Vorlesungsskript

Einführung in die Theoretische Informatik

Wintersemester 2018/19

Prof. Dr. Johannes Köbler
Humboldt-Universität zu Berlin
Lehrstuhl Komplexität und Kryptografie

26. Oktober 2018

Inhaltsverzeichnis

1 Einleitung

Rechenmaschinen spielen in der Informatik eine zentrale Rolle. In dieser Vorlesung beschäftigen wir uns mit mathematischen Modellen für Maschinentypen von unterschiedlicher Berechnungskraft. Unter anderem lernen wir das Rechenmodell der Turingmaschine (TM) kennen, mit dem sich alle anderen Rechenmodelle simulieren lassen. Ein weiteres wichtiges Thema der Vorlesung ist die Frage, welche Probleme algorithmisch lösbar sind und wo die Grenzen der Berechenbarkeit verlaufen.

Schließlich untersuchen wir die Komplexität von algorithmischen Problemen, indem wir den benötigten Rechenaufwand möglichst gut nach oben und unten abschätzen. Eine besondere Rolle spielen hierbei die NP-vollständigen Probleme, deren Komplexität bis heute offen ist.

Themen der Vorlesung

- Welche Rechenmodelle sind für bestimmte Aufgaben adäquat? (Automatentheorie)
- Welche Probleme sind lösbar? (Berechenbarkeitstheorie)
- Welcher Aufwand ist zur Lösung eines algorithmischen Problems nötig? (Komplexitätstheorie)

In den theoretisch orientierten Folgeveranstaltungen wird es dagegen um folgende Themen gehen.

Thema der Vorlesung Algorithmen und Datenstrukturen

- Wie lassen sich praktisch relevante Problemstellungen möglichst effizient lösen? (Algorithmik)

Thema der Vorlesung Logik in der Informatik

- Mathematische Grundlagen der Informatik, Beweise führen, Modellierung (Aussagenlogik, Prädikatenlogik)

Die wichtigsten **Lernziele der Vorlesung** sind:

- Überblick über die wichtigsten Rechenmodelle (Automaten) wie z.B.
  - endliche Automaten
  - Kellerautomaten
  - Turingmaschinen
  - Registermaschinen
  - Schaltkreise
- Charakterisierung der Klassen aller mit diesen Rechenmodellen lösbaren Probleme durch
  - unterschiedliche Typen von formalen Grammatiken
  - Abschlusseigenschaften unter geeigneten Sprachoperationen
  - Reduzierbarkeit auf typische Probleme (Vollständigkeit)
- Erkennen von Grenzen der Berechenbarkeit
- Klassifikation wichtiger algorithmischer Probleme nach ihrer Komplexität

Rechenmaschinen spielen in der Informatik eine zentrale Rolle Es gibt viele unterschiedliche mathematische Modelle für Rechenmaschinen. Diese können sich in ihrer Berechnungskraft unterscheiden Die Turingmaschine (TM) ist ein universales Berechnungsmodell, da sie alle anderen bekannten Rechenmodelle simulieren kann. Wir betrachten zunächst Einschränkungen des TM-Modells, die vielfältige praktische Anwendungen haben, wie z.B.

- endliche Automaten (DFA, NFA)
- Kellerautomaten (PDA, DPDA) etc.

Der Begriff *Algorithmus* geht auf den persischen Gelehrten Muhammed Al Chwarizmi (8./9. Jhd.) zurück. Der älteste bekannte nicht-triviale

Algorithmus ist der nach *Euklid* benannte Algorithmus zur Berechnung des größten gemeinsamen Teilers zweier natürlicher Zahlen (300 v. Chr.). Von einem Algorithmus wird erwartet, dass er für jede zulässige *Problemeingabe* nach endlich vielen Rechenschritten eine korrekte *Ausgabe* liefert. Eine wichtige Rolle spielen *Entscheidungsprobleme*, bei denen jede Eingabe nur mit ja oder nein beantwortet wird. Die (maximale) Anzahl der Rechenschritte bei allen möglichen Eingaben ist nicht beschränkt, d.h. mit wachsender Eingabelänge kann auch die Rechenzeit beliebig anwachsen. Die Beschreibung eines Algorithmus muss jedoch endlich sein. Problemeingaben können Zahlen, Formeln, Graphen etc. sein. Diese werden über einem *Eingabealphabet* Σ kodiert.

Definition 1.

a) Ein ***Alphabet*** $\Sigma = \{a_1, \ldots, a_m\}$ *ist eine geordnete Menge von endlich vielen* ***Zeichen****.*

b) Eine Folge $x = x_1 \ldots x_n$ *von* n *Zeichen heißt* ***Wort*** *(der* ***Länge*** $|x| = n$*).*

c) Die Menge aller Wörter über Σ *ist*

$$\Sigma^* = \bigcup_{n \geq 0} \Sigma^n,$$

wobei $\Sigma^n = \{x_1 \ldots x_n \mid n \geq 0$ *und* $x_i \in \Sigma$ *für* $i = 1, \ldots, n\}$ *alle Wörter der Länge* n *enthält.*

d) Das (einzige) Wort der Länge $n = 0$ *ist das* ***leere Wort****, welches wir mit* ε *bezeichnen.*

e) Jede Teilmenge $L \subseteq \Sigma^*$ *heißt* ***Sprache*** *über dem Alphabet* Σ*.*

Beispiel 2. *Sei* Σ *ein Alphabet. Dann sind* $\varnothing, \Sigma^*, \Sigma$ *und* $\{\varepsilon\}$ *Sprachen über* Σ*. Die Sprache* $\varnothing$ *enthält keine Wörter und heißt* ***leere Sprache****. Die Sprache* Σ^* *enthält dagegen alle Wörter über* Σ*, während die Sprache* Σ *alle Wörter über* Σ *der Länge 1 enthält. Die Sprache* $\{\varepsilon\}$ *enthält nur das leere Wort, ist also einelementig. Einelementige Sprachen werden auch als* ***Singletonsprachen*** *bezeichnet.*

Da Sprachen Mengen sind, können wir sie bzgl. Inklusion vergleichen. Zum Beispiel gilt

$$\varnothing \subseteq \{\varepsilon\} \subseteq \Sigma^*.$$

Wir können Sprachen auch vereinigen, schneiden und komplementieren. Seien A und B Sprachen über Σ. Dann ist

- $A \cap B = \{x \in \Sigma^* \mid x \in A, x \in B\}$ der ***Schnitt*** von A und B,
- $A \cup B = \{x \in \Sigma^* \mid x \in A \vee x \in B\}$ die ***Vereinigung*** von A und B, und
- $\overline{A} = \{x \in \Sigma^* \mid x \notin A\}$ das ***Komplement*** von A.

Neben den Mengenoperationen gibt es auch spezielle Sprachoperationen.

Definition 3.

- *Das* ***Produkt*** *(****Verkettung****,* ***Konkatenation****) der Sprachen* A *und* B *ist*

$$AB = \{xy \mid x \in A, y \in B\}.$$

Ist $A = \{x\}$ *eine Singletonsprache, so schreiben wir für* $\{x\}B$ *auch einfach* xB*.*

- *Die* ***n-fache Potenz*** $\boldsymbol{A^n}$ *einer Sprache* A *ist induktiv definiert durch*

$$A^n = \begin{cases} \{\varepsilon\}, & n = 0, \\ A^{n-1}A, & n > 0. \end{cases}$$

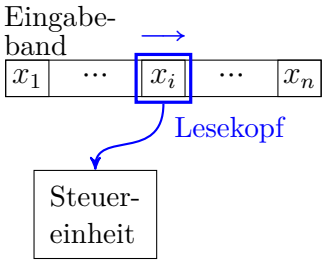
- *Die* ***Sternhülle*** $\boldsymbol{A^*}$ *von* A *ist* $A^* = \bigcup_{n \geq 0} A^n$ *und die* ***Plushülle*** $\boldsymbol{A^+}$ *von* A *ist* $A^+ = \bigcup_{n \geq 1} A^n = AA^*$*.*

2 Reguläre Sprachen

Wir betrachten zunächst Einschränkungen des TM-Modells, die vielfältige praktische Anwendungen haben, wie z.B. endliche Automaten (DFA, NFA), Kellerautomaten (PDA, DPDA) etc.

2.1 Endliche Automaten

Ein endlicher Automat führt bei einer Eingabe der Länge n nur n Rechenschritte aus. Um die gesamte Eingabe lesen zu können, muss der Automat also in jedem Schritt ein Zeichen der Eingabe verarbeiten.



Definition 4. *Ein* ***endlicher Automat*** *(kurz: DFA; deterministic finite automaton) wird durch ein 5-Tupel $M = (Z, \Sigma, \delta, q_0, E)$ beschrieben, wobei*

- *$Z \neq \varnothing$ eine endliche Menge von* ***Zuständen****,*
- *Σ das* ***Eingabealphabet****,*
- *$\delta : Z \times \Sigma \to Z$ die* ***Überführungsfunktion****,*
- *$q_0 \in Z$ der* ***Startzustand*** *und*
- *$E \subseteq Z$ die Menge der* ***Endzustände*** *ist.*

Die von M ***akzeptierte*** *oder* ***erkannte Sprache*** *ist*

$$L(M) = \left\{ x_1 \dots x_n \in \Sigma^* \;\middle|\; \begin{array}{l} \text{es gibt } q_1, \dots, q_{n-1} \in Z, q_n \in E \text{ mit} \\ \delta(q_i, x_{i+1}) = q_{i+1} \text{ für } i = 0, \dots, n-1 \end{array} \right\}.$$

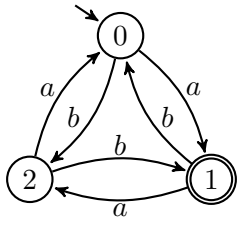
Eine Zustandsfolge $q_0, q_1, \dots, q_n$ heißt ***Rechnung*** *von $M(x_1 \dots x_n)$, falls $\delta(q_i, x_{i+1}) = q_{i+1}$ für $i = 0, \dots, n-1$ gilt. Sie heißt* ***akzeptierend****, falls $q_n \in E$ ist, und andernfalls* ***verwerfend****. Eine von einem DFA akzeptierte Sprache wird als* ***regulär*** *bezeichnet. Die zugehörige Sprachklasse ist*

$$\mathsf{REG} = \{L(M) \mid M \text{ ist ein DFA}\}.$$

Beispiel 5. *Betrachte den DFA $M = (Z, \Sigma, \delta, 0, E)$ mit $Z = \{0, 1, 2\}$, $\Sigma = \{a, b\}$, $E = \{1\}$ und der Überführungsfunktion*

δ	0	1	2
a	1	2	0
b	2	0	1

Graphische Darstellung:



Der Startzustand wird meist durch einen Pfeil und Endzustände werden durch einen doppelten Kreis gekennzeichnet.

Bei Eingabe $w_1 = aba$ führt M die akzeptierende Rechnung $0, 1, 0, 1$ durch, d.h. $w_1 \in L(M)$. Dagegen verwirft M das Wort $w_2 = abba$ (verwerfende Rechnung: $0, 1, 0, 2, 0$). ◁

Bezeichne $\hat{\delta}(q, x)$ denjenigen Zustand, in dem sich M nach Lesen von x befindet, wenn M im Zustand q gestartet wird. Dann können wir die Funktion

$$\hat{\delta} : Z \times \Sigma^* \to Z$$

induktiv wie folgt definieren. Für $q \in Z$, $x \in \Sigma^*$ und $a \in \Sigma$ sei

$$\begin{aligned} \hat{\delta}(q, \varepsilon) &= q, \\ \hat{\delta}(q, xa) &= \delta(\hat{\delta}(q, x), a). \end{aligned}$$

Die von M erkannte Sprache lässt sich nun elegant durch

$$L(M) = \{x \in \Sigma^* \mid \hat{\delta}(q_0, x) \in E\}$$

beschreiben.

Behauptung 6. *Der DFA M aus Beispiel 5 akzeptiert die Sprache*

$$L(M) = \{x \in \Sigma^* \mid \#_a(x) - \#_b(x) \equiv_3 1\},$$

wobei $\#_a(x)$ die Anzahl der Vorkommen des Zeichens a in x bezeichnet und $i \equiv_m j$ (in Worten: i ist kongruent zu j modulo m) bedeutet, dass $i - j$ durch m teilbar ist.

Beweis. Da M nur den Endzustand 1 hat, ist $L(M) = \{x \in \Sigma^* \mid \hat{\delta}(0, x) = 1\}$, d.h. wir müssen folgende Äquivalenz zeigen:

$$\hat{\delta}(0, x) = 1 \Leftrightarrow \#_a(x) - \#_b(x) \equiv_3 1.$$

Hierzu reicht es, die Kongruenz

$$\hat{\delta}(0, x) \equiv_3 \#_a(x) - \#_b(x).$$

zu beweisen, wofür wir Induktion über die Länge n von x benutzen.

Induktionsanfang ***($n = 0$):*** klar, da $\hat{\delta}(0, \varepsilon) = \#_a(\varepsilon) = \#_b(\varepsilon) = 0$ ist.

Induktionsschritt ***($n \rightsquigarrow n+1$):*** Sei $x = x_1 \dots x_{n+1}$ gegeben und sei $i = \hat{\delta}(0, x_1 \dots x_n)$. Nach IV gilt dann

$$i \equiv_3 \#_a(x_1 \dots x_n) - \#_b(x_1 \dots x_n).$$

Wegen $\delta(i, a) \equiv_3 i + 1$ und $\delta(i, b) \equiv_3 i - 1$ folgt daher

$$\begin{aligned} \delta(i, x_{n+1}) &\equiv_3 i + \#_a(x_{n+1}) - \#_b(x_{n+1}) \\ &\equiv_3 \#_a(x_1 \dots x_n) - \#_b(x_1 \dots x_n) + \#_a(x_{n+1}) - \#_b(x_{n+1}) \\ &= \#_a(x) - \#_b(x). \end{aligned}$$

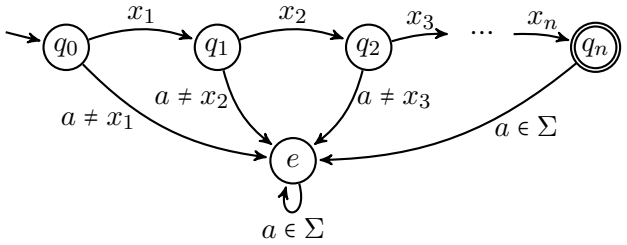
und somit

$$\hat{\delta}(0, x) = \delta(\hat{\delta}(0, x_1 \dots x_n), x_{n+1}) = \delta(i, x_{n+1}) \equiv_3 \#_a(x) - \#_b(x).$$

■

Beobachtung 7. *Alle Singletonsprachen sind regulär.*

Beweis. Für jedes Wort $x = x_1 \dots x_n$ existiert ein DFA M_x mit $L(M_x) = \{x\}$:



Formal ist M_x also das Tupel $(Z, \Sigma, \delta, q_0, E)$ mit $Z = \{q_0, \dots, q_n, e\}$, $E = \{q_n\}$ und der Überführungsfunktion

$$\delta(q, a_j) = \begin{cases} q_{i+1}, & q = q_i \text{ für ein } i \text{ mit } 0 \leq i \leq n-1 \text{ und } a_j = x_{i+1} \\ e, & \text{sonst.} \end{cases}$$

■

Als nächstes betrachten wir Abschlusseigenschaften der Sprachklasse REG.

Definition 8. *Ein* ***k-stelliger Sprachoperator*** *ist eine Abbildung op, die k Sprachen $L_1, \dots, L_k$ auf eine Sprache $op(L_1, \dots, L_k)$ abbildet.*

Beispiel 9. *Der Schnittoperator $\cap$ bildet zwei Sprachen L_1 und L_2 auf die Sprache $L_1 \cap L_2$ ab.* ◁

Definition 10. *Eine Sprachklasse $\mathcal{K}$ heißt unter op* ***abgeschlossen****, wenn gilt:*

$$L_1, \dots, L_k \in \mathcal{K} \Rightarrow op(L_1, \dots, L_k) \in \mathcal{K}.$$

Der ***Abschluss*** *von $\mathcal{K}$ unter op ist die bzgl. Inklusion kleinste Sprachklasse $\mathcal{K}'$, die $\mathcal{K}$ enthält und unter op abgeschlossen ist.*

Beispiel 11. *Der Abschluss der Singletonsprachen unter $\cap$ besteht aus allen Singletonsprachen und der leeren Sprache.*

Der Abschluss der Singletonsprachen unter $\cup$ besteht aus allen nichtleeren endlichen Sprachen.

Der Abschluss der Singletonsprachen unter $\cap$, $\cup$ und Komplement besteht aus allen endlichen und co-endlichen Sprachen.[*] ◁

Definition 12. *Für eine Sprachklasse $\mathcal{C}$ bezeichne co-$\mathcal{C}$ die Klasse $\{\bar{L} \mid L \in \mathcal{C}\}$ aller Komplemente von Sprachen in $\mathcal{C}$.*

Es ist leicht zu sehen, dass $\mathcal{C}$ genau dann unter Komplementbildung abgeschlossen ist, wenn *co-*$\mathcal{C} = \mathcal{C}$ ist.

Beobachtung 13. *Mit $L_1, L_2 \in \mathsf{REG}$ sind auch die Sprachen $\overline{L_1} = \Sigma^* \setminus L_1$, $L_1 \cap L_2$ und $L_1 \cup L_2$ regulär.*

Beweis. Sind $M_i = (Z_i, \Sigma, \delta_i, q_0, E_i)$, $i = 1, 2$, DFAs mit $L(M_i) = L_i$, so akzeptiert der DFA

$$\overline{M_1} = (Z_1, \Sigma, \delta_1, q_0, Z_1 \setminus E_1)$$

das Komplement $\overline{L_1}$ von L_1. Der Schnitt $L_1 \cap L_2$ von L_1 und L_2 wird dagegen von dem DFA

$$M = (Z_1 \times Z_2, \Sigma, \delta, (q_0, q_0), E_1 \times E_2)$$

mit

$$\delta((q,p),a) = (\delta_1(q,a), \delta_2(p,a))$$

akzeptiert (M wird auch **Kreuzproduktautomat** genannt). Wegen $L_1 \cup L_2 = \overline{(\overline{L_1} \cap \overline{L_2})}$ ist dann aber auch die Vereinigung von L_1 und L_2 regulär. (Wie sieht der zugehörige DFA aus?) ■

[*] Eine Sprache $L \subseteq \Sigma^*$ ist co-endlich, wenn ihr Komplement $\bar{L}$ endlich ist.

Aus Beobachtung 13 folgt, dass alle endlichen und alle co-endlichen Sprachen regulär sind. Da die in Beispiel 5 betrachtete Sprache weder endlich noch co-endlich ist, haben wir damit allerdings noch nicht alle regulären Sprachen erfasst.

Es stellt sich die Frage, ob REG neben den mengentheoretischen Operationen Schnitt, Vereinigung und Komplement unter weiteren Operationen wie etwa Produkt oder Sternhülle abgeschlossen ist. Im übernächsten Abschnitt werden wir sehen, dass die Klasse REG als der Abschluss der endlichen Sprachen unter Vereinigung, Produkt und Sternhülle charakterisierbar (und somit auch unter diesen Operationen abgeschlossen) ist.

Beim Versuch, einen endlichen Automaten für das Produkt $L_1 L_2$ zweier regulärer Sprachen zu konstruieren, stößt man auf die Schwierigkeit, den richtigen Zeitpunkt für den Übergang von (der Simulation von) M_1 zu M_2 zu finden. Unter Verwendung eines nichtdeterministischen Automaten lässt sich dieses Problem jedoch leicht lösen, da dieser den richtigen Zeitpunkt „erraten“ kann.

Im nächsten Abschnitt werden wir nachweisen, dass auch nichtdeterministische endliche Automaten nur reguläre Sprachen erkennen können.

2.2 Nichtdeterministische endliche Automaten

Definition 14. *Ein **nichtdeterministischer endlicher Automat** (kurz: NFA; nondeterministic finite automaton) $N = (Z, \Sigma, \Delta, Q_0, E)$ ist ähnlich aufgebaut wie ein DFA, nur dass er mehrere Startzustände (zusammengefasst in der Menge $Q_0 \subseteq Z$) haben kann und seine Überführungsfunktion die Form*

$$\Delta : Z \times \Sigma \to \mathcal{P}(Z)$$

hat. Hierbei bezeichnet $\mathcal{P}(Z)$ die **Potenzmenge** *(also die Menge aller Teilmengen) von Z. Diese wird auch oft mit 2^Z bezeichnet. Die von N akzeptierte Sprache ist*

$$L(N) = \left\{ x_1 \dots x_n \in \Sigma^* \;\middle|\; \begin{array}{c} \exists\, q_0 \in Q_0, q_1, \dots, q_{n-1} \in Z, q_n \in E\colon \\ q_{i+1} \in \Delta(q_i, x_{i+1}) \text{ für } i = 0, \dots, n-1 \end{array} \right\}.$$

Eine Zustandsfolge $q_0, q_1, \dots, q_n$ heißt **Rechnung** *von $N(x_1 \dots x_n)$, falls $q_{i+1} \in \Delta(q_i, x_{i+1})$ für $i = 0, \dots, n-1$ gilt.*

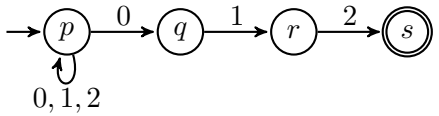
Ein NFA N kann bei einer Eingabe x also nicht nur eine, sondern mehrere verschiedene Rechnungen parallel ausführen. Ein Wort x gehört genau dann zu $L(N)$, wenn $N(x)$ mindestens eine akzeptierende Rechnung hat.

Im Gegensatz zu einem DFA, dessen Überführungsfunktion auf der gesamten Menge $Z \times \Sigma$ definiert ist, kann ein NFA „stecken bleiben“. Das ist dann der Fall, wenn er in einen Zustand q gelangt, in dem das nächste Eingabezeichen x_i wegen $\Delta(q, x_i) = \varnothing$ nicht gelesen werden kann.

Beispiel 15. *Betrachte den NFA $N = (Z, \Sigma, \Delta, Q_0, E)$ mit Zustandsmenge $Z = \{p, q, r, s\}$, Eingabealphabet $\Sigma = \{0, 1, 2\}$, Start- und Endzustandsmenge $Q_0 = \{p\}$ und $E = \{s\}$ sowie der Überführungsfunktion*

Δ	p	q	r	s
0	$\{p, q\}$	$\varnothing$	$\varnothing$	$\varnothing$
1	$\{p\}$	$\{r\}$	$\varnothing$	$\varnothing$
2	$\{p\}$	$\varnothing$	$\{s\}$	$\varnothing$

Graphische Darstellung:



Offensichtlich akzeptiert N die Sprache $L(N) = \{x012 \mid x \in \Sigma^*\}$ aller Wörter, die mit dem Suffix 012 enden. ◁

Beobachtung 16. *Sind $N_i = (Z_i, \Sigma, \Delta_i, Q_i, E_i)$ $(i = 1, 2)$ NFAs, so werden auch die Sprachen $L(N_1)L(N_2)$ und $L(N_1)^*$ von einem NFA erkannt.*

Beweis. Sei $L_i = L(N_i)$. Wir können $Z_1 \cap Z_2 = \varnothing$ annehmen. Dann akzeptiert der NFA

$$N = (Z_1 \cup Z_2, \Sigma, \Delta_3, Q_1, E)$$

mit

$$\Delta_3(p, a) = \begin{cases} \Delta_1(p, a), & p \in Z_1 \setminus E_1, \\ \Delta_1(p, a) \cup \bigcup_{q \in Q_2} \Delta_2(q, a), & p \in E_1, \\ \Delta_2(p, a), & \text{sonst} \end{cases}$$

und

$$E = \begin{cases} E_2, & Q_2 \cap E_2 = \varnothing \\ E_1 \cup E_2, & \text{sonst} \end{cases}$$

die Sprache $L_1 L_2$.

$L_1 L_2 \subseteq L(N)$: Seien $x = x_1 \cdots x_k \in L_1$, $y = y_1 \cdots y_l \in L_2$ und seien $q_0, \dots, q_k$ und $p_0, \dots, p_l$ akzeptierende Rechnungen von $N_1(x)$ und $N_2(y)$. Dann ist $q_0, \dots, q_k, p_1, \dots, p_l$ eine akz. Rechnung von $N(xy)$, da $q_0 \in Q_1$ und $p_l \in E_2$ ist, und

- im Fall $l \geq 1$ wegen $q_k \in E_1$, $p_0 \in Q_2$ und $p_1 \in \Delta_2(p_0, y_1)$ zudem $p_1 \in \Delta(q_k, y_1)$ und
- im Fall $l = 0$ wegen $q_k \in E_1$ und $p_l \in Q_2 \cap E_2$ zudem $q_k \in E$ ist.

$L(N) \subseteq L_1 L_2$: Sei $x = x_1 \cdots x_n \in L(N)$ und sei $q_0, \dots, q_n$ eine akz. Rechnung von $N(x)$. Dann gilt $q_0 \in Q_1$, $q_n \in E$, $q_0, \dots, q_i \in Z_1$ und $q_{i+1}, \dots, q_n \in Z_2$ für ein $i \leq n$. Wir zeigen, dass ein $q \in Q_2$ existiert, so dass $q_0, \dots, q_i$ eine akz. Rechnung von $N_1(x_1 \cdots x_i)$ und $q, q_{i+1}, \dots, q_n$ eine akz. Rechnung von $N_2(x_{i+1} \cdots x_n)$ ist.

- Im Fall $i < n$ impliziert der Übergang $q_{i+1} \in \Delta(q_i, x_{i+1})$, dass $q_i \in E_1$ (also $q_0, \dots, q_i$ eine akz. Rechnung von $N_1(x_1 \cdots x_i)$) und $q_{i+1} \in \Delta_2(q, x_{i+1})$ für ein $q \in Q_2$ ist. Zudem ist $q_n \in E \cap Z_2 = E_2$ (also $q, q_{i+1}, \dots, q_n$ eine akz. Rechnung von $N_2(x_{i+1} \cdots x_n)$).
- Im Fall $i = n$ ist $q_n \in E \cap Z_1$, was $q_n \in E_1$ und $Q_2 \cap E_2 \neq \varnothing$ impliziert (also ist $q_0, \dots, q_n$ eine akz. Rechnung von $N_1(x_1 \cdots x_n)$ und es gibt ein $q \in Q_2$, so dass q eine akz. Rechnung von $N_2(\varepsilon)$ ist).

Ganz ähnlich lässt sich zeigen, dass der NFA

$$N^* = (Z_1 \cup \{q_{neu}\}, \Sigma, \Delta_4, Q_1 \cup \{q_{neu}\}, E_1 \cup \{q_{neu}\})$$

mit

$$\Delta_4(p,a) = \begin{cases} \Delta_1(p,a), & p \in Z_1 \setminus E_1, \\ \Delta_1(p,a) \cup \bigcup_{q \in Q_1} \Delta_1(q,a), & p \in E_1, \\ \varnothing, & \text{sonst} \end{cases}$$

die Sprache L_1^* akzeptiert. ∎

Satz 17 (Rabin und Scott)**.**
$\mathsf{REG} = \{L(N) \mid N \textit{ ist ein NFA}\}$.

Beweis. Die Inklusion von links nach rechts ist klar, da jeder DFA auch als NFA aufgefasst werden kann. Für die Gegenrichtung konstruieren wir zu einem NFA $N = (Z, \Sigma, \Delta, Q_0, E)$ einen DFA $M = (\mathcal{P}(Z), \Sigma, \delta, Q_0, E')$ mit $L(M) = L(N)$. Wir definieren die Überführungsfunktion $\delta : \mathcal{P}(Z) \times \Sigma \to \mathcal{P}(Z)$ von M mittels

$$\delta(Q,a) = \bigcup_{q \in Q} \Delta(q,a).$$

Die Menge $\delta(Q,a)$ enthält also alle Zustände, in die N gelangen kann, wenn N ausgehend von einem beliebigen Zustand $q \in Q$ das Zeichen a liest. Intuitiv bedeutet dies, dass der DFA M den NFA N simuliert, indem M in seinem aktuellen Zustand Q die Information speichert, in welchen Zuständen sich N momentan befinden könnte. Für die Erweiterung $\hat{\delta} : \mathcal{P}(Z) \times \Sigma^* \to \mathcal{P}(Z)$ von δ (siehe Seite 3) können wir nun folgende Behauptung zeigen.

Behauptung. $\hat{\delta}(Q_0, x)$ enthält alle Zustände, die N ausgehend von einem Startzustand nach Lesen von x erreichen kann.

Wir beweisen die Behauptung induktiv über die Länge n von x.

Induktionsanfang **($n = 0$):** klar, da $\hat{\delta}(Q_0, \varepsilon) = Q_0$ ist.

Induktionsschritt **($n-1 \rightsquigarrow n$):** Sei $x = x_1 \dots x_n$ gegeben. Nach Induktionsvoraussetzung enthält

$$Q_{n-1} = \hat{\delta}(Q_0, x_1 \dots x_{n-1})$$

alle Zustände, die $N(x)$ in genau $n-1$ Schritten erreichen kann. Wegen

$$\hat{\delta}(Q_0, x) = \delta(Q_{n-1}, x_n) = \bigcup_{q \in Q_{n-1}} \Delta(q, x_n)$$

enthält dann aber $\hat{\delta}(Q_0, x)$ alle Zustände, die $N(x)$ in genau n Schritten erreichen kann.

Deklarieren wir nun diejenigen Teilmengen $Q \subseteq Z$, die mindestens einen Endzustand von N enthalten, als Endzustände des **Potenzmengenautomaten** M, d.h.

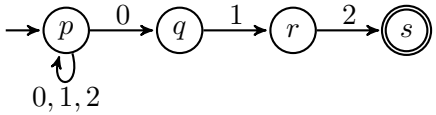
$$E' = \{Q \subseteq Z \mid Q \cap E \neq \varnothing\},$$

so folgt für alle Wörter $x \in \Sigma^*$:

$$\begin{aligned} x \in L(N) &\Leftrightarrow N(x) \text{ kann in genau } |x| \text{ Schritten einen Endzustand erreichen} \\ &\Leftrightarrow \hat{\delta}(Q_0, x) \cap E \neq \varnothing \\ &\Leftrightarrow \hat{\delta}(Q_0, x) \in E' \\ &\Leftrightarrow x \in L(M). \end{aligned}$$

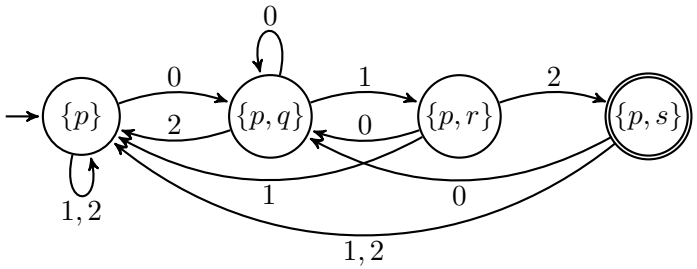
∎

Beispiel 18. *Für den NFA $N = (Z, \Sigma, \Delta, Q_0, E)$ aus Beispiel 15*



ergibt die Konstruktion des vorigen Satzes den folgenden DFA M (nach Entfernen aller vom Startzustand $Q_0 = \{p\}$ aus nicht erreichbaren Zustände):

δ	0	1	2
$Q_0 = \{p\}$	$\{p, q\}$	$\{p\}$	$\{p\}$
$Q_1 = \{p, q\}$	$\{p, q\}$	$\{p, r\}$	$\{p\}$
$Q_2 = \{p, r\}$	$\{p, q\}$	$\{p\}$	$\{p, s\}$
$Q_3 = \{p, s\}$	$\{p, q\}$	$\{p\}$	$\{p\}$



◁

Im obigen Beispiel wurden für die Konstruktion des DFA M aus dem NFA N nur 4 der insgesamt $2^{\|Z\|} = 16$ Zustände benötigt, da die übrigen 12 Zustände in $\mathcal{P}(Z)$ nicht vom Startzustand $Q_0 = \{p\}$ aus erreichbar sind. Es gibt jedoch Beispiele, bei denen alle $2^{\|Z\|}$ Zustände in $\mathcal{P}(Z)$ für die Konstruktion des Potenzmengenautomaten benötigt werden (siehe Übungen).

Korollar 19. *Die Klasse* REG *der regulären Sprachen ist unter folgenden Operationen abgeschlossen:*

- *Komplement,*
- *Schnitt,*
- *Vereinigung,*
- *Produkt,*
- *Sternhülle.*