Vorlesungsskript

Einführung in die Theoretische Informatik

Wintersemester 2011/12

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

24. Oktober 2012

Inhaltsverzeichnis

Inhaltsverzeichnis

1 Einleitung **1**

2 Reguläre Sprachen **2**
2.1 Endliche Automaten 2
2.2 Nichtdeterministische endliche Automaten 4
2.3 Reguläre Ausdrücke 7

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)

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 jede *Problemeingabe* nach endlich vielen Rechenschritten löst (etwa durch Produktion einer Ausgabe). Eine wichtige Rolle spielen Entscheidungsprobleme, bei denen jede Eingabe nur mit ja oder nein beantwortet wird. 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*** 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.*

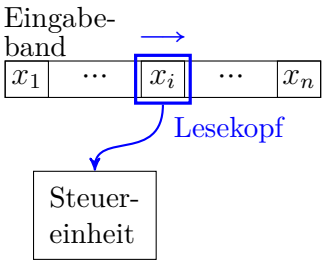
e) Jede Teilmenge $L \subseteq \Sigma^*$ *heißt* ***Sprache*** *über dem Alphabet* Σ*.*

Das zu einer Sprache L gehörige Entscheidungsproblem ist die Frage, ob ein gegebenes Wort x in L enthalten ist oder nicht.

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 2. *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.*

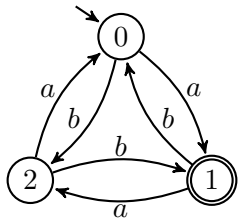
*Die von M **akzeptierte** oder **erkannte Sprache** ist*

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

Beispiel 3. *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. ◁

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 auch in der Form

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

schreiben.

Behauptung 4. *Der DFA M aus Beispiel 3 akzeptiert die Sprache*

$$L(M) = \{x \in \Sigma^* \mid \#_a(x) - \#_b(x) \equiv 1 \mod 3\},$$

wobei $\#_a(x)$ die Anzahl der Vorkommen des Zeichens a in x bezeichnet und $j \equiv k \mod m$ bedeutet, dass $j - k$ durch m teilbar ist. Für Letzteres schreiben wir auch kurz $j \equiv_m k$.

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

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

Folglich ist

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

■

Eine von einem DFA akzeptierte Sprache wird als **regulär** bezeichnet. Die zugehörige Sprachklasse ist

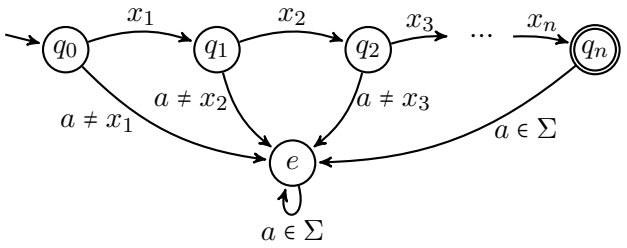
$$\mathsf{REG} = \{L(M) \mid M \text{ ist ein DFA}\}.$$

Um ein intuitives Verständnis für die Berechnungskraft von DFAs zu entwickeln, werden wir Antworten auf folgende Frage suchen.

Frage: Welche Sprachen gehören zu REG und welche nicht?

Dabei legen wir unseren Überlegungen ein beliebiges aber fest gewähltes Alphabet $\Sigma = \{a_1, \dots, a_m\}$ zugrunde.

Beobachtung 5. *Alle Sprachen, die aus einem einzigen Wort $x = x_1 \dots x_n \in \Sigma^*$ bestehen (diese Sprachen werden auch als* Singletonsprachen *bezeichnet), sind regulär. Für folgenden DFA M_x gilt nämlich $L(M_x) = \{x\}$.*



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

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

Als nächstes betrachten wir Abschlusseigenschaften der Sprachklasse REG.

Definition 6. *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 7. *Der Schnittoperator $\cap$ bildet zwei Sprachen L_1 und L_2 auf die Sprache $L_1 \cap L_2$ ab.* ◁

Definition 8. *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 9. *Der Abschluss der Singletonsprachen unter Vereinigung besteht aus allen nichtleeren endlichen Sprachen.* ◁

Definition 10. *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 11. *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. Sind nämlich $M_i = (Z_i, \Sigma, \delta_i, q_0, E_i)$, $i = 1, 2$, 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?)*

Aus Beobachtung 11 folgt, dass alle endlichen und alle co-endlichen Sprachen regulär sind. Da die in Beispiel 3 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 der **Produktbildung**

$$L_1 L_2 = \{xy \mid x \in L_1, y \in L_2\}$$

(auch **Verkettung** oder **Konkatenation** genannt) oder der Bildung der **Sternhülle**

$$L^* = \bigcup_{n \geq 0} L^n$$

abgeschlossen ist. Die n-fache Potenz L^n von L ist dabei induktiv definiert durch

$$L^0 = \{\varepsilon\},\ L^{n+1} = L^n L.$$

Die ***Plushülle*** von L ist

$$L^+ = \bigcup_{n \geq 1} L^n = LL^*.$$

Ist $L_1 = \{x\}$ eine Singletonsprache, so schreiben wir für das Produkt $\{x\}L_2$ auch einfach xL_2.

Im übernächsten Abschnitt werden wir sehen, dass die Klasse REG als der Abschluss der endlichen Sprachen unter Vereinigung, Produktbildung und Sternhülle charakterisierbar 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 beheben, 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 12. *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*

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

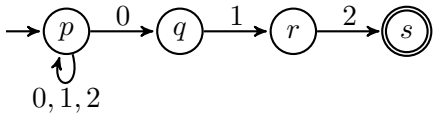
Ein NFA kann also nicht nur eine, sondern mehrere verschiedene Rechnungen ausführen. Die Eingabe gehört bereits dann zu $L(N)$, wenn bei einer dieser Rechnungen nach Lesen des gesamten Eingabewortes ein Endzustand erreicht wird.

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 13. *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 14. *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. 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 $L(N_1)L(N_2)$ *und 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 $L(N_1)^*$*.*

Satz 15 (Rabin und Scott)**.**
$\mathsf{REG} = \{L(N) \mid N \text{ ist ein NFA}\}$.

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 2) können wir nun folgende Behauptung zeigen:

> $\hat{\delta}'(Q_0, x)$ enthält alle Zustände, die N ausgehend von einem Startzustand nach Lesen der Eingabe 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.

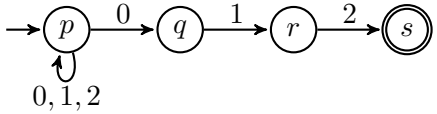
$$E' = \{Q \subseteq Z \mid Q \cap E \neq \varnothing\},$$

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

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

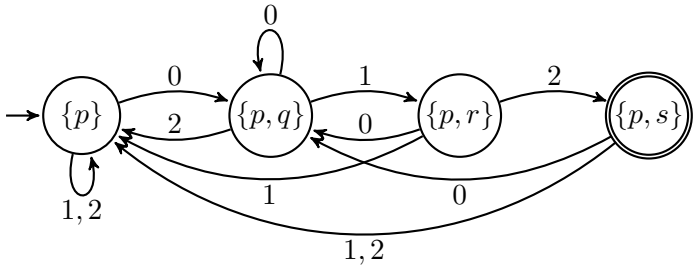
■

Beispiel 16. *Für den NFA $N = (Z, \Sigma, \delta, Q_0, E)$ aus Beispiel 13*



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

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

2.3 Reguläre Ausdrücke

Wir haben uns im letzten Abschnitt davon überzeugt, dass auch NFAs nur reguläre Sprachen erkennen können:

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

In diesem Abschnitt werden wir eine weitere Charakterisierung der regulären Sprachen kennen lernen:

> REG ist die Klasse aller Sprachen, die sich mittels der Operationen Vereinigung, Durchschnitt, Komplement, Produkt und Sternhülle aus der leeren Menge und den Singletonsprachen bilden lassen.

Tatsächlich kann hierbei sogar auf die Durchschnitts- und Komplementbildung verzichtet werden.

Definition 18. *Die Menge der* ***regulären Ausdrücke*** γ *(über einem Alphabet* Σ*) und die durch* γ *dargestellte Sprache* $L(\gamma)$ *sind induktiv wie folgt definiert. Die Symbole* $\varnothing$*,* ϵ *und* a *(*$a \in \Sigma$*) sind reguläre Ausdrücke, die*

- *die leere Sprache* $L(\varnothing) = \varnothing$*,*
- *die Sprache* $L(\epsilon) = \{\varepsilon\}$ *und*
- *für jedes Zeichen* $a \in \Sigma$ *die Sprache* $L(a) = \{a\}$

beschreiben. Sind α *und* β *reguläre Ausdrücke, die die Sprachen* $L(\alpha)$ *und* $L(\beta)$ *beschreiben, so sind auch* $\alpha\beta$*,* $(\alpha|\beta)$ *und* $(\alpha)^*$ *reguläre Ausdrücke, die die Sprachen*

- $L(\alpha\beta) = L(\alpha)L(\beta)$*,*
- $L(\alpha|\beta) = L(\alpha) \cup L(\beta)$ *und*
- $L((\alpha)^*) = L(\alpha)^*$

beschreiben.

Bemerkung 19.

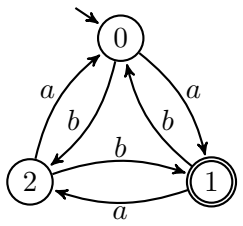
- *Um Klammern zu sparen, definieren wir folgende* ***Präzedenzordnung****: Der Sternoperator* * *bindet stärker als der Produktoperator und dieser wiederum stärker als der Vereinigungsoperator. Für* $((a|b(c)^*)|d)$ *können wir also kurz* $a|bc^*|d$ *schreiben.*
- *Da der reguläre Ausdruck* $\gamma\gamma^*$ *die Sprache* $L(\gamma)^+$ *beschreibt, verwenden wir* γ^+ *als Abkürzung für den Ausdruck* $\gamma\gamma^*$*.*

Beispiel 20. *Die regulären Ausdrücke* ϵ^**,* $\varnothing^*$*,* $(0|1)^*00$ *und* $(\epsilon 0|\varnothing 1^*)$ *beschreiben folgende Sprachen:*

γ	ϵ^*	$\varnothing^*$	$(0\|1)^*00$	$(\epsilon 0\|\varnothing 1^*)$
$L(\gamma)$	$\{\varepsilon\}^* = \{\varepsilon\}$	$\varnothing^* = \{\varepsilon\}$	$\{x00 \mid x \in \{0,1\}^*\}$	$\{0\}$

◁

Beispiel 21. *Betrachte nebenstehenden DFA M. Um für die von M erkannte Sprache*



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

einen regulären Ausdruck zu finden, betrachten wir zunächst die Sprache

$$L_0 = \{x \in \{a,b\}^* \mid \#_a(x) - \#_b(x) \equiv_3 0\}.$$

L_0 enthält also alle Wörter x, die den DFA M ausgehend vom Zustand 0 in den Zustand 0 überführen. Jedes solche x setzt sich aus beliebig vielen Teilwörtern y zusammen, die M vom Zustand 0 in den Zustand 0 überführen, ohne zwischendurch den Zustand 0 anzunehmen. Jedes solche y beginnt entweder mit einem a (Übergang von 0 nach 1) oder mit einem b (Übergang von 0 nach 2). Im ersten Fall folgt eine beliebige Anzahl von Teilwörtern ab (Wechsel zwischen 1 und 2), an die sich entweder das Suffix aa (Rückkehr von 1 nach 0 über 2) oder das Suffix b (direkte Rückkehr von 1 nach 0) anschließt. Analog folgt im zweiten Fall eine beliebige Anzahl von Teilwörtern ba (Wechsel zwischen 2 und 1), an die sich entweder das Suffix a (direkte Rückkehr von 2 nach 0) oder das Suffix bb (Rückkehr von 2 nach 0 über 1) anschließt. Daher lässt sich L_0 durch den regulären Ausdruck

$$\gamma_0 = (a(ab)^*(aa|b) \mid b(ba)^*(a|bb))^*$$

beschreiben. Eine ähnliche Überlegung zeigt, dass sich die Wörter, die M ausgehend von 0 in den Zustand 1 überführen, ohne dass zwischendurch der Zustand 0 nochmals besucht wird, durch den regulären Ausdruck $(a|bb)(ab)^*$ beschrieben werden. Somit erhalten wir für $L(M)$ den regulären Ausdruck $\gamma = \gamma_0(a|bb)(ab)^*$. ◁

Satz 22. *$\{L(\gamma) \mid \gamma$ ist ein regulärer Ausdruck$\} \subseteq \mathsf{REG}$.*

Beweis. Klar, da die Basisausdrücke $\varnothing$, ϵ und a, $a \in \Sigma^*$, nur reguläre Sprachen beschreiben und die Sprachklasse REG unter Produkt, Vereinigung und Sternhülle abgeschlossen ist (siehe Beobachtungen 11 und 14). ■