

Algorithmen und Datenstrukturen

Theoretische Grundlagen der Informatik

Prof. Dr. rer. nat. Jörg Striegnitz

Fachhochschule Aachen
striegnitz@fh-aachen.de

29. April 2024

Algorithmen und Datenstrukturen

Äquivalenz und Grenzen der Modelle

Prof. Dr. rer. nat. Jörg Striegnitz

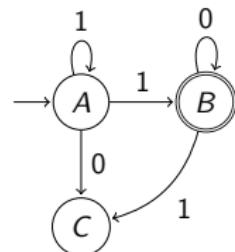
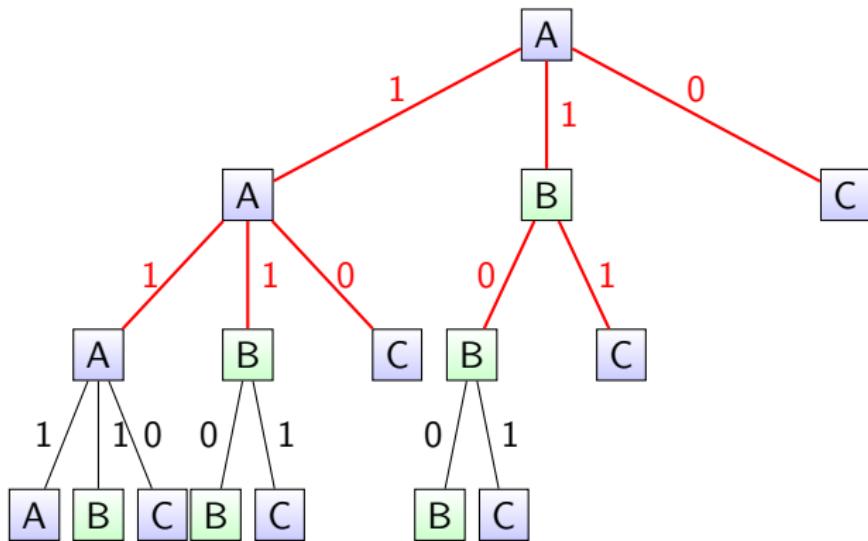
Fachhochschule Aachen
striegnitz@fh-aachen.de

29. April 2024

Motivation

»Entscheidungsbaum«

NEA



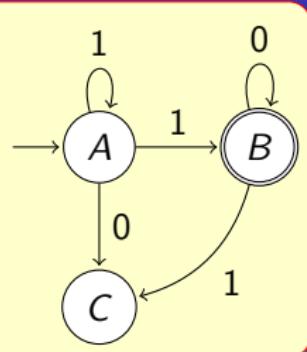
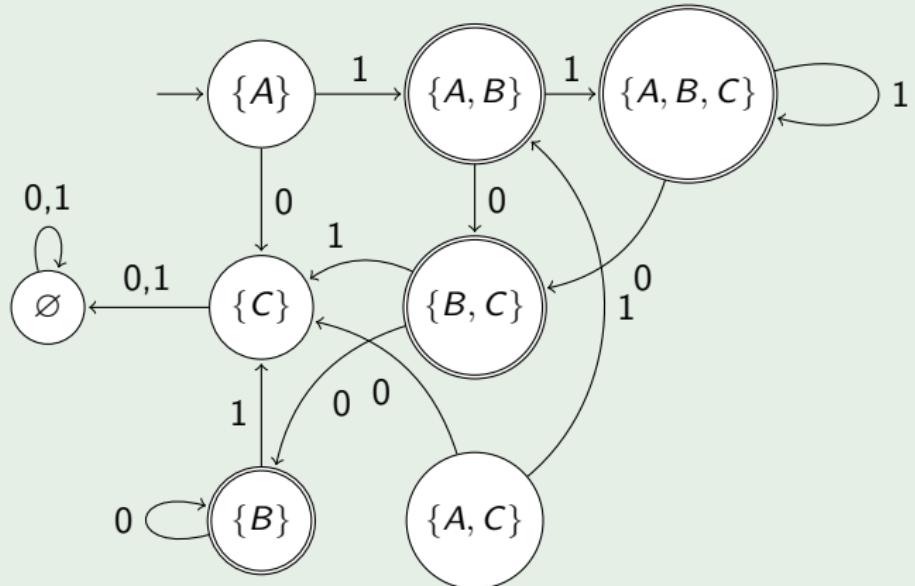
Σ^*	$\mathcal{P}(Q)$
0	{C}
1	{A, B}
10	{B, C}
11	{A, B, C}
:	:



Offenbar reicht Buchführung über Zustände, die NEA erreichen könnte ...

Von NEA zu DEA: die Potenzmengenkonstruktion

Beispiel 1.50



- Konstruktion kann zu unerreichbaren Zuständen führen (hier: $\{A, C\}$).
- Besser: nur erreichbare Zustände erzeugen.

Definition 1.56 (Potenzmengenkonstruktion)

Sei $M_N = (Q_N, \Sigma, \delta_N, q_0, F_N)$ ein NEA. Wir konstruieren aus M_N einen DEA $M_D = (Q_D, \Sigma, \delta_D, \{q_0\}, F_D)$ wie folgt:

- $Q_D = \mathcal{P}(Q_N)$

Beachte: Zustandszahl wächst **exponentiell!**

- $F_D = \{S \in Q_D \mid S \cap F_N \neq \emptyset\}$

Anmerkung: $S \in Q_D$ ist genau dann ein akzeptierender Zustand, wenn es mindestens ein Element $q \in S$ gibt, welches akzeptierender Zustand von M_N ist ($q \in F_N$).

- Für jeden Zustand $S \in Q_D$ und für jedes Eingabesymbol $a \in \Sigma$ gilt

$$\delta_D(S, a) = \bigcup_{q \in S} \delta_N(q, a)$$

Man sagt M_D entsteht aus M_N durch Anwendung der **Potenzmengenkonstruktion (PMK)**.

Korrektheit der Konstruktion 1

Satz 1.1

Sei $M_N = (Q_N, \Sigma, \delta_N, q_0, F_N)$ ein NEA und $M_D = \{Q_D, \Sigma, \delta_D, \{q_0\}, F_D\}$ ein DEA, der durch Potenzmengenkonstruktion aus M_N entstanden. Dann gilt: $L(M_N) = L(M_D)$.

Beweis.

Wir beweisen zunächst

$$\hat{\delta}_D(\{q_0\}, w) = \hat{\delta}_N(q_0, w) \quad (1)$$

denn dann folgt sofort:

$$\begin{aligned} w \in L(M_N) &\Leftrightarrow \exists q_f \in F_N : (q_0, w) \xrightarrow[M_N]{*} (q_f, \varepsilon) \\ &\Leftrightarrow \exists q_f \in F_N : q_f \in \hat{\delta}_N(q_0, w) \\ &\Leftrightarrow \exists q_f \in F_N : q_f \in \hat{\delta}_D(\{q_0\}, w) \\ \text{Gl. 1} & \\ &\Leftrightarrow \hat{\delta}_D(\{q_0\}, w) \cap F_N \neq \emptyset \\ &\Leftrightarrow \hat{\delta}_D(\{q_0\}, w) \in F_D \\ &\Leftrightarrow w \in L(M_D) \end{aligned}$$

Korrektheit der Konstruktion 2

Beweis.

Induktionsanfang: $w = \varepsilon$; dann gilt $\hat{\delta}_N(q_0, \varepsilon) = \{q_0\} = \hat{\delta}_D(\{q_0\}, \varepsilon)$.

Induktionsschritt: Betrachte $w = ua$ ($u \in \Sigma^*, a \in \Sigma$). Induktionsvoraussetzung:
 $\hat{\delta}_N(q_0, u) = \hat{\delta}_D(\{q_0\}, u)$. Angenommen es sei

$$\hat{\delta}_N(q_0, u) = \hat{\delta}_D(\{q_0\}, u) = \{q_1, q_2, \dots, q_k\}.$$

Aus der induktiven Definition von $\hat{\delta}_N$ wissen wir:

$$\hat{\delta}_N(q_0, w) = \bigcup_{q \in \hat{\delta}_N(q_0, u)} \delta_N(q, a) = \bigcup_{i=1}^k \delta_N(q_i, a)$$

Die Potenzmengenkonstruktion liefert uns andererseits:

$$\delta_D(\{q_1, \dots, q_k\}, a) = \bigcup_{i=1}^k \delta_N(q_i, a)$$

Korrektheit der Konstruktion 3

Beweis.

Jetzt nutzen wir die Induktionsvoraussetzung ($\hat{\delta}_D(\{q_0\}, u) = \{q_1, \dots, q_n\}$) und die induktive Definition von $\hat{\delta}_D$:

$$\begin{aligned}\hat{\delta}_D(\{q_0\}, ua) &= \underset{\text{Def. } \hat{\delta}_D}{\delta_D}(\hat{\delta}_D(\{q_0\}, u), a) \\ &= \underset{\text{IV}}{\delta_D}(\delta_N(\{q_0\}, u), a) \\ &= \delta_D(\{q_1, \dots, q_n\}, a) \\ &= \underset{\text{PMK}}{=} \bigcup_{i=1}^k \delta_N(q_i, a) \\ &= \underset{\text{Def. } \hat{\delta}_N}{=} \hat{\delta}_N(q_0, w)\end{aligned}$$



- Bei der PMK wächst die Zahl der erreichbaren Zustände im ungünstigsten Fall exponentiell:

$$|Q| = n \Leftrightarrow |\mathcal{P}(Q)| = 2^n$$

- Zustandszahl \cong Speicherbedarf \Rightarrow **Speicherbedarf steigt**
- DEA benötigt jedoch weniger Rechenschritte (im deterministischen Vergleich)
 \Rightarrow **Laufzeit sinkt**



Dem Phänomen, dass man Laufzeit durch erhöhten Speichereinsatz steigern kann, werden wir wieder begegnen ...



Wir haben

- drei Modelle zur formalen Beschreibung von Algorithmen zur Lösung von Entscheidungsproblemen kennengelernt (DEA, NEA und ε -NEA) ;
- bewiesen, dass alle Modelle die gleiche Ausdrucksstärke besitzen;
- erfahren, dass ε -NEA und NEA u.U. erlauben, Algorithmen »leichter« zu beschreiben.

Neue Fragestellung

Gibt es Sprachen, die nicht regulär sind?

- **Aktuelles Ziel:** minimales und vollständiges Modell, mit dem wir alle berechenbaren Funktionen beschreiben können
- **Frage:** Liefern endliche Automaten geeignetes Modell?
- Betrachten wir die Sprache

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

- Offenbar kann man leicht einen Algorithmus finden, der entscheidet, ob $w \in L$ oder nicht.
- Allerdings gibt es **keinen** endlichen Automaten, der das Entscheidungsproblem $\langle \Sigma_{Bool}, L \rangle$ löst.
- Drei Werkzeuge, um dies zu beweisen (hier nicht behandelt):
 - Pumping-Lemma
 - Abschlusseigenschaften
 - Kolmogorov-Komplexität (hier nicht behandelt)

- **Aktuelles Ziel:** minimales und vollständiges Modell, mit dem wir alle berechenbaren Funktionen beschreiben können
- **Frage:** Liefern endliche Automaten geeignetes Modell?
- Betrachten wir die Sprache

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

Aufgabe

Können Sie ein Computerprogramm in JAVA, C++, Python oder einer anderen Sprache erstellen, welches L entscheidet?



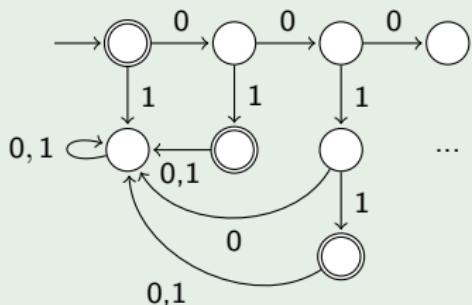
Nein! Man muss das zu untersuchende Wort nur so exorbitant lang machen, dass Ihnen der Speicher ausgeht

Beispiel: Nicht-reguläre Sprache

Beispiel 1.51 (Versuch: DEA für $L = \{0^n1^n \mid n \in \mathbb{N}\}$)

Strategie: Speichere, wie viele Nullen bereits gelesen wurden; für jede mögliche Anzahl von Nullen benötigt man einen eigenen Zustand.

Konsequenz: unendliche Zustandszahl - kein gültiger DEA:



- Strategie war nicht erfolgreich - wie kann man beweisen, dass **keine** Strategie zum Erfolg fhrt?

- **Annahme:** es gibt einen DEA mit n Zuständen (hier exemplarisch 8)
- Für ein Wort $w \in L$ mit $w \geq |Q|$ wird mind. 1 Schleife durchlaufen
- Drei Möglichkeiten, wo **erste Schleife** liegen kann:
 1. im 0er Block 0 0 0 0 0 0 1 1 1 1 1 1 1
Dann würde auch $00(000)^m 0111111$ akzeptiert - für $m \neq 1$ nicht in L !
 2. im 1er Block 0 0 0 0 0 0 1 1 1 1 1 1 1
Dann würde auch $000000(11)^m 1111$ akzeptiert - für $m \neq 1$ nicht in L !
 3. Auf der Grenze 0 0 0 0 0 0 0 1 1 1 1 1 1
Dann würde auch $00000(011)^m 1111$ akzeptiert - für $m \neq 1$ nicht in L !

!

Automat würde Wörter erkennen, die nicht zu L gehören!