Aula 00 – Linguagens

Prof. Eduardo Zambon

Departamento de Informática (DI)
Centro Tecnológico (CT)
Universidade Federal do Espírito Santo (Ufes)

Algoritmos e Fundamentos da Teoria de Computação (TocE)
Engenharia de Computação

Introdução

- Conceito de linguagem abrange várias categorias como:
  - linguagens naturais,
  - linguagens de programação,
  - linguagens matemáticas,
  - etc...
- **Estes *slides***: revisão/introdução de conceitos básicos da teoria de linguagens formais.
- **Objetivos**: fixar termos, definições e notações para o restante do curso.

Referências

Chapter 2 – Languages
T. Sudkamp
Section 0.2 – Mathematical Notions and Terminology
M. Sipser
Section 1.2 – Mathematical preliminaries
A. Maheshwari

Introdução

- Definição geral de linguagem deve cobrir os diferentes tipos.
- Aqui: definição de linguagem usando teoria de conjuntos.
- Uma linguagem é um conjunto de strings sobre um alfabeto.
- Alfabeto: conjunto finito de símbolos da linguagem.
- String sobre um alfabeto: sequência finita de símbolos do alfabeto.
- Linguagens de interesse não são arbitrárias, possuem regras de sintaxe.
- Usa-se definições recursivas e operações de conjuntos para garantir restrições sintáticas sobre as strings.

Strings e Linguagens

- Para descrever uma linguagem é necessário identificar o seu alfabeto.
- Única restrição do alfabeto: conjunto finito.
- Elementos do alfabeto são indivisíveis.
- Em linguagem natural (LN): cada palavra é um elemento do alfabeto.
- $\Rightarrow$ Um dicionário é um alfabeto (segundo os conceitos acima).
- Cuidado: o abecedário (contendo as 26 letras latinas) não corresponde ao conceito de alfabeto usado neste slide para LN!

Strings e Linguagens

Notação:

- Σ é o alfabeto.
- a, b, c, d, e são elementos do alfabeto Σ.
- u, v, w, x, y, z são strings.
- λ é a **string nula**, i.e., a string sem elementos.

O conjunto de strings sobre Σ é definido recursivamente.

Definição 2.1.1 (Sudkamp) – Conjunto de strings sobre Σ

Seja Σ um alfabeto. O conjunto de strings sobre Σ, denotado por Σ^*, é definido recursivamente como a seguir.

1. **Base:** $\lambda \in \Sigma^*$.
2. **Passo recursivo:** se $w \in \Sigma^*$ e $a \in \Sigma$, então $wa \in \Sigma^*$.
3. **Fecho:** $w \in \Sigma^*$ somente se w pode ser obtida a partir de λ por um número finito de aplicações do passo 2.

Strings e Linguagens

- Para qualquer alfabeto não vazio Σ, Σ^* contém infinitos elementos.
- Exemplo: se $\Sigma = \{a\}$, então $\Sigma^* = \{\lambda, a, aa, aaa, \ldots\}$.
- $length(w)$: número de elementos de uma string.
- Se $card(\Sigma) = n$, então existem n^k strings de comprimento k em Σ^*.

Exemplo 2.1.1 (Sudkamp)

Seja $\Sigma = \{a, b, c\}$. Os elementos de Σ^* incluem:

- **Length 0**: λ
- **Length 1**: $a\ b\ c$
- **Length 2**: $aa\ ab\ ac\ ba\ bb\ bc\ ca\ cb\ cc$
- **Length 3**: $\ldots$

Strings e Linguagens

- Uma linguagem é um conjunto de strings sobre um alfabeto.
- Σ^* é o conjunto (infinito) de todas as strings sobre um alfabeto Σ.
- Σ^* é uma linguagem?
- Exemplo:
  - Tome Σ como todas as palavras do português.
  - Σ^* descreve todas as frases da língua portuguesa?
  - Note que a frase “Hoje casa carro talvez.” está em Σ^*.
  - Uma linguagem restringe os elementos de Σ^* pelas regras de sintaxe.

Definição 2.1.2 (Sudkamp) – Linguagem

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

Strings e Linguagens

Concatenação – operação fundamental na geração de strings.

Definição 2.1.3 (Sudkamp) – Concatenação

Sejam $u, v \in \Sigma^*$. A concatenação de u e v, denotada por uv, é uma operação binária sobre Σ^* definida como abaixo.

1. **Base:** se $length(v) = 0$, então $v = \lambda$ e $uv = u$.
2. **Passo recursivo:** seja v uma string com $length(v) = n > 0$. Então $v = wa$, para alguma string w com comprimento $n - 1$ e $a \in \Sigma$, e assim $uv = (uw)a$.

Exemplo 2.1.2 (Sudkamp)

Seja $u = ab$, $v = ca$, e $w = bb$. Então

$$\begin{aligned} uv &= abca & vw &= cabb \\ (uv)w &= abcabb & u(vw) &= abcabb. \end{aligned}$$

- O teorema abaixo mostra que concatenação é uma operação binária associativa.
- Isto é, o resultado da concatenação de u, v e w é independente da ordem das operações.

Teorema 2.1.4 (Sudkamp) – Associatividade da concatenação

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

A prova do teorema acima é feita por indução no comprimento da string w. (Veja o livro.)

Strings e Linguagens

- Concatenação é associativa $\Rightarrow$ omissão de parênteses.
- Abreviação da operação de concatenação por meio de expoentes:
  - $u^2 = uu$
  - $u^n = \underbrace{uuu \dots u}_{n}$
  - $u^0 = \lambda$
- Concatenação não é comutativa.
- Isto é, para strings $u = ab$ e $v = ba$, $uv = abba$ e $vu = baab$.
- Note que $u^2 = abab$ e não $aabb = a^2 b^2$.

- Diz-se que u é uma substring de v se existem strings x e y tal que $v = xuy$.
- Informalmente, u é uma substring de v se u "aparece dentro de" v.
- Note que tanto x quanto y podem ser λ.
- $x = \lambda \Rightarrow u$ é um prefixo de $v = uy$.
- $y = \lambda \Rightarrow u$ é um sufixo de $v = xu$.

- O reverso de uma string é a string escrita ao contrário.
- Exemplo: o reverso de *abbc* é *cbba*.

Definição 2.1.5 (Sudkamp) – Reverso de uma string

Seja $u \in \Sigma^*$. O reverso de u, denotado por u^R, é definido como abaixo.

1. **Base:** se $length(u) = 0$, então $u = \lambda$ e $\lambda^R = \lambda$.
2. **Passo recursivo:** se $length(u) = n > 0$, então $u = wa$, para alguma string w com comprimento $n - 1$ e $a \in \Sigma$, e assim $u^R = aw^R$.

Teorema 2.1.6 (Sudkamp)

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

Especificação Finita de Linguagens

- A especificação (definição) de uma linguagem requer uma descrição inequívoca das strings que a compõem.
- Linguagem finita: definida pela enumeração dos seus elementos.
- Linguagem infinita?
- Se os requisitos sintáticos forem simples $\Rightarrow$ definição recursiva.

Especificação Finita de Linguagens

Exemplo 2.2.1 (Sudkamp)

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

1. **Base:** $aa, ab \in$ L.
2. **Passo recursivo:** se $u \in$ L, então $uaa, uab, uba, ubb \in$ L.
3. **Fecho:** uma string $u \in$ L somente se u pode ser obtida a partir dos elementos da base por um número finito de aplicações do passo 2.

Especificação Finita de Linguagens

Exemplo 2.2.2 (Sudkamp)

A linguagem L sobre o alfabeto $\{a, b\}$ definida por:

1. **Base:** $\lambda \in$ L.
2. **Passo recursivo:** se $u \in$ L, então $ua, uab \in$ L.
3. **Fecho.**

é composta pelas strings nas quais cada ocorrência de b é imediatamente precedida por um a.

Exemplos:

- $\lambda, a, abaab \in$ L.
- $bb, bab, abb \notin$ L.

Especificação Finita de Linguagens

Exemplo 2.2.3 (Sudkamp)

Seja L a linguagem sobre o alfabeto $\{a, b\}$ definida por:

1. **Base:** $\lambda \in$ L.
2. **Passo recursivo:** se $u \in$ L e u pode ser escrita como $u = xyz$, então $xaybz, xbyaz \in$ L.

A linguagem L é composta por todas as strings com o mesmo número de a's e b's.

Interpretação intuitiva do passo recursivo: "insira um a e um b em qualquer posição da string u."

Especificação Finita de Linguagens

- Definições recursivas são um meio de definir as strings de uma linguagem.
- No entanto, gerar strings usando uma única definição recursiva não permite descrever os requisitos sintáticos complexos de LNs e LPs.
- Outra técnica para construir linguagens: usar operações de conjuntos para construir conjuntos complexos de strings a partir de outros mais simples.
- Operações definidas para strings podem ser estendidas para conjuntos (e portanto para linguagens).
- $\Rightarrow$ Descrições de linguagens infinitas podem ser construídas a partir de conjuntos finitos usando operações de conjuntos.

Especificação Finita de Linguagens

Definição 2.2.1 (Sudkamp) – Concatenação de Linguagens

A concatenação das linguagens X e Y, denotada por XY, é a linguagem

$$XY = \{uv \mid u \in X \text{ e } v \in Y\}.$$

A concatenação de X com ele mesmo n vezes é denotada por X^n. X^0 é definido como $\{\lambda\}$.

Exemplo 2.2.4 (Sudkamp)

Sejam $X = \{a, b, c\}$ e $Y = \{abb, ba\}$. Então

$$\begin{aligned} XY &= \{aabb, babb, cabb, aba, bba, cba\} \\ X^0 &= \{\lambda\} \\ X^1 &= X = \{a, b, c\} \\ X^2 &= XX = \{aa, ab, ac, ba, bb, bc, ca, cb, cc\} \end{aligned}$$

Especificação Finita de Linguagens

- Os conjuntos do exemplo anterior já foram vistos.
- Para cada i, X^i contém as strings de comprimento i em Σ^* apresentadas no Exemplo 2.1.1.
- Nova operação de conjunto: fecho de Kleene *(Kleene star)* de um conjunto X, denotada por X^*.
- As strings sobre um conjunto podem ser definidas através do operador * e as operações de concatenação e união de conjuntos.
- Com isso a Definição 2.1.1 pode ser substituída.

Especificação Finita de Linguagens

Definição 2.2.2 (Sudkamp)

Seja X um conjunto. Então

$$X^* = \bigcup_{i=0}^{\infty} X^i \qquad X^+ = \bigcup_{i=1}^{\infty} X^i \quad \text{ou} \quad X^+ = XX^*.$$

- O conjunto X^* contém todas as strings que podem ser construídas com os elementos de X.
- Se X é um alfabeto, X^+ é o conjunto de todas as strings não-nulas sobre X.

Especificação Finita de Linguagens

- Definição de uma linguagem formal requer uma especificação inequívoca.
- $\Rightarrow$ Uma definição usando LN não serve.
- Operações de conjuntos não causam ambiguidade.
- $\Rightarrow$ Operações de conjuntos servem para descrever as strings que formam uma linguagem.

Exemplo 2.2.5 (Sudkamp)

A linguagem $L = \{a, b\}^*\{bb\}\{a, b\}^*$ consiste das strings sobre $\{a, b\}$ que contém a substring bb.

Especificação Finita de Linguagens

Exemplo 2.2.6 (Sudkamp)

- Seja L a linguagem formada por todas as strings que começam com *aa* ou terminam com *bb*.
- O conjunto $\{aa\}\{a, b\}^*$ descreve as strings com prefixo *aa*.
- O conjunto $\{a, b\}^*\{bb\}$ descreve as strings com sufixo *bb*.
- Portanto, $L = \{aa\}\{a, b\}^* \cup \{a, b\}^*\{bb\}$.

Exemplo 2.2.7 (Sudkamp)

- Sejam $L_1 = \{bb\}$ e $L_2 = \{\lambda, bb, bbbb\}$ linguagens sobre $\{b\}$.
- As linguagens L_1^* e L_2^* contém exatamente as strings formadas por um número par de *b*'s.

Especificação Finita de Linguagens

Exemplo 2.2.8 (Sudkamp)

- O conjunto $\{aa, bb, ab, ba\}^*$ consiste de todas as strings de comprimento par sobre $\{a, b\}$.
- O conjunto das strings de comprimento ímpar é definido como $\{a, b\}^* - \{aa, bb, ab, ba\}^*$.
- Outra definição possível: $\{aa, bb, ab, ba\}^*\{a, b\}$.

Conjuntos Regulares e Expressões

- Exemplos anteriores: uso de operações de conjuntos para construir novas linguagens a partir de outras já existentes.
- Não havia nenhuma restrição sobre os conjuntos e operações permitidas na construção.
- Vamos introduzir algumas restrições para definir os conjuntos regulares.
- Os conjuntos regulares formam uma família de linguagens muito importante para as áreas de linguagens formais, reconhecimento de padrões e computabilidade.

Definição 2.3.1 (Sudkamp)

Seja Σ um alfabeto. Os conjuntos regulares sobre Σ são definidos recursivamente como a seguir.

1. **Base:** os conjuntos $\emptyset$, $\{\lambda\}$ e $\{a\}$, para todo $a \in \Sigma$, são conjuntos regulares sobre Σ.
2. **Passo recursivo:** sejam X e Y conjuntos regulares sobre Σ. Os conjuntos

$$X \cup Y \qquad XY \qquad X^*$$

são conjuntos regulares sobre Σ.

Uma linguagem é dita regular se ela for descrita por um conjunto regular.

Conjuntos Regulares e Expressões

Exemplo 2.3.1 (Sudkamp)

A linguagem do Exemplo 2.2.5 é regular pois satisfaz as restrições da Definição 2.3.1.

Exemplo 2.3.2 (Sudkamp)

O conjunto de strings que começam e terminam com um a e contém ao menos um b é regular sobre $\{a, b\}$. Formalmente:

$$\mathsf{L} = \{a\}\{a, b\}^*\{b\}\{a, b\}^*\{a\}.$$

Exemplo acima é desnecessariamente complicado de escrever. Usamos expressões regulares (ER) para abreviar a descrição de conjuntos regulares.

Notação de ERs:

- $\emptyset, \lambda,$ **a**: representam os conjuntos regulares $\emptyset$, $\{\lambda\}$ e $\{a\}$.
- Operações de união, concatenação e *Kleene star* continuam as mesmas.

Definição 2.3.2 (Sudkamp)

Seja Σ um alfabeto. As expressões regulares sobre Σ são definidas recursivamente como a seguir.

1. **Base:** $\emptyset$, λ e **a**, para $a \in \Sigma$, são expressões regulares.
2. **Passo recursivo:** sejam **u** e **v** expressões regulares sobre Σ. As expressões

$$(\mathbf{u} \cup \mathbf{v}) \qquad (\mathbf{uv}) \qquad (\mathbf{u}^*)$$

são expressões regulares sobre Σ.

Obs.: Outros livros denotam a união $(\mathbf{u} \cup \mathbf{v})$ como $(\mathbf{u}|\mathbf{v})$.

Conjuntos Regulares e Expressões

- Parênteses podem ser eliminados quando não há ambiguidade na expressão.
- Qual a diferença entre conjuntos regulares e expressões regulares?
- Uma ER define um padrão e uma string está na linguagem (i.e., pertence ao conjunto regular que define a linguagem) somente se a string se "encaixa" no padrão.

Exemplo 2.3.4 (Sudkamp)

ER que representa o conjunto de strings sobre $\{a, b\}$ que contém ou a substring *aa* ou a substring *bb*:

$$(\mathbf{a} \cup \mathbf{b})^*\mathbf{aa}(\mathbf{a} \cup \mathbf{b})^* \cup (\mathbf{a} \cup \mathbf{b})^*\mathbf{bb}(\mathbf{a} \cup \mathbf{b})^*.$$

Exemplo 2.3.7 (Sudkamp)

ER que representa o conjunto de strings sobre $\{a, b\}$ que contém um número par de *b*'s:

$$\mathbf{a}^*(\mathbf{a}^*\mathbf{ba}^*\mathbf{ba}^*)^*.$$

Outra ER que representa o mesmo conjunto: $\mathbf{a}^*(\mathbf{ba}^*\mathbf{ba}^*)^*$.

- Exemplo anterior mostra que a definição de uma linguagem por uma ER não é única.
- Duas ERs que representam o mesmo conjunto são ditas equivalentes.

TABLE 2.3.1 Regular Expression Identities

1.	$\emptyset u = u\emptyset = \emptyset$
2.	$\lambda u = u\lambda = u$
3.	$\emptyset^* = \lambda$
4.	$\lambda^* = \lambda$
5.	$u \cup v = v \cup u$
6.	$u \cup \emptyset = u$
7.	$u \cup u = u$
8.	$u^* = (u^*)^*$
9.	$u(v \cup w) = uv \cup uw$
10.	$(u \cup v)w = uw \cup vw$
11.	$(uv)^*u = u(vu)^*$
12.	$(u \cup v)^* = (u^* \cup v)^*$ $= u^*(u \cup v)^* = (u \cup vu^*)^*$ $= (u^*v^*)^* = u^*(vu^*)^*$ $= (u^*v)^*u^*$

Conjuntos Regulares e Expressões

- A construção de uma ER é um processo positivo.
- Características da string desejada são incluídas explicitamente pelas operações.
- Não há uma operação negativa para omitir strings com uma certa característica.
- $\Rightarrow$ ER extendidas. (Fora do escopo da disciplina.)

Exemplo 2.3.9 (Sudkamp)

ER que representa o conjunto de strings sobre $\{a, b\}$ que não terminam em *aaa*:

$$(\mathbf{a} \cup \mathbf{b})^*(\mathbf{b} \cup \mathbf{ba} \cup \mathbf{baa}) \cup \lambda \cup \mathbf{a} \cup \mathbf{aa}.$$

Conjuntos Regulares e Expressões

- Ausência de negação é inconveniente na prática (onde geralmente Σ é grande).
- Ferramentas que usam ER (e.g., `grep`) estendem as operações para incluir negação, *ranges*, etc.
- ERs permitem descrever muitos padrões complexos, mas...
- ...há linguagens que não podem ser definidas por nenhuma ER.
- *Exemplo*: não há uma ER que defina a linguagem $\{a^i b^i \mid i \geq 0\}$.

Aula 00 – Linguagens

Prof. Eduardo Zambon

Departamento de Informática (DI)
Centro Tecnológico (CT)
Universidade Federal do Espírito Santo (Ufes)

Algoritmos e Fundamentos da Teoria de Computação (TocE)
Engenharia de Computação