Grundlagen der theoretischen Informatik

Kurt Sieber

Fakultät IV, Department ETI
Universität Siegen

SS 2013

Vorlesung vom 16.04.2013

Konventionen

Als typische Bezeichnungen verwenden wir (bisher):

- a, b, c für Zeichen
- u, v, w, x, y, z für Wörter
- Σ für ein Alphabet
- L für eine Sprache
- A, B für Automaten
- p, q, r, s für Zustände, speziell s für den Startzustand
- P, Q, F für Zustandsmengen, speziell Q für die Menge aller Zustände, F für die Menge der Endzustände

Bisher: Determistische endliche Automaten

- Es gibt *genau einen* Startzustand s.
- Für jeden Zustand $q \in Q$ und jedes Zeichen $a \in \Sigma$ gilt:

  Von q geht *genau ein* mit a markierter Pfeil aus.

  Formal: $\delta : Q \times \Sigma \to Q$ ist eine totale Funktion.

Jetzt: Nichtdeterministische endliche Automaten

- Es darf *beliebig viele* Startzustände geben.
- Für jeden Zustand $q \in Q$ und jedes Zeichen $a \in \Sigma$ gilt:

  Von q dürfen *beliebig viele* mit a markierte Pfeile ausgehen.

  Formal: An die Stelle der Übergangsfunktion $\delta : Q \times \Sigma \to Q$ tritt eine *Übergangsrelation* $\Delta \subseteq Q \times \Sigma \times Q$.

Definition 2.10 *Ein* nichtdeterministischer endlicher Automat *(kurz:* NDEA*) ist ein 5-Tupel* $A = (\Sigma, Q, S, F, \Delta)$ *mit:*

- Σ, Q, F *wie beim DEA*
- $S \subseteq Q$ *ist die Menge der sogenannten* Startzustände
- $\Delta \subseteq Q \times \Sigma \times Q$ *ist die sogenannte* Übergangsrelation

Auch hier gilt wieder:

Definition 2.10 liefert nur die *statische Beschreibung* eines NDEA. Es bleibt noch zu klären, wie sich der NDEA *zur Laufzeit* verhält, d.h.

- wie er ein Eingabewort schrittweise verarbeitet,
- welche Wörter er akzeptiert (d.h. welche Sprache er erkennt).

Definition 2.11

- *Die* Übergangsschrittrelation $\vdash_A$ *auf der Menge* $Q \times \Sigma^*$ *aller* Konfigurationen *von* A *ist definiert durch*

$$(q, w) \vdash_A (q', w') \quad \Leftrightarrow \quad \text{es existiert ein } a \in \Sigma \text{ mit } w = aw' \text{ und } (q, a, q') \in \Delta$$

- *Ein Wort* $w \in \Sigma^*$ *wird von* A akzeptiert *wenn Zustände* $s \in S$ *und* $q \in F$ *existieren mit* $(s, w) \vdash_A^* (q, \varepsilon)$.
- *Die von* A akzeptierte Sprache $L(A)$ *ist wieder definiert durch*

$$L(A) \quad = \quad \{w \in \Sigma^* \mid w \text{ wird von } A \text{ akzeptiert}\}$$

Unterschied zum DEA:

- An die Stelle des *eindeutigen* Zustands $q' = \delta(q, a)$ beim DEA tritt ein *beliebiger* Zustand q' mit $(q, a, q') \in \Delta$.
- An die Stelle des *eindeutigen* Startzustands s beim DEA tritt ein *beliebiger* Startzustand $s \in S$.

Alternative Formulierung:

Definition 2.12 *Sei $A = (\Sigma, Q, S, F, \Delta)$ ein NDEA und sei $w \in \Sigma^*$.*

1. *Ein Lauf des Automaten A für das Wort w ist eine Folge von Übergangsschritten*

$$(s, w) \vdash_A \ldots \vdash_A (q, w')$$

   die mit einem Startzustand s beginnt und sich nicht weiter fortsetzen lässt.

2. *Ein Lauf heißt akzeptierend, wenn $q \in F$ und $w' = \varepsilon$.*

Mit diesen neuen Begriffen gilt: Ein Wort $w \in \Sigma^*$ wird genau dann von A akzeptiert, wenn (mindestens) ein akzeptierender Lauf von A für w existiert. Auf die übrigen Läufe von A für w kommt es dann nicht mehr an.

Intuition:

Wenn es einen akzeptierenden Lauf für das Wort w gibt, dann "errät" der NDEA diesen akzeptierenden Lauf.

Wie schafft es der NDEA, immer richtig zu raten?

Diese Frage interessiert uns (zunächst) nicht. Wir verlassen uns (nur) auf die mathematische Definition des Akzeptierens.

Vergleich zur Literatur:

In der Literatur wird oft nur *ein* Startzustand bei einem NDEA zugelassen. Wir lassen hier beliebig viele Startzustände zu, weil es

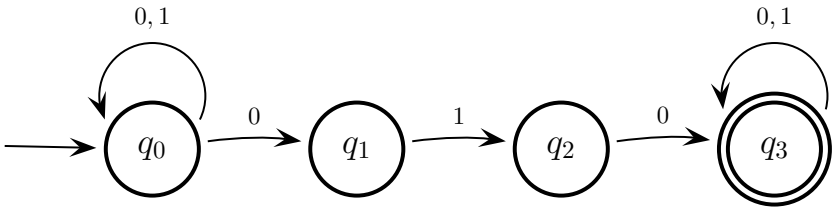
- manchmal nüzlich ist
- die Theorie nicht schwieriger macht
- sogar konsequenter ist als die Beschränkung auf einen einzelnen Startzustand.

Beispiel:

$\Sigma = \{0, 1\}$

$L = \{w \in \{0,1\}^* \mid 010 \text{ ist Teilwort von } w\}$.

Ein NDEA A mit $L(A) = L$ ist:



Formal:

$A = (\Sigma, Q, S, F, \Delta)$ mit

$\Sigma = \{0, 1\}$, $Q = \{q_0, q_1, q_2, q_3\}$, $S = \{q_0\}$, $F = \{q_3\}$ und

$\Delta = \{\ (q_0, 0, q_0), (q_0, 1, q_0), (q_0, 0, q_1), (q_1, 1, q_2), (q_2, 0, q_3),$

$(q_3, 0, q_3), (q_3, 1, q_3)\}$

Wieso gilt $L = L(A)$?

Intuition:

Weil der NDEA A die Stelle, an der das Teilwort 010 beginnt, *erraten* und dann mit 010 in den Endzustand q_3 übergehen kann.

Exakter Beweis:

'$\subseteq$': Sei $w \in L$.

Dann existieren $u, v \in \{0,1\}^*$ mit $w = u010v$.

Also gilt $(q_0, w) = (q_0, u010v) \vdash_A^* (q_0, 010v) \vdash_A^* (q_3, v) \vdash_A^* (q_3, \varepsilon)$.

Damit ist $w \in L(A)$ bewiesen, weil $q_3 \in F$.

'$\supseteq$': Sei $w \in L(A)$.

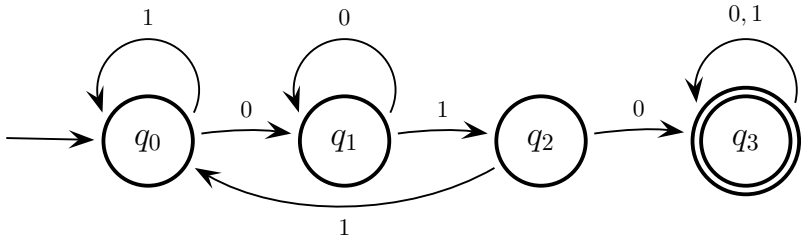
Dann muss $(q_0, w) \vdash_A^* (q_3, \varepsilon)$ gelten, weil q_0 einziger Startzustand und q_3 einziger Endzustand ist.

Da man nur mit 010 von q_0 nach q_3 gelangen kann, muss w das Wort 010 enthalten, d.h. $w \in L$. $\square$

Wieso ist der Beweis so einfach?

Weil dieser NDEA auf einer sehr einfachen Idee beruht.

Zum Vergleich: Ein DEA A' für die gleiche Sprache



Beweisidee:

Sei w das bisher gelesene Wort.

q_0 bedeutet: $w = \varepsilon$, $w = 1$ oder w endet auf 11, aber enthält nicht 010.

q_1 bedeutet: w endet auf 0, aber enthält nicht 010.

q_2 bedeutet: w endet auf 01, aber enthält nicht 010.

q_3 bedeutet: w enthält 010, d.h. $w \in L$.

Fazit:

- Oft findet man leichter einen NDEA als einen DEA für eine vorgegebene Sprache (weil man mehr Freiheiten hat)
- und auch der Korrektheitsbeweis ist oft einfacher (wenn der NDEA auf einer einfacheren Idee beruht).

Wie kann man einen NDEA implementieren?

- Als nichtdeterministisches Programm mit Zufallsgenerator?

  Nein, denn man kann nicht erwarten, dass das Programm den richtigen Weg errät.

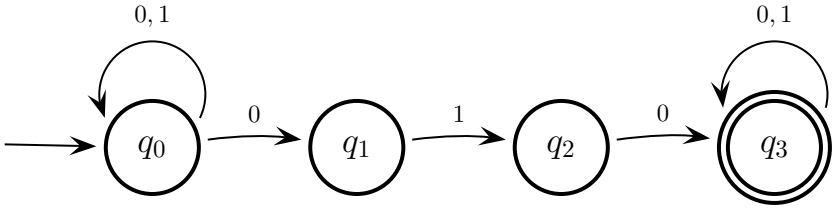
- Durch einen backtracking-Algorithmus, der alle möglichen Wege nacheinander durchspielt?

  Ja, aber man muss sich überlegen, dass es nur endlich viele mögliche Wege gibt, und wie man sie systematisch ausprobiert.

- Durch einen "parallelen" Algorithmus, der alle möglichen Wege gleichzeitig durchspielt?

  Das ist die einfachste Möglichkeit: Man führt einfach Buch über die *Menge* der Zustände, die vom Startzustand aus mit dem bereits gelesenen Wort erreichbar sind. Diese Menge kann auch *leer* sein!

Beispiel: $w = 0011010$ **auf dem NDEA** A



Menge der möglichen Zustände nach Lesen von

- $\varepsilon : \{q_0\}$
- $0 : \{q_0, q_1\}$
- $00 : \{q_0, q_1\}$
- $001 : \{q_0, q_2\}$
- $0011 : \{q_0\}$
- $00110 : \{q_0, q_1\}$
- $001101 : \{q_0, q_2\}$
- $0011010 : \{q_0, q_1, q_3\}$, also $w \in L(A)$, da q_3 mit w erreichbar

Fazit:

- Wir haben einen *determistischen* Algorithmus für einen *N*DEA A,
  - bei dem man das Eingabewort w zeichenweise von links nach rechts liest,
  - bei dem man sich aber Zustands*mengen* anstelle von einzelnen Zuständen merken muss.
- Diesen Algorithmus kann man "auf einem *DEA* simulieren".
- Man muss den DEA nur mit einem hinreichend großen Gedächtnis ausstatten, damit er sich beliebige *Teilmengen* der Zustandsmenge Q des NDEA A merken kann.
- Als Zustände des DEA wählen wir deshalb alle Teilmengen von Q, und legen dann die Übergänge zwischen diesen Zustandsmengen passend fest.
- So erhalten wir einen zu A *äquivalenten* DEA, d.h. einen DEA A' mit $L(A) = L(A')$.

Definition 2.13 *Sei $A = (\Sigma, Q, S, F, \Delta)$ ein NDEA. Der Potenzautomat zu A ist definiert als der DEA $A' = (\Sigma, Q', s', F', \delta)$ mit*

- $Q' = \wp(Q)$, *die Potenzmenge von* Q
- $s' = S \in \wp(Q)$
- $F' = \{P \in \wp(Q) \mid P \cap F \neq \emptyset\}$

  $= \{P \subseteq Q \mid P$ *enthält mindestens einen Zustand aus* $F\}$
- $\delta : \wp(Q) \times \Sigma \to \wp(Q)$

  $\delta(P, a) = \{q \in Q \mid$ *es existiert ein* $p \in P$ *mit* $(p, a, q) \in \Delta\}$

  (d.h. $\delta(P, a)$ *enthält genau die Zustände, die von einem Zustand* $p \in P$ *durch einen mit* a *markierten Pfeil erreichbar sind)*

Satz 2.14 *Sei A ein NDEA und A' der Potenzautomat zu A. Dann gilt $L(A) = L(A')$.*

Beweis

Sei $A = (\Sigma, Q, S, F, \Delta)$ und $A' = (\Sigma, Q', s', F', \delta)$.

Als erstes wird für alle $w \in \Sigma^*$ gezeigt

$$\delta^*(S, w) \;=\; \{q \in Q \mid \exists\, s \in S.\, (s, w) \vdash_A^* (q, \varepsilon)\}$$

d.h.

$$q \in \delta^*(S, w) \;\Leftrightarrow\; \exists\, s \in S.\, (s, w) \vdash_A^* (q, \varepsilon)$$

Intuition: $\delta^*(S, w)$ ist der Zustand, den der Potenzautomat A' von seinem Startzustand S aus mit dem Wort w erreicht. Die Äquivalenz besagt, dass $\delta^*(S, w)$ genau die Zustände enthält, die der NDEA A von *einem* seiner Startzustände aus mit dem Wort w erreichen *kann*, Das bedeutet, dass A' —wie gewünscht—über alle Möglichkeiten des NDEA A Buch führt.

Beweis durch Induktion über $|w|$:

- Für $w = \varepsilon$ ist zu zeigen

$$q \in \delta^*(S, \varepsilon) \;\Leftrightarrow\; \exists\, s \in S.\, (s, \varepsilon) \vdash_A^* (q, \varepsilon)$$

  Das ist klar weil $\delta^*(S, \varepsilon) = S$ per Definition von δ^* und $(s, \varepsilon) \vdash_A^* (q, \varepsilon) \Leftrightarrow s = q$ per Definition von $\vdash_A^*$

- Für $w = va$ ist zu zeigen

$$q \in \delta^*(S, va) \;\Leftrightarrow\; \exists\, s \in S.\, (s, va) \vdash_A^* (q, \varepsilon)$$

  Wegen $\delta^*(S, va) = \delta(\delta^*(S, v), a)$ gilt

$$\begin{aligned} q \in \delta^*(S, va) \;&\Leftrightarrow\; \exists\, p \in \delta^*(S, v).\, (p, a, q) \in \Delta \\ &\quad \text{per Definition von } \delta \text{ im Potenzautomaten } A' \\ &\Leftrightarrow\; \exists\, s \in S, p \in Q.\, (s, v) \vdash_A^* (p, \varepsilon) \wedge (p, a, q) \in \Delta \\ &\quad \text{nach Induktionsannahme für } v \\ &\Leftrightarrow\; (s, va) \vdash_A^* (q, \varepsilon) \\ &\quad \text{nach Lemma 2.9 (gilt auch für NDEAs)} \end{aligned}$$

Damit ist die gewünschte Äquivalenz für alle $w \in \Sigma^*$ bewiesen:

$$q \in \delta^*(S, w) \;\Leftrightarrow\; \exists\, s \in S.\, (s, w) \vdash_A^* (q, \varepsilon)$$

und es folgt:

$$\begin{aligned}
w \in L(A) \;&\Leftrightarrow\; \text{es existieren } s \in S, q \in F \text{ mit } (s, w) \vdash_A^* (q, \varepsilon) \\
&\quad\; \text{per Definition der Sprache } L(A) \\
&\Leftrightarrow\; \delta^*(S, w) \text{ enthält einen Zustand } q \in F \\
&\quad\; \text{nach der bewiesenen Äquivalenz} \\
&\Leftrightarrow\; \delta^*(S, w) \cap F \neq \emptyset \\
&\Leftrightarrow\; \delta^*(S, w) \in F' \\
&\quad\; \text{per Definition von } F' \text{ im Potenzautomaten } A' \\
&\Leftrightarrow\; w \in L(A') \\
&\quad\; \text{weil } s' = S \text{ der Startzustand von } A' \text{ ist}
\end{aligned}$$

Also ist $L(A) = L(A')$, d.h. A und A' sind äquivalent. □

Man beachte:

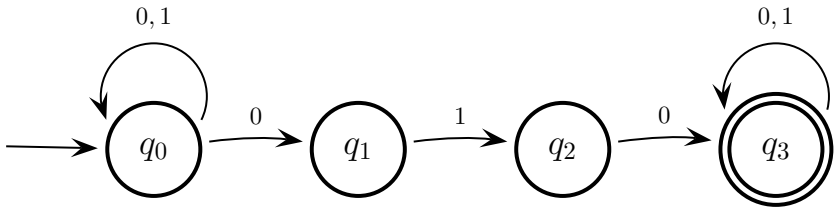
- Im Beweis wurde benutzt, dass Lemma 2.9 auch für NDEAs gilt.
- Die Definition des Potenzautomaten ist *konstruktiv*, d.h. man hat einen *Algorithmus*, um den Potenzautomaten A' aus dem NDEA A zu erhalten.
- Der Potenzautomat ist natürlich sehr groß, denn $|\wp(Q)| = 2^{|Q|}$. Oft sind aber viele Zustände überflüssig, so dass man sie nachträglich entfernen oder sogar von Anfang an weglassen kann.

Definition 2.15 *Sei* $A = (\Sigma, Q, s, F, \delta)$ *ein DEA. Ein Zustand* $q \in Q$ *heißt* erreichbar, *wenn es ein Wort* w *gibt mit* $(s, w) \vdash_A^* (q, \varepsilon)$, *ansonsten heißt er* unerreichbar.

- Unerreichbare Zustände kann man offensichtlich aus einem DEA entfernen, ohne dass sich die akzeptierte Sprache verändert (und ohne dass man die Definition eines DEA verletzt, denn s ist stets erreichbar, und δ lässt sich auf die Menge der erreichbaren Zustände einschränken.)
- Einige "Entwurfsmethoden" für Automaten, wie z.B. die Konstruktion des Potenzautomaten, kann man so abändern, dass der entworfene Automat von vornherein nur erreichbare Zustände enthält.

Beispiel

Sei A wieder der NDEA



Der Potenzautomat von A hat 16 Zustände. Wir bestimmen die erreichbaren unter ihnen.

Der Startzustand $\{q_0\}$ ist erreichbar.

$\delta(\{q_0\}, 0) = \{q_0, q_1\}$, also ist $\{q_0, q_1\}$ erreichbar.

$\delta(\{q_0\}, 1) = \{q_0\}$

$\delta(\{q_0, q_1\}, 0) = \{q_0, q_1\}$

$\delta(\{q_0, q_1\}, 1) = \{q_0, q_2\}$, also ist $\{q_0, q_2\}$ erreichbar.

$\delta(\{q_0, q_2\}, 0) = \{q_0, q_1, q_3\}$, also ist $\{q_0, q_1, q_3\}$ erreichbar.

$\delta(\{q_0, q_2\}, 1) = \{q_0\}$

$\delta(\{q_0, q_1, q_3\}, 0) = \{q_0, q_1, q_3\}$

$\delta(\{q_0, q_1, q_3\}, 1) = \{q_0, q_2, q_3\}$, also ist $\{q_0, q_2, q_3\}$ erreichbar.

$\delta(\{q_0, q_2, q_3\}, 0) = \{q_0, q_1, q_3\}$

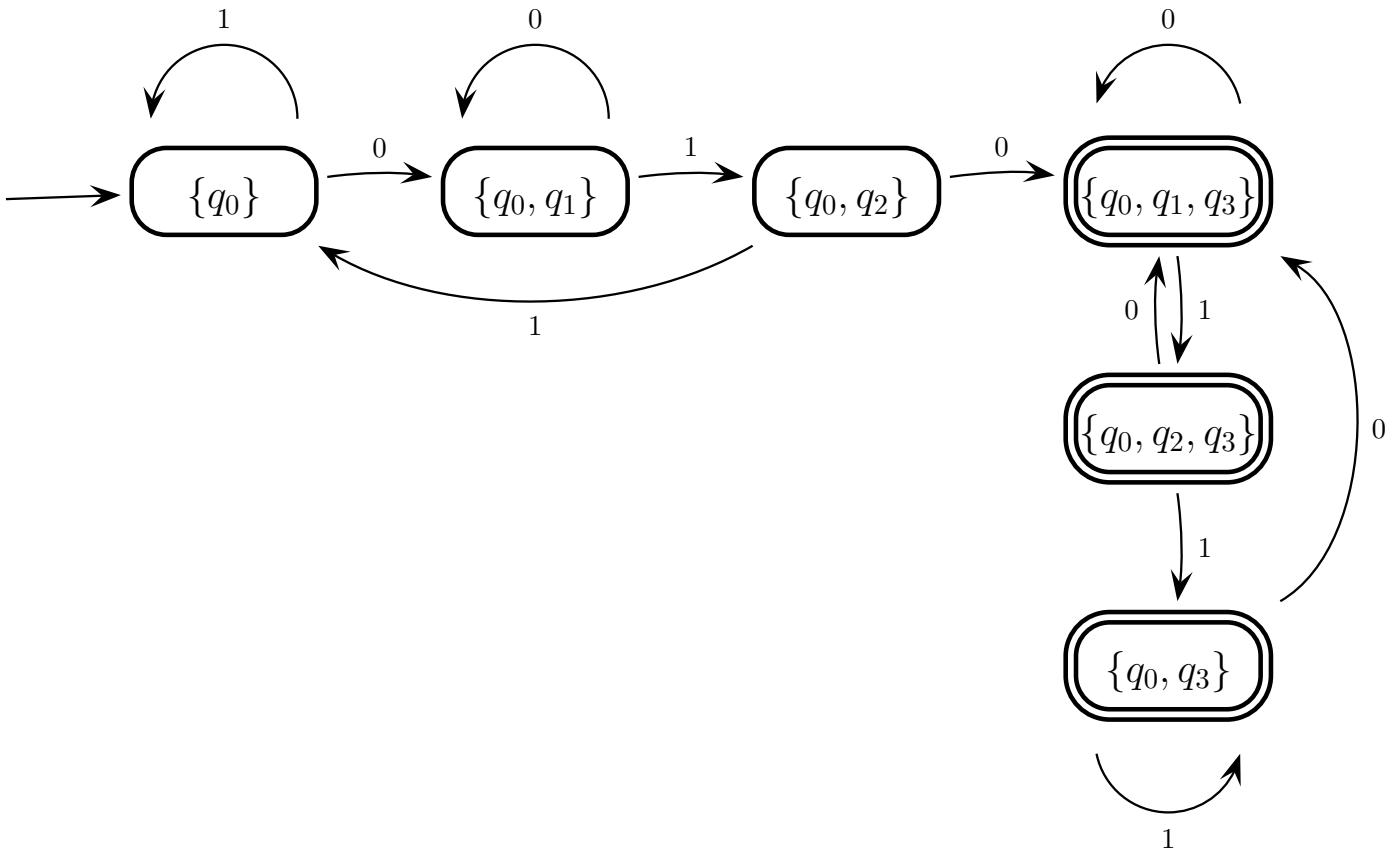
$\delta(\{q_0, q_2, q_3\}, 1) = \{q_0, q_3\}$, also ist $\{q_0, q_3\}$ erreichbar.

$\delta(\{q_0, q_3\}, 0) = \{q_0, q_1, q_3\}$

$\delta(\{q_0, q_3\}, 1) = \{q_0, q_3\}$

Wenn wir uns also auf die erreichbaren Zustände beschränken, so erhalten wir einen DEA, der nur 6 statt 16 Zustände hat. Drei davon sind Endzustände, nämlich diejenigen, die den Endzustand q_3 von A enthalten.

Graphisch: Der ereichbare Teil des Potenzautomaten



Wie bestimmt man die Menge der erreichbaren Zustände?

- Sei $A = (\Sigma, Q, s, F, \delta)$ ein DEA. Für jedes $n \geq 0$ sei

  $R_n = \{q \in Q \mid$ es ex. ein $w \in \Sigma^*$ mit $|w| \leq n$ und $(s, w) \vdash_A^* (q, \varepsilon)\}$
- Die Mengen R_n lassen sich durch Induktion über n definieren:
  - $R_0 = \{s\}$
  - $R_{n+1} = R_n \cup \{\delta(q, a) \mid q \in R_n, a \in \Sigma\}$
- Es gilt $R_0 \subseteq R_1 \subseteq R_2 \subseteq \ldots$ und die Menge R *aller* erreichbaren Zustände erhält man durch

  $R = \bigcup_{n \geq 0} R_n$
- Da alle R_n in der endlichen Menge Q enthalten sind, muss ein n existieren mit $R_n = R_{n+1}$.
- Daraus folgt $R_n = R_m$ für alle $m \geq n$, also $R = R_n$.

Algorithmus

- Berechne die Mengen $R_0, R_1, R_2, \ldots$ mit der induktiven Definition
- Sobald $R_n = R_{n+1}$ gilt, ist R_n die Menge der erreichbaren Zustände.

Man beachte:

- Nach den vorangegangen Überlegungen terminiert der Algorithmus und liefert das korrekte Ergebnis.
- Der gleiche Algorithmus funktioniert auch für NDEAs und auch für einen beliebigen Zustand p anstelle des Startzustands s.
- Er funktioniert sogar, wenn man noch nicht alle Übergänge kennt, denn im Induktionsschritt braucht man nur die Übergänge, die von den bereits gefundenen Zuständen ausgehen.
- Eigentlich handelt es sich um einen Graphalgorithmus (vgl. Vorlesung Algorithmen): Da die Markierung der Pfeile im Algorithmus keine Rolle spielt, kann der Automat als gerichteter Graph gesehen werden.

Gibt es noch weitere überflüssige Zustände?

- Ein Zustand heißt *tot*, wenn man von ihm aus keinen Endzustand mehr erreichen kann.
- Beispiele:

  der Zustand q' im Automaten A_{int}, (der erreicht wird, wenn '$-$' an der falschen Stelle auftritt)

  der Zustand $\emptyset$ in einem Potenzautomaten.
- Auch tote Zustände kann man aus einem Automaten entfernen, ohne dass sich die erkannte Sprache verändert.
- **Aber:** Dabei kann aus der totalen Funktion δ eine partielle Funktion werden, d.h. der entstehende Automat muss kein DEA mehr sein.

Anmerkung zur Literatur:

- Manche Autoren lassen partielle Funktionen in DEAs zu.

  Das entspricht besser dem intuitiven Begriff "deterministisch"

  und man spart sich tote Zustände.

- Aber partielle Funktionen bereiten technische Probleme bei der Schreibweise: Man kann sie nicht auf jedes Element anwenden.

  Deshalb lassen wir sie nicht in DEAs, sondern nur (als spezielle Relationen) in NDEAs zu.

  Das ist kein großer Nachteil, denn man kommt stets mit *einem* toten Zustand aus.