Einfuehrung in die Theoretische Informatik

HS 2014

Contents

1 Erste Woche

1.1 Sprachen

Alphabet Σ: nichtleere endliche Menge (von Zeichen)

Wort ueber Σ: endliche Folge von Zeichen aus Σ

Leeres Wort: ϵ (epsilon)

Menge aller Woerter ueber Σ: Σ^*

Konkatenation von Woertern x, y ueber Σ:

$x = x_1 x_2 ... x_n \qquad ,x_i \in \Sigma$
$y = y_1 y_2 ... y_n \qquad ,y_i \in \Sigma$

$x \cdot y = xy = x_1 x_2 ... x_n y_1 y_2 ... y_n$

Java: + ”” (ϵ)
Haskell: ++ ”” (ϵ)

Monoid: Sei M eine Menge und
$\circ : M \times M \to^{total} M$ eine Verknuepfung

Das Paar $(M, \circ)$ heisst ein Monoid, falls gilt:

1) $a \circ (b \circ c) = (a \circ b) \circ c \qquad ,\forall a, b, c \in M$

2) Es gibt ein $e \in M$ mit $a \circ e = a = e \circ a \qquad ,\forall a \in M$

□

Beispiel 1

$M = \Sigma^*, \circ = \cdot$

$(\Sigma^*, \cdot)$ ist ein Monoid mit ϵ als neutralem Element

□

Beispiel 2

$\{\{x = 5; y = 6;\} z = 7;\} \equiv \{x = 5; \{y = 6; z = 7;\}\}$

Komposition von Anweisungen assoziativ

Neutrales Element: ; (Java) skip, NOP (no operation)

$(x = 2 * x; x = x + 1;) \not\equiv (x = x + 1; x = 2 * x)$

□

Sprache ueber Σ:

Menge von Woerter ueber Σ

Beispiele

$\{\}$ 0 Woerter
$\{0, 1, 01, 10\}$ Sprache uber $\Sigma = \{0, 1\}$
Σ^*
$\{\epsilon\}$ 1 Wort
$\{\epsilon, 0, 00, 000, ...\}$ uber $\Sigma = \{0\}$

Bem
Sprache kann ∞ viele Woerter enthalten
Jedes Wort ist aber endlich

Bem
$\epsilon \in \Sigma^*$
Σ^* immer ∞ gross

Operationen auf Sprachen

Seien L_1, L_2 Sprachen

$L_1 \cup L_2$ Vereinigungsmenge

$L_1 \cdot L_2 = \{xy \mid x \in L_1, y \in L_2\}$ (Kreuzprodukt)

Sei $(M, \circ)$ ein Monoid. Dann def.

$a^0 = e \quad ,a \in M$
$a^n = a \circ a^{n-1} \quad ,n > 0$

$\square$

$L^0 = \{\epsilon\}$

$L^n = L \cdot L^{n-1} \quad ,n > 0$

Kleen' scher Stern

$L^* = L^0 \cup L^1 \cup L^2 \cup ...$

$= \{x_1, x_2, ..., x_k \mid k \geqslant 0, x_i \in L\}$

Aufgabe

$\Sigma = \{a, b, ..., z\}$, $L_1 = \{good, bad\}$, $L_2 = \{cat, dog\}$

$L_1 \cup L_2 = \{bad, cat, dog, good\}$

$L_1 \cdot L_2 = \{goodcat, gooddog, badcat, baddog\}$

$L_1^0 = \{\epsilon\}$

$L_1^1 = \{good, bad\} = L_1 \cdot L_1^0 = L_1$

$L_1^2 = \{goodgood, goodbad, badgood, badbad\}$

$$L_1^3 = \{goodgoodgood, goodgoodbad, goodbadgood, goodbadbad, \\ badgoodgood, badgoodbad, badbadgood, badbadbad\}$$

$$L_1^* = L_1^0 \cup L_1^1 \cup L_1^2 \cup L_1^3 \cup ... \\ = \{x_1, x_2, ..., x_k \mid k \geqslant 0, x_i \in, L_1\} = \{\epsilon, ...\}$$

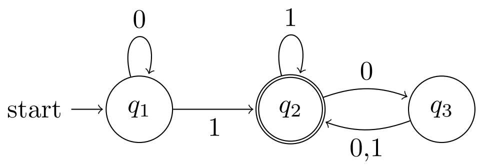
$L_1 \cdot L_2 = \{goodcat, gooddog, badcat, baddog\} \neq L_2 \cdot L_1$

$|M|$ = Anzahl Elemente von M

1.2 Endliche Automaten DFA

deterministic finite automator

Statisch



Dynamisch

Verarbeitung Input: $\xrightarrow{1101}$

1. Start in q_1 Startzustand
2. Lese 1101 , $q_1 \rightarrow q_2$
3. Lese 1101 , $q_2 \rightarrow q_2$
4. Lese 1101 , $q_2 \rightarrow q_3$
5. Lese 1101 , $q_3 \rightarrow q_2$
6. Fertig + akzeptiere, da q_2 akzeptierender Zustand ist und die Eingabe fertig gelesen ist.

Liefert accept oder fertig

Terminiert immer!

Def DFA : Ein DFA ist ein 5-Tupel $(Q, \Sigma, \delta, q_0, F)$ mit:

1. Q ist eine endliche nichtleere Menge von Zustaenden

2. Σ ist das Eingabealphabet (z.B. 1101)

3. $\delta : Q \times \Sigma \rightarrow^{total} Q$
   Transitionsfunktion

4. q_0 Startzustand

5. $F \subseteq Q$ Menge der akzeptierende Zustaende
