Theoretische Informatik 1

WiSe 23/24

Prof. Dr. Sebastian Siebertz
AG Theoretische Informatik
MZH, Raum 3160
siebertz@uni-bremen.de

Universität
Bremen

Entscheidungsprobleme

- Endliche Automaten können auf verschiedene Weise in einer konkreten Anwendung eingesetzt werden.
- Die wichtigste Rolle spielen die folgenden Probleme:
  - das *Wortproblem:*

    Gegeben ein endlicher Automat $\mathcal{A}$ und eine Eingabe $w \in \Sigma^*$ für $\mathcal{A}$, entscheide ob $w \in L(\mathcal{A})$.
  - das *Leerheitsproblem:*

    Gegeben ein endlicher Automat $\mathcal{A}$, entscheide ob $L(\mathcal{A}) = \varnothing$.
  - das *Äquivalenzproblem:*

    Gegeben endliche Automaten $\mathcal{A}_1$ und $\mathcal{A}_2$,
    entscheide ob $L(\mathcal{A}_1) = L(\mathcal{A}_2)$.
- Jedes Problem gibt es in zwei Varianten: für NEAs und für DEAs.
- Unser Ziel: möglichst effiziente Algorithmen für diese Probleme finden.

Laufzeitanalyse

- Laufzeit eines Algorithmus A auf Eingabe x ist die *Anzahl elementarer Rechenschritte*, die A gestartet auf x ausführt (Zuweisungen, Additionen, Multiplikationen, etc.)
- Wir messen die Laufzeit *in Abhängigkeit von der Länge n der Eingabe x*, abstrahiert von konkreter Eingabe.
- Beschreibung also durch Funktion

$$f(n): \mathbb{N} \to \mathbb{N}.$$

Eingabelänge

Maximale Anzahl Rechenschritte auf Eingaben dieser Länge

Polynomielle vs. super-polynomielle Laufzeit

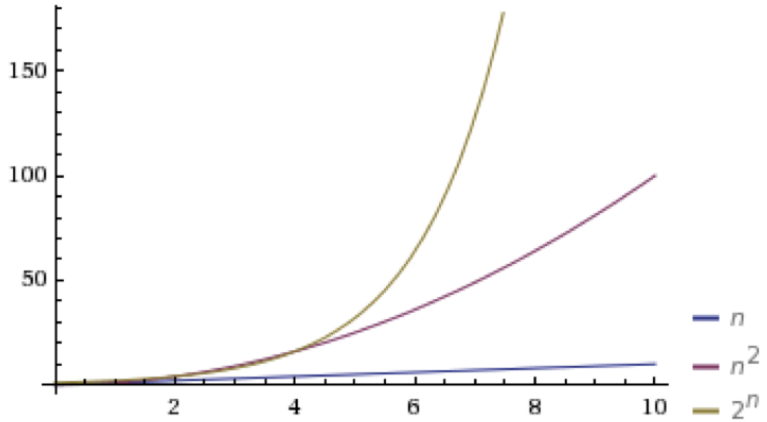
- Die Grenze zwischen *effizient* und *ineffizient*:
  - *effizient: polynomielle Laufzeit*: Funktion $f(n)$ ist Polynom (beliebigen festen Grades),

    z.B. n^2, $5n^8$, ...
  - *ineffizient: super-polynomielle Laufzeit*: Funktion $f(n)$, so dass

$$\limsup_{n\to\infty} \frac{f(n)}{p(n)} = \infty$$

    für jedes Polynom p,

    z.B. $2^{\sqrt{n}}$, 6^{3n}, n^n, ...



$\mathcal{O}$-Notation

- Bei Laufzeitanalyse abstrahieren wir von konkreten Konstanten, d.h. wir unterscheiden nicht zwischen z.B. $3n^2$ und $5n^2$.

Definition

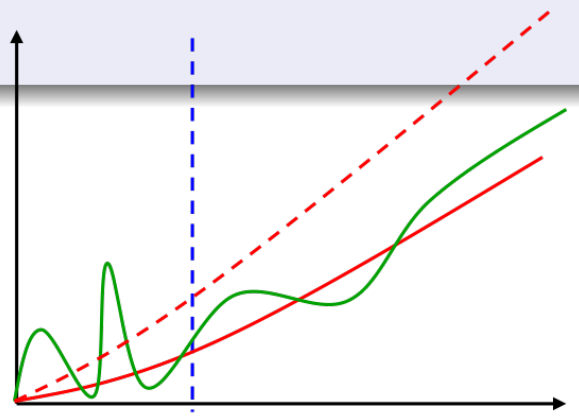
Seien f und g Funktionen von $\mathbb{N}$ nach $\mathbb{N}$. Wir schreiben

$$f \in \mathcal{O}(g)$$

wenn es $c > 0$ und $n_0 \geq 0$ gibt, so dass $f(n) \leq c \cdot g(n)$ für alle $n > n_0$.

Äquivalent: $\limsup_{n\to\infty} \frac{f(n)}{g(n)} < \infty$.

- $f \in \mathcal{O}(g)$ Intuitiv: f wächst nicht wesentlich schneller als g.



$\mathcal{O}$-Notation

- Laufzeit $\mathcal{O}(f(n))$ heißt also intuitiv: *Laufzeit höchstens $f(n)$, bis auf Konstanten.*
- Insbesondere beschreibt
  - $\mathcal{O}(n)$ *Linearzeit*
  - $\mathcal{O}(n^2)$ *quadratische Zeit*
  - $\bigcup_{c \geq 1} \mathcal{O}(n^c)$ *polynomielle Zeit*.

Das Wortproblem

Definition Wortproblem

- *Gegeben:* DEA oder NEA $\mathcal{A}$ und eine Eingabe $w \in \Sigma^*$ für $\mathcal{A}$.
- *Frage:* Gilt $w \in L(\mathcal{A})$?

- Wenn $\mathcal{A} = (Q, \Sigma, q_s, \delta, F)$ DEA:
- Berechne Zustand $q = \hat{\delta}(q_s, w)$ durch wiederholte Anwendung von δ. Überprüfe dann, ob $q \in F$.
- Laufzeitanalyse:
  - δ muss $|w|$-mal angewendet werden.
  - Jede Anwendung braucht max. $\mathcal{O}(|\delta|)$ Zeit.

Satz

Das Wortproblem für DEAs ist in Zeit $\mathcal{O}(|w| \cdot |\delta|)$ entscheidbar.

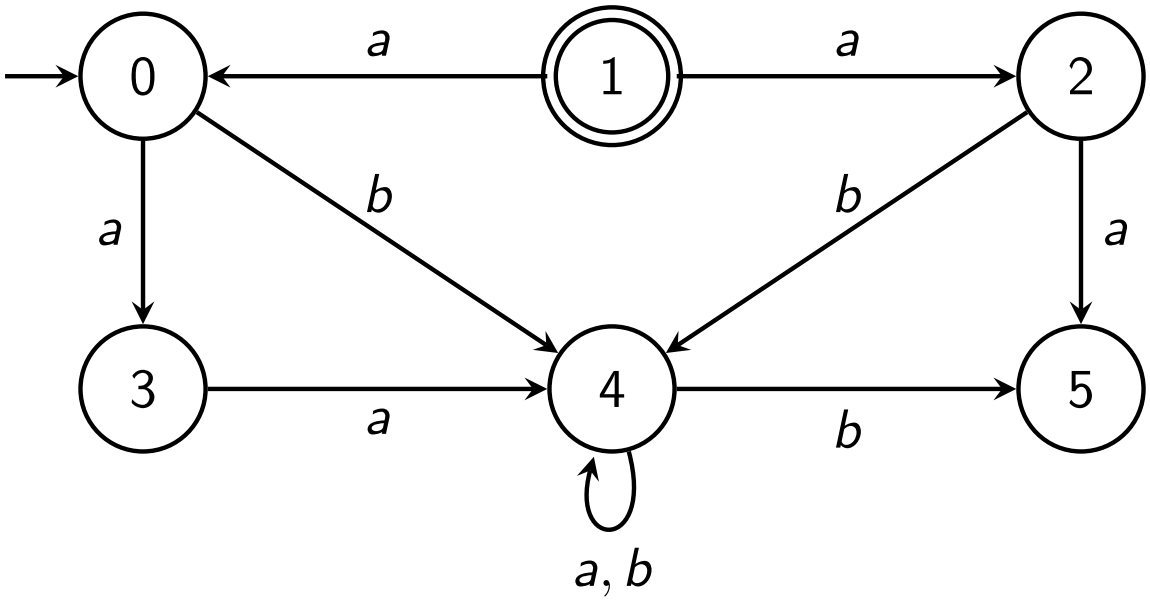
Das Wortproblem

- Für einen NEA ist dies nicht so einfach, da es für eine Eingabe w mehrere Läufe durch den NEA geben kann.
- Naive Ansätze führen zu schlechter Laufzeit:
  - Alle Läufe durchprobieren:
    Im schlimmsten Fall $|Q|^{|w|}$ viele, also *exponentielle Laufzeit*.
  - Erst Potenzmengenkonstruktion anwenden:
    Resultierender DEA hat $2^{|Q|}$ viele Zustände, also $|\delta| \sim 2^{\mathcal{O}(|Q| \cdot |\Sigma|)}$. Insgesamt *exponentielle Laufzeit.*
- Wir werden das Problem auf das Leerheitsproblem reduzieren!

Das Leerheitsproblem

Definition Leerheitsproblem

- *Gegeben:* NEA $\mathcal{A}$.
- *Frage:* Gilt $L(\mathcal{A}) = \varnothing$?



- $L(\mathcal{A}) \neq \varnothing$ gdw. es einen (beliebig beschrifteten) Lauf von q_s in $\mathcal{A}$ zu einem akzeptierenden Zustand gibt.

Das Leerheitsproblem

- Algorithmus für Leerheit von NEA $\mathcal{A} = (Q, \Sigma, q_s, \Delta, F)$:
  1. Berechne eine Folge von Zustandsmengen $P_0, P_1, \ldots$ wie folgt:

$$P_0 := \{q_s\}$$
$$P_{i+1} := P_i \cup \{q \in Q \mid \exists p \in P_i \ \exists a \in \Sigma : (p, a, q) \in \Delta\}.$$

     Stoppe sobald $P_{i+1} = P_i$.
  2. Antworte „ja“, wenn $P_i \cap F = \varnothing$, und „nein“ sonst.

Lemma

Der Algorithmus terminiert nach maximal $|Q|$ Iterationen und gibt „ja“ zurück gdw. $L(\mathcal{A}) = \varnothing$.

Das Leerheitsproblem

- Laufzeitanalyse.
  - Der Algorithmus stoppt nach $|Q|$ Iterationen.
  - Jede Iteration braucht bei naiver Implementierung Zeit $\mathcal{O}(|Q| \cdot |\Delta|)$: laufe über alle Zustände $p \in P_i$, für jeden solchen Zustand laufe über Δ und suche alle Übergänge (p, a, q).
  - Insgesamt also $\mathcal{O}(|Q|^2 \cdot |\Delta|)$.
  - Durch geschicktere Implementierung (Breitensuche in Graphen) können wir die Laufzeit verbessern auf Zeit $\mathcal{O}(|Q| + |\Delta|)$, also *Linearzeit*.

Satz

Das Leerheitsproblem für NEAs ist in Zeit $\mathcal{O}(|Q| + |\Delta|)$ entscheidbar.

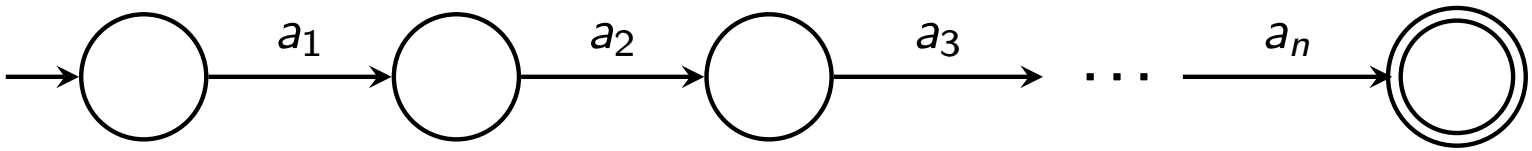
Das Wortproblem für NEAs

Satz

Das Wortproblem für NEAs ist in Zeit $\mathcal{O}(|w| \cdot (|Q| + |\Delta|))$ entscheidbar.

Beweis.

- Konstruiere zunächst einen Automaten $\mathcal{A}_w$, der genau das Wort $w = a_1 \cdots a_n$ akzeptiert:



- Dieser Automat hat $|w| + 1$ Zustände.
- Es gilt

$$w \in L(\mathcal{A}) \quad \Leftrightarrow \quad L(\mathcal{A}) \cap L(\mathcal{A}_w) \neq \varnothing.$$

- (Reduktion des Wortproblems auf das Leerheitsproblem)

Das Wortproblem für NEAs

- Algorithmus um zu entscheiden, ob $w \in L(\mathcal{A})$:
  - Konstruiere den Produktautomaten zu $\mathcal{A}$ und $\mathcal{A}_w$.
  - Prüfe, ob dieser eine nicht-leere Sprache erkennt.
  - Laufzeit: $\mathcal{O}(|Q'| + |\Delta'|)$ wobei $|Q'|$ Zahl der Zustände und $|\Delta'|$ Größe der Übungsgangsrelation des Produktautomaten.
- Größe des Produktautomaten:
  - Anzahl der Zustände: $|Q| \cdot (|w| + 1)$
  - Anzahl der Übergänge:
    Da $\mathcal{A}_w$ genau $|w|$ Übergänge hat, hat der Produktautomat höchstens $|w| \cdot |\Delta|$ Übergänge.

Das Wortproblem für NEAs

- Laufzeitanalyse:
  - Aufwand zum Testen von $L(\mathcal{A}) \cap L(\mathcal{A}_w) \neq \varnothing$ also:

$$\mathcal{O}(|Q| \cdot (|w| + 1) + |w| \cdot |\Delta|) \;=\; \mathcal{O}(|w| \cdot (|Q| + |\Delta|)).$$

  - Konstruktion des Produktautomaten: $\mathcal{O}(|w| \cdot (|Q| + |\Delta|))$.
  - Gesamtlaufzeit:

$$2 \cdot \mathcal{O}(|w| \cdot (|Q| + |\Delta|)) \;=\; \mathcal{O}(|w| \cdot (|Q| + |\Delta|)).$$

Das Äquivalenzproblem

Definition Äquivalenzproblem

- *Gegeben* DEAs oder NEAs $\mathcal{A}_1$ und $\mathcal{A}_2$.
- *Frage:* Gilt $L(\mathcal{A}_1) = L(\mathcal{A}_2)$?

- Reduktion auf das Leerheitsproblem:

$$L_1 = L_2 \quad \Leftrightarrow \quad (L_1 \cap \overline{L_2}) \cup (L_2 \cap \overline{L_1}) = \varnothing.$$

- Vereinigung und Schnitt:
  Konstruktion vergrößert DEAs und NEAs nur *polynomiell*.
- Komplement:
  für DEAs keine Vergrößerung; für NEAs *exponentielle Vergrößerung*

Das Äquivalenzproblem

Satz

Das Äquivalenzproblem für DEAs ist in polynomieller Zeit entscheidbar. Für NEAs ist es in exponentieller Zeit entscheidbar.

- Der exponentielle Zeitaufwand für NEAs lässt sich (wahrscheinlich) nicht vermeiden.
- Das Problem ist PSpace-vollständig und gehört damit zu einer Klasse von Problemen, die wahrscheinlich nicht in polynomieller Zeit lösbar sind (siehe Theoretische Informatik 2).

Übersicht

	Wortproblem	**Leerheitsprob.**	**Äquivalenzprob.**
NEA	$\mathcal{O}(\|w\| \cdot (\|Q\| + \|\Delta\|))$	$\mathcal{O}(\|Q\| + \|\Delta\|)$	Exponentialzeit
DEA	$\mathcal{O}(\|w\| + \|\delta\|)$	$\mathcal{O}(\|Q\| + \|\delta\|)$	Polynomialzeit