Kapitel 1

Alphabete, Wörter, Sprachen

Vorlesung: 25.04.2003

Zeichenreihen als Grundobjekte der Informatik

- Präzisierung des Algorithmusbegriffs durch TM
- Kommunikation mit rechner über Tastatur
- Informationsverarbeitung: Transformation von Bitstrings

1. Grundbegriff:

Σ Alphabet, nicht leere endliche Menge
a $\in \Sigma$ Buchstabe, Zeichen, Character

2. Grundbegriff

: Σ^* Menge der Wörter über Σ.
$\Sigma^* := \{a_1, a_2, \ldots, a_n | n \in \mathbb{N}\}$ Wörter, Zeichen, Strings
$n = 0$: das leere Wort, Bezeichnung: ε

Bsp:
Bitstrings, Webadresse, Dezimalzahl, RGB-Farbwerte, Java-Programme

Operationen auf Σ^*:

- **Verkettung (Konkatenation)**
  $\_ \cdot \_ : \quad \Sigma^* \cdot \Sigma^* \to \Sigma^*$
  $w \cdot v := wv$

  (i) $\cdot$ ist assoziativ: $(u \cdot v) \cdot w = u \cdot (v \cdot w)$
  (ii) ε ist $\cdot$ neutral: $\varepsilon \cdot w = w \cdot \varepsilon$

Sprechweise: $< \Sigma^*, \_ \cdot \_, \varepsilon >$ ist ein Monoid.
Bemerkung:
freie Erzeugung: $a_1 \dots a_n = b_1 \dots b_n$
$\curvearrowright n = m, a_i = b_i (i = 1 \dots n)$

- **Länge eines Wortes**
  $w = a_1 \dots a_n \in \Sigma^*$
  $|a_1 \dots a_n| := u$, falls alle $a_i \in \Sigma^*$,
  also $|\varepsilon| = 0$ und $|v \cdot w| = |v| + |w|$

- **Potenz eines Wertes**
  $w \in \Sigma^*$
  $w^0 := \varepsilon$
  $w^{n+1} := w^n \cdot w$

- **Spiegelbild eines Wortes**
  $\varepsilon^R := \varepsilon$
  $(wa]^R := aw^R$

3. Grundbegriff

$\mathrm{P}(\Sigma^*) := \{L \mid L \subseteq \Sigma^*\}$ Menge der formalen Sprachen über Σ

Bsp:
$\emptyset, \{\varepsilon\}, \{w_1, \dots, w_n\}, \Sigma^*$, Menge der Java-Programme, Menge der URLs, Menge der HTML-Beschreibungen

Operationen auf $\mathbf{P}(\Sigma^*)$

- **boolesche Operationen**:
  $L_1 \cup L_2, L_1 \cap L_2, \overline{L} := \Sigma^* \backslash$

- **Komplexmodul**:
  $L_1 L_2 := \{w_1 w_2 \mid w_i \in L_i\}$
  $L_1 L_2 := \{w_1 w_1 \mid w_1 \in L_1\}$
  besser:
  $L_1 L_2 := \{wv \mid w \in L_1, v \in L_2\}$
  $L_1 L_2 := \{wv \mid w, v \in L_1\}$

- **Potenz einer Sprache**:
  $L^0 := \{\varepsilon\}$
  $L^{n+1} := L^n L$

- **Stern einer Sprache**: (Iteration, Repetition)
  $L^* := \bigcup_{n\ N} L^n \quad \curvearrowright \emptyset^* = \{\varepsilon\}$
- **Spiegelbild einer Sprache**:
  $L^R := \{w^R \mid w \in L\}$
- **Reguläre Operationen**:
  $L_1 \cup L_2, L_1 L_2, L^*$

Kapitel 2

Reguläre Ausdrücke und endliche Automaten

Vorlesung: 29.04.2003

2.1 Reguläre Ausdrücke

Ein regulärer Ausdruck beschreibt eine formale Sprache [Mengen von Zeichenreihen], die sich mit Hilfe regulärer Operationen (Vereinigung, Komplexprodukt, Stern) aus einfachen Sprachen erzeugen lässt.

Definition - Syntax von RegE(Σ)

Sei Σ ein Alphabet. Die Menge RegE(Σ) der *regulären Ausdrücke über* Σ ist induktiv definiert durch

(i) $\Lambda \in \text{RegE}(\Sigma)$
(ii) $a \in \text{RegE}(\Sigma)$ für jedes $a \in \Sigma$

Wenn α und $\beta \in \text{RegE}(\Sigma)$, so auch

(iii) $(\alpha \vee \beta) \in \text{RegE}(\Sigma)$ „Alternative“
(iv) $(\alpha \cdot \beta) \in \text{RegE}(\Sigma)$ „Konkatenation“
(v) $(\alpha^*) \in \text{RegE}(\Sigma)$ „Repetition“

Vereinfachte Schreibweise:

- Präzendenzregel, um Klammern zu sparen:
  $*$ bindet stärker als $\cdot$
  $\cdot$ bindet stärker als $\vee$
- der Punkt „$\cdot$“ wird weggelassen.

Bsp:

$a \vee b^*c$ statt $(a \vee ((b^*) \cdot c))$

Definition - Semantik von RegE(Σ)

Ein regulärer Ausdruck α beschreibt eine formale Sprache $||[\alpha]|| = L(\alpha) \subseteq \Sigma^*$.

1. $L(\Lambda) := \emptyset$
2. $L(a) := \{a\}$
3. $L(\alpha \vee \beta) := L(\alpha) \cup L(\beta)$
4. $L(\alpha \cdot \beta) := L(\alpha) \cdot L(\beta)$
5. $L(\alpha^*) := L(\alpha)^*$

Sprechweise:
$w \in L(\alpha) \quad \curvearrowright w$ ist ein Match für α, α ist ein Muster (Pattern).

Definition

Die Klasse RegL(Σ) der regulären Sprachen über Σ ist induktiv definert durch

- $\emptyset, \{a\} \in$ RegL für alle $a \in \Sigma$
- $L, L' \in$RegL(Σ) $\quad \curvearrowright L \cup L', LL', L^* \in$ RegL(Σ)

Folg.: RegL(Σ) = $\mathcal{L}(\Sigma$, RegE)

$$L(\Lambda^*) = L(\Lambda)^* = \emptyset^* = \bigcup_{n=0}^{\infty} \emptyset^n = \{\varepsilon\} \qquad (\emptyset^0 = \{\varepsilon\})$$

$$L^+ := \bigcup_{n=1}^{\infty}$$

2.2 Deterministische endliche Automaten

Definition

Seien Q und Σ nicht leere, endliche Mengen, $q_0 \in Q, F \subseteq Q$und δ: $Q \times \Sigma \rightarrow$ Q. Dann heisst

$$\mathfrak{a} = < Q, \Sigma, \delta, q_0, F >$$

ein *deterministischer, endlicher Automat über* Σ mit der Zustandsmenge Q, dem Eingabealphabet Σ, der Transitionsfunktion δ, dem Anfangszustand q_0 und der Endzustandsmenge F.

Bsp:

$\mathfrak{a} =< Q, \Sigma, \delta, q_0, F > \ in\ DFA(\Sigma)$
bestimmt die *erweiterte Transitionsfunktion*

$$\overline{\delta} : Q \times \Sigma^* \to Q$$

mit $\overline{\delta}(q, \varepsilon) := q$

$$\overline{\delta}(q, wa) := \delta(\overline{\delta}(q, w), a) \quad \text{für } w \in \Sigma^*, a \in \Sigma$$

und damit die von $\mathfrak{a}$ erkannte Sprache

$$L(\mathfrak{a}) := \{w \in \Sigma^* \mid \overline{\delta}(q_0, w) \in F\}$$

Bsp:

$\mathcal{L}(\Sigma, DFA)$ Klasse der von endlichen Automaten erkennbaren Sprachen über Σ.

Ziel:

$\mathcal{L}(\Sigma, DFA) =$RegL$(\Sigma)$ nachweisen.
Hilfsmittel: nicht-deterministische Automaten
Hinweis: Scanner, Suchmaschinen, SW-Tools