

Einführung in die Theoretische Informatik

Johannes Köbler



Institut für Informatik
Humboldt-Universität zu Berlin

WS 2020/21

Themen dieser VL:

- Welche Rechenmodelle eignen sich zur Lösung welcher algorithmischen Problemstellungen? **Automatentheorie**
- Welche algorithmischen Probleme sind überhaupt lösbar? **Berechenbarkeitstheorie**
- Welcher Aufwand ist zur Lösung eines geg. algorithmischen Problems nötig? **Komplexitätstheorie**

Themen der VL Algorithmen und Datenstrukturen:

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

Themen der VL Logik in der Informatik:

- Mathem. Grundlagen der Informatik, Beweise führen, Modellierung **Aussagenlogik, Prädikatenlogik**

- Überblick über die wichtigsten Rechenmodelle (Automaten) wie z.B.
 - endliche Automaten
 - Kellerautomaten
 - Turingmaschinen
 - Registermaschinen
 - Schaltkreise
- Charakterisierung der Klassen aller mit diesen Rechenmodellen lösbarer 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 math. 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
- Ältester bekannter nicht-trivialer Algorithmus: **Euklidischer Algorithmus** zur Berechnung des ggT (300 v. Chr.)
- Von einem Algorithmus wird erwartet, dass er bei jeder zulässigen **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

- Ein **Alphabet** ist eine endliche linear geordnete Menge

$$\Sigma = \{a_1, \dots, a_m\}$$

von $m \geq 1$ **Zeichen** $a_1 < \dots < a_m$

- Eine Folge $x = x_1 \dots x_n$ von $n \geq 0$ Zeichen $x_i \in \Sigma$ heißt **Wort** der **Länge** n über Σ
- Die Länge von x wird mit $|x|$ und die Menge aller Wörter der Länge n über Σ wird mit Σ^n bezeichnet
- Die Menge aller Wörter über Σ ist

$$\Sigma^* = \bigcup_{n \geq 0} \Sigma^n = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \dots$$

- Das (einzige) Wort der Länge $n = 0$ ist das **leere Wort**, welches wir mit ε bezeichnen, d.h. $\Sigma^0 = \{\varepsilon\}$
- Jede Teilmenge $L \subseteq \Sigma^*$ heißt **Sprache** über dem Alphabet Σ

Beispiel

- Sprachen über Σ sind beispielsweise \emptyset , Σ^* , Σ und $\{\varepsilon\}$
- \emptyset enthält keine Wörter und heißt **leere Sprache**
- Σ^* enthält dagegen alle Wörter über Σ
- Σ enthält alle Wörter über Σ der Länge 1
- $\{\varepsilon\}$ enthält nur das leere Wort, ist also einelementig
- Sprachen, die genau ein Wort enthalten, werden auch als **Singletonsprachen** bezeichnet
- in der Informatik spielen Programmiersprachen eine wichtige Rolle

- Da Sprachen Mengen sind, können wir sie bzgl. Inklusion vergleichen
- Zum Beispiel gilt $\emptyset \subseteq \{\varepsilon\} \subseteq \Sigma^*$
- Wir können Sprachen auch vereinigen, schneiden und komplementieren
- Seien A und B Sprachen über Σ . Dann ist
 - $A \cap B = \{x \in \Sigma^* \mid x \in A \wedge x \in B\}$ 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

Definition

Die **Konkatenation** von zwei Wörtern $x = x_1 \dots x_n$ und $y = y_1 \dots y_m$ ist das Wort $x \circ y = x_1 \dots x_n y_1 \dots y_m$, das wir auch einfach mit xy bezeichnen

Beispiel

- Für $x = aba$ und $y = abab$ erhalten wir $xy = abaabab$ und $yx = abababa$
- Die Konkatenation ist also nicht kommutativ
- Allerdings ist \circ assoziativ, d.h. es gilt $x(yz) = (xy)z$
Daher können wir hierfür auch einfach xyz schreiben
- Es gibt auch ein neutrales Element, da $x\varepsilon = \varepsilon x = x$ ist
- Eine algebraische Struktur (M, \square, e) mit einer assoziativen Operation $\square : M \times M \rightarrow M$ und einem neutralen Element e heißt **Monoid**
- $(\Sigma^*, \circ, \varepsilon)$ ist also ein Monoid

Neben den Mengenoperationen Schnitt, Vereinigung und Komplement gibt es auch spezielle Sprachoperationen

Definition

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

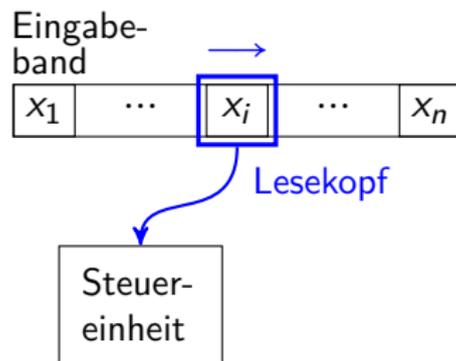
$$AB = \{xy \mid x \in A, y \in B\}$$

- Ist $A = \{x\}$ eine Singletonsprache, so schreiben wir für $\{x\}B$ auch einfach xB
- Die **n -fache Potenz** 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** einer Sprache A ist $A^* = \bigcup_{n \geq 0} A^n$
- Die **Plushülle** einer Sprache A ist $A^+ = \bigcup_{n \geq 1} A^n = AA^*$

- Ein einfaches Rechenmodell zum Erkennen von Sprachen ist der endliche Automat:



- Ein endlicher Automat
 - nimmt zu jedem Zeitpunkt genau einen von endlich vielen Zuständen an
 - macht bei Eingaben der Länge n genau n Rechenschritte und
 - liest in jedem Schritt genau ein Eingabezeichen

Definition

- Ein **endlicher Automat** (kurz: **DFA**; *Deterministic Finite Automaton*) wird durch ein 5-Tupel $M = (Z, \Sigma, \delta, q_0, E)$ beschrieben, wobei
 - $Z \neq \emptyset$ eine **endliche** Menge von **Zuständen**
 - Σ das **Eingabealphabet**
 - $\delta: Z \times \Sigma \rightarrow Z$ die **Überföhrungsfunktion**
 - $q_0 \in Z$ der **Startzustand** und
 - $E \subseteq Z$ die Menge der **Endzustände** ist
- Die von M **akzeptierte (oder erkannte) Sprache** ist

$$L(M) = \left\{ x_1 \dots x_n \in \Sigma^* \mid \begin{array}{l} \text{es gibt } q_1, \dots, q_{n-1} \in Z, q_n \in E \text{ mit} \\ \delta(q_i, x_{i+1}) = q_{i+1} \text{ f\u00fcr } i = 0, \dots, n-1 \end{array} \right\}$$

- Eine Zustandsfolge q_0, q_1, \dots, q_n hei\u00dft **Rechnung** von $M(x_1 \dots x_n)$, falls $\delta(q_i, x_{i+1}) = q_{i+1}$ f\u00fcr $i = 0, \dots, n-1$ gilt
- Sie hei\u00dft **akzeptierend**, falls $q_n \in E$ ist, und andernfalls **verwerfend**

Frage

Welche Sprachen lassen sich durch endliche Automaten erkennen und welche nicht?

Definition

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

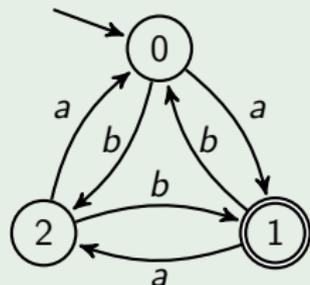
$$\text{REG} = \{L(M) \mid M \text{ ist ein DFA}\}$$

Beispiel

Sei $M_3 = (Z, \Sigma, \delta, 0, E)$ ein DFA mit $Z = \{0, 1, 2\}$, $\Sigma = \{a, b\}$, $E = \{1\}$ und der Überföhrungsfunktion

δ	0	1	2
a	1	2	0
b	2	0	1

Graphische Darstellung:



Endzustände werden durch einen doppelten Kreis und der Startzustand wird durch einen Pfeil gekennzeichnet

Frage: Welche Wörter akzeptiert M_3 ?

- Ist $w_1 = aba \in L(M_3)$? Ja (akzeptierende Rechnung: 0, 1, 0, 1)
- Ist $w_2 = abba \in L(M_3)$? Nein (verwerfende Rechnung: 0, 1, 0, 2, 0)

Behauptung

Die von M_3 erkannte Sprache ist

$$L(M_3) = \{x \in \{a, b\}^* \mid \#_a(x) - \#_b(x) \equiv_3 1\}, \text{ wobei}$$

- $\#_a(x)$ die Anzahl der Vorkommen von 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 der Behauptung durch Induktion über die Länge von x

Wir betrachten zunächst das Erreichbarkeitsproblem für DFAs

Frage

Sei $M = (Z, \Sigma, \delta, q_0, E)$ ein DFA und sei $x = x_1 \dots x_n \in \Sigma^*$. Welchen Zustand erreicht M nach Lesen der Eingabe x ?

Antwort

- nach 0 Schritten: den Startzustand q_0
- nach 1 Schritt: den Zustand $\delta(q_0, x_1)$
- nach 2 Schritten: den Zustand $\delta(\delta(q_0, x_1), x_2)$
- \vdots
- nach n Schritten: den Zustand $\delta(\dots \delta(\delta(q_0, x_1), x_2), \dots x_n)$

Das Erreichbarkeitsproblem für DFAs

Definition

- 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

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

induktiv über die Länge von x wie folgt definieren:

Für $q \in Z$, $x \in \Sigma^*$ und $a \in \Sigma$ sei

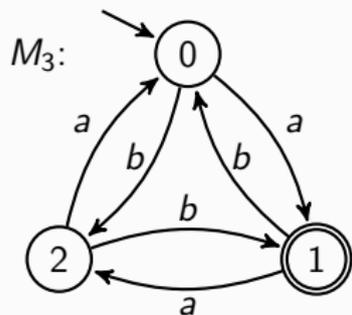
$$\begin{aligned}\hat{\delta}(q, \varepsilon) &= q, \\ \hat{\delta}(q, xa) &= \delta(\hat{\delta}(q, x), a)\end{aligned}$$

- Die von M erkannte Sprache lässt sich nun elegant durch

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

beschreiben

DFAs beherrschen Modulare Arithmetik



Behauptung

Für alle $x \in \{a, b\}^*$ gilt:

$$x \in L(M_3) \Leftrightarrow \#_a(x) - \#_b(x) \equiv_3 1$$

Beweis

- 1 ist der einzige Endzustand von M
- Daher ist $L(M_3) = \{x \in \{a, b\}^* \mid \hat{\delta}(0, x) = 1\}$
- Obige Behauptung ist also äquivalent zu

für alle $x \in \{a, b\}^*$ gilt: $\hat{\delta}(0, x) = 1 \Leftrightarrow \#_a(x) - \#_b(x) \equiv_3 1$
- Folglich reicht es, für alle $x \in \{a, b\}^*$ folgende Kongruenz zu zeigen:

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

Induktionsbehauptung: Für alle $x \in \{a, b\}^n$ gilt $\hat{\delta}(0, x) \equiv_3 \#_a(x) - \#_b(x)$

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

Induktionsschritt ($n \rightsquigarrow n + 1$): Sei $x = x_1 \dots x_{n+1} \in \{a, b\}^{n+1}$ gegeben

- Nach Induktionsvoraussetzung (IV) gilt für $x' = x_1 \dots x_n$:

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

- Zudem gilt für alle $i \in Z = \{0, 1, 2\}$:

$$\begin{aligned} \delta(i, x_{n+1}) &\equiv_3 \begin{cases} i + 1, & x_{n+1} = a \\ i - 1, & x_{n+1} = b \end{cases} \\ &= i + \#_a(x_{n+1}) - \#_b(x_{n+1}) \end{aligned} \quad (*)$$

- Somit folgt

$$\begin{aligned} \hat{\delta}(0, x) &= \delta(\hat{\delta}(0, x'), x_{n+1}) \\ &\equiv_3 \hat{\delta}(0, x') + \#_a(x_{n+1}) - \#_b(x_{n+1}) \quad (*) \\ &\equiv_3 \#_a(x') - \#_b(x') + \#_a(x_{n+1}) - \#_b(x_{n+1}) \quad (IV) \\ &\equiv_3 \#_a(x) - \#_b(x) \end{aligned}$$

Singletonsprachen sind regulär

Vereinbarung

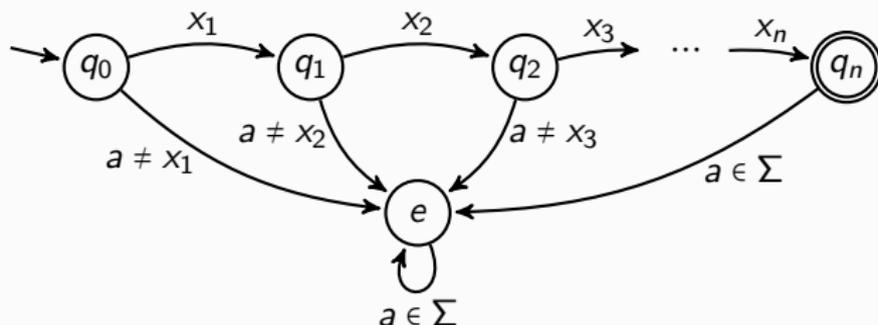
Für das Folgende sei $\Sigma = \{a_1, \dots, a_m\}$ ein fest gewähltes Alphabet

Beobachtung 1

Alle Sprachen, die nur ein Wort $x = x_1 \dots x_n \in \Sigma^*$ enthalten, sind regulär

Beweis

Folgender DFA M erkennt die Sprache $L(M) = \{x\}$:



REG ist unter Komplement abgeschlossen

Beobachtung 2

Ist $L \in \text{REG}$, so ist auch die Sprache $\bar{L} = \Sigma^* \setminus L$ regulär

Beweis

- Sei $M = (Z, \Sigma, \delta, q_0, E)$ ein DFA mit $L(M) = L$
- Dann wird das Komplement \bar{L} von L von dem DFA $\bar{M} = (Z, \Sigma, \delta, q_0, Z \setminus E)$ akzeptiert

□

Definition

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

Korollar

$\text{co-REG} = \text{REG}$

Beobachtung 3

Sind $L_1, L_2 \in \text{REG}$, so ist auch die Sprache $L_1 \cap L_2$ regulär

Beweis

- Seien $M_i = (Z_i, \Sigma, \delta_i, q_i, E_i)$, $i = 1, 2$, DFAs mit $L(M_i) = L_i$.
- Dann wird der Schnitt $L_1 \cap L_2$ von dem DFA

$$M = (Z_1 \times Z_2, \Sigma, \delta, (q_1, q_2), E_1 \times E_2)$$

mit

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

erkannt

- M wird auch als **Kreuzproduktautomat** bezeichnet



Beobachtung 4

Die Vereinigung $L_1 \cup L_2$ von regulären Sprachen L_1 und L_2 ist regulär

Beweis

Es gilt $L_1 \cup L_2 = \overline{(\overline{L_1} \cap \overline{L_2})}$ □

Frage

Wie sieht der zugehörige DFA aus?

Antwort

$$M' = (Z_1 \times Z_2, \Sigma, \delta, (q_1, q_2), (E_1 \times Z_2) \cup (Z_1 \times E_2))$$

Abschlusseigenschaften von Sprachklassen

Definition

- 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
- 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

- Der 2-stellige Schnittoperator \cap bildet L_1 und L_2 auf $L_1 \cap L_2$ ab
- 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

Korollar

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

- Komplement
- Schnitt
- Vereinigung

Folgerung

- Aus den Beobachtungen folgt, dass alle **endlichen** und alle **co-endlichen** Sprachen regulär sind
- Da die reguläre Sprache

$$L(M_3) = \{x \in \{a, b\}^* \mid \#_a(x) - \#_b(x) \equiv_3 1\}$$

weder endlich noch co-endlich ist, haben wir damit allerdings noch nicht alle regulären Sprachen erfasst

Wie umfangreich ist REG?

Nächstes Ziel

Zeige, dass REG unter Produktbildung und Sternhülle abgeschlossen ist

Problem

Bei der Konstruktion eines DFA M für das Produkt $L(M_1)L(M_2)$ bereitet es Schwierigkeiten, den richtigen Zeitpunkt für das Ende der Simulation des DFA M_1 und den Start der Simulation des DFA M_2 zu finden

Lösungsidee

Ein **nichtdeterministischer** endlicher Automat (NFA) kann den richtigen Zeitpunkt „raten“

Verbleibendes Problem

Zeige, dass auch NFAs nur reguläre Sprachen erkennen

Nichtdeterministische endliche Automaten

Definition

- Ein **nichtdet. endl. Automat** (kurz: **NFA**; *Nondet. Finite Automaton*)

$$N = (Z, \Sigma, \Delta, Q_0, E)$$

ist genau so aufgebaut wie ein DFA, nur dass er

- eine Menge $Q_0 \subseteq Z$ von Startzuständen hat und
- die Überföhrungsfunktion folgende Form hat

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

Hierbei bezeichnet $\mathcal{P}(Z)$ die **Potenzmenge** (also die Menge aller Teilmengen) von Z ; diese wird oft auch mit 2^Z bezeichnet

- Die von einem NFA N **akzeptierte (oder erkannte) Sprache** ist

$$L(N) = \left\{ x_1 \dots x_n \in \Sigma^* \mid \begin{array}{l} \text{es gibt } q_0 \in Q_0, q_1, \dots, q_{n-1} \in Z, q_n \in E \\ \text{mit } q_{i+1} \in \Delta(q_i, x_{i+1}) \text{ f\u00fcr } i = 0, \dots, n-1 \end{array} \right\}$$

- Eine Zustandsfolge q_0, \dots, q_n hei\u00dft **Rechnung** von $N(x_1 \dots x_n)$, falls $q_0 \in Q_0$ und $q_{i+1} \in \Delta(q_i, x_{i+1})$ f\u00fcr $i = 0, \dots, n-1$ gilt

Eigenschaften von NFAs

- 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, der jede Eingabe zu Ende liest, kann ein NFA N „stecken bleiben“
- Dieser Fall tritt ein, wenn N in einen Zustand q gelangt, in dem er das nächste Eingabezeichen x_i wegen

$$\Delta(q, x_i) = \emptyset$$

nicht verarbeiten kann

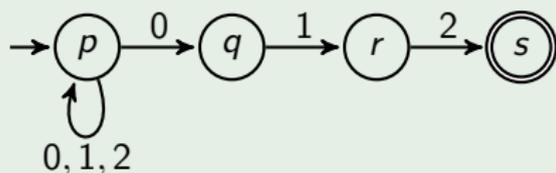
Eigenschaften von NFAs

Beispiel

- Betrachte den NFA $N = (Z, \Sigma, \Delta, Q_0, E)$ mit $Z = \{p, q, r, s\}$, $\Sigma = \{0, 1, 2\}$, $Q_0 = \{p\}$, $E = \{s\}$ und der Überföhrungsfunktion

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

Graphische Darstellung:



- Ist $w_1 = 012 \in L(N)$? **Ja** (akzeptierende Rechnung: p, q, r, s)
Es gibt aber auch verwerfende Rechnungen bei Eingabe w_1 : p, p, p, p
- Ist $w_2 = 021 \in L(N)$? **Nein**, da es keine akzeptierende Rechnung gibt
- Es gilt $L(N) = \{x012 \mid x \in \Sigma^*\}$

Beobachtung 5

Seien $N_i = (Z_i, \Sigma, \Delta_i, Q_i, E_i)$ NFAs mit $L(N_i) = L_i$ für $i = 1, 2$. Dann wird auch das Produkt $L_1 L_2$ von einem NFA erkannt

Beweis

- Wir können $Z_1 \cap Z_2 = \emptyset$ annehmen
- Dann gilt $L(N) = L_1 L_2$ für den NFA $N = (Z_1 \cup Z_2, \Sigma, \Delta, Q_1, E)$ mit

$$\Delta(p, a) = \begin{cases} \Delta_1(p, a), & p \in Z_1 \setminus E_1, \\ \Delta_1(p, a) \cup \bigcup_{q \in Q_2} \Delta_2(q, a), & p \in E_1, \\ \Delta_2(p, a), & p \in Z_2 \end{cases}$$

und

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

Ein NFA für das Produkt von regulären Sprachen

- Dann gilt $L(N) = L_1 L_2$ für den NFA $N = (Z_1 \cup Z_2, \Sigma, \Delta, Q_1, E)$ mit

$$\Delta(p, a) = \begin{cases} \Delta_1(p, a), & p \in Z_1 \setminus E_1, \\ \Delta_1(p, a) \cup \bigcup_{q \in Q_2} \Delta_2(q, a), & p \in E_1, \\ \Delta_2(p, a), & p \in Z_2 \end{cases}$$

und $E = E_2$, falls $Q_2 \cap E_2 = \emptyset$, bzw. $E = E_1 \cup E_2$ sonst

Beweis von $L_1 L_2 \subseteq L(N)$:

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

Ein NFA für das Produkt von regulären Sprachen

- Dann gilt $L(N) = L_1 L_2$ für den NFA $N = (Z_1 \cup Z_2, \Sigma, \Delta, Q_1, E)$ mit

$$\Delta(p, a) = \begin{cases} \Delta_1(p, a), & p \in Z_1 \setminus E_1, \\ \Delta_1(p, a) \cup \bigcup_{q \in Q_2} \Delta_2(q, a), & p \in E_1, \\ \Delta_2(p, a), & p \in Z_2 \end{cases}$$

und $E = E_2$, falls $Q_2 \cap E_2 = \emptyset$, bzw. $E = E_1 \cup E_2$ sonst

Beweis von $L(N) \subseteq L_1 L_2$:

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 q_0, \dots, q_i eine akz. Rechnung von $N_1(x_1 \cdots x_i)$ und

q, q_{i+1}, \dots, q_n für ein $q \in Q_2$ 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 $q_i \in E_1$ und $q_{i+1} \in \Delta_2(q, x_{i+1})$ für ein $q \in Q_2$ ist. Zudem ist $q_n \in E \cap Z_2 = E_2$
- Im Fall $i = n$ ist $q_n \in E \cap Z_1$, was $q_n \in E_1$ und $Q_2 \cap E_2 \neq \emptyset$ impliziert

Ein NFA für die Sternhülle einer regulären Sprache

Beobachtung 6

Ist $N = (Z, \Sigma, \Delta, Q_0, E)$ ein NFA, so wird auch die Sprache $L(N)^*$ von einem NFA erkannt

Beweis

Die Sprache $L(N)^*$ wird von dem NFA

$$N' = (Z \cup \{q_{neu}\}, \Sigma, \Delta', Q_0 \cup \{q_{neu}\}, E \cup \{q_{neu}\})$$

mit

$$\Delta'(p, a) = \begin{cases} \Delta(p, a), & p \in Z \setminus E, \\ \Delta(p, a) \cup \bigcup_{q \in Q_0} \Delta(q, a), & p \in E, \\ \emptyset, & p = q_{neu} \end{cases}$$

erkannt

Ziel

Zeige, dass REG unter Produktbildung und Sternhülle abgeschlossen ist

Problem

Bei der Konstruktion eines DFA für das Produkt L_1L_2 bereitet es Schwierigkeiten, den richtigen Zeitpunkt für den Übergang von (der Simulation von) M_1 zu M_2 zu finden

Lösungsidee (bereits umgesetzt)

Ein **nichtdeterministischer** Automat (NFA) kann den richtigen Zeitpunkt für den Übergang „raten“

Noch zu zeigen

NFAs erkennen genau die regulären Sprachen

NFAs erkennen genau die regulären Sprachen

Satz (Rabin und Scott)

$$\text{REG} = \{L(N) \mid N \text{ ist ein NFA}\}$$

Beweis von $\text{REG} \subseteq \{L(N) \mid N \text{ ist ein NFA}\}$

Diese Inklusion ist klar, da jeder DFA $M = (Z, \Sigma, \delta, q_0, E)$ in einen äquivalenten NFA

$$N = (Z, \Sigma, \Delta, Q_0, E)$$

transformiert werden kann, indem wir $\Delta(q, a) = \{\delta(q, a)\}$ und $Q_0 = \{q_0\}$ setzen. □

Für die umgekehrte Inklusion ist das **Erreichbarkeitsproblem für NFAs** von zentraler Bedeutung

Frage

Sei $N = (Z, \Sigma, \Delta, Q_0, E)$ ein NFA und sei $x = x_1 \dots x_n$ eine Eingabe. Welche Zustände sind in i Schritten erreichbar?

Antwort

- in 0 Schritten: alle Zustände in Q_0
- in einem Schritt: alle Zustände in

$$Q_1 = \bigcup_{q \in Q_0} \Delta(q, x_1)$$

- in i Schritten: alle Zustände in

$$Q_i = \bigcup_{q \in Q_{i-1}} \Delta(q, x_i)$$

Idee

- Wir können einen NFA $N = (Z, \Sigma, \Delta, Q_0, E)$ durch einen DFA $M = (Z', \Sigma, \delta, q'_0, E')$ simulieren, der in seinem Zustand die Information speichert, in welchen Zuständen sich N momentan befinden könnte
- Die Zustände von M sind also Teilmengen Q von Z (d.h. $Z' = \mathcal{P}(Z)$) mit Q_0 als Startzustand (d.h. $q'_0 = Q_0$) und der Endzustandsmenge

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

- Die Überföhrungsfunktion $\delta : \mathcal{P}(Z) \times \Sigma \rightarrow \mathcal{P}(Z)$ von M berechnet dann für einen Zustand $Q \subseteq Z$ und ein Zeichen $a \in \Sigma$ die Menge

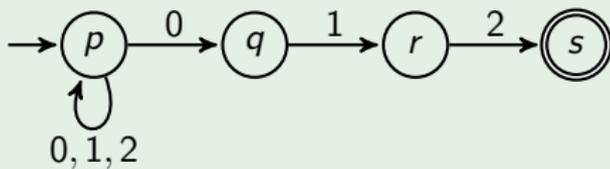
$$\delta(Q, a) = \bigcup_{q \in Q} \Delta(q, a)$$

aller Zustände, in die N gelangen kann, wenn N ausgehend von einem beliebigen Zustand $q \in Q$ das Zeichen a liest

- M wird auch als der zu N gehörige **Potenzmengenautomat** bezeichnet

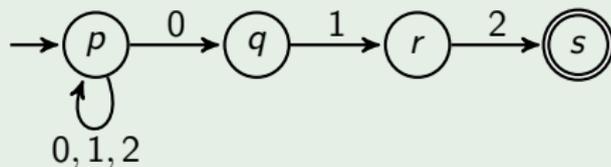
Beispiel

- Betrachte den NFA N



Beispiel

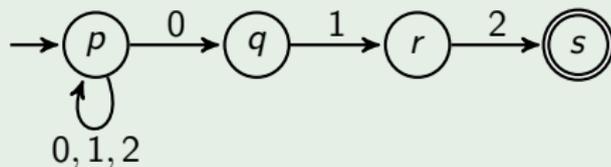
- Betrachte den NFA N



- Ausgehend von $Q_0 = \{p\}$ liefert δ dann die folgenden Werte:

Beispiel

- Betrachte den NFA N

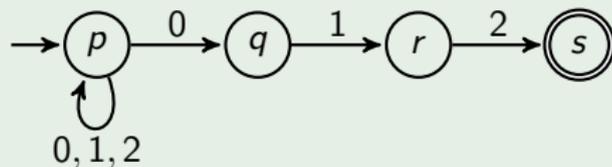


- Ausgehend von $Q_0 = \{p\}$ liefert δ dann die folgenden Werte:

δ	0	1	2

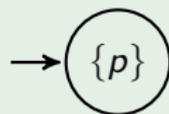
Beispiel

- Betrachte den NFA N



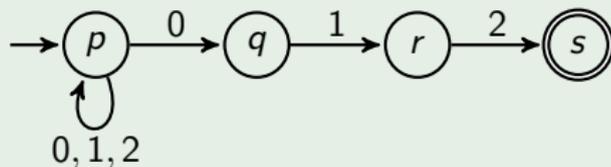
- Ausgehend von $Q_0 = \{p\}$ liefert δ dann die folgenden Werte:

δ	0	1	2
$\{p\}$			



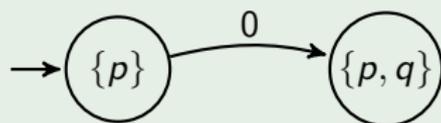
Beispiel

- Betrachte den NFA N



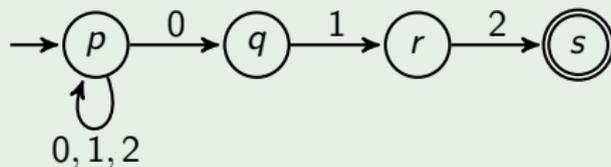
- Ausgehend von $Q_0 = \{p\}$ liefert δ dann die folgenden Werte:

δ	0	1	2
$\{p\}$	$\{p, q\}$		



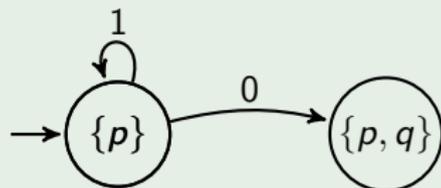
Beispiel

- Betrachte den NFA N



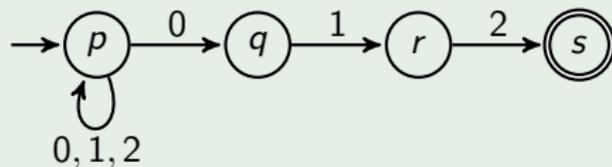
- Ausgehend von $Q_0 = \{p\}$ liefert δ dann die folgenden Werte:

δ	0	1	2
$\{p\}$	$\{p, q\}$	$\{p\}$	



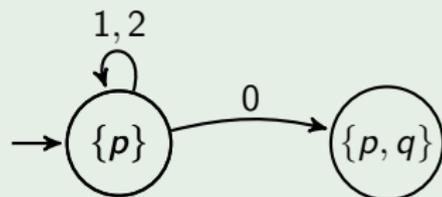
Beispiel

- Betrachte den NFA N



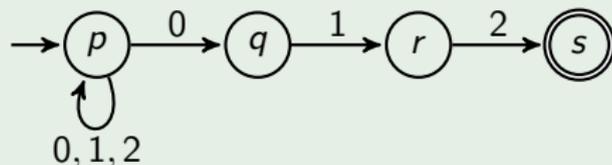
- Ausgehend von $Q_0 = \{p\}$ liefert δ dann die folgenden Werte:

δ	0	1	2
$\{p\}$	$\{p, q\}$	$\{p\}$	$\{p\}$



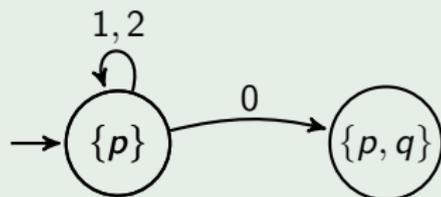
Beispiel

- Betrachte den NFA N



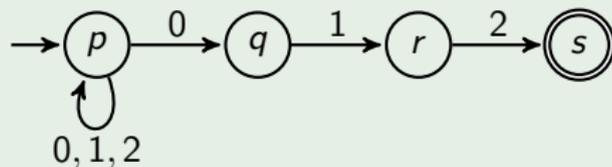
- Ausgehend von $Q_0 = \{p\}$ liefert δ dann die folgenden Werte:

δ	0	1	2
$\{p\}$	$\{p, q\}$	$\{p\}$	$\{p\}$
$\{p, q\}$			



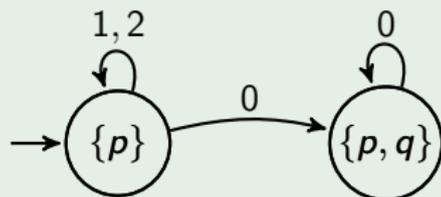
Beispiel

- Betrachte den NFA N



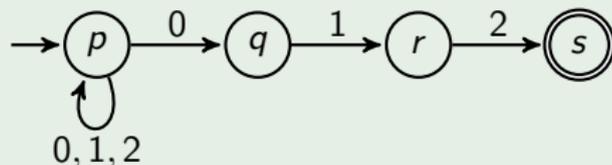
- Ausgehend von $Q_0 = \{p\}$ liefert δ dann die folgenden Werte:

δ	0	1	2
$\{p\}$	$\{p, q\}$	$\{p\}$	$\{p\}$
$\{p, q\}$	$\{p, q\}$		



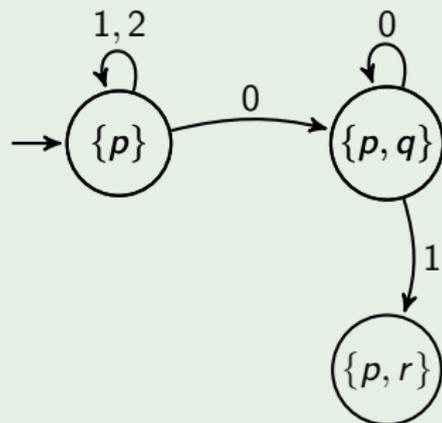
Beispiel

- Betrachte den NFA N



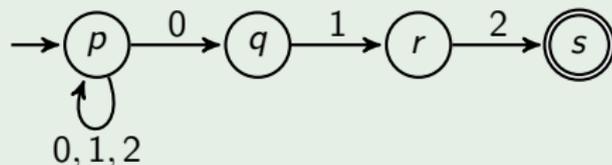
- Ausgehend von $Q_0 = \{p\}$ liefert δ dann die folgenden Werte:

δ	0	1	2
$\{p\}$	$\{p, q\}$	$\{p\}$	$\{p\}$
$\{p, q\}$	$\{p, q\}$	$\{p, r\}$	



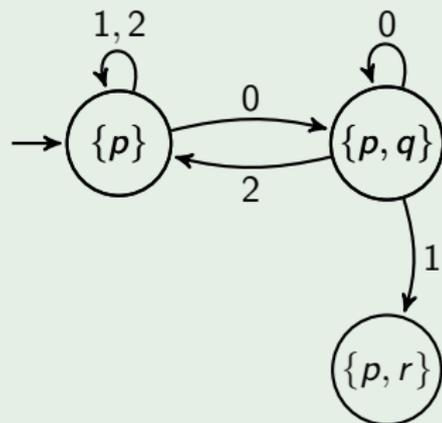
Beispiel

- Betrachte den NFA N



- Ausgehend von $Q_0 = \{p\}$ liefert δ dann die folgenden Werte:

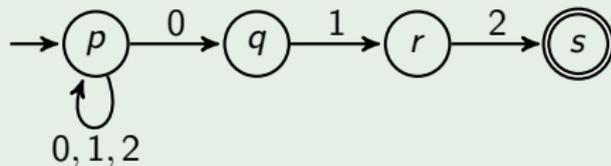
δ	0	1	2
$\{p\}$	$\{p, q\}$	$\{p\}$	$\{p\}$
$\{p, q\}$	$\{p, q\}$	$\{p, r\}$	$\{p\}$



Simulation von NFAs durch DFAs

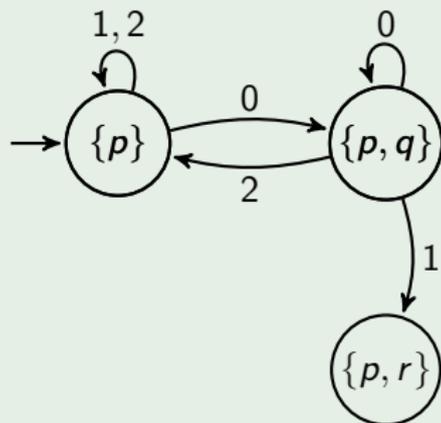
Beispiel

- Betrachte den NFA N



- Ausgehend von $Q_0 = \{p\}$ liefert δ dann die folgenden Werte:

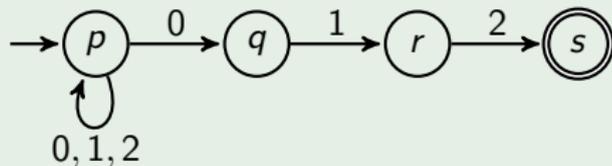
δ	0	1	2
$\{p\}$	$\{p, q\}$	$\{p\}$	$\{p\}$
$\{p, q\}$	$\{p, q\}$	$\{p, r\}$	$\{p\}$
$\{p, r\}$			



Simulation von NFAs durch DFAs

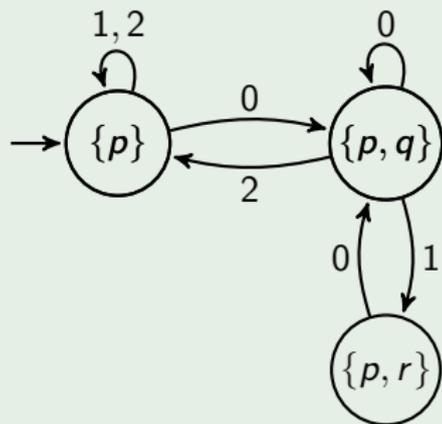
Beispiel

- Betrachte den NFA N



- Ausgehend von $Q_0 = \{p\}$ liefert δ dann die folgenden Werte:

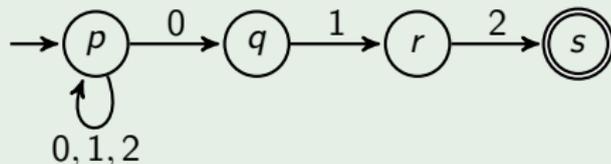
δ	0	1	2
$\{p\}$	$\{p, q\}$	$\{p\}$	$\{p\}$
$\{p, q\}$	$\{p, q\}$	$\{p, r\}$	$\{p\}$
$\{p, r\}$	$\{p, q\}$		



Simulation von NFAs durch DFAs

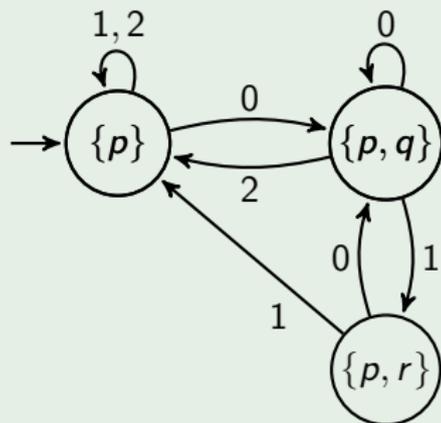
Beispiel

- Betrachte den NFA N



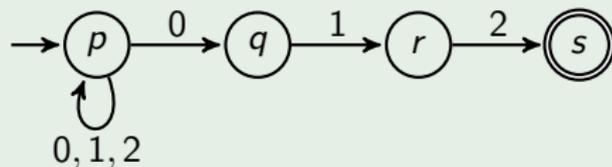
- Ausgehend von $Q_0 = \{p\}$ liefert δ dann die folgenden Werte:

δ	0	1	2
$\{p\}$	$\{p, q\}$	$\{p\}$	$\{p\}$
$\{p, q\}$	$\{p, q\}$	$\{p, r\}$	$\{p\}$
$\{p, r\}$	$\{p, q\}$	$\{p\}$	



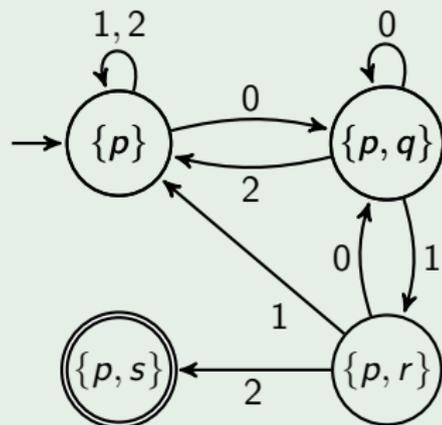
Beispiel

- Betrachte den NFA N



- Ausgehend von $Q_0 = \{p\}$ liefert δ dann die folgenden Werte:

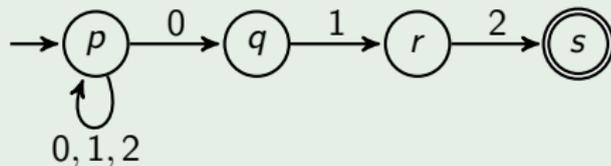
δ	0	1	2
$\{p\}$	$\{p, q\}$	$\{p\}$	$\{p\}$
$\{p, q\}$	$\{p, q\}$	$\{p, r\}$	$\{p\}$
$\{p, r\}$	$\{p, q\}$	$\{p\}$	$\{p, s\}$



Simulation von NFAs durch DFAs

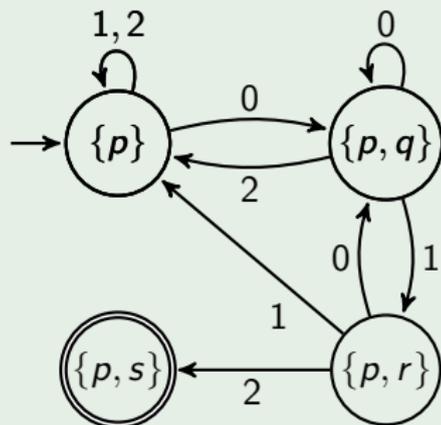
Beispiel

- Betrachte den NFA N



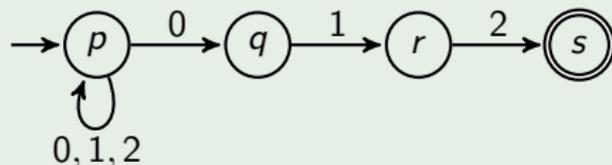
- Ausgehend von $Q_0 = \{p\}$ liefert δ dann die folgenden Werte:

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



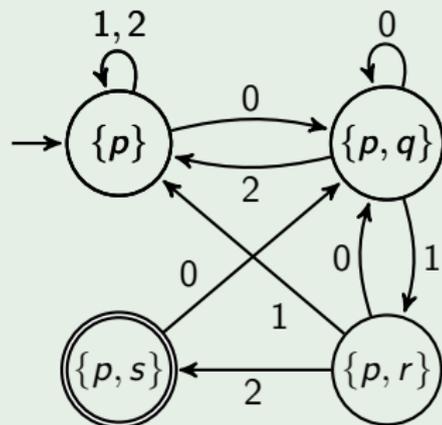
Beispiel

- Betrachte den NFA N



- Ausgehend von $Q_0 = \{p\}$ liefert δ dann die folgenden Werte:

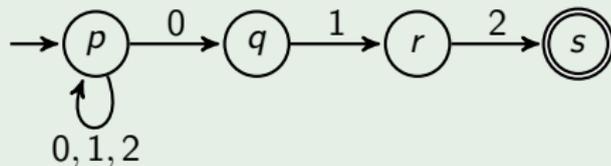
δ	0	1	2
$\{p\}$	$\{p, q\}$	$\{p\}$	$\{p\}$
$\{p, q\}$	$\{p, q\}$	$\{p, r\}$	$\{p\}$
$\{p, r\}$	$\{p, q\}$	$\{p\}$	$\{p, s\}$
$\{p, s\}$	$\{p, q\}$		



Simulation von NFAs durch DFAs

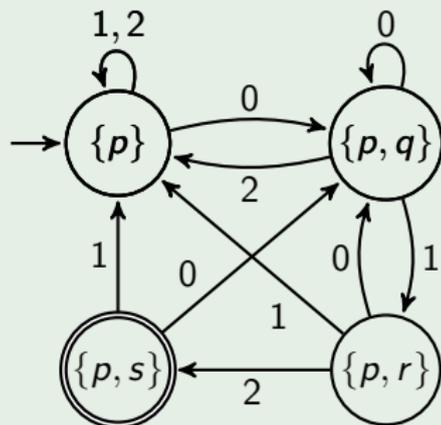
Beispiel

- Betrachte den NFA N



- Ausgehend von $Q_0 = \{p\}$ liefert δ dann die folgenden Werte:

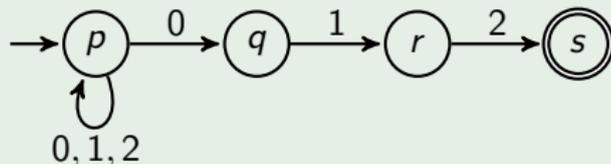
δ	0	1	2
$\{p\}$	$\{p, q\}$	$\{p\}$	$\{p\}$
$\{p, q\}$	$\{p, q\}$	$\{p, r\}$	$\{p\}$
$\{p, r\}$	$\{p, q\}$	$\{p\}$	$\{p, s\}$
$\{p, s\}$	$\{p, q\}$	$\{p\}$	



Simulation von NFAs durch DFAs

Beispiel

- Betrachte den NFA N



- Ausgehend von $Q_0 = \{p\}$ liefert δ dann die folgenden Werte:

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

