§ 3. Abschlußeigenschaften und Entscheidungsprobleme

Endliche Automaten definieren Klasse der erkennbaren Sprachen

Anstatt Eigenschaften einzelner Sprachen zu studieren kann man auch die Eigenschaften ganzer Sprachklassen analysieren.

Wir beweisen hier Abschlußeigenschaften der Klasse der erkennbaren Sprachen, wie zum Beispiel:

Wenn L_1 und L_2 erkennbar sind, dann ist auch $L_1 \cap L_2$ erkennbar.

Derartige Eigenschaften sind sehr nützlich in Konstruktionen und Beweisen,

z.B. um (manchmal!) die Anwendung des Pumping-Lemmas zu vermeiden

Satz 3.1 (Abschlußeigenschaften erkennbarer Sprachen)

Sind L_1, L_2 erkennbar, so auch

1. $L_1 \cup L_2$ (Vereinigung)
2. $\overline{L}_1$ (Komplement)
3. $L_1 \cap L_2$ (Durchschnitt)
4. $L_1 \cdot L_2$ (Konkatenation)
5. L_1^* (Kleene-Stern)

Beweis von Satz 3.1:

Es sei $\mathcal{A}_i = (Q_i, \Sigma, q_{0i}, \Delta_i, F_i)$ ein NEA für L_i $(i = 1, 2)$.
O.B.d.A. sei $Q_1 \cap Q_2 = \emptyset$.

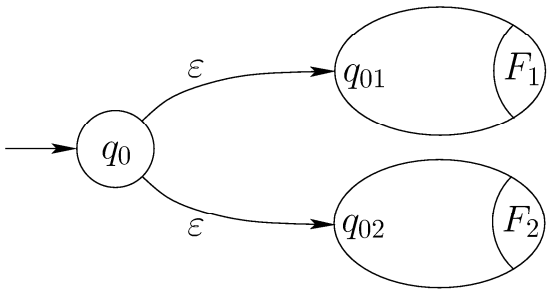
1. Abschluß unter Vereinigung:

Der folgende ε-NEA akzeptiert $L_1 \cup L_2$:

$$\mathcal{A} := (Q_1 \cup Q_2 \cup \{q_0\}, \Sigma, q_0, \Delta, F_1 \cup F_2),$$

wobei $q_0 \notin Q_1 \cup Q_2$ und

$$\Delta := \Delta_1 \cup \Delta_2 \cup \{(q_0, \varepsilon, q_{01}), (q_0, \varepsilon, q_{02})\}.$$



Mit Lemma 1.17 gibt es zu $\mathcal{A}$ einen äquivalenten NEA.

2. Abschluß unter Komplement:

1. Schritt:

Potenzmengenkonstruktion liefert zu $\mathcal{A}_1$ äquivalenten DEA $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$.

2. Schritt:

Vertauschen der Endzustände mit den Nicht-Endzuständen liefert DEA für $\overline{L_1}$:

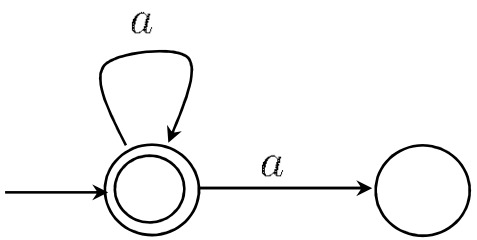
$$\overline{\mathcal{A}} := (Q, \Sigma, q_0, \delta, Q \setminus F).$$

Es gilt nämlich:

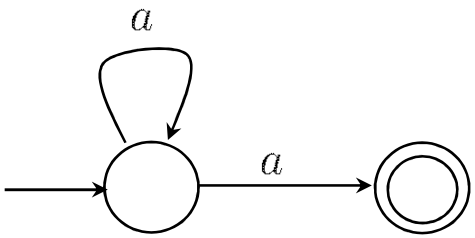
$$\begin{array}{lll} w \in \overline{L} & \text{gdw.} & w \notin L(\mathcal{A}) \\ & \text{gdw} & \delta(q_0, w) \notin F \\ & \text{gdw} & \delta(q_0, w) \in Q \setminus F \\ & \text{gdw} & w \in L(\overline{\mathcal{A}}) \end{array}$$

Beachte: Mit einem NEA funktioniert diese Konstruktion nicht!

z.B.



vs.



3. Abschluß unter Schnitt:

Folgt aus 1) und 2), da

$$L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$$

Es geht aber auch ohne Potenzmengenkonstruktion (ergibt kleineren Automaten: polynomiell statt exponentiell in der Größe von $\mathcal{A}_1$ und $\mathcal{A}_2$)

Der Produktautomat:

$$\mathcal{A} := (Q_1 \times Q_2, \Sigma, (q_{01}, q_{02}), \Delta, F_1 \times F_2)$$

mit

$$\Delta := \{((q_1, q_2), a, (q_1', q_2')) \mid (q_1, a, q_1') \in \Delta_1 \text{ und } (q_2, a, q_2') \in \Delta_2 \}.$$

Ein Übergang in $\mathcal{A}$ ist also genau dann möglich, wenn der entsprechende Übergang in $\mathcal{A}_1$ und $\mathcal{A}_2$ möglich ist.

Es ist nicht schwierig, zu beweisen, dass

$$L(\mathcal{A}) = L_1 \cap L_2.$$

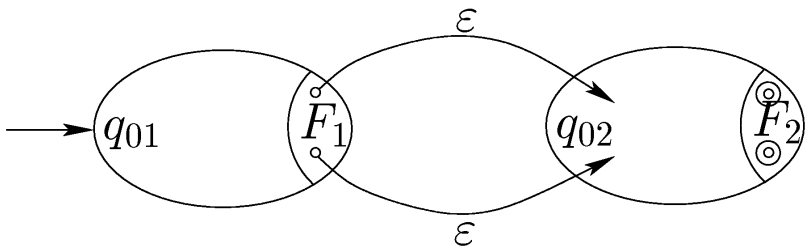
4. Abschluß unter Konkatentation:

Der folgende ε-NEA akzeptiert $L_1 \cdot L_2$:

$$\mathcal{A} := (Q_1 \cup Q_2, \Sigma, q_{01}, \Delta, F_2),$$

wobei

$$\Delta := \Delta_1 \cup \Delta_2 \cup \{(f, \varepsilon, q_{02}) \mid f \in F_1\}.$$



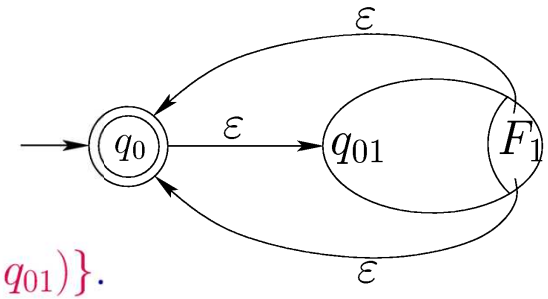
5. Abschluß unter Kleene-Stern:

Der folgende ε-NEA akzeptiert L_1^*:

$$\mathcal{A} := (Q_1 \cup \{q_0\}, \Sigma, q_0, \Delta, \{q_0\}),$$

wobei $q_0 \notin Q_1$ und

$$\Delta := \Delta_1 \cup \{(f, \varepsilon, q_0) \mid f \in F_1\} \cup \{(q_0, \varepsilon, q_{01})\}.$$



Die Automaten für $L_1 \cup L_2$, $L_1 \cap L_2$, $L_1 \cdot L_2$ und L_1^* sind polynomiell in der Größe der Automaten für L_1, L_2.

Beim Komplement kann die Konstruktion exponentiell sein, wenn man von einem NEA ausgeht.

"Trick", mit dem man manchmal das Pumping Lemma vermeiden kann:

Beispiel 3.2 (Abschlußeigenschaften zum Nachweis der Nichterkennbarkeit)

$L = \{a^n b^m \mid n \neq m\}$ ist nicht erkennbar.

Beweis:

Anstatt dies mit dem Pumping-Lemma zu zeigen, kann man auch verwenden, daß bereits bekannt ist, daß

$$L' := \{a^n b^n \mid n \geq 0\}$$

nicht erkennbar ist.

Wäre nämlich L erkennbar, so mit Satz 3.1 auch

$$L' = \overline{L} \cap \{a\}^* \cdot \{b\}^*.$$

Da wir wissen, daß L' nicht erkennbar ist, kann auch L nicht erkennbar sein.

Endliche Automaten können auf verschiedene Weise in einer konkreten Anwendung eingesetzt werden.

Die wichtigste Rolle spielen die folgenden Probleme:

- das Wortproblem:
  gegeben Automat $\mathcal{A}$ und Eingabe w, ist $w \in L(\mathcal{A})$?
- das Leerheitsproblem:
  Gegeben Automat $\mathcal{A}$, ist $L(\mathcal{A}) = \emptyset$?
- das Äquivalenzproblem:
  Gegeben Automaten $\mathcal{A}_1$ und $\mathcal{A}_2$, ist $L(\mathcal{A}_1) = L(\mathcal{A}_2)$?

Jedes Problem gibt es in zwei Varianten: für NEAs und für DEAs!

Unser Ziel: möglichst gute Algorithmen für diese Probleme finden.
Gut bedeutet für den Augenblick: möglichst wenig Rechenschritte

Exkurs Laufzeitanalyse

Grundideen:

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

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

Eingabelänge

Maximal Anzahl Rechenschritte auf Eingaben dieser Länge

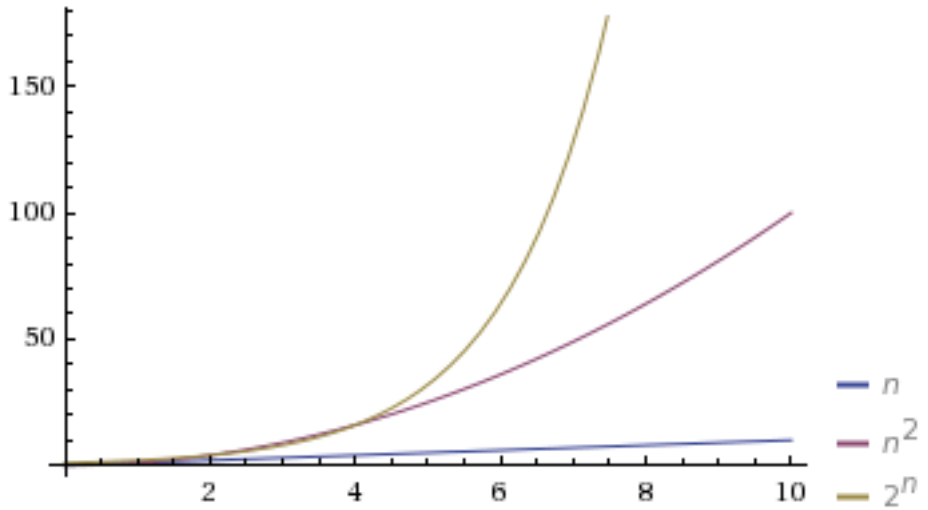
Exkurs Laufzeitanalyse

Die Grenze zwischen effizient und ineffizient wird meist angesetzt bei:

- effizient: polynomielle Laufzeit
  Funktion $f(n)$ ist Polynom (beliebigen Grades): n^2, $5 \cdot n^8$, etc.

- ineffizient: exponentielle Laufzeit
  also Funktionen f der Art 2^n, $6^{3 \cdot n}$, n^n, etc.

Besonders gut ist lineare Laufzeit $c \cdot n$.



Exkurs $\mathcal{O}$-Notation

Bei Laufzeit-Analyse möchte man meist von konkreten Konstanten abstrahieren

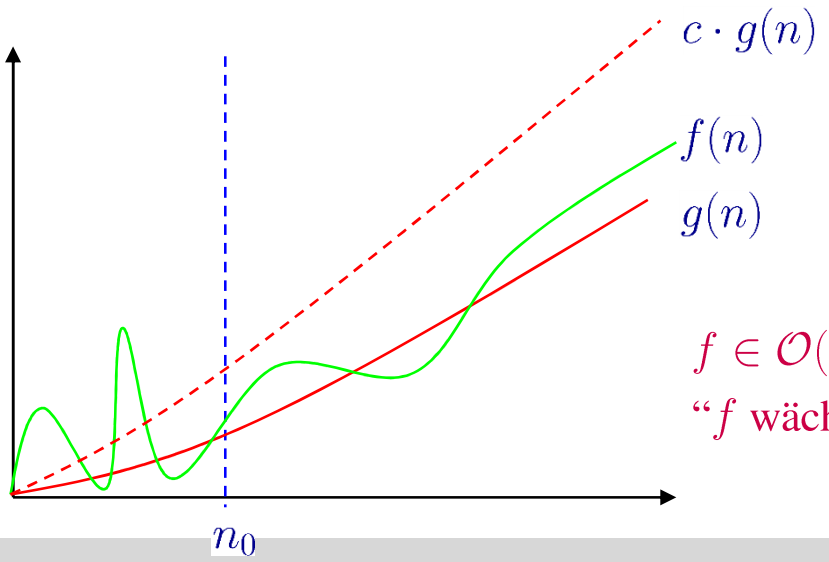
also nicht unterscheiden zwischen z.B. $3 \cdot n^2$ und $5 \cdot n^2$

Definition ($\mathcal{O}$-Notation)

Seien f und g Funktionen von $\mathbb{N}$ nach $\mathbb{N}$. Man schreibt

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

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



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

"f wächst nicht wesentlich schneller als g"

Exkurs $\mathcal{O}$-Notation

Laufzeit $\mathcal{O}(f(n))$ heisst also intuitiv: Laufzeit $f(n)$, 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

Einfache Rechenregeln z.B.:

- $\mathcal{O}(\mathcal{O}(f(n)) = \mathcal{O}(f(n))$
- $\mathcal{O}(f(n)) + \mathcal{O}(g(n)) = \mathcal{O}(f(n) + g(n))$

Das Wortproblem

Gegeben: DEA oder NEA $\mathcal{A}$ und Eingabe $w \in \Sigma^*$ für $\mathcal{A}$

Frage: Gilt $w \in L(\mathcal{A})$?

Wenn $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ DEA:
Berechne Zustand $\delta(q_0, w)$ durch wiederholte Anwendung von δ,
überprüfe dann nur noch, ob das Endzustand ist.

Laufzeitanalyse: δ muss $|w|$ mal angewendet werden,
jede Anwendung braucht max. $\mathcal{O}(|\delta|)$ Zeit.

Dies liefert:

Satz 3.3

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

Für einen NEA ist dies nicht so einfach, da es für Eingabe w mehrere Pfade durch den NEA geben kann.

Naive Ansätze führen zu schlechter Laufzeit:

1. alle Pfade für Eingabewort durchprobieren

   Im schlimmsten Fall $|Q|^{|w|}$ viele, also exponentielle Laufzeit,

2. erst Potenzmengenkonstruktion anwenden

   Resultierender DEA hat $2^{|Q|}$ viele Zustände, also $|\delta| \approx 2^{\mathcal{O}(|Q| \cdot |\Sigma|)}$

   Insgesamt also exponentielle Laufzeit $\mathcal{O}(|w| \cdot 2^{\mathcal{O}(|Q| \cdot |\Sigma|)})$

Wir werden sehen:
es geht auch polynomiell, z.B. unter Verwendung des Leerheitsproblems

Das Leerheitsproblem

Gegeben: NEA $\mathcal{A}$

Frage: Ist $L(\mathcal{A}) = \emptyset$?

Es folgt aus dem (Beweis des) Pumping Lemma, dass man nicht alle (unendlich viele) Wörter auf Enthaltensein in $L(\mathcal{A})$ prüfen muß:

Lemma 3.4

Für jeden NEA $\mathcal{A} = (Q, \Sigma, q_0, \Delta, F)$ gilt:
$L(\mathcal{A}) \neq \emptyset$ gdw. $L(\mathcal{A})$ enthält ein Wort w mit Länge $< |Q|$.

Man kann also $L = \emptyset$ wie folgt entscheiden:

- Betrachte die endlich vielen Wörter $w \in \Sigma^*$ mit $|w| < |Q|$.
- Für jedes solche w, entscheide ob $w \in L$ (naive Methode)

Es gibt $\sum_{i=0..n_0-1} |\Sigma|^i$ viele Wörter der Länge $<n$: exponentielle Laufzeit!

Besserer Algorithmus für Leerheit von NEA $\mathcal{A} = (Q, \Sigma, q_0, \Delta, F)$:

Sei $n_0 = |Q|$

Berechne Folge von Zustandsmengen $P_0, P_1, \ldots, P_{n_0-1}$ wie folgt:

- $P_0 = \{q_0\}$
- $P_{i+1} = P_i \cup \{q \in Q \mid (p, a, q) \in \Delta$ für mind. ein $p \in P_i$ und $a \in \Sigma\}$.

Antworte „ja“ wenn $P_{n_0-1} \cap F = \emptyset$, „nein“ sonst.

Behauptung: für $0 \leq i < n_0$ und alle $q \in Q$ gilt:

$$q \in P_i \quad \text{gdw.} \quad q_0 \stackrel{w}{\Longrightarrow}_{\mathcal{A}} q \quad \text{für Wort } w \text{ mit } |w| \leq i$$

Es folgt: $L(\mathcal{A}) = \emptyset$ gdw. der Algorithmus „ja“ zurückgibt.

Laufzeitanalyse:

- der Algorithmus stoppt nach $|Q|$ Iterationen
- jede Iteration braucht bei naiver Implementierung $\mathcal{O}(|Q| \cdot |\Delta|)$ Zeit

  insgesamt also $\mathcal{O}(|Q|^2 \cdot |\Delta|)$ Zeit
- geschicktere Implementierung mit ausgeklügelten Datenstrukturen

  ermöglicht aber sogar Linearzeit $\mathcal{O}(|Q| + |\Delta|)$

Satz 3.5

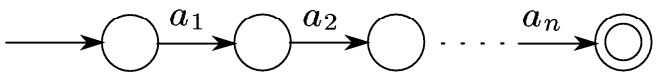
Das Leerheitsproblem für NEAs ist entscheidbar in Zeit $\mathcal{O}(|Q| + |\Delta|)$.

Satz 3.6

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

Beweis:

Gegeben NEA $\mathcal{A} = (Q, \Sigma, q_0, \Delta, F)$ und Eingabe w,
konstruiere zunächst Automaten $\mathcal{A}_w$, der genau $w = a_1 \cdots a_n$ akzeptiert,
d.h. $L(\mathcal{A}_w) = \{a_1 \cdots a_n\}$:



Dieser Automat hat $|w| + 1$ Zustände.

Offenbar ist

$$w \in L(\mathcal{A}) \text{ gdw } L(\mathcal{A}) \cap L(\mathcal{A}_w) \neq \emptyset.$$

(Reduktion Wortproblem auf Leerheitsproblem)

Wir können also folgenden Algorithmus verwenden:

- Produktautomat für $\mathcal{A}$ und $\mathcal{A}_w$ bauen, der erkennt $L(\mathcal{A}) \cap L(\mathcal{A}_w)$
- (Nicht)-Leerheit in Zeit $\mathcal{O}(|Q| + |\Delta|)$ lösen.

Wie groß ist der Produktautomat?

Zustände: $|Q| \cdot (|w| + 1)$

Übergänge:

Da $\mathcal{A}_w$ genau $|w|$ Übergänge hat, ist die Anzahl der Übergänge im Produktautomaten durch $|w| \cdot |\Delta|$ beschränkt.

Nach Satz 3.5 ist daher der Aufwand zum Testen von $L(\mathcal{A}) \cap L(\mathcal{A}_w) \neq \emptyset$:

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

Man überlegt sich leicht, dass der Produktautomat in Zeit $\mathcal{O}(|w| \cdot (|Q| + |\Delta|))$ generiert werden kann, also folgt Satz 3.5.

Das Äquivalenzproblem

Gegeben: DEAs oder NEAs $\mathcal{A}_1, \mathcal{A}_2$.

Frage: Gilt $L(\mathcal{A}_1) = L(\mathcal{A}_2)$?

Reduktion Äquivalenzproblem auf Leerheitsproblem:

$$L_1 = L_2 \text{ gdw } (L_1 \cap \overline{L_2}) \cup (L_2 \cap \overline{L_1}) = \emptyset$$

Vereinigung und Schnitt: Konstruktion vergrößert DEAs+NEAs
nur polynomiell

Komplement: keine Vergrößerung für DEAs,
exponentielle Vergrößerung für NEAs

Satz 3.7

Das Äquivalenzproblem ist für DEAs entscheidbar in polynomieller Zeit und für NEAs in exponentieller Zeit

Man vermutet, dass das Problem für NEAs nicht in Polyzeit lösbar ist.