Prof. Dr. Jasmin Blanchette
Jannis Limperg

Ludwig-Maximilians-Universität München
Institut für Informatik
Abgabe bis 10. Mai 2023, 10:00 Uhr

Lösungsvorschlag zur Übung 3 zur Vorlesung

# Formale Sprachen und Komplexität

**FSK3-1** ***Konstruktion von NFAs*** (2 Punkte)

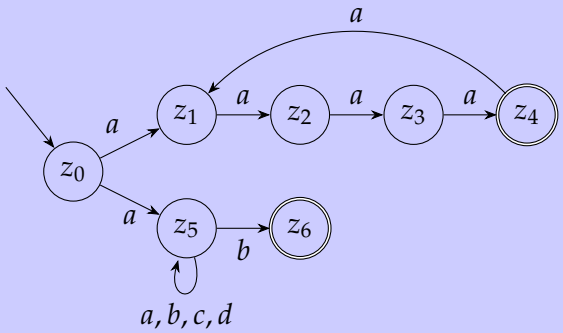
a) Geben Sie den Zustandsgraph eines NFA $A$ mit Eingabealphabet $\{a, b, c, d\}$ an, der genau alle Wörter $w$ akzeptiert, für die gilt:

- $w$ ist von der Form $a^{4i}$ für ein $i \in \mathbb{N}_{>0}$ **oder**
- $w$ fängt mit $a$ an und endet mit einem $b$.

D.h. die von $A$ akzeptierte Sprache kann geschrieben werden als

$$L(A) = \{w \in \{a\}^* \mid \#_a(w) = 4i, i \in \mathbb{N}_{>0}\} \cup \{awb \mid w \in \{a, b, c, d\}^*\}$$

**LÖSUNGSVORSCHLAG:**




b) Sei $\Sigma = \{1, 2, 3\}$ und

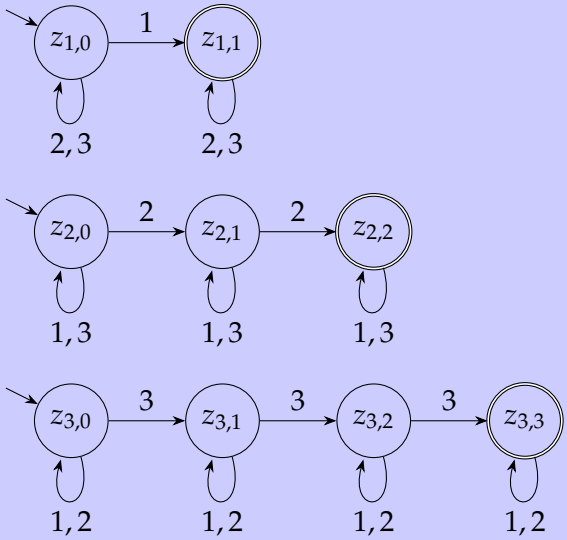
$$L = \{w \mid i \in \Sigma, w \in \Sigma^* \text{ und } \#_i(w) = i\}.$$

Das heißt die Sprache $L$ enthält genau die Wörter $w$, für die gilt: Es gibt eine Zahl $i \in \{1, 2, 3\}$ sodass das Wort $w$ das Symbol $i$ genau $i$-mal enthält.

Z.B. gilt $212323 \in L$, da das Wort 212323 genau 1-mal das Symbol 1 enthält. Ebenso gilt $2311233 \in L$, da das Wort 2311233 genau 2-mal das Symbol 2 enthält. Hingegen ist $112223 \notin L$.

Geben Sie den Zustandsgraph eines NFA $B$ an mit $L(B) = L$.

**LÖSUNGSVORSCHLAG:**




c) Sei $M$ ein beliebiger NFA über einem beliebigen Alphabet $\Sigma$ und sei $a \notin \Sigma$. Konstruieren Sie allgemein daraus einen NFA $N$ über $\Sigma \cup \{a\}$ mit

$$L(N) = \{wa \mid w \in L(M)\}$$

Anders gesagt: Der Automat $N$ soll ein Wort genau dann akzeptieren, wenn das Wort von der Form $wa$ ist und $w$ vom Automaten $M$ akzeptiert wird.

**LÖSUNGSVORSCHLAG:**

Sei $M = (\{z_0, \ldots, z_n\}, \Sigma, \delta, S, E)$ ein NFA und $a \notin \Sigma$.

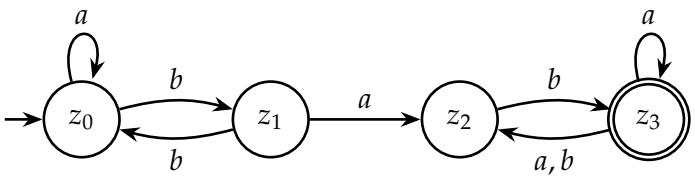
Wir definieren $N = (\{z_0, \ldots, z_n, z_E\}, \Sigma \cup \{a\}, \delta', S, \{z_E\})$ (für $z_E$ neu) mit

$$\begin{array}{lll} \delta'(z_i, x) & = \delta(z_i, x) & \text{für } 0 \leq i \leq n \text{ und alle } x \in \Sigma \\ \delta'(z_i, a) & = \{z_E\} & \text{für alle } z_i \in E \\ \delta'(z_i, a) & = \emptyset & \text{für alle } z_i \notin E \\ \delta'(z_E, x) & = \emptyset & \text{für alle } x \in \Sigma \cup \{a\} \end{array}$$

Der Automat $N$ „startet“ also genau wie $M$. Beim Lesen eines $a$ muss er jedoch in den Endzustand $z_E$ wechseln (mit einem Übergang $z_i \xrightarrow{a} z_E$), falls sich $N$ in einem der Endzustände von $M$ befindet. Falls nicht, steht das Zeichen $a$ an der falschen Position und das Wort wird nicht erkannt. Befindet sich $N$ im Endzustand und liest weitere Zeichen $x$, so akzeptiert er auch nicht.

**FSK3-2** ***Läufe und Potenzmengenkonstruktion*** (2 Punkte)

Für diese Aufgabe betrachten wir folgenden NFA $C$ über dem Alphabet $\{a, b\}$:




a) Geben Sie einen akzeptierenden Lauf des NFA $C$ auf dem Wort $w = aaabbbabab$ an.

**LÖSUNGSVORSCHLAG:**

$$z_0 \xrightarrow{a} z_0 \xrightarrow{a} z_0 \xrightarrow{a} z_0 \xrightarrow{b} z_1 \xrightarrow{b} z_0 \xrightarrow{b} z_1 \xrightarrow{a} z_2 \xrightarrow{b} z_3 \xrightarrow{a} z_2 \xrightarrow{b} z_3$$

ist ein solcher Lauf.

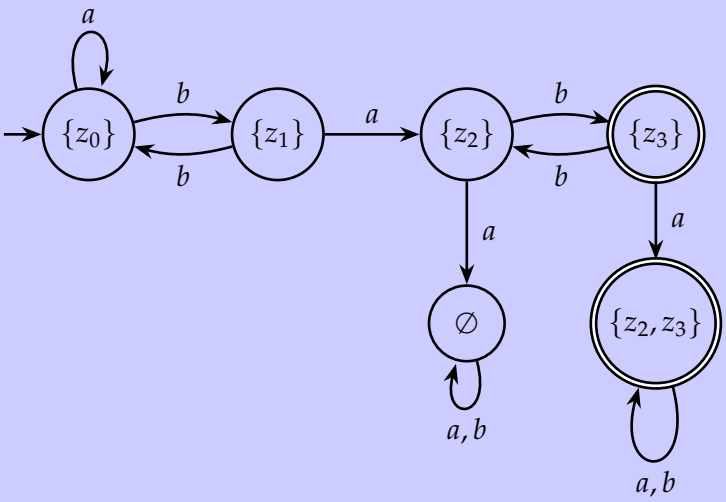
b) Berechnen Sie zum NFA $C$ einen äquivalenten DFA $D$ mit der Potenzmengenkonstruktion. Geben Sie nur den Zustandsgraphen des vom Startzustand erreichbaren Teils des Automaten an. Vergessen Sie nicht, den Startzustand und die Endzustände zu markieren.

**LÖSUNGSVORSCHLAG:**

Potenzmengenkonstruktion, wobei nur die erreichbaren Zustände betrachtet werden:

| Zustand / Nachfolge bei | $a$ | $b$ |
|---|---|---|
| $\{z_0\}$ | $\{z_0\}$ | $\{z_1\}$ |
| $\{z_1\}$ | $\{z_2\}$ | $\{z_0\}$ |
| $\{z_2\}$ | $\emptyset$ | $\{z_3\}$ |
| $\emptyset$ | $\emptyset$ | $\emptyset$ |
| $\{z_3\}$ | $\{z_2, z_3\}$ | $\{z_2\}$ |
| $\{z_2, z_3\}$ | $\{z_2, z_3\}$ | $\{z_2, z_3\}$ |

Startzustand $\{z_0\}$ und Endzustände $\{z_3\}, \{z_2, z_3\}$.




c) Geben Sie den Lauf des DFA $D$ auf dem Wort $w$ an. Ist der Lauf akzeptierend?

**LÖSUNGSVORSCHLAG:**

$\{z_0\} \xrightarrow{a} \{z_0\} \xrightarrow{a} \{z_0\} \xrightarrow{a} \{z_0\} \xrightarrow{b} \{z_1\} \xrightarrow{b} \{z_0\} \xrightarrow{b} \{z_1\} \xrightarrow{a} \{z_2\} \xrightarrow{b} \{z_3\} \xrightarrow{a} \{z_2, z_3\} \xrightarrow{b} \{z_2, z_3\}$

Der Lauf ist akzeptierend, da $\{z_2, z_3\}$ ein Endzustand ist.

Welcher Zusammenhang besteht zwischen dem von Ihnen in der Teilaufgabe a) gefundenen Lauf des Automaten $C$ und dem in dieser Teilaufgabe gefundenen Lauf des Automaten $D$ auf dem Wort $w$?

**LÖSUNGSVORSCHLAG:**

Jeder Zustand im Lauf des NFA $C$ ist Element des entsprechenden Zustands im Lauf des DFA $D$. (Zur Erinnerung: Die Zustände des DFA sind Mengen von Zuständen des NFA.)

Da jeder Zustand von $D$ auflistet, in welchem Zustand $C$ an den entsprechenden Wortpositionen sein kann, simuliert $D$ in gewisser Weise alle möglichen Läufe des Automaten $C$ auf dem Wort.

**FSK3-3** ***Tokenizer*** (0 Punkte)

Ein Einsatzgebiet für endliche Automaten sind Tokenizer. Diese werden verwendet, um den Quelltext einer Programmiersprache in syntaktische Einheiten (Tokens) zu zerlegen. Ein Token ist beispielsweise ein Schlüsselwort, ein Bezeichner, ein Operator, etc.

Zum Beispiel wird das Programm

`if(x==y){z=x};`

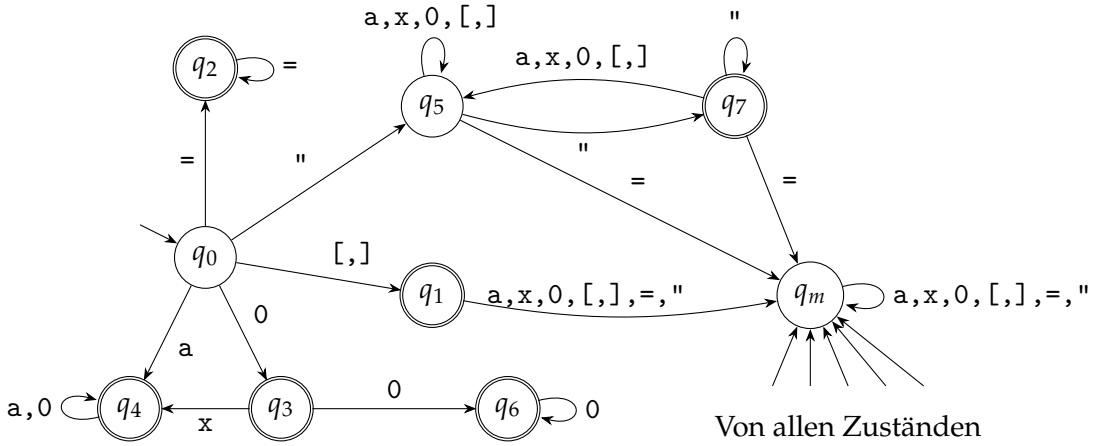
zerlegt in

„`if`“ „`(`“ „`x`“ „`==`“ „`y`“ „`)`“ „`{`“ „`z`“ „`=`“ „`x`“ „`}`“ „`;`“

In dieser Aufgabe erstellen wir einen Tokenizer, indem wir die möglichen Tokens als reguläre Sprache auffassen.

a) Um alle Schritte sinnvoll per Hand rechnen zu können, arbeiten wir mit einem reduzierten Alphabet (`[]` statt `()` oder {}, weniger Buchstaben aus dem Alphabet, nur eine Ziffer, ...):
$\Sigma = \{$`a,x,0,[,],=,"`$\}$

Die Sprache der möglichen Tokens ist als DFA $A$ gegeben:




Von allen Zuständen
alle fehlenden Buchstaben

Um das erste Token aus einem String zu identifizieren, wird $A$ vom Anfang des Strings aus laufen gelassen. Wenn der Lauf nie in einen Endzustand kommt, meldet der Tokenizer einen Fehler. Ansonsten wird die *letzte* Position, in welcher der Automat in einem Endzustand war, als Token-Ende genommen.

Zum Beispiel ist bei Eingabe `==aa[` der Lauf $q_0 \xrightarrow{=} q_2 \xrightarrow{=} q_2 \xrightarrow{a} q_m \xrightarrow{a} q_m \xrightarrow{[} q_m$. Da $q_2$ akzeptierend ist (aber $q_m$ nicht), ist das erkannte Token `==`.

Notieren Sie bei folgenden Strings die Zustände, die $A$ bei Verarbeitung dieser Strings annehmen wird (die Läufe) und geben Sie je die Ausgabe des Tokenizers an. Bezüglich der Ausgabe reicht es, sofern der Tokenizer keinen Fehler zurückgibt, nur das erste erkannte Token anzugeben.

- `aa==a`
- `a[0]`
- `a[[[[]`
- `"a[0]"ax"`
- `"a=[0]"ax"`
- `"a[0]"a=x"`

**LÖSUNGSVORSCHLAG:**

`aa==a`: $\Rightarrow q_0 \xrightarrow{a} q_4 \xrightarrow{a} q_4 \xrightarrow{=} q_m \xrightarrow{=} q_m \xrightarrow{a} q_m \Rightarrow$ Token: `aa`

`a[0]`: $\Rightarrow q_0 \xrightarrow{a} q_4 \xrightarrow{[} q_m \xrightarrow{0} q_m \xrightarrow{]} q_m \Rightarrow$ Token: `a`

`a[[[[]`: $\Rightarrow q_0 \xrightarrow{a} q_4 \xrightarrow{[} q_m \xrightarrow{[} q_m \xrightarrow{[} q_m \xrightarrow{[} q_m \xrightarrow{]} q_m \Rightarrow$ Token: `a`

`"a[0]"ax"`: $\Rightarrow q_0 \xrightarrow{''} q_5 \xrightarrow{a} q_5 \xrightarrow{[} q_5 \xrightarrow{0} q_5 \xrightarrow{]} q_5 \xrightarrow{''} q_7 \xrightarrow{a} q_5 \xrightarrow{x} q_5 \xrightarrow{''} q_7$
$\Rightarrow$ Token: `"a[0]"ax"`

`"a=[0]"ax"`: $\Rightarrow q_0 \xrightarrow{''} q_5 \xrightarrow{a} q_5 \xrightarrow{=} q_m \xrightarrow{[} q_m \xrightarrow{0} q_m \xrightarrow{]} q_m \xrightarrow{''} q_m \xrightarrow{a} q_m \xrightarrow{x} q_m \xrightarrow{''} q_m$
$\Rightarrow$ Fehler

`"a[0]"a=x"`: $\Rightarrow q_0 \xrightarrow{''} q_5 \xrightarrow{a} q_5 \xrightarrow{[} q_5 \xrightarrow{0} q_5 \xrightarrow{]} q_5 \xrightarrow{''} q_7 \xrightarrow{a} q_5 \xrightarrow{=} q_m \xrightarrow{x} q_m \xrightarrow{''} q_m$
$\Rightarrow$ Token: `"a[0]"`

b) Bestimmen Sie asymptotisch (in $O$-Notation), wie viele Schritte der Tokenizer-Automat braucht, um ein Token aus einem String der Länge $n$ zu extrahieren.

**LÖSUNGSVORSCHLAG:** $O(n)$, da die Verarbeitung von einem Zeichen $O(1)$ ist und zweimal über den String gelaufen werden muss: Einmal zur Verarbeitung des Strings und einmal bei der Suche nach dem letzten Zustand, der ein Endzustand ist. (Man kann sich den jeweils letzten Endzustand natürlich auch merken, dann muss man nur einmal über den String laufen. Das ändert aber an der asymptotischen Laufzeit nichts.)

c) Um mehrere Tokens zu extrahieren, wird das gefundene Token von dem String entfernt und wieder von vorne ein Token gesucht. Wenn der verbleibende String leer ist, ist der Tokenizer fertig.

Beispiel: Bei der oben genannten Eingabe `==aa[` mit dem ersten Token `==`, ist der Reststring nach dem Entfernen `aa[`, das zweite Token dann also `aa`.

Zerlegen Sie mit diesem Algorithmus den String `a="ax0"aa[0]=a` in alle Tokens.

**LÖSUNGSVORSCHLAG:**

String: `a="ax0"aa[0]=a`

Automat $q_0 \xrightarrow{a} q_4 \xrightarrow{=} q_m \xrightarrow{"} q_m \xrightarrow{a} q_m \xrightarrow{x} q_m \xrightarrow{0} q_m \xrightarrow{"} q_m \xrightarrow{a} q_m \xrightarrow{a} q_m \xrightarrow{[} q_m \xrightarrow{0} q_m \xrightarrow{]} q_m \xrightarrow{=} q_m \xrightarrow{a} q_m$

Token: `a` Reststring: `="ax0"aa[0]=a`

Automat $q_0 \xrightarrow{=} q_2 \xrightarrow{"} q_m \xrightarrow{a} q_m \xrightarrow{x} q_m \xrightarrow{0} q_m \xrightarrow{"} q_m \xrightarrow{a} q_m \xrightarrow{a} q_m \xrightarrow{[} q_m \xrightarrow{0} q_m \xrightarrow{]} q_m \xrightarrow{=} q_m \xrightarrow{a} q_m$

Token: `=` Reststring: `"ax0"aa[0]=a`

Automat $q_0 \xrightarrow{"} q_5 \xrightarrow{a} q_5 \xrightarrow{x} q_5 \xrightarrow{0} q_5 \xrightarrow{"} q_7 \xrightarrow{a} q_5 \xrightarrow{a} q_5 \xrightarrow{[} q_5 \xrightarrow{0} q_5 \xrightarrow{]} q_5 \xrightarrow{=} q_m \xrightarrow{a} q_m$

Token: `"ax0"` Reststring: `aa[0]=a`

Automat $q_0 \xrightarrow{a} q_4 \xrightarrow{a} q_4 \xrightarrow{[} q_m \xrightarrow{0} q_m \xrightarrow{]} q_m \xrightarrow{=} q_m \xrightarrow{a} q_m$

Token: `aa` Reststring: `[0]=a`

Automat $q_0 \xrightarrow{[} q_1 \xrightarrow{0} q_m \xrightarrow{]} q_m \xrightarrow{=} q_m \xrightarrow{a} q_m$

Token: `[` Reststring: `0]=a`

Automat $q_0 \xrightarrow{0} q_3 \xrightarrow{]} q_m \xrightarrow{=} q_m \xrightarrow{a} q_m$

Token: `0` Reststring: `]=a`

Automat $q_0 \xrightarrow{]} q_1 \xrightarrow{=} q_m \xrightarrow{a} q_m$

Token: `]` Reststring: `=a`

Automat $q_0 \xrightarrow{=} q_2 \xrightarrow{a} q_m$

Token: `=` Reststring: `a`

Automat $q_0 \xrightarrow{a} q_4$

Token: `a` Reststring: $\varepsilon$

Damit ist die Liste der Tokens: „`a`“, „`=`“, „`"ax0"`“, „`aa`“, „`[`“, „`0`“, „`]`“, „`=`“, „`a`“

d) Tatsächlich müssen wir den String nicht verändern, sondern, wenn ein Token gefunden wurde, nur den Automat an der nächsten Position im String starten. Wir ‚kürzen' den String also in $O(1)$.

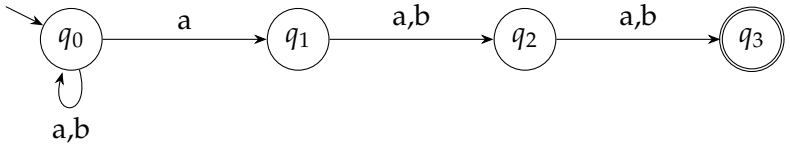
Wie viele Schritte brauchen wir dann asymptotisch, um alle Tokens aus einem String der Länge $n$ zu finden? (Hinweis: Es ist nicht $O(n)$. Man könnte das Verfahren aber optimieren, um eine Laufzeit von $O(n)$ zu erreichen.)

**LÖSUNGSVORSCHLAG:** $O(n^2)$, da es bis zu $n$ Tokens geben kann und jedes in $O(n)$ Schritten gefunden wird.

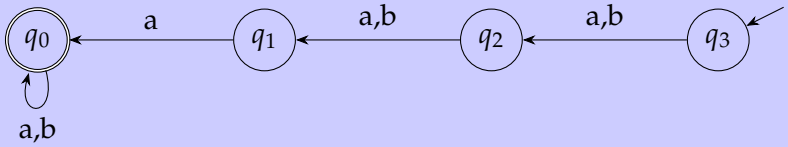
**FSK3-4** ***Umgedrehte Sprache*** (0 Punkte)

Sei $T$ die Funktion, die aus einem NFA $A = (Z, \Sigma, \delta, S, E)$ einen NFA $T(A) = (Z, \Sigma, \delta', E, S)$ erzeugt, wobei $p \in \delta'(q, a) \iff q \in \delta(p, a)$.

a) Berechnen Sie den Zustandsgraph des Automaten $B = T(A)$ für folgenden Automaten $A$:



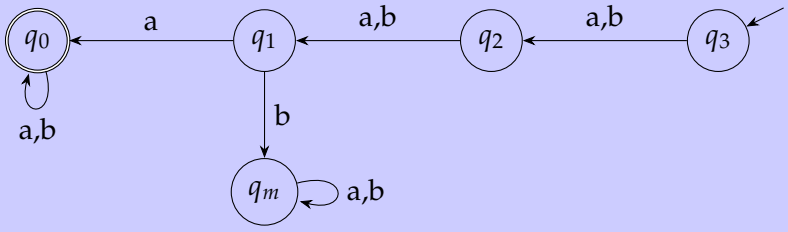

**LÖSUNGSVORSCHLAG:**




b) Geben Sie den Zustandsgraph eines DFA $C$ mit $L(B) = L(C)$ an. (Sie dürfen die Potenzmengenkonstruktion nutzen, müssen aber nicht.)

**LÖSUNGSVORSCHLAG:**

Fast wie in b), aber $q_1$ benötigt noch einen ausgehenden Übergang mit $b$. Darum Mülleimerzustand hinzufügen.



c) Zeigen Sie: Für jeden NFA $A$ ist $L(T(A)) = \{w \mid \overline{w} \in L(A)\}$. Dabei steht $\overline{w}$ wie in der Vorlesung für das rückwärts gelesene Wort $w$.

Hinweis: Es kann hilfreich sein, zuerst $L(T(A)) \subseteq \{w \mid \overline{w} \in L(A)\}$ und dann $\{w \mid \overline{w} \in L(A)\} \subseteq L(T(A))$ zu zeigen.

**LÖSUNGSVORSCHLAG:**

Wir zeigen zunächst $L(T(A)) \subseteq \{w \mid \overline{w} \in L(A)\}$.

Sei $w \in L(T(A))$, $n = |w|$. Damit gibt es einen akzeptierenden Lauf $\varrho$ von $T(A)$ auf $w$.

$\overline{\varrho}$ ist ein akzeptierender Lauf von $A$ auf $\overline{w}$, da

- $\overline{\varrho}$ in einem Startzustand von $A$ anfängt (da $\varrho$ in einem Endzustand von $T(A)$ endet)
- $\overline{\varrho}$ in einem Endzustand von $A$ endet (da $\varrho$ in einem Startzustand von $T(A)$ anfängt)
- $\forall i \in \{1, \ldots, n\}.\ \overline{\varrho}[i] \in \delta(\overline{\varrho}[i-1], \overline{w}[i])$, denn:
  Es ist $\forall i \in \{1, \ldots, n\}.\ w[i] = \overline{w}[n-i+1]$ sowie $\forall i \in \{0, \ldots, n\}.\ \varrho[i] = \overline{\varrho}[n-i]$.
  Damit ist $\overline{\varrho}[i] = \varrho[n-i], \overline{\varrho}[i-1] = \varrho[n-(i-1)] = \varrho[n-i+1], \overline{w}[i] = w[n-i+1]$
  Damit $\forall i \in \{1, \ldots, n\}.\ \overline{\varrho}[i] \in \delta(\overline{\varrho}[i-1], \overline{w}[i]) \longleftrightarrow$
  $\forall i \in \{1, \ldots, n\}.\ \varrho[n-i] \in \delta(\varrho[n-i+1], w[n-i+1]) \longleftrightarrow$
  $\forall i \in \{1, \ldots, n\}.\ \varrho[n-i+1] \in \delta'(\varrho[n-i], w[n-i+1]) \longleftrightarrow$
  $\forall j \in \{0, \ldots, n-1\}.\ \varrho[j+1] \in \delta'(\varrho[j], w[j+1]) \longleftrightarrow$
  $\forall i \in \{1, \ldots, n\}.\ \varrho[i] \in \delta'(\varrho[i-1], w[i])$
  Die letzte Aussage gilt, da $\varrho$ ein Lauf von $T(A)$ auf $w$ ist.

Somit ist $\overline{w} \in L(A)$ und $L(T(A)) \subseteq \{w \mid \overline{w} \in L(A)\}$.

Nun zeigen wir, dass $\{w \mid \overline{w} \in L(A)\} \subseteq L(T(A))$ ist.

Wir bemerken dazu zunächst, dass die oben bewiesene Aussage auch für $T(A)$ gilt. (Wir haben sie ja für alle Automaten bewiesen). Also gilt $L(T(T(A))) \subseteq \{w \mid \overline{w} \in L(T(A))\}$. Nun ist aber $T(T(A)) = A$: wenn man den umgedrehten Automaten nochmal umdreht, erhält man wieder den ursprünglichen Automaten. Somit gilt $L(A) \subseteq \{w \mid \overline{w} \in L(T(A))\}$.

Weiterhin gilt für alle Mengen $X \subseteq Y$ und all Funktionen $f$: $\{f(x) \mid x \in X\} \subseteq \{f(y) \mid y \in Y\}$. Wenden wir diese Regel auf die obige Aussage an, wobei wir $f(w) = \overline{w}$ wählen, so erhalten wir $\{\overline{w} \mid w \in L(A)\} \subseteq \{\overline{w} \mid \overline{w} \in L(T(A))\}$. Die rechte Menge ist genau $L(T(A))$, somit ist die gewünschte Aussage bewiesen.

**Ein alternativer Lösungsvorschlag:**

Diese Lösung verwendet direkt $\delta$ und $\widehat{\delta}$. Wir zeigen für alle $q, p \in Z$ und alle $w \in \Sigma^*$:

$$q \in \widehat{\delta}(p, \overline{w}) \iff p \in \widehat{\delta'}(q, w) \tag{1}$$

durch Induktion über die Länge $|w|$.

Zur Erinnerung: für $X \subseteq Z$, $a \in \Sigma$, $w \in \Sigma^*$ und $z \in Z$ gilt

$$\widehat{\delta}(X, \varepsilon) = X, \quad \widehat{\delta}(X, aw) = \widehat{\delta}(\bigcup_{z \in X} \delta(z, a), w), \quad \widehat{\delta}(z, w) := \widehat{\delta}(\{z\}, w)$$

**Basis:** $w = \varepsilon$**:** Die Definition von $\widehat{\delta}$ liefert sofort $\widehat{\delta}(q, \varepsilon) = \{q\} = \widehat{\delta'}(q, \varepsilon)$

**Schritt:** $a_1 \cdots a_n \to a_1 \cdots a_{n+1}$**:**

$q \in \widehat{\delta}(p, a_{n+1} \cdots a_1)$

g.d.w. $\exists z \in Z : z \in \delta(p, a_{n+1}) \wedge q \in \widehat{\delta}(z, a_n \cdots a_1)$ (Definition von $\widehat{\delta}$)

g.d.w. $\exists z \in Z : z \in \delta(p, a_{n+1}) \wedge z \in \widehat{\delta'}(q, a_1 \cdots a_n)$ (mit Induktionshypothese)

g.d.w. $\exists z \in Z : p \in \delta'(z, a_{n+1}) \wedge z \in \widehat{\delta'}(q, a_1 \cdots a_n)$ (Definition von $\delta'$)

g.d.w. $p \in \widehat{\delta'}(q, a_1 \cdots a_{n+1})$ (Definition von $\widehat{\delta'}$)

Schließlich zeigen wir die Behauptung:

$\overline{w} \in L(A)$

g.d.w. $\exists q \in E, p \in S : q \in \widehat{\delta}(p, \overline{w})$

g.d.w. $\exists q \in E, p \in S : p \in \widehat{\delta'}(q, w)$ (mit Gleichung (1))

g.d.w. $w \in L(T(A))$