Theoretische Informatik 2
Berechenbarkeit und Komplexität

Inoffizielles Skript

Marvin Borner

WARNUNG WIP: Fehler zu erwarten!
Stand: 16. Mai 2023, 00:24
Bitte meldet euch bei mir, falls ihr Fehler findet.

Vorlesung gehalten von
Ulrike von Luxburg
Sommersemester 2023

Inhalt

1 Reguläre Sprachen und endliche Automaten

Motivation

- Eingabe
- Verarbeitung (Berechnungen, Zustände)
- Ausgabe

1.1 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**

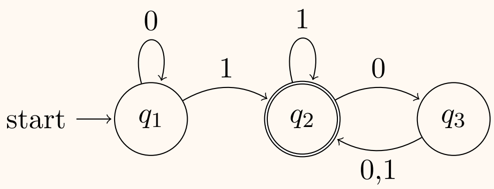
1.2 Endlicher, deterministischer Automat

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

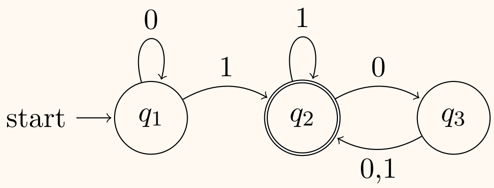
Definition

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 $q_1 q_1 q_1 q_2$

Definition

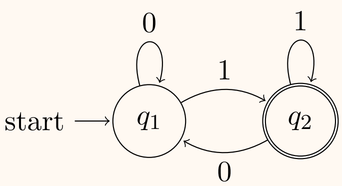
Eine Berechnung *akzeptiert* das Wort w, falls die Berechnung in einem akzeptierten Zustand endet.
Die von einem endlichen Automaten M *akzeptierte* (erkannte) Sprache $L(M)$ ist die Menge der Wörter, die von M akzeptiert werden:

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

Bemerkung

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

Beispiel



- $w = 1101 \to q_1 q_2 q_2 q_1 q_2 \to w$ wird akzeptiert
- $w = 010 \to q_1 q_1 q_2 q_1 \to w$ 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* $\delta^* \colon Q \times \Sigma^* \to Q$ sei induktiv definiert:

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

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

1.3 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

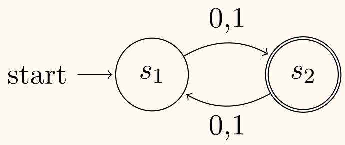
- $Q = Q_1 \times Q_2$
- $\Sigma = \Sigma_1 \cup \Sigma_2$,
- $\underbrace{s}_{\text{neuer Startzustand}} = (s_1, s_2)$, $F = \{(f_1, f_2) \mid f_1 \in F_1 \text{ oder } f_2 \in F_2\}$,
- $\underbrace{\Delta}_{\text{neue Übergangsfunktion}} : Q \times \Sigma \to Q$,

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

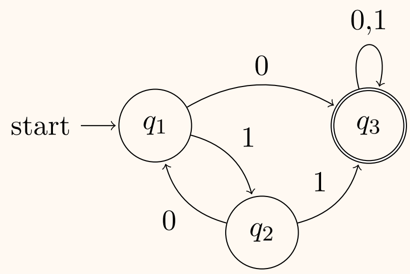
Übertragung der Definition auf erweiterte Übergangsfunktionen: Beweis durch Induktion (ausgelassen).
Nach Definition von F akzeptiert M ein Wort w, wenn M_1 oder M_2 das entsprechende Wort akzeptieren. Der Satz folgt. Q.E.D.

Beispiel

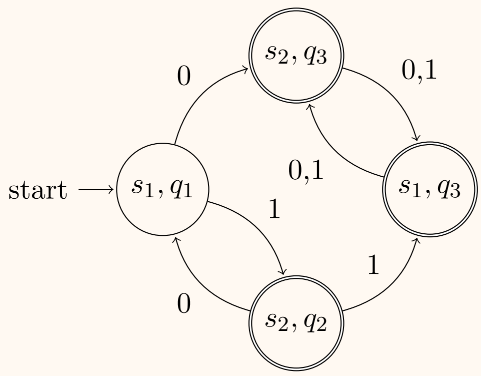
M_1:



M_2:



$M_1 \times M_2$:



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

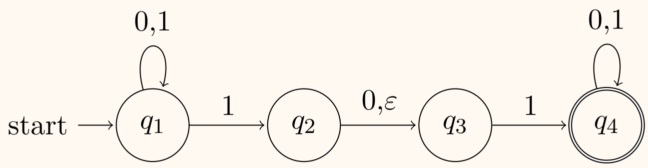
$$F := \{(q_1, q_2) \mid q_1 \in F_1 \textbf{ und } q_2 \in F_2\}.$$

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

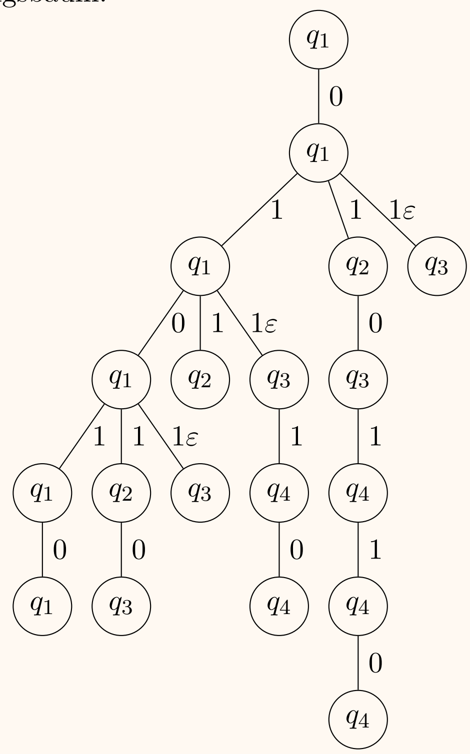
Q.E.D.

1.4 Nicht-deterministische Automaten

Beispiel



Der komplette Berechnungsbaum:



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: Ein nicht-deterministischer Automat kann bei "Lesen" des leeren Wortes ε einen Übergang machen, falls es so in der Übergangsfunktion definiert ist.

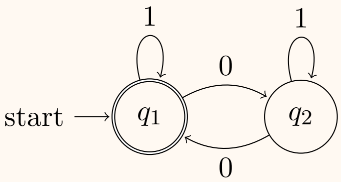
Beispiel

Betrachte die regulären Sprachen

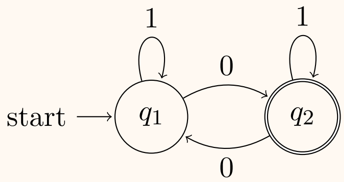
- $A := \{x \in \{0,1\}^* \mid \text{Anzahl 0 gerade}\}$
- $B := \{x \in \{0,1\}^* \mid \text{Anzahl 0 ungerade}\}$

Zugehörige Automaten:

M_A:

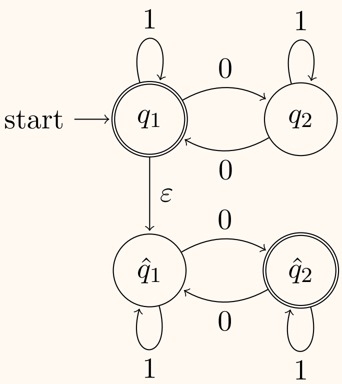


M_B:



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.

$$\text{"Irgendwann"} \implies \text{nicht-deterministisch.}$$



1.5 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.:

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

1.6 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:

- $R = \emptyset \implies L(R) = \emptyset$
- $R = \varepsilon \implies L(R) = \{\varepsilon\}$
- $R = \sigma$ 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.

2 Pumping-Lemma

Motivation

Frage: Gibt es Sprachen, die nicht regulär sind?

Beispiel

$$L = \{0^n 1^n \mid n \in \mathbb{N}\} = \{01, 0011, 00111, ...\}$$

Ein Automat, der L erkennt, müsste vermutlich "zählen" können. Mit endlich vielen Zuständen scheint dies für beliebig große Zahlen nicht möglich zu sein.
Aber: Wir kann man das formal beweisen?

Satz

Sei A eine reguläre Sprache über das Alphabet Σ. Dann gibt es eine natürliche Zahl n, sodass sich alle Wörter $s \in A$ mit Länge $|s| \geq n$ zerlegen lassen in drei Teilworte $s = xyz$, mit $x, y, z \in \Sigma^*$, sodass gilt:

- $|y| > 0$,
- $|xy| \leq n$,
- $\forall i \geq 0 : xy^i z \in A$

Beweis

Idee: Ein Automat mit n Zuständen besucht für die Verarbeitung eines Wortes w mit Länge $> n$ immer $n + 1$ Zustände $\implies$ es gibt einen Zustand, der mindestens zweimal besucht wird.

Bemerkung

Aus der Kontraposition folgt: A nicht regulär $\implies \forall n \exists s \forall x, y, z \in \Sigma^* \exists i : xy^i z \notin A$

Beispiel

Wir zeigen mit dem Pumping-Lemma, dass $L = \{0^n 1^n \mid n \in \mathbb{N}\}$ nicht regulär ist.

Beweis

Betrachte ein beliebiges $n \in \mathbb{N}$. Wähle das Wort $s = 0^n 1^n$ (es gilt $|s| > n$). Sei nun $xyz = s$ eine beliebige Zerlegung (mit $|y| > 0$, $|xy| \leq n$). Man muss nun ein i finden, soadss $xy^i z$ nicht in L ist.

- Fall 1: y besteht nur aus 0en: $s = \overbrace{0}^{x}\overbrace{00}^{y}\overbrace{...0111...1}^{z}$. Dann ist $xy^2z \notin L$ (da es mehr 0en als 1en hat).
- Fall 2: y besteht nur aus 1en: analog.
- Fall 3: y hat 0en und 1en: $s = \overbrace{0...0}^{x}\overbrace{011}^{y}\overbrace{...1}^{z}$. Dann ist aber $xy^2z \notin L$.

2.1 Pushdown automaton

Motivation

Endliche Automaten haben nur endlichen Speicher, können also nicht mal zählen. Deshalb ein erweitertes Modell: Automat mit Stack als Speicher.

Definition

Ein nicht deterministischer *Kellerautomat* besteht aus einem 6-Tupel $(Q, \Sigma, \Gamma, \delta, q_0, F)$, wobei gilt:

- Q endliche Zustandsmenge
- Σ Eingabealphabet
- Γ Stack-Alphabet
- $q_0 \in Q$ Startzustand
- $F \subset Q$ Menge der akzeptierenden Zustände
- Übergangsfunktion:

$$\delta : Q \times (\Sigma \cup \{\varepsilon\}) \times (\Gamma \cup \{\varepsilon\}) \to \mathcal{P}(Q \times (\Gamma \cup \{\varepsilon\}))$$

Beispiel

$$L = \{ww^R \mid w \text{ Wort über } \Sigma\}$$

wobei w^R das Wort w rückwärts ist, kann von einem (nicht-deterministischen) Kellerautomaten erkannt werden.

Beispiel

$$L = \{a^n b^n c^n \mid n \in \mathbb{N}\}$$

, kann von einem Kellerautomaten *nicht* erkannt werden.

Bemerkung

Die Sprachen, die von einem nicht-deterministischen Kellerautomaten erkannt werden können, heißen *kontextfreie Sprachen*. Bei deterministischen Kellerautomaten heißen sie entsprechend *deterministische kontextfreie Sprachen*.

2.2 Grammatiken

Nicht wirklich relevant.

Bemerkung

- Backus-Naur Schreibweise
- Chomsky-Hierarchie

3 Turingmaschinen

Motivation

Ein allgemeineres Modell eines Computers:
- kann eine Eingabe lesen
- hat beliebig viel Speicherplatz
- kann Dinge an beliebigen Stellen in den Speicher schreiben/lesen
- kann beliebig viele Rechenschritte machen

Beispiel

Betrachte die Sprache $L = \{w\#w \mid w \in \{0,1\}^*\}$.
- lies den ersten Buchstaben und merke
- überschreibe mit Symbol $x \notin \{0, 1, \#\}$
- nach rechts bis $\#$ erscheint
- vergleiche nächsten ($\neq y$) Buchstabe mit gemerkten
- falls gleich:
  - überschreibe mit $y \notin \{0, 1, \#, x\}$
  - gehe zurück bis x
  - wiederhole

Definition

Eine *Turingmaschine (TM)* ist ein 7-Tupel $(Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$:
- Q ist eine endliche Menge von *Zuständen*
- Σ ist eine endliche Menge, das *Eingabealphabet*
- Γ ist eine endliche Menge, das *Arbeitsalphabet*, mit $\Sigma \subset \Gamma$ und einem Leerzeichen $\sqcup$
- $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ die *Übergangsfunktion*
- $q_0 \in Q$ der *Startzustand*
- $q_{\text{accept}} \in Q$ der *akzeptierende Endzustand*
- $q_{\text{reject}} \in Q$, $q_{\text{reject}} \neq q_{\text{accept}}$ der *verwerfende Endzustand*

Bemerkung

- Es gibt genau einen akzeptierenden und verwerfenden Zustand
- Die TM beendet ihre Berechnung, sobald sie einen dieser beiden Zustände erreicht
- Das Band der TM hat “ein linkes Ende”, nach rechts ist es unbeschränkt

Beispiel

TM, die $L = \{0^{2^n} \mid n \in \mathbb{N}_0\}$ erkennt.
- $Q = \{q_1, q_2, q_3, q_4, q_5, q_{\text{accept}}, q_{\text{reject}}\}$
- $\Sigma = \{0\}$
- $\Gamma = \{0, x, \sqcup\}$
- $\delta =$ tja? :)

Definition

Eine *Konfiguration* der TM M wird beschrieben durch den Inhalt des Bandes, die Position des Lesekopfes und den derzeitigen Zustand $q \in Q$:

$$uqv$$

- Inhalt des Speicherbandes ist String uv
- Position des Schreibkopfes ist direkt nach u, auf dem ersten Buchstaben von v
- Zustand ist q

Außerdem:
- Die *Startkonfiguration* von M auf Eingabe w ist $q_0 w$
- Eine Konfiguration heißt *akzeptierend/verwerfend*, wenn der Zustand $q_{\text{accept}}/q_{\text{reject}}$ ist

Definition

Eine *Berechnung* der TM M auf Eingabe w ist eine gültige Folge von Konfigurationen $C_0, C_1, C_2, ...$, sodass C_0 die Startkonfiguration ist und die Konfiguration C_{i+1} jeweils in der Übergangsfunktion beschrieben aus C_i hervorgeht.
Eine Berechnung einer TM auf Eingabe w heißt *akzeptierend/verwerfend*, falls sie im Zustand $q_{\text{accept}}/q_{\text{reject}}$ endet.
Eine Berechnung heißt *nicht-akzeptierend*, falls sie entweder in q_{reject} endet oder *nie beendet wird*.

Definition

Eine Sprache L heißt *rekursiv aufzählbar (semi-entscheidbar)*, falls es eine TM M gibt, die L akzeptiert. Das heißt:
- $w \in L \implies M$ akzeptiert w
- $w \notin L \implies M$ verwirft oder hält nicht an

Definition

Eine Sprache L heißt *(rekursiv) entscheidbar*, falls es eine TM M gibt, sodass gilt:
- $w \in L \implies M$ akzeptiert w
- $w \notin L \implies M$ verwirft w

M **hält immer an.**

3.1 Varianten

Definition

Zwei Turing-Machinen M_1, M_2 heißen *äquivalent*, falls sie die gleichen Sprachen akzeptieren: $L(M_1) = L(M_2)$.

$$\begin{aligned} M_1 \text{ akzeptiert } w &\implies M_2 \text{ akzeptiert } w \\ M_1 \text{ akzeptiert } w \text{ nicht} &\implies M_2 \text{ akzeptiert } w \text{ nicht} \end{aligned}$$

Definition

Wir können wie bei endlichen Automaten eine *nicht-deterministische Turingmaschine (NTM)* definieren:

- Zu jedem Zeitpunkt hat die TM mehrere Möglichkeiten, wie sie fortfahren kann
- Formal geht dann die Übergangsfunktion

$$\delta : Q \times \Gamma \to \mathcal{P}(Q \times \Gamma \times \{L, R\})$$

Definition

Eine *Berechnung* der NTM entspricht einem möglichen Pfad im "Baum der möglichen Berechnungen".
Eine Berechnung heißt *akzeptierend/verwerfend*, falls sie in einem *akzeptierenden/verwerfenden* Zustand endet.
Die von der NTM *akzeptierte Sprache* besteht aus den Wörtern, für die es eine akzeptierende Berechnung gibt: Mindestens einer der Pfade im Berechnungsbaum endet im akzeptierenden Zustand.

Bemerkung

- bei DTMs: nicht akzeptierend $\iff$ verwerfen oder nicht terminieren
- bei NTMs: nicht akzeptierend $\iff$ $\forall$Pfade: Pfad verwirft oder endet nicht

Satz

$$\text{DTM} \underbrace{\equiv}_{\text{berechenbarkeitsäquivalent}} \text{NTM}$$

Beweis

Breitensuche im Berechnungsbaum. Blabla offensichtlich.

4 Entscheidbarkeit von Sprachen vs. Berechenbarkeit von Funktionen

Definition

Eine Funktion $f : \Sigma^* \to \Gamma^*$ heißt *(Turing)-berechenbar* oder *totalrekursiv*, falls es eine TM gibt, die bei Eingabe von $w \in \Sigma^*$ den Funktionswert $f(w)$ ausgibt (und insbesondere anhält).

Satz

Eine Sprache $L \subset \Sigma^*$ ist genau dann *entscheidbar*, wenn ihre charakteristische Funktion

$$\mathbb{1}_L : \Sigma^* \to \{0,1\}, \mathbb{1}_L(w) = \begin{cases} 1 & w \in L \\ 0 & \text{sonst} \end{cases}$$

berechenbar ist.

Beweis

- L entscheidbar $\implies \mathbb{1}_L$ berechenbar
  - TM M, die w genau dann akzeptiert, wenn $w \in L$. Erweitern zu $\hat{M}$:
    - falls M akzeptiert, schreibe 1 auf Band und lösche alles andere
    - falls M verwirft, schreibe 0 aufs Band
- $\mathbb{1}_L$ berechenbar $\implies L$ entscheidbar
  - $\hat{M}$ berechnet $\mathbb{1}_L$
    - TM gibt 1 $\implies$ akzeptierender Zustand
    - TM gibt 0 $\implies$ verwerfender Zustand

Q.E.D.

Satz

Eine Funktion $f : \Sigma^* \to \Gamma^*$ ist *berechenbar* genau dann, wenn es eine TM gibt, die die folgende Sprache *entscheidet*:

$$L_f = \{w\#g \mid w \in \Sigma^*, g \in \Gamma^*, f(w) = g\}$$

Beweis

- $f : \Sigma^* \to \Gamma^*$ berechenbar $\implies L_f$ entscheidet
  - TM M_1 berechnet bei $w \in \Sigma^*$ Ausgabe von $f(w)$
  - TM M_2 bekommt $w\#g$ und ruft M_1 mit w auf
  - M_2 wartet auf Ergebnis g_1 von M_1
  - M_2 vergleicht g_1 mit g
- L_f entscheidet $\implies f : \Sigma^* \to \Gamma^*$ berechenbar
  - TM M_2 entscheidet L_f
  - TM M_1 bekommt w
  - M_1 probiert *alle* Antworten von $f(w)$ aus bis die richtige Antwort gefunden wurde
  - sobald L_f akzeptiert, weiß M_1 die Antwort

Q.E.D.

4.1 Gödelnummer

Motivation

Programmierbare Turingmaschinen?
$\implies$ binäre Kodierung für TM?!

Beispiel

Eine beispielhafte Kodierung einer TM $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$ mit $\{0, 1, \#\}$:
- $\Gamma = \{A_1, ..., A_r\}$: $C(A_j) = 10^j1$
- $Q = \{q_1, ..., q_m\}$: $C(q_i) = 110^i11$
- $C(L) = 1110111$ und $C(R) = 11100111$
- $C(\delta(q, a) = (\hat{q}, \hat{a}, B)) = \#C(q)C(a)C(\hat{q})C(\hat{a})C(B)\#$

$\implies C(M) = \#0^m\#0^r\#C(t_1)\#C(t_2)\#...$ mit Transitionen t_i, $m = |Q|$ und $r = |\Gamma|$.
Ein reiner binärer String lässt sich durch $0 \mapsto 00, 1 \mapsto 11, \# \mapsto 01$ bilden.

Bemerkung

Die Kodierung ist
- injektiv: $C(M_1) = C(M_2) \implies M_1 = M_2$.
- *präfixfrei*

Satz

Es gibt eine TM A_{true}, die für einen binären String w entscheidet, ob er eine gültige Kodierung einer TM ist.

Definition

Seien $x, y \in \{0, 1\}^+$ zwei binäre Strings. $x \leq y$ falls n_x, n_y die durch die Strings repräsentiert werden, $n_x \leq n_y$ repräsentieren.
Für $x = \varepsilon, y \in \{0, 1\}^*$ gelte immer $\varepsilon \leq y$.

Bemerkung

Erfüllt die Bedingungen einer *totalen Ordnung*: Transitiv, anti-symmetrisch und total.

Definition

Sei $x \in \{0, 1\}^*$. Für $i \in \mathbb{N}$ nennt man x die Kodierung der *i-ten Turingmaschine*, falls gilt:
- $x = C(M)$ für eine TM M
- $\{y \in \{0, 1\}^* \mid y \leq x\}$ enthält genau $i - 1$ Wörter, die Kodierungen von Turingmaschinen sind.

Satz

Es gibt eine TM A, die für ein $i \in \mathbb{N}$ die Kodierung der i-ten TM berechnet.

Definition

Informell: Die natürliche Zahl (der binäre String), der eine Turingmaschine beschreibt, heißt die *Gödelnummer der TM*. Schreibweise $\langle M \rangle$.
Formell: Sei $\mathcal{M}$ die Menge aller Turingmaschinen. Die *Gödelisierung* sei $g : \mathcal{M} \to \mathbb{N}$, falls gilt:
- g ist injektiv
- $g(\mathcal{M})$ ist entscheidbar (TM konstruierbar, die für jedes $n \in \mathbb{N}$ entscheidet, ob $n \in G(\mathcal{M}) gilt)$
- $g : \mathcal{M} \to N$ und $g^{-1} : g(\mathcal{M}) \to \mathcal{M}$ sind berechenbar

$g(M)$ heißt für $M \in \mathcal{M}$ die *Gödelnummer* von M.

4.2 Die universelle Turingmaschine

Motivation

Turingmaschinen können bis jetzt nur genau ein Programm ausführen, aber wir wollen mehr!! :((

Beispiel

Eine Möglichkeit ist eine *3-Band-TM*:
- Auf Band 1 wird M simuliert
- Auf Band 2 wird die Gödelnummer $\langle M \rangle$ geschrieben
- Auf Band 3 wird der aktuelle Zustand von M vermerkt

Vorbereitung:
- U liest $\langle M \rangle w$ auf Band 1 und teilt sie in $\langle M \rangle$ und w auf
  - bricht ab, falls $\langle M \rangle$ keine korrekte Kodierung ist
- U kopiert $\langle M \rangle$ auf Band 2 und löscht sie von Band 1
- U schreibt die Kodierung des Startzustandes von M auf Band 3

Simulation:
- U weiß mittels Band 3 den aktuellen Zustand
- U liest den aktuellen Buchstaben von Band 1
- U sucht auf Band 2 den zugehörigen Übergang
- U führt Übergang auf Band 1 durch und merkt sich den neuen Zustand in Band 3

Ausgabe:
- U stoppt, sobald Band 3 den akzeptierenden/verwerfenden Zustand erreicht
- Band 1 enthält die Ausgabe der Berechnung

4.3 Abzählbar unendliche Mengen

Motivation

Es gibt **viel** mehr Sprachen als Turingmaschinen $\implies$ es gibt Sprachen, die nicht von TMs erkannt werden können.

Definition

Eine Menge $M \neq \emptyset$ heißt *endlich*, falls es ein $n_0 \in \mathbb{N}$ gibt, für das es eine bijektive Abbildung

$$g : \{1, 2, ..., n_0\} \to M$$

gibt. Andernfalls heißt M *unendlich*.
Im endlichen Fall bezeichnet man mit $|M| := n_0$ die *Kardinalität/Mächtigkeit* der Menge.

Definition

Zwei Mengen M_1, M_2 heißen *gleich mächtig*, falls es eine bijektive Abbildung $M_1 \to M_2$ gibt.
M_2 heißt *mächtiger als* M_1, falls es eine injektive Abbildung $f : M_1 \to M_2$ und keine injektive Abbildung $g : M_2 \to M_1$ gibt.

Definition

Menge M heißt *abzählbar unendlich*, wenn sie gleich mächtig wie $\mathbb{N}$ ist (es existiert Bijektion $f : M \to \mathbb{N}$).
Menge M heißt *höchstens abzählbar*, wenn sie endlich oder abzählbar unendlich ist.
Eine Menge, die weder endlich noch abzählbar ist, heißt *überabzählbar*.

Bemerkung

Menge $\mathbb{N}$ ist unendlich und abzählbar:

$$f : \mathbb{N} \to \mathbb{N}, n \mapsto n \text{ bijektiv}$$

4.3.1 Spaß mit Abzählbarkeit

Motivation

Oh nein mein Hotel hat unendlich viele Zimmer und alle sind besetzt!
Jetzt kommt noch ein Gast aber wo soll der hin??
Dann kommt noch ein Bus mit abzählbar vielen Leuten – was soll ich tun??!?

Beispiel

- $\mathbb{Z}$ ist abzählbar mit folgender Bijektion:

$\mathbb{N}$	1	2	3	4	5	6	7
$\mathbb{Z}$	0	1	-1	2	-2	3	-3

- $\mathbb{N}^2$ ist abzählbar mit cooler zickidizickzack Bijektion
- $\mathbb{Q}$ ist abzählbar mit noch coolerer zickidizickzack Bijektion

Satz

Eine Menge ist genau dann unendlich, wenn es eine echte Teilmenge $U \subset M, M \neq M$ und eine injektive Abbildung $f : M \to U$ gibt.

Satz

Sei M eine beliebige unendliche Menge. Dann ist $\mathbb{N}$ nicht mächtiger als M.

Satz

- Sei A höchstens abzählbar, $f : A \to B$ bijektiv. Dann ist auch B höchstens abzählbar.
- M abzählbar, $N \subset M$. Dann ist N endlich oder abzählbar.
- Seien $M_1, M_2, ...$ abzählbare Mengen. Dann ist $\bigcup_{i \in \mathbb{N}} M_i$ auch abzählbar.
- Endliche Produkte von abzählbaren Mengen sind abzählbar: $M_1, M_2, ..., M_n$ abzählbar. Dann $M_1 \times M_2 \times ... \times M_n$ abzählbar

Bemerkung

Gilt nicht für unendliche Produkte! Bspw. $2^{\mathbb{N}}$ ist überabzählbar.

4.4 Wie groß ist Σ^*?

Satz

Σ^* ist unendlich.

Beweis

- Σ enthält mindestens einen Buchstaben: $a \in \Sigma$
- Σ^* enthält demnach $a, a^2, a^3, ...$
- Also existiert injektive Abbildung $f : \mathbb{N} \to \Sigma^*, f(u) = a^i$, also muss Σ^* unendlich groß sein Q.E.D.

Satz

Σ^* ist abzählbar unendlich.

Beweis

Bijektive Abbildung $f : \mathbb{N} \to \Sigma^*$:

$$\Sigma^* = \{\text{Wörter mit Länge } 0\} \cup \{\text{Wörter mit Länge } 1\} \cup ...$$

- f ist wohldefiniert: Jedes $n \in \mathbb{N}$ erhält genau ein Bildwort $f(n) \in \Sigma^*$ (klar)
- Surjektiv: Jedes Wort von Σ^* kriegt mindestens eine Nummer (klar)
- Injektiv: Jedes Wort von Σ^* kriegt genau eine Nummer (klar) Q.E.D.

Satz

Sei L eine Sprache über einem endlichen Alphabet. Dann ist L höchstens abzählbar.

Satz

Die Menge aller Turingmaschinen ist abzählbar.

Beweis

TM M kann eindeutig durch GM $\langle M \rangle \in \{0,1\}^*$ beschrieben werden und $\{0,1\}^*$ ist abzählbar. Q.E.D.

4.5 Überabzählbare Mengen

Satz

Die Menge der reelen Zahlen ist überabzählbar.

Beweis

Über Cantorsches Diagonalisierungsverfahren. Trivial.

Bemerkung

- $\mathbb{N}$ abzählbar
- $\mathbb{Q}$ abzählbar
- $\mathbb{R}$ überabzählbar

4.5.1 $2^{\mathbb{N}}$ ist überabzählbar

Satz

Die Menge $2^{\mathbb{N}} := \{0,1\}^{\mathbb{N}} := \{(a_i)_{i \in \mathbb{N}} \mid a_i \in \{0,1\}\}$ ist überabzählbar.

Beweis

Widerspruchsbeweis mit Cantor. Trivial.

Bemerkung

$2^{\mathbb{N}}$ und $[0,1] \subset \mathbb{R}$ haben Gemeinsamkeiten:

- Zahlen aus $[0,1]$ als Binärfolge darstellbar
- Erzeugt Bijektion zwischen $2^{\mathbb{N}}$ und $[0,1]$ (außer periodische 1)