Theoretische Informatik HS23

Nicolas Wehrli

Übungsstunde 06
2. November 2023

ETH Zürich
nwehrl@ethz.ch

Heute

1. Feedback zur Serie
2. Nichtdeterministische Turingmaschinen
3. Einstieg Berechnenbarkeit
   - Diagonalisierung
4. Midterm Prep - Repetition
5. Midterm Prep - HS17

Feedback zur Serie

Feedback

- Lemma 3.3 zitieren
- NEA: Konzepte trennen
- Quantoren bei Bonusaufgabe b
- Pumping Lemma üben (aber besser als letzte Woche)

Nichtdeterministische Turingmaschinen

Definition von NTM

Eine **nichtdeterministische Turingmaschine (NTM)** ist ein 7-Tupel $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$, wobei

(i) $Q, \Sigma, \Gamma, q_{\text{accept}}, q_{\text{reject}}$ die gleiche Bedeutung wie bei einer TM haben, und

(ii) $\delta : (Q \setminus \{q_{\text{accept}}, q_{\text{reject}}\}) \times \Gamma \to \mathcal{P}(Q \times \Gamma \times \{\text{L}, \text{R}, \text{N}\})$ die **Übergangsfunktion** von M ist und die folgende Eigenschaft hat:

$$\delta(p, ¢) \subseteq \{(q, ¢, X) \mid q \in Q, X \in \{R, N\}\}$$

für alle $p \in Q$

Konfiguration ähnlich wie bei TMs.

Konfiguration akzeptierend $\iff$ enthält q_{accept}
Konfiguration verwerfend $\iff$ enthält q_{reject}

Die üblichen Sachen

- Schrittrelation $\vdash_M$ "verbindet zwei Konfigurationen, wenn man von der einen in die andere kommen kann"
- Reflexive und transitive Hülle ist $\vdash_M^*$.
- Berechnung von M ist eine Folge von Konfigurationen $C_1, C_2, ...$, so dass $C_i \vdash_M C_{i+1}$.
- Eine Berechnung von M auf x ist beginnt in $q_0 ¢ x$ und endet entweder unendlich oder endet in $\{q_{\text{accept}}, q_{\text{reject}}\}$.

Akzeptierte Sprache

$$L(M) = \{w \in \Sigma^* \mid q_0 ¢ w \vdash_M^* y q_{\text{accept}} z \text{ für irgendwelche } y, z \in \Gamma^*\}$$

Berechnungsbaum

Sei $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$ eine NTM und sei x ein Wort über dem Eingabealphabet Σ von M. Ein **Berechnungsbaum $\mathbf{T_{M,x}}$ von M auf x** ist ein (potentiell unendlicher) gerichteter Baum mit einer Wurzel, der wie folgt definiert wird.

(i) Jeder Knoten von $T_{M,x}$ ist mit einer Konfiguration beschriftet.

(ii) Die Wurzel ist der einzige Knoten von $T_{M,x}$ mit dem Eingangsgrad 0 und ist mit der Startkonfiguration $q_0 ¢ x$ beschriftet.

(iii) Jeder Knoten des Baumes, der mit einer Konfiguration C beschriftet ist, hat genauso viele Kinder wie C Nachfolgekonfigurationen hat, und diese Kinder sind mit diesen Nachfolgekonfigurationen C markiert.

Äquivalenz NTM und TM

Satz 4.2

Sei M eine NTM. Dann existiert eine TM A, so dass

(i) $L(M) = L(A)$ und

(ii) falls M keine unendlichen Berechnungen auf Wörtern aus $L(M)^{\complement}$ hat, dann hält A immer.

Beweisidee:

"BFS im Berechnungsbaum", i.e. wir simulieren einzelne Schritte der verschiedenen Berechnungsstränge.

Einstieg Berechnenbarkeit

Bijektion, Injektion, Schreibweise

Seien A und B zwei Mengen.
Wir sagen, dass

i. $|\mathbf{A}| \leq |\mathbf{B}|$, falls eine injektive Funktion $f : A \to B$ existiert.
ii. $|\mathbf{A}| = |\mathbf{B}|$, falls $|A| \leq |B|$ und $|B| \leq |A|$.
iii. $|\mathbf{A}| < |\mathbf{B}|$, falls $|A| \leq |B|$ und keine injektive Abbildung von B nach A existiert.

Zur Erinnerung:

$$f : A \to B \text{ injektiv } \iff \forall x, y \in A, x \neq y . f(x) \neq f(y)$$

Abzählbarkeit

E

ine Menge A heisst **abzählbar**, falls A endlich ist oder $|A| = |\mathbb{N}|$.

Lemma 5.1

Sei Σ ein beliebiges Alphabet. Dann ist Σ^* abzählbar.

Beweisidee

kanonische Ordnung gibt uns eine Bijektion zwischen $\mathbb{N}$ und Σ^*.

Abzählbarkeit

Satz 5.1

Die Menge **KodTM** der Turingmaschinenkodierungen ist abzählbar.

Beweisidee

KodTM $\subseteq (\Sigma_{\text{bool}})^*$ und Lemma 5.1

Lemma 5.2

$(\mathbb{N} \setminus \{0\}) \times (\mathbb{N} \setminus \{0\})$ ist abzählbar.

Beweisidee

Unendliche 2-dimensionale Tabelle, so dass an der i-ten Zeile und j-ten Spalte, sich das Element $(i, j) \in (\mathbb{N} \setminus \{0\}) \times (\mathbb{N} \setminus \{0\})$ befindet.

Formal definiert man dabei die lineare Ordnung

$$(a, b) < (c, d) \iff a + b < c + d \text{ oder } (a + b = c + d \text{ und } b < d)$$

Abzählbarkeit

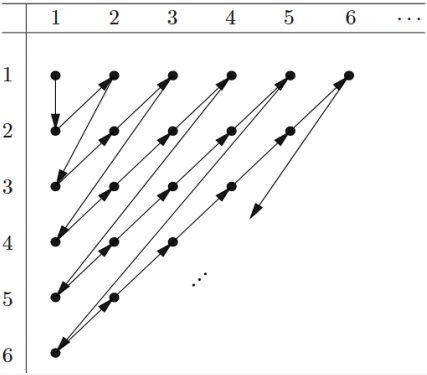


Abbildung 1: Abbildung 5.3 im Buch

Abzählbarkeit

Die i-te Diagonale hat i Elemente. Ein beliebiges Element $(a, b) \in (\mathbb{N} \setminus \{0\}) \times (\mathbb{N} \setminus \{0\})$ ist das b-te Element auf der $(a + b - 1)$-ten Diagonale.

Auf den ersten $a + b - 2$ Diagonalen gibt es

$$\sum_{i=1}^{a+b-2} i = \frac{(a + b - 2) \cdot ((a + b - 2) + 1)}{2} = \binom{a + b - 1}{2}$$

Elemente.

Folglich ist

$$f((a, b)) = \binom{a + b - 1}{2} + b$$

eine Bijektion von $(\mathbb{N} \setminus \{0\}) \times (\mathbb{N} \setminus \{0\})$ nach $\mathbb{N} \setminus \{0\}$.

Überabzählbarkeit

Satz 5.3

$[0, 1]$ ist nicht abzählbar.

Beweisidee

Klassisches Diagonalisierungsargument. Aufpassen auf 0 und 9. I.e. $1 = 0.\overline{99}$.

Überabzählbarkeit

$f(x)$	$x \in [0,1]$							
1	0.	$\boxed{a_{11}}$	a_{12}	a_{13}	a_{14}	$\dots$		
2	0.	a_{21}	$\boxed{a_{22}}$	a_{23}	a_{24}	$\dots$		
3	0.	a_{31}	a_{32}	$\boxed{a_{33}}$	a_{34}	$\dots$		
4	0.	a_{41}	a_{42}	a_{43}	$\boxed{a_{44}}$	$\dots$		
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$		$\dots$		
i	0.	a_{i1}	a_{i2}	a_{i3}	a_{i4}	$\dots$	$\boxed{a_{ii}}$	$\dots$
$\vdots$	$\vdots$							

Abbildung 5.5

Satz 5.4

$\mathcal{P}((\Sigma_{\text{bool}})^*)$ ist nicht abzählbar.

Beweis:

Wir definieren eine injektive Funktion von $f : [0,1] \to \mathcal{P}((\Sigma_{bool})^*)$ und beweisen so $|\mathcal{P}((\Sigma_{bool})^*)| \geq |[0,1]|$.

Sei $a \in [0,1]$ beliebig. Wir können a wie folgt binär darstellen:

$$\text{Nummer}(a) = 0.a_1a_2a_3a_4... \text{ mit } a = \sum_{i=1}^{\infty} a_i \cdot 2^{-i}.$$

Hier ist zu beachten, dass wir für eine Zahl a immer die lexikographisch letzte Darstellung wählen.

Überabzählbarkeit

Dies tun wir, weil eine reelle Zahl 2 verschiedene Binärdarstellungen haben kann. Beispiel: $\frac{1}{2} = 0.1\overline{0} = 0.0\overline{1}$.

Für jedes a definieren wir:

$$f(a) = \{a_1, a_2a_3, a_4a_5a_6, ..., a_{\binom{n}{2}+1}a_{\binom{n}{2}+2}...a_{\binom{n+1}{2}}, ...\}$$

Da $f(a) \subseteq (\Sigma_{bool})^*$ gilt $f(a) \in \mathcal{P}((\Sigma_{bool})^*)$.

Wir haben für alle $n \in \mathbb{N} \setminus \{0\}$, dass $f(a)$ **genau** ein Wort dieser Länge enthält. Nun können wir daraus folgendes schliessen:

Weil die Binärdarstellung zweier unterschiedlichen reellen Zahlen an mindestens einer Stelle unterschiedlich ist, gilt $b \neq c \implies f(b) \neq f(c), \forall b, c \in [0, 1]$.

Überabzählbarkeit

Folglich ist f injektiv und wir haben $|\mathcal{P}((\Sigma_{bool})^*)| \geq |[0,1]|$.

Da $[0,1]$ nicht abzählbar ist, folgt daraus:

$\mathcal{P}((\Sigma_{bool})^*)$ ist nicht abzählbar.

■

Diagonalsprache L_{diag}

Zur Erinnerung:

Rekursiv aufzählbare Sprachen

Eien Sprache $L \subseteq \Sigma^*$ heisst **rekursiv aufzählbar**, falls eine TM M existiert, so dass $L = L(M)$.

$$\mathcal{L}_{\mathbf{RE}} = \{L(M) \mid M \text{ ist eine TM}\}$$

ist die **Klasse aller rekursiv aufzählbaren Sprachen**.

Wir zeigen jetzt per Diagonalisierung, die Existenz einer Sprache die nicht rekursiv aufzählbar ist.

Diagonalsprache L_{diag}

Sei $w_1, w_2, \ldots$ die kanonische Ordnung aller Wörter über Σ_{bool} und sei $M_1, M_2, M_3, \ldots$ die Folge aller Turingmaschinen.

Wir definieren eine unendliche (bool'sche) Matrix $A = [d_{ij}]_{i,j=1,2,\ldots}$ mit

$$d_{ij} = 1 \iff M_i \text{ akzeptiert } w_j.$$

Wir definieren

$$L_{\text{diag}} = \{w \mid w = w_i \text{ und } M_i \text{ akzeptiert } w_i \text{ nicht für ein } i \in \mathbb{N} \setminus \{0\}\}$$

Diagonalsprache L_{diag}

Satz 5.5

$$L_{\text{diag}} \notin \mathcal{L}_{\text{RE}}$$

Beweis:

Wir haben

$$L_{\text{diag}} = \{w \mid w = w_i \text{ und } M_i \text{ akzeptiert } w_i \text{ nicht für ein } i \in \mathbb{N} \setminus \{0\}\}$$

Widerspruchsbeweis:

Sei $L_{\text{diag}} \in \mathcal{L}_{\text{RE}}$. Dann existiert eine TM M, so dass $L(M) = L_{\text{diag}}$. Da diese TM eine TM in der Nummerierung aller TM ist, existiert ein $i \in \mathbb{N}$, so dass $M_i = M$.

Diagonalsprache L_{diag}

Wir betrachten nun das Wort w_i für diese $i \in \mathbb{N}$. Per Definition von L_{diag}, gilt:

$$w_i \in L_{\text{diag}} \iff w_i \notin L(M_i)$$

Da aber $L(M_i) = L_{\text{diag}}$, haben wir folgenden Widerspruch:

$$w_i \in L_{\text{diag}} \iff w_i \notin L_{\text{diag}}$$

Folglich gilt $L_{\text{diag}} \notin \mathcal{L}_{\text{RE}}$.

■

Midterm Prep - Repetition

Kanonische Ordnung

Sei $\Sigma = \{s_1, s_2, ..., s_m\}, m \geq 1$, ein Alphabet und sei $s_1 < s_2 < ... < s_m$ eine Ordnung auf Σ. Wir definieren die **kanonische Ordnung** auf Σ^* für $u, v \in \Sigma^*$ wie folgt:

$$u < v \iff |u| < |v| \vee (|u| = |v| \wedge u = x \cdot s_i \cdot u' \wedge x \cdot s_j \cdot v')$$

für irgendwelche $x, u', v' \in \Sigma^*$ und $i < j$.

Information messen

Wir beschränken uns auf Σ_{bool}

Kolmogorov-Komplexität

Für jedes Wort $x \in (\Sigma_{\text{bool}})^*$ ist die **Kolmogorov-Komplexität** $K(x)$ **des Wortes** x das Minimum der binären Längen, der Pascal-Programme, die x generieren.

$K(x)$ ist die kürzestmögliche Länge einer Beschreibung von x.

Die einfachste (und triviale) Beschreibung von x, ist wenn man x direkt angibt.

x kann aber eine Struktur oder Regelmässigkeit haben, die eine Komprimierung erlaubt.

Welche Programmiersprache gewählt wird verändert die Kolmogorov-Komplexität nur um eine Konstante. (Satz 2.1)

Grundlegende Resultate

Es existiert eine Konstante d, so dass für jedes $x \in (\Sigma_{\text{bool}})^*$

$$K(x) \leq |x| + d$$

Die **Kolmogorov-Komplexität einer natürlichen Zahl** n ist $K(n) = K(\text{Bin}(n))$.

Lemma 2.5 - Nichtkomprimierbar

Für jede Zahl $n \in \mathbb{N} \setminus \{0\}$ existiert ein Wort $w_n \in (\Sigma_{\text{bool}})^n$, so dass

$$K(w_n) \geq |w_n| = n$$

Lemma 2.5 - Beweis

Es gibt 2^n Wörter $x_1, ..., x_{2^n}$ über Σ_{bool} der Länge n. Wir bezeichnen $C(x_i)$ als den Bitstring des kürzesten Programms, der x_i generieren kann. Es ist klar, dass für $i \neq j : C(x_i) \neq C(x_j)$.

Die Anzahl der nichtleeren Bitstrings, i.e. der Wörter der Länge $< n$ über Σ_{bool} ist:

$$\sum_{i=1}^{n-1} 2^i = 2^n - 2 < 2^n$$

Also muss es unter den Wörtern $x_1, ..., x_{2^n}$ mindestens ein Wort x_k mit $K(x_k) \geq n$ geben.

■

Satz 2.2

Sei L eine Sprache über Σ_{bool}. Sei für jedes $n \in \mathbb{N} \setminus \{0\}$, z_n das n-te Wort in L bezüglich der kanonischen Ordnung. Wenn ein Programm A_L existiert, das das Entscheidungsproblem $(\Sigma_{\text{bool}}, L)$ löst, dann gilt für alle $n \in \mathbb{N} \setminus \{0\}$, dass

$$K(z_n) \leq \lceil \log_2(n+1) \rceil + c$$

wobei c eine von n unabhängige Konstante ist.

Satz 2.2 - Beweisidee

Wir können aus A_L, ein Programm entwerfen, dass das kanonisch n-te Wort generiert, indem wir in der kanonischen Reihenfolge alle Wörter $x \in (\Sigma_{bool})^*$ durchgehen und mit A_L entscheiden, ob $x \in L$. Dann können wir einen Counter c haben und den Prozess abbrechen, wenn der Counter $c = n$ wird und dann dieses Wort ausgeben.

Wir sehen, dass dieses Programm ausser der Eingabe n immer gleich ist. Sei die Länge dieses Programms c, dann können wir für das n-te Wort der Sprache L, z_n, die Kolmogorov-Komplexität auf n reduzieren, bzw:

$$K(z_n) \leq \lceil \log_2(n+1) \rceil + c$$

■

Primzahlsatz

Für jede positive ganz Zahl n sei $\mathrm{Prim}(n)$ die Anzahl der Primzahlen kleiner gleich n.

$$\lim_{n\to\infty} \frac{\mathrm{Prim}(n)}{n/\ln n} = 1$$

Nützliche Ungleichung

$$\ln n - \frac{3}{2} < \frac{n}{\mathrm{Prim(n)}} < \ln n - \frac{1}{2}$$

für alle $n \geq 67$.

Lemma 2.6 - schwache Version des Primzahlsatzes

Sei $n_1, n_2, n_3, ...$ eine steigende unendliche Folge natürlicher Zahlen mit $K(n_i) \geq \lceil \log_2 n_i \rceil / 2$. Für jedes $i \in \mathbb{N} \setminus \{0\}$ sei q_i die grösste Primzahl, die die Zahl n_i teilt. Dann ist die Menge $Q = \{q_i \mid i \in \mathbb{N} \setminus \{0\}\}$ unendlich.

Beweis: Wir beweisen diese Aussage per Widerspruch:

Nehmen wir zum Widerspruch an, dass die Menge $Q = \{q_i \mid i \in \mathbb{N} \setminus \{0\}\}$ sei endlich.

Sei q_m die grösste Primzahl in Q. Dann können wir jede Zahl n_i eindeutig als

$$n_i = q_1^{r_{i,1}} \cdot q_2^{r_{i,2}} \cdot \dots \cdot q_m^{r_{i,m}}$$

für irgendwelche $r_{i,1}, r_{i,2}, ..., r_{i,m} \in \mathbb{N}$ darstellen. Sei c die binäre Länge eines Programms, dass diese $r_{i,j}$ als Eingaben nimmt und n_i erzeugt (A ist für alle $i \in \mathbb{N}$ bis auf die Eingaben $r_{i,1}, ..., r_{i,m}$ gleich).

Lemma 2.6 - Beweis continued

Dann gilt:

$$K(n_i) \leq c + 8 \cdot (\lceil \log_2(r_{i,1} + 1) \rceil + \lceil \log_2(r_{i,2} + 1) \rceil + ... + \lceil \log_2(r_{i,m} + 1) \rceil)$$

Die multiplikative Konstante 8 kommt daher, dass wir für die Zahlen $r_{i,1}, r_{i,2}, ..., r_{i,m}$ dieselbe Kodierung, wie für den Rest des Programmes verwenden (z.B. ASCII-Kodierung), damit ihre Darstellungen eindeutig voneinander getrennt werden können. Weil $r_{i,j} \leq \log_2 n_i, \forall j \in \{1, ..., m\}$ erhalten wir

$$K(n_i) \leq c + 8m \cdot \lceil \log_2(\log_2 n_i + 1) \rceil, \forall i \in \mathbb{N} \setminus \{0\}$$

Lemma 2.6 - Beweis continued 2

Weil m und c Konstanten unabhängig von i sind, kann

$$\lceil \log_2 n_i \rceil / 2 \leq K(n_i) \leq c + 8m \cdot \lceil \log_2(\log_2 n_i + 1) \rceil$$

$$\lceil \log_2 n_i \rceil / 2 \leq c + 8m \cdot \lceil \log_2(\log_2 n_i + 1) \rceil$$

nur für endlich viele $i \in \mathbb{N} \setminus \{0\}$ gelten.

Dies ist ein Widerspruch!

Folglich ist die Menge Q unendlich.

■

Kolmogorov - Aufgabentyp 1

Sei $w_n = (010)^{3^{2n^3}} \in \{0,1\}^*$ für alle $n \in \mathbb{N} \setminus \{0\}$. Gib eine möglichst gute obere Schranke für die Kolmogorov-Komplexität von w_n an, gemessen in der Länge von w_n.

Wir zeigen ein Programm, dass n als Eingabe nimmt und w_n druckt:

```
W_n :   begin
            M := n;
            M := 2 × M × M × M;
            J := 1;
            for I = 1 to M
                J := J × 3;
            for I = 1 to J;
                write(010);
        end
```

Kolmogorov - Aufgabentyp 1

Der einzige variable Teil dieses Algorithmus ist n. Der restliche Code ist von konstanter Länge. Die binäre Länge dieses Programms kann von oben durch

$$\lceil \log_2(n+1) \rceil + c$$

beschränkt werden, für eine Konstante c.

Somit folgt

$$K(w_n) \leq \log_2(n) + c'$$

Wir berechnen die Länge von w_n als $|w_n| = |010| \cdot 3^{2n^3} = 3^{2n^3+1}$.

Kolmogorov - Aufgabentyp 1

Mit ein wenig umrechnen erhalten wir

$$n = \sqrt[3]{\frac{\log_3 |w_n| - 1}{2}}$$

und die obere Schranke

$$K(w_n) \leq \log_2 \left(\sqrt[3]{\frac{\log_3 |w_n| - 1}{2}} \right) + c' \leq \log_2 \log_3 |w_n| + c''$$

Kolmogorov - Aufgabentyp 2

Geben Sie eine unendliche Folge von Wörtern $y_1 < y_2 < ...$ an, so dass eine Konstante $c \in \mathbb{N}$ existiert, so dass für alle $i \geq 1$

$$K(y_i) \leq \log_2 \log_2 \log_3 \log_2(|y_i|) + c$$

Wir definieren die Folge $y_1, y_2, ...$ mit $y_i = 0^{2^{3^{2^i}}}$ für alle $i \in \mathbb{N}$. Da $|y_i| < |y_{i+1}|$ folgt die geforderte Ordnung.

Es gilt

$$i = \log_2 \log_3 \log_2 |y_i| \text{ für } i \geq 1$$

Wir zeigen ein Programm, dass i als Eingabe nimmt und y_i druckt:

Kolmogorov - Aufgabentyp 2

```
begin
        M := i;
        M := 2^(3^(2^M));
        for I = 1 to M;
                write(010);
end
```

Das ^ für die Exponentiation ist nicht Teil der originalen Pascal Syntax, aber wir verwenden es um unser Programm lesbarer zu machen.

Kolmogorov - Aufgabentyp 2

Der einzige variable Teil dieses Programms ist das i. Der Rest hat konstante Länge. Demnach kann die Länge diese Programms für eine Konstante c' durch

$$\lceil \log_2(i+1) \rceil + c'$$

von oben beschränkt werden.

Somit folgt

$$\begin{aligned} K(y_i) &\leq \log_2(i) + c \\ &\leq \log_2 \log_2 \log_3 \log_2 |y_i| + c \end{aligned}$$

für eine Konstante c.

Kolmogorov - Aufgabentyp 3

Sei $M = \{7^i \mid i \in \mathbb{N},\ i \leq 2^n - 1\}$. Beweisen Sie, dass mindestens sieben Achtel der Zahlen in M Kolmogorov-Komplexität von mindestens $n - 3$ haben.

Wir zeigen, dass höchstens $\frac{1}{8}$ der Zahlen $x \in M$ eine Kolmogorov-Komplexität $K(x) \leq n - 4$ haben.

Nehmen wir zum Widerspruch an, dass M mehr als $\frac{1}{8}|M|$ Zahlen x enthält, mit $K(x) \leq n - 4$.

Die Programme, die diese Wörter generieren, müssen paarweise verschieden sein, da die Wörter paarweise verschieden sind.

Es gibt aber höchstens

$$\sum_{k=0}^{n-4} 2^k = 2^{n-3} - 1 < \frac{1}{8}|M|$$

Bitstrings mit Länge $\leq n - 4$. **Widerspruch.**

Erster Ansatz zur Modellierung von Algorithmen

Ein (deterministischer) **endlicher Automat (EA)** ist ein Quintupel $M = (Q, \Sigma, \delta, q_0, F)$, wobei

(i) Q eine endliche Menge von **Zuständen** ist,

(ii) Σ ein Alphabet, genannt **Eingabealphabet**, ist,

(iii) $q_0 \in Q$ der Anfangszustand ist,

(iv) $F \subseteq Q$ die **Menge der akzeptierenden Zustände** ist und

(v) $\delta : Q \times \Sigma \to Q$ die **Übergangsfunktion** ist.

Die **von M akzeptierte Sprache** $L(M)$ ist definiert als

$$\begin{aligned} L(M) &= \{w \in \Sigma^* \mid \text{Berechnung von } M \text{ auf } w \text{ endet in } (p, \lambda) \in F \times \{\lambda\}\} \\ &= \{w \in \Sigma^* \mid (q_0, w) \vdash_M^* (p, \lambda) \wedge p \in F\} \\ &= \{w \in \Sigma^* \mid \hat{\delta}(q_0, w) \in F\} \end{aligned}$$

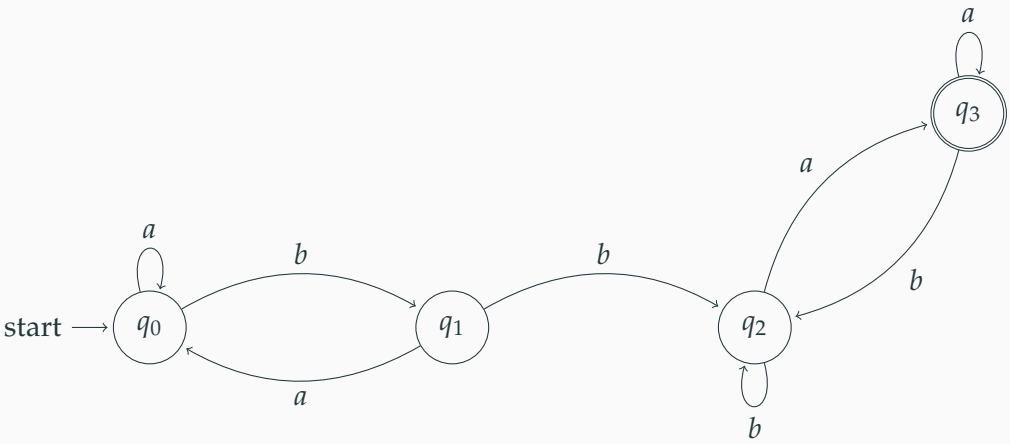
$\mathcal{L}_{\mathbf{EA}} = \{L(M) \mid M \text{ ist ein EA}\}$ ist die Klasse der Sprachen, die von endlichen Automaten akzeptiert werden.
$\mathcal{L}_{\text{EA}}$ bezeichnet man auch als die **Klasse der regulären Sprachen**, und jede Sprache $L \in \mathcal{L}_{\text{EA}}$ wird **regulär** genannt.

EA Konstruktion - Beispielaufgabe

Entwerfen sie für folgende Sprache einen Endlichen Automat und geben Sie eine Beschreibung von Kl$[q]$ für jeden Zustand $q \in Q$.

$$L_1 = \{xbbya \in \{a, b\}^* \mid x, y \in \{a, b\}^*\}$$

EA Konstruktion - Beispielaufgabe



EA Konstruktion - Beispielaufgabe

Wir beschreiben nun die Klassen für die Zustände q_0, q_1, q_2, q_3:

$\text{Kl}[q_0] = \{wa \in \{a,b\}^* \mid \text{Das Wort } w \text{ enthält nicht die Teilfolge } bb\} \cup \{\lambda\}$

$\text{Kl}[q_1] = \{wb \in \{a,b\}^* \mid \text{Das Wort } w \text{ enthält nicht die Teilfolge } bb\}$

$\text{Kl}[q_3] = \{wa \in \{a,b\}^* \mid \text{Das Wort } w \text{ enthält die Teilfolge } bb\} = L_1$

$\text{Kl}[q_2] = \{a,b\}^* - (\text{Kl}[q_0] \cup \text{Kl}[q_1] \cup \text{Kl}[q_3])$

Produktautomaten - Lemma 3.2

Sei Σ ein Alphabet und seien $M_1 = (Q_1, \Sigma, \delta_1, q_{01}, F_1)$ und $M_2 = (Q_2, \Sigma, \delta_2, q_{02}, F_2)$ zwei EA. Für jede Mengenoperation $\odot \in \{\cup, \cap, -\}$ existiert ein EA M, so dass

$$L(M) = L(M_1) \odot L(M_2).$$

Sei $M = (Q, \Sigma, \delta, q_0, F_\odot)$, wobei

(i) $Q = Q_1 \times Q_2$

(ii) $q_0 = (q_{01}, q_{02})$

(iii) für alle $q \in Q_1$, $p \in Q_2$ und $a \in Sigma$, $\delta((q,p),a) = (\delta_1(q,a), \delta_2(p,a))$,

(iv) falls $\odot = \cup$, dann ist $F = F_1 \times Q_2 \cup Q_1 \times F_2$
falls $\odot = \cap$, dann ist $F = F_1 \times F_2$, und
falls $\odot = -$, dann ist $F = F_1 \times (Q_2 - F_2)$.

Produktautomat - Beispielaufgabe

Verwenden Sie die Methode des modularen Entwurfs (Konstruktion eines Produktautomaten), um einen endlichen Automaten (in Diagrammdarstellung) für die Sprache

$$L = \{w \in \{a, b\}^* \mid |w|_a = 2 \text{ oder } w = ya\}$$

zu entwerfen. Zeichnen Sie auch jeden der Teilautomaten und geben Sie für die Teilautomaten für jeden Zustand q die Klasse $\mathrm{Kl}[q]$ an.

Produktautomat - Beispielaufgabe

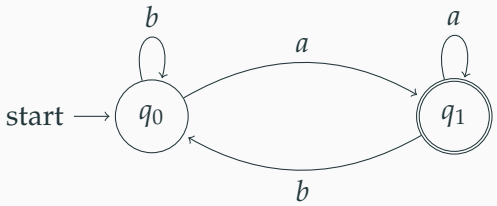
Wir teilen L wie folgt auf:

$$L = L_1 \cup L_2 \text{ wobei gilt:}$$
$$L_1 = \{w \in \{a,b\}^* \mid w = ya\}$$
$$L_2 = \{w \in \{a,b\}^* \mid |w|_a = 2\}$$

Zuerst zeichnen wir die 2 einzelnen Teilautomaten und geben für jeden Zustand q bzw. p die Klasse Kl[q] respektive Kl[p] an:

Produktautomat - Beispielaufgabe

erster Teilautomat: $L_1 = \{w \in \{a,b\}^* \mid w = ya\}$



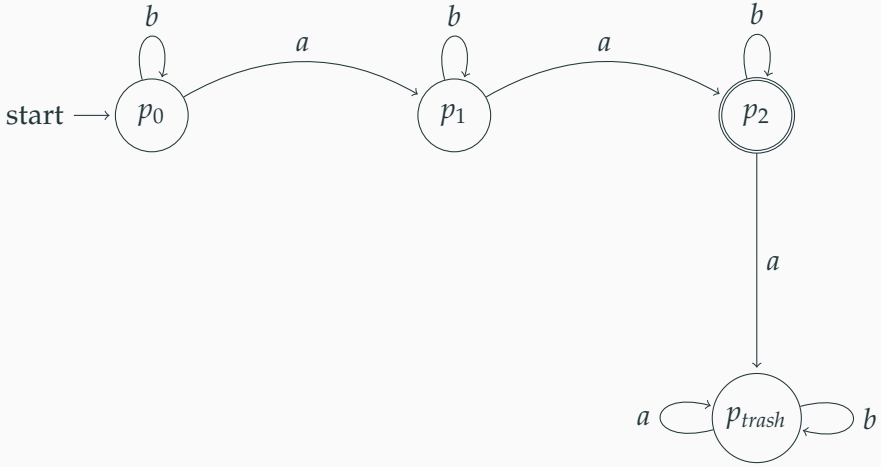
Wir beschreiben nun die Zustände für die Klassen q_0 und q_1:

$\text{Kl}[q_0] = \{yb \mid y \in \{a,b\}^*\} \cup \{\lambda\}$

$\text{Kl}[q_1] = \{ya \mid y \in \{a,b\}^*\}$

Produktautomat - Beispielaufgabe

zweiter Teilautomat: $L_2 = \{w \in \{a, b\}^* \mid |w|_a = 2\}$



Produktautomat - Beispielaufgabe

Wir beschreiben nun die Zustände für die Klassen p_0, p_1, p_2, p_{trash}:

$\mathrm{Kl}[p_0] = \{w \in \{a,b\}^* \mid |w|_a = 0\}$

$\mathrm{Kl}[p_1] = \{w \in \{a,b\}^* \mid |w|_a = 1\}$

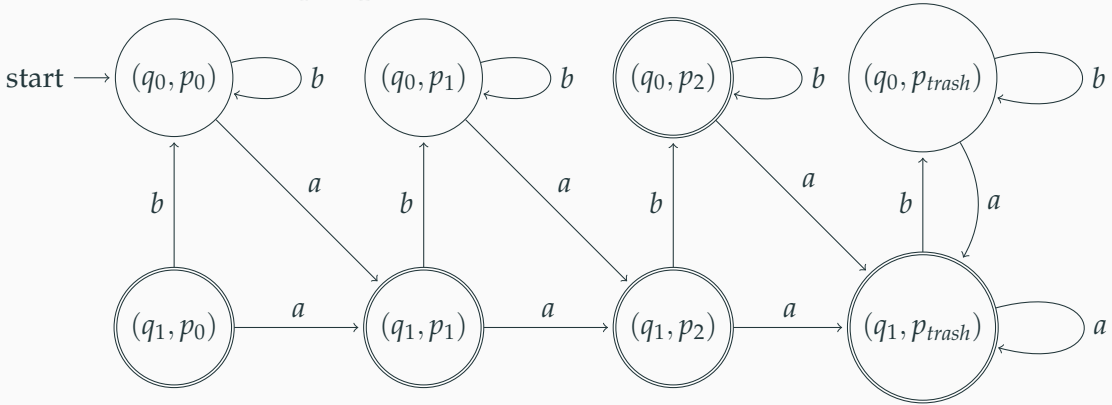
$\mathrm{Kl}[p_2] = \{w \in \{a,b\}^* \mid |w|_a = 2\}$

$\mathrm{Kl}[p_{trash}] = \{w \in \{a,b\}^* \mid |w|_a > 2\}$

Produktautomat - Beispielaufgabe

Zum Schluss kombinieren wir diese Teilautomaten zu einem Produktautomaten:

Produktautomat: $L = L_1 \cup L_2$



Theorie für Nichtregularitätsbeweise - Lemma 3.3

Sei $A = (Q, \Sigma, \delta_A, q_0, F)$ ein EA. Seien $x, y \in \Sigma^*, x \neq y$, so dass

$$\hat{\delta}_A(q_0, x) = p = \hat{\delta}_A(q_0, y)$$

für ein $p \in Q$ (also $x, y \in \mathrm{Kl}[p]$). Dann existiert für jedes $z \in \Sigma^*$ ein $r \in Q$, so dass xz und $yz \in \mathrm{Kl}[r]$, also gilt insbesondere

$$xz \in L(A) \iff yz \in L(A)$$

Theorie für Nichtregularitätsbeweise - Lemma 3.3

Beweis:

Aus der Existenz der Berechnungen

$(q_0, x) \vdash_A^* (p, \lambda)$ und $(q_0, y) \vdash_A^* (p, \lambda)$ von A folgt die Existenz der Berechnungen auf xz und yz:

$(q_0, xz) \vdash_A^* (p, z)$ und $(q_0, yz) \vdash_A^* (p, z)$ für alle $z \in \Sigma^*$.

Wenn $r = \hat{\delta}_A(p, z)$ ist, dann ist die Berechnung von A auf xz und yz:

$(q_0, xz) \vdash_A^* (p, z) \vdash_A^* (r, \lambda)$ und $(q_0, yz) \vdash_A^* (p, z) \vdash_A^* (r, \lambda)$.

Wenn $r \in F$, dann sind beide Wörter xz und yz in $L(A)$. Falls $r \notin F$, dann sind $xz, yz \notin L(A)$.

■

Theorie für Nichtregulärität sbeweise - Pumping Lemma

Sei L regulär. Dann existiert eine Konstante $n_0 \in \mathbb{N}$, so dass jedes Wort $w \in \Sigma^*$ mit $|w| \geq n_0$ in drei Teile x, y und z zerlegen lässt, das heisst $w = yxz$, wobei

(i) $|yx| \leq n_0$

(ii) $|x| \geq 1$

(iii) entweder $\{yx^kz \mid k \in \mathbb{N}\} \subseteq L$ oder $\{yx^kz \mid k \in \mathbb{N}\} \cap L = \emptyset$.

Beweis

Sei $L \in \Sigma^*$ regulär. Dann existiert ein EA $A = (Q, \Sigma, \delta_A, q_0, F)$, so dass $L(A) = L$.

Sei $n_0 = |Q|$ und $w \in \Sigma^*$ mit $|w| \geq n_0$. Dann ist $w = w_1 w_2 ... w_{n_0} u$, wobei $w_i \in \Sigma$ für $i = 1, ..., n_0$ und $u \in \Sigma^*$. Betrachten wir die Berechnung auf $w_1 w_2 ... w_{n_0}$:

$$(q_0, w_1 w_2 w_3 ... w_{n_0}) \vdash_A (q_1, w_2 w_3 ... w_{n_0}) \vdash_A ... \vdash_A (q_{n_0-1}, w_{n_0}) \vdash_A (q_{n_0}, \lambda)$$

In dieser Berechnung kommen $n_0 + 1$ Zustände $q_0, q_1, ..., q_{n_0}$ vor. Da $|Q| = n_0$, existieren $i, j \in \{0, 1, ..., n_0\}, i < j$, so dass $q_i = q_j$. Daher haben wir in der Berechnung die Konfigurationen

$$(q_0, w_1 w_2 w_3 ... w_{n_0}) \vdash_A^* (q_i, w_{i+1} w_{i+2} ... w_{n_0}) \vdash_A^* (q_i, w_{j+1} ... w_{n_0}) \vdash_A^* (q_{n_0}, \lambda)$$

Dies impliziert

$$(q_i, w_{i+1} w_{i+2} ... w_j) \vdash_A^* (q_i, \lambda) \qquad (1)$$

Wir setzen nun $y = w_1 ... w_i$, $x = w_{i+1} ... w_j$ und $z = w_{j+1} ... w_{n_0} u$, so dass $w = yxz$.

Wir überprüfen nun die Eigenschaften (i),(ii) und (iii):

(i) $yx = w_1 ... w_i w_{i+1} ... w_j$ und daher $|yx| = j \leq n_0$.

(ii) Da $|x| \geq j - i$ und $i < j$, ist $|x| \geq 1$.

(iii) (1) impliziert $(q_i, x^k) \vdash_A^* (q_i, \lambda)$ für alle $k \in \mathbb{N}$. Folglich gilt für alle $k \in \mathbb{N}$:

$$(q_0, yx^k z) \vdash_A^* (q_i, x^k z) \vdash_A^* (q_i, z) \vdash_A^* (\hat{\delta}_A(q_i, z), \lambda)$$

Wir sehen, dass für alle $k \in \mathbb{N}$ die Berechnungen im gleichen Zustand $q_{end} = \hat{\delta}_A(q_i, z)$ enden. Falls also $q_{end} \in F$, akzeptiert A alle Wörter aus $\{yx^k z \mid k \in \mathbb{N}\}$. Falls $q_{end} \notin F$, dann akzeptiert A kein Wort aus $\{yx^k z \mid k \in \mathbb{N}\}$.

■

Theorie für Nichtregularitätsbeweise - Satz 3.1 (Kolmogorov)

Sei $L \subseteq (\Sigma_{\text{bool}})^*$ eine reguläre Sprache. Sei $L_x = \{y \in (\Sigma_{\text{bool}})^* \mid xy \in L\}$ für jedes $x \in (\Sigma_{\text{bool}})^*$. Dann existiert eine Konstante **const**, so dass für alle $x, y \in (\Sigma_{\text{bool}})^*$

$$K(y) \leq \lceil \log_2(n+1) \rceil + \textbf{const},$$

falls y das n-te Wort in der Sprache L_x ist.

Wie wir sehen werden, beruht der Nichtregularitätsbeweis darauf, dass die Differenz von $|w_{n+1}| - |w_n|$ für kanonische Wörter $(w_i)_{i \in \mathbb{N}}$ beliebig gross werden kann.

Sprachen mit Einsymbolalphabet

Angenommen es handelt sich bei $L \subseteq \Sigma^*$ um eine Sprache über einem unären Alphabet ($|\Sigma| = 1, \Sigma = \{x\}$).

Dann gilt:

$$\forall w \in \Sigma^* : w = x^{|w|}$$

Insbesondere gibt es für jede Länge nur ein Wort.

Sei die Folge $(w_i)_{i \in \mathbb{N}}$ kanonisch geordnet, so dass $w_i \in L$ (Wenn L endlich betrachten wir nur endlich viele Wörter der Folge).

Durch das gilt folgendes

$$\forall w \in \Sigma^*. \, \forall k \in \mathbb{N}. \, |w_k| < |w| < |w_{k+1}| \implies w \notin L$$

Definition NEA

Ein **nichtdeterministischer endlicher Automat (NEA)** ist ein Quintupel $M = (Q, \Sigma, \delta, q_0, F)$. Dabei ist

(i) Q eine endliche Menge, **Zustandsmenge** genannt,
(ii) Σ ein Alphabet, **Eingabealphabet** genannt,
(iii) $q_0 \in Q$ der **Anfangszustand**,
(iv) $F \subseteq Q$ die Menge der **akzeptierenden Zustände** und
(v) δ eine Funktion von $Q \times \Sigma$ nach $\mathcal{P}(Q)$, **Übergangsfunktion genannt**.

Ein NEA kann zu einem Zustand q und einem gelesenen Zeichen a mehrere oder gar keinen Nachfolgezustand haben.

Potenzmengenkonstruktion

Sei $M = (Q, \Sigma, \delta_M, q_0, F)$ ein NEA. Wir konstrurieren einen äquivalenten Endlichen Automaten $A = (Q_A, \Sigma_A, \delta_A, q_{0A}, F_A)$.

(i) $Q_A = \{\langle P \rangle \mid P \subseteq Q\}$

(ii) $\Sigma_A = \Sigma$

(iii) $q_{0A} = \langle \{q_0\} \rangle$

(iv) $F_A = \{\langle P \rangle \mid P \subseteq Q \text{ und } P \cap F \neq \emptyset\}$

(v) $\delta_A : (Q_A \times \Sigma_A) \to Q_A$ ist eine Funktion, definiert wie folgt. Für jedes $\langle P \rangle \in Q_A$ und jedes $a \in \Sigma_A$ ist

$$\begin{aligned} \delta_A(\langle P \rangle, a) &= \left\langle \bigcup_{p \in P} \delta_M(p, a) \right\rangle \\ &= \langle \{q \in Q \mid \exists p \in P, \text{ so dass } q \in \delta_M(p, a)\} \rangle \end{aligned}$$

Potenzmengenkonstruktion mit Beispiel NEA

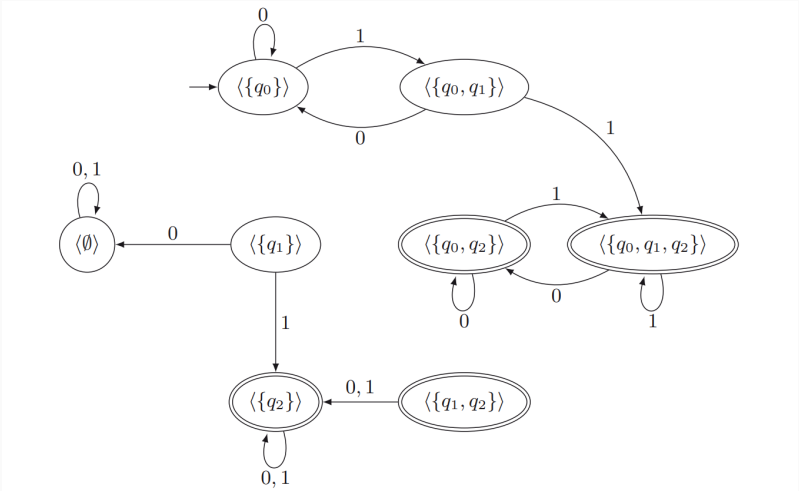


Abbildung 2: Abb. 3.18 im Buch

Exponentiell mehr Zustände - manchmal

Sei

$$L_k = \{x1y \mid x \in (\Sigma_{\text{bool}})^*,\ y \in (\Sigma_{\text{bool}})^{k-1}\}$$

Folgender NEA A_k mit $k+1$ akzeptiert L_k.

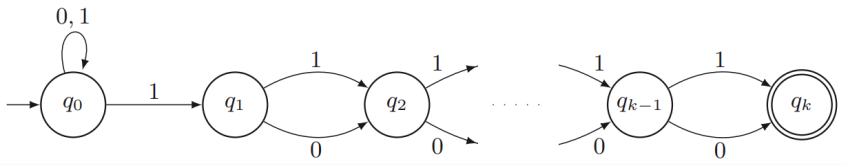


Abbildung 3: Abb. 3.19 im Buch

Exponentiell mehr Zustände - manchmal

Lemma 3.6

Für alle $k \in \mathbb{N} \setminus \{0\}$ muss jeder EA, der L_k akzeptiert, mindestens 2^k Zustände haben.

Beweis

Sei $B_k = (Q_k, \Sigma_{bool}, \delta_k, q_{0k}, F_k)$ ein EA mit $L(B_k) = L_k$.

Nach **Lemma 3.3** gilt für $x, y \in (\Sigma_{bool})^*$:

Wenn $\hat{\delta}_k(q_{0k}, x) = \hat{\delta}_k(q_{0k}, y)$, dann gilt für alle $z \in (\Sigma_{bool})^*$:

$$xz \in L(B_k) \iff yz \in L(B_k)$$

Exponentiell mehr Zustände - manchmal

Die Idee des Beweises ist es, eine Menge S_k von Wörtern zu finden, so dass für keine zwei unterschiedlichen Wörter $x, y \in S_k$ die Gleichung $\hat{\delta}_k(q_{0k}, x) = \hat{\delta}_k(q_{0k}, y)$ gelten darf. Dann müsste B_k mindestens $|S_k|$ viele Zustände haben.

Wir wählen $S_k = (\Sigma_{bool})^k$ und zeigen, dass $\hat{\delta}_k(q_{0k}, u)$ paarweise unterschiedliche Zustände für alle $u \in S_k$ sind.

Wir beweisen dies per Widerspruch.

Exponentiell mehr Zustände - manchmal

Seien $x = x_1x_2...x_k$ und $y = y_1y_2...y_k$ für $x_i, y_i \in \Sigma_{bool}, i \in \{1, ..., k\}$ zwei unterschiedliche Wörter aus S_k.

Nehmen wir zum Widerspruch an, dass $\hat{\delta}_k(q_{0k}, x) = \hat{\delta}_k(q_{0k}, y)$.

Weil $x \neq y$, existiert ein $j \in \{1, ..., k\}$, so dass $x_j \neq y_j$. O.B.d.A. setzen wir $x_j = 1$ und $y_j = 0$. Betrachten wir nun $z = 0^{j-1}$. Dann ist

$xz = x_1...x_{j-1}1x_{j+1}...x_k0^{j-1}$ und $yz = y_1...y_{j-1}0y_{j+1}...y_k0^{j-1}$

und daher $xz \in L_k$ und $yz \notin L_k$.

Dies ist ein Widerspruch! Folglich gilt $\hat{\delta}_k(q_{0k}, x) \neq \hat{\delta}_k(q_{0k}, y)$ für alle paarweise unterschiedliche $x, y \in S_k = (\Sigma_{bool})^k$.

Daher hat B_k mindestens $|S_k| = 2^k$ viele Zustände.

Mindestanzahl Zustände n - Beweisschema

Die Grundidee ist es n Wörter anzugeben und zu beweisen, dass jedes von diesen n Wörtern in einem eigenen Zustand enden muss.

Seien $w_1, ..., w_n$ diese Wörter. Dann geben wir für jedes Paar von Wörtern $w_i \neq w_j$ einen Suffix $z_{i,j}$ an, so dass folgendes gilt:

$$w_i z_{i,j} \in L \iff w_j z_{i,j} \in L$$

Dann folgt aus Lemma 3.3

$$\hat{\delta}(q_0, w_i) \neq \hat{\delta}(q_0, w_j)$$

Es eignet sich die Suffixe als Tabelle anzugeben.

Um die Wörter und Suffixe zu finden, kann es sich als nützlich erweisen, den Endlichen Automaten zu konstruieren.

Mindestanzahl Zustände n - Beweisschema

Wir nehmen zum Widerspruch an, dass es einen EA für L gibt mit weniger als n Zuständen.

Betrachten wir $w_1, ..., w_n$. Per Pigeonhole-Principle existiert $i < j$, so dass

$$\hat{\delta}(q_0, w_i) = \hat{\delta}(q_0, w_j)$$

Per Lemma 3.3 folgt daraus, dass

$$\forall z \in \Sigma^* : w_i z \in L \iff w_j z \in L$$

Für $z = z_{i,j}$ gilt aber per Tabelle

$$w_i z_{i,j} \in L \not\iff w_j z_{i,j} \in L \quad (\mathbf{1})$$

für alle $i < j$.

Da keines der n Wörter im gleichen Zustand enden kann, ergibt sich ein Widerspruch.

Mindestanzahl Zustände n - Beweisschema

Dann noch Angabe der Tabelle für (**1**)

	w_2	...	w_n
w_1	$z_{1,2}$	...	$z_{1,n}$
...		...	...
w_{n-1}			$z_{n-1,n}$

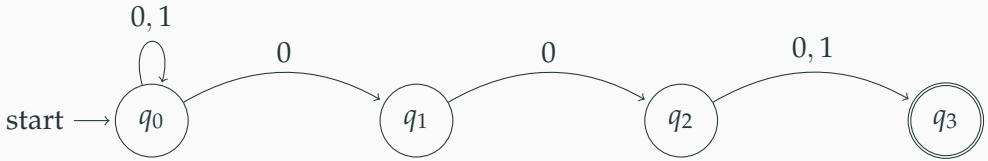
- Wenn es offensichtlich ist, muss (**1**) nicht bei jedem Suffix begründet werden.
- Ein minimaler endlicher Automat ist nicht notwendig für den Beweis. Hilft aber fürs
  i. Finden der w_i
  ii. Finden der $z_{i,j}$
  iii. Beweis von $w_i z_{i,j} \in L \not\Longleftrightarrow w_j z_{i,j} \in L$ (Leicht überprüfbar)

Klassische Aufgabe - HS19 Aufgabe 3.a

Wir betrachten die Sprache

$$L = \{x00y \mid x \in \{0,1\}^* \text{ und } y \in \{0,1\}\}$$

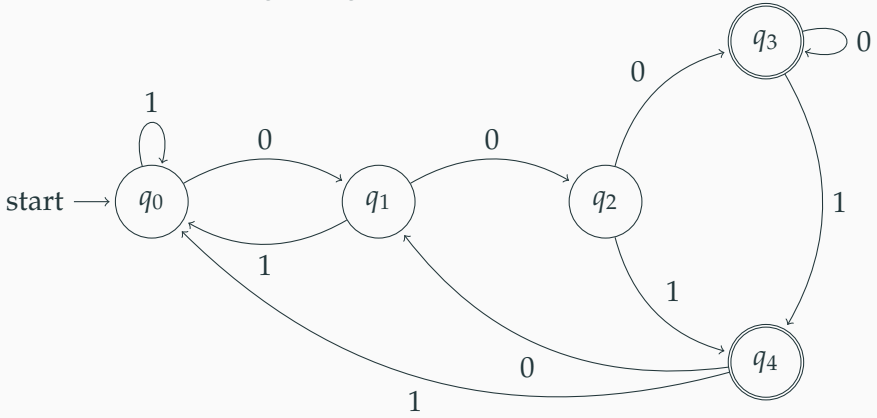
Konstruieren Sie einen nichtdeterminstischen endlichen Automaten mit höchstens 4 Zuständen, der L akzeptiert.



Klassische Aufgabe - HS19 Aufgabe 3.b

Zeigen Sie, dass jeder deterministische endliche Automat, der L akzeptiert, mindestens 5 Zustände braucht.

Wir zeichnen den zugehörigen EA zuerst.



Klassische Aufgabe - HS19 Aufgabe 3.b

Nehmen wir zum Widerspruch an, dass es einen endlichen Automaten gibt, der L akzeptiert und weniger als 4 Zustände hat.

Wir wählen die Wörter $B = \{\lambda, 0, 00, 000, 001\}$.

Nach dem Pigeonhole-Principle existieren zwei Wörter $w_i, w_j \in B, w_i \neq w_j$, so dass

$$\hat{\delta}(q_0, w_i) = \hat{\delta}(q_0, w_j)$$

Per Lemma 3.3 folgt daraus, dass

$$\forall z \in \Sigma^* : w_i z \in L \iff w_j z \in L$$

Klassische Aufgabe - HS19 Aufgabe 3.b

Wir betrachten folgende Tabelle mit Suffixen.

	0	00	000	001
λ	01	1	λ	λ
0		1	λ	λ
00			λ	λ
000				1

Der zeigt für jedes Wortpaar $x, y \in B, x \neq y$ die Existenz eines Suffixes z, so dass

$$(xz \in L \wedge yz \notin L) \vee (xz \notin L \wedge yz \in L)$$

Dies kann man mit den angegebenen Suffixen und dem angegebenen EA einfach überprüfen.

Klassische Aufgabe - HS19 Aufgabe 3.b

Dies widerspricht der vorigen Aussage, dass ein Wortpaar $w_i, w_j \in B, w_i \neq w_j$ existiert, so dass

$$\forall z \in \Sigma^* : w_i z \in L \iff w_j z \in L$$

Somit ist unsere Annahme falsch und L nicht regulär.

■

Mindestanzahl Zustände n - Bemerkung

Manchmal ist es zu schwierig einen minimalen EA zu finden und es funktioniert einfacher die Wörter durch Trial and Error zu finden. (Siehe Midterm HS22)

Turing Maschinen - Formalisierung von Algorithmen

Informell

Eine Turingmaschine besteht aus

(i) einer endlichen Kontrolle, die das Programm enthält,

(ii) einem unendlichen Band, das als Eingabeband, aber auch als Speicher (Arbeitsband) zur Verfügung steht, und

(iii) einem Lese-/Schreibkopf, der sich in beiden Richtungen auf dem Band bewegen kann.

Für formale Beschreibung siehe Buch.

Elementare Operation einer TM - Informell

Input

- Zustand der Maschine (der Kontrolle)
- Symbol auf dem Feld unter dem Lese-/Schreibkopf

Aktion

(i) ändert Zustand

(ii) schreibt auf das Feld unter dem Lese-/Schreibkopf

(iii) bewegt den Lese-/Schreibkopf nach links, rechts oder gar nicht. Ausser wenn ¢, dann ist links nicht möglich.

Berechnung von M, Berechnung von M auf einer Eingabe x etc. durch $\vdash_M$ definiert.

Die Berechnung von M auf x heisst

- **akzeptierend**, falls sie in einer akzeptierenden Konfiguration $w_1 q_{\text{accept}} w_2$ endet (wobei ¢ in w_1 enthalten ist).
- **verwerfend**, wenn sie in in einer verwerfenden Konfiguration $w_1 q_{\text{reject}} w_2$ endet.
- **nicht-akzeptierend**, wenn sie entweder eine **verwerfende** oder unendliche Berechnung ist.

Die von der **Turingmaschine M akzeptierte Sprache** ist

$$\mathbf{L}(\mathbf{M}) = \{w \in \Sigma^* \mid q_0 ¢ w \vdash_M^* y q_{\text{accept}} z, \text{ für irgendwelche } y, z \in \Gamma^*\}$$

Wichtige Klassen

Reguläre Sprachen

$$\mathcal{L}_{\mathbf{EA}} = \{L(A) \mid A \text{ ist ein EA}\} = \mathcal{L}_{\mathbf{NEA}}$$

Rekursiv aufzählbare Sprachen

Eien Sprache $L \subseteq \Sigma^*$ heisst **rekursiv aufzählbar**, falls eine TM M existiert, so dass $L = L(M)$.

$$\mathcal{L}_{\mathbf{RE}} = \{L(M) \mid M \text{ ist eine TM}\}$$

ist die **Klasse aller rekursiv aufzählbaren Sprachen**.

Wichtige Klassen

Halten

Wir sagen das M **immer hält**, wenn für alle Eingaben $x \in \Sigma^*$

(i) $q_0 ¢ x \vdash_M^* y q_{\text{accept}} z, y, z \in \Gamma^*,$ falls $x \in L$ und

(ii) $q_0 ¢ x \vdash_M^* u q_{\text{reject}} v, u, v \in \Gamma^*,$ falls $x \notin L$.

Rekusive Sprachen

Eine Sprache $L \subseteq \Sigma^*$ heisst **rekursiv (entscheidbar)**, falls $L = L(M)$ für eine TM M, die **immer hält**.

$$\mathcal{L}_{\mathbf{R}} = \{L(M) \mid M \text{ ist eine TM, die immer hält}\}$$

ist die **Klasse der rekursiven (algorithmisch erkennbaren) Sprachen**.

Mehrband-Turingmaschine

Mehrband-TM - Informelle Beschreibung

Für $k \in \mathbb{N} \setminus \{0\}$ hat eine k-Band Turingmaschine

- eine endliche Kontrolle
- ein endliches Band mit einem Lesekopf (Eingabeband)
- k Arbeitsbänder, jedes mit eigenem Lese-/Schreibkopf (nach rechts unendlich)

Insbesondere gilt 1-Band TM $\neq$ "normale" TM

Am Anfang der Berechnung einer MTM M auf w

- Arbeitsbänder "leer" und die k Lese-/Schreibköpfe auf Position 0.
- Inhalt des Eingabebands ¢w\$ und Lesekopf auf Position 0.
- Endliche Kontrolle im Zustand q_0.

Midterm Prep - HS17

Aufgabe 1

(a) Konstruieren Sie einen deterministischen endlichen Automaten (in graphischer Darstellung), der die Sprache

$$L = \{xa \mid x \in \{a, b\}^* \text{ und } (|x|_a + 2|x|_b) \bmod 3 = 1\}$$

(b) Geben Sie die Zustandsklasse Kl[q] für jeden Zustand q Ihres in Aufgabenteil (a) konstruierten Automaten.

Aufgabe 2

Zeigen Sie, dass die folgenden Sprachen nicht regulär sind.

(a)

$$L_1 = \{w \in \{0,1\}^* \mid \text{ es gibt ein } k \in \mathbb{N} \text{ mit } |w|_0 = k^2 \text{ oder } |w|_1 = k^3\}$$

(b)

$$L_2 = \{0^{n!} \mid n \in \mathbb{N}\}$$

Aufgabe 3.a

Sei für alle $i \in \mathbb{N} \setminus \{0\}$ die natürliche Zahl x_i definiert durch

$$x_i = 2^i \cdot 3^i \cdot 5^i.$$

Zeigen Sie, dass eine Konstante $c \in \mathbb{N}$ existiert, so dass für alle $i \in \mathbb{N} \setminus \{0\}$ gilt, dass

$$K(x_i) \leq c + \lceil \log_2 \log_2 x_i \rceil.$$

Aufgabe 3.b

Konstruieren Sie eine unendliche Folge von natürlichen Zahlen $n_1, n_2, \ldots$, so dass die folgenden Bedingungen gelten:

(i) $n_i < n_{i+1}$,

(ii) die Menge aller Primfaktoren, die in mindestens einer der Zahlen n_i vorkommen, ist unendlich und

(iii) es existiert eine Konstante $c \in \mathbb{N}$, so dass

$$K(n_i) \leq c + \lceil \log_2 \log_2 n_i \rceil.$$

Weisen Sie nach, dass die von Ihnen konstruierte Zahlenfolge tatsächlich die Bedingungen (i) bis (iii) erfüllt.