Linguagens

O texto apresentado neste documento é uma adaptação (tradução) de:

Thomas A. Sudkamp. 1997. Languages and machines: an introduction to the theory of computer science. Addison-Wesley Longman Publishing Co., Inc., USA.

O conceito de linguagem inclui uma variedade de categorias aparentemente distintas: linguagens naturais, linguagens de computador e linguagens matemáticas. Uma definição geral de linguagem deve abranger esses vários tipos de linguagens. Vamos adotar a seguinte definição: Uma linguagem é um conjunto de palavras (strings) sobre um alfabeto. Esta é a definição mais ampla possível, não há restrições inerentes à forma das strings que constituem uma linguagem.

As linguagens de interesse não são constituídas por strings arbitrárias, mas sim por strings que satisfazem certas propriedades. Essas propriedades definem a sintaxe da linguagem. Vamos mostrar como definições recursivas e operações de conjunto podem ser usadas para impor restrições sintáticas nas strings de uma linguagem.

Vamos ainda apresentar uma família de linguagens conhecidas como **conjuntos regulares**. Um conjunto regular é construído recursivamente a partir do conjunto vazio e de conjuntos unitários. Embora introduzamos os conjuntos regulares por meio de operações sobre conjuntos, à medida que progredirmos veremos que eles surgem naturalmente como as linguagens geradas por gramáticas regulares e reconhecidas por máquinas de estados finitos.

1. Strings e linguagens

Uma string sobre um conjunto X é uma sequência finita de elementos de X. Strings são os elementos fundamentais usados na definição de linguagens. O conjunto de elementos a partir do qual as strings são construídas é chamado de alfabeto da linguagem. Um alfabeto consiste em um conjunto finito de objetos indivisíveis. O alfabeto de uma linguagem é denotado por Σ.

Como os elementos do alfabeto de uma linguagem são indivisíveis, eles geralmente são denotados por caracteres simples. As letras a, b, c, d, e, com ou sem subscritos, são usadas para representar os elementos de um alfabeto. Strings sobre um alfabeto são representadas por letras que ocorrem perto do final do alfabeto. Em particular, p, q, u, v, w, x, y, z são usadas para denotar strings.

Uma definição formal para o conjunto de strings sobre um alfabeto é apresentado a seguir, usando definição recursiva. A base consiste da string que não contém elementos. Essa string é chamada de string nula e denotada como λ. O operador primitivo usado na definição consiste em juntar um único elemento do alfabeto ao lado direito de uma string existente.

Definição 1.1

Seja Σ um alfabeto. Σ*, o conjunto de strings sobre Σ, é definido recursivamente da seguinte forma:

i) Base: $\lambda \in \Sigma^*$.

ii) Passo recursivo: Se $w \in \Sigma^*$ e $a \in \Sigma$, então $wa \in \Sigma^*$.

Para qualquer alfabeto não vazio Σ, Σ* contém infinitos elementos. Se Σ = {*a*), Σ* contém as strings λ, *a*, *aa*, *aaa*,. . . O número de símbolos de uma string *w* é denotado *comprimento(w)*. Se Σ contiver *n* elementos, haverá n^k strings de comprimento *k* em Σ*.

Exemplo 1.1

Seja Σ={*a, b, c*}. Os elementos de Σ* incluem

Comprimento 0:	λ
Comprimento 1:	*a b c*
Comprimento 2:	*aa ab ac babb bc ca cb cc*
Comprimento 3:	*aaa aab aac aba abb abc aca acb acc*
	baa bab bac bba bbb bbc bca bcb bcc
	caa cab cac cba cbb cbc cca ccb ccc

Uma linguagem consiste em strings sobre um alfabeto. Normalmente, algumas restrições são estabelecidas para as strings de uma linguagem. Consequentemente, uma linguagem consiste em um subconjunto do conjunto de todas as possíveis strings do alfabeto. ❑

Definição 1.2

Uma linguagem sobre um alfabeto Σ é um subconjunto de Σ*.

Definição 1.3

Concatenação é a operação binária de usar duas strings e "colá-las" para construir uma nova string. A concatenação é a operação fundamental na geração de strings. Uma definição formal é dada por recursão no comprimento da segunda string na concatenação. Neste ponto, a operação primitiva de juntar um único caractere ao lado direito de uma string é a única operação em strings que foi introduzida. Portanto, qualquer nova operação deve ser definida em termos dessa operação primitiva.

Seja $u, v \in \Sigma^*$. A **concatenação** de *u* e *v*, escrita como *uv*, é uma operação binária em Σ* definida da seguinte forma:

i) Base: Se comprimento (*v*) = 0, então *v* = λ e *uv* = *u*.

ii) Passo recursivo: Seja *v* uma string com *comprimento (v) = n* > 0. Então *v = wa*, para alguma string *w* com comprimento *n* - 1 e $a \in \Sigma$, e *uv = (uw)a*.

Exemplo 1.2

Seja $u=ab$, $v=ca$ e $w=bb$. Então

$uv = abca$ $\quad\quad$ $vw = cabb$
$(uv)\,w = abcabb$ $\quad\quad$ $u(vw) = abcabb$. ❑

O resultado da concatenação de u, v e w é independente da ordem em que as operações são realizadas, indicando associatividade. O Teorema 1.1 prova que a concatenação é realmente uma operação binária associativa.

Teorema 1.1

Sejam $u, v, w \in \Sigma^*$. Então $(uv)\,w = u\,(vw)$.

Prova: A prova é por indução no comprimento da string w.

Base: *comprimento (w)* = 0. Então w = λ., e $(uv)\,w = uv$ pela definição de concatenação. Por outro lado, $u\,(vw) = u\,(v) = uv$.

Hipótese indutiva: assuma que $(uv)\,w = u\,(vw)$ para todas as strings w de comprimento n ou menos.

Passo Indutivo: Precisamos provar que $(uv)\,w = u\,(vw)$ para todas as strings w de comprimento $n + 1$.

Seja w uma string com essa propriedade. Então $w = xa$ para alguma string x de comprimento n e $a \in \Sigma$ e

(uv) w	= (uv) (xa)	
	= ((uv) x) a	(definição de concatenação)
	= (u (vx)) a	(hipótese indutiva)
	= u ((vx) a)	(definição de concatenação)
	= u (v (xa))	(definição de concatenação)
	= u (vw)	(substituição, $xa = w$) ❑

Uma vez que a associatividade garante o mesmo resultado, independentemente da ordem das operações, os parênteses são omitidos de uma sequência de aplicações de concatenação. Expoentes são usados para abreviar a concatenação de uma string consigo mesma. Assim, uu pode ser escrito u^2, uuu pode ser escrito u^3, etc. u^0, que representa a concatenação de u consigo mesmo zero vezes, é definido como a string nula. A operação de concatenação não é comutativa. Para strings $u = ab$ e $v = ba$, $uv = abba$ e $vu = baab$. Observe que $u^2 = abab$ e não $aabb = a^2\,b^2$.

Substrings podem ser definidas usando a operação de concatenação. Intuitivamente, u é uma substring de v se u "ocorre dentro de" v. Formalmente, u é uma substring de v se houver strings x e y tais que $v = xuy$. Um **prefixo** de v é uma substring u na qual x é a string nula na decomposição de v. Ou seja, $v = uy$. Da mesma forma, u é um sufixo de v se $v = xu$.

O **reverso** de uma string é a string escrita ao contrário. O reverso de u é representado por u^R.

2. Especificando linguagens com recursividade

Definimos *linguagem* como um conjunto de strings sobre um alfabeto. As linguagens de interesse não consistem em conjuntos arbitrários de strings, mas sim em strings com formas específicas. As formas que são aceitas definem a sintaxe da linguagem.

A especificação de uma linguagem requer uma descrição não ambígua de suas strings. Uma linguagem finita pode ser definida explicitamente enumerando seus elementos. Algumas linguagens infinitas com requisitos sintáticos simples são definidas recursivamente nos exemplos a seguir.

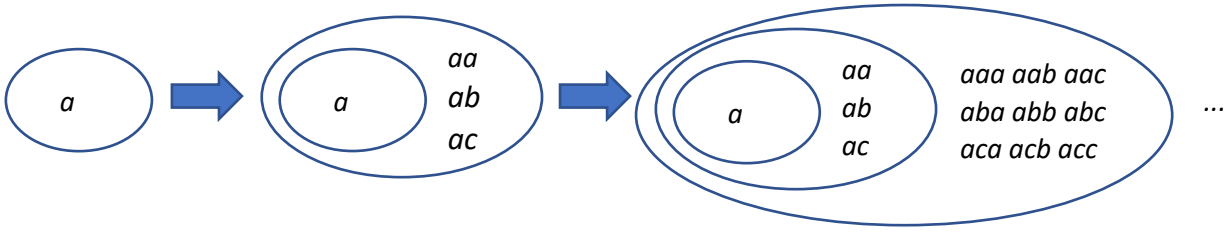
Exemplo 2.1

A linguagem L de strings sobre $\{a, b, c\}$ em que cada string começa com um a:

i) Base: $a \in L$.

ii) Passo recursivo: Se $u \in L$, então ua, ub, $uc \in L$.

As strings em L são construídas juntando um símbolo no lado direito de uma string previamente construída. A base garante que cada string em L comece com um a.



Os diagramas acima mostram como as strings são construídas recursivamente. A base define a como sendo uma primeira string. No próximo passo, são aa, ab, ac construídas concatenando a, b, c no final de a. No passo seguinte, mais 9 strings são definidas, e assim por diante. É possível reconhecer um padrão na construção, determinando que todas as strings comecem com a. ❑

Exemplo 2.2

A linguagem L de strings sobre $\{a, b, c\}$ em que cada string termina com um b é definida por:

i) Base: $b \in L$.

ii) Passo recursivo: Se $w \in L$, então aw, bw, $cw \in L$.

Neste caso, as strings em L são construídas concatenando símbolos no início de string previamente contruída. Esse processo garante que todas as palavras terminem com b, considerando que a base tem essa propriedade. ❑

Exemplo 2.3

A linguagem L de strings sobre $\{a, b\}$ em que cada string começa com um a e tem comprimento par é definida por:

i) Base: aa, $ab \in L$.

ii) Passo recursivo: Se $u \in L$, então uaa, uab, uba, $ubb \in L$.

As strings em L são construídas juntando dois elementos no lado direito de uma string previamente construída. A base garante que cada string em L comece com um a. Adicionando sub-strings de comprimento dois, a paridade par é mantida. ❑

Exemplo 2.4

A linguagem L consiste em strings sobre $\{a, b\}$ nas quais cada ocorrência de b é imediatamente precedida por um a. Por exemplo, λ, a, $abaab$ estão em L e bb, bab, abb não estão em L.

i) Base: $\lambda \in L$.

ii) Passo recursivo: Se $u \in L$, então ua, $uab \in L$. ❑

Definições recursivas fornecem uma ferramenta para definir as strings de uma linguagem. Os exemplos 2.1 a 2.4 mostraram que os requisitos de ordem, posicionamento e paridade podem ser obtidos usando uma geração recursiva de strings. O processo de geração de string em uma definição recursiva, no entanto, é inadequado para impor os requisitos sintáticos de linguagens complexas, como linguagens matemáticas ou de computador.