Kapitel 5

Algorithmen für Reguläre Sprachen

Abschnitt 5.1

Minimierung deterministischer Automaten

Ziel

Zu einer regulären Sprache wollen wir einen DFA mit minimaler Zustandszahl konstruieren. Wir nennen so einen DFA einen minimalen DFA für die Sprache.

Ziel

Zu einer regulären Sprache wollen wir einen DFA mit minimaler Zustandszahl konstruieren. Wir nennen so einen DFA einen minimalen DFA für die Sprache.

Ergebnisse

- Wir werden sehen, dass der Myhill-Nerode DFA einer Sprache ein minimaler DFA ist, und zwar „bis auf Umbenennen der Zustände“ der einzige.

Ziel

Zu einer regulären Sprache wollen wir einen DFA mit minimaler Zustandszahl konstruieren. Wir nennen so einen DFA einen minimalen DFA für die Sprache.

Ergebnisse

- Wir werden sehen, dass der Myhill-Nerode DFA einer Sprache ein minimaler DFA ist, und zwar „bis auf Umbenennen der Zustände“ der einzige.
- Wir werden einen effizienten Algorithmus angeben, um diesen minimalen DFA zu berechnen.

Ziel

Zu einer regulären Sprache wollen wir einen DFA mit minimaler Zustandszahl konstruieren. Wir nennen so einen DFA einen minimalen DFA für die Sprache.

Ergebnisse

- Wir werden sehen, dass der Myhill-Nerode DFA einer Sprache ein minimaler DFA ist, und zwar „bis auf Umbenennen der Zustände“ der einzige.
- Wir werden einen effizienten Algorithmus angeben, um diesen minimalen DFA zu berechnen.

Bemerkung 5.1

Ein effizientes Minimierungsverfahren für NFAs ist nicht bekannt.

Minimalität des Myhill-Nerode DFA

Notation

Mit $|\mathcal{A}|$ bezeichnen wir die Anzahl der Zustände eines endlichen Automaten $\mathcal{A}$.

Minimalität des Myhill-Nerode DFA

Notation

Mit $|\mathcal{A}|$ bezeichnen wir die Anzahl der Zustände eines endlichen Automaten $\mathcal{A}$.

Definition 5.2

Ein DFA $\mathcal{A}$ ist minimal, wenn $|\mathcal{A}| \leq |\mathcal{A}'|$ für alle zu $\mathcal{A}$ äquivalenten DFAs $\mathcal{A}'$.

Minimalität des Myhill-Nerode DFA

Notation

Mit $|\mathcal{A}|$ bezeichnen wir die Anzahl der Zustände eines endlichen Automaten $\mathcal{A}$.

Definition 5.2

Ein DFA $\mathcal{A}$ ist minimal, wenn $|\mathcal{A}| \leq |\mathcal{A}'|$ für alle zu $\mathcal{A}$ äquivalenten DFAs $\mathcal{A}'$.

Ein minimaler DFA für eine Sprache L ist ein minimaler DFA $\mathcal{A}$ mit $L(\mathcal{A}) = L$.

Minimalität des Myhill-Nerode DFA

Notation

Mit $|\mathcal{A}|$ bezeichnen wir die Anzahl der Zustände eines endlichen Automaten $\mathcal{A}$.

Definition 5.2

Ein DFA $\mathcal{A}$ ist minimal, wenn $|\mathcal{A}| \leq |\mathcal{A}'|$ für alle zu $\mathcal{A}$ äquivalenten DFAs $\mathcal{A}'$.

Ein minimaler DFA für eine Sprache L ist ein minimaler DFA $\mathcal{A}$ mit $L(\mathcal{A}) = L$.

Satz 5.3

Sei L eine reguläre Sprache. Dann ist der Myhill-Nerode DFA $\mathcal{A}_L$ ein minimaler DFA für L.

Beweis von Satz 5.3

Zur Erinnerung:

Lemma 4.46

Sei $\mathcal{A}$ ein DFA und $L = L(\mathcal{A})$. Dann gilt $\mathsf{index}(L) \leq |\mathcal{A}|$.

Weiterhin erinnern wir uns, dass die Zustandsmenge des Myhill-Nerode DFA $\mathcal{A}_L$ die Menge aller $\sim_L$-Äquivalenzklassen ist. Also gilt

$$|\mathcal{A}_L| = \mathsf{index}(L).$$

Daraus folgt der Satz sofort. Sei nämlich $\mathcal{A}$ ein beliebiger DFA mit $L(\mathcal{A}) = L$. Dann gilt

$$|\mathcal{A}_L| = \mathsf{index}(L) \leq |\mathcal{A}|.$$

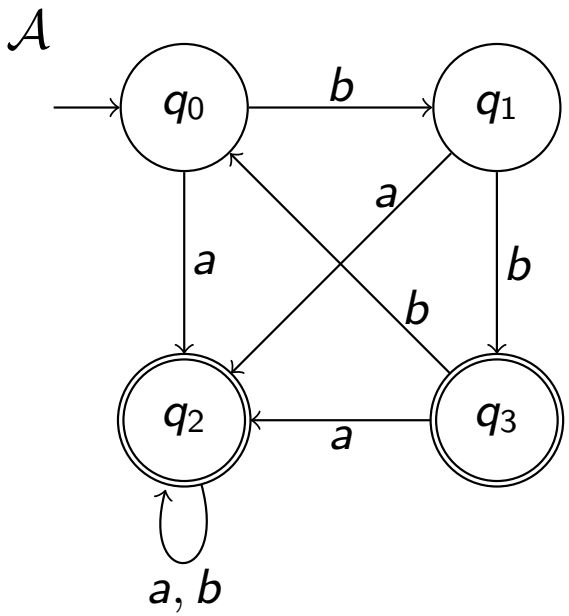
□

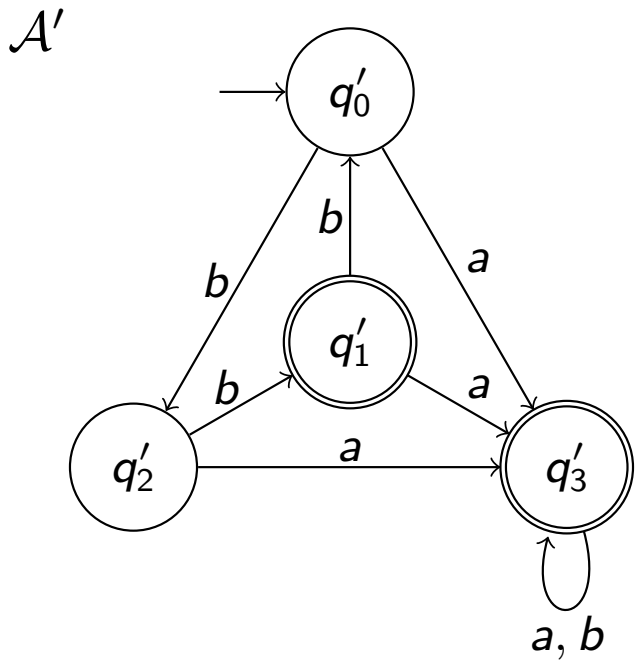
Eindeutigkeit

Als nächstes wollen wir zeigen, dass der Myhill-Nerode DFA „bis auf Umbenennen der Zustände“ der einzige minimale DFA für eine Sprache ist.

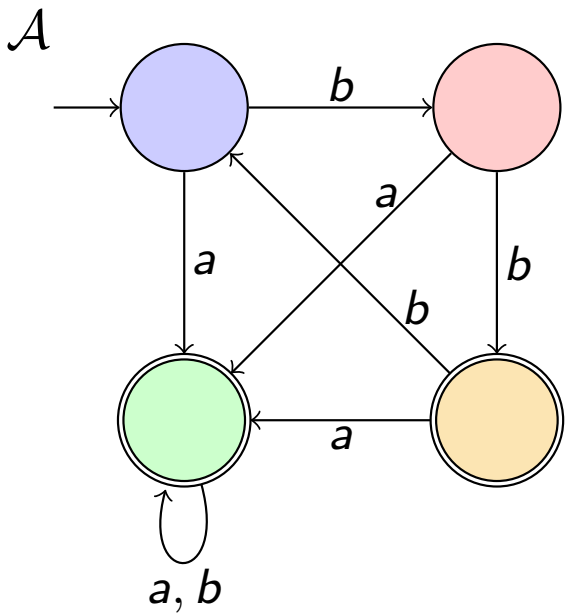
Mathematisch erfassen wir diese Eindeutigkeit „bis auf Umbenennen der Zustände“ durch den Begriff der Isomorphie von DFAs.

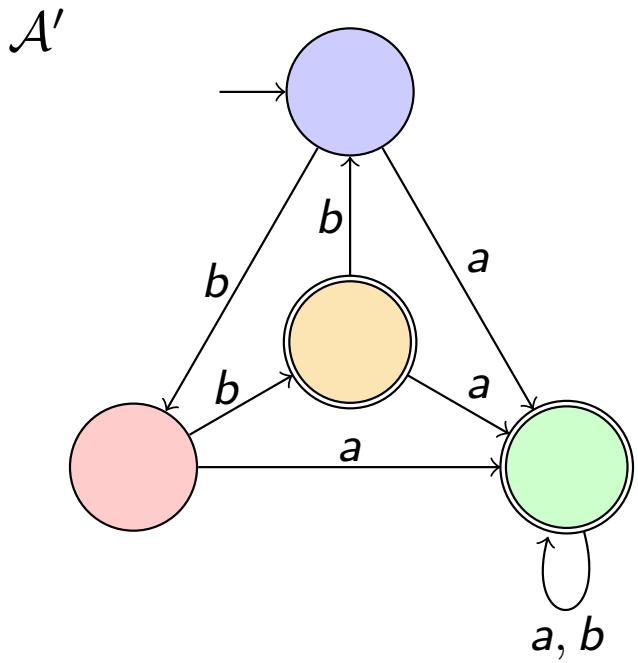
Isomorphie von DFAs: Beispiel 5.4





Isomorphie von DFAs: Beispiel 5.4





Identifiziert man die Zustände gleicher Farbe, so ergibt sich der gleiche DFA.

Definition 5.5

Seien $\mathcal{A}_1 = (Q_1, \Sigma, \delta_1, q_{10}, F_1)$ und $\mathcal{A}_2 = (Q_2, \Sigma, \delta_2, q_{20} F_2)$ DFAs.

Definition 5.5

Seien $\mathcal{A}_1 = (Q_1, \Sigma, \delta_1, q_{10}, F_1)$ und $\mathcal{A}_2 = (Q_2, \Sigma, \delta_2, q_{20} F_2)$ DFAs.

Ein Isomorphismus von $\mathcal{A}_1$ auf $\mathcal{A}_2$ ist eine bijektive Funktion $f : Q_1 \to Q_2$ mit

- $f(\delta_1(q, a)) = \delta_2(f(q), a)$ für alle $q \in Q_1$ und $a \in \Sigma$
- $f(q_{10}) = q_{20}$
- $f(F_1) = F_2$

Definition 5.5

Seien $\mathcal{A}_1 = (Q_1, \Sigma, \delta_1, q_{10}, F_1)$ und $\mathcal{A}_2 = (Q_2, \Sigma, \delta_2, q_{20} F_2)$ DFAs.

Ein Isomorphismus von $\mathcal{A}_1$ auf $\mathcal{A}_2$ ist eine bijektive Funktion $f : Q_1 \to Q_2$ mit

- $f(\delta_1(q, a)) = \delta_2(f(q), a)$ für alle $q \in Q_1$ und $a \in \Sigma$
- $f(q_{10}) = q_{20}$
- $f(F_1) = F_2$

Wir schreiben $f : \mathcal{A}_1 \cong \mathcal{A}_2$.

Definition 5.5

Seien $\mathcal{A}_1 = (Q_1, \Sigma, \delta_1, q_{10}, F_1)$ und $\mathcal{A}_2 = (Q_2, \Sigma, \delta_2, q_{20} F_2)$ DFAs.

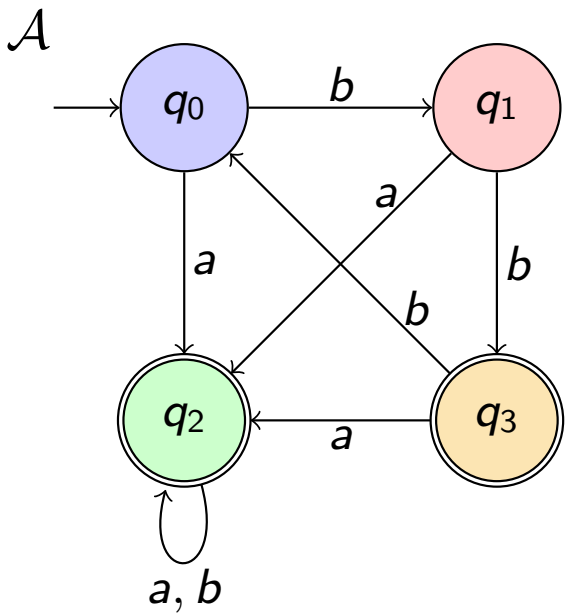
Ein Isomorphismus von $\mathcal{A}_1$ auf $\mathcal{A}_2$ ist eine bijektive Funktion $f : Q_1 \to Q_2$ mit

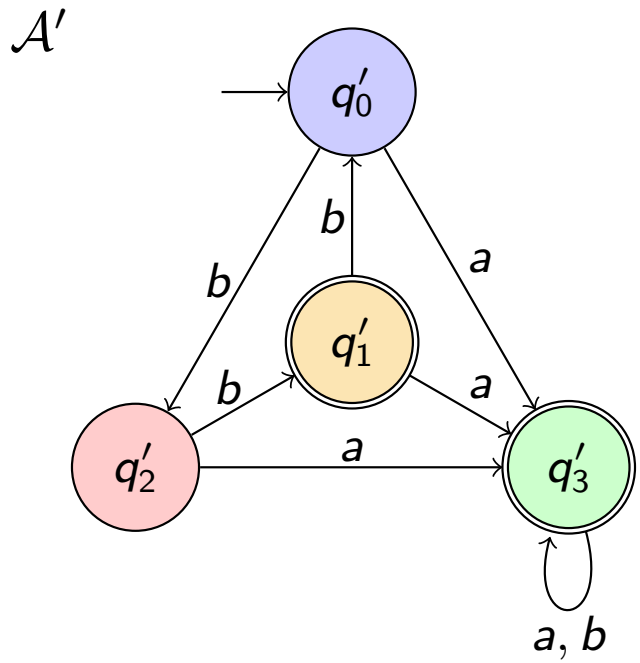
- $f(\delta_1(q, a)) = \delta_2(f(q), a)$ für alle $q \in Q_1$ und $a \in \Sigma$
- $f(q_{10}) = q_{20}$
- $f(F_1) = F_2$

Wir schreiben $f : \mathcal{A}_1 \cong \mathcal{A}_2$.

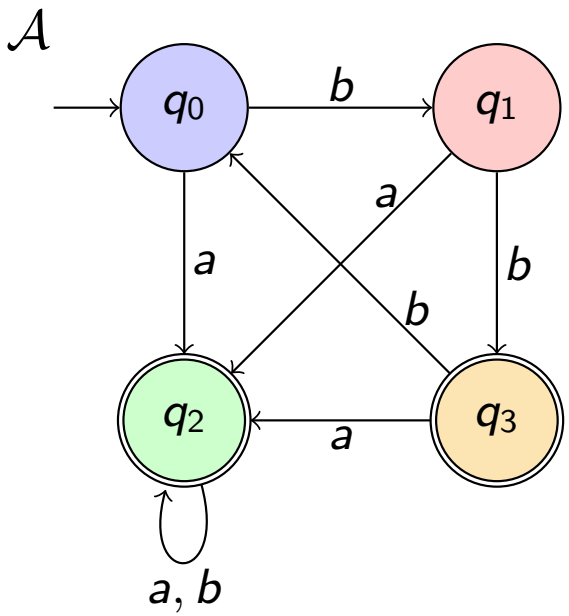
$\mathcal{A}_1$ ist isomorph zu $\mathcal{A}_2$ (wir schreiben: $\mathcal{A}_1 \cong \mathcal{A}_2$), falls es einen Isomorphismus von $\mathcal{A}_1$ auf $\mathcal{A}_2$ gibt.

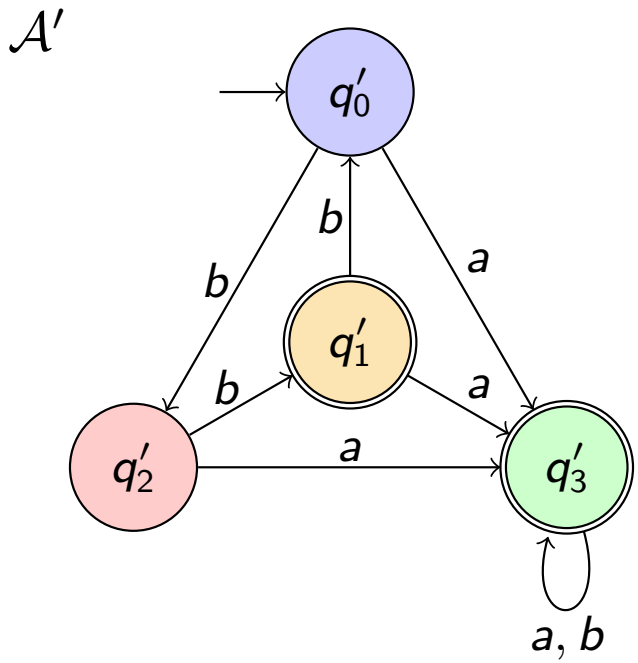
Beispiel 5.4 (Forts.)





Beispiel 5.4 (Forts.)





Wir definieren $f : \{q_0, \ldots, q_3\} \to \{q'_0, \ldots, q'_3\}$ durch

q	q_0	q_1	q_2	q_3
$f(q)$	q'_0	q'_2	q'_3	q'_1

Dann $f : \mathcal{A} \cong \mathcal{A}'$.

Eindeutigkeit des minimalen DFA

Satz 5.6

Sei $\mathcal{A}$ ein minimaler DFA für eine Sprache L. Dann gilt

$$\mathcal{A} \cong \mathcal{A}_L.$$

Sei $L \subseteq \Sigma^*$.

Zur Erinnerung:

$$\mathcal{A}_L = (Q_L, \Sigma, \delta_L, q_{L0}, F_L)$$

mit

- $Q_L = \{v/_L \mid v \in \Sigma^*\}$,
- $\delta_L : Q_L \times \Sigma \to Q_L$ definiert durch

$$\delta_L(v/_L, a) = va/_L,$$

- $q_{L0} = \varepsilon/_L$,
- $F_L = \{v/_L \mid v/_L \cap L \neq \emptyset\}$.

Sei $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ ein minimaler DFA für L.

Wie im Beweis von Lemma 4.46 sei für alle $v \in \Sigma^*$

$$q_v \in Q$$

der eindeutige Zustand mit

$$\mathcal{A} : q_0 \xrightarrow{v} q_v.$$

Im Beweis von Lemma 4.46 haben wir folgende Behauptung bewiesen:

Behauptung 1

Seien $v, w \in \Sigma^*$ mit $q_v = q_w$. Dann gilt

$$v \sim_L w.$$

Man beachte, dass alle Zustände in $\mathcal{A}$ erreichbar sind, denn sonst wäre die Reduktion von $\mathcal{A}$ auf die erreichbaren Zustände ein kleinerer äquivalenter DFA, was der Minimalität von $\mathcal{A}$ widerspräche.

Also gibt es für alle $q \in Q$ ein $v \in \Sigma^*$, so dass $\mathcal{A} : q_0 \xrightarrow{v} q$, was $q = q_v$ bedeutet. Damit gilt

$$Q = \{q_v \mid v \in \Sigma^*\}.$$

Wir definieren jetzt eine Abbildung $f : Q \to Q_L$ durch

$$f(q_v) := v/_L.$$

Behauptung 2

f ist wohldefiniert.

Beweis.

Wir müssen zeigen, dass der Wert $f(q_v)$ nur vom Zustand q_v und nicht vom Wort v abhängt.

Seien dazu $v, w \in \Sigma^*$ mit $q_v = q_w$. Nach Behauptung 1 gilt dann $v \sim_L w$. Also gilt

$$f(q_v) = v/_L = w/_L = f(q_w).$$

□

Behauptung 3

f ist bijektiv.

Beweis.

f ist surjektiv, denn für alle $v/_L \in Q_L$ ist $f(q_v) = v/_L$.

Weil $|Q| \leq |Q_L|$ wegen der Minimalität von $\mathcal{A}$ muss f dann auch injektiv sein. □

Behauptung 4

$f : \mathcal{A} \cong \mathcal{A}_L$.

Beweis.

Wir müssen noch zeigen:

(i) $f(\delta(q, a)) = \delta_L(f(q), a)$ für alle $q \in Q$ und $a \in \Sigma$,

(ii) $f(q_0) = q_{L0}$,

(iii) $f(F) = F_L$.

Beweis von (i).

Sei $q \in Q$ und $a \in \Sigma$. Sei $v \in \Sigma^*$, so dass $q = q_v$.

Dann ist $\delta(q, a) = q_{va}$, denn

$$\mathcal{A} : q_0 \xrightarrow{v} q \xrightarrow{a} \delta(q, a),$$

und $\mathcal{A} : q_0 \xrightarrow{qa} q_{va}$. Weil $\mathcal{A}$ deterministisch ist bedeutet das $\delta(q, a) = q_{va}$.

Daraus folgt

$$f(\delta(q, a)) = f(q_{va}) = va/_L = \delta_L(v/_L, a) = \delta_L(f(q_v), a).$$

Beweis von (ii).

Es gilt $\mathcal{A} : q_0 \xrightarrow{\varepsilon} q_0$, also $q_0 = q_\varepsilon$ und damit

$$f(q_0) = \varepsilon/_L = q_{L0}.$$

Beweis von (iii).

Wir haben bereits gezeigt (Behauptung 2 im Beweis von Lemma 4.51):

Für alle $w \in \Sigma^*$ gilt

$$w \in L \iff w/_L \in F_L.$$

Außerdem gilt für alle $w \in \Sigma^*$:

$$w \in L \iff \exists q \in F : \mathcal{A} : q_0 \xrightarrow{w} q \iff q_w \in F.$$

Damit gilt für $q = q_w \in Q$:

$$q \in F \iff w \in L \iff f(q) = w/_L \in F_L.$$

□

Minimierungsverfahren

Ziel

Ein möglichst effizientes Verfahren, das zu einem gegebenen DFA einen äquivalenten minimalen DFA konstruiert.

Reduktion auf die erreichbaren Zustände

Zur Erinnerung (vgl. Definition 2.20)

Sei $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ ein DFA.

- Ein Zustand $q \in Q$ ist erreichbar, wenn es ein Wort $w \in \Sigma^*$ gibt, so dass $\mathcal{A} : q_0 \xrightarrow{w} q$.

Reduktion auf die erreichbaren Zustände

Zur Erinnerung (vgl. Definition 2.20)

Sei $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ ein DFA.

- Ein Zustand $q \in Q$ ist erreichbar, wenn es ein Wort $w \in \Sigma^*$ gibt, so dass $\mathcal{A} : q_0 \xrightarrow{w} q$.
- Sind alle Zustände von $\mathcal{A}$ erreichbar, so nennen wir $\mathcal{A}$ reduziert.

Reduktion auf die erreichbaren Zustände

Zur Erinnerung (vgl. Definition 2.20)

Sei $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ ein DFA.

- Ein Zustand $q \in Q$ ist erreichbar, wenn es ein Wort $w \in \Sigma^*$ gibt, so dass $\mathcal{A} : q_0 \xrightarrow{w} q$.
- Sind alle Zustände von $\mathcal{A}$ erreichbar, so nennen wir $\mathcal{A}$ reduziert.
- Die Reduktion von $\mathcal{A}$ auf die erreichbaren Zustände ist der DFA $\mathcal{A}'$, der aus $\mathcal{A}$ durch Weglassen aller nicht erreichbaren Zustände und deren Transitionen entsteht.

Reduktion auf die erreichbaren Zustände

Zur Erinnerung (vgl. Definition 2.20)

Sei $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ ein DFA.

- Ein Zustand $q \in Q$ ist erreichbar, wenn es ein Wort $w \in \Sigma^*$ gibt, so dass $\mathcal{A} : q_0 \xrightarrow{w} q$.
- Sind alle Zustände von $\mathcal{A}$ erreichbar, so nennen wir $\mathcal{A}$ reduziert.
- Die Reduktion von $\mathcal{A}$ auf die erreichbaren Zustände ist der DFA $\mathcal{A}'$, der aus $\mathcal{A}$ durch Weglassen aller nicht erreichbaren Zustände und deren Transitionen entsteht.

  Beobachtung 2.22: $\mathcal{A}$ und $\mathcal{A}'$ sind äquivalent.

Reduktion auf die erreichbaren Zustände

Zur Erinnerung (vgl. Definition 2.20)

Sei $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ ein DFA.

- Ein Zustand $q \in Q$ ist erreichbar, wenn es ein Wort $w \in \Sigma^*$ gibt, so dass $\mathcal{A} : q_0 \xrightarrow{w} q$.
- Sind alle Zustände von $\mathcal{A}$ erreichbar, so nennen wir $\mathcal{A}$ reduziert.
- Die Reduktion von $\mathcal{A}$ auf die erreichbaren Zustände ist der DFA $\mathcal{A}'$, der aus $\mathcal{A}$ durch Weglassen aller nicht erreichbaren Zustände und deren Transitionen entsteht.

  Beobachtung 2.22: $\mathcal{A}$ und $\mathcal{A}'$ sind äquivalent.

Erster Schritt des Minimierungsverfahrens

Reduziere den gegebenen DFA auf die erreichbaren Zustände.

Reduktion auf die erreichbaren Zustände

Zur Erinnerung (vgl. Definition 2.20)

Sei $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ ein DFA.

- Ein Zustand $q \in Q$ ist erreichbar, wenn es ein Wort $w \in \Sigma^*$ gibt, so dass $\mathcal{A} : q_0 \xrightarrow{w} q$.
- Sind alle Zustände von $\mathcal{A}$ erreichbar, so nennen wir $\mathcal{A}$ reduziert.
- Die Reduktion von $\mathcal{A}$ auf die erreichbaren Zustände ist der DFA $\mathcal{A}'$, der aus $\mathcal{A}$ durch Weglassen aller nicht erreichbaren Zustände und deren Transitionen entsteht.

  Beobachtung 2.22: $\mathcal{A}$ und $\mathcal{A}'$ sind äquivalent.

Erster Schritt des Minimierungsverfahrens

Reduziere den gegebenen DFA auf die erreichbaren Zustände.

Im Folgenden werden wir uns auf reduzierte DFAs konzentrieren.

Zustandsäquivalenz

Idee

Zwei Zustände sind äquivalent, wenn von ihnen aus dieselben Wörter akzeptiert werden.

Zustandsäquivalenz

Idee

Zwei Zustände sind äquivalent, wenn von ihnen aus dieselben Wörter akzeptiert werden.

Notation

Sei $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ und $p \in Q$. Wir schreiben $\mathcal{A} : p \xrightarrow{w} F$, wenn es ein $r \in F$ gibt, so dass $\mathcal{A} : p \xrightarrow{w} r$.

Idee

Zwei Zustände sind äquivalent, wenn von ihnen aus dieselben Wörter akzeptiert werden.

Notation

Sei $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ und $q \in Q$. Wir schreiben $\mathcal{A} : p \xrightarrow{w} F$, wenn es ein $r \in F$ gibt, so dass $\mathcal{A} : p \xrightarrow{w} r$.

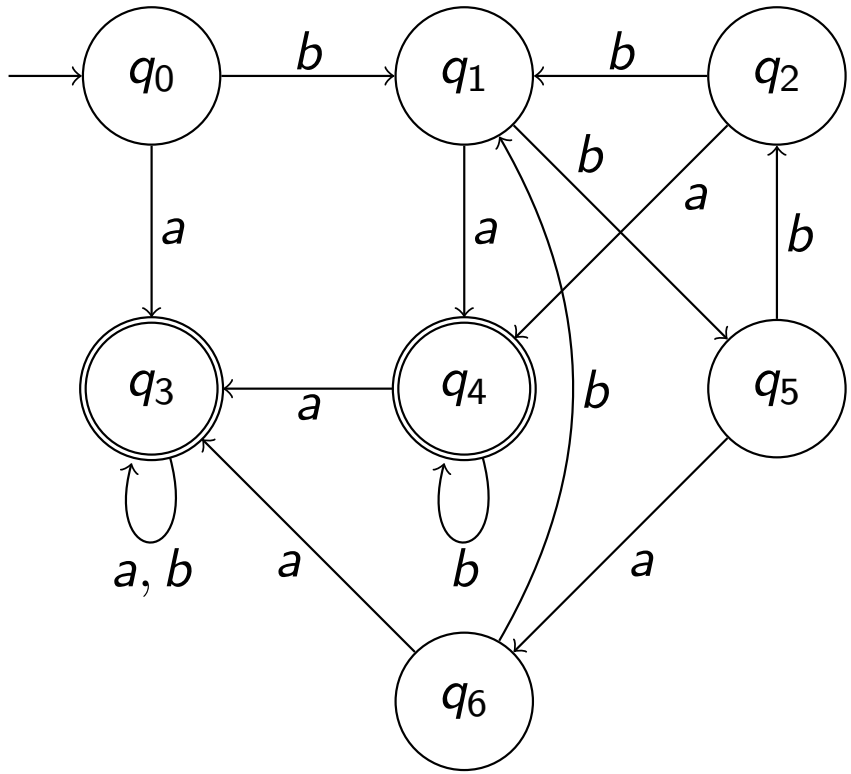
Definition 5.7

Sei $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ ein DFA.
Wir definieren eine zweistellige Relation $\sim_{\mathcal{A}}$ auf Q durch

$$p \sim_{\mathcal{A}} q :\Longleftrightarrow \text{für alle } x \in \Sigma^* : \left(\mathcal{A} : p \xrightarrow{x} F \iff \mathcal{A} : q \xrightarrow{x} F\right)$$

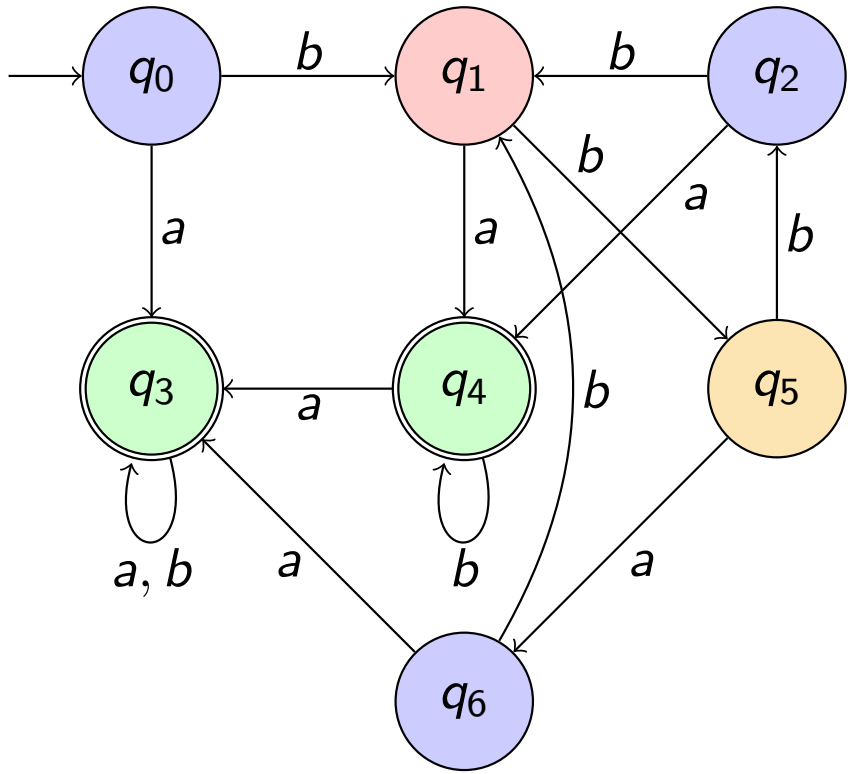
Beispiel 5.8

$\mathcal{A}$



Beispiel 5.8

$\mathcal{A}$



Äquivalenzklassen von $\sim_{\mathcal{A}}$:

$$\{q_0, q_2, q_6\},$$
$$\{q_1\},$$
$$\{q_3, q_4\},$$
$$\{q_5\}$$

Eigenschaften der Zustandsäquivalenz

Lemma 5.9

Sei $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ ein DFA.

1. *$\sim_{\mathcal{A}}$ ist eine Äquivalenzrelation.*

Eigenschaften der Zustandsäquivalenz

Lemma 5.9

Sei $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ ein DFA.

1. *$\sim_{\mathcal{A}}$ ist eine Äquivalenzrelation.*
2. *Für alle $p, q \in Q$ und $a \in \Sigma$ gilt*

$$p \sim_{\mathcal{A}} q \implies \delta(p, a) \sim_{\mathcal{A}} \delta(q, a).$$

Beweis von Lemma 5.9

1. Folgt leicht aus der Tatsache, dass „$\Longleftrightarrow$“ eine Äquivalenzrelation ist.

2. Seien $p, q \in Q$, so dass $p \sim_{\mathcal{A}} q$, und $a \in \Sigma$.

   Zu zeigen: Für alle $x \in \Sigma^*$ gilt

   $$\mathcal{A} : \delta(p, a) \xrightarrow{x} F \iff \mathcal{A} : \delta(q, a) \xrightarrow{x} F.$$

   Sei $x \in \Sigma^*$. Dann gilt

   $$\begin{aligned} \mathcal{A} : \delta(p, a) \xrightarrow{x} F &\iff \mathcal{A} : p \xrightarrow{ax} F \\ &\iff \mathcal{A} : q \xrightarrow{ax} F && \text{weil } p \sim_{\mathcal{A}} q \\ &\iff \mathcal{A} : \delta(q, a) \xrightarrow{x} F. \end{aligned}$$

□

Zustandsäquivalenz vs Myhill-Nerode Äquivalenz

Lemma 5.10

Sei $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ ein DFA und $L = L(\mathcal{A})$. Für alle $w \in \Sigma^*$ sei $q_w \in Q$ der eindeutige Zustand mit $\mathcal{A} : q_0 \xrightarrow{w} q_w$.
Dann gilt für alle $v, w \in \Sigma^*$:

$$q_v \sim_{\mathcal{A}} q_w \iff v \sim_L w.$$

Beweis von Lemma 5.10

$$
\begin{aligned}
q_v \sim_{\mathcal{A}} q_w &\iff \text{für alle } x \in \Sigma^* : \left(q_v \xrightarrow{x} F \iff q_w \xrightarrow{x} F\right) \\
&\iff \text{für alle } x \in \Sigma^* : \left(q_0 \xrightarrow{vx} F \iff q_0 \xrightarrow{wx} F\right) \\
&\iff \text{für alle } x \in \Sigma^* : \left(vx \in L \iff wx \in L\right) \\
&\iff v \sim_L w.
\end{aligned}
$$

□

Minimierung mit Zustandsäquivalenz

Satz 5.11

Seien $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ ein reduzierter DFA, $\sim := \sim_{\mathcal{A}}$, und $L := L(\mathcal{A})$. Sei

$$\widetilde{\mathcal{A}} := \big(\widetilde{Q}, \Sigma, \widetilde{\delta}, \widetilde{q}_0, \widetilde{F}\big)$$

mit

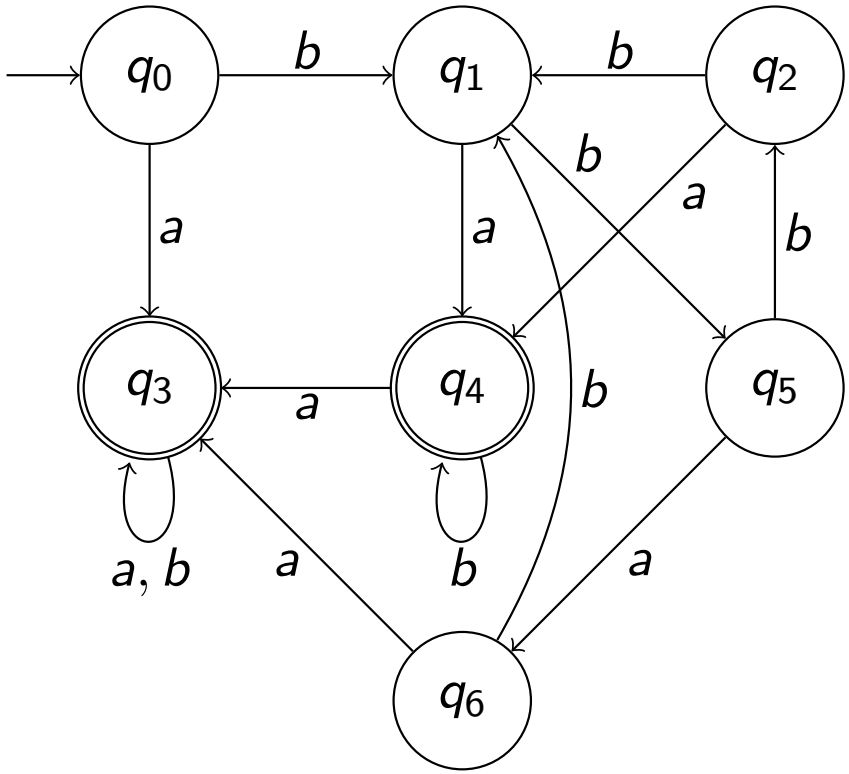
- $\widetilde{Q} := \{q/_\sim \mid q \in Q\}$,
- $\widetilde{\delta}(q/_\sim, a) := \delta(q, a)/_\sim$ *für alle* $q \in Q, a \in \Sigma$,
- $\widetilde{q}_0 := q_0/_\sim$,
- $\widetilde{F} := \{q/_\sim \mid q/_\sim \cap F \neq \emptyset\}$.

Dann ist $\widetilde{\mathcal{A}}$ ein DFA, und es gilt

$$\widetilde{\mathcal{A}} \cong \mathcal{A}_L.$$

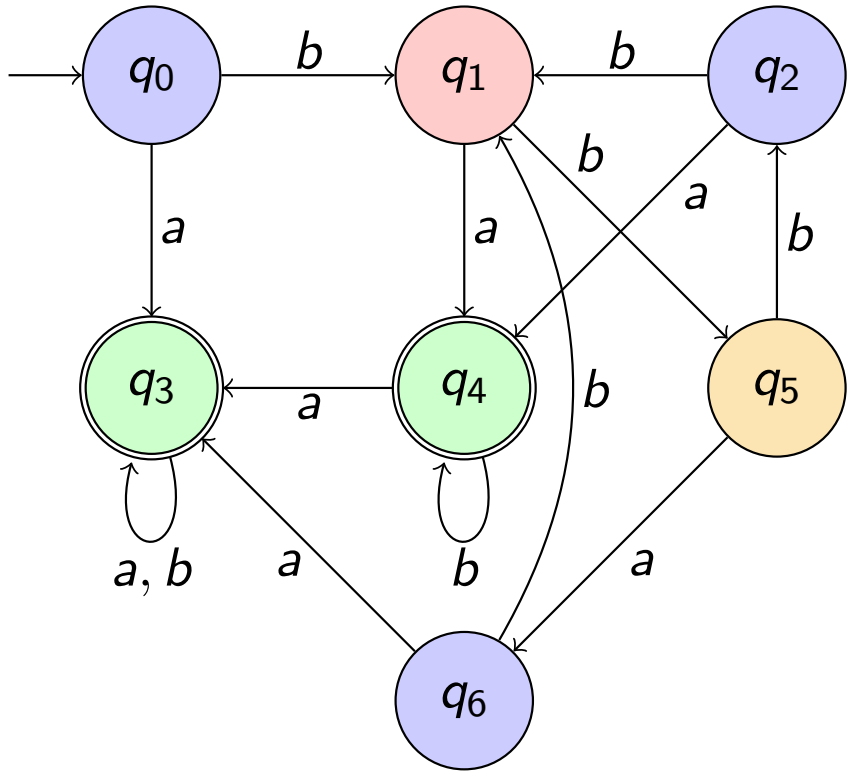
Beispiel 5.8 (Forts.)

$\mathcal{A}$



Beispiel 5.8 (Forts.)

$\mathcal{A}$

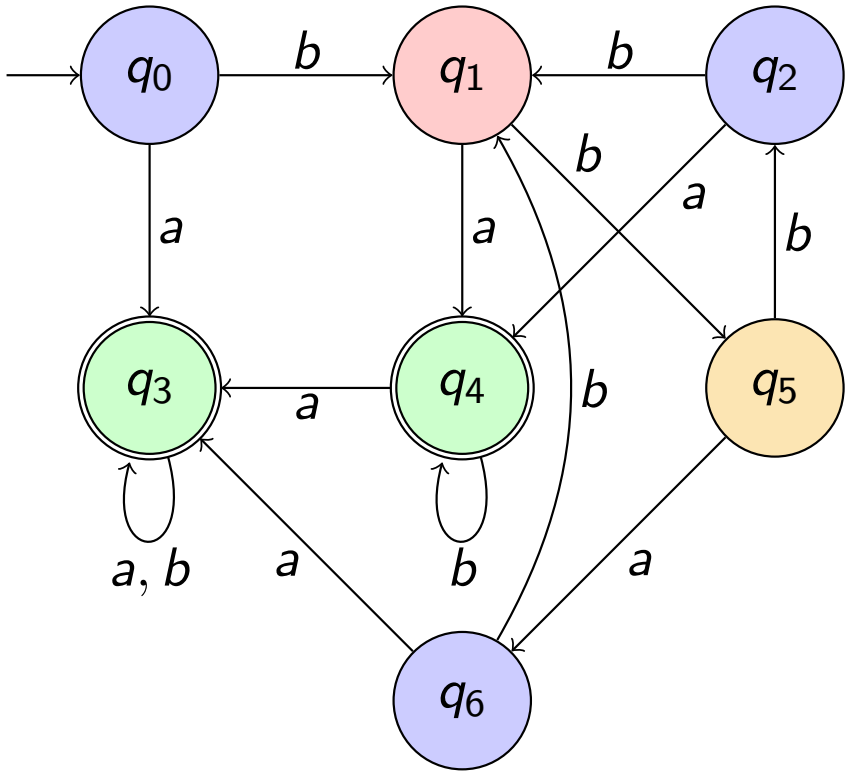


Äquivalenzklassen von $\sim_{\mathcal{A}}$:

$$\widetilde{q_0} = \{q_0, q_2, q_6\},$$
$$\widetilde{q_1} = \{q_1\},$$
$$\widetilde{q_3} = \{q_3, q_4\},$$
$$\widetilde{q_5} = \{q_5\}$$

Beispiel 5.8 (Forts.)

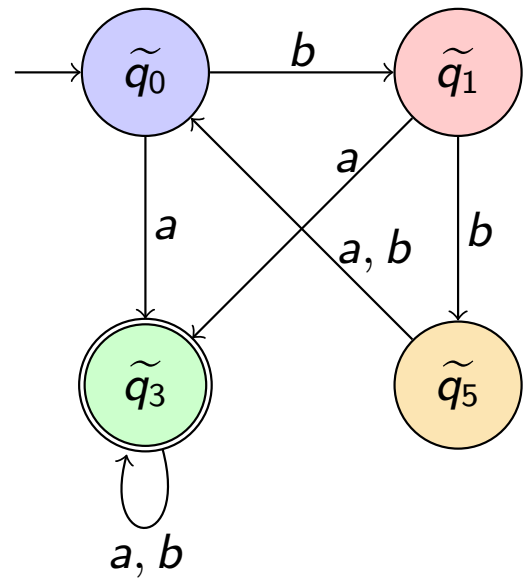
$\mathcal{A}$



Äquivalenzklassen von $\sim_{\mathcal{A}}$:

$$\widetilde{q_0} = \{q_0, q_2, q_6\},$$
$$\widetilde{q_1} = \{q_1\},$$
$$\widetilde{q_3} = \{q_3, q_4\},$$
$$\widetilde{q_5} = \{q_5\}$$

$\widetilde{\mathcal{A}}$



Zur Erinnerung:

$$\widetilde{\mathcal{A}} = \big(\widetilde{Q}, \Sigma, \widetilde{\delta}, \widetilde{q}_0, \widetilde{F}\big) \qquad \mathcal{A}_L = (Q_L, \Sigma, \delta_L, q_{L0}, F_L)$$

mit

- $\widetilde{Q} = \{q/_\sim \mid q \in Q\}$,
- $\widetilde{\delta}(q/_\sim, a) = \delta(q, a)/_\sim$ für alle $q \in Q, a \in \Sigma$,
- $\widetilde{q}_0 = q_0/_\sim$,
- $\widetilde{F} = \{q/_\sim \mid q/_\sim \cap F \neq \emptyset\}$.

mit

- $Q_L = \{v/_L \mid v \in \Sigma^*\}$,
- $\delta_L(v/_L, a) = va/_L$ für alle $v \in \Sigma^*, a \in \Sigma$.
- $q_{L0} = \varepsilon/_L$,
- $F_L = \{v/_L \mid v/_L \cap L \neq \emptyset\}$.

$\widetilde{\delta}$ ist wohldefiniert, weil nach Lemma 5.9 für alle $p, q \in Q$ mit $p/_\sim = q/_\sim$ gilt

$$\delta(p, a)/_\sim = \delta(q, a)/_\sim.$$

Also ist $\widetilde{\mathcal{A}}$ ein DFA.

Weil $\mathcal{A}$ reduziert ist, gibt es für alle $q \in Q$ ein $v \in \Sigma^*$, so dass $q = q_v$.
Wir definieren eine Abbildung $f : \widetilde{Q} \to Q_L$ durch

$$f(q_v/_\sim) := v/_L.$$

Weil $q_v \sim q_w \iff v \sim_L w$ (nach Lemma 5.10) ist f wohldefiniert und bijektiv.

Behauptung

$$f : \widetilde{\mathcal{A}} \cong \mathcal{A}_L.$$

Beweis.

Wir müssen noch zeigen:

(i) $f(\widetilde{\delta}(q/_\sim, a)) = \delta_L(f(q/_\sim), a)$ für alle $q \in Q$ und $a \in \Sigma$,

(ii) $f(\widetilde{q}_0) = q_{L0}$,

(iii) $f(\widetilde{F}) = F_L$.

Beweis von (i).

Sei $q \in Q$ und $a \in \Sigma$. Sei $v \in \Sigma^*$, so dass $q = q_v$.
Dann ist $\widetilde{\delta}(q/_\sim, a) = q_{va}/_\sim$, denn $\delta(q_v, a) = q_{va}$.
Daraus folgt

$$f(\widetilde{\delta}(q/_\sim, a)) = f(q_{va}/_\sim) = va/_L = \delta_L(v/_L, a) = \delta_L(f(q_v/_\sim), a).$$

Beweis von (ii).

Es gilt $q_0 = q_\varepsilon$ und damit

$$f(\widetilde{q}_0) = f(q_0/_\sim) = \varepsilon/_L = q_{L0}.$$

Beweis von (iii).

Für alle $p, q \in Q$ gilt

$$\begin{aligned} p \sim q &\iff \big(\mathcal{A} : p \xrightarrow{\varepsilon} F \iff \mathcal{A} : q \xrightarrow{\varepsilon} F\big) \\ &\implies \big(p \in F \iff q \in F\big). \end{aligned}$$

Für alle $q \in Q$ gilt deswegen

$$q \in F \iff \widetilde{q} \in \widetilde{F}.$$

Daraus folgt für alle $w \in \Sigma^*$:

$$w \in L \iff q_w \in F \iff q_w/_\sim \in \widetilde{F}.$$

Wir haben bereits gezeigt (Behauptung 2 im Beweis von Lemma 4.51), dass für alle $w \in \Sigma^*$ gilt:

$$w \in L \iff w/_L \in F_L.$$

Insgesamt ergibt sich für alle $q = q_w \in Q$:

$$q/_\sim \in \widetilde{F} \iff w \in L \iff f(q/_\sim) = w/_L \in F_L.$$

□

Minimierungsalgorithmus

Eingabe: DFA $\mathcal{A}$

Minimierungsalgorithmus

Eingabe: DFA $\mathcal{A}$

Algorithmus

1. Berechne Menge der erreichbaren Zustände von $\mathcal{A}$ (mittels Tiefensuche im Transitionsgraphen, ausgehend vom Anfangszustand)

Minimierungsalgorithmus

Eingabe: DFA $\mathcal{A}$

Algorithmus

1. Berechne Menge der erreichbaren Zustände von $\mathcal{A}$ (mittels Tiefensuche im Transitionsgraphen, ausgehend vom Anfangszustand)
2. Berechne Reduktion $\mathcal{A}'$ von $\mathcal{A}$ auf die erreichbaren Zustände

Minimierungsalgorithmus

Eingabe: DFA $\mathcal{A}$

Algorithmus

1. Berechne Menge der erreichbaren Zustände von $\mathcal{A}$
   (mittels Tiefensuche im Transitionsgraphen, ausgehend vom Anfangszustand)
2. Berechne Reduktion $\mathcal{A}'$ von $\mathcal{A}$ auf die erreichbaren Zustände
3. Berechne Äquivalenzklassen von $\sim_{\mathcal{A}'}$
   (mit dem Verfeinerungsalgorithmus)

Minimierungsalgorithmus

Eingabe: DFA $\mathcal{A}$

Algorithmus

1. Berechne Menge der erreichbaren Zustände von $\mathcal{A}$
   (mittels Tiefensuche im Transitionsgraphen, ausgehend vom Anfangszustand)
2. Berechne Reduktion $\mathcal{A}'$ von $\mathcal{A}$ auf die erreichbaren Zustände
3. Berechne Äquivalenzklassen von $\sim_{\mathcal{A}'}$
   (mit dem Verfeinerungsalgorithmus)
4. Berechne den minimalen DFA $\widetilde{\mathcal{A}}'$.

Minimierungsalgorithmus

Eingabe: DFA $\mathcal{A}$

Algorithmus

1. Berechne Menge der erreichbaren Zustände von $\mathcal{A}$ (mittels Tiefensuche im Transitionsgraphen, ausgehend vom Anfangszustand)
2. Berechne Reduktion $\mathcal{A}'$ von $\mathcal{A}$ auf die erreichbaren Zustände
3. Berechne Äquivalenzklassen von $\sim_{\mathcal{A}'}$ (mit dem Verfeinerungsalgorithmus)
4. Berechne den minimalen DFA $\widetilde{\mathcal{A}}'$.

Korrektheit: Sei $L = L(\mathcal{A})$. Es gilt $L(\mathcal{A}') = L(\mathcal{A}) = L$ und $\widetilde{\mathcal{A}}' \cong \mathcal{A}_L$.

Minimierungsalgorithmus

Eingabe: DFA $\mathcal{A}$

Algorithmus

1. Berechne Menge der erreichbaren Zustände von $\mathcal{A}$ (mittels Tiefensuche im Transitionsgraphen, ausgehend vom Anfangszustand)
2. Berechne Reduktion $\mathcal{A}'$ von $\mathcal{A}$ auf die erreichbaren Zustände
3. Berechne Äquivalenzklassen von $\sim_{\mathcal{A}'}$ (mit dem Verfeinerungsalgorithmus)
4. Berechne den minimalen DFA $\widetilde{\mathcal{A}}'$.

Korrektheit: Sei $L = L(\mathcal{A})$. Es gilt $L(\mathcal{A}') = L(\mathcal{A}) = L$ und $\widetilde{\mathcal{A}}' \cong \mathcal{A}_L$.

Laufzeit: Wird bestimmt durch die Laufzeit des Verfeinerungsalgorithmus. Bei guter Implementierung ist das

$$O\big(|\Sigma| \cdot |\mathcal{A}| \cdot \log |\mathcal{A}|\big).$$

Verfeinerungsalgorithmus (Idee)

Beobachtung 5.12

Seien $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ ein DFA und $p, q \in Q$.

1. *Wenn $p \in F$ und $q \notin F$, dann $p \not\sim_{\mathcal{A}} q$.*

Verfeinerungsalgorithmus (Idee)

Beobachtung 5.12

Seien $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ ein DFA und $p, q \in Q$.

1. *Wenn $p \in F$ und $q \notin F$, dann $p \not\sim_{\mathcal{A}} q$.*
2. *Wenn es ein $a \in \Sigma$ gibt, so dass $\delta(p, a) \not\sim_{\mathcal{A}} \delta(q, a)$, so $p \not\sim_{\mathcal{A}} q$.*

Verfeinerungsalgorithmus (Idee)

Beobachtung 5.12

Seien $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ ein DFA und $p, q \in Q$.

1. *Wenn $p \in F$ und $q \notin F$, dann $p \not\sim_{\mathcal{A}} q$.*
2. *Wenn es ein $a \in \Sigma$ gibt, so dass $\delta(p, a) \not\sim_{\mathcal{A}} \delta(q, a)$, so $p \not\sim_{\mathcal{A}} q$.*

Idee des Verfeinerungsalgorithmus

Wir verfeinern schrittweise eine Partition von Q (unter Verwendung der obigen Beobachtung).

Verfeinerungsalgorithmus (Idee)

Beobachtung 5.12

Seien $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ ein DFA und $p, q \in Q$.

1. *Wenn $p \in F$ und $q \notin F$, dann $p \not\sim_{\mathcal{A}} q$.*
2. *Wenn es ein $a \in \Sigma$ gibt, so dass $\delta(p, a) \not\sim_{\mathcal{A}} \delta(q, a)$, so $p \not\sim_{\mathcal{A}} q$.*

Idee des Verfeinerungsalgorithmus

Wir verfeinern schrittweise eine Partition von Q (unter Verwendung der obigen Beobachtung).

- Wir beginnen mit der Partition in die Klassen F und $Q \setminus F$.

Verfeinerungsalgorithmus (Idee)

Beobachtung 5.12

Seien $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ ein DFA und $p, q \in Q$.

1. *Wenn $p \in F$ und $q \notin F$, dann $p \not\sim_{\mathcal{A}} q$.*
2. *Wenn es ein $a \in \Sigma$ gibt, so dass $\delta(p, a) \not\sim_{\mathcal{A}} \delta(q, a)$, so $p \not\sim_{\mathcal{A}} q$.*

Idee des Verfeinerungsalgorithmus

Wir verfeinern schrittweise eine Partition von Q (unter Verwendung der obigen Beobachtung).

- Wir beginnen mit der Partition in die Klassen F und $Q \setminus F$.
- Solange es in der aktuellen Partition eine Klasse P, zwei Zustände $p, q \in P$ und ein $a \in \Sigma$ gibt, so dass $\delta(p, a)$ und $\delta(q, a)$ in verschiedenen Klassen sind, so unterteilen wir die Klasse.

Verfeinerungsalgorithmus (Idee)

Beobachtung 5.12

Seien $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ ein DFA und $p, q \in Q$.

1. *Wenn $p \in F$ und $q \notin F$, dann $p \not\sim_{\mathcal{A}} q$.*
2. *Wenn es ein $a \in \Sigma$ gibt, so dass $\delta(p, a) \not\sim_{\mathcal{A}} \delta(q, a)$, so $p \not\sim_{\mathcal{A}} q$.*

Idee des Verfeinerungsalgorithmus

Wir verfeinern schrittweise eine Partition von Q (unter Verwendung der obigen Beobachtung).

- Wir beginnen mit der Partition in die Klassen F und $Q \setminus F$.
- Solange es in der aktuellen Partition eine Klasse P, zwei Zustände $p, q \in P$ und ein $a \in \Sigma$ gibt, so dass $\delta(p, a)$ und $\delta(q, a)$ in verschiedenen Klassen sind, so unterteilen wir die Klasse.

Wir wiederholen Schritt 2, bis keine Verfeinerung mehr möglich ist.

Verfeinerungsalgorithmus (Idee)

Beobachtung 5.12

Seien $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ ein DFA und $p, q \in Q$.

1. *Wenn $p \in F$ und $q \notin F$, dann $p \not\sim_{\mathcal{A}} q$.*
2. *Wenn es ein $a \in \Sigma$ gibt, so dass $\delta(p, a) \not\sim_{\mathcal{A}} \delta(q, a)$, so $p \not\sim_{\mathcal{A}} q$.*

Idee des Verfeinerungsalgorithmus

Wir verfeinern schrittweise eine Partition von Q (unter Verwendung der obigen Beobachtung).

- Wir beginnen mit der Partition in die Klassen F und $Q \setminus F$.
- Solange es in der aktuellen Partition eine Klasse P, zwei Zustände $p, q \in P$ und ein $a \in \Sigma$ gibt, so dass $\delta(p, a)$ und $\delta(q, a)$ in verschiedenen Klassen sind, so unterteilen wir die Klasse.

Wir wiederholen Schritt 2, bis keine Verfeinerung mehr möglich ist.

Die resultierende Partition ist genau die Partition der Zustandsmenge in die $\sim_{\mathcal{A}}$-Klassen.

Der Verfeinerungsalgorithmus (Details)

Eingabe: DFA $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$

Der Verfeinerungsalgorithmus (Details)

Eingabe: DFA $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$

Algorithmus

1. Teile Q in die Klassen $P_1 = F$ und $P_2 = Q \setminus F$ auf.

Der Verfeinerungsalgorithmus (Details)

Eingabe: DFA $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$

Algorithmus

1. Teile Q in die Klassen $P_1 = F$ und $P_2 = Q \setminus F$ auf.
2. Solange unter den vorhandenen Klassen $P_1, \ldots, P_k$
   - eine Klasse P_i,
   - eine Klasse P_j,
   - Zustände $p, q \in P_i$,
   - ein Symbol $a \in \Sigma$

   existieren, so dass $\delta(p, a) \in P_j$ und $\delta(q, a) \notin P_j$,

Der Verfeinerungsalgorithmus (Details)

Eingabe: DFA $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$

Algorithmus

1. Teile Q in die Klassen $P_1 = F$ und $P_2 = Q \setminus F$ auf.
2. Solange unter den vorhandenen Klassen $P_1, \ldots, P_k$
   - eine Klasse P_i,
   - eine Klasse P_j,
   - Zustände $p, q \in P_i$,
   - ein Symbol $a \in \Sigma$

   existieren, so dass $\delta(p, a) \in P_j$ und $\delta(q, a) \notin P_j$, spalte P_i auf in die Klassen

$$P_{i_1} = \{r \in P_i \mid \delta(r, a) \in P_j\} \text{ und}$$
$$P_{i_2} = \{r \in P_i \mid \delta(r, a) \notin P_j\}$$

Der Verfeinerungsalgorithmus (Details)

Eingabe: DFA $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$

Algorithmus

1. Teile Q in die Klassen $P_1 = F$ und $P_2 = Q \setminus F$ auf.
2. Solange unter den vorhandenen Klassen $P_1, \ldots, P_k$

   - eine Klasse P_i,
   - eine Klasse P_j,
   - Zustände $p, q \in P_i$,
   - ein Symbol $a \in \Sigma$

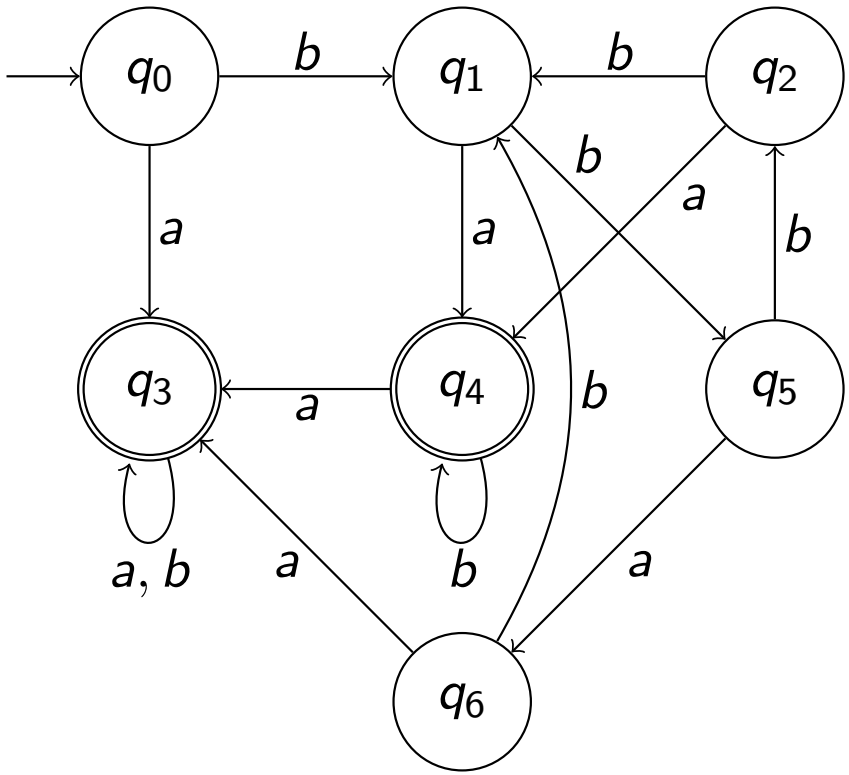
   existieren, so dass $\delta(p, a) \in P_j$ und $\delta(q, a) \notin P_j$, spalte P_i auf in die Klassen

$$P_{i_1} = \{r \in P_i \mid \delta(r, a) \in P_j\} \text{ und}$$
$$P_{i_2} = \{r \in P_i \mid \delta(r, a) \notin P_j\}$$

Ausgabe: Die gelieferten Klassen sind die $\sim_{\mathcal{A}}$-Äquivalenzklassen

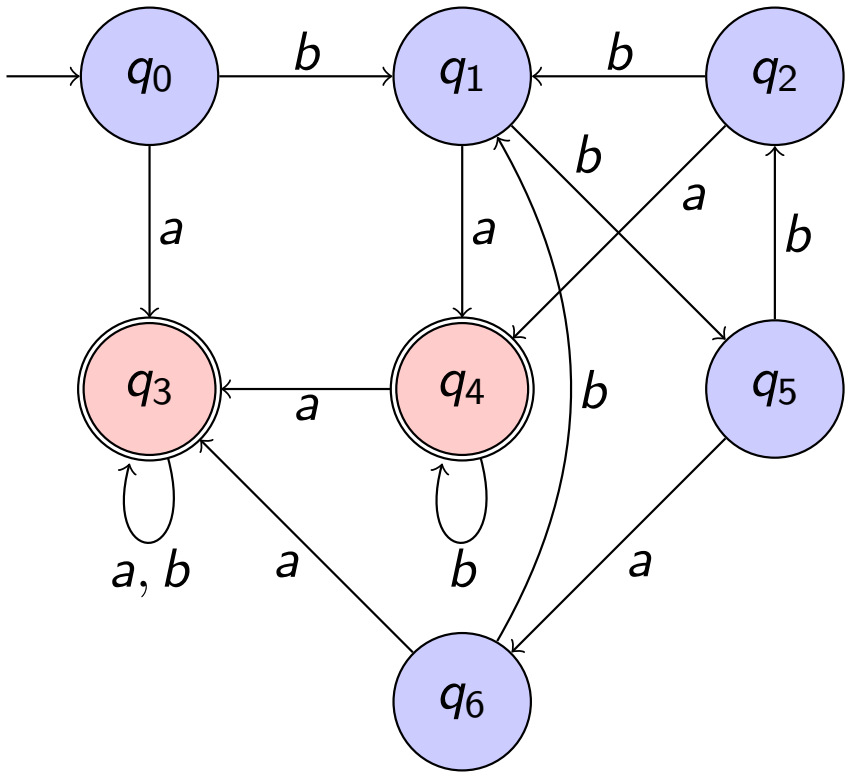
Beispiel 5.8 (Forts.)

$\mathcal{A}$



Beispiel 5.8 (Forts.)

$\mathcal{A}$



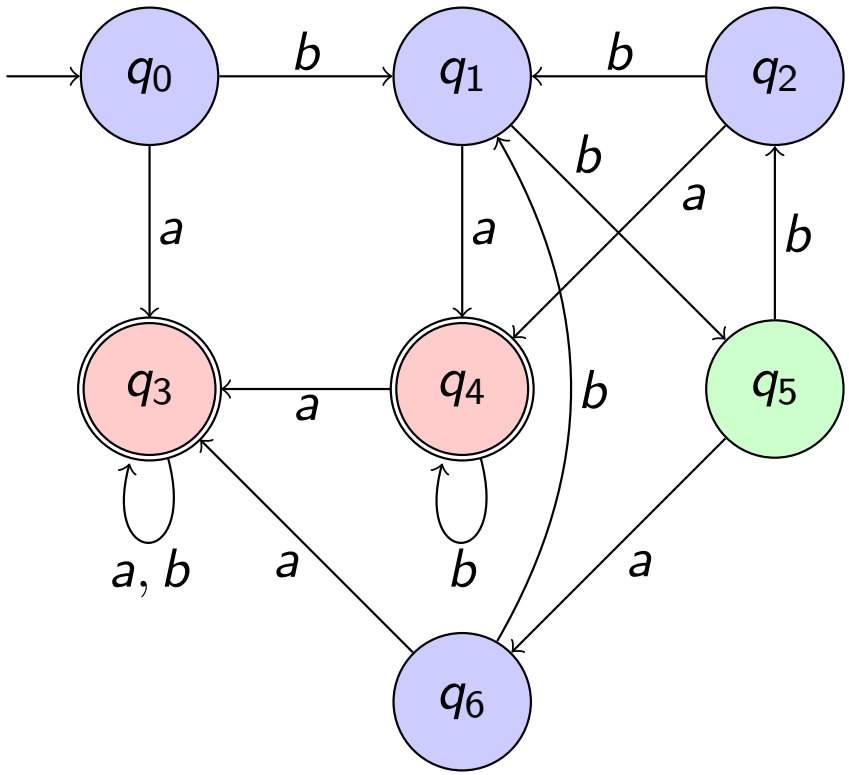
Initialisierung:

$P_1 = \{q_3, q_4\}$

$P_2 = \{q_0, q_1, q_2, q_5, q_6\}$

Beispiel 5.8 (Forts.)

$\mathcal{A}$

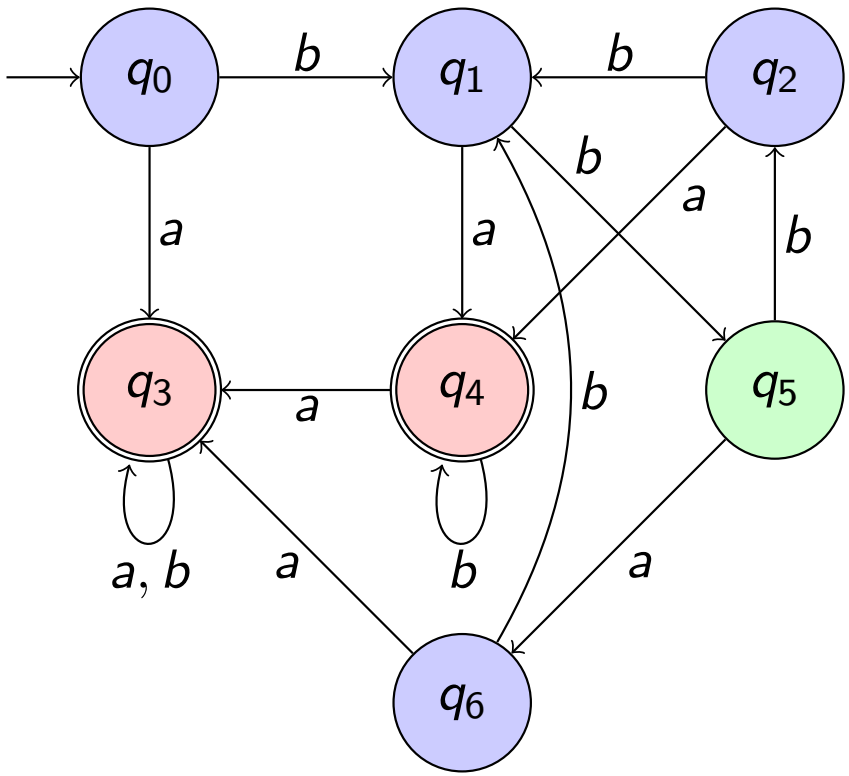


Initialisierung:
$P_1 = \{q_3, q_4\}$
$P_2 = \{q_0, q_1, q_2, q_5, q_6\}$

Verfeinere P_2 mit P_1 bezgl. a:
$P_3 = \{q_0, q_1, q_2, q_6\}$
$P_4 = \{q_5\}$

Beispiel 5.8 (Forts.)

$\mathcal{A}$



Initialisierung:

$P_1 = \{q_3, q_4\}$

$P_2 = \{q_0, q_1, q_2, q_5, q_6\}$

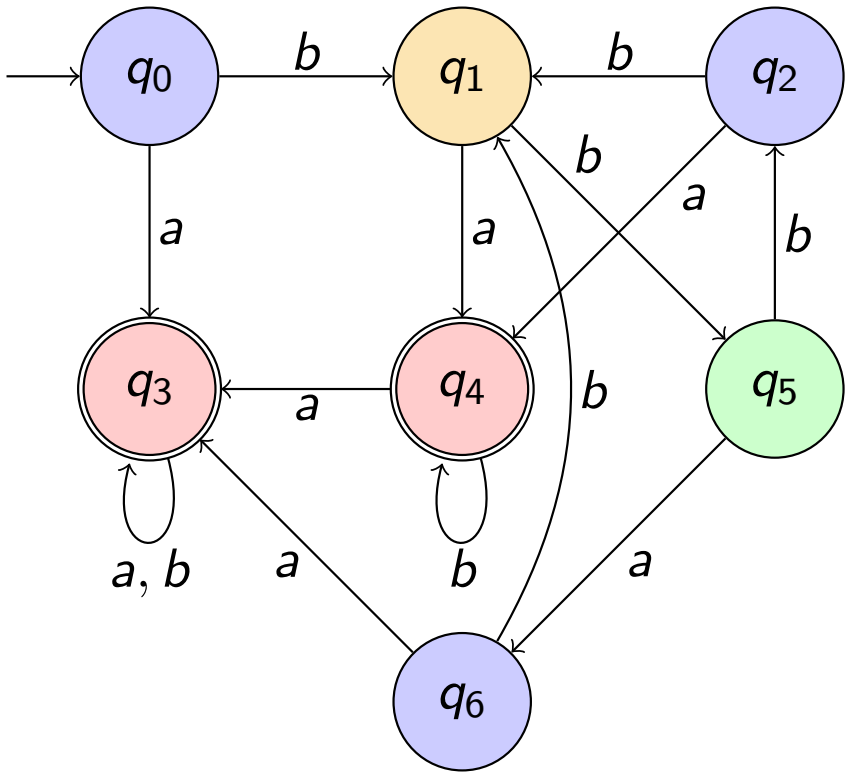
Verfeinere P_2 mit P_1 bezgl. a:

$P_3 = \{q_0, q_1, q_2, q_6\}$

$P_4 = \{q_5\}$

Beispiel 5.8 (Forts.)

$\mathcal{A}$



Initialisierung:

$P_1 = \{q_3, q_4\}$

$P_2 = \{q_0, q_1, q_2, q_5, q_6\}$

Verfeinere P_2 mit P_1 bezgl. a:

$P_3 = \{q_0, q_1, q_2, q_6\}$

$P_4 = \{q_5\}$

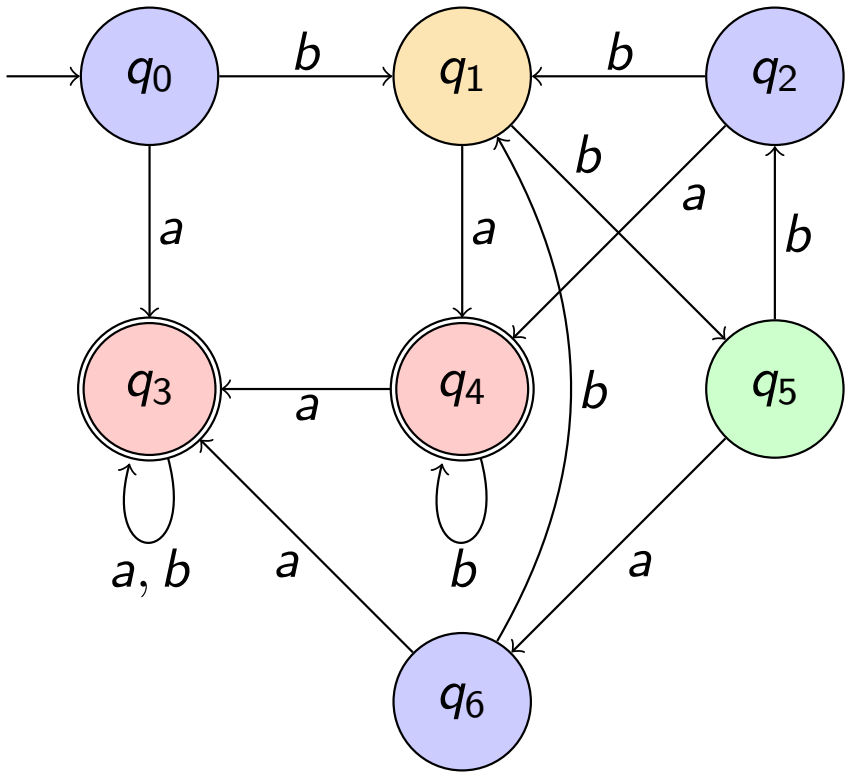
Verfeinere P_3 mit P_3 bezgl. b:

$P_5 = \{q_0, q_2, q_6\}$

$P_6 = \{q_1\}$

Beispiel 5.8 (Forts.)

$\mathcal{A}$



Initialisierung:

$P_1 = \{q_3, q_4\}$

$P_2 = \{q_0, q_1, q_2, q_5, q_6\}$

Verfeinere P_2 mit P_1 bezgl. a:

$P_3 = \{q_0, q_1, q_2, q_6\}$

$P_4 = \{q_5\}$

Verfeinere P_3 mit P_3 bezgl. b:

$P_5 = \{q_0, q_2, q_6\}$

$P_6 = \{q_1\}$

Korrektheit des Verfeinerungsalgorithmus

Sei $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ und $n := |\mathcal{A}|$.

Wir nennen eine Ausführung von Zeile 2 einen Verfeinerungsschritt.

Behauptung 1

Der Algorithmus hält nach höchstens $n-2$ Verfeinerungsschritten.

Beweis.

Vor dem ersten Verfeinerungsschritt besteht die Partition aus zwei Klassen.

In jedem Verfeinerungsschritt erhöht sich die Anzahl der Klassen.

Eine Partition einer n-elementigen Menge kann aber höchstens n Klassen enthalten, also können wir höchstens $n-2$ Verfeinerungsschritte durchführen. □

Behauptung 2

Zustände in unterschiedlichen Klassen der Partition sind nicht äquivalent.

Beweis.

Induktion über die Anzahl s der Verfeinerungsschritte.

Induktionsanfang: $s = 0$.

Zustände $p \in F = P_1$ und $q \in Q \setminus F = P_2$ sind nicht äquivalent (Beobachtung 5.12-1).

Induktionsschritt: $s \to s+1$.

Wir betrachten die Unterteilung einer Klasse P_i in Klassen

$$P_{i_1} = \{r \in P_i \mid \delta(r,a) \in P_j\} \text{ und}$$
$$P_{i_2} = \{r \in P_i \mid \delta(r,a) \notin P_j\}$$

im $(s+1)$. Verfeinerungsschritt.

Sei $p \in P_{i_1}$ und $q \in P_{i_2}$. Dann ist $\delta(p,a) \in P_j$ und $\delta(q,a) \notin P_j$.

Nach Induktionsannahme gilt $\delta(p,a) \not\sim_{\mathcal{A}} \delta(q,a)$.

Nach Beobachtung 5.12-2 gilt dann auch $p \not\sim_{\mathcal{A}} q$. □

Behauptung 3

Zustände in der gleichen Klasse sind äquivalent.

Beweis.

Wir zeigen die Umkehrung:

$$p \not\sim_{\mathcal{A}} q \implies p \text{ und } q \text{ landen in verschiedenen Klassen der Partition}$$

Wir sagen, ein Wort $x \in \Sigma^*$ trennt Zustände p und q, wenn

$$\left(p \xrightarrow{x} F \text{ und nicht } q \xrightarrow{x} F \right) \text{ oder } \left(q \xrightarrow{x} F \text{ und nicht } p \xrightarrow{x} F \right).$$

Dann gilt

$$p \not\sim_{\mathcal{A}} q \iff \exists x \in \Sigma^* : \; x \text{ trennt } p \text{ und } q.$$

Zwei Zustände p, q sind t-trennbar, für ein $t \in \mathbb{N}$, wenn es ein Wort $x \in \Sigma^*$ der Länge $|x| = t$ gibt, das p und q trennt.

Wir zeigen per Induktion über $t \in \mathbb{N}$:

> *Für alle $p, q \in Q$, sind p und q t-trennbar, so landen sie in verschiedenen Klassen der Partition.*

Induktionsanfang: $t = 0$.

Sind p und q 0-trennbar, so gilt entweder $p \in F$ und $q \in Q \setminus F$ oder $q \in F$ und $p \in Q \setminus F$.

Also gehören p und q von Anfang an zu verschiedenen Klassen.

Induktionsschritt: $t \to t+1$.

Sei x ein Wort der Länge $(t+1)$, das p und q trennt.

Dann ist $x = ax'$ für ein $a \in \Sigma$ und $x' \in \Sigma^*$ der Länge $|x'| = t$.

Das Wort x' trennt $\delta(p,a)$ von $\delta(q,a)$, also gehören nach Induktionsannahme $\delta(p,a)$ von $\delta(q,a)$ zu verschiedenen Klassen der Partition.

Angenommen: p und q gehören bis zum Schluss zur selben Klasse.

Seien $P_1, \ldots, P_k$ die Klassen nach dem letzten Verfeinerungsschritt. Sei P_i die Klasse mit $p, q \in P_i$ und P_j die Klasse mit $\delta(p,a) \in P_j$. Dann gilt $\delta(q,a) \notin P_j$.

Jetzt ist es aber noch möglich, die Klasse P_i bezgl. P_j mit a zu verfeinern.

Das widerspricht der Annahme, dass der letzte Verfeinerungsschritt bereits durchgeführt wurde. □

Berechnung der Myhill-Nerode Äquivalenz

Eingabe: Regulärerer Ausdruck, NFA, oder DFA für eine reguläre Sprache $L \subseteq \Sigma^*$

Berechnung der Myhill-Nerode Äquivalenz

Eingabe: Regulärerer Ausdruck, NFA, oder DFA für eine reguläre Sprache $L \subseteq \Sigma^*$

Algorithmus

1. Berechne minimalen DFA $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ für L.

Berechnung der Myhill-Nerode Äquivalenz

Eingabe: Regulärerer Ausdruck, NFA, oder DFA für eine reguläre Sprache $L \subseteq \Sigma^*$

Algorithmus

1. Berechne minimalen DFA $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ für L.
2. Für jeden Zustand $q \in Q$, berechne Wort $w_q \in \Sigma^*$, so dass $\mathcal{A} : q_0 \xrightarrow{w_q} q$.

Berechnung der Myhill-Nerode Äquivalenz

Eingabe: Regulärerer Ausdruck, NFA, oder DFA für eine reguläre Sprache $L \subseteq \Sigma^*$

Algorithmus

1. Berechne minimalen DFA $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ für L.
2. Für jeden Zustand $q \in Q$, berechne Wort $w_q \in \Sigma^*$, so dass $\mathcal{A} : q_0 \xrightarrow{w_q} q$.

Ergebnis: Die Myhill-Nerode Äquivalenzklassen sind $w_q/_L$, für $q \in Q$.

Berechnung der Myhill-Nerode Äquivalenz

Eingabe: Regulärerer Ausdruck, NFA, oder DFA für eine reguläre Sprache $L \subseteq \Sigma^*$

Algorithmus

1. Berechne minimalen DFA $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ für L.
2. Für jeden Zustand $q \in Q$, berechne Wort $w_q \in \Sigma^*$, so dass $\mathcal{A} : q_0 \xrightarrow{w_q} q$.

Ergebnis: Die Myhill-Nerode Äquivalenzklassen sind $w_q/_L$, für $q \in Q$.

Ergänzung

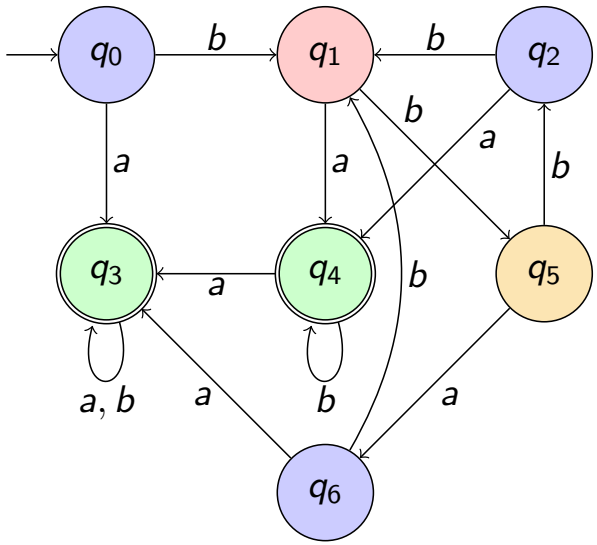
Um die Äquivalenzklasse eines Wortes $w \in \Sigma^*$ zu finden:

- Bestimme $q \in Q$, so dass $\mathcal{A} : q_0 \xrightarrow{w} q$.

Dann ist $w \in w_q/_L$.

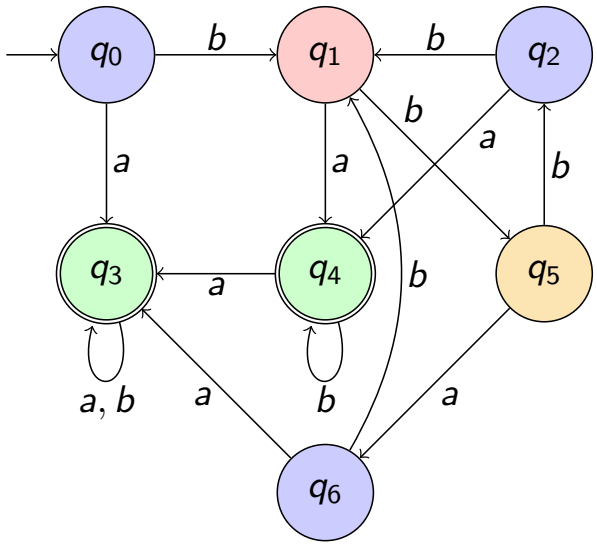
Beispiel 5.8 (Forts.)

Sprache L gegeben durch DFA:

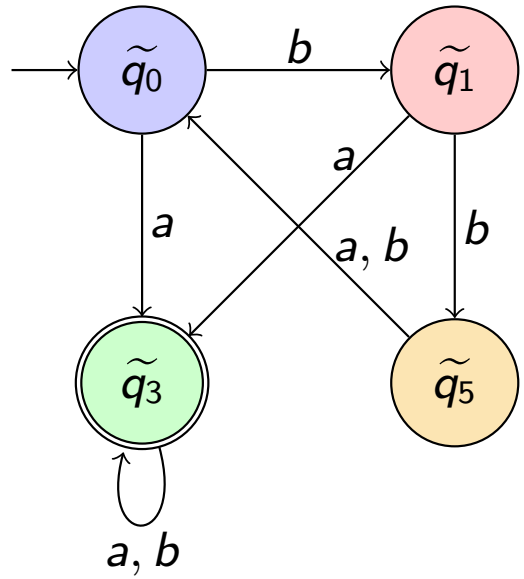


Beispiel 5.8 (Forts.)

Sprache L gegeben durch DFA:

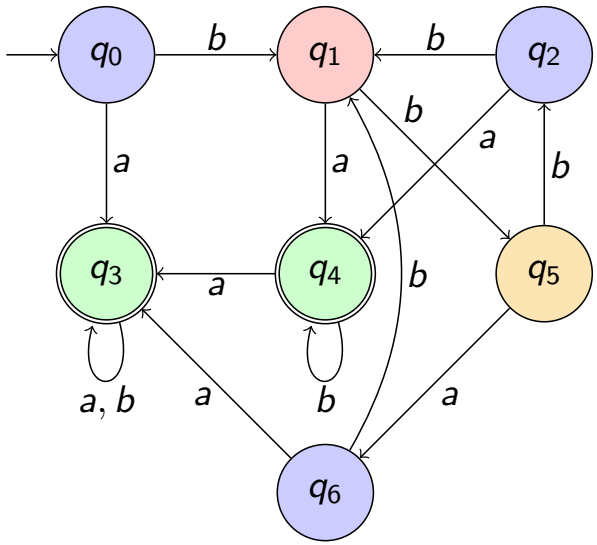


Minimaler DFA $\widetilde{\mathcal{A}}$ für L:

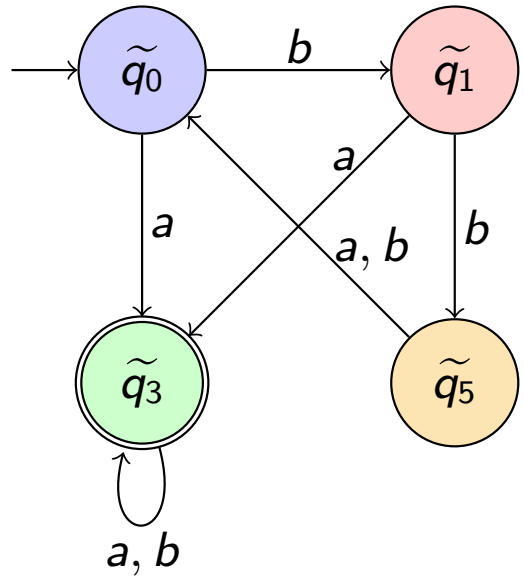


Beispiel 5.8 (Forts.)

Sprache L gegeben durch DFA:



Minimaler DFA $\widetilde{\mathcal{A}}$ für L:

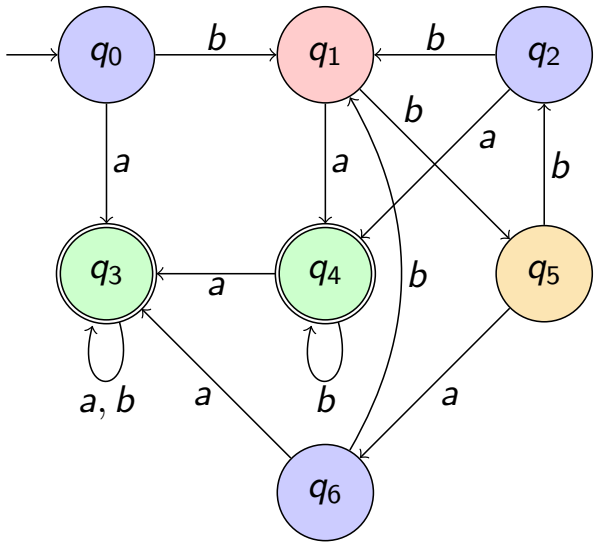


Wörter w_i mit $\widetilde{\mathcal{A}} : \widetilde{q}_0 \xrightarrow{w_i} \widetilde{q}_i$:

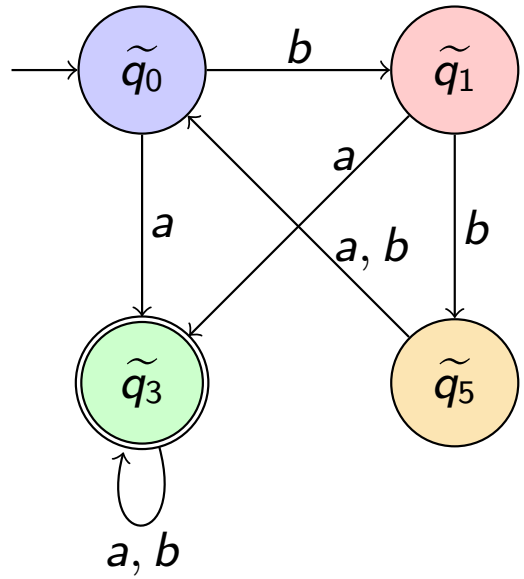
$$w_0 = \varepsilon, \qquad w_1 = b, \qquad w_3 = a, \qquad w_5 = bb$$

Beispiel 5.8 (Forts.)

Sprache L gegeben durch DFA:



Minimaler DFA $\widetilde{\mathcal{A}}$ für L:



Wörter w_i mit $\widetilde{\mathcal{A}} : \widetilde{q}_0 \xrightarrow{w_i} \widetilde{q}_i$:

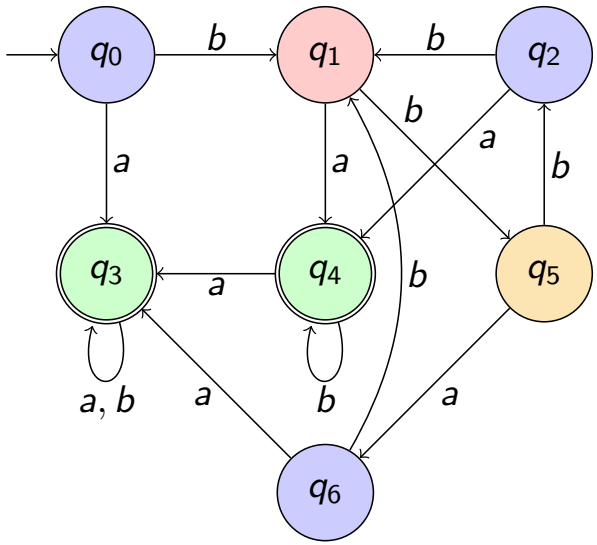
$$w_0 = \varepsilon, \qquad w_1 = b, \qquad w_3 = a, \qquad w_5 = bb$$

Die Myhill-Nerode Äquivalenzklassen von L sind also

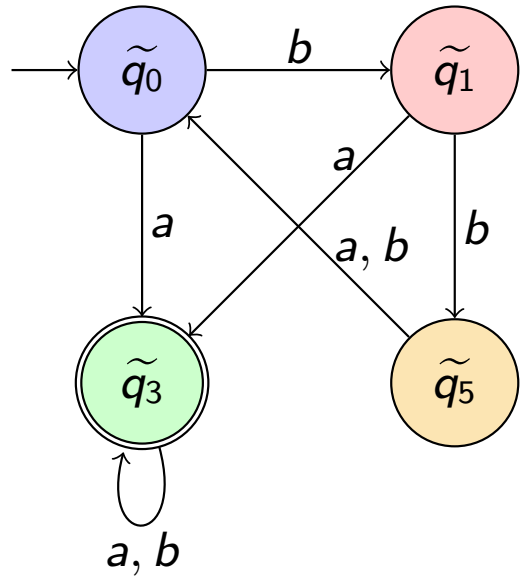
$$\varepsilon/_L, \qquad b/_L, \qquad a/_L, \qquad bb/_L$$

Beispiel 5.8 (Forts.)

Sprache L gegeben durch DFA:



Minimaler DFA $\widetilde{\mathcal{A}}$ für L:



Wörter w_i mit $\widetilde{\mathcal{A}} : \widetilde{q}_0 \xrightarrow{w_i} \widetilde{q}_i$:

$$w_0 = \varepsilon, \qquad w_1 = b, \qquad w_3 = a, \qquad w_5 = bb$$

Die Myhill-Nerode Äquivalenzklassen von L sind also

$$\varepsilon/_L, \qquad b/_L, \qquad a/_L, \qquad bb/_L$$

Für das Wort $w = bbabbbaaa$ gilt $\widetilde{\mathcal{A}} : \widetilde{q}_0 \xrightarrow{w} \widetilde{q}_3$. Also $w \in a/_L$.

Abschnitt 5.2

Weitere Algorithmen für endliche Automaten

Grundlegende algorithmische Probleme

1. Wortproblem: Gegeben NFA $\mathcal{A}$, Wort w.
   Akzeptiert $\mathcal{A}$ das Wort w, d.h. ist $w \in L(\mathcal{A})$?

Grundlegende algorithmische Probleme

1. Wortproblem: Gegeben NFA $\mathcal{A}$, Wort w.
   Akzeptiert $\mathcal{A}$ das Wort w, d.h. ist $w \in L(\mathcal{A})$?
2. Leerheitsproblem: Gegeben NFA $\mathcal{A}$.
   Akzeptiert $\mathcal{A}$ die leere Sprache, d.h. ist $L(\mathcal{A}) = \emptyset$?

Grundlegende algorithmische Probleme

1. Wortproblem: Gegeben NFA $\mathcal{A}$, Wort w.
   Akzeptiert $\mathcal{A}$ das Wort w, d.h. ist $w \in L(\mathcal{A})$?
2. Leerheitsproblem: Gegeben NFA $\mathcal{A}$.
   Akzeptiert $\mathcal{A}$ die leere Sprache, d.h. ist $L(\mathcal{A}) = \emptyset$?
3. Unendlichkeitsproblem: Gegeben NFA $\mathcal{A}$.
   Ist $L(\mathcal{A})$ unendlich?

Grundlegende algorithmische Probleme

1. Wortproblem: Gegeben NFA $\mathcal{A}$, Wort w.
   Akzeptiert $\mathcal{A}$ das Wort w, d.h. ist $w \in L(\mathcal{A})$?
2. Leerheitsproblem: Gegeben NFA $\mathcal{A}$.
   Akzeptiert $\mathcal{A}$ die leere Sprache, d.h. ist $L(\mathcal{A}) = \emptyset$?
3. Unendlichkeitsproblem: Gegeben NFA $\mathcal{A}$.
   Ist $L(\mathcal{A})$ unendlich?
4. Inklusionsproblem: Gegeben NFAs $\mathcal{A}$, $\mathcal{B}$.
   Gilt $L(\mathcal{A}) \subseteq L(\mathcal{B})$?

Grundlegende algorithmische Probleme

1. Wortproblem: Gegeben NFA $\mathcal{A}$, Wort w.
   Akzeptiert $\mathcal{A}$ das Wort w, d.h. ist $w \in L(\mathcal{A})$?
2. Leerheitsproblem: Gegeben NFA $\mathcal{A}$.
   Akzeptiert $\mathcal{A}$ die leere Sprache, d.h. ist $L(\mathcal{A}) = \emptyset$?
3. Unendlichkeitsproblem: Gegeben NFA $\mathcal{A}$.
   Ist $L(\mathcal{A})$ unendlich?
4. Inklusionsproblem: Gegeben NFAs $\mathcal{A}$, $\mathcal{B}$.
   Gilt $L(\mathcal{A}) \subseteq L(\mathcal{B})$?
5. Äquivalenzproblem: Gegeben NFAs $\mathcal{A}$, $\mathcal{B}$.
   Gilt $L(\mathcal{A}) = L(\mathcal{B})$?

Wortproblem

- Für DFAs trivial:
  Lasse Automat über Eingabewort laufen und gib Antwort gemäß Zustand am Wortende aus.

Wortproblem

- Für DFAs trivial:
  Lasse Automat über Eingabewort laufen und gib Antwort gemäß Zustand am Wortende aus.
- Für NFAs:
  Analog mit sukzessiver Berechnung der Erreichbarkeitsmengen (implizit: Potenzmengenkonstruktion).

Erreichbarkeit im Transitionsgraphen

Notation

Für einen NFA $\mathcal{A} = (Q, \Sigma, q_0, \Delta, F)$.

- $|\mathcal{A}| := |Q|$ Anzahl der Zustände.

Erreichbarkeit im Transitionsgraphen

Notation

Für einen NFA $\mathcal{A} = (Q, \Sigma, q_0, \Delta, F)$.

- $|\mathcal{A}| := |Q|$ Anzahl der Zustände.
- $||\mathcal{A}|| := |\Delta|$ Anzahl der Transitionen.

Erreichbarkeit im Transitionsgraphen

Notation

Für einen NFA $\mathcal{A} = (Q, \Sigma, q_0, \Delta, F)$.

- $|\mathcal{A}| := |Q|$ Anzahl der Zustände.
- $||\mathcal{A}|| := |\Delta|$ Anzahl der Transitionen.
- Für Zustände $p, q \in Q$ schreiben $\mathcal{A} : p \xrightarrow{*} q$, wenn es ein Wort $w \in \Sigma^*$ gibt, so dass $\mathcal{A} : p \xrightarrow{w} q$.

Erreichbarkeit im Transitionsgraphen

Notation

Für einen NFA $\mathcal{A} = (Q, \Sigma, q_0, \Delta, F)$.

- $|\mathcal{A}| := |Q|$ Anzahl der Zustände.
- $||\mathcal{A}|| := |\Delta|$ Anzahl der Transitionen.
- Für Zustände $p, q \in Q$ schreiben $\mathcal{A} : p \xrightarrow{*} q$, wenn es ein Wort $w \in \Sigma^*$ gibt, so dass $\mathcal{A} : p \xrightarrow{w} q$.

  Außerdem schreiben wir $\mathcal{A} : p \xrightarrow{*} F$, wenn $\mathcal{A} : p \xrightarrow{*} q$ für ein $q \in F$.

Erreichbarkeit im Transitionsgraphen

Notation

Für einen NFA $\mathcal{A} = (Q, \Sigma, q_0, \Delta, F)$.

- $|\mathcal{A}| := |Q|$ Anzahl der Zustände.
- $||\mathcal{A}|| := |\Delta|$ Anzahl der Transitionen.
- Für Zustände $p, q \in Q$ schreiben $\mathcal{A} : p \xrightarrow{*} q$, wenn es ein Wort $w \in \Sigma^*$ gibt, so dass $\mathcal{A} : p \xrightarrow{w} q$.

  Außerdem schreiben wir $\mathcal{A} : p \xrightarrow{*} F$, wenn $\mathcal{A} : p \xrightarrow{*} q$ für ein $q \in F$.

Beobachtung 5.13

Es gilt $\mathcal{A} : p \xrightarrow{*} q$, wenn es einen Weg von p nach q im Transitionsgraphen von $\mathcal{A}$ gibt.

Erreichbarkeit im Transitionsgraphen

Notation

Für einen NFA $\mathcal{A} = (Q, \Sigma, q_0, \Delta, F)$.

- $|\mathcal{A}| := |Q|$ Anzahl der Zustände.
- $||\mathcal{A}|| := |\Delta|$ Anzahl der Transitionen.
- Für Zustände $p, q \in Q$ schreiben $\mathcal{A} : p \xrightarrow{*} q$, wenn es ein Wort $w \in \Sigma^*$ gibt, so dass $\mathcal{A} : p \xrightarrow{w} q$.

  Außerdem schreiben wir $\mathcal{A} : p \xrightarrow{*} F$, wenn $\mathcal{A} : p \xrightarrow{*} q$ für ein $q \in F$.

Beobachtung 5.13

Es gilt $\mathcal{A} : p \xrightarrow{*} q$, wenn es einen Weg von p nach q im Transitionsgraphen von $\mathcal{A}$ gibt.

Das lässt sich mittels Tiefensuche in Linearzeit, also Zeit $O(|\mathcal{A}| + ||\mathcal{A}||)$, entscheiden (Vorlesung ADS).

Beobachtung 5.14

Sei $\mathcal{A}$ ein NFA. Dann gilt

$$L(\mathcal{A}) = \emptyset \iff \mathcal{A} : q_0 \xrightarrow{*} F.$$

Beobachtung 5.14

Sei $\mathcal{A}$ ein NFA. Dann gilt

$$L(\mathcal{A}) = \emptyset \iff \mathcal{A} : q_0 \xrightarrow{*} F.$$

Satz 5.15

Das Leerheitsproblem für NFAs lässt sich in Linearzeit entscheiden.

Beobachtung 5.14

Sei $\mathcal{A}$ ein NFA. Dann gilt

$$L(\mathcal{A}) = \emptyset \iff \mathcal{A} : q_0 \xrightarrow{*} F.$$

Satz 5.15

Das Leerheitsproblem für NFAs lässt sich in Linearzeit entscheiden.

Beweis.

Folgt aus der Beobachtung und der Tatsache, dass sich $\mathcal{A} : q_0 \xrightarrow{*} F$ mittels Tiefensuche im Transitionsgraphen in Linearzeit entscheiden lässt. □

Lemma 5.16

Sei $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ ein NFA. Dann gilt

$$L(\mathcal{A}) \text{ unendlich} \iff \exists q \in Q : \ \mathcal{A} : q_0 \xrightarrow{*} q \xrightarrow{+} q \xrightarrow{*} F,$$

wobei $q \xrightarrow{+} q$ bedeutet, dass es ein $w \in \Sigma^* \setminus \{\varepsilon\}$ mit $\mathcal{A} : q \xrightarrow{w} q$ gibt.

Lemma 5.16

Sei $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ ein NFA. Dann gilt

$$L(\mathcal{A}) \text{ unendlich} \iff \exists q \in Q : \ \mathcal{A} : q_0 \xrightarrow{*} q \xrightarrow{+} q \xrightarrow{*} F,$$

wobei $q \xrightarrow{+} q$ bedeutet, dass es ein $w \in \Sigma^* \setminus \{\varepsilon\}$ mit $\mathcal{A} : q \xrightarrow{w} q$ gibt.

Satz 5.17

Das Unendlichkeitsproblem für NFAs lässt sich in Polynomialzeit entscheiden.

Beweis von Lemma 5.16.

„$\Longrightarrow$“: Sei $L(\mathcal{A})$ unendlich. Dann gibt es ein Wort $w = a_1 \dots a_n \in L(\mathcal{A})$ für ein $n \geq |Q|$.
Sei

$$(r_0, a_1, r_1, a_2, \dots, a_n, r_n)$$

ein akzeptierender Lauf von $\mathcal{A}$ auf w.
Weil $n \geq |Q|$ gibt es $i < j$, so dass $r_i = r_j =: q$. Dann gilt

$$\mathcal{A} : q_0 \xrightarrow{a_1 \dots a_i} q \xrightarrow{a_{i+1} \dots a_j} q \xrightarrow{a_j+1 \dots a_n} r_n \in F$$

Also $\mathcal{A} : q_0 \xrightarrow{*} q \xrightarrow{+} q \xrightarrow{*} F$.

„$\Longleftarrow$“: Gelte $\mathcal{A} : q_0 \xrightarrow{*} q \xrightarrow{+} q \xrightarrow{*} F$. Seien $v, w, x \in \Sigma^*$ mit $w \neq \varepsilon$, so dass

$$\mathcal{A} : q_0 \xrightarrow{v} q \xrightarrow{w} q \xrightarrow{x} F$$

Dann gilt $vw^i x \in L(\mathcal{A})$ für alle $i \geq 0$. Also ist $L(\mathcal{A})$ unendlich.

□

Beweis von Lemma 5.17.

Folgt aus dem Lemma und der Tatsache, dass die drei Probleme

$$\mathcal{A} : q_0 \xrightarrow{*} q, \quad \mathcal{A} : q \xrightarrow{+} q, \quad \mathcal{A} : q \xrightarrow{*} F$$

alle in Linearzeit entscheidbar sind. □

Algorithmus für das Inklusionsproblem

Beobachtung 5.18

Seien $L_1, L_2 \subseteq \Sigma^*$. Dann gilt

$$L_1 \subseteq L_2 \iff L_1 \cap \overline{L}_2 = \emptyset.$$

Algorithmus für das Inklusionsproblem

Beobachtung 5.18

Seien $L_1, L_2 \subseteq \Sigma^*$. Dann gilt

$$L_1 \subseteq L_2 \iff L_1 \cap \overline{L}_2 = \emptyset.$$

Inklusionsproblem für DFAs

Eingabe: DFAs $\mathcal{A}_1$ und $\mathcal{A}_2$

Algorithmus für das Inklusionsproblem

Beobachtung 5.18

Seien $L_1, L_2 \subseteq \Sigma^*$. Dann gilt

$$L_1 \subseteq L_2 \iff L_1 \cap \overline{L}_2 = \emptyset.$$

Inklusionsproblem für DFAs

Eingabe: DFAs $\mathcal{A}_1$ und $\mathcal{A}_2$

Algorithmus

1. Berechne DFA $\overline{\mathcal{A}}_2$ mit $L(\overline{\mathcal{A}}_2) = \overline{L(\mathcal{A}_2)}$.
2. Berechne Produktautomaten $\mathcal{A}$ von $\mathcal{A}_1$ und $\overline{\mathcal{A}}_2$ mit $L(\mathcal{A}) = L(\mathcal{A}_1) \cap L(\overline{\mathcal{A}}_2)$
3. Entscheide, ob $L(\mathcal{A}) = \emptyset$.

Algorithmus für das Inklusionsproblem

Beobachtung 5.18

Seien $L_1, L_2 \subseteq \Sigma^*$. Dann gilt

$$L_1 \subseteq L_2 \iff L_1 \cap \overline{L}_2 = \emptyset.$$

Inklusionsproblem für DFAs

Eingabe: DFAs $\mathcal{A}_1$ und $\mathcal{A}_2$

Algorithmus

1. Berechne DFA $\overline{\mathcal{A}}_2$ mit $L(\overline{\mathcal{A}}_2) = \overline{L(\mathcal{A}_2)}$.
2. Berechne Produktautomaten $\mathcal{A}$ von $\mathcal{A}_1$ und $\overline{\mathcal{A}}_2$ mit $L(\mathcal{A}) = L(\mathcal{A}_1) \cap L(\overline{\mathcal{A}}_2)$
3. Entscheide, ob $L(\mathcal{A}) = \emptyset$.

Korrektheit: Folgt aus der Beobachtung

Algorithmus für das Inklusionsproblem

Beobachtung 5.18

Seien $L_1, L_2 \subseteq \Sigma^*$. Dann gilt

$$L_1 \subseteq L_2 \iff L_1 \cap \overline{L}_2 = \emptyset.$$

Inklusionsproblem für DFAs

Eingabe: DFAs $\mathcal{A}_1$ und $\mathcal{A}_2$

Algorithmus

1. Berechne DFA $\overline{\mathcal{A}}_2$ mit $L(\overline{\mathcal{A}}_2) = \overline{L(\mathcal{A}_2)}$.
2. Berechne Produktautomaten $\mathcal{A}$ von $\mathcal{A}_1$ und $\overline{\mathcal{A}}_2$ mit $L(\mathcal{A}) = L(\mathcal{A}_1) \cap L(\overline{\mathcal{A}}_2)$
3. Entscheide, ob $L(\mathcal{A}) = \emptyset$.

Korrektheit: Folgt aus der Beobachtung

Laufzeit: $O\big(|\Sigma| \cdot |\mathcal{A}_1| \cdot |\mathcal{A}_2|\big)$.

Satz 5.19

Das Inklusionsproblem für DFAs ist in Polynomialzeit entscheidbar.

Satz 5.19

Das Inklusionsproblem für DFAs ist in Polynomialzeit entscheidbar.

Korollar 5.20

Das Inklusionsproblem für NFAs ist entscheidbar.

Beobachtung 5.21

Seien $L_1, L_2 \subseteq \Sigma^*$. Dann gilt

$$L_1 = L_2 \iff L_1 \subseteq L_2 \text{ und } L_2 \subseteq L_1.$$

Beobachtung 5.21

Seien $L_1, L_2 \subseteq \Sigma^*$. Dann gilt

$$L_1 = L_2 \iff L_1 \subseteq L_2 \text{ und } L_2 \subseteq L_1.$$

Korollar 5.22

Das Äquivalenzproblem für DFAs ist in Polynomialzeit entscheidbar.

Beobachtung 5.21

Seien $L_1, L_2 \subseteq \Sigma^*$. Dann gilt

$$L_1 = L_2 \iff L_1 \subseteq L_2 \text{ und } L_2 \subseteq L_1.$$

Korollar 5.22

Das Äquivalenzproblem für DFAs ist in Polynomialzeit entscheidbar.

Korollar 5.23

Das Äquivalenzproblem für NFAs ist entscheidbar.

Model-Checking

Zu den wichtigsten Anwendungen der Automatentheorie gehört die automatische Verifikation von Hard- und Software.

Model-Checking

Zu den wichtigsten Anwendungen der Automatentheorie gehört die automatische Verifikation von Hard- und Software.

Ein grundlegendes algorithmisches Problem in diesem Bereich ist das Model-Checking Problem:

Eingabe: Modell $\mathcal{S}$ eines (Hard- oder Software) Systems und Spezifikation φ einer Systemeigenschaft.

Problem: Entscheide, ob $\mathcal{S}$ die Eigenschaft φ hat.

Model-Checking

Zu den wichtigsten Anwendungen der Automatentheorie gehört die automatische Verifikation von Hard- und Software.

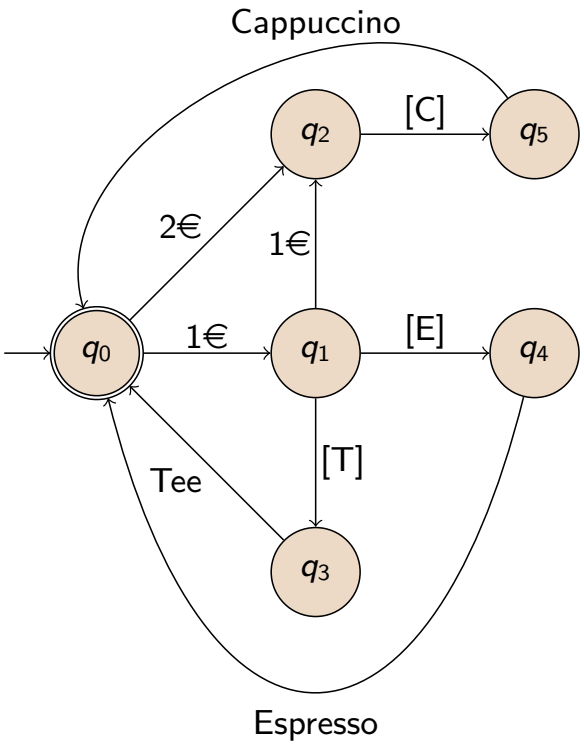
Ein grundlegendes algorithmisches Problem in diesem Bereich ist das Model-Checking Problem:

Eingabe: Modell $\mathcal{S}$ eines (Hard- oder Software) Systems und Spezifikation φ einer Systemeigenschaft.

Problem: Entscheide, ob $\mathcal{S}$ die Eigenschaft φ hat.

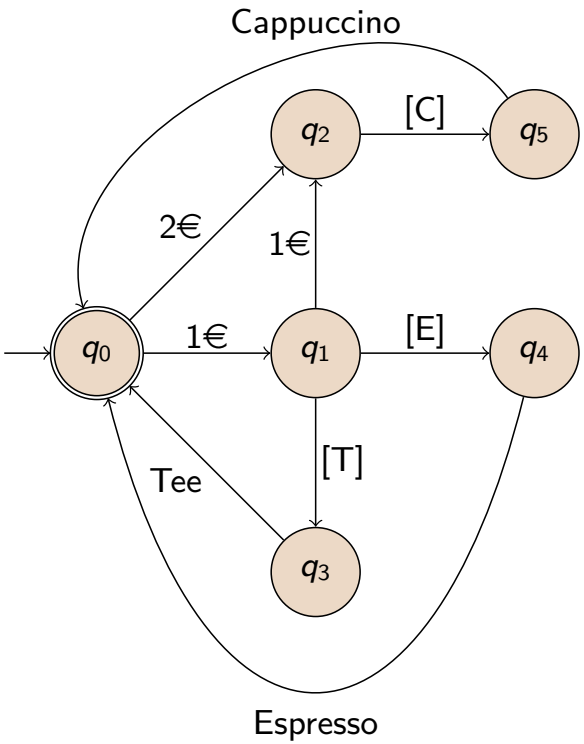
Häufig ist das Systemmodell ein NFA. Die Spezifikation der Systemeigenschaft ist normalerweise eine Formel einer geeigneten Spezifikationslogik, manchmal reicht schon ein regulärerer Ausdruck.

Beispiel 5.24



Wir betrachten wieder unseren Getränkeautomaten, modelliert durch den üblichen NFA.

Beispiel 5.24



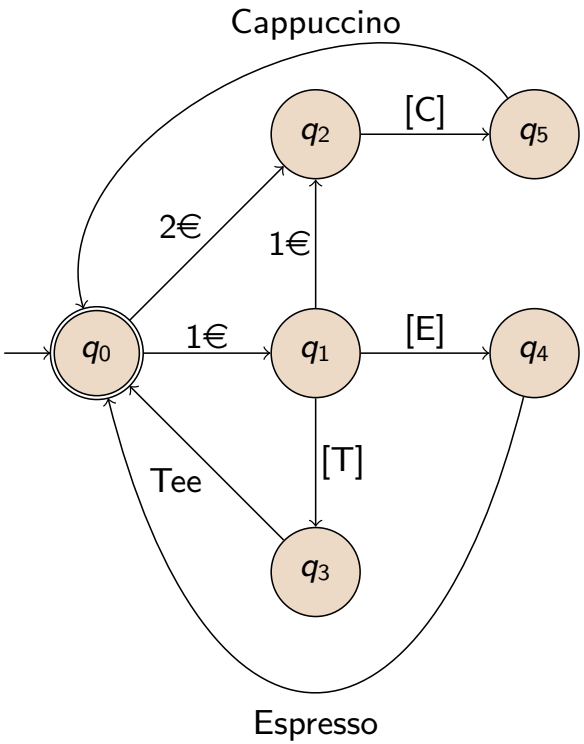
Wir betrachten wieder unseren Getränkeautomaten, modelliert durch den üblichen NFA.

Der Einfachheit halber verwenden wir als Alphabet

$$\Sigma := \{1, 2, [C], [E], [T], C, E, T\}$$

wobei $1, 2, C, E, T$ für 1€, 2€, Cappuccino, Espresso, Tee stehen.

Beispiel 5.24



Wir betrachten wieder unseren Getränkeautomaten, modelliert durch den üblichen NFA.

Der Einfachheit halber verwenden wir als Alphabet

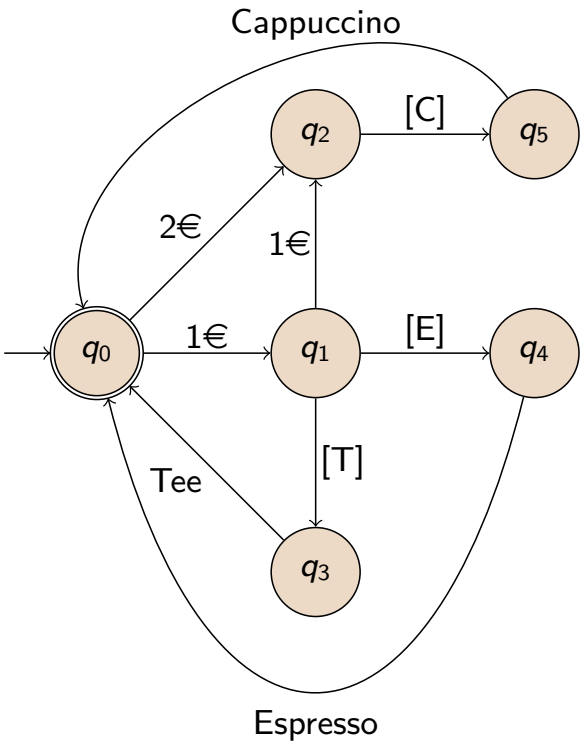
$$\Sigma := \{1, 2, [C], [E], [T], C, E, T\}$$

wobei $1, 2, C, E, T$ für 1€, 2€, Cappuccino, Espresso, Tee stehen.

Die Systemeigenschaft, die wir überprüfen wollen, ist:

> *Immer wenn Geld eingeworfen wird, kommt irgendwann auch ein Getränk raus.*

Beispiel 5.24



Wir betrachten wieder unseren Getränkeautomaten, modelliert durch den üblichen NFA.

Der Einfachheit halber verwenden wir als Alphabet

$$\Sigma := \{1, 2, [C], [E], [T], C, E, T\}$$

wobei $1, 2, C, E, T$ für 1€, 2€, Cappuccino, Espresso, Tee stehen.

Die Systemeigenschaft, die wir überprüfen wollen, ist:

> *Immer wenn Geld eingeworfen wird, kommt irgendwann auch ein Getränk raus.*

Diese Eigenschaft können wir durch folgenden regulären Ausdruck spezifizieren:

$$\Sigma^*(1+2)\Sigma^*(C+E+T)\Sigma^* + ([C]+[E]+[T]+C+E+T)^*.$$

Hier schreiben wir Σ als Abkürzung für $(1+2+[C]+[E]+[T]+C+E+T)$.

Automatentheoretisches Model-Checking Verfahren

Eingabe: NFA $\mathcal{S}$ und regulärer Ausdruck φ.

Automatentheoretisches Model-Checking Verfahren

Eingabe: NFA $\mathcal{S}$ und regulärer Ausdruck φ.

Wir sagen, dass $\mathcal{S}$ die Spezifikation φ erfüllt, wenn jeder akzeptierende Lauf von $\mathcal{A}$ die durch φ spezifizierte Eigenschaft hat

Automatentheoretisches Model-Checking Verfahren

Eingabe: NFA $\mathcal{S}$ und regulärer Ausdruck φ.

Wir sagen, dass $\mathcal{S}$ die Spezifikation φ erfüllt, wenn jeder akzeptierende Lauf von $\mathcal{A}$ die durch φ spezifizierte Eigenschaft hat

Aufgabenstellung: Überprüfe, ob $L(\mathcal{A}) \subseteq L(\varphi)$.

Automatentheoretisches Model-Checking Verfahren

Eingabe: NFA $\mathcal{S}$ und regulärer Ausdruck φ.

Wir sagen, dass $\mathcal{S}$ die Spezifikation φ erfüllt, wenn jeder akzeptierende Lauf von $\mathcal{A}$ die durch φ spezifizierte Eigenschaft hat

Aufgabenstellung: Überprüfe, ob $L(\mathcal{A}) \subseteq L(\varphi)$.

Algorithmus
Wir konstruieren einen zu φ äquivalenten NFA $\mathcal{A}_\varphi$ und führen einen Inklusionstest für die NFAs $\mathcal{S}$ und $\mathcal{A}_\varphi$ durch.