Theoretische Informatik HS24

Nicolas Wehrli

Übungsstunde 07

5. November 2024

ETH Zürich
nwehrl@ethz.ch

Heute

1. Feedback zur Serie
2. Repetition - Aufgabenschema
   - How To Kolmogorov
   - Nichtregularitätsbeweise
   - Mindestanzahl Zustände
3. Midterm Prep - Aufgaben HS18
4. Appendix - Theory Recap

Feedback zur Serie

Feedback

- **Aufgabe 16**: Typos, n_0 war frei wählbar (und musste auf > 0 eingeschränkt werden), Quantoren
- **Aufgabe 17**: Transitionen innerhalb von A_1, A_2 für c mussten definiert werden (sowohl bei NEAs als auch bei EAs).
- Bei **EA**:

  $$\delta : Q \times \Sigma \to Q$$

  Bei **NEA**:

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

Repetition - Aufgabenschema

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.

Aufgabentyp 1

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

$$
\begin{array}{lll}
W_n: & \textbf{begin} & \\
 & & M := n; \\
 & & M := 2 \times M \times M \times M; \\
 & & J := 1; \\
 & & \textbf{for } I = 1 \textbf{ to } M \\
 & & \qquad J := J \times 3; \\
 & & \textbf{for } I = 1 \textbf{ to } J; \\
 & & \qquad \textbf{write}(010); \\
 & \textbf{end} &
\end{array}
$$

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 $|w_n| = |010| \cdot 3^{2n^3} = 3^{2n^3+1}$.

Aufgabentyp 1

Mit ein wenig umrechnen erhalten wir

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

und die obere Schranke

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

Aufgabentyp 2

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

$$K(y_i) \leq \log_2 \log_4 \log_3 \log_2(|y_i|) + c$$

Wir definieren die Folge $y_1, y_2, \ldots$ mit $y_i = 0^{2^{3^{4^i}}}$ für alle $i \in \mathbb{N}$. Da $|y_i| < |y_{i+1}|$ folgt die geforderte Ordnung.

Es gilt

$$i = \log_4 \log_3 \log_2 |y_i| \text{ für } i \geq 1$$

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

Aufgabentyp 2

```
begin
        M := i;
        M := 2^(3^(4^M));
        for I = 1 to M;
                write(0);
end
```

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

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

$$\begin{aligned} K(y_i) &\leq \log_2(i) + c \\ &\leq \log_2 \log_4 \log_3 \log_2 |y_i| + c \end{aligned}$$

für eine Konstante c.

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 $K(x) \leq n - 4$.

Aufgabentyp 3

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.**

Beispielaufgabe - Lemma 3.3

Nehmen wir zum Widerspruch an $L = \{0^n 1^n \mid n \in \mathbb{N}\}$ sei regulär.

Dann existiert ein EA $A = (Q, \Sigma, \delta, q_0, F)$ mit $L(A) = L$.

Wir betrachten die Wörter $0^1, \ldots, 0^{|Q|+1}$. Per Pigeonhole-Principle existiert O.B.d.A. $i < j$, so dass

$$\hat{\delta}(q_0, 0^i) = \hat{\delta}(q_0, 0^j)$$

Nach Lemma 3.3 gilt

$$0^i z \in L \iff 0^j z \in L$$

für alle $z \in (\Sigma_{\text{bool}})^*$.

Dies führt aber zu einem Widerspruch, weil für $z = 1^i$ das Wort $0^i 1^i \in L$ aber $0^j 1^i \notin L$.

Schema Lemma 3.3

Hier ist der markierte Teil, der einzige Teil der je nach Aufgabe anders ausgefüllt werden muss.

Wahl der Wörter

Wahl des Suffixes

Argumentation zum Widerspruch. Die Argumentation kann more involved sein, wenn nicht offensichtlich.

Beispielaufgabe - Pumping Lemma

Versuchen wir zu beweisen, dass

$$L_2 = \{wabw^{\mathbf{R}} \mid w \in \{a,b\}^*\}$$

nicht regulär ist.

Beispielaufgabe - Pumping Lemma

Wir nehmen zum Widerspruch an, dass L_2 regulär ist.

Das Pumping-Lemma (Lemma 3.4) besagt, dass dann eine Konstante $n_0 \in \mathbb{N}$ existiert, so dass sich jedes Wort $w \in \Sigma*$ mit $|w| \geq n_0$ in drei Teile y, x, und z zerlegen lässt. ($\implies w = yxz$). Wobei folgendes gelten muss:

(i) $|yx| \leq n_0$

(ii) $|x| \geq 1$

(iii) **entweder** $\{yx^k z \mid k \in \mathbb{N}\} \subseteq L_2$ **oder** $\{yx^k z \mid k \in \mathbb{N}\} \cap L_2 = \emptyset$

Beispielaufgabe - Pumping Lemma

Wir wählen $w = a^{n_0}aba^{n_0}$. Da $|w| = 2n_0 + 2 \geq n_0$ gilt das PL.

Sei y, x, z die Zerlegung die (i)-(iii) nach dem PL erfüllt. Da nach (i), $|yx| \leq n_0$ gelten muss, haben wir $y = a^l$ und $x = a^m$ für beliebige $l, m \in \mathbb{N}, l + m \leq n_0$.

Somit gilt $z = a^{n_0-(l+m)}aba^{n_0}$.

Nach (ii) ist $m \geq 1$.

Für $k = 1$ haben wir $yx^1z = a^{n_0}aba^{n_0} \in L_2$.

Beispielaufgabe - Pumping Lemma

Für $k = 0$:

$$\Rightarrow yx^0z = yz = a^{n_0 - m}aba^{n_0} \notin L_2$$

da $m \geq 1$.

Dies ist ein Widerspruch zu (iii). Somit ist L_2 nicht regulär.

Schema - Pumping Lemma

Der markierte Teil ändert sich je nach Aufgabe.

Wahl des Wortes

Anwendung von (i) und (ii) (kann eine Case-Distinction sein)

Wahl des k für den Widerspruch

Widerspruch herleiten (braucht manchmal ausführliche Argumente)

Beispielaufgabe - Kolmogorov Methode

Verwenden Sie die Methode der Kolmogorov-Komplexität, um zu zeigen, dass die Sprache

$$L_1 = \{0^{n^2 \cdot 2^n} \mid n \in \mathbb{N}\}$$

nicht regulär ist.

Beispielaufgabe - Kolmogorov Methode

Angenommen L_1 sei regulär.

Wir betrachten

$$L_{0^{m^2 \cdot 2^m+1}} = \{y \mid 0^{m^2 \cdot 2^m+1} y \in L_1\}$$

für $m \in \mathbb{N}$ beliebig.

Da

$$\begin{aligned}(m+1)^2 \cdot 2^{m+1} - (m^2 \cdot 2^m + 1) &= (m^2 + 2m + 1) \cdot 2^{m+1} - m^2 \cdot 2^m - 1 \\ &= m^2 \cdot 2^m + m^2 \cdot 2^m + (2m+1) \cdot 2^{m+1} - m^2 \cdot 2^m - 1 \\ &= (m^2 + 4m + 2) \cdot 2^m - 1\end{aligned}$$

ist $y_1 = 0^{(m^2+4m+2)\cdot 2^m - 1}$ das kanonisch erste Wort der Sprache $L_{0^{m^2 \cdot 2^m+1}}$

Nach Satz 3.1 existiert eine Konstante c, unabhängig von m, so dass

$$K(y_1) \leq \lceil \log_2(1+1) \rceil + c = 1 + c.$$

Die Anzahl aller Programme, deren Länge kleiner oder gleich $1 + c$ sind, ist endlich.

Da es aber unendlich **unterschiedliche** Wörter der Form $0^{(m^2+4m+2)\cdot 2^m - 1}$ gibt, ist dies ein Widerspruch. Demzufolge ist L_1 nicht regulär.

$\blacksquare$

Kolmogorov Methode - Schema

Der markierte Teil ändert sich je nach Aufgabe.

Wahl der Präfixsprache

Richtiges erstes/zweites Wort

Beweis dass es unendlich viele unterschiedliche y_1 gibt, für Widerspruch

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

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

für alle $i < j$.

Da keines der n Wörter im gleichen Zustand enden kann: Widerspruch.

Mindestanzahl Zustände n - Beweisschema

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

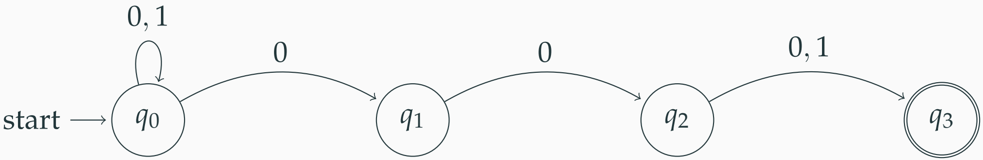
	w_2	...	w_n
w_1	$z_{1,2}$	...	$z_{1,n}$
...		...	...
w_{n-1}			$z_{n-1,n}$

- 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 $w_i z_{i,j} \in L \iff\!\!\!\!\!\!\!/\;\; w_j z_{i,j} \in L$ (Leicht überprüfbar)

Wir betrachten die Sprache

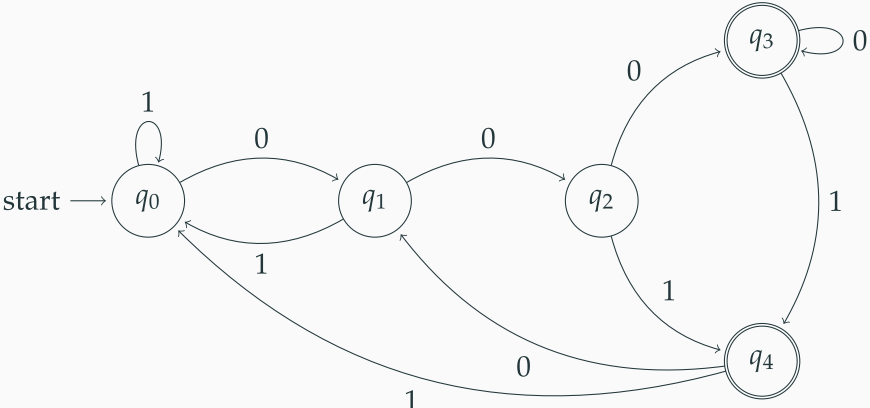
$$L = \{x00y \mid x \in \{0,1\}^* \text{ und } y \in \{0,1\}\}$$

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



> 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 5 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.

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 es existiert kein EA mit < 5 Zuständen für L.

$\blacksquare$

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)

Midterm Prep - Aufgaben HS18

Aufgabe 1.a - HS18

Wir betrachten die Sprache

$$L = \{1x \mid x = y1 \text{ für ein } y \in \{0,1\}^* \text{ oder } x = z00 \text{ für ein } z \in \{0,1\}^*\}$$

Konstruieren Sie einen NEA (in graphischer Darstellung) mit höchstens 4 Zuständen, der L akzeptiert, und beschreiben Sie informell die Idee Ihrer Konstruktion.

Aufgabe 1.b - HS18

Wir betrachten die Sprache

$$L = \{1x \mid x = y1 \text{ für ein } y \in \{0,1\}^* \text{ oder } x = z00 \text{ für ein } z \in \{0,1\}^*\}$$

Konstruieren Sie einen (det.) EA (in graphischer Darstellung) mit höchstens 6 Zuständen, der L akzeptiert.
Sie dürfen hierfür entweder die Potenzmengenkonstruktion auf Ihren NEA aus (a) anwenden oder den Automaten direkt konstruieren und informell die Idee Ihrer Konstruktion beschreiben.

Zeigen Sie, dass die Sprache

$$L = \{u\#v \mid u, v \in \{0,1\}^* \text{ und Nummer}(v) = 2 \cdot \text{ Nummer}(u)\}$$

nicht regulär ist, mit einer der 3 Methoden der Vorlesung.

Aufgabe 2a HS18 - Lemma 3.3

Nehmen wir zum Widerspruch an L sei regulär.

Dann existiert ein EA $A = (Q, \Sigma, \delta, q_0, F)$ mit $L(A) = L$.

Wir betrachten die Wörter $10^1, \ldots, 10^{|Q|+1}$. Per Pigeonhole-Principle existiert O.B.d.A. $i < j$, so dass

$$\hat{\delta}(q_0, 10^i) = \hat{\delta}(q_0, 10^j)$$

Nach Lemma 3.3 gilt

$$10^i z \in L \iff 10^j z \in L$$

für alle $z \in (\Sigma_{\text{bool}})^*$.

Dies führt aber zu einem Widerspruch, weil für $z = \#10^{i+1}$ das Wort $10^i\#10^{i+1} \in L$ aber $10^j\#10^{i+1} \notin L$, da

$2\cdot$ Nummer(10^i) =Nummer(10^{i+1}) = $2^{i+1} < 2 \cdot 2^j = 2 \cdot$ Nummer(10^j).

Aufgabe 2a HS18 - Pumping Lemma

Wir nehmen zum Widerspruch an, dass L regulär ist.

Wir wählen $w = 1^{n_0}\#1^{n_0+1}$. Da $|w| = 2n_0 + 3 \geq n_0$ gilt das Pumping Lemma.

Sei y, x, z die Zerlegung die (i)-(iii) nach dem PL erfüllt. Da nach (i), $|yx| \leq n_0$ gelten muss, haben wir $y = 1^l$ und $x = 1^m$ für $l, m \in \mathbb{N}, l + m \leq n_0$.

Somit gilt $z = 1^{n_0-(l+m)}\#1^{n_0+1}$.

Nach (ii) ist $m \geq 1$.

Wir haben für $k = 1$, $yxz = w \in L$.

Aber für $k = 3$ gilt $yx^3z = 1^{n_0+2m}\#1^{n_0+1} \notin L$, da

Nummer(1^{n_0+1}) = $2^{n_0+1} - 1 < 2^{n_0+2m} - 1 < 2 \cdot (2^{n_0+2m} - 1) = 2 \cdot$ Nummer(1^{n_0+2m}).

Dies ist ein Widerspruch zu (iii) und somit ist L nicht regulär.

Aufgabe 2b - HS18

Zeigen Sie, dass die Sprache

$$L = \{0^{n^3} \mid n \in \mathbb{N}\}$$

nicht regulär ist, mit einer anderen der 3 Methoden der Vorlesung.

Aufgabe 3a - HS18

Sei für alle $n \in \mathbb{N} \setminus \{0\}$ die Sprache L_n definiert durch

$$L_n = \{x \in \{0,1\}^* \mid |x|_1 \geq n\}$$

Geben Sie einen (det.) EA (in graphischer Darstellung) für L_4 an, der höchstens 5 Zustände hat, und geben Sie für jeden Zustand q Ihres Automaten die Klasse $\mathrm{Kl}[q]$ an.

Aufgabe 3b - HS18

Zeigen Sie, dass jeder (det.) EA, der L_n akzeptiert, mindestens $n+1$ Zustände hat.

Appendix - Theory Recap

Entscheidungsproblem

Das **Entscheidungsproblem** (Σ, L) für ein gegebenes Alphabet Σ und eine gegebene Sprache $L \subseteq \Sigma^*$ ist, für jedes $x \in \Sigma^*$ zu entscheiden, ob

$$x \in L \text{ oder } x \notin L.$$

Ein Algorithmus A **löst** das Entscheidungsproblem (Σ, L), falls für alle $x \in \Sigma^*$ gilt:

$$A(x) = \begin{cases} 1, & \text{falls } x \in L, \\ 0, & \text{falls } x \notin L. \end{cases}$$

Wir sagen auch, dass A die Sprache L erkennt.

Sei Σ ein Alphabet und $x \in \Sigma^*$. Wir sagen, dass ein Algorithmus A das Wort x **generiert**, falls A für die Eingabe λ die Ausgabe x liefert.

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.

$\Longrightarrow$ Kolmogorov Komplexität eines Wortes ist die Länge des kürzesten Programms, dass keinen Input nimmt und das Wort ausgibt!

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

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

Definition!

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.

■

Ein zufälliges Wort

Definition

Ein Wort $x \in (\Sigma_{\text{bool}})^*$ heisst **zufällig**, falls $K(x) \geq |x|$.
Eine Zahl n heisst **zufällig**, falls $K(n) = K(\text{Bin}(n)) \geq \lceil \log_2(n+1) \rceil - 1$.

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.

Gehe Wörter z_i in kanonischer Reihenfolge durch, verwende A_L um zu entscheiden, ob z_i in L ist.

Primzahlsatz

Für jede positive ganz Zahl n sei 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 $n_1, n_2, n_3, \dots$ 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.

Lemma 2.6 - Beweis

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

$$n_i = p_1^{r_{i,1}} \cdot p_2^{r_{i,2}} \cdot \dots \cdot p_m^{r_{i,m}}$$

für irgendwelche $r_{i,1}, r_{i,2}, \dots, r_{i,m} \in \mathbb{N}$ darstellen.

Bemerke das die p_i ausser p_m nicht notwendigerweise in Q sein müssen, wir verwenden nur den Fakt, dass es endlich viele davon hat.

Lemma 2.6 - Beweis continued

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 $r_{i,1}, \dots, r_{i,m}$ gleich).

Dann gilt:

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

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

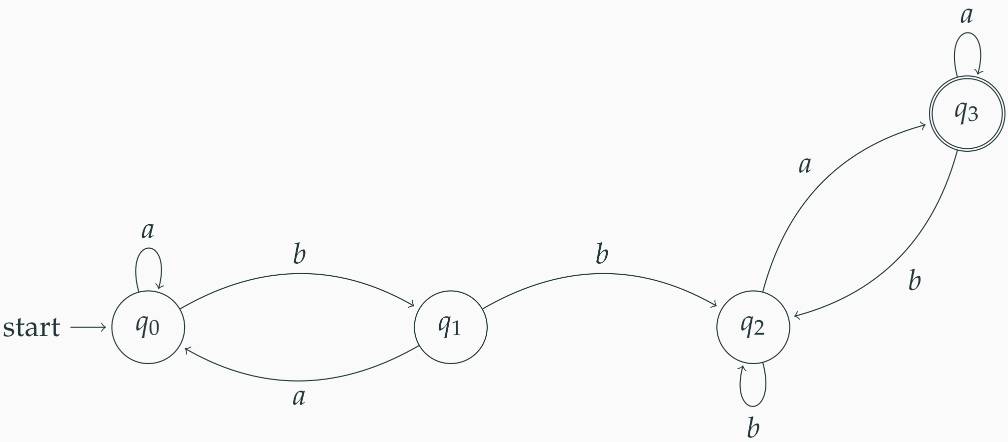
$\blacksquare$

EA Konstruktion - Beispielaufgabe

Entwerfen sie für folgende Sprache einen Endlichen Automat und geben Sie eine Beschreibung von $\mathrm{Kl}[q]$ 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:

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

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

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

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

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

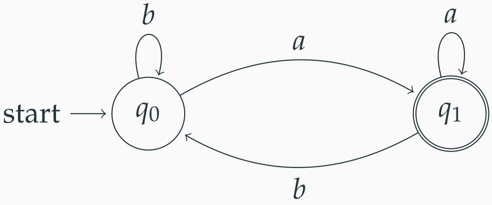
Wir teilen L wie folgt auf:

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

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

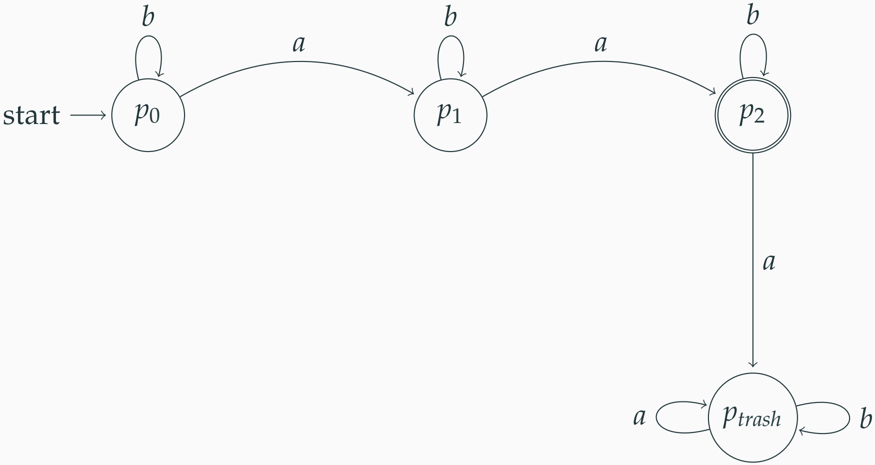


$\mathrm{Kl}[q_0] = \{yb \mid y \in \{a,b\}^*\} \cup \{\lambda\}$

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

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}:

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

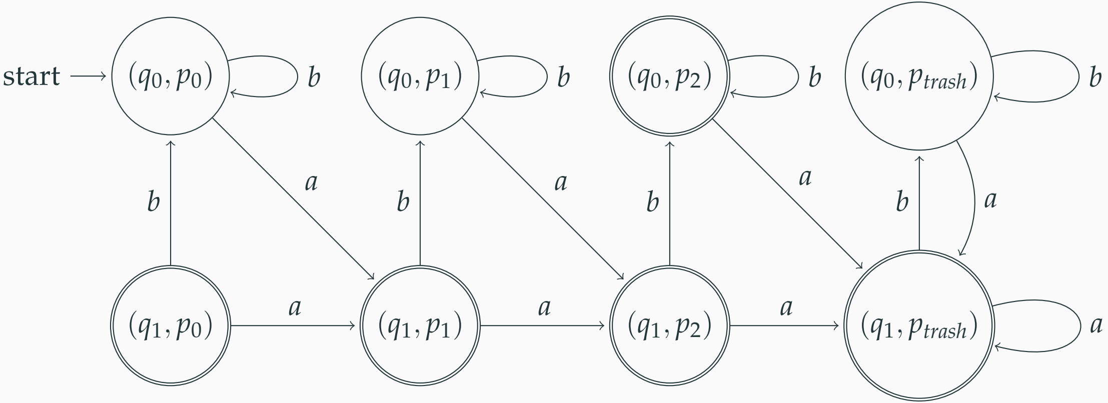
$\mathrm{Kl}[p_1] = \{w \in \{a,b\}^* \mid |w|_a = 1\}$

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

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

Zum Schluss kombinieren wir diese Teilautomaten zu einem Produktautomaten:

Produktautomat: $L = L_1 \cup L_2$



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

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

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 $xz, yz \notin L(A)$.

$\blacksquare$

Bemerkungen

- Von den 3 vorgestellten Methoden, ist diese Methode die einzige, die (unter der richtigen Anwendung) garantiert für jede nichtreguläre Sprache funktioniert.
- Um die Nichtregularität von L zu beweisen, verwenden wir die Endlichkeit von Q und das Pigeonhole-Principle.

Betrachten wir mal eine Beispielaufgabe mit dieser Methode am Paradebeispiel

$$L = \{0^n 1^n \mid n \in \mathbb{N}\}$$

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 $w_1 w_2 ... w_{n_0}$:

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

Theorie für Nichtregularitätsbeweise - Pumping Lemma

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

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

Dies impliziert

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

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 $w = yxz$.

Theorie für Nichtregularitätsbeweise - Pumping Lemma

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.

Beweis TODO. Der Beweis find ich auch relevant, auch wenn nicht aufgeschrieben.

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.

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.

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

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

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 $k+1$ Zuständen akzeptiert L_k.

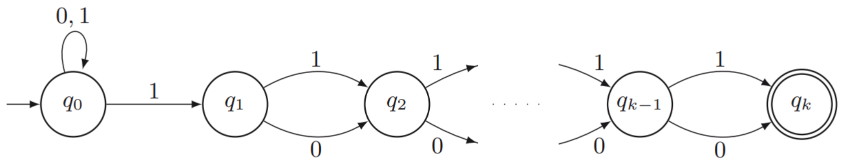


Abbildung 1: Abb. 3.19 im Buch

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 $x = x_1 x_2 ... x_k$ und $y = y_1 y_2 ... y_k$ 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

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

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. ■

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.

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.

Äquivalenz von Maschinen (TM, MTM)

Seien A und B zwei Maschinen mit **gleichem** Σ.
Wir sagen, dass **A äquivalent zu B ist**, wenn für jede Eingabe $x \in \Sigma^*$

(i) A akzeptiert $x \iff B$ akzeptiert x

(ii) A verwirft $x \iff B$ verwirft x

(iii) A arbeitet unendlich lange auf $x \iff B$ arbeitet unendlich lange auf x

Wir haben

$$A \text{ und } B \text{ äquivalent} \implies L(A) = L(B)$$

aber

$$L(A) = L(B) \not\Longrightarrow A \text{ und } B \text{ äquivalent}$$

da A auf x unendlich lange arbeiten könnte, während B x verwirft.

Äquivalenz von TM zu k-Band-TM

Lemma 4.2

Zu jeder Mehrband-TM A existiert eine zu A äquivalente TM B

Beweisidee

Vergrösserung des Alphabets, jedes Zeichen enthält jetzt $2(k+1)$ Zeichen.

B simuliert A einen Schritt von A indem es den ganzen Inhalt liest und dann durch die endliche Kontrolle von A jede Schreib und Bewegungsoperation einzeln ausführt.

Dies verwendet immer nur **endlich** viele Schritte um einen Schritt von A zu simulieren.

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.

$$\begin{aligned} \text{Konfiguration akzeptierend} &\iff \text{enthält } q_{\text{accept}} \\ \text{Konfiguration verwerfend} &\iff \text{enthält } q_{\text{reject}} \end{aligned}$$

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 mit zwei Bändern, wobei das erste Band alle Konfigurationen der besuchten Schicht speichert und das zweite alle Konfigurationen der nächsten Schicht.

Wenn eine akzeptierende Konfiguration erreicht wird, dann akzeptiert A. Wenn keine weitere Konfiguration erreichbar ist, dann verwirft A (eine verwerfende Konfiguration wird ohne weiteres normal behandelt).

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

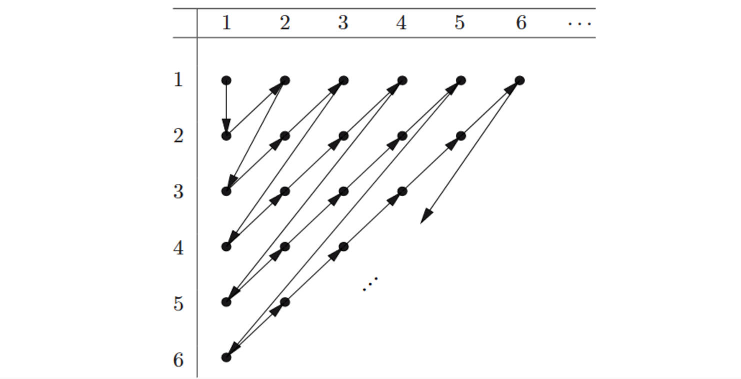


Abbildung 2: Abbildung 5.3 im Buch

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 $\mathbb{N} \setminus \{0\}$.

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}	$\ldots$		
2	0.	a_{21}	$\boxed{a_{22}}$	a_{23}	a_{24}	$\ldots$		
3	0.	a_{31}	a_{32}	$\boxed{a_{33}}$	a_{34}	$\ldots$		
4	0.	a_{41}	a_{42}	a_{43}	$\boxed{a_{44}}$	$\ldots$		
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$		$\ldots$		
i	0.	a_{i1}	a_{i2}	a_{i3}	a_{i4}	$\ldots$	$\boxed{a_{ii}}$	$\ldots$
$\vdots$	$\vdots$							

Abbildung 5.5

Überabzählbarkeit

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:

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

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:

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

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.

$\blacksquare$

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

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

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

$\blacksquare$