Vorlesungsskript

Theoretische Informatik II

Wintersemester 2007/2008

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

30. November 2007

Inhaltsverzeichnis

1 Reguläre Sprachen **1**

1.1 Endliche Automaten 1
1.2 Nichtdeterministische endliche Automaten 5
1.3 Reguläre Ausdrücke 8
1.4 Relationalstrukturen 11
1.4.1 Eigenschaften von Relationen 13
1.4.2 Äquivalenz- und Ordnungsrelationen 17
1.4.3 Abbildungen 22
1.4.4 Homo- und Isomorphismen 23
1.5 Minimierung von DFAs 25
1.6 Grammatiken 30
1.7 Das Pumping-Lemma 34

2 Kontextfreie Sprachen **37**

2.1 Kellerautomaten 37
2.2 Äquivalenz von kontextfreien Grammatiken und Kellerautomaten . . 39
2.3 Das Wortproblem für kontextfreie Grammatiken 44
2.3.1 Chomsky-Normalform 44
2.3.2 Der CYK-Algorithmus 47
2.4 Das Pumping-Lemma für kontextfreie Sprachen 49

Kapitel 1

Reguläre Sprachen

1.1 Endliche Automaten

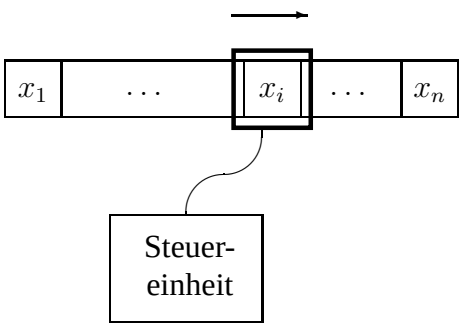
Rechenmaschinen spielen in der Informatik eine zentrale Rolle. Hier beschäftigen wir uns mit mathematischen Modellen für Maschinentypen von unterschiedlicher Berechnungskraft. In der Vorlesung Theoretische Informatik 1 wurde die Turingmaschine als ein universales Berechnungsmodell eingeführt. In dieser Vorlesung werden wir eine Reihe von Einschränkungen dieses Maschinenmodells kennenlernen, die vielfältige praktische Anwendungen haben. Dabei betrachten wir zunächst nur Entscheidungsprobleme, was der Berechnung von $\{0,1\}$-wertigen Funktionen entspricht. Zur Beschreibung der Problemeingaben wird ein Eingabealphabet Σ verwendet.

Definition 1.1 *Ein **Alphabet** ist eine geordnete endliche Menge $\Sigma = \{a_1, \ldots, a_m\}$ von **Zeichen**. Eine Folge $x = x_1 \ldots x_n \in \Sigma^n$ heißt **Wort** (der **Länge** n). Die Menge aller Wörter über Σ ist*

$$\Sigma^* = \bigcup_{n \geq 0} \Sigma^n = \{x_1 \cdots x_n \mid n \geq 0 \text{ und } x_i \in \Sigma \text{ für } i = 1, \ldots, n\}.$$

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

Ein endlicher Automat ist eine auf ein Minimum „abgespeckte“ Turingmaschine, die nur kostant viel Speicherplatz zur Verfügung hat und bei Eingaben der Länge n nur n Rechenschritte ausführen darf. Um die gesamte Eingabe lesen zu können, muss der Automat also in jedem Schritt ein Zeichen der Eingabe verarbeiten.



Definition 1.2 *Ein **endlicher Automat** (kurz: DFA; deterministic finite automaton) wird durch ein 5-Tupel $M = (Z, \Sigma, \delta, q_0, E)$ beschrieben, wobei*

- *Z 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.*

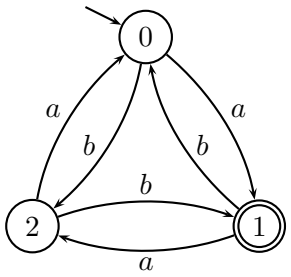
*Die von M **akzeptierte** oder **erkannte Sprache** ist*

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

Beispiel 1.3 Betrachte den DFA $M = (Z, \Sigma, \delta, 0, E)$ mit $Z = \{0, 1, 2\}$, $\Sigma = \{a, b\}$, $E = \{1\}$ und der Überführungsfunktion

δ	a	b
0	1	2
1	2	0
2	0	1

Graphische Darstellung:



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

Behauptung 1.4 *M akzeptiert die Sprache*

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

wobei $\#_a(x)$ die Anzahl der Vorkommen des Buchstabens a in x bezeichnet. (Für $j \equiv k \pmod m$ schreiben wir im Folgenden auch kurz $j \equiv_m k$.)

Beweis 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

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

Da 1 der einzige Endzustand von M ist, reicht es, folgende Kongruenzgleichung zu zeigen:

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

Wir beweisen die Kongruenz induktiv über die Länge von x.

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

Induktionsschritt ($n \rightsquigarrow n+1$): Sei $x = x_1 \cdots x_{n+1}$ gegeben. Nach IV ist

$$\hat{\delta}(0, x_1 \cdots x_n) \equiv_3 \#_a(x_1 \cdots x_n) - \#_b(x_1 \cdots x_n).$$

Wegen

$$\delta(i, a) \equiv_3 i + 1 \text{ und } \delta(i, b) \equiv_3 i - 1$$

folgt daher sofort

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

□

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

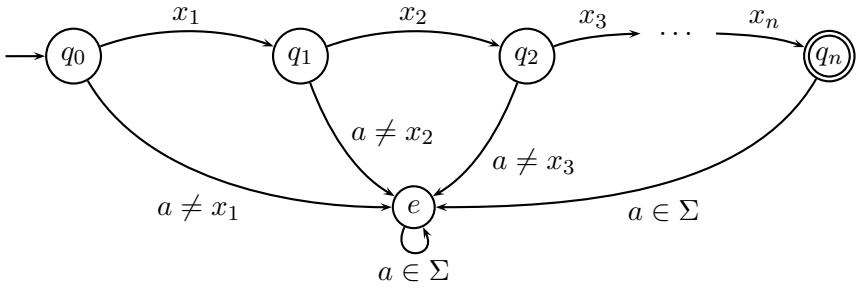
$$\mathsf{REG} = \{L(M) \mid M \text{ ist ein DFA}\}.$$

Um ein intuitives Verständnis für die Berechnungskraft von DFAs zu entwickeln, werden wir uns zunächst intensiv mit der Beantwortung folgender Frage beschäftigen.

Frage: Welche Sprachen gehören zu REG und welche nicht?

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

Beobachtung 1.5 *Alle Sprachen, die aus einem einzigen Wort $x = x_1 \cdots x_n \in \Sigma^*$ bestehen, sind regulär. Für folgenden DFA M gilt nämlich $L(M) = \{x\}$.*



Formal lässt sich M also durch das Tupel $M = (Z, \Sigma, \delta, q_0, E)$ mit $Z = \{q_0, \ldots, q_n, e\}$, $E = \{q_n\}$ und der Überführungsfunktion

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

beschreiben.

Als nächstes betrachten wir Abschlusseigenschaften der Sprachklasse REG.

Definition 1.6 *Ein* **(k-stelliger) Sprachoperator** *ist eine Abbildung op, die k Sprachen $L_1, \ldots, L_k$ auf eine Sprache $op(L_1, \ldots, L_k)$ abbildet.*

Beispiel 1.7 Der 2-stellige Schnittoperator bildet zwei Sprachen L_1 und L_2 auf die Sprache $L_1 \cap L_2$ ab. ◁

Definition 1.8 *Eine Sprachklasse $\mathcal{K}$ heißt unter op* **abgeschlossen**, *wenn gilt:*

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

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

Beobachtung 1.9 *Mit $L_1, L_2 \in$ 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 1.9 folgt, dass alle endlichen und alle co-endlichen Sprachen regulär sind. Da die in Beispiel 1.3 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

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

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_1L_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.

1.2 Nichtdeterministische endliche Automaten

Definition 1.10 *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*

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

die Potenzmenge $\mathcal{P}(Z)$ von Z als Wertebereich hat. Die von N akzeptierte Sprache ist

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

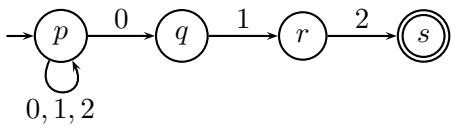
Ein NFA kann also nicht nur eine, sondern mehrere verschiedene Rechnungen ausführen. Die Eingabe gehört bereits dann zu $L(N)$, wenn bei einer dieser Rechnungen nach Lesen des gesamten Eingabewortes ein Endzustand erreicht wird.

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 $\delta(q, x_i) = \emptyset$ nicht gelesen werden kann.

Beispiel 1.11 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	$\{p, q\}$	$\emptyset$	$\emptyset$	$\emptyset$
1	$\{p\}$	$\{r\}$	$\emptyset$	$\emptyset$
2	$\{p\}$	$\emptyset$	$\{s\}$	$\emptyset$

Graphische Darstellung:



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

Beobachtung 1.12 *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 $Z_1 \cap Z_2 = \emptyset$ annehmen. Dann akzeptiert der NFA*

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

mit

$$\delta(p,a) = \begin{cases} \delta_1(p,a), & p \in Z_1 \setminus E_1, \\ \delta_1(p,a) \cup \bigcup_{q \in Q_2} \delta_2(q,a), & p \in E_1, \\ \delta_2(p,a), & \textit{sonst} \end{cases}$$

und

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

die Sprache $L(N_1)L(N_2)$ und der NFA

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

mit

$$\delta^*(p,a) = \begin{cases} \delta(p,a) \cup \bigcup_{q \in Q_1} \delta(q,a), & p \in E_1, \\ \delta(p,a), & \textit{sonst} \end{cases}$$

die Sprache $L(N_1)^*$.

Theorem 1.13 $\mathsf{REG} = \{L(N) \mid N \textit{ ist ein NFA}\}$.

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 mit $L(M) = L(N)$. Zunächst erweitern wir die Überführungsfunktion $\delta : Z \times \Sigma \to \mathcal{P}(Z)$ zu $\delta' : \mathcal{P}(Z) \times \Sigma \to \mathcal{P}(Z)$ mittels

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

und zeigen folgende Behauptung:

> $\hat{\delta}'(Q_0, x)$ enthält alle von N bei Eingabe x in $|x|$ Schritten erreichbaren Zustände.

Wir beweisen die Behauptung induktiv über die Länge von x.

Induktionsanfang ($|x| = 0$): klar, da $\hat{\delta}'(Q_0, \varepsilon) = Q_0$ ist.

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

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

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

Nun ist leicht zu sehen, dass der DFA

$$M = (\mathcal{P}(Z), \Sigma, \delta', Q_0, E')$$

mit

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

und

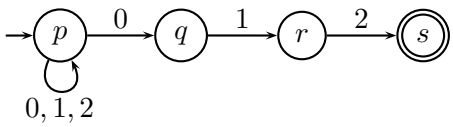
$$E' = \{Q \subseteq Z \mid Q \cap E \neq \emptyset\}$$

äquivalent zu N ist, da für alle Wörter $x \in \Sigma^*$ gilt:

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

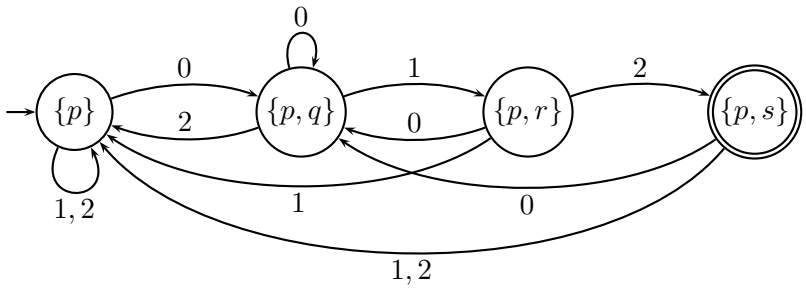
□

Beispiel 1.14 Für den NFA $N = (Z, \Sigma, \delta, Q_0, E)$ aus Beispiel 1.11



ergibt die Konstruktion des vorigen Satzes den folgenden DFA M (nach Entfernung 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 so genannten **Potenzmengenautomaten** M benötigt werden (siehe Übungen).

1.3 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 einelementigen Sprachen bilden lassen.

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

Definition 1.15 *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 $\emptyset$, ϵ und a ($a \in \Sigma$) sind reguläre Ausdrücke, die*

- *die leere Sprache $L(\emptyset) = \emptyset$,*
- *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.

Beispiel 1.16 Über $\Sigma = \{0, 1\}$ sind ϵ^*, $\emptyset^*$, $(0|1)^*00$ und $(\epsilon 0|\emptyset 1^*)$ reguläre Ausdrücke, die folgende Sprachen beschreiben:

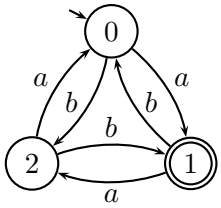
γ	ϵ^*	$\emptyset^*$	$(0\|1)^*00$	$(\epsilon 0\|\emptyset 1^*)$
$L(\gamma)$	$\{\varepsilon\}^* = \{\varepsilon\}$	$\emptyset^* = \{\varepsilon\}$	$\{x00 \mid x \in \Sigma^*\}$	$\{0\}$

◁

Bemerkung 1.17

- *Um Klammern zu sparen, definieren wir folgende **Präzedenzordnung**:*
  Der Sternoperator * bindet stärker als der Produktoperater und dieser wiederum stärker als der Vereinigungsoperator $|$.
- *Da der reguläre Ausdruck $\gamma\gamma^*$ die Sprache $L(\gamma)^+$ beschreibt, verwenden wir γ^+ als Abkürzung für den Ausdruck $\gamma\gamma^*$.*

Beispiel 1.18 Betrachte den 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 $L_1 = \{x \in \{a, b\}^* \mid \#_a(x) - \#_b(x) \equiv_3 0\}$. Da sich L_1 durch den regulären Ausdruck

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

beschreiben lässt, erhalten wir für $L(M)$ den regulären Ausdruck $\gamma(ba)^*(a|bb)$. ◁

Theorem 1.19 $\mathsf{REG} = \{L(\gamma) \mid \gamma$ *ist ein regulärer Ausdruck*$\}$.

Beweis Die Inklusion von rechts nach links ist klar, da die Basisausdrücke $\emptyset$, ϵ und a, $a \in \Sigma^*$, nur reguläre Sprachen beschreiben und die Sprachklasse REG unter Produkt, Vereinigung und Sternhülle abgeschlossen ist (siehe Beobachtungen 1.9 und 1.12).

Für die Gegenrichtung konstruieren wir zu einem DFA M einen regulären Ausdruck γ mit $L(\gamma) = L(M)$. Sei also $M = (Z, \Sigma, \delta, q_0, E)$ ein DFA, wobei wir annehmen können, dass $Z = \{1, \ldots, m\}$ und $q_0 = 1$ ist. Dann lässt sich $L(M)$ als Vereinigung

$$L(M) = \bigcup_{q \in E} L_{1,q}$$

von Sprachen der Form

$$L_{p,q} = \{x \in \Sigma^* \mid \hat{\delta}(p, x) = q\}$$

darstellen. Folglich reicht es zu zeigen, dass die Sprachen $L_{p,q}$ durch reguläre Ausdrücke beschreibbar sind. Hierzu betrachten wir die Sprachen

$$L^r_{p,q} = \{x \in \Sigma^* \mid \hat{\delta}(p, x) = q \text{ und für } i = 1, \ldots, n-1 \text{ gilt } \hat{\delta}(p, x_1 \cdots x_i) \leq r\}.$$

Wegen $L_{p,q} = L^m_{p,q}$ reicht es, reguläre Ausdrücke $\gamma^r_{p,q}$ für die Sprachen $L^r_{p,q}$ anzugeben. Im Fall $r = 0$ enthält

$$L^0_{p,q} = \begin{cases} \{a \in \Sigma \mid \delta(p, a) = q\} \cup \{\varepsilon\}, & p = q, \\ \{a \in \Sigma \mid \delta(p, a) = q\}, & \text{sonst} \end{cases}$$

nur Buchstaben (und eventuell das leere Wort) und ist somit leicht durch einen regulären Ausdruck $\gamma^0_{p,q}$ beschreibbar. Wegen

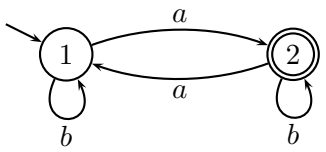
$$L^{r+1}_{p,q} \quad = \quad L^r_{p,q} \cup L^r_{p,r+1}(L^r_{r+1,r+1})^* L^r_{r+1,q}$$

lassen sich aus den regulären Ausdrücken $\gamma^r_{p,q}$ für die Sprachen $L^r_{p,q}$ leicht reguläre Ausdrücke für die Sprachen $L^{r+1}_{p,q}$ gewinnen:

$$\gamma^{r+1}_{p,q} = \gamma^r_{p,q} | \gamma^r_{p,r+1}(\gamma^r_{r+1,r+1})^* \gamma^r_{r+1,q}.$$

□

Beispiel 1.20 Betrachte den DFA



Da M insgesamt $m = 2$ Zustände und nur den Endzustand 2 besitzt, ist

$$L(M) = \bigcup_{q \in E} L_{1,q} = L_{1,2} = L^2_{1,2} = L(\gamma^2_{1,2}).$$

Um $\gamma^2_{1,2}$ zu berechnen, benutzen wir die Rekursionsformel

$$\gamma^{r+1}_{p,q} = \gamma^r_{p,q} | \gamma^r_{p,r+1} (\gamma^r_{r+1,r+1})^* \gamma^r_{r+1,q}$$

und erhalten

$$\begin{aligned}
\gamma^2_{1,2} &= \gamma^1_{1,2} | \gamma^1_{1,2} (\gamma^1_{2,2})^* \gamma^1_{2,2}, \\
\gamma^1_{1,2} &= \gamma^0_{1,2} | \gamma^0_{1,1} (\gamma^0_{1,1})^* \gamma^0_{1,2}, \\
\gamma^1_{2,2} &= \gamma^0_{2,2} | \gamma^0_{2,1} (\gamma^0_{1,1})^* \gamma^0_{1,2}.
\end{aligned}$$

Um einen regulären Ausdruck für $L(M) = L(b^*a(b|ab^*a)^*)$ zu erhalten, genügt es also, die regulären Ausdrücke $\gamma^0_{1,1}$, $\gamma^0_{1,2}$, $\gamma^0_{2,1}$, $\gamma^0_{2,2}$, $\gamma^1_{1,2}$, $\gamma^1_{2,2}$ und $\gamma^2_{1,2}$ zu berechnen:

r	p,q			
	$1,1$	$1,2$	$2,1$	$2,2$
0	$\epsilon \mid b$	a	a	$\epsilon \mid b$
1	-	$\underbrace{a \mid (\epsilon \mid b)(\epsilon \mid b)^* a}_{b^*a}$	-	$\underbrace{(\epsilon \mid b) \mid a(\epsilon \mid b)^* a}_{\epsilon \mid b \mid ab^*a}$
2	-	$\underbrace{b^*a \mid b^*a(\epsilon \mid b \mid ab^*a)^*(\epsilon \mid b \mid ab^*a)}_{b^*a(b \mid ab^*a)^*}$	-	-

◁

1.4 Relationalstrukturen

Sei A eine nichtleere Menge, R_i eine k_i-stellige Relation auf A, d.h. $R_i \subseteq A^{k_i}$ für $i = 1, \ldots, n$. Dann heißt $(A; R_1, \ldots, R_n)$ ***Relationalstruktur***. Die Menge A heißt ***Grundmenge***, ***Trägermenge*** oder ***Individuenbereich*** der Relationalstruktur.

Bemerkung 1.21

- *Wir werden hier hauptsächlich den Fall $n = 1$, $k_1 = 2$, also (A, R) mit $R \subseteq A \times A$ betrachten. Man nennt dann R eine **(binäre) Relation** auf A.*
- *Oft wird für $(a, b) \in R$ auch die **Infix-Schreibweise** aRb benutzt.*

Beispiel 1.22

- (F, M) mit $F := \{f \mid f$ ist Fluss in Europa$\}$ und
  $M := \{(f, g) \in F \times F \mid f$ mündet in $g\}$.

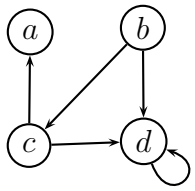
- (U, B) mit $U := \{x \mid x \text{ ist Berliner }\}$ und
  $B := \{(x, y) \in U \times U \mid x \text{ ist Bruder von } y\}$.
- $(\mathsf{P}(M), \subseteq)$, wobei $\mathsf{P}(M)$ die Potenzmenge einer beliebigen Menge M und $\subseteq$ die Inklusionsbeziehung auf den Teilmengen von M ist.
- (A, Id_A), wobei $Id_A = \{(x, x) \mid x \in A\}$ die ***Identität auf*** A ist.
- $(\mathbb{R}, \leq)$.
- $(\mathbb{Z}, |)$, wobei $|$ die "teilt"-Relation bezeichnet.
- $(\mathcal{F}ml, \Rightarrow)$ mit $\mathcal{F}ml := \{F \mid F \text{ ist aussagenlogische Formel}\}$ und
  $\Rightarrow = \{(F, G) \in \mathcal{F}ml \times \mathcal{F}ml \mid G \text{ ist aussagenlogische Folgerung von } F\}$.

◁

Graphische Darstellung von Relationen

Eine Relation R auf einer endlichen Menge A kann durch einen ***gerichteten Graphen*** $G = (V, E)$ mit ***Knotenmenge*** $V = A$ und ***Kantenmenge*** $E = R$ veranschaulicht werden. Hierzu stellen wir jedes Element $x \in A$ als einen Knoten dar und verbinden jedes Knotenpaar $(x, y) \in R$ durch eine gerichtete Kante (Pfeil). Zwei durch eine Kante verbundene Knoten heißen ***benachbart*** oder ***adjazent***.

Beispiel 1.23 Für die Relation (A, R) mit $A = \{a, b, c, d\}$ und $R = \{(b, c), (b, d), (c, a), (c, d), (d, d)\}$ erhalten wir folgende graphische Darstellung.



◁

Der ***Ausgangsgrad*** eines Knotens $x \in V$ ist $d_{out}(x) = \|R(x)\|$, wobei $R(x) = \{y \in V \mid xRy\}$ der ***Nachbereich*** von x ist. Entsprechend ist $d_{in}(x) = \|\{y \in V \mid yRx\}\|$ der ***Eingangsgrad*** von x. Falls R symmetrisch ist, können die Pfeilspitzen auch weggelassen werden. In diesem Fall ist $d(x) = d_{in}(x) = d_{out}(x)$ der ***Grad*** von x. Ist R zudem irreflexiv, so erhalten wir einen (schleifenfreien) ***Graphen***.

Darstellung durch eine Adjazenzmatrix

Eine Relation R auf einer endlichen (geordneten) Menge $A = \{a_1, \ldots, a_n\}$ lässt sich durch eine boolesche $n \times n$-Matrix $M_R = (m_{ij})$ mit

$$m_{ij} := \begin{cases} 1, & a_i R a_j, \\ 0, & \text{sonst} \end{cases}$$

darstellen. Beispielsweise hat die Relation

$$R = \{(b,c),(b,d),(c,a),(c,d),(d,d)\}$$

auf der Menge $A = \{a,b,c,d\}$ die Matrixdarstellung

$$M_R = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Darstellung durch eine Adjazenzliste

Eine weitere Möglichkeit besteht darin, eine endliche Relation R in Form einer Tabelle darzustellen, die jedem Element $x \in A$ seinen Nachbereich $R(x)$ in Form einer Liste zuordnet:

x	$R(x)$
a	-
b	c,d
c	a,d
d	d

1.4.1 Eigenschaften von Relationen

Sei R eine Relation auf A. Dann heißt R

reflexiv, falls $\forall x \in A : xRx$ (d.h. $Id_A \subseteq R$),
irreflexiv, falls $\forall x \in A : \neg xRx$ (d.h. $Id_A \subseteq \overline{R}$),
symmetrisch, falls $\forall x,y \in A : xRy \Rightarrow yRx$ (d.h. $R \subseteq R^T$),
asymmetrisch, falls $\forall x,y \in A : xRy \Rightarrow \neg yRx$ (d.h. $R \subseteq \overline{R^T}$),
antisymmetrisch, falls $\forall x,y \in A : xRy \wedge yRx \Rightarrow x = y$ (d.h. $R \cap R^T \subseteq Id$),
konnex, falls $\forall x,y \in A : xRy \vee yRx$ (d.h. $A \times A \subseteq R \cup R^T$),
semikonnex, falls $\forall x,y \in A : x \neq y \Rightarrow xRy \vee yRx$ (d.h. $\overline{Id} \subseteq R \cup R^T$),
transitiv, falls $\forall x,y,z \in A : xRy \wedge yRz \Rightarrow xRz$ (d.h. $R^2 \subseteq R$)

gilt.

Beispiel 1.24

- Die Relation ”ist Schwester von” ist zwar in einer reinen Damengesellschaft symmetrisch, i.a. jedoch weder symmetrisch noch asymmetrisch noch antisymmetrisch.
- $(\mathbb{R}, <)$ ist irreflexiv, asymmetrisch, transitiv und semikonnex.

- $(\mathbb{R}, \leq)$ und $(\mathsf{P}(M), \subseteq)$ sind reflexiv, antisymmetrisch und transitiv.
- $(\mathbb{R}, \leq)$ ist auch konnex und $(\mathsf{P}(M), \subseteq)$ ist im Fall $\|M\| \leq 1$ zwar auch konnex, aber im Fall $\|M\| \geq 2$ weder semikonnex noch konnex.

$\triangleleft$

Operationen auf Relationen

Da Relationen Mengen sind, sind auf ihnen die mengentheoretischen Operationen ***Durchschnitt***, ***Vereinigung***, ***Komplement*** und ***Differenz*** definiert. Seien R und S Relationen auf A, dann ist

$$\begin{array}{rcl} R \cap S & = & \{(x,y) \in A \times A \mid xRy \wedge xSy\}, \\ R \cup S & = & \{(x,y) \in A \times A \mid xRy \vee xSy\}, \\ R - S & = & \{(x,y) \in A \times A \mid xRy \wedge \neg xSy\}, \\ \overline{R} & = & (A \times A) - R. \end{array}$$

Sei allgemeiner $\mathcal{M} \subseteq \mathcal{P}(A \times A)$ eine beliebige Menge von Relationen auf A. Dann sind der **Schnitt über** $\mathcal{M}$ und die **Vereinigung über** $\mathcal{M}$ folgende Relationen:

$$\begin{array}{rcl} \bigcap \mathcal{M} & = & \{(x,y) \mid \forall R \in \mathcal{M} : xRy\} \\ \bigcup \mathcal{M} & = & \{(x,y) \mid \exists R \in \mathcal{M} : xRy\} \end{array}$$

Weiterhin ist die **Inklusionsrelation** $R \subseteq S$ auf Relationen definiert:

$$R \subseteq S \Leftrightarrow \forall x, y : xRy \rightarrow xSy.$$

Die ***transponierte (konverse) Relation*** zu R ist

$$R^T := \{(y,x) \mid xRy\}.$$

R^T wird oft auch mit R^{-1} bezeichnet. Zum Beispiel ist $(\mathbb{R}, \leq^T) = (\mathbb{R}, \geq)$.
Eine wichtige zweistellige Operation auf der Menge $\mathsf{P}(A \times A)$ aller Relationen auf A ist das Relationenprodukt (auch Komposition genannt).
Das ***Produkt*** zweier Relationen R und S auf A ist

$$R \circ S := \{(x,z) \mid \exists y : xRy \wedge ySz\}.$$

Übliche Bezeichnungen für das Relationenprodukt sind auch $R;S$ und $R \cdot S$ oder einfach RS. Für $\underbrace{R \circ \cdots \circ R}_{\text{n-mal}}$ wird auch R^n geschrieben. Dabei ist $R^0 = Id$.
Vorsicht: Das n-fache Relationenprodukt von R sollte nicht mit dem n-fachen kartesischen Produkt der *Menge* R verwechselt werden. Wir vereinbaren, dass R^n das n-fache Relationenprodukt bezeichnen soll, falls R eine Relation ist.

Beispiel 1.25 Ist B die Relation "ist Bruder von", V "ist Vater von", M "ist Mutter von" und $E = V \cup M$ "ist Elternteil von", so ist $B \circ E$ die Relation "ist Onkel von".◁

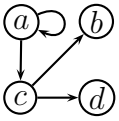
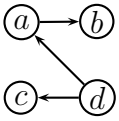
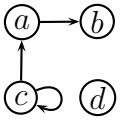
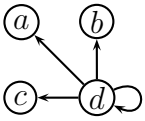
Sind $M_R = (r_{ij})$ und $M_S = (s_{ij})$ boolesche $n \times n$-Matrizen für R und S, so erhalten wir für $T = R \circ S$ die Matrix $M_T = (t_{ij})$ mit

$$t_{ij} := \bigvee_{k=1,\ldots,n} (r_{ik} \wedge s_{kj})$$

Der Nachbereich $T(x)$ von x bzgl. der Relation $T = R \circ S$ berechnet sich zu

$$T(x) = \bigcup\{S(y) \mid y \in R(x)\} = \bigcup_{y \in R(x)} S(y).$$

Beispiel 1.26 Betrachte die Relationen $R = \{(a,a),(a,c),(c,b),(c,d)\}$ und $S = \{(a,b),(d,a),(d,c)\}$ auf der Menge $A = \{a,b,c,d\}$.

Relation	R	S	$R \circ S$	$S \circ R$
Graph				
Adjazenz-matrix	1 0 1 0 0 0 0 0 0 1 0 1 0 0 0 0	0 1 0 0 0 0 0 0 0 0 0 0 1 0 1 0	0 1 0 0 0 0 0 0 1 0 1 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1
Adjazenz-liste	a : a, c b : - c : b, d d : -	a : b b : - c : - d : a, c	a : b b : - c : a, c d : -	a : - b : - c : - d : a, b, c, d

◁

Beobachtung: Das Beispiel zeigt, dass das Relationenprodukt nicht kommutativ ist, d.h. i.a. gilt nicht $R \circ S = S \circ R$.

Als nächstes zeigen wir, dass die Menge $\mathcal{R} = \mathcal{P}(A \times A)$ aller binären Relationen auf A mit dem Relationenprodukt $\circ$ als binärer Operation und der Relation Id_A als neutralem Element eine Halbgruppe (oder ***Monoid***) bildet.

Theorem 1.27 *Seien Q, R, S Relationen auf A. Dann gilt*

(i) $(Q \circ R) \circ S = Q \circ (R \circ S)$, *d.h. $\circ$ ist assoziativ,*

(ii) $Id \circ R = R \circ Id = R$, *d.h. Id ist neutrales Element.*

Beweis

(i) Es gilt:

$$\begin{aligned} x\ (Q \circ R) \circ S\ y &\Leftrightarrow \exists u \in A : x\ (Q \circ R)\ u\ \wedge\ u\ S\ y \\ &\Leftrightarrow \exists u \in A : (\exists v \in A : x\ Q\ v\ R\ u)\ \wedge\ u\ S\ y \\ &\Leftrightarrow \exists u, v \in A : x\ Q\ v\ R\ u\ S\ y \\ &\Leftrightarrow \exists v \in A : x\ Q\ v\ \wedge\ (\exists u \in A : v\ R\ u\ \wedge\ u\ S\ y) \\ &\Leftrightarrow \exists v \in A : x\ Q\ v\ (R \circ S)\ y \\ &\Leftrightarrow x\ Q \circ (R \circ S)\ y \end{aligned}$$

(ii) Wegen $x\ Id \circ R\ y \Leftrightarrow \exists z : x = z\ \wedge\ z\ R\ y \Leftrightarrow x\ R\ y$ folgt $Id \circ R = R$. Die Gleichheit $R \circ Id = R$ folgt analog. □

Manchmal steht man vor der Aufgabe, eine gegebene Relation R durch eine möglichst kleine Modifikation in eine Relation R' mit vorgegebenen Eigenschaften zu überführen. Will man dabei alle in R enthaltenen Paare beibehalten, dann sollte R' aus R durch Hinzufügen möglichst weniger Paare hervorgehen.

Es lässt sich leicht nachprüfen, dass der Schnitt über eine Menge reflexiver (bzw. transitiver oder symmetrischer) Relationen wieder reflexiv (bzw. transitiv oder symmetrisch) ist. Folglich existiert zu jeder Relation R auf einer Menge A eine kleinste reflexive (bzw. transitive oder symmetrische) Relation R', die R enthält.

Definition 1.28 *Sei R eine Relation.*

- *Die **reflexive Hülle** von R ist*

$$h_{\mathit{refl}}(R) = \bigcap \{S \subseteq A \times A \mid S \text{ ist reflexiv und } R \subseteq S\}.$$

- *Die **symmetrische Hülle** von R ist*

$$h_{\mathit{sym}}(R) = \bigcap \{S \subseteq A \times A \mid S \text{ ist symmetrisch und } R \subseteq S\}.$$

- *Die **transitive Hülle** von R ist*

$$R^{+} = \bigcap \{S \subseteq A \times A \mid S \text{ ist transitiv und } R \subseteq S\}.$$

- *Die **reflexiv-transitive Hülle** von R ist*

$$R^{*} = \bigcap \{S \subseteq A \times A \mid S \text{ ist reflexiv, transitiv und } R \subseteq S\}.$$

Theorem 1.29 *Sei R eine Relation auf A.*

(i) $h_{\mathit{refl}}(R) = R \cup Id_A$,

(ii) $h_{\mathit{sym}}(R) = R \cup R^{T}$,

(iii) $R^+ = \bigcup_{n \geq 1} R^n$,

(iv) $R^* = \bigcup_{n \geq 0} R^n$.

Beweis Siehe Übungen. □

Anschaulich besagt der vorhergehende Satz, dass ein Paar (a, b) genau dann in der reflexiv-transitiven Hülle R^* von R ist, wenn es ein $n \geq 0$ gibt mit aR^nb, d.h. es gibt Elemente $x_0, \ldots, x_n \in A$ mit $x_0 = a$, $x_n = b$ und

$$x_0 R x_1 R x_2 \cdots x_{n-1} R x_n.$$

In der Graphentheorie nennt man $x_0 \cdots x_n$ einen ***Weg der Länge*** n ***von*** a ***nach*** b.

1.4.2 Äquivalenz- und Ordnungsrelationen

Die nachfolgende Tabelle gibt einen Überblick über die definierenden Eigenschaften der wichtigsten Relationalstrukturen.

	refl.	sym.	trans.	asym.	antisym.	konnex	semikon.
Äquivalenzrel.	✓	✓	✓				
(Halb-)Ordnung	✓		✓		✓		
Striktordnung			✓	✓			
lineare Ord.			✓		✓	✓	
lin. Striktord.			✓	✓			✓
schwache Ord.			✓			✓	
Quasiordnung	✓		✓				

In der Tabelle sind nur die definierenden Eigenschaften durch ein "✓" gekennzeichnet. Das schließt nicht aus, dass gleichzeitig auch noch weitere Eigenschaften vorliegen können.

Wir betrachten zunächst eine Reihe von Beispielen für ***Äquivalenzrelationen***, die durch die drei Eigenschaften reflexiv, symmetrisch und transitiv definiert sind.

Beispiel 1.30

- Auf der Menge aller Geraden im $\mathbb{R}^2$ die Parallelität.
- Auf der Menge aller Menschen "im gleichen Jahr geboren wie".
- Auf $\mathbb{Z}$ die Relation "gleicher Rest bei Division durch m".
- Auf der Menge der aussagenlogischen Formeln die semantische Äquivalenz. ◁

Ist E eine Äquivalenzrelation, so nennt man den zu x gehörigen Nachbereich $E(x)$ die ***von*** x ***repräsentierte Äquivalenzklasse*** und bezeichnet sie mit $[x]_E$ oder einfach mit $[x]$. Die durch E auf A induzierte Partition $\{[x] \mid x \in A\}$ wird ***Quotienten- oder Faktormenge*** genannt und mit A/E bezeichnet. Die Anzahl der Äquivalenzklassen von E wird auch als der ***Index*** von E bezeichnet.

Definition 1.31 *Eine Familie $\{M_i \mid i \in I\}$ von nichtleeren Teilmengen $M_i \subseteq A$ heißt* ***Partition*** *der Menge A, falls gilt:*

a) die Mengen M_i ***überdecken*** *A, d.h. $A = \bigcup_{i\in I} M_i$ und*

b) die Mengen M_i sind ***paarweise disjunkt****, d.h. für je zwei verschiedene Mengen $M_i \neq M_j$ gilt $M_i \cap M_j = \emptyset$.*

Wie der nächste Satz zeigt, beschreiben Äquivalenzrelationen auf A und Partitionen von A den selben Sachverhalt.

Theorem 1.32 *Sei E eine Relation auf A. Dann sind folgende Aussagen äquivalent.*

(i) E ist eine Äquivalenzrelation auf A.

(ii) Für alle $x, y \in A$ gilt

$$xEy \Leftrightarrow E(x) = E(y) \qquad (*)$$

(iii) E ist reflexiv und $\{E(x) \mid x \in A\}$ ist eine Partition von A.

Beweis

$(i) \Rightarrow (ii)$ Sei E eine Äquivalenzrelation auf A. Da E transitiv ist, impliziert xEy die Inklusion $E(y) \subseteq E(x)$:

$$z \in E(y) \Rightarrow yEz \Rightarrow xEz \Rightarrow z \in E(x).$$

Da E symmetrisch ist, folgt aus xEy aber auch $E(x) \subseteq E(y)$.

Umgekehrt folgt aus $E(x) = E(y)$ wegen der Reflexivität von E, dass $x \in E(x) = E(y)$ enthalten ist, und somit xEy. Dies zeigt, dass E die Äquivalenz $(*)$ erfüllt.

$(ii) \Rightarrow (iii)$ Falls E die Bedingung $(*)$ erfüllt, so folgt sofort xEx (wegen $E(x) = E(x)$) und folglich überdecken die Nachbereiche $E(x)$ (wegen $x \in E(x)$) die Menge A.

Ist $E(x) \cap E(y) \neq \emptyset$ und z ein Element in $E(x) \cap E(y)$, so gilt xEz und yEz und daher folgt $E(x) = E(z) = E(y)$. Da also je zwei Nachbereiche $E(x)$ und $E(y)$ entweder gleich oder disjunkt sind, bildet $\{E(x) \mid x \in A\}$ sogar eine Partition von A.

$(iii) \Rightarrow (i)$ Wird schließlich A von den Mengen $E(x)$ partitioniert, wobei $x \in E(x)$ für alle $x \in A$ gilt, so folgt

$$xEy \Leftrightarrow y \in E(x) \cap E(y) \Leftrightarrow E(x) = E(y).$$

Daher übertragen sich die Eigenschaften Reflexivität, Symmetrie und Transitivität unmittelbar von der Gleichheitsrelation auf E.

□

Beispiel 1.33 Für die weiter oben betrachteten Äquivalenzrelationen erhalten wir folgende Klasseneinteilungen.

- Für die Parallelität auf der Menge aller Geraden im $\mathbb{R}^2$: alle Geraden mit derselben Richtung (oder Steigung) bilden jeweils eine Äquivalenzklasse.
- Für die Relation "im gleichen Jahr geboren wie" auf der Menge aller Menschen: jeder Jahrgang bildet eine Äquivalenzklasse.
- Für die Relation "gleicher Rest bei Division durch m" auf $\mathbb{Z}$: jede der m Restklassen $[0], [1], \ldots, [m-1]$ mit

  $$[r] = \{a \in \mathbb{Z} \mid a \bmod m = r\}$$

  bildet eine Äquivalenzklasse. ◁

Die kleinste Äquivalenzrelation auf A ist die **Identität** Id_A, die größte die **Allrelation** $A \times A$. Die Äquivalenzklassen der Identität enthalten jeweils nur ein Element, d.h. $A/Id_A = \{\{x\} \mid x \in A\}$, und die Allrelation erzeugt nur eine Äquivalenzklasse, nämlich $A/(A \times A) = \{A\}$.

Für zwei Äquivalenzrelationen $E \subseteq E'$ sind auch die Äquivalenzklassen $[x]_E$ von E in den Klassen $[x]_{E'}$ von E' enthalten. Folglich ist jede Äquivalenzklasse von E' die Vereinigung von (evtl. mehreren) Äquivalenzklassen von E_1. Im Fall $E \subseteq E'$ sagt man auch, E_1 bewirkt eine **feinere** Partitionierung als E_2. Demnach ist die Identität die **feinste** und die Allrelation die **gröbste** Äquivalenzrelation.

Da der Schnitt über eine Menge von Äquivalenzrelationen wieder eine Äquivalenzrelation ist, können wir für eine beliebige Relation R auf einer Menge A die kleinste R umfassende Äquivalenzrelation definieren:

$$h_{\text{äq}}(R) := \bigcap \{E \mid E \text{ ist eine Äquivalenzrelation auf } A \text{ mit } R \subseteq E\}$$

In der Sprache der Graphentheorie werden die durch $h_{\text{äq}}(R)$ generierten Äquivalenzklassen auch die **schwachen Zusammenhangskomponenten** des gerichteten Graphen (A, R) genannt (siehe Übungen). Als nächstes betrachten wir Ordnungen.

Definition 1.34 (A, R) *heißt* ***Ordnung*** *(auch* ***Halbordnung*** *oder* ***partielle Ordnung****), wenn* R *eine reflexive, antisymmetrische und transitive Relation auf* A *ist.*

Beispiel 1.35

- $(\mathbb{Z}, \leq)$ und $(\mathbb{N}, |)$ sind Ordnungen. Erstere ist linear, letztere nicht.
- Für jede Menge M ist die relationale Struktur $(\mathcal{P}(M); \subseteq)$ eine Ordnung. Diese ist nur im Fall $\|M\| \leq 1$ linear.
- Auf der Menge $\mathcal{A}(M)$ aller Äquivalenzrelationen auf M die Relation "feiner als". Dabei ist, wie wir gesehen haben, E_1 eine Verfeinerung von E_2, falls E_1 in E_2 enthalten ist. In diesem Fall bewirkt E_1 nämlich eine feinere Klasseneinteilung auf M als E_2, da jede Äquivalenzklasse von E_1 in einer Äquivalenzklasse von E_2 enthalten ist.
- Ist R eine Ordnung auf A und $B \subseteq A$, so heißt die Ordnung $R_B = R \cap (B \times B)$ die ***Einschränkung*** (oder ***Restriktion***) von R auf B. Beispielsweise ist $(\mathcal{A}(M); \subseteq)$ die Einschränkung von $(\mathcal{P}(M \times M); \subseteq)$ auf $\mathcal{A}(M)$. $\triangleleft$

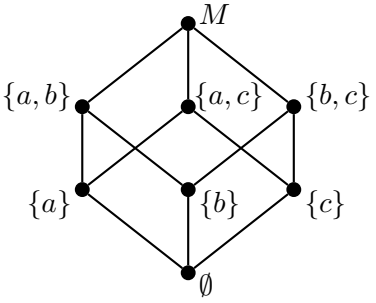
Ordnungen lassen sich sehr anschaulich durch Hasse-Diagramme darstellen. Sei $\leq$ eine Ordnung auf A und sei $<$ die Relation $\leq \cap \overline{Id}_A$. Um die Ordnung $\leq$ in einem ***Hasse-Diagramm*** darzustellen, wird nur der Graph der ***Nachbarrelation***

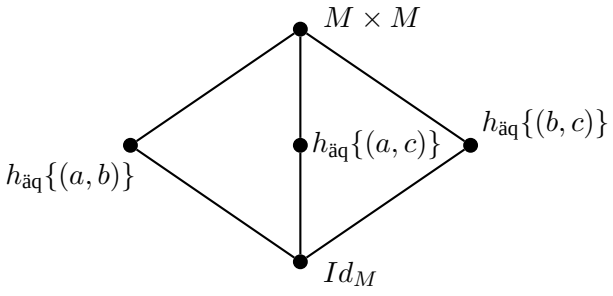
$$\lhd \; = \; < \setminus <^2, \text{ d.h. } x \lhd y \;\Leftrightarrow\; x < y \wedge \neg \exists z : x < z < y$$

gezeichnet. Für $x \lhd y$ sagt man auch, y ist ***oberer Nachbar*** von x. Weiterhin wird im Fall $x \lhd y$ der Knoten y oberhalb vom Knoten x gezeichnet, so dass auf Pfeilspitzen verzichtet werden kann.

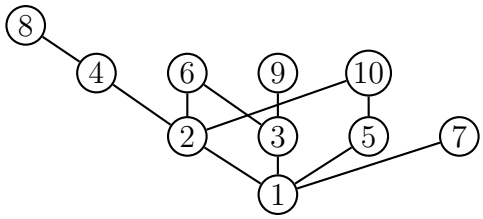
Beispiel 1.36

1. Die Inklusionsrelation auf der Potenzmenge $\mathcal{P}(M)$ von $M = \{a, b, c\}$ und die Verfeinerungsrelation auf der Menge $\mathcal{A}(M)$ aller Äquivalenzrelationen auf $M = \{a, b, c\}$ lassen sich durch folgende Hasse-Diagramme darstellen.





2. Die "teilt"-Relation auf $\{1, 2, \ldots, 10\}$ ist durch folgendes Hasse-Diagramme darstellbar.



◁

Definition 1.37 *Sei $\leq$ eine Ordnung auf A und sei $h \in H$ Element einer Teilmenge $H \subseteq A$.*

- *h heißt **kleinstes Element** oder **Minimum** von H ($h = \min H$), falls gilt:*

$$\forall h' \in H : h \leq h'.$$

- *h heißt **größtes Element** oder **Maximum** von H ($h = \max H$), falls gilt:*

$$\forall h' \in H : h' \leq h.$$

- *a heißt **minimal** in H, falls es in H kein kleineres Element gibt:*

$$\forall h' \in H : h' \leq h \Rightarrow h' = h.$$

- *a heißt **maximal** in H, falls es in H kein größeres Element gibt:*

$$\forall h' \in H : h \leq h' \Rightarrow h = h'.$$

Bemerkung 1.38 *Wegen der Antisymmetrie kann es in H höchstens ein kleinstes und höchstens ein größtes Element geben.*

Definition 1.39 *Sei $\leq$ eine Ordnung auf A und sei $H \subseteq A$.*

- *Jedes Element $u \in A$ mit $u \leq h$ für alle $h \in H$ heißt **untere** und jedes $o \in A$ mit $h \leq o$ für alle $h \in H$ heißt **obere Schranke** von H.*
- *H heißt **nach oben beschränkt**, wenn H eine obere Schranke hat, und **nach unten beschränkt**, wenn H eine untere Schranke hat.*
- *H heißt **beschränkt**, wenn H nach oben und nach unten beschränkt ist.*
- *Besitzt H eine größte untere Schranke i, d.h. besitzt die Menge U aller unteren Schranken von H ein größtes Element i, so heißt i das **Infimum** von H ($i = \inf H$):*

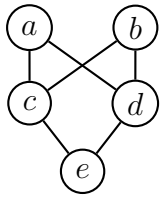
$$(\forall h \in H : h \geq i) \wedge [\forall u \in A : (\forall h \in H : h \geq u) \Rightarrow u \leq i].$$

- *Besitzt H eine kleinste obere Schranke s, d.h. besitzt die Menge O aller oberen Schranken von H ein kleinstes Element s, so heißt s das* ***Supremum*** *von H ($s = \sup H$):*

$$(\forall h \in H : h \leq s) \wedge [\forall o \in A : (\forall h \in H : h \leq o) \Rightarrow s \leq o]$$

Bemerkung 1.40 *H kann nicht mehr als ein Supremum und ein Infimum haben.*

Beispiel 1.41 Betrachte nebenstehende Ordnung auf der Menge $A = \{a, b, c, d, e\}$. Die folgende Tabelle zeigt für verschiedene Teilmengen $H \subseteq A$ alle minimalen und maximalen Elemente in H, alle unteren und oberen Schranken, sowie Minimum, Maximum, Infimum und Supremum von H (falls existent).



H	minimal	maximal	Minimum	Maximum	unt. Schr.	ob. Schr.	Inf.	Sup.
$\{a, b\}$	a, b	a, b	-	-	c, d, e	-	-	-
$\{c, d\}$	c, d	c, d	-	-	e	a, b	e	-
$\{a, b, c\}$	c	a, b	c	-	c, e	-	c	-
$\{a, b, c, e\}$	e	a, b	e	-	e	-	e	-
$\{a, c, d, e\}$	e	a	e	a	e	a	e	a

◁

Bemerkung 1.42

- *Auch in linearen Ordnungen muss nicht jede beschränkte Teilmenge ein Supremum oder Infimum besitzen.*
- *So hat in der linear geordneten Menge $(\mathbb{Q}, \leq)$ die Teilmenge*

$$H \;=\; \{x \in \mathbb{Q} \mid x^2 \leq 2\}$$

  weder ein Supremum noch ein Infimum.

- *Dagegen hat in einer linearen Ordnung jede* endliche *Teilmenge ein kleinstes und ein größtes Element und somit erst recht ein Supremum und ein Infimum.*

1.4.3 Abbildungen

Definition 1.43 *Sei R eine binäre Relation auf einer Menge M.*

- *R heißt* ***rechtseindeutig****, falls gilt:*

$$\forall x, y, z \in M : xRy \wedge xRz \Rightarrow y = z.$$

- *R heißt **linkseindeutig**, falls gilt:*

$$\forall x, y, z \in M : xRz \wedge yRz \Rightarrow x = y.$$

- *Der **Nachbereich** $N(R)$ und der **Vorbereich** $V(R)$ von R sind*

$$N(R) = \bigcup_{x \in M} R(x) \text{ und } V(R) = \bigcup_{x \in M} R^T(x).$$

- *Eine rechtseindeutige Relation R mit $V(R) = A$ und $N(R) \subseteq B$ heißt **Abbildung** oder **Funktion von A nach B** (kurz $R : A \to B$).*

Bemerkung 1.44

- *Wie üblich werden wir Abbildungen meist mit kleinen Buchstaben $f, g, h, ...$ bezeichnen und für $(x, y) \in f$ nicht xfy sondern $f(x) = y$ oder $f : x \mapsto y$ schreiben.*
- *Ist $f : A \to B$ eine Abbildung, so wird der Vorbereich $V(f) = A$ der **Definitionsbereich** und die Menge B der **Wertebereich** oder **Wertevorrat** von f genannt.*
- *Der Nachbereich $N(f)$ wird als **Bild** von f bezeichnet.*

Definition 1.45

- *Im Fall $N(f) = B$ heißt f **surjektiv**.*
- *Ist f linkseindeutig, so heißt f **injektiv**. In diesem Fall impliziert $f(x) = f(y)$ die Gleichheit $x = y$.*
- *Eine injektive und surjektive Abbildung heißt **bijektiv**.*
- *Für eine injektive Abbildung $f : A \to B$ ist auch f^T eine Abbildung, die mit f^{-1} bezeichnet und die **inverse Abbildung** zu f genannt wird.*

Man beachte, dass der Definitionsbereich $V(f^{-1}) = N(f)$ nur dann gleich B ist, wenn f auch surjektiv, also eine Bijektion ist.

1.4.4 Homo- und Isomorphismen

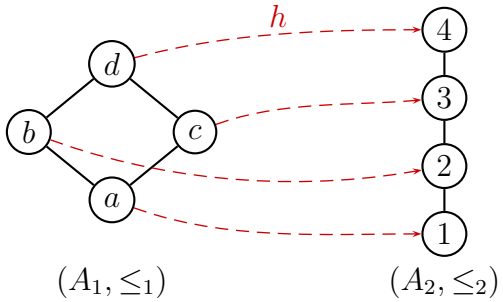
Definition 1.46 *Seien (A_1, R_1) und (A_2, R_2) Relationalstrukturen.*

- *Eine Abbildung $h : A_1 \to A_2$ heißt **Homomorphismus**, falls für alle $a, b \in A_1$ gilt:*

$$aR_1b \Rightarrow h(a)R_2h(b).$$

- *Sind (A_1, R_1) und (A_2, R_2) Ordnungen, so spricht man von **Ordnungshomomorphismen** oder einfach von **monotonen** Abbildungen.*
- *Injektive Ordnungshomomorphismen werden auch **streng monotone** Abbildungen genannt.*

Beispiel 1.47 Folgende Abbildung $h : A_1 \to A_2$ ist ein bijektiver Ordnungshomomorphismus.



Obwohl h ein bijektiver Homomorphismus ist, ist die Umkehrung h^{-1} kein Homomorphismus, da h^{-1} nicht monoton ist. Es gilt nämlich

$$2 \leq_2 3, \text{ aber } h^{-1}(2) = b \not\leq_1 c = h^{-1}(3).$$

◁

Definition 1.48 *Ein bijektiver Homomorphismus $h : A_1 \to A_2$, bei dem auch h^{-1} ein Homomorphismus ist, d.h. es gilt*

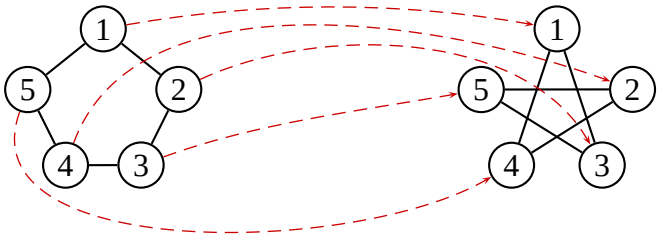
$$\forall a, b \in A_1 : aR_1b \Leftrightarrow h(a)R_2h(b).$$

*heißt **Isomorphismus**. In diesem Fall heißen die Strukturen (A_1, R_1) und (A_2, R_2) **isomorph** (kurz: $(A_1, R_1) \cong (A_2, R_2)$).*

Beispiel 1.49

- Die beiden folgenden Graphen sind isomorph. Zwei Isomorphismen sind beispielsweise h_1 und h_2.

v	1 2 3 4 5
$h_1(v)$	1 3 5 2 4
$h_2(v)$	1 4 2 5 3



- Die Bijektion $h : x \mapsto e^x$ ist ein Ordnungsisomorphismus zwischen $(\mathbb{R}, \leq)$ und $((0, \infty), \leq)$.

- Für $n \in \mathbb{N}$ sei
$$T_n \;=\; \{k \in \mathbb{N} \mid k \text{ teilt } n\}$$
die Menge aller Teiler von n und $P_n \;=\; T_n \cap \text{Prim}$ die Menge aller Primteiler von n. Dann ist für quadratfreies n, d.h. es gibt kein $k \geq 2$, so dass k^2 die Zahl n teilt, die Abbildung
$$h : k \mapsto P_k$$
ein Ordnungsisomorphismus zwischen $(T_n, |)$ und $(\mathcal{P}(P_n), \subseteq)$.

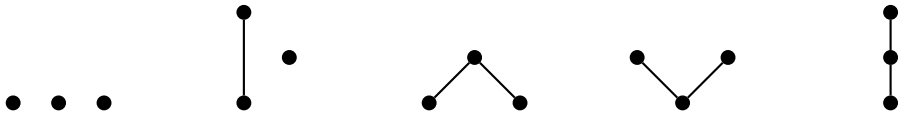
- Während auf der Knotenmenge $V = [3]$ insgesamt $2^3 = 8$ verschiedene Graphen existieren, gibt es auf dieser Menge nur 4 unterschiedliche nichtisomorphe Graphen:

◁

Bemerkung 1.50 *Auf der Knotenmenge $V = [n]$ existieren genau $2^{\binom{n}{2}}$ verschiedene Graphen. Sei $a(n)$ die Anzahl aller nichtisomorphen Graphen auf V. Da maximal $n!$ verschiedene Graphen auf V in einer Isomorphieklasse liegen können, ist $a(n) \geq 2^{\binom{n}{2}}/n!$.*

Tatsächlich ist $a(n)$ ***asymptotisch gleich*** *$g(n) = 2^{\binom{n}{2}}/n!$ (in Zeichen: $a(n) \sim g(n)$), d.h.*
$$\lim_{n\to\infty} a(n)/g(n) = 1.$$
Also gibt es auf $V = [n]$ nicht wesentlich mehr als $g(n)$ nichtisomorphe Graphen.

Beispiel 1.51 Es existieren genau 5 nichtisomorphe Ordnungen mit 3 Elementen:



Anders ausgedrückt: Die Klasse aller dreielementigen Ordnungen zerfällt unter der Äquivalenzrelation $\cong$ in fünf Äquivalenzklassen, die durch obige fünf Hasse-Diagramme repräsentiert werden. ◁

1.5 Minimierung von DFAs

Wie können wir feststellen, ob ein DFA $M = (Z, \Sigma, \delta, q_0, E)$ unnötige Zustände enthält? Zunächst einmal können alle Zustände entfernt werden, die nicht vom Startzustand aus erreichbar sind. Im folgenden gehen wir daher davon aus, dass M keine

unerreichbaren Zustände enthält. Offensichtlich können zwei Zustände q und p zu einem Zustand verschmolzen werden (kurz: $q \sim p$), wenn M von q und von p ausgehend jeweils dieselben Wörter akzeptiert. Bezeichnen wir den DFA $(Z, \Sigma, \delta, q, E)$ mit M_q und $L(M_q)$ mit L_q, so sind q und p genau dann verschmelzbar, wenn $L_q = L_p$ ist.

Fassen wir alle zu einem Zustand z äquivalenten Zustände in dem neuen Zustand

$$[z]_\sim = \{z' \in Z \mid L_{z'} = L_z\}$$

zusammen (wofür wir auch kurz $[z]$ oder $\tilde{z}$ schreiben) und ersetzen wir Z und E durch $\tilde{Z} = \{\tilde{z} \mid z \in Z\}$ und $\tilde{E} = \{\tilde{z} \mid z \in E\}$, so erhalten wir den DFA $M' = (\tilde{Z}, \Sigma, \delta', [q_0], \tilde{E})$ mit

$$\delta'([q], a) = [\delta(q, a)].$$

Im nächsten Satz zeigen wir, dass M' tatsächlich der gesuchte Minimalautomat für $L(M)$ ist. Für eine Teilmenge $Q \subseteq Z$ bezeichne $\tilde{Q}$ die Menge $\{\tilde{q} \mid q \in Q\}$ aller Äquivalenzklassen $\tilde{q}$, die einen Repräsentanten q in Q haben.

Theorem 1.52 *Sei $M = (Z, \Sigma, \delta, q_0, E)$ ein DFA, der nur Zustände enthält, die vom Startzustand q_0 aus erreichbar sind. Dann ist $M' = (\tilde{Z}, \Sigma, \delta', [q_0], \tilde{E})$ mit*

$$\delta'([q], a) = [\delta(q, a)]$$

ein DFA für $L(M)$ mit einer minimalen Anzahl von Zuständen.

Beweis

- Wir zeigen zuerst, dass δ' wohldefiniert ist, also der Wert von $\delta'(\tilde{q}, a)$ nicht von der Wahl des Repräsentanten q abhängt. Hierzu zeigen wir, dass im Fall $p \sim q$ auch $\delta(q, a)$ und $\delta(p, a)$ äquivalent sind:

$$\begin{aligned} L_q = L_p &\Rightarrow \forall x \in \Sigma^* : x \in L_q \leftrightarrow x \in L_p \\ &\Rightarrow \forall x \in \Sigma^* : ax \in L_q \leftrightarrow ax \in L_p \\ &\Rightarrow \forall x \in \Sigma^* : x \in L_{\delta(q,a)} \leftrightarrow x \in L_{\delta(p,a)} \\ &\Rightarrow L_{\delta(q,a)} = L_{\delta(p,a)}. \end{aligned}$$

- Als nächstes zeigen wir, dass $L(M') = L(M)$ ist. Sei $x = x_1 \cdots x_n$ eine Eingabe und seien

  $$q_i = \hat{\delta}(q_0, x_1 \cdots x_i),\ i = 0, \ldots, n$$

  die von M beim Abarbeiten von x durchlaufenen Zustände. Wegen

  $$\delta'([q_{i-1}], x_i) = [\delta(q_{i-1}, x_i)] = [q_i]$$

  durchläuft M' dann die Zustände

  $$[q_0], [q_1], \cdots, [q_n].$$

  Da aber q_n genau dann zu E gehört, wenn $[q_n] \in \tilde{E}$ ist, folgt $L(M') = L(M)$.

- Es bleibt zu zeigen, dass M' eine minimale Anzahl $\|\tilde{Z}\|$ von Zuständen hat. Dies ist sicher dann der Fall, wenn bereits M minimal ist. Es reicht also zu zeigen, dass die Anzahl $k = \|\tilde{Z}\| = \|\{L_q \mid q \in Z\}\|$ nicht von M, sondern nur von $L(M)$ abhängt. Für $x \in \Sigma^*$ sei

$$L_x = \{y \in \Sigma^* \mid xy \in L(M)\}.$$

  Dann gilt $\{L_x \mid x \in \Sigma^*\} \subseteq \{L_q \mid q \in Z\}$, da $L_x = L_{\hat{\delta}(q_0,x)}$ ist. Die umgekehrte Inklusion gilt ebenfalls, da nach Voraussetzung jeder Zustand $q \in Z$ über ein $x \in \Sigma^*$ erreichbar ist. Also hängt $k = \|\{L_q \mid q \in Z\}\| = \|\{L_x \mid x \in \Sigma^*\}\|$ nur von $L(M)$ ab. □

Für die algorithmische Konstruktion von M' aus M steht man vor der Aufgabe, festzustellen, ob zwei Zustände p und q verschmelzbar sind oder nicht. Zur Beantwortung dieser Frage machen wir folgende Beobachtungen.

Beobachtung 1.53

- *Kein Endzustand $p \in E$ ist mit einem Zustand $q \in Z \setminus E$ verschmelzbar (wegen $\varepsilon \in L_p \Delta L_q$).*
- *Wenn $\delta(p,a)$ und $\delta(q,a)$ unverschmelzbar sind, dann sind auch p und q unverschmelzbar (wegen $p \sim q \Rightarrow \delta(p,a) \sim \delta(q,a)$).*
- *Wenn also D nur unverschmelzbare Zustandspaare enthält, dann sind auch alle Paare in der Menge*

$$D' = \{\{p,q\} \mid \exists a \in \Sigma : \{\delta(p,a), \delta(q,a)\} \in D\}.$$

  unverschmelzbar.

Daher berechnen wir ausgehend von der Menge

$$D_0 = \{\{p,q\} \mid p \in E, q \notin E\}$$

eine Folge von Mengen

$$D_0 \subseteq D_1 \subseteq \cdots \subseteq \{\{z,z'\} \subseteq Z \mid z \neq z'\}$$

mittels

$$D_{i+1} = D_i \cup \{\{p,q\} \mid \exists a \in \Sigma : \{\delta(p,a), \delta(q,a)\} \in D_i\},$$

indem wir zu D_i alle Paare $\{q,p\}$ hinzufügen, für die eines der Paare $\{\delta(q,a), \delta(p,a)\}$, $a \in \Sigma$, bereits zu D_i gehört. Da Z endlich ist, muss es ein j mit $D_{j+1} = D_j$ geben. In diesem Fall gilt (siehe Übungen):

$$p \sim q \Leftrightarrow \{p,q\} \notin D_j.$$

Daher kann nun M' durch Verschmelzen aller Zustände p, q mit $\{p,q\} \notin D_j$ gebildet werden. Damit erhalten wir folgenden Algorithmus zur Berechnung eines minimalen DFA.

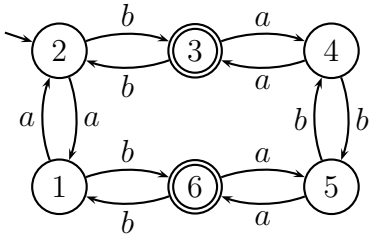
Algorithmus 1.54 MIN-DFA

1 **Eingabe:** DFA $M = (Z, \Sigma, \delta, q_0, E)$.
2 Entferne alle von q_0 aus unerreichbaren Zustände aus Z.
3 Markiere alle Paare $\{z, z'\}$ mit $z \in E$ und $z' \notin E$.
4 Solange noch ein unmarkiertes Paar $\{z, z'\} \subseteq Z$ existiert, für das eines der Paare $\{\delta(z, a), \delta(z', a)\}$, $a \in \Sigma$, bereits markiert ist, markiere auch $\{z, z'\}$.
5 Bilde die Verschmelzungsmengen

$$\tilde{z} = \{z\} \cup \{z' \in Z \mid \{z, z'\} \text{ ist nicht markiert}\}, \ z \in Z.$$

6 **Ausgabe:** Minimal-DFA $M' = (\tilde{Z}, \Sigma, \delta', \tilde{z}_0, \tilde{E})$ mit $\tilde{Z} = \{\tilde{z} \mid z \in Z\}$, $\tilde{E} = \{\tilde{z} \mid z \in E\}$ und $\delta'(\tilde{z}, a) = \widetilde{\delta(z, a)}$.

Beispiel 1.55 Betrachte den DFA M



Dann enthält D_0 die Paare

$$\{1,3\}, \{1,6\}, \{2,3\}, \{2,6\}, \{3,4\}, \{3,5\}, \{4,6\}, \{5,6\}.$$

Die Paare in D_0 sind in der folgenden Matrix durch eine Null markiert.

2					
3	0	0			
4	1	1	0		
5	1	1	0		
6	0	0		0	0
	1	2	3	4	5

Wegen

$\{p, q\}$	$\{1,4\}$	$\{1,5\}$	$\{2,4\}$	$\{2,5\}$
$\{\delta(q, a), \delta(p, a)\}$	$\{2,3\}$	$\{2,6\}$	$\{1,3\}$	$\{1,6\}$

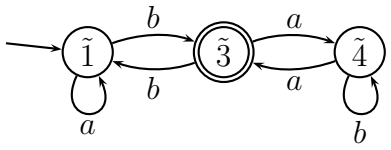
enthält D_1 zusätzlich die Paare $\{1,4\}$, $\{1,5\}$, $\{2,4\}$, $\{2,5\}$ (in obiger Matrix durch eine Eins markiert). Da die verbliebenen Paare $\{1,2\}$, $\{3,6\}$, $\{4,5\}$ wegen

$\{p, q\}$	$\{1,2\}$	$\{3,6\}$	$\{4,5\}$
$\{\delta(p, a), \delta(q, a)\}$	$\{1,2\}$	$\{4,5\}$	$\{3,6\}$
$\{\delta(p, b), \delta(q, b)\}$	$\{3,6\}$	$\{1,2\}$	$\{4,5\}$

nicht zu D_1 hinzugefügt werden können, ist $D_2 = D_1$. Aus den unmarkierten Paaren $\{1,2\}$, $\{3,6\}$ und $\{4,5\}$ erhalten wir die Verschmelzungsmengen

$$\tilde{1} = \{1,2\},\ \ \tilde{3} = \{3,6\} \text{ und } \tilde{4} = \{4,5\},$$

die auf folgenden Minimal-DFA M' führen:



◁

Durch eine leichte Modifikation von M' ist es möglich, einen Minimalautomaten M_L direkt aus einer regulären Sprache L zu gewinnen (also ohne den Umweg über einen DFA M für L). Da nämlich zwei Eingaben x und y den DFA M' genau dann in denselben Zustand $[\hat{\delta}(q_0,x)] = [\hat{\delta}(q_0,y)]$ überführen, wenn $L_x = L_y$ ist, können wir den von M' bei Eingabe x erreichten Zustand $[\hat{\delta}(q_0,x)]$ auch mit der Sprache L_x bezeichnen. Dies führt auf den DFA $M_L = (Z_L, \Sigma, \delta_L, L_\varepsilon, E_L)$ mit

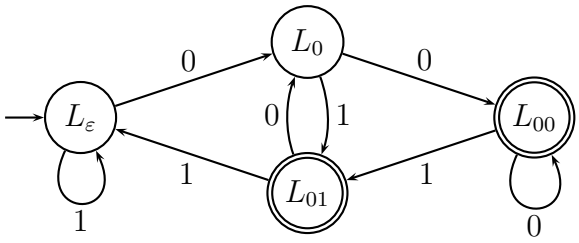
$$\begin{array}{rcl} Z_L & = & \{L_x \mid x \in \Sigma^*\}, \\ E_L & = & \{L_x \mid x \in L\} \text{ und} \\ \delta_L(L_x, a) & = & L_{xa}, \end{array}$$

welcher isomorph zu M' ist (also bis auf die Benennung der Zustände mit diesem identisch ist).

Beispiel 1.56 Für $L = \{x_1 \cdots x_n \mid x_i \in \{0,1\} \text{ für } i = 1, \ldots n \text{ und } x_{n-1} = 0\}$ ist

$$L_x = \begin{cases} L, & x \in \{\varepsilon, 1\} \text{ oder } x \text{ endet mit } 11, \\ L \cup \{0,1\}, & x = 0 \text{ oder } x \text{ endet mit } 10, \\ L \cup \{\varepsilon, 0, 1\}, & x \text{ endet mit } 00, \\ L \cup \{\varepsilon\}, & x \text{ endet mit } 01. \end{cases}$$

Somit erhalten wir den folgenden Minimalautomaten M_L:



◁

Im Fall, dass M bereits ein Minimalautomat ist, sind alle Zustände von M' von der Form $\tilde{q} = \{q\}$, so dass M isomorph zu M' und damit auch isomorph zu M_L ist. Dies zeigt, dass alle Minimalautomaten für eine Sprache L isomorph sind.

Um für eine reguläre Sprache L einen Minimalautomaten zu konstruieren, ist es nicht nötig, die Sprachen L_x zu bestimmen. Um die Überführungsfunktion aufzustellen, müssen wir nur herausfinden, welche Eigaben jeweils in denselben Zustand L_x führen. Wie die Sprachen L_x konkret aussehen, spielt dagegen keine Rolle. Daher werden häufig einfach die Äquivalenzklassen $[x] = \{y \mid x \, R_L \, y\}$ der durch

$$x \, R_L \, y \Leftrightarrow L_x = L_y$$

definierten Relation R_L als Zustände des Minimalautomaten verwendet. Dies führt auf den so genannten **Äquivalenzklassenautomaten** $M_{R_L} = (Z, \Sigma, \delta, [\varepsilon], E)$ mit

$$\begin{array}{rcl} Z & = & \{[x] \mid x \in \Sigma^*\}, \\ E & = & \{[x] \mid x \in L\} \text{ und} \\ \delta([x], a) & = & [xa]. \end{array}$$

Die Relation R_L gibt uns eine Möglichkeit an die Hand, herauszufinden, ob eine gegebene Sprache L regulär ist oder nicht. Offensichtlich gibt es genau dann einen DFA für L, wenn R_L die Menge Σ^* in endlich viele Klassen zerlegt.

Satz 1.57 (Myhill und Nerode) *Für eine Sprache L bezeichne $index(R_L) = \|\{[x]_{R_L} \mid x \in \Sigma^*\}\|$ den Index der Äquivalenzrelation R_L.*

1. $\mathsf{REG} = \{L \mid index(R_L) < \infty\}$.
2. *Ist L regulär, so gibt es bis auf Isomorphie nur einen DFA für L mit einer minimalen Anzahl von Zuständen.*

Beispiel 1.58 Sei $L = \{a^i b^i \mid i \geq 0\}$. Wegen $b^i \in L_{a^i} \Delta L_{a^j}$ für $i \neq j$ hat R_L unendlichen Index, d.h. L ist nicht regulär. ◁

1.6 Grammatiken

Eine beliebte Methode, Sprachen zu beschreiben, sind Grammatiken. Implizit haben wir hiervon bei der Definition der regulären Ausdrücke bereits Gebrauch gemacht.

Beispiel 1.59 Die Sprache RA aller regulären Ausdrücke über einem Alphabet $\Sigma = \{a_1, \ldots, a_k\}$ lässt sich aus dem Symbol R durch wiederholte Anwendung folgender Regeln erzeugen:

$$\begin{array}{lll} R \to \emptyset, & & R \to RR, \\ R \to \epsilon, & & R \to (R|R), \\ R \to a_i, \ i = 1, \ldots, k, & & R \to (R)^*. \end{array}$$

◁

Definition 1.60 *Eine **Grammatik** ist ein 4-Tupel $G = (V, \Sigma, P, S)$, wobei*

- *V eine endliche Menge von **Variablen** (auch **Nichtterminalsymbole** genannt),*
- *Σ das **Terminalalphabet**,*
- *$P \subseteq (V \cup \Sigma)^+ \times (V \cup \Sigma)^*$ eine endliche Menge von **Regeln** (oder **Produktionen**) und*
- *$S \in V$ die **Startvariable** ist.*

Für $(u, v) \in P$ schreiben wir auch kurz $u \to_G v$ bzw. $u \to v$, wenn die benutzte Grammatik aus dem Kontext ersichtlich ist.

Definition 1.61 *Seien $\alpha, \beta \in (V \cup \Sigma)^*$.*

*a) Wir sagen, β **ist aus** α **in** G **ableitbar** (kurz: $\alpha \Rightarrow_G \beta$), falls eine Regel $u \to_G v$ und Wörter $l, r \in (V \cup \Sigma)^*$ existieren mit*

$$\alpha = lur \text{ und } \beta = lvr.$$

Hierfür schreiben wir auch $l\underline{u}r \Rightarrow_G lvr$. (Man beachte, dass durch Unterstreichen von u in α sowohl die benutzte Regel als auch die Stelle in α, an der u durch v ersetzt wird, eindeutig erkennbar sind.)

*b) Die durch G **erzeugte Sprache** ist*

$$L(G) = \{x \in \Sigma^* \mid S \Rightarrow_G^* x\}.$$

Das n-fache Produkt von $\Rightarrow_G$ ist $\Rightarrow_G^n$, d.h. $\alpha \Rightarrow_G^n \beta$ besagt, dass β aus α **in** n **Schritten ableitbar** ist. Die reflexiv-transitive Hülle von $\Rightarrow_G$ ist $\Rightarrow_G^*$, d.h. $\alpha \Rightarrow_G^* \beta$ besagt, dass β aus α (in endlich vielen Schritten) **ableitbar** ist. Ein Wort $\alpha \in (V \cup \Sigma)^*$ heißt **Satzform**, wenn es aus dem Startsymbol S ableitbar ist.

Definition 1.62 *Eine **Ableitung** von x ist eine Folge*

$$\sigma = (l_0, u_0, r_0), \ldots, (l_m, u_m, r_m)$$

von Tripeln (l_i, u_i, r_i) mit $(l_0, u_0, r_0) = (\varepsilon, S, \varepsilon)$, $l_m u_m r_m = x$ und

$$l_i \underline{u_i} r_i \Rightarrow l_{i+1} u_{i+1} r_{i+1} \text{ für } i = 0, \ldots, m-1.$$

*Die **Länge** von σ ist m. Wir notieren eine Ableitung σ wie oben auch in der Form*

$$l_0 \underline{u_0} r_0 \Rightarrow l_1 \underline{u_1} r_1 \Rightarrow \cdots \Rightarrow l_{m-1} \underline{u_{m-1}} r_{m-1} \Rightarrow l_m u_m r_m.$$

Beispiel 1.63 Wir betrachten nochmals die Grammatik $G = (\{R\}, \Sigma \cup \{\emptyset, \epsilon, (,), ^*, |\}, P, R)$, die die Menge der regulären Ausdrücke über dem Alphabet Σ erzeugt, wobei P die oben angegebenen Regeln enthält. Ist $\Sigma = \{0, 1\}$, so lässt sich der reguläre Ausdruck $(01)^*(\epsilon|\emptyset)$ beispielsweise wie folgt ableiten:

$$\begin{aligned}\underline{R} &\Rightarrow \underline{R}R \Rightarrow (\underline{R})^*R \Rightarrow (RR)^*R \Rightarrow (\underline{R}R)^*(R|R)\\ &\Rightarrow (0\underline{R})^*(R|R) \Rightarrow (01)^*(\underline{R}|R) \Rightarrow (01)^*(\epsilon|\underline{R}) \Rightarrow (01)^*(\epsilon|\emptyset)\end{aligned}$$

◁

Man unterscheidet vier verschiedene Typen von Grammatiken.

Definition 1.64 *Sei $G = (V, \Sigma, P, S)$ eine Grammatik.*

1. *G heißt **vom Typ 3** oder **regulär**, falls für alle Regeln $u \to v$ gilt: $u \in V$ und $v \in \Sigma V \cup \Sigma \cup \{\epsilon\}$.*
2. *G heißt **vom Typ 2** oder **kontextfrei**, falls für alle Regeln $u \to v$ gilt: $u \in V$.*
3. *G heißt **vom Typ 1** oder **kontextsensitiv**, falls für alle Regeln $u \to v$ gilt: $|v| \geq |u|$ (mit Ausnahme der ε-Sonderregel, siehe unten).*
4. *Jede Grammatik ist automatisch **vom Typ 0**.*

ε-Sonderregel: In einer kontextsensitiven Grammatik $G = (V, \Sigma, P, S)$ kann auch die Regel $S \to \varepsilon$ benutzt werden. Aber nur, wenn das Startsymbol S nicht auf der rechten Seite einer Regel in P vorkommt.

Die Sprechweisen „vom Typ i“ bzw. „regulär“, „kontextfrei“ und „kontextsensitiv“ werden auch auf die durch solche Grammatiken erzeugte Sprachen angewandt. (Der folgende Satz rechtfertigt dies für die regulären Sprachen, die wir bereits mit Hilfe von DFAs definiert haben.) Die zugehörigen neuen Sprachklassen sind

$$\mathsf{CFL} = \{L(G) \mid G \text{ ist eine kontextfreie Grammatik}\},$$

(*context free languages*) und

$$\mathsf{CSL} = \{L(G) \mid G \text{ ist eine kontextsensitive Grammatik}\}$$

(*context sensitive languages*). Da die Klasse der Typ 0 Sprachen mit der Klasse der rekursiv aufzählbaren (*recursively enumerable*) Sprachen übereinstimmt, bezeichnen wir diese Sprachklasse mit

$$\mathsf{RE} = \{L(G) \mid G \text{ ist eine Grammatik}\}.$$

Die Sprachklassen

$$\mathsf{REG} \subset \mathsf{CFL} \subset \mathsf{CSL} \subset \mathsf{RE}$$

bilden eine Hierarchie (d.h. alle Inklusionen sind echt), die so genannte **Chomsky-Hierarchie**.

Als nächstes zeigen wir, dass sich mit regulären Grammatiken gerade die regulären Sprachen erzeugen lassen. Hierbei erweist sich folgende Beobachtung als nützlich.

Lemma 1.65 *Zu jeder regulären Grammatik $G = (V, \Sigma, P, S)$ gibt es eine äquivalente reguläre Grammatik G', die keine Produktionen der Form $A \to a$ hat.*

Beweis Betrachte die Grammatik $G' = (V', \Sigma, P', S)$ mit

$$\begin{aligned} V' &= V \cup \{X_{neu}\}, \\ P' &= \{A \to aX_{neu} \mid A \to_G a\} \cup \{X_{neu} \to \varepsilon\} \cup P \setminus (V \times \Sigma). \end{aligned}$$

Es ist leicht zu sehen, dass G' die gleiche Sprache wie G erzeugt. □

Satz 1.66 $\mathsf{REG} = \{L(G) \mid G \text{ ist eine reguläre Grammatik}\}$.

Beweis Sei $L \in \mathsf{REG}$ und sei $M = (Z, \Sigma, \delta, q_0, E)$ ein DFA mit $L(M) = L$. Wir konstruieren eine reguläre Grammatik $G = (V, \Sigma, P, S)$ mit $L(G) = L$. Setzen wir

$$\begin{aligned} V &= Z, \\ S &= q_0 \text{ und} \\ P &= \{q \to ap \mid \delta(q, a) = p\} \cup \{q \to \varepsilon \mid q \in E\}, \end{aligned}$$

so gilt für alle Wörter $x = x_1 \cdots x_n \in \Sigma^*$:

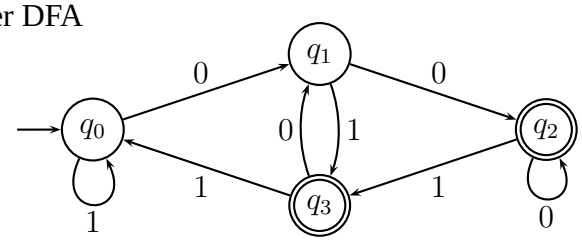
$$\begin{aligned} x \in L(M) &\Leftrightarrow \exists\, q_1, \ldots, q_{n-1} \in Z\ \exists\, q_n \in E : \delta(q_{i-1}, x_i) = q_i \text{ für } i = 1, \ldots, n \\ &\Leftrightarrow \exists\, q_1, \ldots, q_n \in V : q_{i-1} \to_G x_i q_i \text{ für } i = 1, \ldots, n \text{ und } q_n \to_G \varepsilon \\ &\Leftrightarrow \exists\, q_1, \ldots, q_n \in V : q_0 \Rightarrow_G^i x_1 \cdots x_i q_i \text{ für } i = 1, \ldots, n \text{ und } q_n \to_G \varepsilon \\ &\Leftrightarrow x \in L(G) \end{aligned}$$

Für die entgegengesetzte Inklusion sei nun $G = (V, \Sigma, P, S)$ eine reguläre Grammatik, die keine Produktionen der Form $A \to a$ enthält. Dann können wir die gerade beschriebene Konstruktion einer Grammatik aus einem DFA „umdrehen“, um ausgehend von G einen NFA $M = (Z, \Sigma, \delta, \{S\}, E)$ mit

$$\begin{aligned} Z &= V, \\ E &= \{A \mid A \to_G \varepsilon\} \text{ und} \\ \delta(A, a) &= \{B \mid A \to_G aB\} \end{aligned}$$

zu erhalten. Genau wie oben folgt nun $L(M) = L(G)$. □

Beispiel 1.67 Der DFA



führt auf die Grammatik $(\{q_0, q_1, q_2, q_3\}, \{0, 1\}, P, q_0)$ mit

$$\begin{aligned} P: \quad & q_0 \to 1q_0, 0q_1, \\ & q_1 \to 0q_2, 1q_3, \\ & q_2 \to 0q_2, 1q_3, \varepsilon, \\ & q_3 \to 0q_1, 1q_0, \varepsilon. \end{aligned}$$

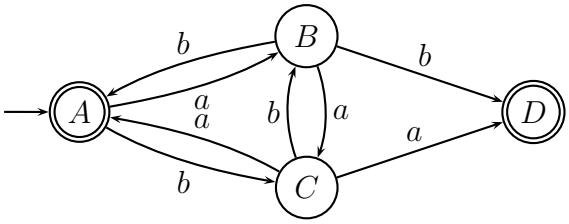
Umgekehrt führt die Grammatik $G = (\{A, B, C\}, \{a, b\}, P, A)$ mit

$$\begin{aligned} P: \quad & A \to aB, bC, \varepsilon, \\ & B \to aC, bA, b, \\ & C \to aA, bB, a \end{aligned}$$

über die Grammatik $G' = (\{A, B, C, D\}, \{a, b\}, P', A)$ mit

$$\begin{aligned} P': \quad & A \to aB, bC, \varepsilon, \\ & B \to aC, bA, bD, \\ & C \to aA, bB, aD, \\ & D \to \varepsilon \end{aligned}$$

auf den NFA



◁

1.7 Das Pumping-Lemma

Wie kann man von einer Sprache nachweisen, dass sie nicht regulär ist? Eine Möglichkeit besteht darin, die Kontraposition folgender Aussage anzuwenden.

Theorem 1.68 (Pumping-Lemma für reguläre Sprachen) *Zu jeder regulären Sprache L gibt es eine Zahl l, so dass sich alle Wörter $x \in L$ mit $|x| \geq l$ in $x = uvw$ zerlegen lassen mit*

1. $v \neq \varepsilon$,
2. $|uv| \leq l$ *und*
3. $uv^iw \in L$ *für alle* $i \geq 0$.

*Falls eine Zahl l mit diesen Eigenschaften existiert, wird das kleinste solche l die **Pumping-Zahl** von L genannt.*

Beweis Sei $M = (Z, \Sigma, \delta, q_0, E)$ ein DFA mit $L(M) = L$ und sei l die Anzahl der Zustände von M. Setzen wir nun M auf eine Eingabe $x = x_1 \cdots x_n \in L$ der Länge $n \geq l$ an, so muss M nach spätestens l Schritten einen Zustand $q \in Z$ zum zweiten Mal annehmen:

$$\exists\, 0 \leq j < k \leq l : \hat{\delta}(q_0, x_1 \cdots x_j) = \hat{\delta}(q_0, x_1 \cdots x_k) = q.$$

Wählen wir nun $u = x_1 \cdots x_j$, $v = x_{j+1} \cdots x_k$ und $w = x_{k+1} \cdots x_n$, so ist $|v| = k - j \geq 1$ und $|uv| = k \leq l$. Ausserdem gilt $uv^i w \in L$ für $i \geq 0$, da wegen $\hat{\delta}(q, v) = q$

$$\hat{\delta}(q_0, uv^i w) = \hat{\delta}(\hat{\delta}(\underbrace{\hat{\delta}(q_0, u)}_{q}, v^i), w) = \hat{\delta}(\underbrace{\hat{\delta}(q, v^i)}_{q}, w) = \hat{\delta}(q_0, x) \in E$$

ist. □

Beispiel 1.69 Die Sprache

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

hat die Pumping-Zahl $l = 3$. Sei nämlich $x \in L$ beliebig mit $|x| \geq 3$. Dann lässt sich innerhalb des Präfixes von x der Länge drei ein nichtleeres Teilwort v finden, das gepumpt werden kann:

1. Fall: x hat das Präfix ab (oder ba).
Zerlege $x = uvw$ mit $u = \varepsilon$ und $v = ab$ (bzw. $v = ba$).

2. Fall: x hat das Präfix aab (oder bba).
Zerlege $x = uvw$ mit $u = a$ (bzw. $u = b$) und $v = ab$ (bzw. $v = ba$).

3. Fall: x hat das Präfix aaa (oder bbb).
Zerlege $x = uvw$ mit $u = \varepsilon$ und $v = aaa$ (bzw. $v = bbb$). ◁

Beispiel 1.70 Eine endliche Sprache L hat die Pumping-Zahl

$$l = \begin{cases} 0, & L = \emptyset, \\ \max\{|x| + 1 \mid x \in L\}, & \text{sonst.} \end{cases}$$

Tatsächlich lässt sich jedes Wort $x \in L$ der Länge $|x| \geq l$ „pumpen“ (da solche Wörter gar nicht existieren), weshalb die Pumping-Zahl höchstens l ist. Zudem gibt es im Fall $l > 0$ ein Wort $x \in L$ der Länge $|x| = l - 1$, das sich nicht „pumpen“ lässt, weshalb die Pumping-Zahl nicht $l - 1$ sein kann. ◁

Wollen wir mit Hilfe des Pumping-Lemmas von einer Sprache L zeigen, dass sie nicht regulär ist, so genügt es, für jede Zahl l ein Wort $x \in L$ der Länge $|x| \geq l$ anzugeben, so dass für jede Zerlegung von x in drei Teilwörter u, v, w mindestens eine der drei in Satz 1.68 aufgeführten Eigenschaften verletzt ist.

Beispiel 1.71

- Die Sprache
$$L = \{a^j b^j \mid j \geq 0\}$$
ist nicht regulär, da sich für jede Zahl $l \geq 0$ das Wort $x = a^l b^l$ der Länge $|x| = 2l \geq l$ in der Sprache L befindet, welches offensichtlich nicht in Teilwörter u, v, w mit $v \neq \varepsilon$ und $uv^2w \in L$ zerlegbar ist.

- Die Sprache
$$L = \{a^{n^2} \mid n \geq 0\}$$
ist ebenfalls nicht regulär. Andernfalls müsste es nämlich eine Zahl l geben, so dass jede Quadratzahl $n^2 \geq l$ als Summe von natürlichen Zahlen $u + v + w$ darstellbar ist mit der Eigenschaft, dass $v \geq 1$ und $u + v \leq l$ ist, und für jedes $i \geq 0$ auch $u+iv+w$ eine Quadratzahl ist. Insbesondere müsste also $u+2v+w = n^2 + v$ eine Quadratzahl sein, was wegen
$$n^2 < n^2 + v \leq n^2 + l < n^2 + 2l + 1 = (n+1)^2$$
ausgeschlossen ist.

- Schließlich ist auch die Sprache
$$L = \{a^p \mid p \text{ prim }\}$$
nicht regulär, da sich sonst jede Primzahl p einer bestimmten Mindestgröße l als Summe von natürlichen Zahlen $u + v + w$ darstellen ließe, so dass $v \geq 1$, $u + v \leq l$ und für alle $i \geq 0$ auch $u + iv + w$ prim ist. Insbesondere müsste also $u + (u + w)v + w$ eine Primzahl sein, was wegen
$$u + (u+w)v + w = (u+w)(v+1) = \underbrace{(p-v)}_{\geq p-l}\underbrace{(v+1)}_{\geq 2}$$
für alle Primzahlen $p \geq l + 2$ ausgeschlossen ist. ◁

Bemerkung 1.72 *Mit dem Pumping-Lemma können nicht alle Sprachen $L \notin \mathsf{REG}$ als nicht regulär nachgewiesen werden, da seine Umkehrung falsch ist. Beispielsweise hat die Sprache*
$$L = \{a^i b^j c^k \mid i = 0 \text{ oder } j = k\}$$
die Pumping-Zahl 1 (jedes Wort $x \in L$ mit Ausnahme von ε kann also „gepumpt“ werden), obwohl L nicht regulär ist (siehe Übungen).

Kapitel 2

Kontextfreie Sprachen

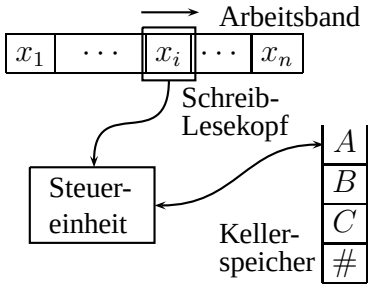
2.1 Kellerautomaten

Wie wir gesehen haben, ist die Sprache $L = \{a^n b^n \mid n \geq 0\}$ nicht regulär. Es ist aber leicht, eine kontextfreie Grammatik für L zu finden:

$$G = (\{S\}, \{a, b\}, \{S \to aSb, S \to \varepsilon\}, S).$$

In diesem Abschnitt befassen wir uns mit der Frage, wie sich das Maschinenmodell des DFA erweitern lässt, um die Sprache L und alle anderen kontextfreien Sprachen erkennen zu können. Dass ein DFA die Sprache $L = \{a^n b^n \mid n \geq 0\}$ nicht erkennen kann, liegt an seinem beschränkten Speichervermögen, das zwar von L aber nicht von der Eingabe abhängen darf. Um L erkennen zu können, reicht bereits ein so genannter Kellerspeicher (engl. *stack, pushdown memory*) aus. Dieser erlaubt nur den Zugriff auf die höchste belegte Speicheradresse. Ein Kellerautomat

- verfügt über einen Kellerspeicher,
- kann ε-Übergänge machen,
- liest in jedem Schritt das aktuelle Eingabezeichen und das oberste Kellersymbol,
- kann das oberste Kellersymbol entfernen (durch eine **pop- Operation**) und



- danach beliebig viele Symbole einkellern (mittels **push- Operationen**).

Für eine Menge M bezeichne $\mathcal{P}_e(M)$ die Menge aller endlichen Teilmengen von M, d.h.

$$\mathcal{P}_e(M) = \{A \subseteq M \mid A \text{ ist endlich}\}.$$

Definition 2.1 *Ein **Kellerautomat** (kurz: PDA; pushdown automaton) wird durch ein 6-Tupel $M = (Z, \Sigma, \Gamma, \delta, q_0, \#)$ beschrieben, wobei*

- *Z, Σ und q_0 wie bei einem DFA,*
- *Γ das **Kelleralphabet**,*
- *$\delta : Z \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \to \mathcal{P}_e(Z \times \Gamma^*)$ die **Überführungsfunktion** und*
- *$\# \in \Gamma$ das **Kelleranfangszeichen** ist.*

Arbeitsweise eines PDA

Wenn q der momentane Zustand, A das oberste Kellerzeichen und $u \in \Sigma$ das nächste Eingabezeichen (bzw. $u = \varepsilon$) ist, so kann M im Fall $(p, B_1 \cdots B_k) \in \delta(q, u, A)$

- in den Zustand p wechseln,
- den Lesekopf auf dem Eingabeband um $|u|$ Positionen vorrücken und
- das Zeichen A im Keller durch die Zeichenfolge $B_1 \cdots B_k$ ersetzen.

Hierfür sagen wir auch, M führt die **Anweisung** $quA \to pB_1 \cdots B_k$ aus. Da im Fall $u = \varepsilon$ kein Eingabezeichen gelesen wird, spricht man auch von einem **spontanen** Übergang (oder **ε-Übergang**). Eine **Konfiguration** wird durch ein Tripel

$$K = (q, x_i \cdots x_n, A_1 \cdots A_l) \in Z \times \Sigma^* \times \Gamma^*$$

beschrieben und besagt, dass

- q der momentane Zustand,
- $x_i \cdots x_n$ der ungelesene Rest der Eingabe und
- $A_1 \cdots A_l$ der aktuelle Kellerinhalt ist (oberstes Kellerzeichen ist A_1).

Eine Anweisung $quA_1 \to pB_1 \cdots B_k$ (mit $u \in \{\varepsilon, x_i\}$) überführt die Konfiguration K in die **Folgekonfiguration**

$$K' = (p, x_j \cdots x_n, B_1 \cdots B_k A_2 \cdots A_l) \text{ mit } j = i + |u|.$$

Hierfür schreiben wir auch kurz $K \vdash K'$. Die reflexive, transitive Hülle von $\vdash$ bezeichnen wir wie üblich mit $\vdash^*$. Die von M **akzeptierte** oder **erkannte Sprache** ist

$$L(M) \;=\; \{x \in \Sigma^* \mid \exists\, p \in Z : (q_0, x, \#) \vdash^* (p, \varepsilon, \varepsilon)\}.$$

Ein Wort x wird also genau dann von M akzeptiert, wenn es eine Rechnung (Folge von Konfigurationen) von M bei Eingabe x gibt, die ausgehend von der **Startkonfiguration** $(q_0, x, \#)$ das gesamte Wort bis zum Ende liest und den Keller leert. Man beachte, dass bei leerem Keller kein weiterer Übergang mehr möglich ist.

Beispiel 2.2 Sei M der PDA $(\{q,p\},\{a,b\},\{A,\#\},\delta,q,\#)$ mit

$$\begin{array}{lllll}\delta: & q\varepsilon\# \to q & (1) & qaA \to qAA & (3) \quad pbA \to p \quad (5)\\ & qa\# \to qA & (2) & qbA \to p & (4)\end{array}$$

Dann wird das Wort $x = aabb$ beispielsweise durch folgende Rechnung akzeptiert:

$$(q, aabb, \#) \underset{(2)}{\vdash} (q, abb, A) \underset{(3)}{\vdash} (q, bb, AA) \underset{(4)}{\vdash} (p, b, A) \underset{(5)}{\vdash} (p, \varepsilon, \varepsilon).$$

Allgemein kann das Wort $x = a^n b^n$ wie folgt akzeptiert werden:

$n = 0$**:** $(q, \varepsilon, \#) \underset{(1)}{\vdash} (p, \varepsilon, \varepsilon)$.

$n \geq 1$**:** $\begin{aligned}(q, a^n b^n, \#) &\underset{(2)}{\vdash} (q, a^{n-1} b^n, A) \underset{(3)}{\vdash^{n-1}} (q, b^n, A^n)\\ &\underset{(4)}{\vdash} (p, b^{n-1}, A^{n-1}) \underset{(5)}{\vdash^{n-1}} (p, \varepsilon, \varepsilon).\end{aligned}$

Dies zeigt $\{a^n b^n \mid n \geq 0\} \subseteq L(M)$. Als nächstes zeigen wir, dass jede von M akzeptierte Eingabe $x = x_1 \ldots x_n \in L(M)$ die Form $x = a^m b^m$ hat.

Ausgehend von der Startkonfiguration $(q, x, \#)$ sind nur die Anweisungen (1) oder (2) anwendbar. Da Anweisung (1) den Keller leert, ohne ein Zeichen zu lesen, muss x in diesem Fall das leere Wort $x = \varepsilon = a^0 b^0$ sein.

Falls M die Rechnung mit Anweisung (2) beginnt, muss M in den Zustand p gelangen, um den Keller leeren zu können. Da eine Rückkehr von p nach q nicht möglich ist, wechselt M den Zustand nur einmal, und zwar mittels Anweisung (4). Bis dahin kann M nur a's lesen, wobei für jedes gelesene a ein A eingekellert wird. Liest M bis zum Wechsel in den Zustand p insgesamt m a's, so muss die Rechnung wie folgt verlaufen:

$$\begin{aligned}(q, x_1 \ldots x_n, \#) &\underset{(2)}{\vdash} (q, x_2 \ldots x_n, A) \underset{(3)}{\vdash^{m-1}} (q, x_{m+1} \cdots x_n, A^m)\\ &\underset{(4)}{\vdash} (p, x_{m+2} \cdots x_n, A^{m-1})\end{aligned}$$

mit $x_1 = x_2 = \cdots = x_m = a$ und $x_{m+1} = b$. Um den Keller zu leeren, muss M noch $m-1$ weitere b's lesen. Also muss $n = 2m$ und $x_{m+2} = \cdots = x_{2m} = b$ sein. x hat also auch in diesem Fall die Form $a^m b^m$. ◁

2.2 Äquivalenz von kontextfreien Grammatiken und Kellerautomaten

Als nächstes wollen wir zeigen, dass eine Sprache genau dann kontextfrei ist, wenn sie von einem PDA erkannt wird. Hierzu benötigen wir den Begriff der Linksableitung.

Definition 2.3 *Sei $G = (V, \Sigma, P, S)$ eine kontextfreie Grammatik.*

a) Eine Ableitung

$$\alpha_0 = l_0 \underline{A_0} r_0 \Rightarrow l_1 \underline{A_1} r_1 \Rightarrow \cdots \Rightarrow l_{m-1} \underline{A_{m-1}} r_{m-1} \Rightarrow \alpha_m.$$

*heißt **Linksableitung** von α (kurz $\alpha_0 \Rightarrow_L^* \alpha_m$), falls in jedem Ableitungsschritt die am weitesten links stehende Variable ersetzt wird, d.h. es gilt $l_i \in \Sigma^*$ für $i = 0, \ldots, m-1$.*

*b) **Rechtsableitungen** $\alpha_0 \Rightarrow_R^* \alpha_m$ sind analog definiert.*

*c) G heißt **mehrdeutig**, wenn es ein Wort $x \in L(G)$ gibt, das zwei verschiedene Linksableitungen $S \Rightarrow_L^* x$ hat.*

Es ist leicht zu sehen, dass für alle Wörter $x \in \Sigma^*$ folgende Äquivalenzen gelten:

$$x \in L(G) \;\Leftrightarrow\; S \Rightarrow_L^* x \;\Leftrightarrow\; S \Rightarrow_R^* x.$$

Als nächstes wollen wir zeigen, dass PDAs genau die kontextfreien Sprachen erkennen.

Satz 2.4 $\mathsf{CFL} = \{L(M) \mid M \textit{ ist ein PDA}\}$.

Beweis Wir zeigen zuerst die Inklusion $\mathsf{CFL} \subseteq \{L(M) \mid M \text{ ist ein PDA}\}$.
Idee: Konstruiere zu einer kontextfreien Grammatik $G = (V, \Sigma, P, S)$ einen PDA $M = (\{q\}, \Sigma, \Gamma, \delta, q_0, S)$ mit $\Gamma = V \cup \Sigma$, so dass

$$S \Rightarrow_L^* x_1 \cdots x_n \text{ gdw. } (q, x_1 \cdots x_n, S) \vdash^* (q, \varepsilon, \varepsilon)$$

gilt. Hierzu fügen wir die Anweisungen

$$\begin{aligned} q\varepsilon A &\to q\alpha, \quad A \to_G \alpha, \\ qaa &\to q\varepsilon, \quad a \in \Sigma. \end{aligned}$$

zu δ hinzu. Dann ist leicht zu sehen, dass sogar

$$(q, x_1 \cdots x_n, S) \vdash^l (q, \varepsilon, \varepsilon) \text{ gdw. } S \Rightarrow_L^{l-n} x_1 \cdots x_n$$

gilt. Daher folgt

$$x \in L(G) \;\Leftrightarrow\; S \Rightarrow_L^* x \;\Leftrightarrow\; (q, x, S) \vdash^* (q, \varepsilon, \varepsilon) \;\Leftrightarrow\; x \in L(M).$$

Als nächstes zeigen wir die entgegengesetzte Inklusion $\{L(M) \mid M \text{ ist ein PDA}\} \subseteq \mathsf{CFL}$.

Idee: Konstruiere zu einem PDA $M = (Z, \Sigma, \Gamma, \delta, q_0, \#)$ eine kontextfreie Grammatik $G = (V, \Sigma, P, S)$ mit Variablen X_{pAq}, $A \in \Gamma$, $p, q \in Z$, so dass

$$(p, x, A) \vdash^* (q, \varepsilon, \varepsilon) \text{ gdw. } X_{pAq} \Rightarrow^* x.$$

gilt. Hierzu fügen wir für jede Anweisung $puA \to q_1 B_1 \cdots B_k$, $k \geq 0$, die folgenden Regeln zu P hinzu:

$$X_{pAq_{k+1}} \to u X_{q_1 B_1 q_2} \cdots X_{q_k B_k q_{k+1}}, \quad q_2, \ldots, q_{k+1} \in Z.$$

Um damit alle Wörter $x \in L(M)$ aus S ableiten zu können, benötigen wir jetzt nur noch die Regeln $S \to X_{q_0 \# q}$, $q \in Z$. Damit hat M die Variablenmenge

$$V = \{S\} \cup \{X_{pAq} \mid p, q \in Z, A \in \Gamma\},$$

und P enthält neben den Regeln $S \to X_{q_0 \# q}$, $q \in Z$, für jede Anweisung $puA \to q_1 B_1 \cdots B_k$ von M die folgenden Regeln:

$$X_{pAq_{k+1}} \to u X_{q_1 B_1 q_2} \cdots X_{q_k B_k q_{k+1}}, \; q_2, \ldots, q_{k+1} \in Z.$$

Dann folgt $L(G) = L(M)$ aus der Äquivalenz

$$(p, x, A) \vdash^* (q, \varepsilon, \varepsilon) \text{ gdw. } X_{pAq} \Rightarrow^* x,$$

deren Beweis wir gleich nachholen. Es gilt nämlich

$$\begin{aligned} x \in L(M) &\Leftrightarrow (q_0, x, \#) \vdash^* (q, \varepsilon, \varepsilon) \text{ für ein } q \in Z \\ &\Leftrightarrow S \Rightarrow X_{q_0 \# q} \Rightarrow^* x \text{ für ein } q \in Z \\ &\Leftrightarrow x \in L(G). \end{aligned}$$

Es bleibt zu zeigen, dass für alle $p, q \in Z$, $A \in \Gamma$ und $x \in \Sigma^*$ die Äquivalenz

$$(p, x, A) \vdash^* (q, \varepsilon, \varepsilon) \text{ gdw. } X_{pAq} \Rightarrow^* x.$$

gilt. Hierzu zeigen wir durch Induktion über m folgende stärkere Behauptung:

> Für alle $p, q \in Z$, $A \in \Gamma$, $x \in \Sigma^*$ und $m \geq 0$ gilt:
>
> $$(p, x, A) \vdash^m (q, \varepsilon, \varepsilon) \text{ gdw. } X_{pAq} \Rightarrow^m x.$$

$m = 0$**:** Da sowohl $(p, x, A) \vdash^0 (q, \varepsilon, \varepsilon)$ als auch $X_{pAq} \Rightarrow^0 x$ falsch sind, ist die Äquivalenz für $m = 0$ erfüllt.

$m \rightsquigarrow m + 1$**:** Wir zeigen unter der Induktionsvoraussetzung (IV), dass für alle $i \leq m$, $p, q \in Z$, $A \in \Gamma$ und $x \in \Sigma^*$ die Äquivalenz

$$(p, x, A) \vdash^i (q, \varepsilon, \varepsilon) \text{ gdw. } X_{pAq} \Rightarrow^i x$$

gilt, folgnde Induktionsbehauptung (IB):

Für alle $p, q \in Z$, $A \in \Gamma$ und $x \in \Sigma^*$ gilt

$$(p, x, A) \vdash^{m+1} (q, \varepsilon, \varepsilon) \text{ gdw. } X_{pAq} \Rightarrow^{m+1} x.$$

Wir beginnen mit dem Induktionsschritt für die Implikation von links nach rechts. Sei eine Rechnung $(p, x, A) \vdash^{m+1} (q, \varepsilon, \varepsilon)$ der Länge $m + 1$ gegeben und sei $puA \to q_1 B_1 \cdots B_k$, $k \geq 0$, die im ersten Rechenschritt ausgeführte Anweisung:

$$(p, x, A) \vdash (q_1, x', B_1 \cdots B_k) \vdash^m (q, \varepsilon, \varepsilon), \text{ wobei } x = ux' \text{ ist.}$$

Für $i = 2, \ldots, k$ sei q_i der Zustand von M, wenn B_i oberstes Kellerzeichen wird, und für $i = 1, \ldots, k$ sei u_i das Teilwort von x', das M zwischen den Besuchen von q_i und q_{i+1} verarbeitet (wobei $q_{k+1} = q$ ist).

Dann enthält P die Regel $X_{pAq} \to u X_{q_1 B_1 q_2} \cdots X_{q_k B_k q_{k+1}}$ und es gibt Zahlen $m_i \geq 1$ mit $m_1 + \cdots + m_k = m$ und

$$(q_i, u_i, B_i) \vdash^{m_i} (q_{i+1}, \varepsilon, \varepsilon) \text{ für } i = 1, \ldots, k.$$

Nach IV gibt es daher Ableitungen

$$X_{q_i B_i q_{i+1}} \Rightarrow^{m_i} u_i, \ i = 1, \ldots, k,$$

die wir zu der gesuchten Ableitung zusammensetzen können:

$$\begin{array}{lll} X_{pAq} & \Rightarrow & u X_{q_1 B_1 q_2} \cdots X_{q_{k-1} B_{k-1} q_k} X_{q_k B_k q} \\ & \Rightarrow^{m_1} & u u_1 X_{q_2 B_2 q_3} \cdots X_{q_{k-1} B_{k-1} q_k} X_{q_k B_k q} \\ & \vdots & \\ & \Rightarrow^{m_{k-1}} & u u_1 \cdots u_{k-1} X_{q_k B_k q} \\ & \Rightarrow^{m_k} & u u_1 \cdots u_k = x. \end{array}$$

Zuletzt zeigen wir den Induktionsschritt für die Implikation von rechts nach links. Gelte also umgekehrt $X_{pAq} \Rightarrow^{m+1} x$ und sei α die im ersten Schritt abgeleitete Satzform, d.h.

$$X_{pAq} \Rightarrow \alpha \Rightarrow^m x.$$

Wegen $X_{pAq} \to_G \alpha$ gibt es eine Anweisung $puA \to q_1 B_1 \cdots B_k$, $k \geq 0$, und Zustände $q_1, \ldots, q_k \in Z$ mit

$$\alpha = u X_{q_1 B_1 q_2} \cdots X_{q_{k-1} B_{k-1} q_k} X_{q_k B_k q_{k+1}}, \text{ wobei } q_{k+1} = q \text{ ist.}$$

Wegen $\alpha \Rightarrow^m x$ ex. eine Zerlegung $x = u u_1 \cdots u_k$ und Zahlen $m_i \geq 1$ mit $m_1 + \cdots + m_k = m$ mit

$$X_{q_i B_i q_{i+1}} \Rightarrow^{m_i} u_i \ (i = 1, \ldots, k), \text{ wobei } q_{k+1} = q \text{ ist.}$$

Nach IV gibt es somit Rechnungen

$$(q_i, u_i, B_i) \vdash^{m_i} (q_{i+1}, \varepsilon, \varepsilon), \ i = 1, \ldots, k,$$

aus denen sich die gesuchte Rechnung der Länge $m+1$ zusammensetzen lässt:

$$\begin{aligned}(q, uu_1\cdots u_k, A) &\vdash \quad (q_1, u_1\cdots u_k, B_1\cdots B_k)\\ &\vdash^{m_1} \quad (q_2, u_2\cdots u_k, B_2\cdots B_k)\\ &\vdots\\ &\vdash^{m_{k-1}} (q_k, u_k, B_k)\\ &\vdash^{m_k} \quad (q, \varepsilon, \varepsilon).\end{aligned}$$

□

Beispiel 2.5 Sei $G = (\{S\}, \{a, b\}, P, S)$ mit

$$P\colon\ S \to aSbS,\ (1) \quad S \to a.\ (2)$$

Der zugehörige PDA besitzt dann die Anweisungen

$$\begin{aligned}\delta:\ qaa \to q\varepsilon,\ (0) \quad q\varepsilon S \to qaSbS,\ (1')\\ qbb \to q\varepsilon,\ (0') \quad q\varepsilon S \to qa \qquad (2')\end{aligned}$$

und der Linksableitung

$$S \underset{(1)}{\Rightarrow} aSbS \underset{(2)}{\Rightarrow} aabS \underset{(2)}{\Rightarrow} aaba$$

entspricht dann die Rechnung

$$\begin{aligned}(q, aaba, S) &\underset{(1')}{\vdash} (q, aaba, aSbS) \underset{(0)}{\vdash} (q, aba, SbS)\\ &\underset{(2')}{\vdash} (q, aba, abS) \underset{(0)}{\vdash} (q, ba, bS)\\ &\underset{(0')}{\vdash} (q, a, S) \underset{(2')}{\vdash} (q, a, a) \underset{(0)}{\vdash} (q, \varepsilon, \varepsilon).\end{aligned}$$

◁

Beispiel 2.6 Sei M der PDA $(\{p, q\}, \{a, b\}, \{A, \#\}, \delta, p, \#)$ mit

$$\begin{aligned}\delta:\ p\varepsilon\# \to q\varepsilon,\ (1) \quad paA \to pAA,\ (3) \quad qbA \to q\varepsilon.\ (5)\\ pa\# \to pA,\ (2) \quad pbA \to q\varepsilon, \quad (4)\end{aligned}$$

Dann erhalten wir die Grammatik $G = (V, \Sigma, P, S)$ mit

$$V = \{S, X_{p\#p}, X_{p\#q}, X_{q\#p}, X_{q\#q}, X_{pAp}, X_{pAq}, X_{qAp}, X_{qAq}\}$$

und den Regeln P:

$$\begin{array}{llll}S \to X_{p\#p}, X_{p\#q}, (0, 0') & X_{pAp} \to aX_{pAp}X_{pAp}, & (3') & X_{pAq} \to b, (4')\\ X_{p\#q} \to \varepsilon, & (1') & X_{pAp} \to aX_{pAq}X_{qAp}, & (3'') \quad X_{qAq} \to b.\ (5')\\ X_{p\#p} \to aX_{pAp}, & (2') & X_{pAq} \to aX_{pAp}X_{pAq}, & (3''')\\ X_{p\#q} \to aX_{pAq}, & (2'') & X_{pAq} \to aX_{pAq}X_{qAq}, & (3'''')\end{array}$$

Der Rechnung

$$(p, aabb, \#) \underset{(2)}{\vdash} (p, abb, A) \underset{(3)}{\vdash} (p, bb, AA) \underset{(4)}{\vdash} (q, b, A) \underset{(5)}{\vdash} (q, \varepsilon, \varepsilon)$$

entspricht dann die Ableitung

$$S \underset{(0')}{\Rightarrow} X_{p\#q} \underset{(2'')}{\Rightarrow} aX_{pAq} \underset{(3'''')}{\Rightarrow} aaX_{pAq}X_{qAq} \underset{(4')}{\Rightarrow} aabX_{qAq} \underset{(5')}{\Rightarrow} aabb.$$

◁

2.3 Das Wortproblem für kontextfreie Grammatiken

Wie wir im letzten Abschnitt gesehen haben, können wir zwar zu jeder kontextfreien Grammatik G einen PDA M mit $L(M) = L(G)$ konstruieren. Dabei ist M bei Eingabe von G auch effizient berechenbar. Dennoch liefert dieser Ansatz keinen effizienten Algorithmus für das Wortproblem, da M nichtdeterministisch ist. In diesem Abschnitt stellen wir einen effizienten Algorithmus zur Lösung des Wortproblems für kontextfreie Grammatiken vor, das wie folgt definiert ist.

Gegeben: Eine kontextfreie Grammatik G und ein Wort x.

Gefragt: Ist $x \in L(G)$?

2.3.1 Chomsky-Normalform

Um das Wortproblem für kontextfreie Grammatiken zu lösen, transformieren wir die gegebene Grammatik zunächst in Chomsky-Normalform, die wie folgt definiert ist.

Definition 2.7 *Eine Grammatik (V, Σ, P, S) ist in **Chomsky-Normalform (CNF)**, falls alle Regeln die Form $A \to BC$ oder $A \to a$ haben.*

Außer einer effizienten Lösung des Wortproblems für kontextfreie Sprachen ermöglichen CNF-Grammatiken den Beweis des Pumping-Lemmas für kontextfreie Sprachen, mit dem sich viele Sprachen als nicht kontextfrei nachweisen lassen. Um eine gegebene Grammatik in Chomsky-Normalform zu bringen, entfernen wir zuerst alle ε-Produktionen.

Satz 2.8 *Zu jeder kontextfreien Grammatik G gibt es eine kontextfreie Grammatik G' ohne ε-Produktionen mit $L(G') = L(G) \setminus \{\varepsilon\}$.*

Beweis Zuerst sammeln wir mit folgendem Algorithmus alle Variablen A, aus denen das leere Wort ableitbar ist.

Algorithmus 2.9 BESTIMMUNG VON $E = \{A \in V \mid A \Rightarrow^* \varepsilon\}$

Eingabe: $G = (V, \Sigma, P, S)$
  $E \leftarrow \emptyset$
  $E' \leftarrow \{A \in V \mid A \to_G \varepsilon\}$
  while $E' \neq E$ **do**
    $E \leftarrow E'$
    $E' \leftarrow E \cup \{A \in V \mid \exists B_1, \ldots, B_k \in E : A \to_G B_1 \cdots B_k\}$
  end
Ausgabe: E

Nun konstruieren wir $G' = (V, \Sigma, P', S)$ wie folgt:

> Nehme zu P' alle Regeln $A \to \alpha'$ mit $\alpha' \neq \varepsilon$ hinzu, für die P eine Regel $A \to \alpha$ enthält, so dass α' aus α durch Entfernen von beliebig vielen Variablen $A \in E$ hervorgeht.

□

Beispiel 2.10 Betrachte die Grammatik $G = (\{S, T, U, X, Y, Z\}, \{a, b, c\}, P, S)$ mit

$$\begin{array}{lll} P: & S \to aY, bX, Z; & Y \to bS, aYY; \quad T \to U; \\ & X \to aS, bXX; & Z \to \varepsilon, S, T, cZ; \quad U \to abc. \end{array}$$

Bei der Berechnung von $E = \{A \in V \mid A \Rightarrow^* \varepsilon\}$ ergeben sich der Reihe nach folgende Belegungen für die Mengenvariabln E und E':

E	$\emptyset$	$\{Z\}$	$\{Z, S\}$
E'	$\{Z\}$	$\{Z, S\}$	$\{Z, S\}$

Um nun die Regelmenge P' zu bilden, entfernen wir aus P die einzige ε-Regel $Z \to \varepsilon$ und fügen die Regeln $X \to a$ (wegen $X \to aS$), $Y \to b$ (wegen $Y \to bS$) und $Z \to c$ (wegen $Z \to cZ$) hinzu:

$$\begin{array}{lll} P': & S \to aY, bX, Z; & Y \to b, bS, aYY; \quad T \to U; \\ & X \to a, aS, bXX; & Z \to c, S, T, cZ; \quad U \to abc. \end{array}$$

◁

Als direkte Anwendung des obigen Satzes können wir die Inklusion der Klasse der Typ 2 Sprachen in der Klasse der Typ 1 Sprachen zeigen.

Korollar 2.11 $\mathsf{REG} \subsetneq \mathsf{CFL} \subseteq \mathsf{CSL} \subseteq \mathsf{RE}$.

Beweis Die Inklusionen $\mathsf{REG} \subseteq \mathsf{CFL}$ und $\mathsf{CSL} \subseteq \mathsf{RE}$ sind klar. Wegen $\{a^n b^n | n \geq 0\} \in \mathsf{CFL} - \mathsf{REG}$ ist die Inklusion $\mathsf{REG} \subseteq \mathsf{CFL}$ auch echt. Also ist nur noch die Inklusion $\mathsf{CFL} \subseteq \mathsf{CSL}$ zu zeigen. Nach obigem Satz ex. zu $L \in \mathsf{CFL}$ eine kontextfreie Grammatik $G = (V, \Sigma, P, S)$ ohne ε-Produktionen mit $L(G) = L \setminus \{\varepsilon\}$. Da G dann auch kontextsensitiv ist, folgt hieraus im Fall $\varepsilon \notin L$ unmittelbar $L(G) = L \in \mathsf{CSL}$. Im Fall $\varepsilon \in L$ erzeugt die kontextsensitive Grammatik

$$G' = (V \cup \{S'\}, \Sigma, P \cup \{S' \to S, \varepsilon\}, S')$$

die Sprache $L(G') = L$, d.h. $L \in \mathsf{CSL}$. □

Als nächstes entfernen wir sämtliche Variablenumbenennungen.

Definition 2.12 *Regeln der Form $A \to B$ heißen **Variablenumbenennungen**.*

Satz 2.13 *Zu jeder kontextfreien Grammatik G ex. eine kontextfreie Grammatik G' ohne Variablenumbenennungen mit $L(G') = L(G)$.*

Beweis Zuerst entfernen wir sukzessive alle Zyklen

$$A_1 \to A_2 \to \cdots \to A_k \to A_1,$$

indem wir diese Regeln aus P entfernen und alle übrigen Vorkommen der Variablen $A_2, \ldots, A_k$ durch A_1 ersetzen. Falls sich unter den entfernten Variablen $A_2, \ldots, A_k$ die Startvariable S befindet, sei A_1 die neue Startvariable.

Nun entfernen wir sukzessive die restlichen Variablenumbenennungen, indem wir

- eine Regel $A \to B$ wählen, so dass in P keine Variablenumbenennung $B \to C$ mit B auf der rechten Seite existiert,
- diese Regel $A \to B$ aus P entfernen und
- für jede Regel $B \to \alpha$ in P die Regel $A \to \alpha$ zu P hinzunehmen. □

Beispiel 2.14 Ausgehend von den Produktionen

$$\begin{array}{lll} P\colon S \to aY, bX, Z; & Y \to b, bS, aYY; & T \to U; \\ \quad X \to a, aS, bXX; & Z \to c, S, T, cZ; & U \to abc \end{array}$$

entfernen wir den Zyklus $S \to Z \to S$, indem wir die Regeln $S \to Z$ und $Z \to S$ entfernen und dafür die Produktionen $S \to c, T, cS$ (wegen $Z \to c, T, cZ$) hinzunehmen:

$$\begin{array}{lll} S \to aY, bX, c, T, cS; & Y \to b, bS, aYY; & T \to U; \\ X \to a, aS, bXX; & U \to abc. & \end{array}$$

Nun entfernen wir die Regel $T \to U$ und fügen die Regel $T \to abc$ (wegen $U \to abc$) hinzu:

$$\begin{array}{lll} S \to aY, bX, c, T, cS; & Y \to b, bS, aYY; & T \to abc; \\ X \to a, aS, bXX; & U \to abc. & \end{array}$$

Als nächstes entfernen wir dann auch die Regel $S \to T$ und fügen die Regel $S \to abc$ (wegen $T \to abc$) hinzu:

$$\begin{array}{lll} S \to abc, aY, bX, c, cS; & Y \to b, bS, aYY; & T \to abc; \\ X \to a, aS, bXX; & U \to abc. & \end{array}$$

Da T und U nun nirgends mehr auf der rechten Seite vorkommen, können wir die Regeln $T \to abc$ und $U \to abc$ weglassen:

$$S \to abc, aY, bX, c, cS;\ Y \to b, bS, aYY;\ X \to a, aS, bXX.$$

◁

Nach diesen Vorarbeiten ist es nun leicht, eine gegebene kontextfreie Grammatik in Chomsky-Normalform umzuwandeln.

Satz 2.15 *Zu jeder kontextfreien Sprache $L \in \mathsf{CFL}$ gibt es eine CNF-Grammatik G' mit $L(G') = L \setminus \{\varepsilon\}$.*

Beweis Aufgrund der beiden vorigen Sätze hat $L \setminus \{\varepsilon\}$ eine kontextfreie Grammatik $G = (V, \Sigma, P, S)$ ohne ε-Produktionen und ohne Variablenumbenennungen. Wir transformieren G wie folgt in eine CNF-Grammatik.

- Füge für jedes Terminalsymbol $a \in \Sigma$ eine neue Variable X_a zu V und eine neue Regel $X_a \to a$ zu P hinzu.
- Ersetze alle Vorkommen von a durch X_a, ausser wenn a alleine auf der rechten Seite einer Regel steht.
- Ersetze jede Regel $A \to B_1 \cdots B_k$, $k \geq 3$, durch die $k-1$ Regeln

  $$A \to B_1 A_1,\ A_1 \to B_2 A_2,\ \ldots,\ A_{k-3} \to B_{k-2} A_{k-2},\ A_{k-2} \to B_{k-1} B_k,$$

  wobei $A_1, \ldots, A_{k-2}$ neue Variablen sind. □

Beispiel 2.16 In der Produktionenmenge

$$P\colon\ S \to abc, aY, bX, c, cS;\quad X \to a, aS, bXX;\quad Y \to b, bS, aYY$$

ersetzen wir die Terminalsymbole a, b und c durch die Variablen A, B und C (ausser wenn sie alleine auf der rechten Seite einer Regel vorkommen) und fügen die Regeln $A \to a$, $B \to b$, $C \to c$ hinzu:

$$\begin{aligned} &S \to c, ABC, AY, BX, CS;\quad X \to a, AS, BXX;\\ &Y \to b, BS, AYY;\quad A \to a;\quad B \to b;\quad C \to c. \end{aligned}$$

Ersetze nun die Regeln $S \to ABC$, $X \to BXX$ und $Y \to AYY$ durch die Regeln $S \to AS'$, $S' \to BC$, $X \to BX'$, $X' \to XX$ und $Y \to AY'$, $Y' \to YY$:

$$\begin{aligned} &S \to c, AS', AY, BX, CS;\quad S' \to BC;\\ &X \to a, AS, BX';\quad X' \to XX;\quad Y \to b, BS, AY';\quad Y' \to YY;\\ &A \to a;\ B \to b;\ C \to c. \end{aligned}$$

◁

2.3.2 Der CYK-Algorithmus

Als erste Anwendung der Chomsky-Normalform stellen wir einen effizienten Algorithmus zur Entscheidung des Wortproblems für kontextfreie Grammatiken vor.

Satz 2.17 *Das Wortproblem für kontextfreie Grammatiken ist effizient entscheidbar.*

Beweis Sei eine Grammatik $G = (V, \Sigma, P, S)$ und ein Wort $x = x_1 \cdots x_n$ gegeben. Falls $x = \varepsilon$ ist, können wir effizient prüfen, ob $S \Rightarrow^* \varepsilon$ gilt. Andernfalls bringen wir G in Chomsky-Normalform und setzen den nach seinen Autoren Cocke, Younger und Kasami benannten CYK-Algorithmus auf das Paar (G, x) an.

Der CYK-Algorithmus bestimmt für $l = 1, \ldots, n$ die Mengen

$$V_{l,k} = \{A \in V \mid A \Rightarrow^* x_k \cdots x_{k+l-1}\},\ k = 1, \ldots, n-l+1.$$

aller Variablen, aus denen das Teilwort $x_k \cdots x_{k+l-1}$ ableitbar ist. Dann gilt offensichtlich $x \in L(G) \Leftrightarrow S \in V_{n,1}$. Für $l = 1$ ist

$$V_{1,k} = \{A \in V \mid A \to x_k\}$$

und für $l = 2, \cdots, n$ ist

$$V_{l,k} = \{A \in V \mid \exists l' < l\, \exists B \in V_{l',k}\, \exists C \in V_{l-l',k+l'} : A \to BC\}.$$

Eine Variable A gehört also genau dann zu $V_{l,k}$, falls eine Zahl $l' \in \{1, \ldots, l-1\}$ und eine Regel $A \to BC$ existieren, so dass $B \in V_{l',k}$ und $C \in V_{l-l',k+l'}$ ist. Da der Zeitaufwand für die Berechnung der Menge $V_{l,k}$ durch $O(l|G|)$ beschränkt ist, und insgesamt $n(n+1)/2$ solche Mengen zu bestimmen sind, lässt sich die Zeitkomplexität durch $O(n^3|G|)$ abschätzen

Beispiel 2.18 Betrachte die CNF-Grammatik mit den Produktionen

$$S \to AS', AY, BX, CS, c;\ \ S' \to BC;\ \ X \to AS, BX', a;\ \ X' \to XX;$$
$$Y \to BS, AY', b;\ \ Y' \to YY;\ \ A \to a;\ B \to b;\ C \to c.$$

Dann erhalten wir für das Wort $x = abb$ folgende Mengen $V_{l,k}$:

k:	1	2	3
	a	b	b
l: 1	$\{X, A\}$	$\{Y, B\}$	$\{Y, B\}$
2	$\{S\}$	$\{Y'\}$	
3	$\{Y\}$		

Wegen $S \notin V_{3,1}$ ist $x \notin L(G)$. Dagegen gehört das Wort $y = aababb$ wegen $S \in V_{6,1}$ zu $L(G)$:

	1	2	3	4	5	6
	a	a	b	a	b	b
1	$\{X, A\}$	$\{X, A\}$	$\{Y, B\}$	$\{X, A\}$	$\{Y, B\}$	$\{Y, B\}$
2	$\{X'\}$	$\{S\}$	$\{S\}$	$\{S\}$	$\{Y'\}$	
3	$\{X\}$	$\{X\}$	$\{Y\}$	$\{Y\}$		
4	$\{X'\}$	$\{S\}$	$\{Y'\}$			
5	$\{X\}$	$\{Y\}$				
6	$\{S\}$					

◁

2.4 Das Pumping-Lemma für kontextfreie Sprachen

Als zweite Anwendung der Chomsky-Normalform beweisen wir nun das Pumping-Lemma für kontextfreie Sprachen.

Satz 2.19 (Pumping-Lemma für kontextfreie Sprachen) *Zu jeder kontextfreien Sprache L gibt es eine Zahl l, so dass sich alle Wörter $z \in L$ mit $|z| \geq l$ in $z = uvwxy$ zerlegen lassen mit*

1. $vx \neq \varepsilon$,
2. $|vwx| \leq l$ *und*
3. $uv^iwx^iy \in L$ *für alle* $i \geq 0$.

Beispiel 2.20 Betrachte die Sprache $L = \{a^nb^n | n \geq 0\}$. Dann lässt sich jedes Wort $z = a^nb^n$ mit $|z| \geq 2$ pumpen: Zerlege $z = uvwxy$ mit $u = a^{n-1}$, $v = a$, $w = \varepsilon$, $x = b$ und $y = b^{n-1}$. ◁

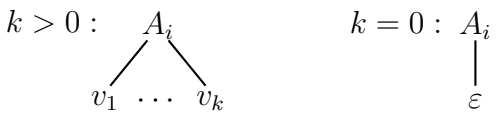
Für den Beweis des Pumping-Lemmas benötigen wir noch den Begriff des Syntaxbaums.

Definition 2.21 *Sei G eine kontextfreie Grammatik. Dann ordnen wir einer Ableitung*

$$\underline{A_0} \Rightarrow l_1\underline{A_1}r_1 \Rightarrow \cdots \Rightarrow l_{m-1}\underline{A_{m-1}}r_{m-1} \Rightarrow \alpha_m.$$

den Syntaxbaum T_m *zu, wobei die Bäume $T_0, \ldots, T_m$ induktiv wie folgt definiert sind:*

- *T_0 besteht aus einem einzigen Knoten, der mit A_0 markiert ist.*
- *Wird im $(i+1)$-ten Ableitungsschritt die Regel $A_i \to v_1 \cdots v_k$ mit $v_j \in \Sigma \cup V$ für $j = 1, \ldots, k$ angewandt, so ensteht T_{i+1} aus T_i, indem wir das Blatt A_i in T_i durch folgenden Unterbaum ersetzen:*

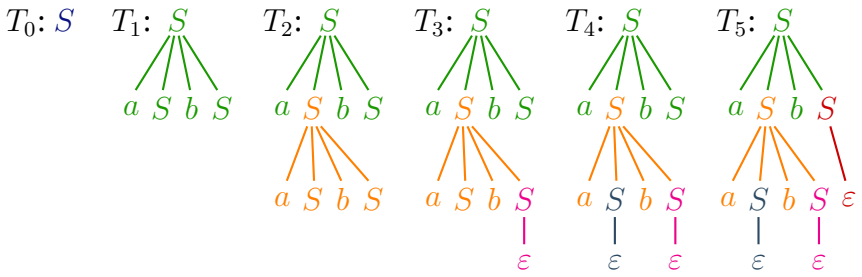


- *Hierbei stellen wir uns die Kanten von oben nach unten gerichtet und die Kinder $v_1 \cdots v_k$ von links nach rechts geordnet vor.*

Beispiel 2.22 Betrachte die Grammatik $G = (\{S\}, \{a, b, c\}, \{S \to aSbS, \varepsilon\}, S)$ und die Ableitung

$$\underline{S} \Rightarrow a\underline{S}bS \Rightarrow aaSb\underline{S}bS \Rightarrow aa\underline{S}bbS \Rightarrow aabb\underline{S} \Rightarrow aabb.$$

Die zugehörigen Syntaxbäume sind dann



$$\underline{S} \Rightarrow a\underline{S}bS \Rightarrow aaSb\underline{S}bS \Rightarrow aa\underline{S}bbS \Rightarrow aabb\underline{S} \Rightarrow aabb.$$

◁

Bemerkung 2.23

- *Zwei Ableitungen, die sich nur in der Reihenfolge der Regelanwendungen unterscheiden, führen auf denselben Syntaxbaum.*
- *Da aus einem Syntaxbaum die zugehörige Linksableitung eindeutig rekonstruierbar ist, führen unterschiedliche Linksableitungen auch auf unterschiedliche Syntaxbäume.*
- *Linksableitungen und Syntaxbäume entsprechen sich also eineindeutig.*
- *Ebenso Rechtsableitungen und Syntaxbäume.*
- *Die Syntaxbäume T einer CNF-Grammatik haben die spezielle Eigenschaft, dass jeder Knoten höchstens zwei Kinder besitzt (d.h. T ist ein Binärbaum).*

Für den Beweis des Pumping-Lemmas benötigen wir noch eine untere Schranke für die Tiefe eines Binärbaums in Abhängigkeit von der Anzahl seiner Blätter.

Lemma 2.24 *Ein Binärbaum B, der nur Pfade der Länge $\leq k$ besitzt, hat höchstens 2^k Blätter.*

Beweis Wir führen den Beweis durch Induktion über k:

$k = 0$**:** Falls B nur Pfade der Länge 0 hat, muss B aus einem einzigen Knoten bestehen.

$k \rightsquigarrow k+1$**:** Sei B ein Binärbaum, der nur Pfade der Länge $\leq k+1$ besitzt. Dann hängen an B's Wurzel maximal zwei Teilbäume. Da diese nur Pfade der Länge $\leq k$ besitzen, haben sie nach IV höchstens 2^k Blätter. Also hat B höchstens 2^{k+1} Blätter. □

Korollar 2.25 *Ein Binärbaum B mit mehr als 2^{k-1} Blättern hat mindestens Tiefe k, d.h. in B existiert ein Pfad von der Wurzel zu einem Blatt der Länge $\geq k$.*

Beweis Würde B nur Pfade der Länge $\leq k-1$, aber mehr als 2^{k-1} Blätter besitzen, so würde dies im Widerspruch zu Lemma 2.24 stehen.