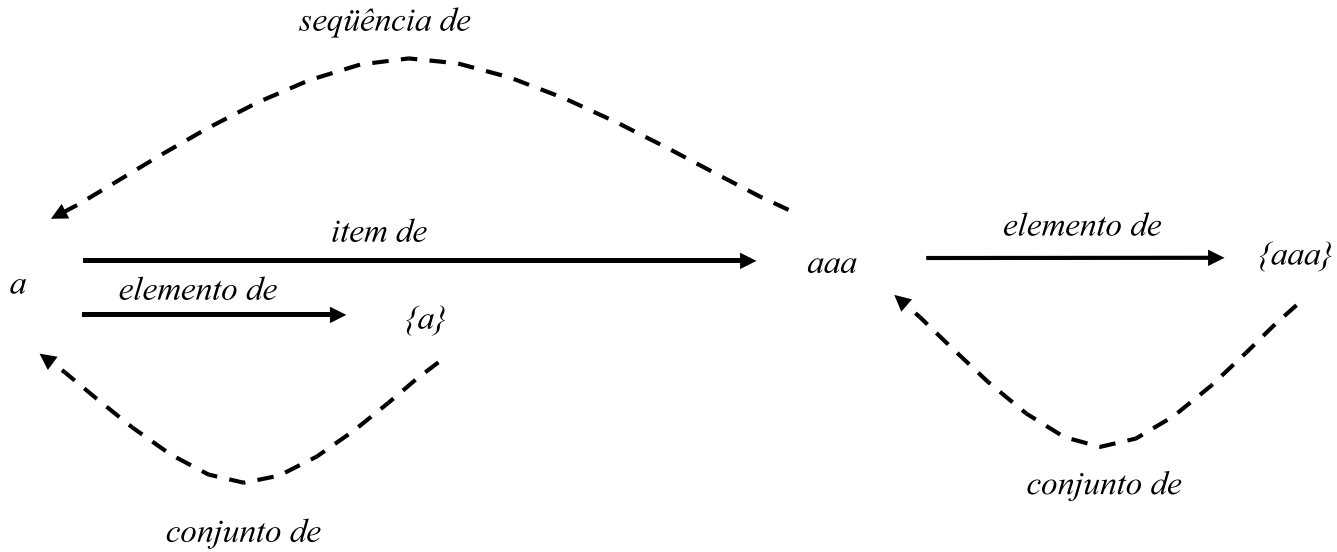


Figura 3: $a, \{a\}, aaa, \{aaa\}$

Exemplo

Exemplo 2.2
A Figura 4 ilustra uma aplicação dos conceitos da Figura 2 ao alfabeto $\{a, b\}$. A linguagem apresentada é, naturalmente, apenas uma das inúmeras que podem ser criadas a partir desse alfabeto.

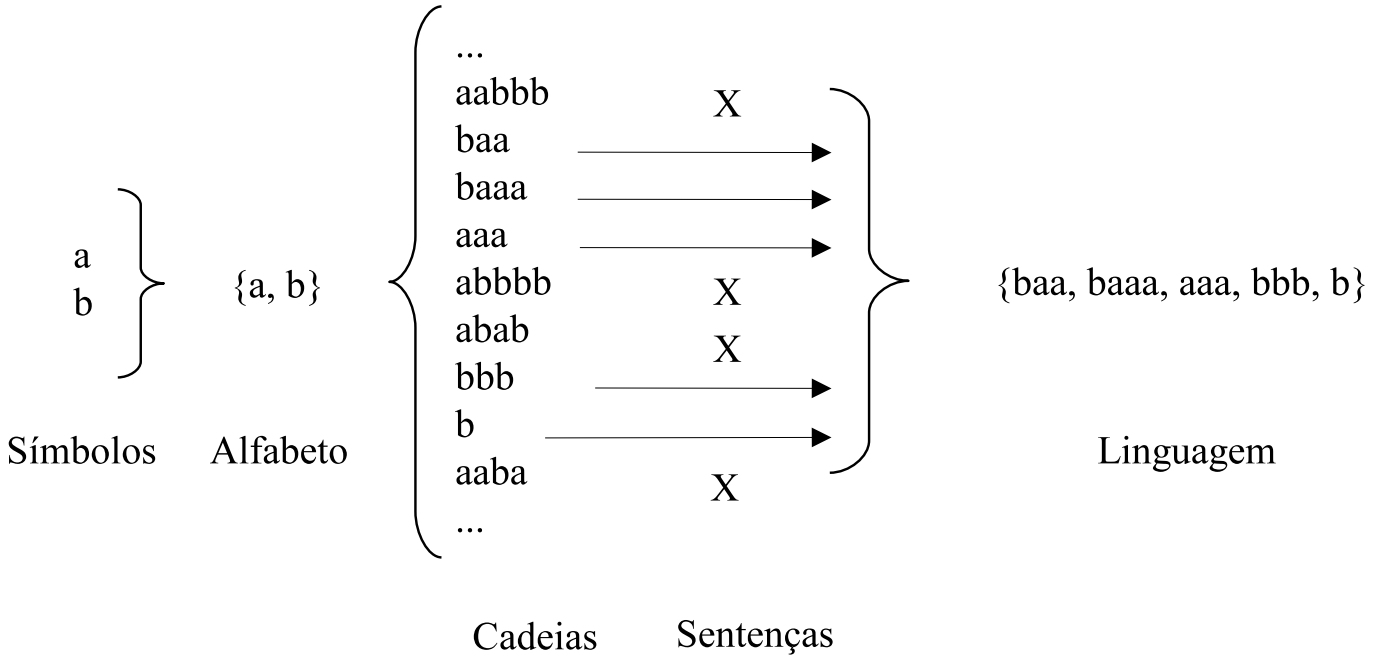


Figura 4: Símbolos a e b, cadeias, sentenças e linguagem

Concatenação de linguagens

A **concatenação** de duas linguagens X e Y, denotada por $X \cdot Y$ ou simplesmente XY, corresponde a um conjunto Z formado pela coleção de todas as cadeias que possam ser obtidas pela concatenação de cadeias $x \in X$ com cadeias $y \in Y$, nesta ordem. Formalmente,

$$Z = X \cdot Y = XY = \{xy \mid x \in X \text{ e } y \in Y\}$$

A concatenação $\Sigma\Sigma$, que gera cadeias de comprimento 2 formadas sobre um alfabeto Σ, é também representada por Σ^2. Analogamente, a concatenação $\Sigma\Sigma\Sigma$, que gera cadeias de comprimento 3 sobre o alfabeto Σ, é representada como Σ^3, e assim sucessivamente. Generalizando-se:

$$\Sigma^i = \Sigma\Sigma^{i-1}, i > 0$$

Por definição, $\Sigma^0 = \{\varepsilon\}$

Exemplo

Exemplo 2.3
Considere-se $\Sigma = \{a, b, c\}$. Então,

$$\begin{aligned} \Sigma^0 &= \{\varepsilon\} \\ \Sigma^1 &= \{a, b, c\} \\ \Sigma^2 &= \{aa, ab, ac, ba, bb, bc, ca, cb, cc\} \\ \Sigma^3 &= \{aaa, aab, aac, aba, abb, abc, \ldots, ccc\} \end{aligned}$$

etc.

Fechamento reflexivo e transitivo

O **fechamento reflexivo e transitivo** (às vezes chamado **fechamento recursivo e transitivo**) de um alfabeto Σ é definido como o conjunto (infinito) que contém todas as possíveis cadeias que podem ser construídas sobre o alfabeto dado, incluindo a cadeia vazia. Esse conjunto, denotado por Σ^*, contém, naturalmente, todas as cadeias que podem ser definidas sobre o alfabeto Σ. Formalmente, o fechamento reflexivo e transitivo de um conjunto Σ é definido como:

$$\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup ... = \bigcup_{i=0}^{\infty} \Sigma^i$$

Fechamento reflexivo e transitivo

A definição de uma linguagem pode, portanto, ser formulada de maneira mais rigorosa com o auxílio da operação de fechamento reflexivo e transitivo: sendo uma linguagem qualquer coleção de cadeias sobre um determinado alfabeto Σ, e como Σ^* contém todas as possíveis cadeias sobre Σ, então toda e qualquer linguagem L sobre um alfabeto Σ sempre poderá ser definida como sendo um subconjunto de Σ^*, ou seja, $L \subseteq \Sigma^*$.

“Maior” linguagem

Diz-se que a **maior** linguagem que se pode definir sobre um alfabeto Σ, observando-se um conjunto qualquer P de propriedades, corresponde ao conjunto de **todas** as cadeias $w \in \Sigma^*$ tais que w satisfaz simultaneamente a **todas** as propriedades $p_i \in P$.
De uma forma geral, sempre que for feita uma referência a uma determinada linguagem L cujas cadeias satisfaçam a um certo conjunto de propriedades P, estará implícita (a menos de ressalva em contrário) a condição de que se trata da maior linguagem definida sobre L, cujas cadeias satisfaçam o conjunto de propriedades P.

“Maior” linguagem

Um caso particular importante a se considerar é a linguagem cujo conjunto P de propriedades seja o menos restritivo possível, considerando toda e qualquer cadeia de qualquer comprimento (finito) como sendo válida. Assim, a maior linguagem dentre todas as que podem ser definidas sobre um alfabeto Σ qualquer é $L = \Sigma^*$ (note-se, neste caso, que a única propriedade a ser satisfeita pelas cadeias de L é que elas “sejam definidas sobre Σ”, ou seja, obtidas a partir da simples justaposição de símbolos de Σ). Qualquer outra linguagem definida sobre esse mesmo alfabeto corresponderá obrigatoriamente a um subconjunto (eventualmente próprio) de Σ^*.

“Menor” linguagem

Por outro lado, a **menor** linguagem que pode ser definida sobre um alfabeto Σ qualquer é $\emptyset$, ou seja, a linguagem vazia ou a linguagem composta por zero sentenças.

Todas as linguagens

Finalmente, como o conjunto de todos os subconjuntos possíveis de serem obtidos a partir de Σ^* é 2^{Σ^*}, tem-se que 2^{Σ^*} representa o conjunto de **todas** as linguagens que podem ser definidas sobre o alfabeto Σ.

Menor, maior, todas

Em resumo:

- $\emptyset$ é o conjunto constituído por zero cadeias e corresponde à menor linguagem que se pode definir sobre um alfabeto Σ qualquer;
- Σ^* é o conjunto de todas as cadeias possíveis de serem construídas sobre Σ e corresponde à maior de todas as linguagens que pode ser definida sobre Σ;
- 2^{Σ^*} é o conjunto de todos os subconjuntos possíveis de serem obtidos a partir de Σ^*, e corresponde ao conjunto formado por todas as possíveis linguagens que podem ser definidas sobre Σ. Observe-se que $\emptyset \in 2^{\Sigma^*}$, e também que $\Sigma^* \in 2^{\Sigma^*}$.

Exemplos

Exemplo 2.4
Seja $\Sigma = \{a, b, c\}$ e P o conjunto formado pela única propriedade "todas as cadeias são iniciadas com o símbolo a". Então:

- A linguagem $L_0 = \emptyset$ é a menor linguagem que pode ser definida sobre Σ;
- A linguagem $L_1 = \{a, ab, ac, abc, acb\}$ é finita e observa P;
- A linguagem $L_2 = \{a\}\{a\}^*\{b\}^*\{c\}^*$ é infinita e observa P;
- A linguagem $L_3 = \{a\}\{a, b, c\}^*$ é infinita, observa P e, dentre todas as que observam P, trata-se da maior linguagem, pois não existe nenhuma outra cadeia em Σ^* que satisfaça a P e não pertença a L_3;
- $L_0 \subseteq \Sigma^*, L_1 \subseteq \Sigma^*, L_2 \subseteq \Sigma^*, L_3 \subseteq \Sigma^*$;
- $L_0 \in 2^{\Sigma^*}, L_1 \in 2^{\Sigma^*}, L_2 \in 2^{\Sigma^*}, L_3 \in 2^{\Sigma^*}$;
- Além de L_0, L_1, L_2 e L_3, existem inúmeras outras linguagens que podem ser definidas sobre Σ.

Fechamento transitivo

O **fechamento transitivo** de um alfabeto Σ, denotado por Σ^+, é definido de maneira análoga ao fechamento reflexivo e transitivo, diferindo deste apenas por não incluir o conjunto Σ^0:

$$\Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \ldots = \bigcup_{i=1}^{\infty} \Sigma^i$$

Como se pode perceber, $\Sigma^* = \Sigma^+ \cup \{\varepsilon\}$. Observe-se ainda que, embora aparentemente seja conseqüência da anterior, a afirmação $\Sigma^+ = \Sigma^* - \{\varepsilon\}$ só será válida nos casos em que Σ não contiver a cadeia vazia.

Exemplo

Exemplo 2.5
Seja $\Sigma = \{n, (,), +, *, -, /\}$. Neste caso:

- $\Sigma^* = \{\varepsilon, n, n+n, -n), */, n()), n-(n*n), ...\}$
- $\Sigma^+ = \{n, n+n, -n), */, n()), n-(n*n), ...\}$
- $\Sigma^+ = \Sigma^* - \{\varepsilon\}$, pois $\varepsilon \notin \Sigma$

Complementação de uma linguagem

A **complementação** de uma linguagem X definida sobre um alfabeto Σ é definida como:

$$\overline{X} = \Sigma^* - X$$

Reverso de uma linguagem

Diz-se que uma linguagem L_1 é o **reverso** de uma linguagem L_2, denotando-se o fato por $L_1 = L_2^R$ (ou $L_2 = L_1^R$), quando as sentenças de L_1 corresponderem ao reverso das sentenças de L_2. Formalmente:

$$L_1 = L_2^R = \{x^R \mid x \in L_2\}$$

Exemplo 2.6
Seja $L_2 = \{\varepsilon, a, ab, abc\}$. Então, $L_1 = L_2^R = \{\varepsilon, a, ba, cba\}$.

Exemplo

Conforme já mencionado, o conjunto Σ^* contém todas as possíveis linguagens que podem ser definidas sobre o alfabeto Σ. Na prática, as linguagens de interesse costumam ser definidas como um subconjunto próprio de Σ^*, para um dado alfabeto Σ.

Exemplo 2.7
Considere-se o alfabeto $\Sigma_1 = \{n, (,), +, *, -, /\}$. Uma linguagem L_1, passível de ser definida sobre o alfabeto Σ_1, e que corresponde a um subconjunto próprio de Σ_1^*, é aquela em que as cadeias se assemelham, do ponto vista estrutural, às cadeias que representam expressões aritméticas bem-formadas em muitas linguagens populares de programação de alto nível.
A seguir estão relacionadas algumas das cadeias que fazem parte de L_1, de acordo com esse critério.

Exemplo

- n
- n+n
- (n*n)
- n*(n/(n+n+n))

Portanto, $L_1 = \{n, n+n, (n*n), n*(n/(n+n+n)), ...\} \subset \Sigma_1^*$. São exemplos de cadeias pertencentes a $\Sigma_1^* - L_1$:

- n++
- nn*
- (n*n)))
- n*-(n(+n+/))
- ε

Métodos e notações

As linguagens de interesse prático, como é o caso das linguagens de programação, normalmente correspondem a um subconjunto próprio do fechamento reflexivo e transitivo do alfabeto sobre o qual as suas cadeias são construídas. Assim, tornam-se necessários métodos e notações que permitam, dentro do conjunto fechamento, identificar as cadeias que efetivamente pertencem à linguagem que estiver sendo definida, descartando-se as demais.

Métodos e notações

Um outro aspecto importante, referente à definição rigorosa das linguagens formais, diz respeito ao fato de que, em sua maioria, as linguagens de interesse contêm, se não uma quantidade infinita, ao menos um número finito, porém muito grande, de cadeias.

Métodos e notações

Por esses dois motivos, há um interesse muito grande em relação a métodos que permitam especificar linguagens, sejam elas finitas ou não, através de representações finitas. Por outro lado, nem todas as linguagens podem ser representadas por meio de uma especificação finita. Apesar do reduzido interesse prático que recai sobre tais linguagens, é possível comprovar a existência de linguagens para as quais é impossível se obter uma representação finita.

Gramáticas

1. **Gramáticas**: correspondem a especificações finitas de dispositivos de geração de cadeias. Um dispositivo desse tipo deve ser capaz de gerar toda e qualquer cadeia pertencente à linguagem definida pela gramática, e nada mais. Assim, as cadeias não pertencentes à linguagem não devem poder ser geradas pela gramática em questão. Essa forma de especificação é aplicável para linguagens finitas e infinitas.

Reconhecedores

2 **Reconhecedores**: correspondem a especificações finitas de dispositivos de aceitação de cadeias. Um dispositivo desse tipo deverá aceitar toda e qualquer cadeia pertencente à linguagem por ele definido, e rejeitar todas as cadeias não-pertencentes à linguagem. O método é aplicável para a especificação formal de linguagens finitas e infinitas.

Enumerações

3 **Enumerações**: relacionam, de forma explícita e exaustiva, todas as cadeias pertencentes à particular linguagem a ser especificada. Toda e qualquer cadeia pertencente à linguagem deve constar desta relação. As cadeias não pertencentes à linguagem não fazem parte dessa relação. Aplicam-se apenas para a especificação de linguagens finitas e preferencialmente não muito extensas.

Usos

Na prática, as gramáticas e os reconhecedores, além de oferecerem uma grande concisão na representação de linguagens de cardinalidade elevada, também podem ser empregados na definição de linguagens infinitas, ao contrário das enumerações. Em contraste com o que ocorre com as enumerações, as gramáticas e os reconhecedores geralmente possibilitam uma percepção melhor da estrutura sintática inerente às sentenças das linguagens por eles definidas.

Equivalências

Diz-se que uma gramática é **equivalente** a um reconhecedor se as duas seguintes condições forem simultaneamente verificadas (lembrar que os formalismos devem definir todas as sentenças da linguagem desejada, e nenhuma outra cadeia):

1. Toda cadeia gerada pela gramática é também aceita pelo reconhecedor.
2. Toda cadeia aceita pelo reconhecedor é também gerada pela gramática.

Exemplo

Exemplo 2.8
Considerem-se a gramática G e o reconhecedor M, respectivamente definidos através das linguagens gerada e aceita:

- $G \mid L_1(G) = \{\alpha \in \{a,b\}^* \mid$ o primeiro símbolo de α é "a"$\}$
- $M \mid L_2(M) = \{\beta \in \{a,b\}^* \mid$
  o primeiro símbolo de β é "a" e o último símbolo é "b"$\}$

Portanto:

- $L_1 = \{a, aa, ab, aaa, aab, aba, abb, aaa...\}$
- $L_2 = \{ab, aab, abb, aaab, aabb, abab, abbb...\}$

É fácil perceber que a condição (2) acima é verificada, mas a condição (1) não. Por exemplo, a cadeia $ab \in L_2$ e $ab \in L_1$. Por outro lado, a cadeia $aba \in L_1$, porém $aba \notin L_2$. Logo, $L_1 \subset L_2$ e não se pode dizer que G e M sejam equivalentes.

Metalinguagem

Conforme discutido mais adiante, as gramáticas e os reconhecedores são formas duais de representação de linguagens, ou seja, para cada gramática é possível obter um reconhecedor que aceite a linguagem correspondente e vice-versa. À particular notação utilizada para representar uma linguagem, seja através de gramáticas ou de reconhecedores, dá-se o nome de **metalinguagem**.

Conceito

Também conhecidas como **dispositivos generativos**, **dispositivos de síntese**, ou ainda dispositivos de geração de cadeias, as **gramáticas** constituem sistemas formais baseados em regras de substituição, através dos quais é possível sintetizar, de forma exaustiva, o conjunto das cadeias que compõem uma determinada linguagem.

Conceito

Assim como ocorre no caso das linguagens naturais, as linguagens formais também podem ser especificadas através de “gramáticas” a elas associadas. No caso das gramáticas das linguagens formais, que constituem o objeto deste estudo, a analogia com as “gramáticas” das linguagens naturais é muito grande. Tratam-se, as primeiras, de conjuntos de regras que, quando aplicadas de forma recorrente, possibilitam a geração de todas as cadeias pertencentes a uma determinada linguagem.

Metalinguagens

Diferentemente das gramáticas das linguagens naturais, que são descritas por intermédio de linguagens também naturais (muitas vezes a mesma que está sendo descrita pela gramática), as gramáticas das linguagens formais são descritas utilizando notações matemáticas rigorosas que visam, entre outros objetivos, evitar dúvidas na sua interpretação. Tais notações recebem a denominação de **metalinguagens** — linguagens que são empregadas para definir outras linguagens.

Definição

Formalmente, uma gramática G pode ser definida como sendo uma quádrupla:

$$G = (V, \Sigma, P, S)$$

onde:

- V é o **vocabulário** da gramática; corresponde a um conjunto (finito e não-vazio) de símbolos;
- Σ é o conjunto (finito e não-vazio) dos símbolos **terminais** da gramática; também denominado **alfabeto**;
- P é o conjunto (finito e não-vazio) de **produções** ou **regras de substituição** da gramática;
- S é a **raiz** (ou **símbolo inicial**) da gramática, $S \in V$.

Definição

Adicionalmente, define-se $N = V - \Sigma$ como sendo o conjunto dos símbolos **não-terminais** da gramática. Σ corresponde ao conjunto dos símbolos que podem ser justapostos para compor as sentenças da linguagem que se está definindo, e N corresponde ao conjunto dos símbolos intermediários (classes sintáticas) utilizados na estruturação e na geração de sentenças, sem no entanto fazer parte das mesmas. Σ, N e P são conjuntos finitos e não-vazios. P é o conjunto das produções gramaticais, que obedecem à forma geral:

$$\alpha \rightarrow \beta, \quad \text{com } \alpha \in V^* N V^* \text{ e } \beta \in V^*$$

Definição

De fato, "$\rightarrow$" é uma relação sobre os conjuntos V^*NV^* e V^*, uma vez que:

$$P = \{(\alpha, \beta) \mid (\alpha, \beta) \in V^*NV^* \times V^*\}$$

ou seja, P é um subconjunto de $V^*NV^* \times V^*$.

Exemplo

Exemplo 3.1
Seja $G_1 = (V_1, \Sigma_1, P_1, S)$, com:

$$\begin{aligned} V_1 &= \{0,1,2,3,S,A\} \\ \Sigma_1 &= \{0,1,2,3\} \\ N_1 &= \{S,A\} \\ P_1 &= \{S \to 0S33, S \to A, A \to 12, A \to \varepsilon\} \end{aligned}$$

É fácil verificar que G_1 está formulada de acordo com as regras gerais acima enunciadas para a especificação de gramáticas.

Forma sentencial

Denomina-se **forma sentencial** qualquer cadeia obtida pela aplicação recorrente das seguintes regras de substituição:

1. S (a raiz da gramática) é por definição uma forma sentencial;
2. Seja $\alpha\rho\beta$ uma forma sentencial, com α e β cadeias quaisquer de terminais e/ou não-terminais da gramática, e seja $\rho \rightarrow \gamma$ uma produção da gramática. Nessas condições, a aplicação dessa produção à forma sentencial, substituindo a ocorrência de ρ por γ, produz uma nova forma sentencial $\alpha\gamma\beta$.

Derivação direta

Denota-se a substituição acima definida, também conhecida como **derivação direta**, por:

$$\alpha\rho\beta \Rightarrow_G \alpha\gamma\beta$$

O índice "G" designa o fato de que a produção aplicada, no caso $\rho \rightarrow \gamma$, pertence ao conjunto de produções que define a gramática G. Nos casos em que a gramática em questão puder ser facilmente identificada, admite-se a eliminação de referências explícitas a ela.

$\rightarrow$ e $\Rightarrow$

Note-se a distinção gráfica e de significado que se faz entre o símbolo empregado na denotação das produções da gramática ($\rightarrow$) e o símbolo utilizado na denotação das derivações ($\Rightarrow$).

Derivação e derivação não-trivial

Uma seqüência de zero ou mais derivações diretas, como, por exemplo, $\alpha \Rightarrow \beta \Rightarrow ... \Rightarrow \mu$ é chamada simplesmente **derivação**, e pode ser abreviada como $\alpha \Rightarrow^* \mu$.
Derivações em que ocorre a aplicação de pelo menos uma produção são denominadas **derivações não-triviais**, e são denotadas por $\alpha \Rightarrow^+ \mu$.

Sentença

Se, pela aplicação de uma derivação não-trivial à raiz S de uma gramática, for possível obter uma cadeia w formada exclusivamente de símbolos terminais, diz-se que w, além de ser uma forma sentencial, é também uma **sentença**, e denota-se a sua derivação por:

$$S \Rightarrow^{+} w$$

Substituições

Gramáticas são sistemas formais baseados na substituição, em uma forma sentencial, de uma cadeia, coincidente com o lado esquerdo de uma regra de produção, pela cadeia correspondente ao lado direito da mesma regra, visando com isso, ao final de uma seqüência de substituições, a obtenção de uma cadeia sobre Σ. O processo de substituição tem início sempre a partir da raiz S da gramática, e é finalizado assim que for obtida uma forma sentencial isenta de símbolos não-terminais, isto é, assim que se obtiver uma sentença.

Exemplo

Exemplo 3.2
Considere-se a gramática G_1, definida no Exemplo 3.1.

- S é por definição uma forma sentencial;
- $0S33$ é uma forma sentencial, pois $S \Rightarrow 0S33$;
- $S \Rightarrow 0S33$ é uma derivação direta;
- $00S3333$ e $00A3333$ são formas sentenciais, pois $0S33 \Rightarrow 00S3333 \Rightarrow 00A3333$ através das produções $S \rightarrow 0S33$ e $S \rightarrow A$, aplicadas nesta ordem;
- $S \Rightarrow^+ 00A3333$ e $S \Rightarrow^+ 0S33$ são exemplos de derivações não-triviais;
- $00S3333 \Rightarrow^* 00S3333$ e $0S33 \Rightarrow^* 00A3333$ são exemplos de derivações;
- 12 e 00123333 são exemplos de sentenças, pois ambas são formadas exclusivamente por símbolos terminais e $S \Rightarrow A \Rightarrow 12$, ou seja, $S \Rightarrow^+ 12$, e $S \Rightarrow 0S33 \Rightarrow 00S3333 \Rightarrow 00A3333 \Rightarrow 00123333$, ou seja, $S \Rightarrow^+ 00123333$.

Linguagem definida por uma gramática

Ao conjunto de todas as sentenças w geradas por uma gramática G dá-se o nome de **linguagem definida pela gramática** G, ou simplesmente $L(G)$. Formalmente,

$$L(G) = \{w \in \Sigma^* \mid S \Rightarrow^+ w\}$$

Exemplo 3.3
Pela inspeção das produções da gramática G_1 do Exemplo 3.1, pode-se concluir que:

$$L_1(G_1) = \{0^m 1^n 2^n 3^{2m} \mid m \geqslant 0 \text{ e } (n = 0 \text{ ou } n = 1)\}$$

São exemplos de sentenças pertencentes a L_1 : $\varepsilon, 12, 033, 01233, 003333, 00123333$ etc.

Linguagem definida por uma gramática

Como se pode perceber, diferentes seqüências de produções aplicadas à (única) raiz da gramática possibilitam a geração das diferentes (em geral, infinitas) sentenças da linguagem. Por outro lado, muitas vezes há mais de uma alternativa de substituição para um mesmo trecho de uma forma sentencial, ou, ainda, pode haver mais de um trecho em uma mesma forma sentencial que pode ser objeto de substituição pelo lado direito de alguma produção. Assim, a identificação correta das sentenças de uma dada linguagem deve ser feita levando-se em conta a possibilidade de ocorrência de todos esses fatores durante o processo de síntese de cadeias.

Linguagem definida por uma gramática

A completa identificação da linguagem gerada por uma determinada gramática é uma tarefa que exige abstração e alguma prática na manipulação das produções. Algumas gramáticas, como é o caso de G_1, podem ser analisadas sem dificuldade. Contudo, na prática, podem ocorrer gramáticas consideravelmente mais complexas. Observe-se, a título de ilustração, o caso da gramática G_2 apresentada no Exemplo 3.4.

Exemplo

Exemplo 3.4
Considere $G_2 = (V_2, \Sigma_2, P_2, S)$, com:

$$\begin{aligned} V_2 &= \{a, b, c, S, B, C\} \\ \Sigma_2 &= \{a, b, c\} \\ P_2 &= \{S \to aSBC, S \to abC, CB \to BC, bB \to bb, bC \to bc, cC \to cc\} \end{aligned}$$

A linguagem gerada por G_2 é $\{a^n b^n c^n \mid n \geqslant 1\}$. De fato, a seqüência de derivações iniciada com a regra $S \to abC$ conduz à geração da sentença abc ($S \Rightarrow abC \Rightarrow abc$). Por outro lado, seqüências iniciadas com a aplicação repetida i vezes da regra $S \to aSBC$ conduzem à geração da seguinte forma sentencial subseqüente:

$$S \Rightarrow^i a^i S(BC)^i$$

A posterior aplicação da regra $S \to abC$ faz com que:

$$a^i S(BC)^i \Rightarrow a^i abC(BC)^i = a^{i+1} b(CB)^i C$$

Exemplo

Através da aplicação sucessiva da regra $CB \rightarrow BC$, obtém-se agora:

$$a^{i+1}b(CB)^iC \Rightarrow^* a^{i+1}bB^iC^iC = a^{i+1}bB^iC^{i+1}$$

Finalmente, a aplicação i vezes da regra $bB \rightarrow bb$ faz com que todos os símbolos B sejam substituídos por símbolos b:

$$a^{i+1}bB^iC^{i+1} \Rightarrow^i a^{i+1}bb^iC^{i+1} = a^{i+1}b^{i+1}C^{i+1}$$

A aplicação uma única vez da regra $bC \rightarrow bc$ substitui o primeiro símbolo da cadeia de símbolos C pelo símbolo c:

$$a^{i+1}b^{i+1}C^{i+1} \Rightarrow a^{i+1}b^{i+1}cC^i$$

Para terminar, a aplicação i vezes da regra $cC \rightarrow cc$ substitui todos os demais símbolos C por símbolos c:

$$a^{i+1}b^{i+1}cC^i \Rightarrow^i a^{i+1}b^{i+1}cc^i = a^{i+1}b^{i+1}c^{i+1}$$

Exemplo

A forma sentencial:

$$a^{i+1}b^{i+1}c^{i+1}$$

gera, portanto, as sentenças *aabbcc*, *aaabbbccc* etc. A sentença *aabbcc*, por exemplo, é derivada da seguinte forma nessa gramática:

$$S \Rightarrow aSBC \Rightarrow aabCBC \Rightarrow aabBCC \Rightarrow aabbCC \Rightarrow aabbbcC \Rightarrow aabbcc$$

pela aplicação, respectivamente, das produções:

$$S \rightarrow aSBC, S \rightarrow abC, CB \rightarrow BC, bB \rightarrow bb, bC \rightarrow bc \text{ e } cC \rightarrow cc$$

Exercício

1. Obter gramáticas que geram as linguagens cujas sentenças satisfazem às regras apresentadas a seguir. Considerar $\Sigma = \{a, b\}$;
2. Repetir, considerando $\Sigma = \{a, b, c\}$.

Exercício

- Começam com aa;
- Não começam com aa;
- Terminam com bbb;
- Não terminam com bbb;
- Contém a subcadeia $aabbb$;
- Não contém a subcadeia aaa;
- Possuem comprimento maior ou igual a 3;
- Possuem comprimento menor ou igual a 3;
- Possuem comprimento diferente de 3;
- Possuem comprimento par;
- Possuem comprimento ímpar;
- Possuem comprimento múltiplo de 4;
- Possuem quantidade par de símbolos a;
- Possuem quantidade ímpar de símbolos b.

Gramáticas equivalentes

É possível definir uma mesma linguagem através de duas ou mais gramáticas distintas. Quando isso ocorre, diz-se que as gramáticas que definem a linguagem em questão são sintaticamente equivalentes ou, simplesmente, **equivalentes** uma à outra.

Exemplo 3.5
As gramáticas G_3 e G_4 a seguir definidas são equivalentes:

$$\begin{aligned} G_3 &= (\{a,b,S\},\{a,b\},\{S \to aS, S \to a, S \to bS, S \to b, S \to aSb\},S) \\ G_4 &= (\{a,b,S,X\},\{a,b\},\{S \to XS, S \to X, X \to a, X \to b\},S) \end{aligned}$$

Uma rápida análise de G_3 e G_4 permite concluir que $L_3(G_3) = L_4(G_4) = \{a,b\}^+$.

Notação algébrica

São muitas as notações (metalinguagens) utilizadas na definição gramatical das linguagens formais e de programação. Nos exemplos acima, bem como na maior parte do presente texto, utiliza-se a **notação algébrica**. No entanto, diversas outras metalinguagens são largamente empregadas para representar dispositivos generativos. A escolha de uma ou outra metalinguagem varia de acordo com o tipo da linguagem que estiver sendo definida, com o uso que se pretenda fazer de tal representação formal e com as próprias preferências pessoais de quem a estiver elaborando.

Outras notações

À exceção da notação algébrica, empregada para representar diversos tipos de linguagens e utilizada com especial ênfase no estudo teórico das propriedades das linguagens formais, outras notações, como, por exemplo, as Expressões Regulares, o BNF (*Backus-Naur Form* ou Notação de Backus-Naur), a Notação de Wirth (ou Expressões Regulares Estendidas) e os Diagramas de Sintaxe (também conhecidos como Diagramas Ferroviários), são bastante populares e amplamente utilizadas na definição de linguagens de programação de alto nível.

Linguagens como conjuntos

Linguagens são conjuntos. Conjuntos que formam linguagens são coleções de cadeias construídas sobre um alfabeto, por exemplo, através de gramáticas. A fim de explicitar e facilitar o entendimento de linguagens vistas como conjuntos, é conveniente considerar-se uma coleção finita de linguagens distintas, todas infinitas, definidas sobre um mesmo alfabeto qualquer. Analisando-se as respectivas gramáticas, sugere-se que o leitor procure compreender de que modo as linguagens foram definidas e, principalmente, verifique a relação que existe entre elas.

Linguagens como conjuntos

Linguagens são conjuntos. Conjuntos que formam linguagens são coleções de cadeias construídas sobre um É interessante verificar: estas linguagens possuem elementos (sentenças) em comum? Qual é a sua intersecção? Alguma dessas linguagens é subconjunto de outra? Superconjunto? Subconjunto próprio? São inúmeras as possibilidades. O importante é desenvolver a percepção de que linguagens são conjuntos. Isso facilitará o estudo de novas operações sobre linguagens, e também de suas propriedades.

Exemplo

Exemplo 4.1
A gramática $G_0 = (\{a,b,S\},\{a,b\},\{S \to aS, S \to bS, S \to \varepsilon\},S)$ é tal que $L_0(G_0) = \Sigma^*$.
Substituindo a regra $S \to \varepsilon$ pelas regras $S \to a$ e $S \to b$, em G, obtém-se uma nova gramática $G_1 = (\{a,b,S\},\{a,b\},\{S \to aS, S \to bS, S \to a, S \to b\},S)$, de tal forma que agora $L_1(G_1) = \Sigma^+$.
A linguagem L_2, formada por cadeias sobre $\Sigma = \{a,b\}$, de tal modo que todas elas sejam iniciadas pelo símbolo a, é gerada pela gramática $G_2 = (\{a,b,S,X\},\{a,b\},\{S \to aX, X \to aX, X \to bX, X \to \varepsilon\},S)$. São exemplos de cadeias pertencentes a L_2 : a, abb, $abaaa$, $aabbbba$, aaa etc.

Exemplo

Por outro lado, considere-se a linguagem L_3, composta por cadeias sobre $\Sigma = \{a, b\}$, de tal forma que todas elas sejam iniciadas com o símbolo a e terminadas com o símbolo b. Uma possível gramática que gera L_3 é $G_3 = (\{a, b, S, X\}, \{a, b\}, \{S \to aX, X \to aX, X \to bX, X \to b\}, S)$. Perceba a sutil diferença que existe entre G_2 e G_3.

Em seguida, considere-se a linguagem L_4 que compreende todas as cadeias sobre $\Sigma = \{a, b\}$ que possuam exatamente dois símbolos b (nem mais, nem menos). São exemplos: *bb*, *bab*, *abb*, *aaaaabaab* etc. L_4 é gerada pela gramática $G_4 = (\{a, b, S, X\}, \{a, b\}, \{S \to XbXbX, X \to aX, X \to \varepsilon\}, S)$.

Finalmente, a linguagem L_5 é definida pelas cadeias sobre $\Sigma = \{a, b\}$, de tal forma que todas elas sejam iniciadas com o símbolo b e contenham um único símbolo b. São exemplos $b, ba, baa, baaa$ etc. A linguagem L_5 é gerada por $G_5 = (\{a, b, S, X\}, \{a, b\}, \{S \to bX, X \to aX, X \to \varepsilon\}, S)$.

Esquematicamente, a relação entre as linguagens L_0, L_1, L_2, L_3, L_4 e L_5 pode ser observada na Figura 5 (é bom ter em mente que linguagens são conjuntos).

Exemplo

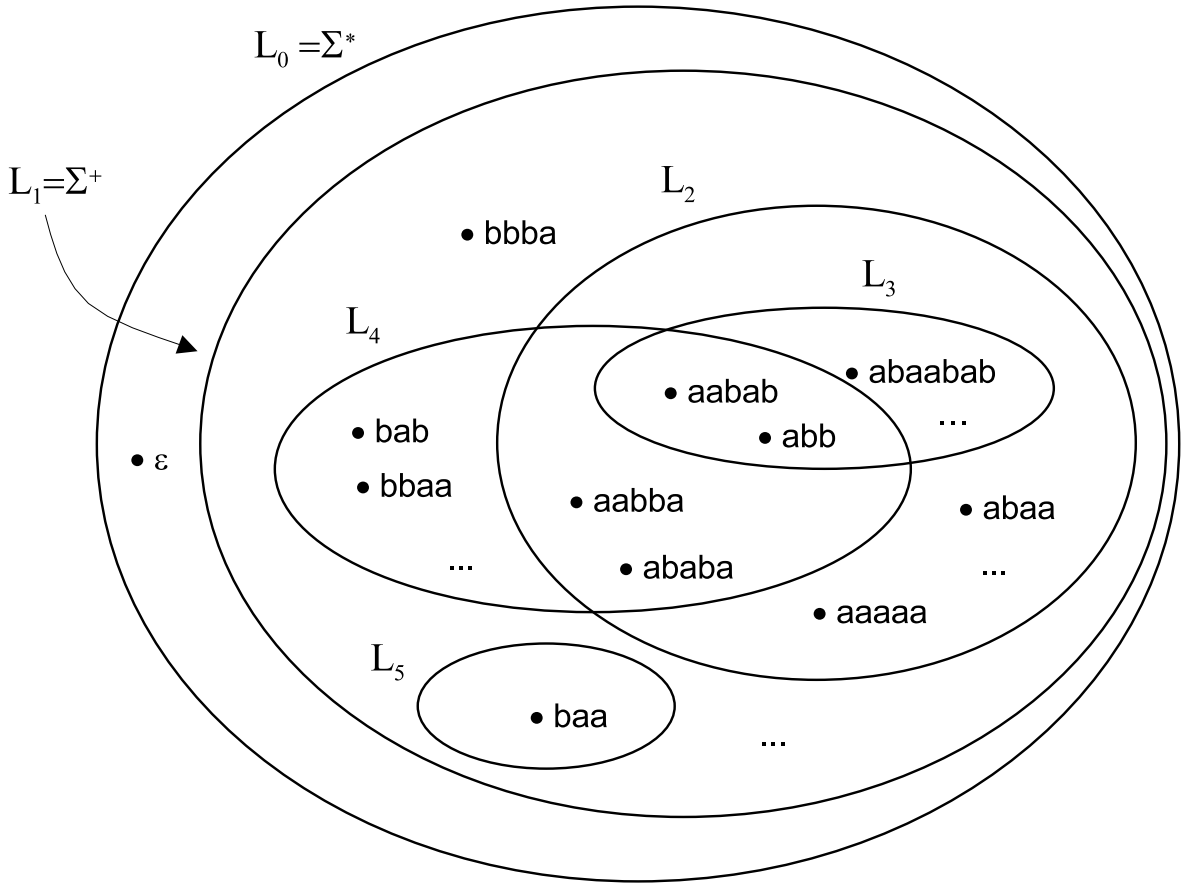


Figura 5: As linguagens do Exemplo 4.1 como conjuntos

Exemplo

Observe-se a relação de inclusão própria entre as linguagens estudadas. A cadeia ε ilustra a inclusão própria de L_1 em L. A cadeia *bbba*, de L_2 em L_1. Através da cadeia *abaa*, exemplifica-se a inclusão própria de L_3 em L_2. Sobre L_4 pode-se apenas dizer que está incluída propriamente em L_1.
A cadeia *abb* pertence simultaneamente às linguagens L_0, L_1, L_2, L_3 e L_4. A cadeia *abaabab* pertence a L_0, L_1, L_2 e L_3 apenas. A cadeia *ababa*, somente às linguagens L_0, L_1, L_2 e L_4, e assim por diante.
Cumpre, novamente, notar que:

- A **maior linguagem** que pode ser definida sobre $\{a, b\}$ é $\{a, b\}^*$, que, neste caso, corresponde a L_0;
- O conjunto de **todas as linguagens** que podem ser definidas sobre $\{a, b\}$ é dado por $2^{\{a,b\}^*}$. Assim, L_0, L_1, L_2, L_3, L_4 e L_5 pertencem, todas, a $2^{\{a,b\}^*}$. Em outras palavras, cada uma dessas linguagens é um elemento de $2^{\{a,b\}^*}$, que por sua vez é um conjunto infinito.

Conceito

Conhecidos também como **dispositivos cognitivos**, **dispositivos de aceitação**, **aceitadores sintáticos** ou simplesmente **autômatos**, os **reconhecedores** são sistemas formais capazes de aceitar todas as sentenças que pertençam a uma determinada linguagem, rejeitando todas as demais. Por esse motivo, constituem uma forma alternativa às gramáticas para a representação finita de linguagens.

Organização geral

Os reconhecedores — esboçados em sua forma geral na Figura 6 — apresentam quatro componentes fundamentais: uma memória (fita) contendo o texto de entrada do reconhecedor, um cursor, que indica o próximo elemento da fita a ser processado, uma máquina de estados finitos, sem memória, e uma memória auxiliar opcional.

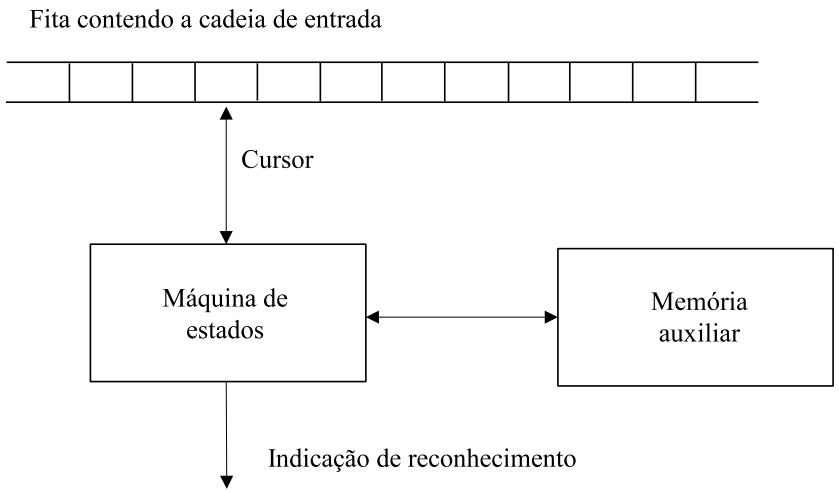


Figura 6: Organização de um reconhecedor genérico

Fita de entrada

A **fita de entrada** contém a cadeia a ser analisada pelo reconhecedor. Ela é dividida em células, e cada célula pode conter um único símbolo da cadeia de entrada, pertencente ao **alfabeto de entrada** escolhido para o reconhecedor. A cadeia de entrada é disposta da esquerda para a direita, sendo o seu primeiro símbolo colocado na posição mais à esquerda da fita.
Dependendo do tipo de reconhecedor considerado, a fita (ou o conjunto de fitas) de entrada pode apresentar comprimento finito ou infinito. Neste último caso, a fita pode ter ou não limitação à esquerda e/ou à direita. A cadeia de entrada registrada na fita de entrada pode estar delimitada por símbolos especiais, não pertencentes ao alfabeto de entrada, à sua esquerda e/ou à sua direita, porém isso não é obrigatório.

Cursor

A leitura dos símbolos gravados na fita de entrada é feita através de um cabeçote de acesso, normalmente denominado **cursor**, o qual sempre aponta o próximo símbolo da cadeia a ser processado. Os movimentos do cursor são controlados pela máquina de estados, e podem, dependendo do tipo de reconhecedor, ser unidirecionais (podendo deslocar-se para um lado apenas, tipicamente para a direita) ou bidirecionais (podendo deslocar-se para a esquerda e para a direita). Determinados tipos de reconhecedores utilizam o cursor não apenas para lerem os símbolos da fita de entrada, mas também para escreverem sobre a fita, substituindo símbolos nela presentes por outros, de acordo com comandos determinados pela máquina de estados.

Máquina de estados

A **máquina de estados** funciona como um controlador central do reconhecedor, e contém uma coleção finita de **estados**, responsáveis pelo registro de informações colhidas no passado, mas consideradas relevantes para decisões futuras, e **transições**, que promovem as mudanças de estado da máquina em sincronismo com as operações efetuadas através do cursor sobre a fita de entrada. Além disso, a máquina de estados finitos pode utilizar uma memória auxiliar para armazenar e consultar outras informações, também coletadas ao longo do processamento, que sejam eventualmente necessárias ao completo reconhecimento da cadeia de entrada.

Memória auxiliar

A **memória auxiliar** é opcional, e torna-se necessária apenas em reconhecedores de linguagens que apresentam uma certa complexidade. Normalmente, ela assume a forma de uma estrutura de dados de baixa complexidade, como, por exemplo, uma pilha (no caso do reconhecimento de linguagens livres de contexto). As informações registradas na memória auxiliar são codificadas com base em um **alfabeto de memória**, e todas as operações de manipulação da memória auxiliar (leitura e escrita) fazem referência apenas aos símbolos que compõem esse alfabeto. Os elementos dessa memória são referenciados através de um **cursor auxiliar** que, eventualmente, poderá coincidir com o próprio cursor da fita de entrada. Seu tamanho não é obrigatoriamente limitado e, por definição, seu conteúdo pode ser consultado e modificado.

Características dos componentes

O diagrama abaixo resume os componentes de um reconhecedor genérico e as diversas formas como cada um deles pode se apresentar:

- Reconhecedor
  - Máquina de Estados
    - Finita
  - Fita de entrada + Cursor
    - Limitada / Não limitada
    - Leitura apenas / Leitura e escrita
    - Direita apenas / Direita e esquerda
  - Memória auxiliar + Cursor
    - Não limitada
    - Leitura e escrita

Configuração

A operação de um reconhecedor baseia-se em uma seqüência de movimentos que o conduzem, de uma configuração inicial única, para alguma configuração de parada, indicativa do sucesso ou do fracasso da tentativa de reconhecimento da cadeia de entrada.
Cada **configuração** de um autômato é caracterizada pela quádrupla:

1. Estado;
2. Conteúdo da fita de entrada;
3. Posição do cursor;
4. Conteúdo da memória auxiliar.

Configuração inicial

A **configuração inicial** de um autômato é definida como sendo aquela em que as seguintes condições são verificadas:

1. Estado: inicial, único para cada reconhecedor;
2. Conteúdo da fita de entrada: com a cadeia completa a ser analisada;
3. Posição do cursor: apontando para o símbolo mais à esquerda da cadeia;
4. Conteúdo da memória auxiliar: inicial, predefinido e único.

Configuração final

A **configuração final** de um autômato é aquela na qual as seguintes condições são obedecidas:

1. Estado: algum dos estados finais, que não são necessariamente únicos no reconhecedor;
2. Conteúdo da fita de entrada: inalterado ou alterado, em relação à configuração inicial, conforme o tipo de reconhecedor;
3. Posição do cursor: apontando para a direita do último símbolo da cadeia de entrada ou apontando para qualquer posição da fita, conforme o tipo de reconhecedor;
4. Conteúdo da memória auxiliar: final e predefinido, não necessariamente único ou idêntico ao da configuração inicial, ou apenas indefinido.

Movimentação

A especificação de uma possibilidade de **movimentação** entre uma configuração e outra é denominada **transição**. A movimentação do autômato da configuração corrente para uma configuração seguinte é feita, portanto, levando-se em conta todas as transições passíveis de serem aplicadas pelo reconhecedor à configuração corrente.

Movimentação

Uma transição mapeia triplas formadas por:

- Estado corrente;
- Símbolo correntemente apontado pelo cursor da fita de entrada;
- Símbolo correntemente apontado pelo cursor da memória auxiliar;

em triplas formadas por:

- Próximo estado;
- Símbolo que substituirá o símbolo correntemente apontado pelo cursor da fita de entrada e o sentido do deslocamento do cursor;
- Símbolo que substituirá o símbolo correntemente apontado pelo cursor da memória auxiliar.

Linguagem

Diz-se que um autômato **aceita** (ou reconhece) uma cadeia se lhe for possível atingir alguma configuração final a partir de sua configuração inicial única, através de movimentos executados sobre tal cadeia. Caso contrário, diz-se que o autômato **rejeita** a cadeia. A maneira como tais configurações sucedem umas às outras durante o reconhecimento (ou aceitação) da cadeia de entrada define uma característica fundamental dos autômatos, conforme explicado a seguir.

Seja o autômato determinístico ou não-determinístico, a linguagem por ele aceita (ou definida) corresponde ao conjunto de todas as cadeias que ele aceita.