Theoretische Informatik 2
Berechenbarkeit und Komplexität

Inoffizielles Skript
Marvin Borner

WARNUNG WIP: Fehler zu erwarten!
Stand: 28. April 2023, 16:22
Bitte meldet euch bei mir, falls ihr Fehler findet.

Vorlesung gehalten von
Ulrike von Luxburg
Sommersemester 2023

Inhalt

1 Reguläre Sprachen und endliche Automaten

1.1 Motivation

- Eingabe
- Verarbeitung (Berechnungen, Zustände)
- Ausgabe

1.2 Wörter und Sprachen

Definition

Ein *Alphabet* Σ sei eine nicht-leere, endliche Menge. Ein *Wort* w ist entsprechend eine Folge von Elementen aus Σ.

Beispiel

- $\Sigma = \{a, ..., z\}$, $w =$ luxburg, $|w| = 7$

Definition

Σ^n ist die Menge aller Wörter der Länge n. Die *Kleene'sche Hülle* ist $\Sigma^* := \bigcup_{n=0}^{\infty} \Sigma^n$.
$\Sigma^+ := \bigcup_{n=1}^{\infty} \Sigma^n$.
Sprache L *über* Σ ist eine Teilmenge von Σ^*.

Definition

Eine *Konkatenation* ist eine Aneinanderhängung zweier Wörter u und w. Eine Konkatenation zweier *Sprachen* L_1, L_2 ist $L_1 \circ L_2 := \{uw \mid u \in L_1,\ w \in L_2\}$. Die Kleene'sche Hülle einer Sprache L ist dann $L^* := \{\underbrace{x_1 ... x_k}_{\text{Konkatenation von } k \text{ Wörtern}} \mid x_i \in L,\ k \in \mathbb{N}_0\}$.
Eine k-fache Aneinanderhängung von Wörtern ist $w_k = \underbrace{w...w}_{k\text{-mal}}$.

Beispiel

- $w = 010$, $u = 001$, $wu = \underbrace{010}_{w}\underbrace{001}_{u}$, $uwu = \underbrace{001}_{u}\underbrace{010}_{w}\underbrace{001}_{u}$
- $w^3 = 010\ 010\ 010$

Bemerkung

Die Konkatenation auf Σ^* hat die Eigenschaften:
- assoziativ: $a(bc) = (ab)c$
- nicht kommutativ: $ab \neq ba$
- neutrales Element ε: $\varepsilon a = a\varepsilon = a$
- ein inverses Element

Definition

Ein Wort x heißt *Teilwort* eines Wortes y, falls es Wörter u und v gibt, sodass $y = uxv$.
- Falls $u = \varepsilon$, x *Präfix* von y
- Falls $v = \varepsilon$, x *Suffix* von y

Beispiel

- 01 ist Teilwort von 0**01**11
- 10 ist Präfix von **10**10011
- 011 ist Suffix von 10101110**011**

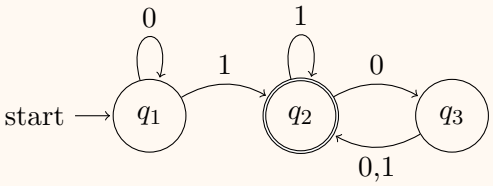
1.3 Endlicher, deterministischer Automat

Definition

Für einen *endlichen, deterministischen Automat* $(Q, \Sigma, \delta, q_0, F)$ ist

- Q eine endliche Menge der *Zustände*
- Σ das *Alphabet*
- $\delta : Q \times \Sigma \to Q$ die Übergangsfunktion
- $q_0 \in Q$ der *Startzustand*
- $F \subset Q$ die Menge der *akzeptierenden Zustände*

Beispiel



$Q = \{q_1, q_2, q_3\}$, $\Sigma = \{0, 1\}$, q_1 Startzustand, $F = \{q_2\}$.
δ kann dargestellt werden durch

/	0	1
q_1	q_1	q_2
q_2	q_3	q_2
q_3	q_2	q_2

Die Zustandsfolge ist mit $w = 001$

$$q_1 \xrightarrow{0} q_1 \xrightarrow{0} q_1 \xrightarrow{1} q_2.$$

Definition

- partielle Übergangsfunktion: nicht alle Übergänge sind definiert
- totale Übergangsfunktion: alle Übergänge sind definiert

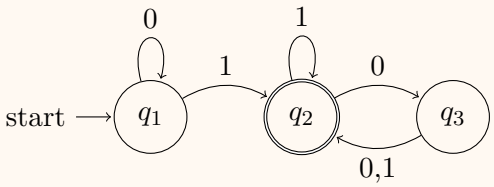
Definition

Eine Folge $s_0, ..., s_n \in Q$ von Zuständen heißt *Berechnung* des Automaten $M = (Q, \Sigma, \delta, q_0, F)$ auf dem Wort $w = w_1 ... w_n$, falls

- $s_0 = q_0$,q
- $\forall i = 0, ..., n-1 : s_{i+1} = \delta(s_i, w_{i+1})$

Es ist also eine “gültige” Folge von Zuständen, die man durch Abarbeiten von w erreicht.

Beispiel



- $w = 001$ ergibt die Zustandsfolge $q_1q_1q_1q_2$

Definition

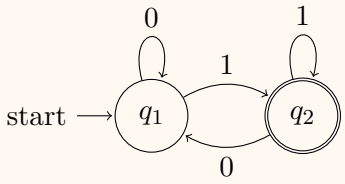
Eine Berechnung *akzeptiert* das Wort w, falls die Berechnung in einem akzeptierten Zustand endet.
Die von einem endlichen Automaten M *akzeptierte* (erkannte) Sprache $L(M)$ ist die Menge der Wörter, die von M akzeptiert werden:

$$L(M) := \{w \in \Sigma^* \mid M \text{ akzeptiert } w\}$$

Bemerkung

Eine Berechnung kann mehrmals in akzeptierenden Zuständen eintreten/austreten. Wichtig ist der Endzustand, nachdem der letzte Buchstabe des Eingabewortes verarbeitet wurde.

Beispiel



- $w = 1101 \to q_1q_2q_2q_1q_2 \to w$ wird akzeptiert
- $w = 010 \to q_1q_1q_2q_1 \to w$ wird **nicht** akzeptiert

Es folgt:

$$L(M) = \{w \in \Sigma^* \mid w = \varepsilon \text{ oder } w \text{ endet mit } 0\}$$

Definition

Sei $\delta : Q \times \Sigma \to Q$ eine Übergangsfunktion. Die *erweiterte Übergangsfunktion* δ^*: $Q \times \Sigma^* \to Q$ sei induktiv definiert:

- $\delta^*(q, \varepsilon) = q$ für alle $q \in Q$
- Für $w \in \Sigma^*$, $a \in \Sigma$ ist:

$$\delta^*(q, wa) = \delta(\underbrace{\delta^*(q, w)}_{\text{Zustand nach Lesen von } w}, \overbrace{a}^{\text{Lesen von Buchstabe } a}).$$

1.4 Reguläre Sprachen und Abschlusseigenschaften

Definition

Eine Sprache $L \subset \Sigma^*$ heißt *reguläre Sprache*, wenn es einen endlichen Automaten M gibt, der diese Sprache akzeptiert.
Die Menge aller regulären Sprachen ist *REG*.

Satz

Sei L eine reguläre Sprache über Σ. Dann ist auch $\bar{L} := \Sigma^* \setminus L$ eine reguläre Sprache.

Beweis

- L regulär $\implies$ es gibt Automaten $M = (Q, \Sigma, \delta, q_0, F)$, der L akzeptiert
- Definiere "Komplementautomat" $\bar{M} = (Q, \Sigma, \delta, q_0, \bar{F})$ mit $\bar{F} := Q \setminus F$.
- Dann gilt:

$$\begin{aligned} w \in \bar{L} &\iff M \text{ akzeptiert } w \text{ nicht} \\ &\iff \bar{M} \text{ akzeptiert } w. \end{aligned}$$

Q.E.D.

Satz

Die Menge der regulären Sprachen ist abgeschlossen bezüglich der Vereinigung:

$$L_1, L_2 \in \text{REG} \implies L_1 \cup L_2 \in \text{REG}.$$

Beweis

Sei $M_1 = (Q_1, \Sigma_1, \delta_1, s_1, F_1)$ ein Automat, der L_1 erkennt, $M_2 = (Q_2, \Sigma_2, \delta_2, s_2, F_2)$ ein Automat, der L_2 erkennt.
Wir definieren den Produktautomaten $M := M_1 \times M_2$: $M = (Q, \Sigma, \Delta, s, F)$ mit

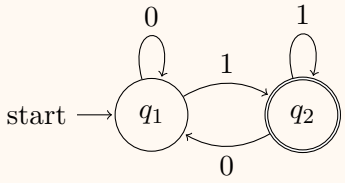
- $Q = Q_1 \times Q_2$
- $\Sigma = \Sigma_1 \cup \Sigma_2$,
- $\underbrace{s}_{\text{neuer Startzustand}} = (s_1, s_2)$, $F = \{(f_1, f_2) \mid f_1 \in F_1 \text{ oder } f_2 \in F_2\}$,
- $\underbrace{\Delta}_{\text{neue Übergangsfunktion}} : Q \times \Sigma \to Q$,

$$\Delta((\underbrace{r_1}_{\in Q_1}, \underbrace{r_2}_{\in Q_2}), \underbrace{a}_{\in \Sigma}) = (\delta_1(r_1, a), \delta(r_2, a)).$$

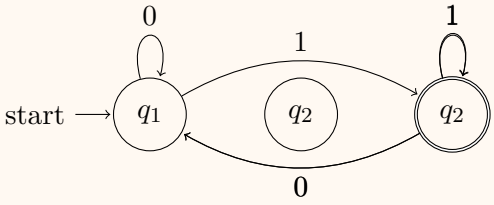
Übertragung der Definition auf erweiterte Übergangsfunktionen: Beweis durch Induktion (ausgelassen).
Nach Definition von F akzeptiert M ein Wort w, wenn M_1 oder M_2 das entsprechende Wort akzeptieren. Der Satz folgt. Q.E.D.

Beispiel

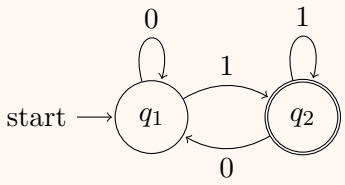
M_1:



M_2: TODO



$M_1 \times M_2$: TODO



Satz

Seien L_1, L_2 zwei reguläre Sprachen. Dann sind auch $L_1 \cap L_2$ und $L_1 \setminus L_2$ reguläre Sprachen.

Beweis

- $L_1 \cap L_2$: Beweis funktioniert analog wie für $L_1 \cup L_2$, nur mit

$$F := \{(q_1, q_2) \mid q_1 \in F_1 \textbf{ und } q_2 \in F_2\}.$$

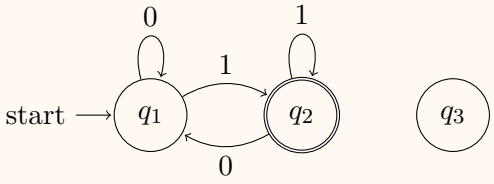
- $L_1 \setminus L_2 = L_q \cap \bar{L_2}$

Q.E.D.

1.5 Nicht-deterministische Automaten

Beispiel

TODO (ggf. auch Sipser).



Definition

Ein *nicht-deterministischer Automat* besteht aus einem 5-Tupel $(Q, \Sigma, \delta, q_0, F)$.

- Q, Σ, q_0, F wie beim deterministischen Automat,
- $\delta : Q \times \Sigma \cup \{\varepsilon\} \to \overbrace{\mathcal{P}(Q)}^{(*)}$ Übergangsfunktion

$(*)$: Die Funktion definiert die **Menge** der möglichen Zustände, in die man von einem Zustand durch Lesen eines Buchstabens gelangen kann.

Definition

Sei $M = (Q, \Sigma, \delta, q_0, F)$ ein nicht-deterministischer endlicher Automat, $w = w_1...w_n \in \Sigma^*$. Eine Folge von Zuständen $s_0, s_1, ..., s_m \in Q$ heißt *Berechnng von M auf w*, falls man w schreiben kann als $w = u_1 u_2 ... u_m$ mit $u_i \in \Sigma \cup \{\underbrace{\varepsilon}_{\text{Übergänge } \varepsilon\text{, hier } u_i = \varepsilon}\}$, sodass

- $s_0 = q_0$,
- für alle $0 \leq i \leq m-1 : s_{i+1} \in \delta(s_1, u_{i+1})$.

Die Berechnung heißt *akzeptierend*, falls $s_m \in F$.
Der nicht-deterministische Automat M *akzeptiert Wort w*, falls es eine akzeptierende Berechnung von M auf w gibt.

Bemerkung

ε-Transitionen: TODO.

Beispiel

Betrachte die regulären Sprachen

- $A := \{x \in \{0,1\}^* \mid \text{Anzahl 0 gerade}\}$
- $B := \{x \in \{0,1\}^* \mid \text{Anzahl 0 ungerade}\}$

Zugehörige Automaten: TODO
Nun betrachte *Konkatenation AB*. Um die Sprache zu erkennen, müsste der Automat bei einer Eingabe zunächst einen ersten Teil A des Wortes betrachten und schauen, ob die Anzahl der 0 gerade ist. **Irgendwann** müsste er beschließen, dass nun der zweite Teil B des Wortes anfängt und er müsste schauen, ob dort die Anzahl der 0 ungerade ist.

"Irgendwann" $\implies$ nicht-deterministisch.

TODO: Graph.

1.6 Mächtigkeit

Bemerkung

Die Mächtigkeit eines Automaten wird hierbei beschrieben durch die Anzahl an Sprachen, die dieser erkennen kann.

Definition

Zwei Automaten M_1, M_2 heißen *äquivalent*, wenn sie die gleiche Sprache erkennen:

$$L(M_1) = L(M_2)$$

Satz

Zu jedem nicht-deterministischen endlichen Automaten gibt es einen äquivalenten deterministischen endlichen Automaten.

Beweis

Lang aber trivial. Basically konstruiert man einfach eine deterministische Übergangsfunktion auf den nicht-deterministischen Verzweigungen.

Satz

Es folgt:
Eine Sprache L ist regulär $\iff$ es gibt einen nicht-deterministischen Automaten, der L akzeptiert.

Satz

Die Klasse der regulären Sprachen ist abgeschlossen unter Konkatenation:

$$L_1, L_2 \in \text{REG} \implies L_1 L_2 \in \text{REG}$$

Satz

Die Klasse REG ist abgeschlossen unter Bildung der Kleene'schen Hülle, d.h.:

$$L \in \text{REG} \implies L^* \in \text{REG}$$

1.7 Reguläre Ausdrücke

Definition

Sei Σ ein Alphabet. Dann:

- $\underbrace{\emptyset}_{\text{leere Sprache}}$ und $\overbrace{\varepsilon}^{\text{leeres Wort}}$ sind reguläre Ausdrücke.
- Alle Buchstaben aus Σ sind reguläre Ausdrücke.
- Falls R_1, R_2 reguläre Ausdrücke sind, dann sind auch die folgenden Ausdrücke regulär:
  - $R_1 \cup R_2$,
  - $R_1 \circ R_2$,
  - R_1^*.

Definition

Sei R ein regulärer Ausdruck. Dann ist die *von R induzierte Sprache* $L(R)$ wie folgt definiert:

- $R = \emptyset \implies L(R) = \emptyset$
- $R = \epsilon \implies L(R) = \{\varepsilon\}$
- $R = \sigma$ für ein $\sigma \in \Sigma \implies L(R) = \{\sigma\}$
- $R = R_1 \cup R_2 \implies L(R) = L(R_1) \cup L(R_2)$
- $R = R_1 \circ R_2 \implies L(R) = L(R_1) \circ L(R_2)$
- $R = R_1^* \implies L(R) = (L(R_1))^*$

Satz

Eine Sprache ist genau dann regulär, wenn sie durch einen regulären Ausdruck beschrieben wird.

Beweis

Strukturelle Induktion. Tja.