Ein Skriptum zur Vorlesung

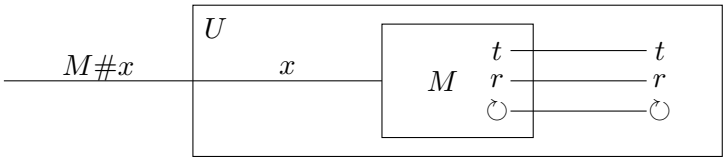
Diskrete Mathematik

für Informatiker

5. Auflage

Georg Moser
basierend auf den Unterlagen von Arne Dür und Harald Zankl

25. März 2016



© die Autoren Innsbruck, Österreich

This document was typeset using LaTeX together with KOMA-Script.

Vorwort

Die *Inhalte* der Lehrveranstaltung „Diskrete Mathematik“ sind

- Beweismethoden
- Relationen und Ordnungen
- Wachstum von Funktionen
- Graphentheorie
- Elementare Wahrscheinlichkeits-, Zähl- und Zahlentheorie
- Reguläre Sprachen (Endliche Automaten, Reguläre Ausdrücke)
- Berechenbarkeit (Turing-Maschinen)
- Grundbegriffe der Komplexitätstheorie

Die *Ziele* dieser Lehrveranstaltung sind

- der Aufbau eines Grundwissens über formale Techniken sowie
- die Vermittlung eines methodischen Vorgehens beim Lösen von Problemen.

Die Umsetzung dieser Techniken in der Softwareentwicklung wird etwa in der Schwesterlehrveranstaltung „Algorithmen und Datenstrukturen“ vertieft werden.

Das vorliegende Skriptum soll den Hörerinnen und Hörern der Vorlesung das Mitschreiben erleichtern und Zeit zum *Mitdenken* schaffen. In der Vorlesung werden die Definitionen motiviert, die wesentlichen Ergebnisse ausführlich erläutert und Rechenverfahren in konkreten Beispielen vorgeführt. Daher ist dieses Skriptum als Grundlage, aber nicht als Ersatz für eine Vorlesungsmitschrift zu verstehen.

Dieses Skriptum wurde basierend auf „Diskrete Mathematik, 4. Auflage, 2014“ von Harald Zankl neu überarbeitet. Ich möchte mich bei den Herrn Zankl für die Bereitstellung der LaTeX Quellen herzlich bedanken.

Um die Lesbarkeit zu erleichtern, wird in der direkten Anrede des Lesers, der Leserin prinzipiell die weibliche Form gewählt.

Inhaltsverzeichnis

1 Beweismethoden **1**
1.1 Motivation 1
1.1.1 Wozu exakte Definitionen? 1
1.1.2 Wozu Beweise? 1
1.2 Beweisformen 2
1.2.1 Deduktive Beweise 2
1.2.2 Beweise von Mengeninklusionen 3
1.2.3 Kontraposition 4
1.2.4 Widerspruchsbeweise (indirekte Beweise) 4
1.2.5 Induktive Beweise 5
1.2.6 Widerlegung durch ein Gegenbeispiel 5
1.3 Aufgaben 6

2 Relationen und Ordnungen **7**
2.1 Äquivalenzrelationen 8
2.2 Partielle Ordnungen 10
2.3 Das Wortmonoid 14
2.4 Aufgaben 17

3 Induktion **21**
3.1 Vollständige Induktion 21
3.2 Wohlfundierte Induktion 22
3.3 Strukturelle Induktion 24
3.4 Aufgaben 26

4 Wachstum von Funktionen **27**
4.1 Asymptotisches Wachstum 27
4.2 Aufgaben 30

5 Graphentheorie **31**
5.1 Gerichtete Graphen 31
5.2 Ungerichtete Graphen 41
5.3 Aufgaben 50

6 Zähltheorie **55**
6.1 Aufzählen und Nummerieren von Objekten 55

6.2 Abzählbarkeit von Mengen 61
6.3 Diskrete Wahrscheinlichkeitstheorie 67
6.4 Lösen von Rekursionsformeln 70
6.5 Aufgaben 74

7 Zahlentheorie 77
7.1 Rechnen mit ganzen Zahlen 77
7.2 Der euklidische Algorithmus 79
7.3 Primzahlen 84
7.4 Restklassen 86
7.5 Aufgaben 90

8 Reguläre Sprachen 91
8.1 Deterministische endliche Automaten 91
8.1.1 Minimierung 93
8.2 Nichtdeterministische endliche Automaten 96
8.2.1 Teilmengenkonstruktion 97
8.3 Endliche Automaten mit Epsilon-Übergängen 100
8.4 Reguläre Ausdrücke 103
8.4.1 Endliche Automaten und reguläre Ausdrücke 106
8.5 Abgeschlossenheit regulärer Sprachen 109
8.6 Schleifen-Lemma 110
8.7 Aufgaben 113

9 Berechenbarkeit 117
9.1 Deterministische Turingmaschinen 117
9.2 Nichtdeterministische Turingmaschinen 122
9.3 Unentscheidbarkeit 124
9.3.1 Universelle Turingmaschine 125
9.3.2 Diagonalisierung 126
9.3.3 Reduktion 129
9.4 Äquivalente Formulierungen 130
9.5 Aufgaben 134

10 Komplexitätstheorie 137
10.1 Einführung in die Komplexitätstheorie 137
10.2 Laufzeitkomplexität 140
10.2.1 Die Klassen P und NP 141
10.2.2 Polynomielle Reduktion 142
10.3 Speicherplatzkomplexität 144
10.3.1 Logarithmisch Platzbeschränkte Reduktion 144
10.4 Aufgaben 147

1

Beweismethoden

1.1 Motivation

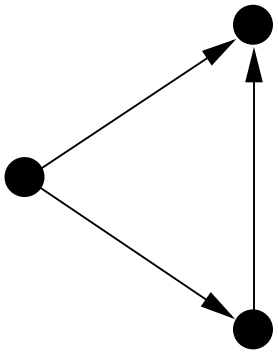
1.1.1 Wozu exakte Definitionen?

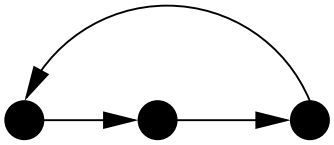
Alle Begriffe der diskreten Mathematik werden aus den Begriffen „Menge“ und „Abbildung“ abgeleitet, z.B.

Nummerierung, Ordnung, Graph, Wort .

Dem Nachteil des Aufwandes für die exakte Definition stehen folgende Vorteile gegenüber:

- Reduktion auf das Wesentliche (abstrakte Repräsentation)
- Gleichheit mit ja oder nein entscheidbar
- Programmierung naheliegend





baba

b
a
b
a

1.1.2 Wozu Beweise?

- Mit einem ausformulierten Beweis kann man sich selbst oder Kollegen überzeugen, dass richtig überlegt wurde. Oft stellt sich erst im Beweis an Hand der Argumentationskette heraus, dass gewisse Voraussetzungen fehlen oder überflüssig sind.
- Durch das Studium von Beweisen trainieren Sie das logische Denken und werden befähigt, eigene Ideen korrekt zu formulieren und durch einen Beweis zu verifizieren.

- Beweise führen oft zu programmierbaren Verfahren, weil die einzelnen Schritte im Beweis so klein sind, dass sie leicht in eine Programmiersprache übertragen werden können.
- In sicherheitskritischen Anwendungen (Auto, Flugzeug, Medizin) kann fehlerbehaftete Software Menschen gefährden. Es ist unabdingbar, bestimmte Eigenschaften von Programmen zu beweisen.

1.2 Beweisformen

1.2.1 Deduktive Beweise

Ein *deduktiver Beweis* besteht aus einer Folge von Aussagen, die von einer *Hypothese* zu einer *Konklusion* führen. Jeder Beweisschritt muss sich nach einer akzeptierten logischen Regel aus den gegebenen Fakten oder aus vorangegangenen Aussagen ergeben. Der Aussage, dass die Folge der Beweisschritte von einer Hypothese H zu einer Konklusion K führt, entspricht der Satz:

$$\text{Wenn } H\text{, dann } K.$$

„Wenn-dann"-Sätze können auch in anderer Form auftreten.

- Wenn H gilt, folgt daraus K.
- H nur dann, wenn K.
- K, wenn H.
- H impliziert K.
- $H \to K$.

Die *Wahrheitstafel* betrachtet alle möglichen Situationen in denen H und K jeweils *wahr* (w) bzw. *falsch* (f) sind. Der Wahrheitswert der Aussage $H \to K$ ist dann wie folgt:

H	K	$H \to K$	$H \wedge K$	$H \vee K$	$\neg H$
w	w	w	w	w	f
w	f	f	f	w	f
f	w	w	f	w	w
f	f	w	f	f	w

Somit ist ein „Wenn-dann"-Satz immer wahr, wenn die Hypothese H falsch ist. Um zu zeigen, dass $H \to K$ wahr ist, kann man sich somit darauf beschränken aus der Wahrheit von H die Wahrheit von K zu zeigen. Der Vollständigkeit halber seien hier noch die anderen Verknüpfungen wie das „logische und ($\wedge$)", das „logische oder ($\vee$)" bzw. das „logische nicht ($\neg$)" angeführt. Hierbei bezeichnet das logische oder immer das *einschließende-oder* (im Vergleich zum ausschließenden *entweder-oder*).

Beispiel 1.1. Sei n eine natürliche Zahl. Die Aussage

„n ist ein Vielfaches von 9, dann ist n ein Vielfaches von 3"

ist wahr (und somit ein Satz). Um den Satz zu zeigen nehmen wir an, dass n ein Vielfaches von 9 ist. Somit gibt es eine natürliche Zahl m mit $n = 9 \cdot m$ und weiters $n = 3 \cdot (3 \cdot m)$, also ist n auch ein Vielfaches von 3.

Gelegentlich finden wir Aussagen der Form

$$F \text{ genau dann, wenn } G.$$

Alternative Formulierungen sind etwa:

- F dann und nur dann, wenn G.
- F ist äquivalent zu G.
- $F \Leftrightarrow G$.

„Genau dann, wenn“-Aussagen können bewiesen werden, indem man *zwei* Behauptungen zeigt:

- „Wenn F, dann G.“
- „Wenn G, dann F.“

Beispiel 1.2. Sei n eine natürliche Zahl. Dann gilt:

$$n \text{ ist gerade} \Leftrightarrow n + 1 \text{ ist ungerade}$$

Um obige Äquivalenz zu zeigen, muss man die folgenden zwei Sätze beweisen:

- Wenn n gerade, dann ist $n + 1$ ungerade.
- Wenn $n + 1$ ungerade, dann ist n gerade.

Wir überlassen die Details der Leserin.

1.2.2 Beweise von Mengeninklusionen

Seien A und B Mengen. Um die Teilmengeneigenschaft (Inklusion)

$$A \subseteq B$$

zu zeigen, genügt es nach der Definition, die folgende „Wenn-dann“-Aussage zu beweisen:

$$\text{Wenn } x \in A, \text{ dann } x \in B\,.$$

Die *Gleichheit von Mengen* A und B kann bewiesen werden, indem man *zwei* Behauptungen zeigt:

- Wenn $x \in A$, dann $x \in B$.
- Wenn $x \in B$, dann $x \in A$.

1.2.3 Kontraposition

Die Aussage

„Wenn H, dann K."

und ihre *Kontraposition*

„Wenn (nicht K), dann (nicht H)."

sind äquivalent, d.h. aus dem einen Satz folgt der andere und umgekehrt. Diese Äquivalenz sieht man, indem man die Wahrheitstafel aus Kapitel 1.2.1 entsprechend erweitert (hier bezeichnet $\neg A$ die Negation einer Aussage A).

H	K	$H \to K$	$\neg K$	$\neg H$	$\neg K \to \neg H$
w	w	w	f	f	w
w	f	f	w	f	f
f	w	w	f	w	w
f	f	w	w	w	w

Statt zu zeigen, dass die Aussage H die Aussage K impliziert, ist es manchmal leichter zu beweisen, dass die Negation von K die Negation von H impliziert.

Beispiel 1.3. Die Kontraposition der Aussage

„es regnet, also ist die Straße ist nass"

ist

„die Straße ist trocken, also es regnet nicht"

1.2.4 Widerspruchsbeweise (indirekte Beweise)

Um zu zeigen, dass eine Aussage A gilt, nehmen *Widerspruchsbeweise* an, dass die Negation von A gilt. Kann aus der Annahme (dass die Negation von A gilt, also, dass A falsch ist) ein Widerspruch abgeleitet werden, so muss die Annahme selbst falsch sein und somit A gelten. Ein Widerspruchsbeweis folgt dem Schema (wobei $\perp$ eine Aussage bezeichnet die immer falsch ist):

$$(\neg A \to \perp) \to A$$

Beispiel 1.4. Die Aussage

„Es gibt unendlich viele natürliche Zahlen."

ist wahr (und somit ein Satz). Um dies zu zeigen, nehmen wir die Negation des Satzes an, also

„Es gibt nur endlich viele natürliche Zahlen."

Wenn es aber nur endlich viele natürliche Zahlen gibt, wählen wir die größte davon und nennen sie n. Dann ist $n+1$ aber auch eine natürliche Zahl und größer als n. Dies führt zum Widerspruch und somit muss unsere Annahme (dass es nur endlich viele natürliche Zahlen gibt) falsch sein. Also gibt es unendlich viele natürliche Zahlen.

Widerspruchsbeweise können auch benutzt werden, um zu zeigen, dass eine Aussage H eine Aussage K impliziert. Dann ergibt sich das Schema:

$$(H \to (\neg K \to \perp)) \to (H \to K)$$

1.2.5 Induktive Beweise

Aus der Lehrveranstaltung „Lineare Algebra 1“ ist das *Prinzip der vollständigen Induktion* bekannt. Wir widmen uns dieser und weiterer Arten der Induktion in Kapitel 3.

1.2.6 Widerlegung durch ein Gegenbeispiel

Wenn Sätze *allgemeine* Aussagen behandeln, genügt es, die Aussage für bestimmte Werte zu widerlegen, um den Satz zu widerlegen. In dieser Situation haben wir dann ein *Gegenbeispiel* gefunden. Gegenbeispiele können auch verwendet werden, um allgemein gefasste Aussagen so weit einzuschränken, dass sie dann als Satz gezeigt werden können.

Beispiel 1.5. Die Aussage

„Für alle natürlichen Zahlen n gilt: $n^2 \geq 2n$.“

ist falsch (und somit kein Satz). Für den Fall $n = 1$ ist die Gleichung nicht erfüllt, was leicht nachgerechnet werden kann: $1^2 = 1 \ngeq 2 = 2 \cdot 1$. Schließt man den Fall $n = 1$ aus, so erhält man den allgemein gültigen Satz

„Für alle natürlichen Zahlen $n > 1$ gilt: $n^2 \geq 2n$.“

welcher in Aufgabe 3.1 bewiesen wird.

1.3 Aufgaben

Aufgabe 1.1. Untersuchen Sie den Wahrheitswert der folgenden Aussagen:

1. es regnet $\to$ die Straße ist nass
2. die Straße ist nass $\to$ es regnet
3. die Erde umrundet die Sonne $\to$ die Sonne umrundet die Erde
4. die Sonne umrundet die Erde $\to$ die Erde umrundet die Sonne
5. die Sonne umrundet die Erde $\to$ die Sonne umrundet die Erde

Aufgabe 1.2. Seien A, B und C Aussagen. Untersuchen Sie den Wahrheitswert der folgenden Aussage:

$$((A \to B) \wedge (B \to C)) \to (A \to C)$$

Aufgabe 1.3. Seien A, B und C Aussagen. Sind die Aussagen

$$(A \to B) \to C \quad \text{und} \quad A \to (B \to C)$$

äquivalent?

Aufgabe 1.4. Betrachten Sie die Aussagen $\neg A \vee B$ und $A \to B$. Sind die beiden Aussagen äquivalent?

Aufgabe 1.5. Seien A, B und C Mengen. Beweisen oder widerlegen Sie die folgenden Äquivalenzen:

1. $(A \cup B) \cap C = (A \cap C) \cup (B \cap C)$
2. $(A \cap B) \cup C = (A \cap C) \cup (B \cap C)$

Aufgabe 1.6. Sei $f\colon M \to N$ eine Abbildung und seien A und B Teilmengen von M.

1. Beweisen oder widerlegen Sie:
$$f(A \cup B) = f(A) \cup f(B)\,.$$
2. Widerlegen Sie die allgemeine Gültigkeit von:
$$f(A \cap B) = f(A) \cap f(B)\,.$$
3. Beweisen Sie:
$$\text{Wenn } f \text{ injektiv ist, dann gilt } f(A \cap B) = f(A) \cap f(B)\,.$$

Aufgabe 1.7. Seien A, H und K Aussagen. Zeigen oder widerlegen Sie folgende Aussagen betreffend den indirekten Beweis (siehe Sektion 1.2.4):

1. $(\neg A \Rightarrow \bot) \Rightarrow A$ ist äquivalent zu A.
2. $(H \Rightarrow (\neg K \Rightarrow \bot))$ ist äquivalent zu $H \to K$.

2

Relationen und Ordnungen

Relationen werden verwendet um Beziehungen zwischen Objekten zu beschreiben.

Definition 2.1 (Relation)**.** Sei M eine Menge. Eine Teilmenge R von $M \times M$ heißt eine *Relation auf* M. Eine Relation R auf M heißt

- *reflexiv*, wenn für alle $x \in M \quad (x, x) \in R$ gilt,
- *irreflexiv*, wenn für kein $x \in M \quad (x, x) \in R$ gilt,
- *symmetrisch*, wenn für alle $x, y \in M$ aus $(x, y) \in R$ auch $(y, x) \in R$ folgt,
- *antisymmetrisch*, wenn für alle $x, y \in M$ aus $(x, y) \in R$ und $(y, x) \in R$ folgt, dass $x = y$ ist,
- *transitiv*, wenn für alle $x, y, z \in M$ aus $(x, y) \in R$ und $(y, z) \in R$ folgt, dass auch $(x, z) \in R$ ist.

Beispiel 2.1. Sei M die Menge aller Menschen. Die *Geschwister-relation*

$$R := \{(m, n) \mid m \text{ ist ein Geschwister von } n\} \subseteq M \times M$$

ist irreflexiv und symmetrisch, aber nicht transitiv. Sie ist nicht reflexiv und auch nicht antisymmetrisch.

Beispiel 2.2. Sei M die Menge aller Menschen. Die *Nachkommens-relation*

$$R := \{(m, n) \mid m \text{ ist ein Nachkomme von } n\} \subseteq M \times M$$

ist irreflexiv, antisymmetrisch (da die Prämisse niemals erfüllt ist) und transitiv (wenn man Nachkomme nicht als direkten Nachkomme auffasst). Sie ist aber weder reflexiv noch symmetrisch.

Beispiel 2.3. Sei

$$R := \{(x, y) \mid x \text{ ist gerade genau dann, wenn } y \text{ gerade ist}\} \subseteq \mathbb{N} \times \mathbb{N} \ .$$

Dann ist

$$\begin{aligned} R = \{&(0,0),(0,2),(0,4),\ldots,\\ &(1,1),(1,3),(1,5),\ldots,\\ &(2,0),(2,2),(2,4),\ldots,\\ &\ldots\} \end{aligned}$$

reflexiv, symmetrisch und transitiv. Die Relation R ist aber weder irreflexiv, noch antisymmetrisch.

Beispiel 2.4. Die *natürliche Ordnung* auf $\mathbb{Z}$

$$\leq_{\mathbb{Z}} := \{(x,y) \mid y - x \in \mathbb{N}\}$$

ist reflexiv, antisymmetrisch und transitiv. Sie ist weder irreflexiv noch symmetrisch.

2.1 Äquivalenzrelationen

Äquivalenzrelationen werden verwendet, um ähnliche Objekte in Klassen zusammenzufassen und damit Datenmengen zu reduzieren.

Definition 2.2 (Äquivalenzrelation)**.** Eine *Äquivalenzrelation* auf M ist eine reflexive, symmetrische und transitive Relation auf M.

Definition 2.3 (Äquivalenzklasse)**.** Sei $\sim$ eine Äquivalenzrelation auf M. Elemente $x, y \in M$ heißen *äquivalent*, wenn $(x,y) \in \sim$ ist. Man schreibt dann kurz $x \sim y$. Für $x \in M$ heißt die Menge

$$[x] := \{y \in M \mid x \sim y\}$$

die *Äquivalenzklasse von x*. Die Elemente einer Äquivalenzklasse K heißen die *Repräsentanten* von K. Ein *Repräsentantensystem* von $\sim$ ist eine Teilmenge von M, die aus jeder Äquivalenzklasse genau ein Element enthält.

Beispiel 2.5. Die Relation R aus Beispiel 2.3 ist eine Äquivalenzrelation. Die Äquivalenzklasse von 0 enthält alle geraden Zahlen, d.h. $[0] = \{0,2,4,\ldots\}$, die Äquivalenzklasse von 1 enthält alle ungeraden Zahlen, d.h. $[1] = \{1,3,5,\ldots\}$. Repräsentantensysteme von R sind z.B. $\{0,1\}$ oder $\{3,8\}$. Hingegen sind $\{0,4\}$ und $\{3,5\}$ keine Repräsentantensysteme von R.

Beispiel 2.6. Tupel $(x_1, x_2, \ldots, x_n)$ und $(y_1, y_2, \ldots, y_n)$ natürlicher Zahlen seien äquivalent, wenn sie durch Umordnen der Komponenten ineinander übergeführt werden können, d.h. wenn es eine Permutation $s \in S_n$ gibt mit

$$y_i = x_{s(i)} \qquad \text{für } i = 1, 2, \ldots, n.$$

Dann ist die Menge aller monotonen Tupel

$$\{(x_1, x_2, \ldots, x_n) \in \mathbb{N}^n \mid x_1 \leq x_2 \leq \ldots \leq x_n\}$$

ein Repräsentantensystem.

Satz 2.1 (Äquivalenz entspricht Gleichheit der Äquivalenzklassen)**.** *Sei $\sim$ eine Äquivalenzrelation auf M. Dann sind Elemente von M genau dann äquivalent, wenn ihre Äquivalenzklassen gleich sind.*

Beweis. *Wir können annehmen, dass $\sim$ eine Äquivalenzrelation ist (reflexiv, symmetrisch, transitiv) und müssen zeigen, dass $x \sim z \Leftrightarrow [x] = [z]$ für alle $x, z \in M$.*

- $\Leftarrow$: *Seien $x, z \in M$. Wenn die Äquivalenzklassen von x und z gleich sind, dann ist nach Definition der Äquivalenzklasse $x \sim z$.*

- $\Rightarrow$: *Sei umgekehrt $x \sim z$. Wir zeigen $[x] \subseteq [z]$ (die andere Inklusion ist analog). Da $\sim$ symmetrisch ist, ist auch $z \sim x$. Wenn y ein Element der Äquivalenzklasse von x ist, dann ist $x \sim y$. Da $\sim$ transitiv ist, folgt daraus $z \sim y$, also ist y auch ein Element der Äquivalenzklasse von z. Daher ist die Äquivalenzklasse von x eine Teilmenge der Äquivalenzklasse von z ($[x] \subseteq [z]$).*

□

Satz 2.2 (Abbildungen liefern Äquivalenzrelationen)**.** *Sei $f\colon M \to N$ eine beliebige Abbildung. Dann wird durch*

$$x \sim z \;:\Leftrightarrow f(x) = f(z)$$

eine Äquivalenzrelation definiert. Die Äquivalenzklassen sind die Urbildmengen

$$f^{-1}(y) = \{x \in M \mid f(x) = y\}$$

mit $y \in f(M)$.

Anschaulich vorstellen kann man sich diese Konstruktion von Äquivalenzklassen als das „Zusammenfassen aller Objekte mit dem gleichen Merkmal“.

Beweis. *Offensichtlich ist $\sim$ reflexiv, symmetrisch und transitiv. Für ein Element x aus M ist die Äquivalenzklasse von x die Menge aller Elemente aus M mit dem gleichen Bild, d.h. $[x] = f^{-1}(f(x))$.* □

Definition 2.4 (Partition)**.** Sei M eine Menge. Eine *Partition* von M ist eine Menge von paarweise disjunkten nichtleeren Teilmengen von M, deren Vereinigung ganz M ist. Diese Teilmengen nennt man dann die *Blöcke* der Partition.

Satz 2.3 (Äquivalenzrelationen und Partitionen entsprechen sich)**.** *Sei M eine Menge.*

(1) Sei P eine Partition von M. Für Elemente x und y von M sei

$$x \sim y \text{ genau dann, wenn } x \text{ und } y \text{ im gleichen Block von } P \text{ liegen.}$$

Dann ist $\sim$ eine Äquivalenzrelation auf M.

(2) Sei $\sim$ eine Äquivalenzrelation auf M. Dann ist die Menge P aller Äquivalenzklassen bezüglich $\sim$ eine Partition von M.

(3) Die Abbildungen $P \mapsto \sim$ aus (1) und $\sim \mapsto P$ aus (2) sind zueinander invers.

Beweis.
(1) Man wendet Satz 2.2 an auf die Abbildung $f\colon M \to P$, die einem Element x von M jenen Block B von P zuordnet, in dem x liegt.
(2) Da R reflexiv ist, liegt jedes Element $x \in M$ in der Äquivalenzklasse von x, also ist die Menge M die Vereinigung aller Äquivalenzklassen.
Wenn die Äquivalenzklassen von $x \in M$ und $z \in M$ nicht disjunkt sind, dann gibt es ein $y \in M$ mit

$$x \sim y \text{ und } z \sim y.$$

Da $\sim$ symmetrisch ist, gilt auch $y \sim z$. Da $\sim$ transitiv ist, folgt daraus $x \sim z$. Nach Satz 2.1 sind dann die Äquivalenzklassen von x und z gleich.
(3) Wenn man von einer Partition P ausgeht, die Äquivalenzrelation „im gleichen Block“ nimmt und dazu die Partition in Äquivalenzklassen bildet, dann erhält man die Partition P zurück. Wenn man umgekehrt von einer Äquivalenzrelation R ausgeht, die Partition in Äquivalenzklassen nimmt und dazu die Relation „im gleichen Block“ bildet, dann bekommt man nach Satz 2.1 die Äquivalenzrelation R zurück. □

Beispiel 2.7. Sei $M = \{a, b, c, d\}$ und $P := \{B_1, B_2, B_3\}$ eine Partition von M mit $B_1 = \{a, b\}$, $B_2 = \{c\}$ und $B_3 = \{d\}$. Durch die Abbildung $f\colon M \to P$ (aus dem Beweis von Satz 2.3(1)) mit den Zuordnungen $a \mapsto B_1$, $b \mapsto B_1$, $c \mapsto B_2$ und $d \mapsto B_3$ erhält man die Äquivalenzrelation

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

auf M. Startet man umgekehrt (Satz 2.3(2)) mit der Äquivalenzrelation $\sim$, dann sind die Äquivalenzklassen $[a] = \{a, b\}$, $[c] = \{c\}$, $[d] = \{d\}$ eine Partition von M.

2.2 Partielle Ordnungen

Ordnungen erlauben, Datenmengen hierarchisch zu strukturieren.

Definition 2.5 (partielle Ordnung, Vorgängerrelation)**.** Sei M eine Menge. Eine *(partielle) Ordnung* R auf M ist eine reflexive, antisymmetrische und transitive Relation auf M. In diesem Fall schreiben wir statt $(x, y) \in R$ kürzer

$$x \leq y \quad \text{(Sprechweise: „}x\text{ ist kleiner oder gleich }y\text{“)}\,.$$

Wir schreiben

$$x < y \quad \text{(Sprechweise: „}x\text{ ist kleiner als }y\text{“)}\,,$$

wenn $x \leq y$ und $x \neq y$ ist, und nennen x einen *Vorgänger* von y und y einen *Nachfolger* von x. Die Relation $<$ heißt *Vorgängerrelation* von $\leq$. Statt $x \leq y$ oder $x < y$ schreiben wir auch $y \geq x$ bzw. $y > x$. Eine Ordnung $\leq$ auf M heißt *total* (linear), wenn für je zwei verschiedene Elemente $x, y \in M$ entweder $x < y$ oder $y < x$ gilt.

Beispiel 2.8. Die natürliche Ordnung $\leq_{\mathbb{Z}}$ auf $\mathbb{Z}$ (siehe Beispiel 2.4) ist eine totale Ordnung.

Beispiel 2.9. Eine Zahl $m \in \mathbb{Z}$ *teilt* eine Zahl $p \in \mathbb{Z}$, wenn es eine Zahl $n \in \mathbb{Z}$ gibt, sodass

$$p = m \cdot n\,.$$

Die *Teilbarkeitsordnung* $\{(m,p) \mid m \text{ teilt } p\}$ ist eine partielle, aber keine totale Ordnung auf $\mathbb{N}$.

Beispiel 2.10. Sei M eine Menge und $\leq$ eine partielle Ordnung auf M. Für Tupel $x = (x_1, x_2, \ldots, x_k)$ und $y = (y_1, y_2, \ldots, y_k)$ in M^k sei

$$x \leq_{\text{komp}} y \quad \text{genau dann, wenn} \quad x_i \leq y_i \quad \text{für alle } i = 1, \ldots, k\,.$$

Die so definierte partielle Ordnung heißt die *komponentenweise Ordnung* auf M^k. Da im Allgemeinen keine Verwechslungsgefahr besteht schreiben wir oft nur $\leq$ anstatt $\leq_{\text{komp}}$.

Satz 2.4. *Wenn R eine partielle bzw. totale Ordnung auf einer Menge M ist und N eine Teilmenge von M ist, dann ist*

$$R \cap (N \times N)$$

eine partielle bzw. totale Ordnung auf N.

Beweis. Man prüft Reflexivität, Antisymmetrie und Transitivität bzw. Totalität von $R \cap (N \times N)$ nach. □

Beispiel 2.11. Aus Satz 2.4 und Beispiel 2.8 folgt, dass die natürliche Ordnung auf $\mathbb{N}$

$$\leq_{\mathbb{N}} := \leq_{\mathbb{Z}} \cap (\mathbb{N} \times \mathbb{N})$$

eine partielle bzw. totale Ordnung ist.

Definition 2.6 (Potenzmenge)**.** Sei M eine Menge. Die Menge aller Teilmengen von M

$$\mathcal{P}(M) := \{T \mid T \subseteq M\}$$

heißt die *Potenzmenge* von M. Für $k \in \mathbb{N}$ bezeichne

$$\mathcal{P}_k(M) := \{T \mid T \subseteq M \text{ und } \#(T) = k\}$$

die Menge aller k-elementigen Teilmengen von M.

Beispiel 2.12. Für eine Menge M ist die Teilmengenrelation oder *Inklusion*

$$S \subseteq T$$

eine partielle Ordnung auf $\mathcal{P}(M)$.

Definition 2.7 (Verfeinerung und Vergröberung von Partitionen)**.** Sei M eine Menge. Für Partitionen P und Q von M sei

$$P \leq Q$$

genau dann, wenn jeder Block von P Teilmenge eines Blocks von Q ist. In diesem Fall ist für jeden Block $T \in Q$ die Menge

$$\{S \in P \mid S \subseteq T\}$$

eine Partition von T. Wenn $P < Q$ ist, dann heißt P *feiner* als Q und Q *gröber* als P.

Satz 2.5. *Die Verfeinerungsordnung von Partitionen ist eine partielle Ordnung.*

Beweis. Aufgabe 2.15. □

Satz 2.6 (partielle Ordnung und Vorgängerrelation)**.** *Sei M eine Menge.*

(1) Wenn $\leq$ eine partielle Ordnung auf M ist, dann ist ihre Vorgängerrelation $<$ irreflexiv und transitiv.

(2) Wenn R eine irreflexive und transitive Relation auf M ist, dann wird durch

$$x \leq y \;:\Leftrightarrow\; x\,R\,y \text{ oder } x = y$$

eine partielle Ordnung auf M definiert.

(3) Die Abbildungen

$$\leq \mapsto < \text{ aus (1)} \quad \text{und} \quad R \mapsto \leq \text{ aus (2)}$$

sind zueinander invers.

Beweis.
(1) Nach Definition gilt $x < y$ genau dann, wenn $x \leq y$ und $x \neq y$. Somit ist $<$ irreflexiv. Um die Transitivität von $<$ zu zeigen, seien $x, y, z \in M$ mit $x < y$ und $y < z$. Aus der Transitivität von $\leq$ folgt $x \leq z$. Wir zeigen $x \neq z$ indirekt. Sei $x = z$. Dann folgt aus der Antisymmetrie von $\leq$ auch $x = y$, im Widerspruch zu $x < y$. Daher ist $x < z$.
(2) Nach Definition ist $\leq$ reflexiv. Um die Antisymmetrie von $\leq$ zu zeigen, seien $x, y \in M$ mit $x \leq y$ und $y \leq x$. Wenn $x \neq y$, dann wäre $x\,R\,y$ und $y\,R\,x$, wegen der Transitivität von R somit $x\,R\,x$, was der Irreflexivität von R widerspricht. Daher muss $x = y$ sein. Um die Transitivität von $\leq$ zu zeigen, seien $x, y, z \in M$ mit $x \leq y$ und $y \leq z$. Wenn $x = y$ oder $y = z$ ist, dann folgt $x \leq z$. Andernfalls ist $x\,R\,y$ und $y\,R\,z$, wegen der Transitivität von R somit $x\,R\,z$ und $x \leq z$.
(3) Wenn man von einer partiellen Ordnung $\leq$ ausgeht, dann bekommt man durch $x < y \vee x = y$ die partielle Ordnung $x \leq y$ zurück. Wenn man von einer irreflexiven und transitiven Relation R ausgeht, dann bekommt man durch $x \leq y \wedge x \neq y$ die Relation R zurück. □

Gemäß Satz 2.6 kann eine partielle Ordnung auch über eine irreflexive und transitive Relation definiert werden (und umgekehrt). Intuitiv erhält man aus einer partiellen Ordnung durch „Streichen der reflexiven Elemente“ ihre Vorgängerrelation. Umgekehrt erhält man aus einer irreflexiven und transitiven Ordnung durch „Hinzufügen der reflexiven Elemente“ eine partielle Ordnung.

Beispiel 2.13. Die partielle Ordnung $\{(a,a),(a,b),(a,c),(b,b),(b,c),(c,c)\}$ auf der Menge $\{a,b,c\}$ hat die Vorgängerrelation $\{(a,b),(a,c),(b,c)\}$, welche umgekehrt oben genannte partielle Ordnung induziert.

Definition 2.8 (kleinste, größte, minimale, maximale Elemente)**.** Sei $\leq$ eine partielle Ordnung auf einer Menge M. Ein Element $x \in M$ heißt

– *kleinstes Element* von M, falls x kleiner als alle anderen Elemente von M ist, d.h. für alle $y \in M$ ist $x \leq y$,

- *größtes Element* von M, falls x größer als alle anderen Elemente von M ist, d.h. für alle $y \in M$ ist $x \geq y$,
- *minimales Element* von M, falls kein anderes Element von M kleiner als x ist, d.h. es gibt kein $y \in M$ mit $y < x$,
- *maximales Element* von M, falls kein anderes Element von M größer als x ist, d.h. es gibt kein $y \in M$ mit $y > x$.

Beispiel 2.14. Für die partiellen Ordnungen

$$\begin{aligned} R_1 &= \{(a,a),(b,b),(c,c)\} \\ R_2 &= \{(a,a),(a,b),(b,b),(c,c)\} \\ R_3 &= \{(a,a),(a,b),(a,c),(b,b),(c,c)\} \\ R_4 &= \{(a,a),(a,b),(a,c),(b,b),(b,c),(c,c)\} \end{aligned}$$

über $\{a,b,c\}$ ergibt sich folgende Tabelle:

	kl. Element	gr. Element	min. Elemente	max. Elemente
R_1	–	–	$\{a,b,c\}$	$\{a,b,c\}$
R_2	–	–	$\{a,c\}$	$\{b,c\}$
R_3	a	–	$\{a\}$	$\{b,c\}$
R_4	a	c	$\{a\}$	$\{c\}$

Beispiel 2.15. Sei $\leq$ eine totale Ordnung auf M und sei $x \in M$. Dann ist x kleinstes Element von M genau dann, wenn x minimales Element von M ist. Analog ist x größtes Element von M genau dann, wenn x maximales Element von M ist.

Satz 2.7 (über kleinste, größte, minimale, maximale Elemente)**.** *Sei $\leq$ eine partielle Ordnung auf einer Menge M.*

(1) Wenn ein kleinstes Element von M existiert, dann ist es eindeutig bestimmt und das einzige minimale Element von M.

(2) Wenn ein größtes Element von M existiert, dann ist es eindeutig bestimmt und das einzige maximale Element von M.

(3) Wenn M endlich ist, dann gibt es zu jedem Element $x \in M$ ein minimales Element $w \in M$ mit $w \leq x$ und ein maximales Element $z \in M$ mit $x \leq z$.

(4) Wenn M endlich ist und nur ein minimales Element besitzt, dann ist dieses Element das kleinste Element von M.

(5) Wenn M endlich ist und nur ein maximales Element besitzt, dann ist dieses Element das größte Element von M.

Beweis.
(1) Wenn sowohl x als auch w kleinste Elemente von M sind, dann ist $w \leq x \leq w$ und somit $w = x$. Die andere Behauptung zeigen wir indirekt. Sei dazu x ein kleinstes Element von M, aber kein minimales Element von M. Weil x nicht minimal, gibt es ein $y \in M$ mit $y < x$ und somit $y \neq x$, was zusammen mit $x \leq y$ (da x ein kleinstes Element ist) $x < y$ gibt.

Transitivität von $<$ liefert $y < y$, was im Widerspruch zur Irreflexivität von $<$ steht. Somit ist x minimal. Da kleinste Elemente eindeutig sind (siehe oben), ist x das einzige minimale Element. (2) beweist man analog.
(3) Wir zeigen nur die Existenz eines minimalen Elements: Wenn x selbst minimal ist, kann man $w = x$ wählen. Andernfalls gibt es ein $x_1 \in M$ mit $x_1 < x$. Wenn x_1 nicht minimal ist, dann gibt es ein $x_2 \in M$ mit $x_2 < x_1$, usw. Da

$$x > x_1 > x_2 > \dots$$

verschiedene Elemente von M sind, erreicht man nach endlich vielen Schritten ein minimales Element x_n mit $x_n < x$.
(4) und (5) folgen aus (3) und Definition 2.8. □

2.3 Das Wortmonoid

In diesem Kapitel werden wir Ordnungen auf einzelnen Zeichen (Buchstaben) zu Ordnungen auf Zeichenketten (Wörtern) erweitern.

Definition 2.9 (Alphabet)**.** Sei Σ eine beliebige Menge, die im Folgenden ein *Alphabet* und deren Elemente *Zeichen* genannt werden. Wir verwenden üblicherweise Kleinbuchstaben vom Anfang des lateinischen Alphabets (a, b, c, ...) bzw. indizierte Kleinbuchstaben vom Ende des lateinischen Alphabets (u_i, v_i, w_i, ...), um beliebige Elemente von Σ zu notieren.

Beispiel 2.16.

- $\mathbb{B} = \{0, 1\}$ ist das *binäre* Alphabet
- $\{a, b, \dots, z\}$ ist das Alphabet aller (lateinischen) Kleinbuchstaben
- $\{\ , \texttt{!}, \texttt{"}, \texttt{\#}, \texttt{\$}, \texttt{\%}, \texttt{\&}, \dots, \texttt{\~{}}\}$ ist das Alphabet der druckbaren ASCII-Zeichen.

Satz 2.8 (Wortmonoid)**.** *Sei Σ ein Alphabet. Dann heißt ein Tupel*

$$(w_0, \dots, w_{n-1}),$$

wobei $n \in \mathbb{N}$ beliebig ist und $w_0, \dots, w_{n-1} \in \Sigma$ sind, ein Wort *(eine Zeichenkette, ein String) der Länge n über dem Alphabet Σ. Üblicherweise schreiben wir Buchstaben vom Ende des lateinischen Alphabets (..., u, v, w, x, y, z), um Wörter über Σ zu notieren. Bezeichne*

$$\Sigma^*$$

die Menge aller Wörter über dem Alphabet Σ. Für Wörter

$$v = (v_0, \dots, v_{m-1}) \in \Sigma^* \qquad \textit{und} \qquad w = (w_0, \dots, w_{n-1}) \in \Sigma^*$$

ist die Verkettung *oder* Konkatenation *definiert als das Wort*

$$vw := (v_0, \dots, v_{m-1}, w_0, \dots, w_{n-1}) \in \Sigma^* .$$

Dann gilt für Wörter $u, v, w \in \Sigma^*$ das Assoziativgesetz

$$(uv)w = u(vw)\,,$$

und das leere Wort $\epsilon = ()$ *ist das neutrale Element:*

$$w\epsilon = \epsilon w = w\,.$$

Die Menge Σ^* mit der Verkettung wird das Wortmonoid *über dem* Alphabet Σ *genannt. Für die* Längenfunktion

$$\ell\colon \Sigma^* \to \mathbb{N}\ ,\ (w_0, \ldots, w_{n-1}) \mapsto n\,,$$

gilt

$$\ell(vw) = \ell(v) + \ell(w) \quad \textit{und} \quad \ell(\epsilon) = 0\,.$$

Üblicherweise lässt man in den Wörtern die Klammern und Beistriche weg, weil keine Verwechslungsgefahr besteht. Insbesondere werden Wörter der Länge 1 wie Zeichen des Alphabets geschrieben.

Beispiel 2.17. Das Wort 01101 über dem Alphabet $\{0, 1\}$ hat Länge 5. Für die Wörter $x = 01101$, $y = 110$ und $z = 10101$ ist

$$\begin{aligned} xy &= 01101110 \\ yx &= 11001101 \\ (xy)z &= (01101110)10101 = 0110111010101 \\ x(yz) &= 01101(11010101) = 0110111010101\,. \end{aligned}$$

Die Menge $\{0, 1\}^*$ enthält alle (endlichen) Wörter, die aus Nullen und Einsen bestehen, also $\{\epsilon, 0, 1, 00, 01, 10, 11, 000, 001, \ldots\}$.

Satz 2.9 (lexikographische Ordnung)**.** *Sei $\leq$ eine totale Ordnung auf Σ. Für Wörter $v, w \in \Sigma^*$ sei*

$$v <_{lex} w\,,$$

falls ein $k \in \mathbb{N}$ mit $k \leq \ell(v)$ und $k \leq \ell(w)$ existiert, sodass

(1) $v_i = w_i$ *für* $i = 0, \ldots, k-1$ *und*

(2) $(\ell(v) = k$ *und* $\ell(w) > k)$ *oder*
$(\ell(v) > k$ *und* $\ell(w) > k$ *und* $v_k < w_k)$

ist. Dann ist $\leq_{lex}$ eine totale Ordnung auf Σ^* und heißt lexikographische Ordnung*.*

Beweis. Um zu zeigen, dass $\leq_{\text{lex}}$ eine (partielle) Ordnung ist, zeigen wir (Satz 2.6(2)) Irreflexivität und Transitivität von $<_{\text{lex}}$. Offensichtlich ist $<_{\text{lex}}$ irreflexiv. Um die Transitivität zu zeigen, seien $u, v, w \in \Sigma^*$ mit

$$u <_{lex} v \qquad \textit{und} \qquad v <_{lex} w\,.$$

Dann gibt es ein $k \in \mathbb{N}$ mit $k \leq \ell(u)$ und $k \leq \ell(v)$ und

(1) $u_i = v_i$ *für* $i = 0, \ldots, k-1$ *und*

(2) $(\ell(u) = k$ *und* $\ell(v) > k)$ *oder*
$(\ell(u) > k$ *und* $\ell(v) > k$ *und* $u_k < v_k)$

sowie ein $l \in \mathbb{N}$ *mit* $l \leq \ell(v)$ *und* $l \leq \ell(w)$ *und*

(1) $v_i = w_i$ *für* $i = 0, \ldots, l-1$ *und*

(2) $(\ell(v) = l$ *und* $\ell(w) > l)$ *oder*
$(\ell(v) > l$ *und* $\ell(w) > l$ *und* $v_l < w_l)$.

Dann gilt für $m := \min(k, l)$ *auch* $m \leq \ell(u)$ *und* $m \leq \ell(w)$ *und*

(a) $u_i = w_i$ *für* $i = 0, \ldots, m-1$ *und*

(b) $(\ell(u) = m$ *und* $\ell(w) > m)$ *oder*
$(\ell(u) > m$ *und* $\ell(w) > m$ *und* $u_m < w_m)$,

sodass $u <_{\text{lex}} w$ *folgt. Um zu zeigen, dass* $\leq_{\text{lex}}$ *total ist, seien* $v, w \in \Sigma^*$ *mit* $v \neq w$*. Dann existiert ein* $k \in \mathbb{N}$ *mit* $k \leq \ell(v)$ *und* $k \leq \ell(w)$*, sodass*

(a) $v_i = w_i$ *für* $i = 0, \ldots, k-1$ *und*

(b) $(\ell(v) = k$ *und* $\ell(w) > k)$ *oder* $(\ell(v) > k$ *und* $\ell(w) = k)$ *oder*
$(\ell(v) > k$ *und* $\ell(w) > k$ *und* $v_k \neq w_k)$

ist. Da $\leq$ *total auf* Σ *ist, gilt somit entweder* $v <_{\text{lex}} w$ *oder* $w <_{\text{lex}} v$*.* □

Beispiel 2.18. In den meisten Programmiersprachen sind die Zeichen nach dem ASCII-Code (siehe Anhang) total geordnet und die Zeichenketten nach der lexikographischen Ordnung, z.B. mit der Funktion `strcmp` von C.

Satz 2.10 (graduiert-lexikographische Ordnung auf Wörtern)**.** *Sei* $\leq$ *eine totale Ordnung auf* Σ*. Für Wörter* $v, w \in \Sigma^*$ *sei*

$$v <_{\text{gradlex}} w \ ,$$

falls entweder $\ell(v) < \ell(w)$ *oder* $(\ell(v) = \ell(w)$ *und* $v <_{\text{lex}} w)$ *ist. Dann ist* $\leq_{\text{gradlex}}$ *eine totale Ordnung auf* Σ^* *und heißt* graduiert-lexikographische Ordnung auf Wörtern*.*

Beweis. Man prüft die Irreflexivität und Transitivität von $<_{\text{gradlex}}$ *nach. Gemäß Satz 2.6(2) ist* $\leq_{\text{gradlex}}$ *dann eine partielle Ordnung. Totalität von* $\leq_{\text{gradlex}}$ *folgt aus der Totalität von* $\leq$*.* □

Definition 2.10 (formale Sprache)**.** Sei Σ ein Alphabet. Eine Teilmenge von Σ^* heißt eine *formale Sprache* über Σ.

Beispiel 2.19. Die formale Sprache der *Palindrome* über dem Alphabet $\Sigma = \{a, b\}$ ist

$$\{w_0 w_1 \ldots w_{n-1} \mid w_0 w_1 \ldots w_{n-1} = w_{n-1} w_{n-2} \ldots w_0\} = \\ \{\epsilon, a, b, aa, bb, aaa, aba, bab, bbb, aaaa, abba, baab, bbbb, \ldots\} \ .$$

In Kapitel 8 lernen wir *reguläre Sprachen* kennen. Diese Sprachen sind oftmals unendlich, können aber durch endlich viele Informationen dargestellt werden.

2.4 Aufgaben

Aufgabe 2.1. Betrachten Sie die folgenden Relationen auf $\mathbb{N}$:

- $R_1 := \{(0,0),(1,1),(2,2)\}$
- $R_2 := \{(n,m) \mid n < m\}$
- $R_3 := \{(n,m) \mid n = 2 \cdot m\}$
- $R_4 := \{(n,m) \mid n = m\}$
- $R_5 := \varnothing$

Welche Relation besitzt welche Eigenschaften? Welche Relationen sind Äquivalenzrelationen?

	reflexiv	irreflexiv	symmetrisch	antisymmetrisch	transitiv
R_1					
R_2					
R_3					
R_4					
R_5					

Aufgabe 2.2. Betrachten Sie die folgenden Relationen:

- $R_1 := \{(0,0),(1,1),(2,2)\}$ auf $\{0,1,2\}$
- $R_2 := \{(n,m) \mid n < m\}$ auf $\{0\}$
- $R_3 := \{(n,m) \mid n = 2 \cdot m\}$ auf $\{0,1,2\}$
- $R_4 := \{(0,0),(1,2),(2,1)\}$ auf $\{0,1,2\}$
- $R_5 := \varnothing$ auf $\varnothing$

Welche Relation besitzt welche Eigenschaften? Welche Relationen sind Äquivalenzrelationen?

	reflexiv	irreflexiv	symmetrisch	antisymmetrisch	transitiv
R_1					
R_2					
R_3					
R_4					
R_5					

Aufgabe 2.3. Finden Sie (falls möglich) Relationen mit den gesuchten Eigenschaften:

- Relation R_1, die weder reflexiv noch irreflexiv ist.
- Relation R_2, die weder symmetrisch noch antisymmetrisch ist.
- Relation R_3, die reflexiv und irreflexiv ist.
- Relation R_4, die symmetrisch und antisymmetrisch ist.

Aufgabe 2.4. Sei $R := \{(m,n) \mid (m \bmod 5) = (n \bmod 5)\}$ eine Relation auf $\mathbb{N}$. Zeigen Sie, dass R eine Äquivalenzrelation ist. Was sind die Äquivalenzklassen von R? Geben Sie für jede Äquivalenzklasse zwei verschiedene Repräsentanten an. Geben Sie zwei verschiedene Repräsentantensysteme von R an. Induziert R eine Partition von $\mathbb{N}$?

Hinweis: Die Operation „mod“ wird in Definition 7.5 auf Seite 86 eingeführt.

Aufgabe 2.5. Sei $M = \{0,1,2,3,4,5\}$. Geben Sie die Äquivalenzrelation auf M an, die durch die Partition $\{\{0\},\{1,2\},\{3,4,5\}\}$ von M beschrieben wird.

Aufgabe 2.6. Wie ist die Potenzmenge einer Menge M definiert? Zeigen Sie, dass für jede Menge M die Teilmengenrelation $\subseteq$ eine partielle Ordnung auf $\mathcal{P}(M)$ ist.

Aufgabe 2.7. Geben Sie die Teilbarkeitsordnung auf der Menge $\{0,1,2,\ldots,10\}$ an und bestimmen Sie minimale/maximale/kleinste/größte Elemente.

Aufgabe 2.8. Geben Sie alle Partitionen der Menge $\{a,b,c,d\}$ an und ordnen Sie diese bezüglich der Verfeinerungsordnung (siehe Definition 2.7). Bestimmen Sie minimale/maximale/kleinste/größte Elemente.

Hinweis: Die Relation wird recht groß, kann aber gut programmiert werden (siehe Aufgabe 2.9).

Aufgabe 2.9. Sei $M = \{m_0,\ldots,m_{n-1}\}$ und $P = \{B_1,\ldots,B_\ell\}$ eine Partition von M. Wir stellen P als Feld dar, wobei

$$p[i] = k\,.$$

Die Partition $\{\{a,c\},\{b\},\{d\}\}$ von $\{a,b,c,d\}$ wird z.B. als $[1,2,1,3]$ dargestellt. Wie kann P *feiner* als Q implementiert werden?

Bonus: Schreiben Sie ein Programm, welches für eine gegebene Partition alle Vorgänger bezüglich der Verfeinerungsordnung bestimmt.

Aufgabe 2.10. Geben Sie die Teilmengenrelation auf der Menge $\mathcal{P}(\{a,b,c\})$ an. Wie unterscheidet sie sich von ihrer Vorgängerrelation? Welche der beiden Relationen ist transitiv? Welche der beiden Relationen ist irreflexiv? Welche der beiden Relationen ist total? Bestimmen sie minimale, maximale, kleinste sowie größte Elemente.

Aufgabe 2.11. Ist die Einschränkung auf endliche Mengen essentiell?

1. In Satz 2.7(3)?
2. In Satz 2.7(4) bzw. Satz 2.7(5)?

Aufgabe 2.12. Betrachen Sie die natürliche Ordnung $\leq_{\mathbb{N}}$ auf $\mathbb{N}$. Ordnen Sie die Menge $\{0,1\}^3$ mit der komponentenweisen Ordnung $<_{\text{komp}}$. Ordnen Sie die Menge $\{0,1\}^*$ mit der lexikographischen Ordnung $<_{\text{lex}}$. Ordnen Sie die Menge $\{0,1\}^*$ mit der graduiert-lexikographischen Ordnung $<_{\text{gradlex}}$. Bestimmen sie jeweils minimale, maximale, kleinste sowie größte Elemente.

Hinweis: Bestimmen Sie zuerst $\leq_{\text{komp}}$ und dann davon die Vorgängerrelation $<_{\text{komp}}$.

Aufgabe 2.13. Ordnen Sie die formale Sprache der Palindrome über dem Alphabet $\{a,b\}$ mit der lexikographischen Ordnung und der graduiert-lexikographischen Ordnung an.

Aufgabe 2.14. Betrachten Sie die Teilbarkeitsrelation (Definition 2.9).

1. Zeigen Sie, dass die Teilbarkeitsrelation eine partielle Ordnung auf $\mathbb{N}$ ist.

   Hinweis: Prüfen Sie Reflexivität, Antisymmetrie und Transitivität.

2. Warum ist die Teilbarkeitsrelation auf $\mathbb{N}$ keine totale Ordnung?

3. Ist die Teilbarkeitsrelation eine partielle Ordnung auf $\mathbb{Z}$?

Aufgabe 2.15. Zeigen Sie, dass die Verfeinerungsordnung (Definition 2.7) eine partielle Ordnung ist. Ist sie total?

Aufgabe 2.16. Sei $\leq$ eine partielle Ordnung auf M und $<$ ihre Vorgängerrelation.

1. Kann man aus den Paaren in $\leq$ die Grundmenge M bestimmen?

2. Kann man aus den Paaren in $<$ die Grundmenge M bestimmen?

3

Induktion

3.1 Vollständige Induktion

Sei m eine fest gewählte natürliche Zahl, z.B. $m = 0$ oder $m = 1$. Eine Aussage $A(n)$ soll für alle natürlichen Zahlen $n \geq m$ gezeigt werden. In diesem Fall gehen wir wie folgt vor:

- Induktionsbasis: Wir zeigen, dass A für den Basiswert m gilt.
- Induktionsschritt: Wir zeigen, dass für alle $n \geq m$ aus $A(n)$ auch $A(n+1)$ folgt.

Nach Satz 16 von [9] gilt dann $A(n)$ für alle $n \geq m$.

Obiges Prinzip kann präzise so beschrieben werden:

$$(A(m) \wedge \forall n \geq m.(A(n) \Rightarrow A(n+1))) \Rightarrow (\forall n \geq m.A(n))$$

In einigen Beispielen sind folgende *Erweiterungen* nützlich:

- Es gibt mehrere Basiswerte

  $$A(m), A(m+1), \ldots, A(l),$$

  und wir setzen im Beweis des Induktionsschritts $n \geq l$ voraus. Da man $A(m)$, $A(m+1)$, $\ldots$, $A(l)$ eigens zeigt und damit $A(n)$ impliziert $A(n+1)$ für $n = m, m+1, \ldots, l-1$ bewiesen ist, folgt diese Erweiterung aus der Urform durch Umgruppieren.

  Obiges Prinzip kann präzise so beschrieben werden:

  $$(A(m) \wedge \ldots \wedge A(l) \wedge \forall n \geq l.(A(m) \wedge \ldots \wedge A(n) \Rightarrow A(n+1))) \Rightarrow (\forall n \geq m.A(n))$$

- Um $A(n+1)$ zu beweisen, können als Hypothesen alle Aussagen

  $$A(m), A(m+1), \ldots, A(n)$$

  verwendet werden. Diese Erweiterung folgt aus der Urform durch Wechsel von den Aussagen $A(n)$ zu den Aussagen

  $$B(n) := A(m) \wedge A(m+1) \wedge \ldots \wedge A(n),$$

  weil $B(m) = A(m)$ ist und $B(n)$ impliziert $B(n+1)$ äquivalent zu

$$A(m) \wedge A(m+1) \wedge \ldots \wedge A(n) \quad \text{impliziert} \quad A(n+1)$$

ist.

Diese Erweiterungen des Prinzips der vollständigen Induktion stellen Erweiterungen in der Anwendbarkeit des Prinzips dar, fügen aber der Beweisstärke des Prinzips nichts hinzu. Eine echte Erweiterung der Beweiskraft bringt die wohlfundierte Induktion.

3.2 Wohlfundierte Induktion

Definition 3.1 (wohlfundierte Ordnung)**.** Sei $\leq$ eine partielle Ordnung auf einer Menge M. Eine Folge $(x_0, x_1, x_2, \ldots)$ von Elementen in M heißt eine *unendliche absteigende Kette*, falls

$$x_0 > x_1 > x_2 > \ldots .$$

Man nennt $\leq$ *wohlfundiert*, wenn es in M keine unendliche absteigende Kette gibt.

Beispiel 3.1. Die natürliche Ordnung auf $\mathbb{N}$ ist wohlfundiert, da es (zwar beliebig lange, aber) keine unendliche absteigende Kette gibt:

$$\ldots >_{\mathbb{N}} 3 >_{\mathbb{N}} 2 > 1_{\mathbb{N}} > 0_{\mathbb{N}}$$

Die natürliche Ordnung auf $\mathbb{Z}$ ist nicht wohlfundiert, da es folgende unendliche absteigende Kette gibt:

$$\ldots >_{\mathbb{Z}} 3 >_{\mathbb{Z}} 2 >_{\mathbb{Z}} 1 >_{\mathbb{Z}} 0 >_{\mathbb{Z}} -1 >_{\mathbb{Z}} -2 > \ldots$$

Beispiel 3.2. Für Alphabete mit mindestens zwei Buchstaben ist die lexikographische Ordnung nicht wohlfundiert, weil

$$b >_{\text{lex}} ab >_{\text{lex}} aab >_{\text{lex}} aaab >_{\text{lex}} \ldots$$

eine unendliche absteigende Kette ist. Hingegen ist die graduiert-lexikographische Ordnung auf einem endlichen Alphabet wohlfundiert.

Beispiel 3.3. Sei $f\colon M \to \mathbb{N}$ eine beliebige Abbildung. Dann wird durch

$$x < y \quad :\Leftrightarrow \quad f(x) <_{\mathbb{N}} f(y)$$

eine wohlfundierte Ordnung auf M definiert. In induktiven Beweisen über Wörter kann man deshalb z.B. Induktion über die Länge der Wörter anwenden.

Satz 3.1 (Existenz minimaler Elemente)**.** *Sei $\leq$ eine partielle Ordnung auf einer Menge M. Dann ist $\leq$ wohlfundiert genau dann, wenn jede nichtleere Teilmenge von M ein minimales Element besitzt.*

Beweis. Sei $\leq$ wohlfundiert und sei N eine nichtleere Teilmenge von M. Dann gibt es ein Element x_0 in N. Wenn x_0 minimal in N ist, ist man fertig. Andernfalls gibt es ein Element $x_1 \in N$ mit $x_1 < x_0$. Wenn x_1 nicht minimal ist, dann gibt es ein $x_2 \in N$ mit $x_2 < x_1$, usw. Wegen

$$x_0 > x_1 > x_2 > \ldots$$

erreicht man nach endlich vielen Schritten ein minimales Element x_n.

Um die umgekehrte Richtung zu beweisen, nehmen wir an, M sei nicht wohlfundiert. Dann gibt es eine unendliche absteigende Kette

$$x_0 > x_1 > x_2 > \dots ,$$

und die nichtleere Teilmenge $N = \{x_0, x_1, x_2, \dots\}$ hat kein minimales Element. □

Satz 3.2 (Grundlage der wohlfundierten Induktion)**.** *Sei $\leq$ eine wohlfundierte Ordnung auf einer Menge M, und sei W eine Teilmenge von M mit folgenden zwei Eigenschaften:*

- *W enthält alle minimalen Elemente von M.*
- *Wenn für ein nicht-minimales Element x in M alle Vorgänger in W liegen, dann liegt auch x in W.*

Dann ist $W = M$.

Beweis. Wir führen einen Widerspruchsbeweis und nehmen an, dass $W \neq M$ ist. Dann ist die Menge

$$N := \{x \in M \mid x \notin W\}$$

nichtleer und hat nach Satz 3.1 ein minimales Element y. Nach der ersten Eigenschaft von W ist y kein minimales Element von M. Da alle Vorgänger von y in W liegen, folgt nach der zweiten Eigenschaft von W der Widerspruch $y \in W$. □

Folgerung (wohlfundierte Induktion)**.** *Sei $\leq$ eine wohlfundierte Ordnung auf einer Menge M. Eine Aussage $A(x)$ soll für alle Elemente x in M gezeigt werden. In diesem Fall gehen wir wie folgt vor:*

- Induktionsbasis*: Wir zeigen, dass $A(m)$ wahr ist für alle minimalen Elemente m von M.*
- Induktionsschritt*: Sei x ein nicht-minimales Element von M, und sei $A(y)$ wahr für alle Vorgänger y von x. Wir zeigen, dass auch $A(x)$ wahr ist.*

Nach Satz 3.2 ist die Menge W aller Elemente x, für die $A(x)$ wahr ist, ganz M.

Beispiel 3.4. Sei P die formale Sprache der Palindrome über dem Alphabet $\{a, b\}$. Die Aussage

„Wenn $x \in P$ und $\ell(x)$ gerade, dann hat x eine gerade Anzahl von as.“

kann mit wohlfundierter Induktion bewiesen werden.

Satz 3.3 ($\leq_{\text{lex}}$ auf $\mathbb{N}^k$ wohlfundiert)**.** *Sei $\mathbb{N}$ mit der natürlichen Ordnung versehen. Dann ist die lexikographische Ordnung auf der Menge $\mathbb{N}^k$ wohlfundiert.*

Beweis. Wir führen eine (vollständige) Induktion über k. Im Basisfall ist $k = 0$ und $\mathbb{N}^0 = \{\epsilon\}$ und somit $\leq_{lex}$ auf $\mathbb{N}^0$ wohlfundiert. Den Induktionsschritt zeigen wir indirekt. Sei

$$x_1 >_{lex} x_2 >_{lex} \dots$$

eine unendliche absteigende Kette in $\mathbb{N}^{k+1}$. Weil die natürliche Ordnung auf $\mathbb{N}$ wohlfundiert ist, hat die Menge der ersten Komponenten der x_i ein kleinstes Element $m = (x_n)_1$. Dann ist

$$x_n >_{lex} x_{n+1} >_{lex} \dots$$

eine unendliche absteigende Kette in $\mathbb{N}^{k+1}$ mit konstanter erster Komponente m was der Wohlfundiertheit von $\mathbb{N}^k$ widerspricht. □

Beispiel 3.5. Die lexikographische Ordnung auf der Menge $\mathbb{N}^2$ ist wohlfundiert, obwohl z.B. das Paar $(1, 0)$ unendlich viele Vorgänger $(0, n)$ hat. Daher bricht die Rekursion

$$f(n,m) := \begin{cases} m+1 & \text{falls } n = 0 \\ f(n-1,1) & \text{falls } n > 0 \text{ und } m = 0 \\ f(n-1, f(n, m-1)) & \text{sonst} \end{cases}$$

nach endlich vielen Schritten ab. Die Abbildung

$$Ack(n) := f(n,n)$$

heißt *Ackermannfunktion* und wächst sehr schnell.

3.3 Strukturelle Induktion

In der theoretischen Informatik ist man weniger an Induktion über natürlichen Zahlen interessiert, sondern mehr an Induktion über Strukturen (z.B. Aussagen, arithmetische Ausdrücke, Bäume).

Definition 3.2 (induktive Definition)**.** Eine Menge M kann induktiv definiert werden durch:

- Induktionsbasis: Man gibt ein oder mehr Elemente von M an.
- Induktionsschritt: Man spezifiziert, wie man neue Elemente von M aus den vorliegenden Elementen von M bekommt.

Die Menge M besteht dann aus genau jenen Elementen, die man durch Induktionsbasis und ein- oder mehrmalige Anwendung des Induktionsschritts erhält.

Beispiel 3.6 (Syntax der Aussagenlogik)**.** Seien $E_1, E_2, \dots$ Aussagen, die entweder wahr oder falsch sind. Diese Aussagen werden *atomare Aussagen* genannt. Die Menge aller Aussagen ist dann als formale Sprache mit Hilfe der *logischen Symbole* „$\neg$“, „$\wedge$“ sowie „$\vee$“ und der *Trennzeichen* „(“ und „)“ induktiv definiert:

(1) Die atomaren Aussagen $E_1, E_2, \dots$ sind Aussagen.

(2) Ist A eine Aussage, so ist auch $\neg A$ eine Aussage. Die Aussage $\neg A$ heißt die *Negation* von A.

(3) Sind A und B Aussagen, so sind auch $(A \vee B)$ und $(A \wedge B)$ Aussagen. Die Aussage $(A \vee B)$ heißt die *Disjunktion* von A und B, die Aussage $(A \wedge B)$ die *Konjunktion* von A und B.

Satz 3.4 (Prinzip der strukturellen Induktion)**.** *Die Aussage $A(x)$ soll für alle Strukturen $x \in M$, die induktiv definiert sind, gezeigt werden. In diesem Fall gehen wir wie folgt vor:*

- Induktionsbasis*: Wir zeigen, dass $A(x)$ für die Basisstruktur(en) x gilt.*
- Induktionsschritt*: Wir wählen eine Struktur y, die rekursiv aus den Strukturen $y_1, y_2, \ldots, y_k$ gebildet wird. Unsere Induktionshypothese besagt, dass die Aussagen $A(y_1)$, $A(y_2)$, ..., $A(y_k)$ wahr sind. Mit Hilfe der Induktionshypothese zeigen wir nun $A(y)$.*

Beweis. Laut Beispiel 3.3 reicht es aus eine Abbildung $f\colon M \to \mathbb{N}$ zu finden. Wir definieren f, sodass $f(x)$ der maximalen Anzahl von Ableitungsschritten entspricht:

- Basis*: Für alle Basiselemente $x \in M$ sei $f(x) = 0$.*
- Schritt*: Wenn y aus Strukturen $y_1, y_2, \ldots, y_k$ gebildet wird, sei*

$$f(y) := \max\{f(y_1), f(y_2), \ldots, f(y_k)\} + 1\,.$$

Dann wird durch $x < y :\Leftrightarrow f(x) <_{\mathbb{N}} f(y)$ eine wohlfundierte Ordnung auf M definiert, und wir können die strukturelle Induktion durch eine wohlfundierte Induktion beweisen. Dazu nehmen wir an, dass Basis und Schritt der strukturellen Induktion erfüllt sind.

Nach Konstruktion sind die minimalen Elemente genau die Basiselemente von M, sodass auch die Basis der wohlfundierten Induktion erfüllt ist. Für den Schritt der wohlfundierten Induktion sei y nicht minimal mit $A(z)$ wahr für alle $z < y$. Dann wird y aus Strukturen $y_1, y_2, \ldots, y_k$ gebildet. Nach Definition von f gilt $y_i < y$ für alle i. Somit sind alle Aussagen $A(y_i)$ wahr. Nach dem Schritt der strukturellen Induktion ist auch $A(z)$ wahr. Aus der wohlfundierten Induktion folgt schließlich die Gültigkeit der Aussagen $A(x)$ für alle $x \in M$. □

Beispiel 3.7. Für die zusammengesetzte Aussage

$$A = ((E_1 \wedge E_2) \vee \neg E_2)$$

ergeben sich im Beweis von Satz 3.4 die Funktionswerte

$$f(E_1) = 0\,,\ f(E_2) = 0\,,\ f(E_1 \wedge E_2) = 1\,,\ f(\neg E_2) = 1 \quad \text{und} \quad f(A) = 2\,.$$

Abschließend präsentieren wir eine induktive Definition von Wörtern (als Alternative zu der Definition in Satz 2.8).

Definition 3.3 (Induktive Definition von Wörtern)**.** Sei Σ ein Alphabet. Wir definieren Σ^* induktiv.

- Basis: $\epsilon \in \Sigma^*$
- Schritt: Wenn $x \in \Sigma^*$ und $a \in \Sigma$, dann ist $xa \in \Sigma^*$

3.4 Aufgaben

Aufgabe 3.1. Zeigen Sie mittels vollständiger Induktion, dass für alle natürlichen Zahlen $n \geq 2$:

$$n^2 \geq 2n\,.$$

Aufgabe 3.2. Zeigen Sie mittels vollständiger Induktion, dass für alle natürlichen Zahlen $n \geq 13$:

$$4n^2 \geq 3n^2 + 5n + 100\,.$$

Aufgabe 3.3. Zeigen Sie mittels vollständiger Induktion, dass für alle natürlichen Zahlen $n \geq 0$:

$$\sum_{i=0}^{n} i = \frac{n(n+1)}{2}$$

Aufgabe 3.4. Ist $\leq_{\text{lex}}$ auf $\mathbb{N}^*$ wohlfundiert? Funktioniert der Beweis von Satz 3.3?

Aufgabe 3.5. Beweisen Sie Beispiel 3.4 mittels wohlfundierter Induktion. Wählen Sie die wohlfundierte Ordnung $\leq$ derart, dass es nur einen Basisfall gibt.

Aufgabe 3.6. Berechnen Sie: $Ack(0)$, $Ack(1)$, $Ack(2)$, $Ack(3)$, $Ack(4)$.

Aufgabe 3.7. Betrachten Sie die folgende induktive Definition von Palindromen über Σ.

- Basis: das leere Wort ϵ ist ein Palindrom.
- Basis: für jedes $e \in \Sigma$ ist e ein Palindrom.
- Schritt: Wenn w ein Palindrom ist, dann ist für jedes $e \in \Sigma$ auch ewe ein Palindrom.

Beweisen Sie mittels struktureller Induktion, dass jedes Palindrom w über $\{a, b\}$ gerader Länge eine gerade Anzahl an as hat.

Aufgabe 3.8. Beweisen Sie mittels struktureller Induktion, dass die Anzahl der öffnenden und schließenden Klammern in jeder Aussage A (siehe Beispiel 3.6) gleich ist.

Aufgabe 3.9. Sei Σ ein Alphabet. Zeigen Sie:

$$\text{„}w \text{ Palindrom} \Leftrightarrow w = \mathsf{rev}(w)\text{“}.$$

Hier ist $\mathsf{rev}(w_0 \dots w_{\ell(v)-1}) = w_{\ell(v)-1} \dots w_0$.

Hinweis: Verwenden Sie die induktive Definition von Palindromen aus Aufgabe 3.7.

Aufgabe 3.10. Sei Σ ein endliches Alphabet. Zeigen Sie: Wenn w ein Palindrom über Σ ist, dann ist ww ein Palindrom gerader Länge über Σ.
Hinweis: Verwenden Sie die Eigenschaft aus Aufgabe 3.9.

4

Wachstum von Funktionen

4.1 Asymptotisches Wachstum

Um die Größe von Datenmengen oder die Laufzeit von Algorithmen in Abhängigkeit von der Größe der Eingabe asymptotisch abzuschätzen, vergleicht man ihr Wachstum mit jenem bekannter Funktionen. Diese Funktionen haben als Definitionsbereiche eine Menge natürlicher Zahlen der Form

$$\{\ell, \ell+1, \ell+2, \ldots\}$$

mit $\ell \in \mathbb{N}$. Als Wertebereiche verwendet man die reellen Intervalle

$$[0,\infty) := \{x \in \mathbb{R} \mid x \geq 0\} \quad \text{bzw.} \quad (0,\infty) := \{x \in \mathbb{R} \mid x > 0\}.$$

Definition 4.1 (asymptotische Notation)**.** Sei $g\colon \{\ell, \ell+1, \ell+2, \ldots\} \to [0,\infty)$ mit $\ell \in \mathbb{N}$.

(1) *(Groß-O)*
Die Menge $\mathrm{O}(g)$ umfasst alle Funktionen

$$f\colon \{k, k+1, k+2, \ldots\} \to [0,\infty) \quad \text{mit} \quad k \in \mathbb{N}\,,$$

für die eine positive reelle Zahl c und eine natürliche Zahl m mit $m \geq k$ und $m \geq \ell$ existieren, sodass für alle natürlichen Zahlen n mit $n \geq m$

$$f(n) \leq c \cdot g(n)$$

gilt. In Kurzform ist $f \in \mathrm{O}(g)$, wenn für hinreichend große Argumente der Funktionswert von f durch ein konstantes Vielfaches des Funktionswerts von g *nach oben beschränkt* ist.

(2) *(Groß-Omega)*
Die Menge $\Omega(g)$ umfasst alle Funktionen

$$f\colon \{k, k+1, k+2, \ldots\} \to [0,\infty) \quad \text{mit} \quad k \in \mathbb{N}\,,$$

für die eine positive reelle Zahl c und eine natürliche Zahl m mit $m \geq k$ und $m \geq \ell$ existieren, sodass für alle natürlichen Zahlen n mit $n \geq m$

$$f(n) \geq c \cdot g(n)$$

gilt. In Kurzform ist $f \in \Omega(g)$, wenn für hinreichend große Argumente der Funktionswert von f durch ein konstantes Vielfaches des Funktionswerts von g *nach unten beschränkt* ist.

(3) *(Groß-Theta)*
Schließlich ist
$$\Theta(g) := \mathrm{O}(g) \cap \Omega(g)\,.$$

Beispiel 4.1. Seien $f\colon \mathbb{N} \to \mathbb{N}$ mit $n \mapsto 3n^2 + 5n + 100$ und $g\colon \mathbb{N} \to \mathbb{N}$ mit $n \mapsto n^2$. Dann ist $f \in \Theta(g)$.

Definition 4.2 (Infimum, Supremum)**.** Sei $\leq$ eine partielle Ordnung auf M und $S \subseteq M$.

- Dann ist $y \in M$ ein *Infimum von* S, wenn $y \leq x$ für alle $x \in S$ gilt und es kein $z \in M$ mit dieser Eigenschaft gibt, sodass $y < z$ ist.
- Dann ist $y \in M$ ein *Supremum von* S, wenn $x \leq y$ für alle $x \in S$ gilt und es kein $z \in M$ mit dieser Eigenschaft gibt, sodass $z < y$ ist.

Infima (Suprema) werden auch *größte untere Schranke* (*kleinste obere Schranke*) genannt.

Das folgende Beispiel zeigt, dass Infima/Suprema nicht immer vorhanden sind bzw. nicht eindeutig bestimmt sind.

Beispiel 4.2. Sei $\leq$ eine partielle Ordnung auf der Menge $\{1, 2, 3, 4, 5, 6\}$, die durch ihre Vorgängerrelation
$$< := \{(1,3), (1,4), (1,5), (1,6), (2,3), (2,4), (2,5), (2,6), (3,5), (4,5), (4,6)\}$$
gegeben ist. Dann hat die Menge $\{3, 4\}$ die Infima 1 und 2 und das Supremum 5. Die Menge $\{5, 6\}$ hat das Infimum 4, aber kein Supremum. Die Menge $\{4, 6\}$ hat Infimum 4 und Supremum 6.

Definition 4.3 (Limes)**.** Sei $f\colon \mathbb{N} \to [0, \infty)$ eine Abbildung. Dann ist
$$\lim_{n\to\infty} f(n) = L$$
wenn es für alle positiven reellen ε ein $m \in \mathbb{N}$ gibt, sodass $|f(n) - L| < \varepsilon$ für alle $n \geq m$. Man nennt L den *Grenzwert* bzw. *Limes* von f.

Beispiel 4.3. Seien $f\colon \mathbb{N} \to [0, \infty)$ mit $n \mapsto n^2$ und $g\colon \mathbb{N} \to [0, \infty)$ mit $n \mapsto \frac{1}{n}$. Dann ist $\lim\limits_{n\to\infty} f(n) = \infty$ und $\lim\limits_{n\to\infty} g(n) = 0$. Die Abbildung $h\colon \mathbb{N} \to [0, \infty)$ mit
$$h(n) = \begin{cases} 1 & \text{wenn } n \text{ gerade} \\ 0 & \text{wenn } n \text{ ungerade} \end{cases}$$
besitzt keinen Grenzwert.

Definition 4.4 (Limes Inferior, Limes Superiror)**.** Sei $f\colon \mathbb{N} \to [0, \infty)$. Dann ist
$$\liminf_{n\to\infty} f(n) := \lim_{n\to\infty} \left(\inf\{f(m) \mid m \geqslant n\}\right)$$
und
$$\limsup_{n\to\infty} f(n) := \lim_{n\to\infty} \left(\sup\{f(m) \mid m \geqslant n\}\right).$$

Satz 4.1 (Limes-Kriterium für O und Ω)**.** *Seien* $f\colon \{k, k+1, \ldots\} \to [0, \infty)$ *und* $g\colon \{\ell, \ell+1, \ldots\} \to (0, \infty)$ *. Dann gilt*

$$f \in \mathrm{O}(g) \;\Leftrightarrow\; \limsup_{n\to\infty} \frac{f(n)}{g(n)} < \infty$$

und

$$f \in \Omega(g) \;\Leftrightarrow\; \liminf_{n\to\infty} \frac{f(n)}{g(n)} > 0\,.$$

Beweis. Wir zeigen die erste Äquivalenz, die zweite ist analog. Wenn $f(n) \le c \cdot g(n)$ *für hinreichend große* n *gilt, dann ist* $\limsup_{n\to\infty} \frac{f(n)}{g(n)} \le c$ *.*
Wenn umgekehrt $s := \limsup_{n\to\infty} \frac{f(n)}{g(n)} < \infty$ *ist, dann gilt* $\frac{f(n)}{g(n)} \le s+1$ *für hinreichend große* n*.* □

Satz 4.2. *Sei* $f\colon \mathbb{N} \to [0, \infty)$*. Wenn* $\lim_{n\to\infty} f(n)$ *definiert ist, dann gilt*

$$\lim_{n\to\infty} f(n) = \limsup_{n\to\infty} f(n) = \liminf_{n\to\infty} f(n)\,.$$

Beweis. Übung. □

Beispiel 4.4. Sei $p(n)$ eine Polynomfunktion in n vom Grad d mit Leitkoeffizient $c > 0$. Dann ist

$$\lim_{n\to\infty} \frac{p(n)}{n^d} = \lim_{n\to\infty} \frac{c \cdot n^d + c_1 \cdot n^{d-1} + \ldots + c_d}{n^d} = \lim_{n\to\infty} (c + c_1 \cdot \frac{1}{n} + \ldots + c_d \frac{1}{n^d}) = c$$

und somit $p(n) \in \Theta(n^d)$.

Definition 4.5 (Klein-o)**.** Seien $f\colon \{k, k+1, \ldots\} \to [0, \infty)$ und $g\colon \{\ell, \ell+1, \ldots\} \to (0, \infty)$. Dann ist $f \in \mathrm{o}(g)$, wenn

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0\,,$$

d.h. asymptotisch ist f vernachlässigbar gegenüber g.

Beispiel 4.5. Es gilt $n \in \mathrm{o}(n^2)$, da

$$\lim_{n\to\infty} \frac{n}{n^2} = \lim_{n\to\infty} \frac{1}{n} = 0\,,$$

aber $n \notin \mathrm{o}(2n)$, da

$$\lim_{n\to\infty} \frac{n}{2n} = \lim_{n\to\infty} \frac{1}{2} = \frac{1}{2}\,.$$

Beispiel 4.6. Die Relation $f \in \mathrm{o}(g)$ ist irreflexiv und transitiv und kann daher suggestiv als $f < g$ geschrieben werden. In der Hierarchie bekannter Funktionen ist

$$1 < \log\log n < \log n < n < n \log n < n^2 < n^3 < 2^n < 3^n < n! < n^n\,.$$

4.2 Aufgaben

Aufgabe 4.1. Sei $\leq$ eine partielle Ordnung auf der Menge $\{1, 2, 3, 4, 5\}$, die durch ihre Vorgängerrelation

$$< = \{(1, 2), (1, 4), (1, 5), (2, 4), (2, 5), (3, 4), (3, 5), (4, 5)\}$$

gegeben ist. Bestimmen Sie die Infima und Suprema der Mengen $\{2\}, \{2, 3\}, \{3, 4\}$ und $\{2, 5\}$.

Aufgabe 4.2. Seien $f\colon \mathbb{N} \to \mathbb{N}$ mit $n \mapsto 2n^2+100n+27$ und $g\colon \mathbb{N} \to \mathbb{N}$ mit $n \mapsto \sqrt{n} + 12$. Zeigen oder widerlegen Sie: $f \in \mathrm{O}(n)$, $f \in \mathrm{O}(n^2)$, $f \in \mathrm{O}(n^3)$, $g \in \mathrm{O}(n)$, $g \in \mathrm{O}(n^2)$.

Aufgabe 4.3. Seien $f\colon \mathbb{N} \to \mathbb{N}$, $g\colon \mathbb{N} \to \mathbb{N}$ und $h\colon \mathbb{N} \to \mathbb{N}$. Zeigen oder widerlegen Sie:

$$\text{Wenn } f \in \mathrm{O}(g) \text{ und } g \in \mathrm{O}(h), \text{ dann } f \in \mathrm{O}(h)\,.$$

Aufgabe 4.4. Seien $f\colon \mathbb{N} \to \mathbb{N}$, $g\colon \mathbb{N} \to \mathbb{N}$ und $h\colon \mathbb{N} \to \mathbb{N}$. Welche der folgenden Eigenschaften gelten? Informelle Begründungen reichen aus.

- Wenn $f \in \mathrm{O}(h)$ und $g \in \mathrm{O}(h)$, dann $f + g \in \mathrm{O}(h)$.
- Wenn $f \in \mathrm{O}(h)$ und $g \in \mathrm{O}(h)$, dann $f \times g \in \mathrm{O}(h)$.
- Wenn $f \in \mathrm{O}(h)$ und $g \in \mathrm{O}(h)$, dann $f \circ g \in \mathrm{O}(h)$.
- Wenn $f \in \mathrm{O}(h)$ und $g \in \mathrm{O}(h)$, dann $g \circ f \in \mathrm{O}(h)$.

Hinweis: Hier bezeichnet $f + g\colon \mathbb{N} \to \mathbb{N}$, $n \mapsto f(n) + g(n)$, etc.

5

Graphentheorie

5.1 Gerichtete Graphen

Definition 5.1 (gerichteter Multigraph)**.** Ein *gerichteter Multigraph* G ist gegeben durch

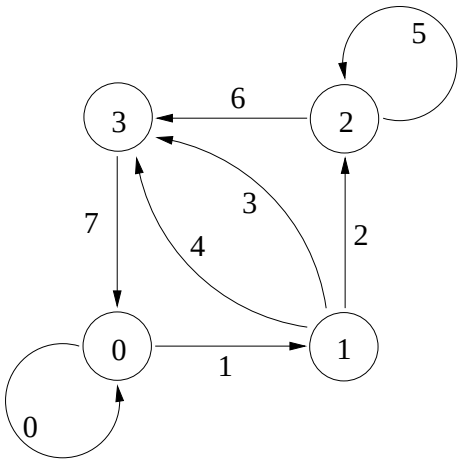
- eine *Eckenmenge* (oder *Knotenmenge*) E
- eine *Kantenmenge* K
- zwei Abbildungen
$$q\colon K \to E \quad \text{und} \quad z\colon K \to E\,,$$
die jeder Kante k ihre *Anfangsecke* $q(k)$ bzw. ihre *Endecke* $z(k)$ zuordnen (q für Quelle, z für Ziel). Man nennt dann k eine Kante von $q(k)$ nach $z(k)$.

Eine Ecke c heißt *unmittelbarer Vorgänger* der Ecke d, wenn es eine Kante von c nach d gibt. Man nennt d dann *unmittelbarer Nachfolger* von c. *Schleifen* sind Kanten mit der gleichen Anfangs- wie Endecke. Kanten mit den gleichen Anfangsecken und den gleichen Endecken heißen *parallel*. Für eine Ecke e heißt die Zahl der Kanten mit Endecke e der *Eingangsgrad* von e und die Zahl der Kanten mit Anfangsecke e der *Ausgangsgrad* von e. Wenn zusätzlich Abbildungen
$$a\colon E \to M \quad \text{oder} \quad b\colon K \to N$$
gegeben werden, dann heißt der Multigraph *ecken-* bzw. *kantenbeschriftet*, im Spezialfall $M = \mathbb{R}$ oder $N = \mathbb{R}$ *ecken-* bzw. *kantenbewertet*.

Beispiel 5.1. Sei ein gerichteter Multigraph durch die Eckenmenge $E = \{0, 1, 2, 3\}$, die Kantenmenge $K = \{0, 1, 2, \ldots, 7\}$ und die Abbildungen q und z laut folgender Tabelle gegeben:

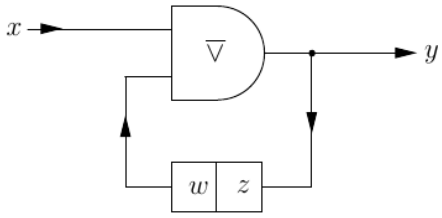
k	$q(k)$	$z(k)$
0	0	0
1	0	1
2	1	2
3	1	3
4	1	3
5	2	2
6	2	3
7	3	0

Visualisiert kann dieser Graph durch folgendes Bild werden:



Die Ecken 1 und 2 sind unmittelbare Vorgänger der Ecke 3, die Ecke 0 ist der einzige unmittelbare Nachfolger der Ecke 3. Die Kanten 0 und 5 sind Schleifen, die Kanten 3 und 4 sind parallel. Der Eingangsgrad von Ecke 3 ist drei, der Ausgangsgrad eins.

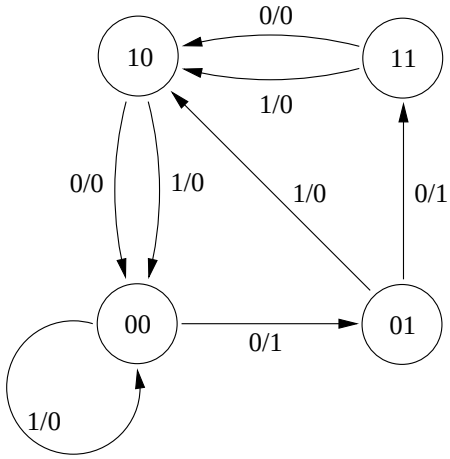
Beispiel 5.2. Im synchronen Schaltwerk mit Eingang x, Ausgang y, einem Nor-Gatter und einem Schieberegister der Länge 2



lauten die Gleichungen für die binären Signalfolgen

$$\begin{aligned} y(t) &= x(t) \,\overline{\vee}\, w(t) \\ w(t+1) &= z(t) \\ z(t+1) &= y(t)\ , \end{aligned}$$

wobei $t \in \mathbb{N}$ eine diskrete Zeit ist. Eine anschauliche Beschreibung des Schaltverhaltens liefert das Zustandsdiagramm:



Das Zustandsdiagramm ist ein gerichteter Multigraph mit Eckenmenge $E = \mathbb{B}^2$ und einer Kantenmenge $K \subseteq \mathbb{B}^6$, wobei $e = (e_0, e_1)$ den Inhalt des Schieberegisters beschreibt und

$$k = (k_0, k_1, k_2, k_3, k_4, k_5)$$

den momentanen Zustand (k_0, k_1), die Eingabe k_2, die Ausgabe k_3 und den Folgezustand (k_4, k_5) angibt. Dann sind Quelle und Ziel

$$q(k) = (k_0, k_1) \quad \text{bzw.} \quad z(k) = (k_4, k_5)\,,$$

und

$$b(k) = (k_2, k_3)$$

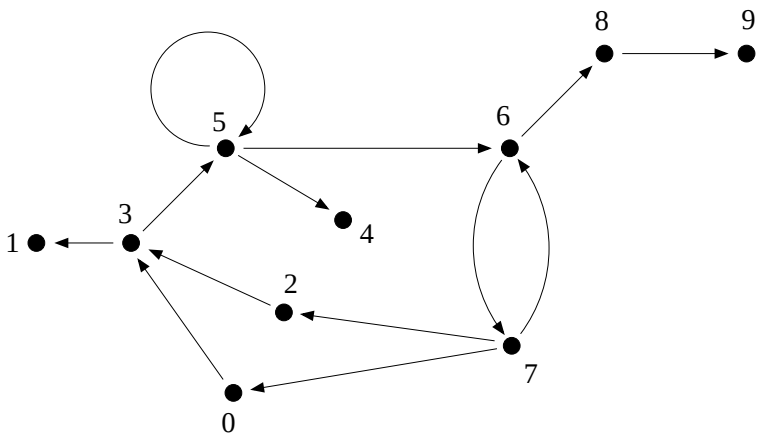
ist die Beschriftung der Kanten durch Eingabe und Ausgabe.

Definition 5.2 (gerichteter Graph)**.** Ein *gerichteter Graph* (oder *Digraph* für directed graph) ist ein gerichteter Multigraph ohne parallele Kanten. Dann gibt es zu jedem Eckenpaar (c, d) höchstens eine Kante $k \in K$ mit $q(k) = c$ und $z(k) = d$, und statt der abstrakten Kante k wird häufig das Eckenpaar (c, d) genommen.

Beispiel 5.3. Sei R eine Relation auf einer Menge M. Dann ist der *gerichtete Graph der Relation* gegeben durch

- die Eckenmenge M
- die Kantenmenge R
- die Abbildungen $q((x, y)) = x$ und $z((x, y)) = y$.

Offenbar ist jeder gerichtete Graph der Graph einer Relation.



Definition 5.3 (Teilmultigraph, Teilgraph)**.** Sei $G = (E, K, q, z)$ ein gerichteter Multigraph. Dann heißt der gerichtete Multigraph $G' = (E', K', q', z')$ *Teilmultigraph* von G, wenn $E' \subseteq E$, $K' \subseteq K$ und

$$q'(k) = q(k) \quad \text{und} \quad z'(k) = z(k) \quad \text{für alle } k \in K'\,.$$

Ein *Teilgraph* ist ein *Teilmultigraph*, der selbst Graph ist.

Definition 5.4. Sei (E, K, q, z) ein gerichteter Multigraph, und seien c, d Ecken. Ein Tupel

$$(k_0, k_1, \ldots, k_{\ell-1}) \in K^\ell$$

heißt ein *Weg* von c nach d der Länge ℓ, wenn es Ecken $e_0, e_1, \ldots, e_\ell$ gibt mit $e_0 = c$, $e_\ell = d$, und

$$q(k_i) = e_i \quad \text{sowie} \quad z(k_i) = e_{i+1} \quad \text{für } i = 0, 1, \ldots, \ell - 1\,.$$

Die Ecken $e_0, e_1, \ldots, e_\ell$ sind dann eindeutig bestimmt, und man nennt e_0 die *Anfangsecke*, e_ℓ die *Endecke* sowie $e_1, e_2, \ldots, e_{\ell-1}$ die *Zwischenecken* des Weges. Für jede Ecke $e \in E$ wird das leere Tupel $() \in K^0$ der *leere Weg* mit *Anfangsecke* e und *Endecke* e genannt.

Der gerichtete Multigraph heißt *stark zusammenhängend*, wenn es von jeder Ecke zu jeder (anderen) Ecke einen Weg gibt. Ein Weg heißt *einfach*, wenn er nichtleer ist und die Ecken

$$e_0, e_1, \ldots, e_\ell$$

paarweise verschieden sind (mögliche Ausnahme $e_0 = e_\ell$). Ist $(k_0, k_1, \ldots, k_{\ell-1})$ ein Weg von c nach d und $q = (h_0, h_1, \ldots, h_{m-1})$ ein Weg von d nach e, dann ist die *Verkettung*

$$(k_0, k_1, \ldots, k_{\ell-1}, h_0, h_1, \ldots, h_{m-1})$$

ein Weg von c nach e. Ein Weg $(k_0, k_1, \ldots, k_{\ell-1})$ heißt *geschlossen*, wenn Anfangs- und Endecke gleich sind. Ein nichtleerer geschlossener Weg mit paarweise verschiedenen Kanten wird ein *Zykel* genannt. Der gerichtete Multigraph heißt *zyklenfrei* oder *azyklisch*, wenn es keine Zyklen gibt.

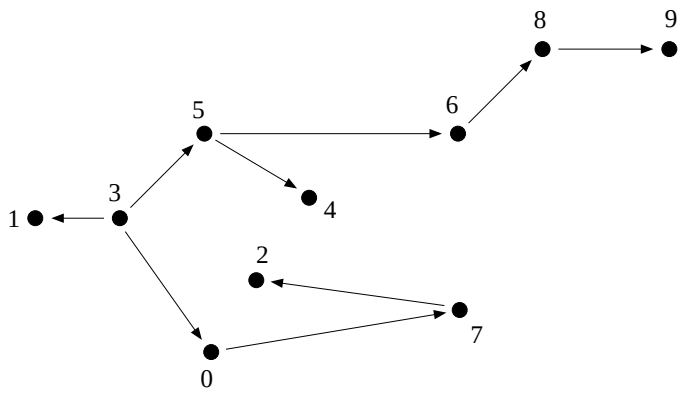
Ein *Wurzelbaum* ist ein gerichteter Graph, in dem es eine Ecke gibt, von der zu jeder Ecke genau ein Weg führt. Diese Ecke ist dann eindeutig bestimmt und heißt die *Wurzel* des Baumes. Insbesondere ist jeder Wurzelbaum zyklenfrei (siehe Aufgabe 5.3). Ecken eines Wurzelbaums vom Ausgangsgrad 0 nennt man *Blätter*.

Beispiel 5.4. Im gerichteten Multigraphen von Beispiel 5.1 sind $(1, 2, 6)$, $(1, 2, 5, 6)$, $(1, 3)$, $(1, 4)$, $(1, 3, 7, 1, 3)$, $(1, 3, 7, 1, 4)$, $(1, 4, 7, 1, 3)$, $(1, 4, 7, 1, 4)$, ... Wege von Ecke 0 nach Ecke 3. Jedoch sind $(1, 2, 6)$, $(1, 3)$ und $(1, 4)$ alle einfachen Wege von Ecke 0 zu Ecke 3. Dieser gerichtete Multigraph ist stark zusammenhängend. Die Zykel mit Anfangsecke 0 sind

$$\begin{array}{l}
(0), \\
(1,2,6,7), (0,1,2,6,7), (0,1,2,5,6,7), (1,2,5,6,7), (1,2,5,6,7,0), (1,2,6,7,0), \\
(1,3,7), \quad (0,1,3,7), (1,3,7,0), \\
(1,4,7), \quad (0,1,4,7), (1,4,7,0)\,.
\end{array}$$

Allerdings sind nur jene in der ersten Spalte einfach.

Der gerichtete Graph



ist ein Wurzelbaum (mit Wurzel 3 und Blättern $1, 2, 4, 9$).

Satz 5.1 (nichtleere Wege enthalten einfache Wege)**.** *Sei G ein gerichteter Multigraph.*

(1) Wenn es einen nichtleeren Weg p von der Ecke c zur Ecke d gibt, dann kann man aus p durch Weglassen von Kanten einen einfachen Weg von c nach d erhalten.

(2) Jeder einfache geschlossene Weg ist ein Zykel.

Beweis. (1) Seien $e_0, e_1, \ldots, e_\ell$ die Ecken des Weges $(k_0, k_1, \ldots, k_{\ell-1})$ von c nach d. Wenn es Indizes i, j mit $i < j$ und $e_i = e_j$ gibt, dann führt auch der verkürzte Weg

$$(k_0, k_1, \ldots, k_{i-1}, k_j, \ldots, k_{\ell-1})$$

von c nach d. Wenn $i > 0$ oder $j < \ell$ ist, dann ist der verkürzte Weg nichtleer. Nach endlich vielen Verkürzungen erhält man einen einfachen Weg von c nach d.
(2) Sei p ein einfacher Weg von e nach e. Wenn zwei Kanten gleich sind, dann wären auch ihre Anfangsecken gleich, was der Einfachheit von p widerspricht. □

Satz 5.2 (Adjazenzmatrix)**.** *Sei (E, K, q, z) ein gerichteter Multigraph mit endlicher Ecken- und Kantenmenge. Wir nummerieren die Ecken als $e_0, e_1, \ldots, e_{n-1}$. Dann heißt die Matrix $A \in \mathbb{Z}^{n \times n}$,*

$$A_{ij} := \#(\{k \in K \mid q(k) = e_i \text{ und } z(k) = e_j\}) \quad \text{für } i, j = 0, 1, \ldots, n-1\,,$$

die Adjazenzmatrix *(oder* Nachbarschaftsmatrix*) des Multigraphen. Für $\ell \in \mathbb{N}$ und $i, j = 0, 1, \ldots, n-1$ gibt*

$$(A^\ell)_{ij}$$

die Anzahl der Wege von e_i nach e_j der Länge ℓ an.

Beweis. Für $\ell = 0$ ist

$$A^\ell = I_n$$

und es gibt nur den leeren Weg. Für $\ell = 1$ erhalten wir die Definition. Für $\ell > 1$ ist

$$(A^\ell)_{ij} = \sum_{r=0}^{n-1} (A^{\ell-1})_{ir} \cdot A_{rj}\,.$$

Nach Induktionsannahme zählt $(A^{\ell-1})_{ir}$ alle Wege von e_i nach e_r der Länge $\ell - 1$, und A_{rj} zählt alle Kanten von e_r nach e_j. Somit zählt die Summe alle Wege von e_i nach e_j der Länge ℓ. □

Beispiel 5.5. Für den gerichteten Multigraphen aus Beispiel 5.1 erhält man als Adjazenzmatrix

$$\begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 2 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix} .$$

Satz 5.3 (transitive Hülle)**.** *Sei R eine Relation auf einer Menge M und sei G der gerichtete Graph von R. Dann ist*

$$T := \{(x, y) \in M^2 \mid \textit{es gibt in } G \textit{ einen einfachen Weg von } x \textit{ nach } y\}$$

die kleinste transitive Relation, die R enthält, und heißt transitive Hülle *von R.*

Beweis. Offensichtlich ist die Relation T transitiv und enthält die Relation R. Um zu zeigen, dass T die kleinste transitive Relation ist, die R enthält, sei S eine transitive Relation mit $R \subseteq S$. Wir zeigen $T \subseteq S$. Wenn $(x, y) \in T$ ist, dann gibt es einen einfachen Weg in G von x nach y mit Zwischenecken $z_1, z_2, \ldots, z_{\ell-1}$, somit gilt

$$(x, z_1) \in R\,,\ (z_1, z_2) \in R\,, \ldots,\ (z_{\ell-1}, y) \in R\,,$$

daher auch

$$(x, z_1) \in S\,,\ (z_1, z_2) \in S\,, \ldots,\ (z_{\ell-1}, y) \in S\,,$$

und wegen der Transitivität von S auch $(x, y) \in S$. □

Satz 5.4 (Algorithmus von Warshall)**.** *Sei R eine Relation auf einer Menge M mit n Elementen und sei A die Adjazenzmatrix von R. Der folgende Algorithmus mit $\mathrm{O}(n^3)$ Bitoperationen überschreibt A mit der Adjazenzmatrix der transitiven Hülle von R.*

> *Für r von 0 bis $n-1$ wiederhole:*
>
> *Setze $N = A$.*
>
> *Für i von 0 bis $n-1$ wiederhole:*
>
> *Für j von 0 bis $n-1$ wiederhole:*
>
> *Falls $A_{ij} = 0$ und $A_{ir} = 1$ und $A_{rj} = 1$, setze $N_{ij} = 1$.*
>
> *Setze $A = N$.*

Beweis. Für $r \in \{0, 1, \ldots, n\}$ sei P_r die Menge aller einfachen Wege im Graphen von R, die nur Zwischenecken aus der Menge $\{e_0, e_1, \ldots, e_{r-1}\}$ haben. Dann ist

- *P_0 die Menge aller Kanten von G und*
- *P_n ist die Menge aller einfachen Wege in G.*

Sei nun $r < n$. Für einen Weg p in P_{r+1} gibt es zwei Fälle:

- *e_r ist keine Zwischenecke von p. Dann ist p in P_r.*

– *e_r ist eine Zwischenecke von p. Dann kann der Weg p von e nach d als Verkettung eines Weges u von e nach e_r und eines Weges v von e_r nach d geschrieben werden, die beide in P_r liegen.*

Für $r = 0, 1, \ldots, n$ sei

$$R_r := \{(x, y) \in M^2 \mid \text{es gibt einen Weg in } P_r \text{ von } x \text{ nach } y\}\,.$$

Dann ist $R_0 = R$ und R_n ist die transitive Hülle von R. Der Algorithmus von Warshall berechnet für $r = 0, 1, \ldots, n-1$ aus der Adjazenzmatrix von R_r die Adjazenzmatrix von R_{r+1}. □

Beispiel 5.6. Für die Relation $R = \{(0,2), (1,0), (2,1)\}$ auf der Menge $\{0, 1, 2\}$ ist die transitive Hülle $T = \{(0,0), (0,1), (0,2), (1,0), (1,1), (1,2), (2,0), (2,1), (2,2)\}$.

Definition 5.5 (Länge von Wegen, Abstand von Ecken)**.** Sei G ein gerichteter Multigraph mit nichtnegativer Kantenbewertung b. Die *Länge eines Weges* $(k_0, k_1, \ldots, k_{\ell-1})$ *bezüglich* b ist die Summe der Bewertungen seiner Kanten k_i. (Somit ergibt sich bei Einheitsbewertung der Kanten die Länge ℓ.) Der *Abstand von Ecke* e zu Ecke d ist die minimale Länge eines Weges von e nach d, falls ein solcher existiert, und ∞ sonst.

Satz 5.5 (Algorithmus von Floyd)**.** *Sei G ein gerichteter Multigraph mit einer endlichen Eckenmenge E, einer endlichen Kantenmenge K und einer nichtnegativen Kantenbewertung b. Wir nummerieren die Ecken als*

$$e_0, e_1, \ldots, e_{n-1}\,.$$

Sei B die $n \times n$-Matrix mit den Einträgen

$$B_{ij} := \begin{cases} 0 & \text{falls } i = j \\ \min\{b(k) \mid k \text{ Kante von } e_i \text{ nach } e_j\} & \text{falls } i \neq j \text{ und eine Kante} \\ & \text{von } e_i \text{ nach } e_j \text{ existiert} \\ \infty & \text{sonst.} \end{cases}$$

Der folgende Algorithmus mit $\mathrm{O}(n^3)$ Rechenoperationen überschreibt die Matrix B mit der Matrix der Eckenabstände.

```
Für r von 0 bis n − 1 wiederhole:
  Setze N = B.
  Für i von 0 bis n − 1 wiederhole:
    Für j von 0 bis n − 1 wiederhole:
      Falls B_ir + B_rj < B_ij, setze N_ij = B_ir + B_rj.
  Setze B = N.
```

Beweis. *Der Abstand der Ecke e zu sich ist 0, der Abstand zu einer anderen Ecke d ist gleich der kleinsten Länge eines einfachen Weges von e nach d. Wir verwenden dieselbe Idee wie im Beweis des Algorithmus von Warshall, um die Wege im Graphen zu analysieren.*

Für $r \in \{0, 1, \ldots, n\}$ sei P_r die Menge aller einfachen Wege in G, die nur Zwischenecken aus der Menge $\{e_0, e_1, \ldots, e_{r-1}\}$ haben. Dann ist

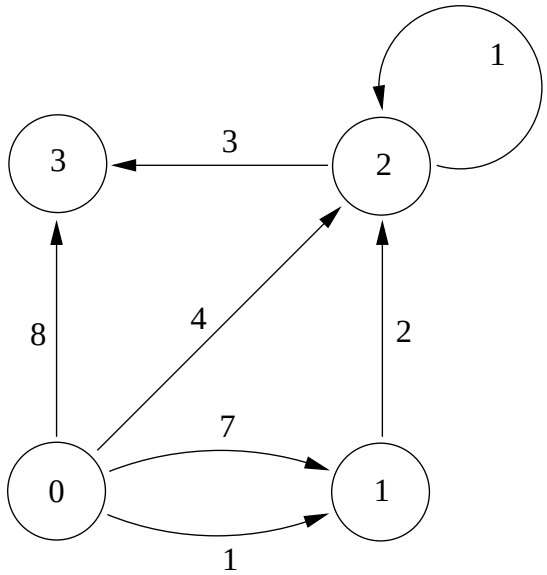
- *P_0 die Menge aller Kanten von G, und*
- *P_n ist die Menge aller einfachen Wege in G.*

Sei nun $r < n$. Für einen kürzesten Weg p von e nach d in P_{r+1} gibt es zwei Fälle:

- *e_r ist keine Zwischenecke von p. Dann ist p ein kürzester Weg von e nach d in P_r.*
- *e_r ist eine Zwischenecke von p. Dann kann der Weg p von e nach d als Verkettung eines kürzesten Weges u von e nach e_r und eines kürzesten Weges v von e_r nach d geschrieben werden, die beide in P_r liegen.*

Der Algorithmus von Floyd berechnet für $r = 0, 1, \ldots, n-1$ aus den minimalen Längen von Wegen in P_r die minimalen Längen von Wegen in P_{r+1}. □

Beispiel 5.7. Für den kantenbewerteten gerichteten Multigraphen



erhält man die Abstandsmatrix

$$\begin{pmatrix} 0 & 1 & 3 & 6 \\ \infty & 0 & 2 & 5 \\ \infty & \infty & 0 & 3 \\ \infty & \infty & \infty & 0 \end{pmatrix}.$$

Definition 5.6 (erreichbare Ecken)**.** Sei G ein gerichteter Multigraph und sei S eine Teilmenge der Eckenmenge. Eine Ecke d von G heißt *von S erreichbar*, wenn es einen Weg in G gibt mit der Endecke d und der Anfangsecke in S.

Satz 5.6 (Nachfolgersuche)**.** *Sei G ein gerichteter Multigraph mit endlicher Eckenmenge E und endlicher Kantenmenge K. Der folgende Algorithmus markiert alle von einer Startmenge S erreichbaren Ecken mit $\mathrm{O}(\#(E) \cdot \#(K))$ Operationen.*

Markiere die Ecken in S.
Solange S nichtleer ist, wiederhole:
Wähle eine Ecke e in S und entferne sie aus S.
Bestimme alle unmarkierten unmittelbaren Nachfolger von e, markiere sie und gebe sie zu S dazu.

Beweis. Da jede Ecke von G höchstens einmal aus S entfernt wird, terminiert der Algorithmus. Eine Ecke d ist genau dann von der Ecke e erreichbar, wenn d gleich e ist oder d von den unmittelbaren Nachfolgern von e erreicht werden kann. Bei jeder Iteration bleibt in Zeile 2 die Eigenschaft von Ecken,

markiert oder von S erreichbar zu sein,

gleich. Am Anfang bedeutet dies, von der Startmenge erreichbar zu sein, und am Ende, markiert zu sein. □

Beispiel 5.8. Im gerichteten Graphen aus Beispiel 5.3 sind alle Ecken von der Startecke 0 aus erreichbar.

Satz 5.7 (azyklische gerichtete Multigraphen legen Hierarchie auf Ecken fest)**.** *Sei G ein gerichteter Multigraph mit Eckenmenge E. Für Ecken e und d schreiben wir $d < e$, falls es in G einen einfachen Weg von d nach e gibt.*

Dann ist $\leq$ eine partielle Ordnung auf E genau dann, wenn G zyklenfrei ist.

Beweis. Laut Satz 2.6(1) ist $\leq$ eine partielle Ordnung, wenn ihre Vorgängerrelation $<$ irreflexiv und transitiv ist.

- $\Rightarrow$: *Für einen Beweis mittels Kontraposition besitze G einen Zykel (und somit einen nichtleeren Weg) von e nach e. Nach Satz 5.1(1) gibt es einen einfachen Weg von e nach e und somit ist $e < e$. Folglich ist $<$ nicht irreflexiv.*
- $\Leftarrow$: *Offensichtlich ist $<$ transitiv. Wenn $<$ nicht irreflexiv ist, dann enthält G einen einfachen geschlossenen Weg und nach Satz 5.1(2) auch einen Zykel.*

□

Definition 5.7 (unmittelbarer Vorgänger bzw. Nachfolger)**.** Sei $\leq$ eine partielle Ordnung auf einer Menge M, und seien x und y Elemente von M mit $x < y$. Dann heißt x ein *unmittelbarer Vorgänger* von y bzw. y ein *unmittelbarer Nachfolger* von x, in Zeichen

$$x \prec y\,,$$

wenn es kein $z \in M$ mit $x < z < y$ gibt.

Beispiel 5.9. In der Potenzmenge von M ist eine Menge S unmittelbarer Vorgänger (bezüglich der Inklusion) einer Menge T genau dann, wenn

$$T = S \cup \{x\}$$

für ein $x \in M \setminus S$ ist.

Beispiel 5.10. Eine Partition P einer Menge M ist unmittelbarer Vorgänger (bezüglich der Verfeinerungsordnung) einer Partition Q von M genau dann, wenn Q aus P durch Vereinigen zweier Blöcke von P entsteht.

Der folgende Satz zeigt, dass eine partielle Ordnung auf einer endlichen Menge durch den Graphen der unmittelbaren Vorgängerrelation visualisiert werden kann.

Satz 5.8. *Sei $\leq$ eine partielle Ordnung auf der endlichen Menge M und sei G der Graph der unmittelbaren Vorgängerrelation $\prec$ auf M. Dann gilt für $x, y \in M$*

$$x \leq y \text{ genau dann, wenn in } G \text{ ein Weg von } x \text{ nach } y \text{ existiert},$$

d.h. $<$ ist die transitive Hülle von $\prec$.

Beweis.

$\Rightarrow$: *Seien $x, y \in M$ mit $x \leq y$. Da für $x = y$ der leere Weg von x nach y führt, können wir $x < y$ annehmen. Wir zeigen nun durch Induktion nach der Zahl der Elemente des Intervalls*

$$[x, y] := \{z \in M \mid x \leq z \leq y\} ,$$

dass es Elemente $z_1, \ldots, z_n \in M$ gibt mit

$$x \prec z_1 \prec \ldots \prec z_n \prec y .$$

Dann ist $((x, z_1), (z_1, z_2), \ldots, (z_n, y))$ ein Weg in G von x nach y.
Für $\#([x, y]) = 2$ ist offenbar $x \prec y$.
Für $\#([x, y]) > 2$ gibt es ein $z \in M \setminus \{x, y\}$ mit

$$x < z < y .$$

Da die Intervalle $[x, z]$ und $[z, y]$ weniger Elemente als $[x, y]$ enthalten, existieren nach Induktionsannahme Elemente $v_1, \ldots, v_k, w_1, \ldots, w_\ell \in M$ mit

$$x \prec v_1 \prec \ldots \prec v_k \prec z \prec w_1 \prec \ldots \prec w_\ell \prec y .$$

$\Leftarrow$: *Wenn in G ein Weg von x nach y einer Länge ℓ existiert, dann ist entweder $\ell = 0$ und $x = y$, oder $\ell > 0$ und*

$$x \prec z_1 \prec z_2 \prec \ldots \prec z_{\ell-1} \prec y ,$$

sodass insgesamt $x \leq y$ folgt.

□

5.2 Ungerichtete Graphen

Definition 5.8 (ungerichteter Multigraph)**.** Ein *ungerichteter Multigraph* ist gegeben durch

(1) eine *Eckenmenge* (oder *Knotenmenge*) E

(2) eine *Kantenmenge* K

(3) eine Abbildung

$$\begin{aligned} r : K &\to \{\{c,d\} \mid c,d \in E\}\,, \\ k &\mapsto r(k)\,, \end{aligned}$$

die jeder Kante k eine Menge $r(k)$ mit einer oder zwei *Endecken* zuordnet (r für Rand). Man nennt dann k eine Kante zwischen diesen Ecken.

Eine Ecke c heißt *Nachbar* der Ecke d, wenn es eine Kante zwischen c und d gibt. Man nennt eine Kante mit nur einer Endecke eine *Schleife*. Kanten mit den gleichen Endecken heißen *parallel*. Für eine Ecke e heißt die Zahl der Kanten mit Endecke e der *Grad* von e. Wenn zusätzlich Abbildungen

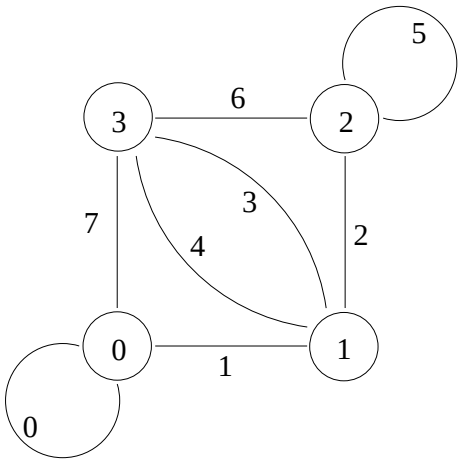
$$a : E \to M \quad \text{oder} \quad b : K \to N$$

gegeben werden, dann heißt der Multigraph *ecken-* bzw. *kantenbeschriftet*, im Spezialfall $M = \mathbb{R}$ oder $N = \mathbb{R}$ *ecken-* bzw. *kantenbewertet*.

Beispiel 5.11. Sei ein ungerichteter Multigraph mit Eckenmenge $E = \{0,1,2,3\}$, Kantenmenge $K = \{0,1,2,\ldots,7\}$ und der Abbildung r laut folgender Tabelle gegeben:

k	$r(k)$
0	{0}
1	{0,1}
2	{1,2}
3	{1,3}
4	{1,3}
5	{2}
6	{2,3}
7	{0,3}

Visualisiert kann dieser Graph durch folgendes Bild werden:

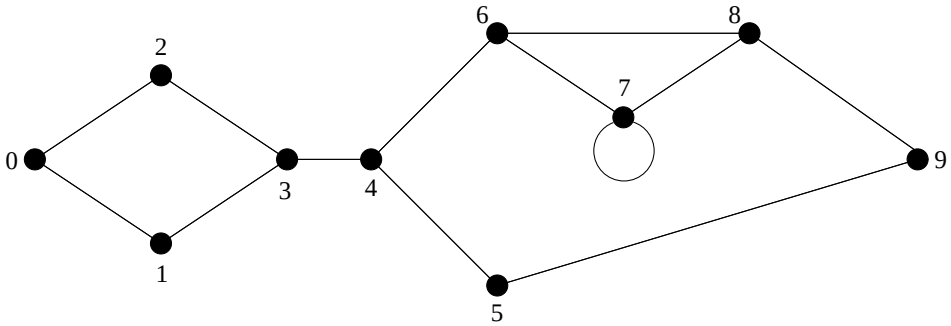


Definition 5.9 (ungerichteter Graph)**.** Ein *ungerichteter Graph* ist ein ungerichteter Multigraph ohne parallele Kanten. Dann gibt es zu jeder Eckenmenge $\{c, d\}$ höchstens eine Kante $k \in K$ mit $r(k) = \{c, d\}$.

Beispiel 5.12. Eine symmetrische Relation S auf einer Menge M kann durch den ungerichteten Graphen mit

- der Eckenmenge M
- der Kantenmenge $\{\{x, y\} \mid (x, y) \in S\}$
- der Abbildung $r(\{x, y\}) = \{x, y\}$

visualisiert werden. Offenbar ist jeder ungerichtete Graph der Graph einer symmetrischen Relation.



Wenn G der Graph einer symmetrischen Relation ist, so geben wir die Kantenmenge oft nur durch ihren Rand an, wie z.B. in Aufgabe 5.12.

Definition 5.10 (Teilmultigraph, Teilgraph)**.** Sei $G = (E, K, r)$ ein ungerichteter Multigraph. Dann heißt ein ungerichteter Multigraph $G' = (E', K', r')$ *Teilmultigraph* von G, wenn $E' \subseteq E$, $K' \subseteq K$ und

$$r'(k) = r(k) \quad \text{für alle } k \in K' .$$

Ein *Teilgraph* ist ein *Teilmultigraph*, der selbst Graph ist.

Definition 5.11 (Wege, Zusammenhang, Zyklen, Wälder, Bäume, Blätter)**.** Sei (E, K, r) ein ungerichteter Multigraph, und seien c, d Ecken. Ein Tupel

$$(k_0, k_1, \ldots, k_{\ell-1}) \in K^\ell$$

heißt ein *Weg* von c nach d der Länge ℓ, wenn es Ecken $e_0, e_1, \ldots, e_\ell$ gibt mit $e_0 = c,\ e_\ell = d$, und

$$r(k_i) = \{e_i, e_{i+1}\} \quad \text{für } i = 0, 1, \ldots, \ell - 1\,.$$

Die Ecken $e_0, e_1, \ldots, e_\ell$ sind dann eindeutig bestimmt, und man nennt e_0 die *Anfangsecke*, e_ℓ die *Endecke* sowie $e_1, e_2, \ldots, e_{\ell-1}$ die *Zwischenecken* des Weges. Für jede Ecke $e \in E$ wird das leere Tupel $() \in K^0$ der *leere Weg* mit *Anfangsecke* e und *Endecke* e genannt.

Der ungerichtete Multigraph heißt *zusammenhängend*, wenn es von jeder Ecke zu jeder (anderen) Ecke einen Weg gibt. Ein Weg heißt *einfach*, wenn er nichtleer ist und die Ecken

$$e_0, e_1, \ldots, e_\ell$$

paarweise verschieden sind bis auf die mögliche Ausnahme $e_0 = e_\ell$. Für jeden Weg

$$(k_0, k_1, \ldots, k_{\ell-2}, k_{\ell-1})\ ,$$

von c nach d ist der *reziproke* Weg

$$(k_{\ell-1}, k_{\ell-2}, \ldots, k_1, k_0)\ ,$$

ein Weg von d nach c. Sind $(k_0, k_1, \ldots, k_{\ell-1})$ ein Weg von c nach d und $(h_0, h_1, \ldots, h_{m-1})$ ein Weg von d nach e, dann ist die *Verkettung*

$$(k_0, k_1, \ldots, k_{\ell-1}, h_0, h_1, \ldots, h_{m-1})$$

ein Weg von c nach e. Ein Weg $(k_0, k_1, \ldots, k_{\ell-1}) \in K^\ell$ heißt *geschlossen*, wenn Anfangs- und Endecke gleich sind. Ein nichtleerer geschlossener Weg mit paarweise verschiedenen Kanten wird ein *Zykel* genannt. Der ungerichtete Multigraph heißt *zyklenfrei* oder *azyklisch*, wenn es keine Zyklen gibt.

Ein *Wald* ist ein zyklenfreier ungerichteter Multigraph, ein *Baum* ist ein zusammenhängender Wald. Ecken eines Waldes vom Grad ≤ 1 nennt man *Blätter*.

Beispiel 5.13. Im Multigraphen von Beispiel 5.11 sind

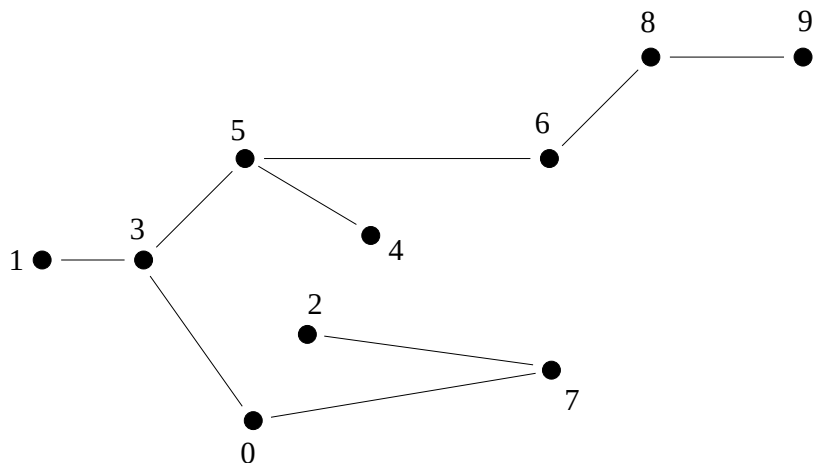
$$(1,2,6), (1,2,5,6), (1,3), (1,4), (1,3,7,1,3), (1,4,7,1,3), (7), \ldots$$

Wege von Ecke 0 nach Ecke 3. Jedoch sind $(1,2,6)$, $(1,3)$, $(1,4)$ und (7) alle einfachen Wege von Ecke 0 zu Ecke 3. Dieser Multigraph ist zusammenhängend. Die Zykel mit Anfangsecke 0 sind

$$\begin{array}{ll}
(0), & \\
(1,2,6,7), & (0,1,2,6,7), (0,1,2,5,6,7), (1,2,5,6,7), (1,2,5,6,7,0), (1,2,5,7,0), \\
(1,3,7), & (0,1,3,7), (1,3,7,0), \\
(1,4,7), & (0,1,4,7), (1,4,7,0), \\
(7,3,1), & (0,7,3,1), (7,3,1,0), \\
(7,4,1), & (0,7,4,1), (7,4,1,0), \\
(7,6,2,1), & (0,7,6,2,1), (0,7,6,5,2,1), (7,6,5,2,1), (7,6,2,1,0), (7,6,5,2,1,0)\,.
\end{array}$$

Allerdings sind nur jene in der ersten Spalte einfach.

Der Graph



ist ein zusammenhängender Wald und somit ein Baum. Die Blätter sind $1, 2, 4, 9$.

Satz 5.9 (nichtleere Wege enthalten einfache Wege)**.** *Sei G ein ungerichteter Multigraph.*

(1) Wenn es einen nichtleeren Weg p von der Ecke c zur Ecke d gibt, dann kann aus p durch Weglassen von Kanten ein einfacher Weg von c nach d gewonnen werden.

(2) Jeder einfache geschlossene Weg der Länge mindestens 3 ist ein Zykel.

(3) Aus jedem Zykel kann durch Weglassen von Kanten ein einfacher Zykel erhalten werden.

Beweis. (1) beweist man analog zu Satz 5.1.
(2) Für einen indirekten Beweis sei $p = (k_0, k_1, \ldots, k_{\ell-1})$ mit $\ell > 2$ ein einfacher Weg von e nach e, aber kein Zykel. Weil p geschlossen, aber kein Zykel ist, besitzt er zwei gleiche Kanten. Dann stimmen auch ihre Eckenmengen überein. Da p einfach ist, enthält er nur die Ecke e doppelt. Daher müssen diese Kanten erste und letzte Kante des Weges sein und aufeinander folgen. Somit ist $\ell = 2$. Widerspruch. Insbesondere folgt die Behauptung (2) des Satzes.
(3) folgt aus (1). □

Beispiel 5.14. In einem ungerichteten Multigraphen kann es Zyklen der Länge 2 geben, in einem ungerichteten Graphen nicht.

Satz 5.10 (Eindeutigkeit von einfachen Wegen in Bäumen)**.** *Sei G ein Baum. Dann existiert zu verschiedenen Ecken c und d genau ein einfacher Weg von c nach d.*

Beweis. Für einen indirekten Beweis seien G ein Baum und c und d Ecken, sodass nicht genau ein einfacher Weg von c nach d führt. Wir betrachten zwei Fälle:

- *Es gibt keinen einfachen Weg von c nach d. Dann gibt es nach (der Kontraposition von) Satz 5.9(1) auch keinen Weg von c nach d. Da G ein Baum (und damit zusammenhängend) ist, erreicht man einen Widerspruch.*

- *Es gibt mindestens zwei einfache Wege p und q von c nach d mit $p \neq q$. Wir können annehmen, dass die Wege keine gemeinsamen Kanten haben (ansonsten zerlegen wir die Wege in entsprechende Teilwege). Dann ist die Verkettung eines Weges mit dem*

reziproken Weg des anderen Weges ein einfacher geschlossener Weg mit paarweise verschiedenen Kanten und somit ein Zykel. Widerspruch (da G ein Baum und somit zyklenfrei).

□

Definition 5.12 (schwacher Zusammenhang, Orientierung)**.**

(1) Wenn $G = (E, K, q, z)$ ein gerichteter Multigraph ist, dann erhält man durch

$$r(k) := \{q(k), z(k)\} \quad \text{für } k \in K$$

einen ungerichteten Multigraphen (E, K, r), indem man die Richtung der Kanten vergisst. Man nennt G *schwach zusammenhängend*, wenn sein ungerichteter Multigraph zusammenhängend ist.

(2) Umgekehrt liegen zwei Verfahren nahe, aus einem ungerichteten Multigraphen G einen gerichteten Multigraphen zu konstruieren:

(a) Man dupliziert jede Kante von G und wählt für Original und Kopie jeweils eine andere Richtung, d.h. man interpretiert die Kanten als Doppelpfeile.

(b) Man wählt für jede Kante k von G eine Richtung aus, d.h. man setzt für $r(k) = \{c, d\}$

$$\text{entweder} \quad q(k) := c \quad \text{und} \quad z(k) := d \quad \text{oder umgekehrt}\,.$$

Der gerichtete Multigraph in (b) heißt dann eine *Orientierung* von G.

Satz 5.11 (Wurzelbaum als Baum mit ausgezeichneter Ecke)**.**

(1) Für jeden Wurzelbaum W mit Wurzel w ist der zugehörige ungerichtete Graph B ein Baum mit Ecke w.

(2) Zu jedem nichtleeren Baum B und jeder Ecke e von B gibt es genau einen Wurzelbaum mit Wurzel e, der eine Orientierung von B ist.

(3) Die Zuordnungen

$$W \mapsto (B, w)$$

von (1) und

$$(B, e) \mapsto W$$

von (2) sind zueinander invers. Daher kann man einen Wurzelbaum auch als einen Baum mit einer ausgezeichneten Ecke auffassen.

Beweis. (1) Offensichtlich ist B zusammenhängend. Wenn B einen Zykel enthält, dann gäbe es in W auch einen Zykel oder zwei verschiedene Kanten mit gleichem Endpunkt und somit zwei verschiedene Wege von der Wurzel zu dieser Ecke.
(2) Da B zyklenfrei ist, gibt es zu jeder Ecke d ungleich e genau einen einfachen Weg von e nach d. Die dadurch festgelegte Orientierung W ist ein Wurzelbaum mit der Wurzel e.
(3) Der ungerichtete Multigraph einer Orientierung ist der ursprüngliche Multigraph. Daher erhält man aus dem Wurzelbaum durch Vergessen der Richtungen den alten Baum zurück.

Wenn im neuen Wurzelbaum eine Kante anders orientiert wäre als im alten Wurzelbaum, dann gäbe es im Baum zwei verschiedene einfache Wege von der Wurzel zu einem der Endknoten. □

Definition 5.13 (Zusammenhangskomponenten)**.** Sei G ein ungerichteter Multigraph. Zwei Ecken c und d seien äquivalent, wenn es einen Weg von c nach d gibt. Dann heißen die Äquivalenzklassen die *Zusammenhangskomponenten* von G. Offensichtlich kann man in G Schleifen und parallele Kanten entfernen, ohne die Partition in Zusammenhangskomponenten zu verändern.

Satz 5.12 (Anzahlbedingung für Bäume)**.** *Sei G ein zusammenhängender ungerichteter Multigraph mit mindestens einer Ecke. Dann ist G genau dann ein Baum, wenn er eine Ecke mehr als Kanten hat.*

Beweis.

$\Rightarrow$: *Sei $G = (E, K, r)$ ein Baum. Wir zeigen*

$$\#(E) = \#(K) + 1$$

durch Induktion nach $\#(K)$. Wenn $\#(K) = 0$ ist, dann gibt es nur eine Ecke. Sei nun $\#(K) > 0$. Herausnehmen einer beliebigen Kante liefert einen Wald mit zwei Zusammenhangskomponenten E_1 und E_2, die wieder Bäume sind. Nach Induktionsannahme sind

$$\#(E_1) = \#(K_1) + 1 \quad \text{und} \quad \#(E_2) = \#(K_2) + 1\,.$$

Somit erhalten wir

$$\#(E) = \#(E_1) + \#(E_2) = \#(K_1) + 1 + \#(K_2) + 1 = \#(K) + 1\,.$$

$\Leftarrow$: *Sei umgekehrt G kein Baum. Dann gibt es einen einfachen Zykel*

$$(k_0, k_1, \ldots, k_{\ell-1})\,.$$

Sei Z die Menge der Ecken des Zykels. Für jede Ecke e in $E \setminus Z$ wählen wir einen Weg minimaler Länge zu einer Ecke aus Z und bezeichnen die erste Kante mit $k(e)$. Dann ist die Abbildung

$$E \setminus Z \to K \setminus \{k_0, k_1, \ldots, k_{\ell-1}\}\,,\ e \mapsto k(e)\,,$$

injektiv. Es folgt

$$\#(E) - \ell \leq \#(K) - \ell$$

und somit

$$\#(E) \leq \#(K)\,.$$

□

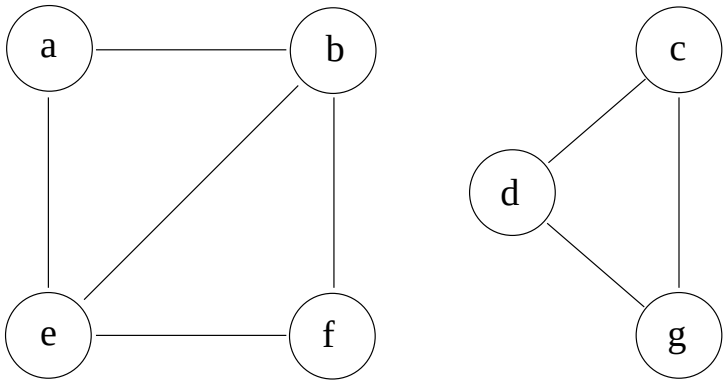
Definition 5.14 (spannender Wald)**.** Sei G ein ungerichteter Multigraph. Ein Teilgraph G' von G heißt ein *spannender Wald* von G, wenn

(a) G' ein Wald ist und

(b) die Partitionen in Zusammenhangskomponenten von G bzw. G' übereinstimmen.

Notwendigerweise ist dann $E' = E$.

Beispiel 5.15. Für den Graphen



gibt es

$$(10 - 2) \cdot 3 = 24$$

spannende Wälder.

Interessiert man sich für einen spannenden Wald mit minimaler Kantenbewertung (z.B. um eine Stadt mit einem Kanalnetz auszustatten), so zeigt Beispiel 5.15, dass es ineffizient ist alle spannenden Wälder auszurechnen und einen mit minimaler Kantenbewertung auszuwählen. Das folgende Verfahren schafft Abhilfe:

Satz 5.13 (Algorithmus von Kruskal)**.** *Sei $G = (E, K, r)$ ein ungerichteter Multigraph mit Kantenbewertung b. Gesucht werden die Partition von E in Zusammenhangskomponenten sowie die Kantenmenge W eines spannenden Waldes von G mit minimaler Bewertung*

$$\sum_{k \in W} b(k)\,.$$

Als Vorbereitung werden im Multigraphen alle Schleifen entfernt, parallele Kanten bis auf jene mit kleinster Bewertung gestrichen, und die verbleibenden Kanten sortiert, sodass

$$b(k_0) \leq b(k_1) \leq \ldots \leq b(k_{m-1})$$

gilt. Der eigentliche Algorithmus operiert dann mit $\mathrm{O}(\#(E) \cdot \#(K))$ Operationen wie folgt.

> *Setze $W = \varnothing$ und $P = \{\{e\} \mid e \in E\}$.*
>
> *Für i von 0 bis $m - 1$ wiederhole:*
>
> *Falls die Ecken e und d von k_i in verschiedenen Blöcken von P liegen,*
>
> *vereinige die beiden Blöcke von P und nimm k_i in die Menge W auf.*

Beweis. Sei G_i der Teilgraph von G mit Eckenmenge E und Kantenmenge

$$\{k_0, k_1, , \ldots, k_i\}\,.$$

Der Algorithmus startet mit der Partition in einzelne Ecken und vereinigt anschließend Blöcke mit Verbindungskante. Nach Schritt i ist P die Partition in Zusammenhangskomponenten von G_i. Die Menge W ist zunächst leer und wird im Schritt i um eine etwaige Verbindungskante erweitert, deren Ecken dann im Vereinigungsblock liegen. Für jeden Block B ist der Teilgraph mit Eckenmenge B und den entsprechenden Kanten aus W ein Baum, weil er zusammenhängt und die Anzahlbedingung

$$\#(E_1) + \#(E_2) = (\#(K_1) + 1) + (\#(K_2) + 1) = (\#(K_1) + \#(K_2) + 1) + 1$$

erfüllt. Somit ist nach Schritt i der Teilgraph mit Eckenmenge E und Kantenmenge W ein spannender Wald von G_i.

Zu zeigen bleibt noch, dass die Greedy-Strategie (greedy, engl. für gierig) bei der Wahl der Kanten einen spannenden Wald mit minimaler Bewertung liefert. Sei dazu M die Kantenmenge eines spannenden Waldes mit minimaler Bewertung. Wenn $M = W$ ist, haben wir die Behauptung gezeigt. Wenn $M \neq W$ ist, dann existiert eine Kante k_i in W, die nicht in M liegt. Seien e_1, e_2 die Ecken von k_i und E_1, E_2 die zugehörigen Blöcke im Algorithmus. Da es einen Weg p von e_1 nach e_2 aus Kanten in M gibt, existiert eine Kante k_j im Weg p, die eine Endecke in E_1 und die andere außerhalb von E_1 hat. Somit ist

$$j > i \quad \text{und} \quad b(k_j) \geq b(k_i)\,.$$

Der neue Teilgraph mit Eckenmenge E und Kantenmenge

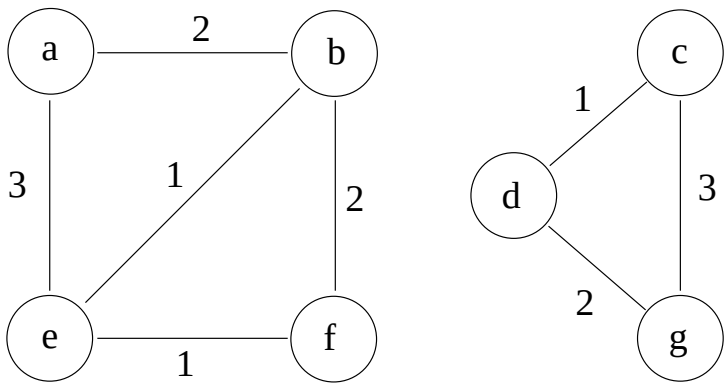
$$N := (M \setminus \{k_j\}) \cup \{k_i\}$$

ist ein spannender Wald, weil jeder Weg über k_j auch über k_i und die restlichen Kanten von p geführt werden kann und umgekehrt. Wegen

$$\sum_{k \in N} b(k) = \sum_{k \in M} b(k) - b(k_j) + b(k_i) \leq \sum_{k \in M} b(k)$$

hat der neue spannende Wald ebenfalls eine minimale Bewertung und zusätzlich eine kleinere Indexsumme. Somit erhält man durch endlich viele Austäusche die Kantenmenge W aus dem Algorithmus von Kruskal. Weil M eine minimale Kantenbewertung hat, so auch W. □

Beispiel 5.16. Für den bewerteten Graphen



startet der Algorithmus von Kruskal mit

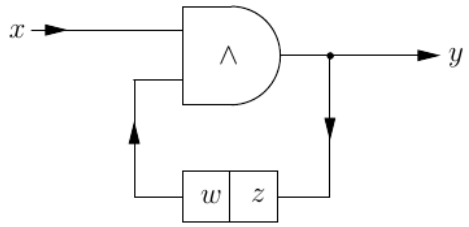
$$W = \varnothing \quad \text{und} \quad P = \{\{a\}, \{b\}, \{c\}, \{d\}, \{e\}, \{f\}, \{g\}\}$$

und endet mit

$$W = \{\{a, b\}, \{b, e\}, \{c, d\}, \{d, g\}, \{e, f\}\} \quad \text{und} \quad P = \{\{a, b, e, f\}, \{c, d, g\}\}\,.$$

5.3 Aufgaben

Aufgabe 5.1. Im synchronen Schaltwerk mit Eingang x, Ausgang y, einem And-Gatter und einem Schieberegister der Länge 2



lauten die Gleichungen für die binären Signalfolgen

$$\begin{aligned} y(t) &= x(t) \wedge w(t) \\ w(t+1) &= z(t) \\ z(t+1) &= y(t)\ , \end{aligned}$$

wobei $t \in \mathbb{N}$ eine diskrete Zeit ist. Berechnen Sie das Zustandsdiagramm. Was können Sie aus dem Zustandsdiagramm herauslesen?

Aufgabe 5.2. Was ist ein gerichteter Multigraph? Was ist ein gerichteter Graph? Gegeben ein gerichteter Graph G mit Eckenmenge

$$E = \{1, 2, 3, 4\}$$

und Kantenmenge

$$\{(1,1), (1,2), (2,2), (2,3), (3,3), (3,4), (4,4)\}\ .$$

Visualisieren Sie G. Wie viele Wege gibt es in G von Ecke 1 zu Ecke 4? Welche dieser Wege sind einfach? Ist G stark zusammenhängend? Wie viele Zyklen gibt es in G?

Hinweis: Nummerieren Sie die Kanten in G von 1 bis 7.

Aufgabe 5.3. Beweisen Sie:

„Jeder Wurzelbaum ist zyklenfrei.“

Aufgabe 5.4. Gegeben ein gerichteter Graph G durch die Relation

$$R = \{(1,1), (1,2), (2,2), (2,3), (3,3), (3,4), (4,4)\}$$

auf $M = \{1, 2, 3, 4\}$. Wie viele Wege der Länge 3 gibt es in G? Berechnen Sie mit dem Algorithmus von Warshall die transitive Hülle von R.

Aufgabe 5.5. Sei G der gerichtete Graph, den man aus dem gerichteten Multigraphen aus Beispiel 5.1 nach Weglassen der Kante 3 erhält. Sei R die Relation des gerichteten Graphen G. Berechnen Sie mit dem Algorithmus von Warshall die transitive Hülle von R.

Aufgabe 5.6. Gegeben ein gerichteter Graph G durch die Relation

$$\{(1,2),(2,3),(1,3),(1,4),(4,3)\}$$

auf $M = \{1,2,3,4\}$ und Kantenbewertung

$$b((1,2)) = 1,\ b((1,3)) = 4,\ b((1,4)) = 1,\ b((2,3)) = 2,\ b((4,3)) = 1\,.$$

Ist G ein Wurzelbaum? Berechnen Sie mit dem Algorithmus von Floyd die Eckenabstände im Graphen G.

Aufgabe 5.7. Berechnen Sie mit dem Algorithmus von Floyd die Eckenabstände im gerichteten Multigraphen aus Beispiel 5.1. Nehmen Sie die Kantenbewertung $b\colon K \to K$, $b(k) = k$.

Aufgabe 5.8. Gegeben ein gerichteter Graph G durch die Relation

$$\{(1,2),(2,3),(1,3),(1,4),(4,3),(5,6),(7,6),(7,2)\}$$

auf $M = \{1,2,3,4,5,6,7\}$. Berechnen Sie mittels Nachfolgersuche die erreichbaren Ecken ausgehend von der Startmenge $S = \{1,5\}$.

Aufgabe 5.9. Berechnen Sie die unmittelbare Vorgängerrelation für die Relation

$$\leq\ = \{x \text{ ist ein } \textit{nicht-trivialer} \text{ Teiler von } y \mid x,y \in \{2,3,4,5,6,7,8,9,10,11,12\}\}\,.$$

Hinweis: Siehe Beispiel 2.9 bzw. Definition 7.2 für die Definition eines Teilers.

Aufgabe 5.10. Bestimmen Sie die unmittelbare Vorgängerrelation (bezüglich der Verfeinerungsordnung) auf der Menge der Partitionen von $\{a,b,c,d\}$.

Hinweis: Definition 2.4 und 2.7 führen die Begriffe *Partition* und *Verfeinerungsordnung* ein. Die Relation ist recht groß.

Aufgabe 5.11. Sei G ein ungerichteter Graph gegeben durch die Eckenmenge

$$E = \{1,2,3,4\}\,,$$

die Kantenmenge

$$K = \{0,1,2,3\}\,,$$

sowie der Abbildung r laut folgender Tabelle:

k	$r(k)$
0	$\{1\}$
1	$\{1,2\}$
2	$\{2,3\}$
3	$\{1,4\}$

Visualisieren Sie G. Ist G ein Baum? Berechnen Sie den Grad jeder Ecke.

Aufgabe 5.12. Gegeben ein ungerichteter Graph G durch die Eckenmenge

$$E = \{1, 2, 3, 4, 5, 6\}$$

und Kantenmenge

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

Visualisieren Sie den Graphen G. Wie viele Wege gibt es von der Ecke 1 zur Ecke 4? Welche dieser Wege sind einfach? Was sind reziproke Wege? Ist der Graph G zusammenhängend, zyklenfrei, ein Baum?

Aufgabe 5.13. Gegeben ein ungerichteter Graph G mit Eckenmenge

$$E = \{1, 2, 3, 4, 5, 6\}$$

und Kantenmenge

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

Wie viele Orientierungen von G gibt es? Finden Sie eine Orientierung von G, die stark zusammenhängend ist. Gibt es für jeden zusammenhängenden ungerichteten Multigraphen eine Orientierung, die stark zusammenhängend ist?

Aufgabe 5.14. Was ist ein Wald? Was ist ein Baum? Ein Baum mit e Ecken hat wie viele Kanten? Ein Wald mit e Ecken hat mindestens/maximal wie viele Kanten? Ein Wald mit e Ecken und k Kanten hat wie viele Zusammenhangskomponenten?

Aufgabe 5.15. Sei G ein bewerteter ungerichteter Multigraph gegeben durch die Eckenmenge

$$E = \{0, 1, 2, 3, 4, 5\}\,,$$

die Kantenmenge

$$K = \{0, 1, 2, 3, 4, 5, 6, 7\}\,,$$

sowie der Abbildungen r und b laut folgender Tabelle:

k	$r(k)$	$b(k)$
0	$\{1\}$	2
1	$\{1, 2\}$	3
2	$\{1, 3\}$	1
3	$\{2, 3\}$	1
4	$\{2, 3\}$	4
5	$\{2, 4\}$	5
6	$\{2, 5\}$	6
7	$\{4, 5\}$	1

Berechnen Sie mit dem Algorithmus von Kruskal einen spannenden Wald mit minimaler Bewertung. Ist dieser eindeutig?

Aufgabe 5.16. Anton und Berta implementieren den Algorithmus von Kruskal (Satz 5.13). Sie wundern sich, dass ihre Programme manchmal unterschiedliche Ergebnisse liefern, die Resultate aber immer spannende Wälder sind und zudem die Summe der Kantenbewertungen übereinstimmen.

Beide haben ihr Programm bereits oft überprüft und sind sich sicher, dass sie den Algorithmus korrekt implementiert haben. Helfen Sie Anton und Berta, indem Sie

- den Unterschied in den Implementierungen finden,
- den Unterschied anhand eines einfaches Beispiels demonstrieren.

Hinweis: Studieren Sie Satz 5.13 sowie dessen Beweis und finden Sie die Stelle mit der Wahlmöglichkeit?

Aufgabe 5.17. Beschreiben Sie den Zusammenhang zwischen „eine Ecke e ist unmittelbarer Vorgänger einer Ecke d“ aus Definition 5.1 und der unmittelbaren Vorgängerrelation (siehe Definition 5.7). Bestimmen Sie die Inklusionsordnung $\subseteq$ auf der Menge $\{a, b, c\}$ sowie deren Vorgängerrelation (Definition 2.5) und deren unmittelbare Vorgängerrelation. Visualisieren Sie die Relationen als gerichtete Graphen.

6

Zähltheorie

6.1 Aufzählen und Nummerieren von Objekten

Definition 6.1 (Aufzählung, Nummerierung)**.** Eine Menge M heißt *endlich*, wenn es eine natürliche Zahl m und eine bijektive Abbildung

$$\alpha\colon \{0, 1, \ldots, m-1\} \to M$$

gibt. In diesem Fall ist m eindeutig bestimmt und man nennt

$$\#(M) := m$$

die Anzahl der Elemente von M. Die Abbildung α ist im Allgemeinen nicht eindeutig und heißt eine *Aufzählung* von M. Eine bijektive Abbildung

$$\nu\colon M \to \{0, 1, \ldots, m-1\}$$

wird eine *Nummerierung* von M genannt. Offenbar ist die Umkehrabbildung einer Aufzählung von M eine Nummerierung von M und die Umkehrabbildung einer Nummerierung von M ist eine Aufzählung von M. Wenn M nicht endlich ist, dann heißt M *unendlich* und man schreibt

$$\#(M) = \infty\,.$$

Satz 6.1 (elementare Zählregeln)**.**

(1) Gleichheitsregel: *Sind M und N endliche Mengen und ist $f\colon M \to N$ eine bijektive Abbildung, so gilt*

$$\#(M) = \#(N)\,.$$

(2) Summenregel: *Sind $A_1, A_2, \ldots, A_k$* paarweise disjunkte *endliche Mengen, so gilt für die Vereinigung*

$$\#(A_1 \cup A_2 \cup \ldots \cup A_k) = \sum_{i=1}^{k} \#(A_i)\,.$$

(3) Differenzregel: *Für endliche Mengen A und B gilt*

$$\#(A \setminus B) = \#(A) - \#(A \cap B)\,.$$

(4) Siebformel: *Für endliche Mengen $A_1, A_2, \ldots, A_k$ gilt*

$$\#(A_1 \cup \ldots \cup A_k) = \sum_{\substack{I \subseteq \{1,2,\ldots,k\} \\ I \neq \varnothing}} (-1)^{\#(I)-1} \, \#(\bigcap_{i \in I} A_i) \, .$$

Insbesondere ist für endliche Mengen A und B

$$\#(A \cup B) = \#(A) + \#(B) - \#(A \cap B) \, .$$

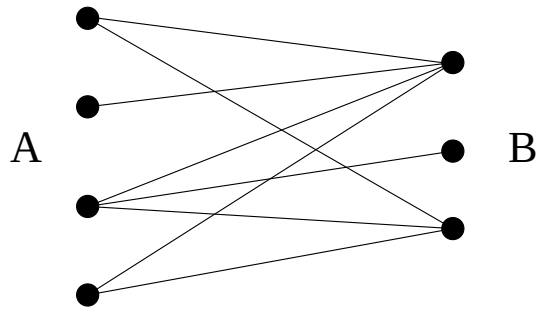
(5) Produktregel: *Sind $M_1, M_2, \ldots, M_k$ endliche Mengen, so gilt für das kartesische Produkt*

$$\#(M_1 \times M_2 \times \ldots \times M_k) = \prod_{i=1}^{k} \#(M_i) \, .$$

Insbesondere ist für eine endliche Menge M

$$\#(M^k) = \#(M)^k \, .$$

(6) Regel des zweifachen Abzählens:
Ein ungerichteter Graph heißt bipartit*, wenn es eine Partition der Eckenmenge in zwei Blöcke E_1 und E_2 gibt, sodass jede Kante eine Endecke in E_1 und eine Endecke in E_2 hat.*



Für einen endlichen bipartiten Graphen ist dann

$$\sum_{e_1 \in E_1} \mathrm{Grad}(e_1) = \sum_{e_2 \in E_2} \mathrm{Grad}(e_2) \, .$$

Beweis. (1) Da M endlich ist, gibt es eine natürliche Zahl m und eine bijektive Abbildung $\alpha\colon \{0, 1, \ldots, m-1\} \to M$, woraus wir $\#(M) = m$ erhalten. Da die zusammengesetzte Abbildung

$$f \circ \alpha\colon \{0, 1, \ldots, m-1\} \to N \ , \ i \mapsto f(\alpha(i)) \, ,$$

ebenfalls bijektiv ist erhalten wir $\#(N) = m$ und daraus die Behauptung.
(2) Seien $\alpha_1\colon \{0, 1, \ldots, m_1 - 1\} \to M_1, \ldots, \alpha_k\colon \{0, 1, \ldots, m_k - 1\} \to M_k$ bijektiv. Dann ist auch die zusammengesetzte Abbildung

$$\alpha\colon \{0, 1, \ldots, m_1 + \ldots + m_k - 1\} \to M_1 \cup \ldots \cup M_k \, ,$$

$$i \mapsto \begin{cases} \alpha_1(i) & \text{falls } i \in \{0,1,\ldots,m_1-1\} \\ \alpha_2(i-m_1) & \text{falls } i \in \{m_1,\ldots,m_1+m_2-1\} \\ \vdots & \vdots \\ \alpha_k(i-m_1-\ldots-m_{k-1}) & \text{falls } i \in \{m_1+\ldots+m_{k-1},\ldots,m_1+\ldots+m_k-1\}, \end{cases}$$

bijektiv.
(3) Aus der disjunkten Vereinigung

$$A = (A \setminus B) \cup (A \cap B)$$

folgt nach (2)

$$\#(A \setminus B) = \#(A) - \#(A \cap B)\,.$$

(4) Wir führen eine Induktion über k. Aus der disjunkten Vereinigung

$$A_1 \cup A_2 = A_1 \cup (A_2 \setminus A_1)$$

folgt

$$\#(A_1 \cup A_2) = \#(A_1) + \#(A_2 \setminus A_1) = \#(A_1) + \#(A_2) - \#(A_1 \cap A_2)\,.$$

Für $k > 2$ ist nach Induktionsannahme

$$\#(\bigcup_{i=1}^{k} A_i) = \#((\bigcup_{i=1}^{k-1} A_i) \cup A_k) = \#(\bigcup_{i=1}^{k-1} A_i) + \#(A_k) - \#(\bigcup_{i=1}^{k-1}(A_i \cap A_k)) =$$
$$\sum_{\substack{I \subseteq \{1,\ldots,k-1\} \\ I \neq \varnothing}} (-1)^{\#(I)-1}\, \#(\bigcap_{i \in I} A_i) + \#(A_k) - \sum_{\substack{I \subseteq \{1,\ldots,k-1\} \\ I \neq \varnothing}} (-1)^{\#(I)-1}\, \#(\bigcap_{i \in I} A_i \cap A_k) =$$
$$\sum_{\substack{J \subseteq \{1,\ldots,k\} \\ J \neq \varnothing}} (-1)^{\#(J)-1}\, \#(\bigcap_{i \in J} A_i)\,,$$

weil entweder $J = I$ oder $J = \{k\}$ oder $J = I \cup \{k\}$ gewählt werden kann.
(5) Seien $\alpha_1\colon \{0,1,\ldots,m_1-1\} \to M_1, \ldots, \alpha_k\colon \{0,1,\ldots,m_k-1\} \to M_k$ bijektiv. Dann ist auch die Abbildung

$$\alpha\colon \{0,1,\ldots,m_1 \cdots m_k - 1\} \quad \to \quad M_1 \times \ldots \times M_k$$

mit

$$I \quad \mapsto \quad (\alpha_1(I/m_2 \cdots m_k), \ldots, \alpha_{k-1}((I/m_k) \bmod m_{k-1}), \alpha_k(I \bmod m_k))$$

bijektiv, weil man aus den Einzelnummern

$$\begin{aligned} i_k &= I \bmod m_k \\ i_{k-1} &= (I/m_k) \bmod m_{k-1} \\ &\vdots \\ i_2 &= (I/(m_3 \cdots m_k)) \bmod m_2 \\ i_1 &= I/(m_2 \cdots m_k) \end{aligned}$$

die Gesamtnummer

$$I := i_1 \cdot m_2 \cdots m_k + i_2 \cdot m_3 \cdots m_k + \ldots + i_{k-1} \cdot m_k + i_k$$

zurückerhält.
(6) Beide Summen geben die Zahl der Kanten an, einmal über die Endecken in E_1 gezählt, das andere Mal über die Endecken in E_2. □

Beispiel 6.1. In C-Programmen werden die Elemente mehrdimensionaler Felder hintereinander im Speicher abgelegt, wobei die Reihenfolge so geregelt ist, dass

„hintere Indizes schneller laufen als vordere".

Zum Beispiel liegen für

```
int M[2][3] = {{3,5,-2},{1,0,2}};
```

die Feldelemente wie folgt im Speicher:

M[0][0]	M[0][1]	M[0][2]	M[1][0]	M[1][1]	M[1][2]
3	5	-2	1	0	2

M

Wird ein mehrdimensionales Feld an eine Funktion übergeben, die variable Dimensionen zulässt, dann muss der Programmierer in der Funktion die relativen Adressen der Feldelemente selbst berechnen, z.B. im folgenden Codefragment:

```
...
double f(double *z, int m1, int m2, int m3)
{
  ...
}
...
int main( void)
{
   double x, y, A[2][3][4], B[3][4][2];
   ...
   x = f(&A[0][0][0],2,3,4);
   y = f(&B[0][0][0],3,4,2);
   ...
}
```

Nach Satz 6.1 kann in der Funktion `f` das Feldelement „`z[i][j][k]`" als

```
*(z+i*m2*m3+j*m3+k)
```

angesprochen werden. Die Indizes `i`, `j`, `k` des Feldelements an der Adresse `z+l` können aus den Formeln

```
k = l%m3
j = (l/m3)%m2
i = l/(m2*m3)
```

berechnet werden.

Satz 6.2 (Schubfachprinzip, Taubenschlagprinzip)**.** *Seien $f\colon M \to N$ eine Abbildung und M, N endliche Mengen. Wenn $\#(M) > \#(N)$ ist, dann gibt es mindestens ein Element $y \in N$ mit mehr als einem Urbild.*

Beweis. Für einen indirekten Beweis nehmen wir an, dass jedes Element von N höchstens ein Urbild hat. Dann ist f injektiv und somit die eingeschränkte Abbildung $M \to f(M)$ bijektiv. Satz 6.1(1) liefert $\#(M) = \#(f(M))$ und aus $f(M) \subseteq N$ folgt $\#(M) \leq \#(N)$. Widerspruch. □

Satz 6.3 (Zahl der Abbildungen)**.** *Seien K und M endliche Mengen mit k bzw. m Elementen. Dann gibt es genau m^k verschiedene Abbildungen von K nach M.*

Beweis. Schreibt man $K = \{x_1, \ldots, x_k\}$, dann ist jede Abbildung $f\colon K \to M$ durch das Tupel $(f(x_i))_{i=1}^k$ in M^k eindeutig bestimmt. Damit folgt die Behauptung durch Anwenden der Gleichheitsregel und der Produktregel. □

Satz 6.4 (Zahl der injektiven Abbildungen)**.** *Seien K und M endliche Mengen mit k bzw. m Elementen. Dann gibt es genau*

$$(m)_k := \begin{cases} m(m-1)(m-2)\cdots(m-k+1) & \text{falls } k \geq 1 \\ 1 & \text{falls } k = 0 \end{cases}$$

verschiedene injektive Abbildungen von K nach M. Man nennt die Zahl $(m)_k$ die fallende Faktorielle *von m und k.*

Beweis. Wir zeigen die Formel durch Induktion über k. Am Induktionsanfang ist $k = 0$, somit K leer und die einzige injektive Abbildung die leere Abbildung. Für den Induktionsschluss schreiben wir

$$K = \{x_0, x_1, \ldots, x_k\}$$

und überlegen uns, wie viele injektive Abbildungen $f\colon K \to M$ es geben kann. Für x_0 gibt es m Möglichkeiten, ein Bild $f(x_0) \in M$ zu wählen. Dieses Element

$$y_0 := f(x_0)$$

darf dann aber nicht mehr als Bild eines anderen Elements in K gewählt werden, sodass für die Wahl der Bilder von $x_1, \ldots, x_k$ nur die Elemente in $M \setminus \{y_0\}$ in Frage kommen. Nach Induktionsannahme gibt es dafür $(m-1)_k$ Möglichkeiten. Die Gesamtzahl der Möglichkeiten ist somit

$$m \cdot (m-1)_k = (m)_{k+1}\,.$$

□

Beispiel 6.2. Offensichtlich gibt es keine injektive Abbildung von $\{0,1,2,3\}$ nach $\{0,1\}$, was mit Satz 6.4 übereinstimmt, da $(2)_4 = 2 \cdot 1 \cdot 0 \cdot -1 = 0$.

Beispiel 6.3. Sei M eine endliche Menge von Objekten. In der Literatur werden injektive Abbildungen

$$\{0,1,\ldots,k-1\} \to M\,,\; i \mapsto x_i\,,$$

als k-Tupel

$$(x_0, x_1, \ldots, x_{k-1})$$

von *unterschiedlichen* Elementen von M beschrieben und *Permutationen* von jeweils k Objekten aus M genannt.

Satz 6.5 (Zahl der bijektiven Abbildungen)**.** *Seien K und M endliche Mengen mit jeweils m Elementen. Dann gibt es genau*

$$m! \;:=\; \begin{cases} m(m-1)(m-2)\cdots 3\cdot 2\cdot 1 & \textit{falls } m \geq 1 \\ 1 & \textit{falls } m = 0 \end{cases}$$

verschiedene bijektive Abbildungen von K nach M. Man nennt die Zahl $m!$ die Faktorielle *oder* Fakultät *von m.*

Beweis. Wegen $\#(K) = \#(M) = m$ ist jede injektive Abbildung von K nach M bijektiv. Damit folgen die Behauptungen aus Satz 6.4 mit $(m)_m = m!$. □

Bemerkung. Die Faktorielle wächst sehr schnell und kann für große m mit der *Stirlingschen Formel*

$$m! \approx \sqrt{2\pi} \cdot e^{(m+\frac{1}{2})\log m - m} = \Theta(\sqrt{m} \cdot \left(\frac{m}{e}\right)^m)$$

approximiert werden (ohne Beweis).

Beispiel 6.4. Für m verschiedene Objekte gibt es $m!$ verschiedene Möglichkeiten, eine Reihenfolge festzulegen, d.h. die Objekte zu nummerieren.

Satz 6.6 (Zahl von Teilmengen)**.** *Sei M eine endliche Menge mit m Elementen. Dann gilt*

$$\#(\mathcal{P}(M)) = 2^m\,.$$

Beweis. Wir fixieren eine Aufzählung $\alpha\colon \{0,1,\ldots,m-1\} \to M$. Dann ist die folgende Abbildung bijektiv, insbesondere können Teilmengen als Bitmuster programmiert werden.

$$F\colon \mathcal{P}(M) \to \{0,1\}^m\,,\; T \mapsto (t_0,\ldots,t_{m-1})\,,\; t_i := \begin{cases} 1 & \textit{falls } \alpha(i) \in T \\ 0 & \textit{sonst.} \end{cases}$$

□

Satz 6.7 (Zahl von Teilmengen mit vorgegebener Anzahl von Elementen)**.** *Sei M eine endliche Menge mit m Elementen und sei k eine natürliche Zahl. Dann gilt*

$$\#(\mathcal{P}_k(M)) = \binom{m}{k}\,.$$

Dabei ist der Binomialkoeffizient *„m über k“ definiert als*

$$\binom{m}{k} := \frac{m \cdot (m-1) \cdots (m-k+1)}{k \cdot (k-1) \cdots 1} = \begin{cases} \dfrac{m!}{k!(m-k)!} & \text{falls } k \leq m \\ 0 & \text{sonst.} \end{cases}$$

Beweis. Eine Aufzählung $\alpha\colon \{0, 1, \ldots, k-1\} \to T$ einer k-elementigen Teilmenge T von M erhält man durch Wählen

- *eines beliebigen Elements $\alpha(0) \in M$,*
- *eines beliebigen Elements $\alpha(1) \in M \setminus \{\alpha(0)\}$,*
- *eines beliebigen Elements $\alpha(2) \in M \setminus \{\alpha(0), \alpha(1)\}$, usw.*

Da es bei der Teilmenge T nicht auf die Reihenfolge der gewählten Elemente ankommt, ergibt sich die gesuchte Anzahl als

$$m \cdot (m-1) \cdots (m-k+1)/k!\,.$$

□

Beispiel 6.5. Sei M eine endliche Menge von Elementen. In der Literatur werden k-elementige Teilmengen von M *Kombinationen* von jeweils k Elementen aus M genannt. Im Gegensatz zu den Permutationen von Elementen aus M spielt bei den Kombinationen die Reihenfolge der ausgewählten Elemente keine Rolle.

Beispiel 6.6. Sei M eine endliche Menge mit m Elementen und sei $k \leq m$ eine natürliche Zahl. Dann ist die Abbildung

$$\mathcal{P}_k(M) \to \mathcal{P}_{m-k}(M)\ ,\ T \mapsto M \setminus T\,,$$

bijektiv und somit

$$\binom{m}{k} = \binom{m}{m-k}\,.$$

6.2 Abzählbarkeit von Mengen

Die folgende Definition erweitert Definition 6.1 auf unendliche Mengen.

Definition 6.2 (abzählbare Unendlichkeit)**.** Eine Menge M heißt *abzählbar unendlich*, wenn eine bijektive Abbildung

$$\alpha\colon \mathbb{N} \to M\ ,\ i \mapsto x_i\,,$$

existiert. Man schreibt dann

$$M = \{x_0, x_1, x_2, \ldots\}\,,$$

nennt α eine *Aufzählung* von M und α^{-1} eine *Nummerierung* von M.

Beispiel 6.7. Die Menge $\mathbb{N}$ der natürlichen Zahlen ist abzählbar unendlich, weil die identische Abbildung bijektiv ist. Auch die Menge $\mathbb{Z}$ der ganzen Zahlen ist abzählbar unendlich, weil die Abbildung

$$\mathbb{N} \to \mathbb{Z}\ ,\ i \mapsto \begin{cases} i/2 & \text{falls } i \text{ gerade} \\ -(i+1)/2 & \text{falls } i \text{ ungerade} \end{cases}$$

bijektiv ist.

Satz 6.8. *Die Menge $\mathbb{N} \times \mathbb{N}$ ist abzählbar unendlich.*

Beweis. Anstatt einer Aufzählung $\alpha\colon \mathbb{N} \to \mathbb{N} \times \mathbb{N}$ geben wir eine Nummerierung $\nu\colon \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ an. Wir schreiben die Paare (m,n) zweidimensional

$$\begin{array}{ccccc} (0,0) & (1,0) & (2,0) & (3,0) & \dots \\ (0,1) & (1,1) & (2,1) & (3,1) & \dots \\ (0,2) & (1,2) & (2,2) & (3,2) & \dots \\ (0,3) & (1,3) & (2,3) & (3,3) & \dots \\ \vdots & & & & \end{array}$$

auf und nummerieren diagonal

$$\begin{array}{ccc} (0,0) & \mapsto & 0 \\ (0,1) & \mapsto & 1 \\ (1,0) & \mapsto & 2 \\ (0,2) & \mapsto & 3 \\ (1,1) & \mapsto & 4 \\ (2,0) & \mapsto & 5 \\ (0,3) & \mapsto & 6 \\ & \vdots & \end{array}$$

Dabei bekommt das Paar (m,n) die Nummer

$$\left(\sum_{i=0}^{m+n-1} (i+1) \right) + m\,.$$

Somit ist die Abbildung

$$\mathbb{N} \times \mathbb{N} \to \mathbb{N}\ ,\ (m,n) \mapsto \frac{(m+n)(m+n+1)}{2} + m\,,$$

bijektiv. □

Definition 6.3 (graduiert-lexikographische Ordnung auf Zahlentupeln)**.** Für $x, y \in \mathbb{N}^k$ sei

$$x <_{\text{gradlex}} y\,,$$

falls entweder

$$\sum_{i=1}^{k} x_i <_{\mathbb{N}} \sum_{i=1}^{k} y_i$$

oder

$$\sum_{i=1}^{k} x_i = \sum_{i=1}^{k} y_i \quad \text{und} \quad x <_{\text{lex}} y$$

ist. Wir nennen $\leq_{\text{gradlex}}$ die *graduiert-lexikographische Ordnung auf Zahlentupeln*. Zudem ist $\leq_{\text{gradlex}}$ eine totale Ordnung auf $\mathbb{N}^k$.

Der folgende Satz ist eine Verallgemeinerung von Satz 6.8.

Satz 6.9. *Die Menge $\mathbb{N}^k$ ist abzählbar unendlich.*

Beweis. Jedes $x \in \mathbb{N}^k$ hat nur endlich viele Vorgänger. Daher liefert die Nummerierung der Elemente in der graduiert-lexikographischen Ordnung eine bijektive Abbildung $\mathbb{N}^k \to \mathbb{N}$. □

Definition 6.4. Eine Menge heißt *abzählbar*, wenn sie endlich oder abzählbar unendlich ist.

Satz 6.10 (Abzählbarkeitseigenschaften)**.**

(1) Jede Teilmenge einer abzählbaren Menge ist abzählbar.

(2) Das Bild einer abzählbaren Menge ist abzählbar.

(3) Die Vereinigung einer Folge von abzählbaren Mengen ist abzählbar.

(4) Das kartesische Produkt endlich vieler abzählbarer Mengen ist abzählbar.

Beweis. (1) Sei $M = \{x_0, x_1, x_2, \ldots\}$ abzählbar unendlich und sei N eine unendliche Teilmenge von M. Wähle y_0 als Element x_{n_0} von N mit dem kleinsten Index n_0, dann y_1 als Element x_{n_1} von $N \setminus \{y_0\}$ mit dem kleinsten Index n_1, weiters y_2 als Element x_{n_2} von $(N \setminus \{y_0\}) \setminus \{y_1\}$ mit dem kleinsten Index n_2, usw. Da jedes Element von N einen Index hat, ist

$$N = \{y_0, y_1, y_2, \ldots\}$$

abzählbar.
(2) Sei $M = \{x_0, x_1, x_2, \ldots\}$ und sei $f\colon M \to N$ mit $f(M)$ unendlich. Dann ist

$$f(M) = \{f(x_n) \mid n = 0, 1, 2, \ldots\}\,.$$

Wähle $y_0 = f(x_0)$, dann $y_1 = f(x_{n_1})$, wobei $x_{n_1} \in M \setminus f^{-1}(y_0)$ den kleinsten Index n_1 hat, weiters $y_2 = f(x_{n_2})$, wobei $x_{n_2} \in (M \setminus f^{-1}(y_0)) \setminus f^{-1}(y_1)$ den kleinsten Index n_2 hat, usw. Dann ist $f(M) = \{y_0, y_1, y_2, \ldots\}$.
(3) Seien

$$\begin{aligned} M_0 &= \{x_{00}, x_{01}, x_{02}, \ldots\} \\ M_1 &= \{x_{10}, x_{11}, x_{12}, \ldots\} \\ M_2 &= \{x_{20}, x_{21}, x_{22}, \ldots\} \\ &\vdots \end{aligned}$$

Dann ist die Abbildung

$$f\colon \mathbb{N}^2 \to \bigcup_{n\geq 0} M_n\,,\ (i,j) \mapsto x_{ij}\,,$$

surjektiv. Nach Satz 6.8 und (2) ist die Vereinigung abzählbar.
(4) Wir können annehmen, dass die Mengen $M_0, \ldots, M_{k-1}$ abzählbar unendlich sind, ansonsten ergänzen wir Elemente. Aus den Aufzählungen $\alpha_0, \ldots, \alpha_{k-1}$ von $M_0, \ldots, M_{k-1}$ bekommt man die bijektive Abbildung

$$\mathbb{N}^k \to M_0 \times \ldots \times M_{k-1}\,,\ (i_0, \ldots, i_{k-1}) \mapsto (\alpha_0(i_0), \ldots, \alpha_{k-1}(i_{k-1})).$$

Nach Satz 6.9 gibt es eine bijektive Abbildung von $\mathbb{N}$ nach $\mathbb{N}^k$. Hintereinanderausführen liefert eine bijektive Abbildung von $\mathbb{N}$ nach $M_0 \times \ldots \times M_{k-1}$. □

Beispiel 6.8. Sei Σ ein endliches Alphabet. Dann ist das Wortmonoid

$$\Sigma^* := \bigcup_{n\geqslant 0} \Sigma^n = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \cdots$$

abzählbar. Die Menge Σ^* enthält zwar beliebig lange, aber ausschließlich Wörter endlicher Länge.

Der folgende Satz erlaubt eine Alternative, um die Abzählbarkeit nachzuweisen.

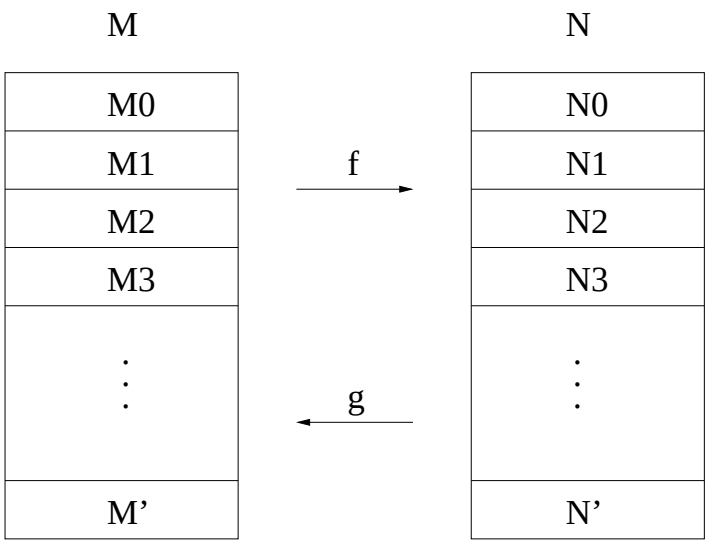
Satz 6.11 (Satz von Bernstein)**.** *Seien $f\colon M \to N$ und $g\colon N \to M$ injektive Abbildungen. Dann existiert eine bijektive Abbildung $h\colon M \to N$.*

Beweis. Wir definieren induktiv Teilmengen $M_0, M_1, M_2, \ldots$ von M sowie Teilmengen N_0, N_1, N_2, … von N durch

$$M_0 := M \setminus g(N) \quad \text{und} \quad N_0 := f(M_0)$$

und, für $n > 0$,

$$M_n := g(N_{n-1}) \quad \text{und} \quad N_n := f(M_n)\,.$$



Wir zeigen durch wohlfundierte Induktion, dass die Mengen $M_0, M_1, \ldots$ paarweise disjunkt sind: Für $n > 0$ ist

$$M_n \subseteq g(N)\,,$$

daher sind M_0 und M_n disjunkt. Für $m, n \in \mathbb{N}$ mit $0 < m < n$ folgt mit der Injektivität von f und g

$$M_m \cap M_n = gf(M_{m-1}) \cap gf(M_{n-1}) = gf(M_{m-1} \cap M_{n-1}) = \varnothing\,.$$

Da f injektiv ist, sind auch die Mengen $N_0, N_1, \ldots$ paarweise disjunkt, und die eingeschränkten Abbildungen

$$f_n \colon M_n \to N_n\,,\; x \mapsto f(x)\,,$$

sind bijektiv. Seien

$$M' := M \setminus \bigcup_{n \geq 0} M_n \quad \text{und} \quad N' := N \setminus \bigcup_{n \geq 0} N_n\,.$$

Wenn für $y \in N$ ein $n \geq 0$ existiert mit $g(y) \in M_n$, dann ist $n > 0$ und $y \in N_{n-1}$. Somit erhalten wir eine injektive Abbildung

$$g' \colon N' \to M'\,,\; y \mapsto g(y)\,.$$

Für $x \in M'$ gibt es wegen $M' \subseteq g(N)$ ein $y \in N$ mit $g(y) = x$. Wenn ein $n \geq 0$ existiert mit $y \in N_n$, dann wäre

$$x \in g(N_n) = M_{n+1}$$

im Widerspruch zu $x \in M'$. Somit ist $y \in N'$ und g' bijektiv.

Nach Konstruktion sind die Mengen $M_0, M_1, \ldots, M'$ paarweise disjunkt und

$$\left(\bigcup_{n \geq 0} M_n\right) \cup M' = M\,.$$

Ebenso sind die Mengen $N_0, N_1, \ldots, N'$ paarweise disjunkt und

$$\left(\bigcup_{n \geq 0} N_n\right) \cup N' = N\,.$$

Daher können die bijektiven Abbildungen $f_0, f_1, \ldots, (g')^{-1}$ zu einer bijektiven Abbildung $h \colon M \to N$ zusammengesetzt werden:

$$h(x) := \begin{cases} f_n(x) & \text{falls } x \in M_n \text{ für ein } n \geq 0 \\ (g')^{-1}(x) & \text{sonst.} \end{cases}$$

□

Beispiel 6.9. Seien $M = N = \mathbb{N}$,

$$f\colon M \to N\,,\ x \mapsto 2x\,, \quad \text{und} \quad g\colon N \to M\,,\ y \mapsto 2y+1\,.$$

Dann sind

$$\begin{aligned} M_0 &= \{2z \mid z \in \mathbb{N}\} \quad , \quad N_0 = \{4z \mid z \in \mathbb{N}\} \\ M_1 &= \{8z+1 \mid z \in \mathbb{N}\} \quad , \quad N_1 = \{16z+2 \mid z \in \mathbb{N}\} \\ M_2 &= \{32z+5 \mid z \in \mathbb{N}\} \quad , \quad N_2 = \{64z+10 \mid z \in \mathbb{N}\} \\ &\vdots \end{aligned}$$

Die folgende Methode erlaubt Mengen als nicht abzählbar zu beweisen.

Satz 6.12 (Diagonalisierung)**.** *Sei Σ ein Alphabet mit mindestens zwei Buchstaben a und b, und sei $s_0, s_1, s_2, \ldots$ eine Folge von Folgen in Σ:*

$$\begin{aligned} s_0 &= s_{00}s_{01}s_{02}\ldots \\ s_1 &= s_{10}s_{11}s_{12}\ldots \\ s_2 &= s_{20}s_{21}s_{22}\ldots \\ &\vdots \end{aligned}$$

Dann ist die Folge

$$d_n = \begin{cases} b & \textit{falls } s_{nn} = a \\ a & \textit{falls } s_{nn} \neq a \end{cases}$$

eine neue Folge.

Beweis. Wenn d keine neue Folge ist, dann gibt es einen Index n mit $d = s_n$, woraus $d_n = s_{nn}$ im Widerspruch zur Konstruktion von d folgt. □

Beispiel 6.10. Die Menge $\mathbb{B}^{\mathbb{N}}$ aller binären Folgen ist nicht abzählbar.

Definition 6.5 (Mächtigkeit, Kardinalität)**.** Wenn es eine bijektive Abbildung $f\colon M \to N$ gibt, dann heißen die Mengen M und N *gleichmächtig*. Offensichtlich ist Gleichmächtigkeit eine Äquivalenzrelation, und man nennt die Äquivalenzklasse

$$|M| := \{N \mid N \text{ gleichmächtig wie } M\}$$

die *Mächtigkeit* oder *Kardinalität* der Menge M. Für eine endliche Menge M ist die Menge

$$\{0, 1, 2, \ldots, \#(M) - 1\}$$

ein Repräsentant von $|M|$. Daher werden die Kardinalitäten endlicher Mengen oft mit den natürlichen Zahlen identifiziert.

Satz 6.13 (Ordnung von Kardinalitäten)**.** *Für Mengen M und N sei*

$$|M| \leq |N|\,,$$

wenn es eine injektive Abbildung $f\colon M \to N$ gibt. Dann ist $\leq$ eine partielle Ordnung auf den Kardinalitäten.

Beweis. Die Relation $\leq$ ist wohldefiniert, weil für andere Repräsentanten M' von $|M|$ bzw. N' von $|N|$ es bijektive Abbildungen $g\colon M \to M'$ und $h\colon N \to N'$ gibt und dann die Abbildung

$$hfg^{-1}\colon M' \to N'$$

injektiv ist. Klarerweise ist $\leq$ reflexiv und transitiv. Die Antisymmetrie ergibt sich aus Satz 6.11. □

Satz 6.14 (Hierarchie der Kardinalitäten)**.** *Für endliche Mengen M und N gilt*

$$|M| \leq |N| \quad \text{genau dann, wenn} \quad \#(M) \leq \#(N)\,.$$

Die kleinste Kardinalität einer unendlichen Menge ist $|\mathbb{N}|$, und es gilt

$$|\mathbb{N}| < |\mathbb{B}^{\mathbb{N}}|\,.$$

Beweis. Für endliche Mengen M und N existiert genau dann eine injektive Abbildung $f\colon M \to N$, wenn M höchstens so viele Elemente wie N hat. Wenn M unendlich ist, dann kann man ein Element x_0 aus M wählen, ein Element x_1 aus $M \setminus \{x_0\}$, ein Element x_2 aus $(M \setminus \{x_0\}) \setminus \{x_1\}$, usw. Somit erhält man eine injektive Abbildung

$$f\colon \mathbb{N} \to M\,,\ n \mapsto x_n\,,$$

woraus

$$|\mathbb{N}| \leq |M|$$

folgt. Insbesondere ist $|\mathbb{N}| \leq |\mathbb{B}^{\mathbb{N}}|$. Nach Beispiel 6.2 ist $\mathbb{B}^{\mathbb{N}}$ nicht abzählbar und somit $|\mathbb{N}| < |\mathbb{B}^{\mathbb{N}}|$. □

6.3 Diskrete Wahrscheinlichkeitstheorie

Unter Wahrscheinlichkeit verstehen wir den Grad der Zufälligkeit eines Ereignisses. Man wählt hier den Begriff des Experiments und legt einen *Ergebnisraum* S zugrunde, der die Menge aller möglichen Ergebnisse beschreibt. Intutitiv kann man sich S als die Menge der Ergebnisse eines Experimentes vorstellen. Wir betrachten nur endliche Ergebnisräume.

Definition 6.6. Eine *Wahrscheinlichkeitsverteilung* auf einem Ergebnisraumes S ist eine Abbildung $\mathrm{P}\colon S \to [0,1]$, also eine Funktion von S auf das geschlossene Intervall zwischen 0 und 1. Ferner muss gelten dass $\sum_{x\in S} \mathrm{P}(x) = 1$.

Eine Teilmenge $E \subseteq S$ wird *Ereignis* genannt.

Definition 6.7. Die *Wahrscheinlichkeit* eines Ereignisses E, bezeichnet als $\mathrm{P}(E)$ ist wie folgt definiert:

$$\mathrm{P}(E) := \sum_{x\in E} \mathrm{P}(x)\,.$$

Das nächste Lemma folgt sofort aus der Definition, siehe Satz 6.1.

Lemma 6.1. *Seien A und B Ereignisse, dann gilt:*

$$\mathrm{P}(A \cup B) = \mathrm{P}(A) + \mathrm{P}(B) - \mathrm{P}(A \cap B) \qquad \mathrm{P}(\sim A) = 1 - \mathrm{P}(A) \ ,$$

wobei $\sim A$ wie üblich die Komplementärmenge von A bezeichnet, also $S = A \cup \sim A$.

Definition 6.8. Wenn A und B Ereignesse sind und $\mathrm{P}(B) \neq 0$, dann wird die *bedingte Wahrscheinlichkeit von A unter der Vorausetzung B* als $\mathrm{P}(A \,|\, B)$ bezeichnet und wie folgt definiert:

$$\mathrm{P}(A \,|\, B) := \frac{\mathrm{P}(A \cap B)}{\mathrm{P}(B)} \ .$$

Abkürzend sprechen wir auch von der *Wahrscheinlichkeit von A, vorausgesetzt B.*

Lemma 6.2. *Seien A und B Ereignisse, dann gilt:*

$$\mathrm{P}(A \cap B) = \mathrm{P}(A) \cdot \mathrm{P}(B \,|\, A) = \mathrm{P}(B) \cdot \mathrm{P}(A \,|\, B)$$

Beweis. Übung. □

Satz 6.15 (Satz von Bayes)**.** *Sei S ein Ergebnisraum und sei $\{H_1, \dots H_n\}$ eine Partition von S. Weiters sei E ein weiteres Ereignis, sodass $\mathrm{P}(E) \neq 0$. Dann gilt für $i = 1, \dots, n$:*

$$\begin{aligned} \mathrm{P}(H_i \,|\, E) &= \frac{\mathrm{P}(H_i \cap E)}{\mathrm{P}(H_1 \cap E) + \cdots + \mathrm{P}(H_n \cap E)} \\ &= \frac{\mathrm{P}(H_1)\,\mathrm{P}(E \,|\, H_i)}{\mathrm{P}(H_1)\,\mathrm{P}(E \,|\, H_1) + \cdots + \mathrm{P}(H_n)\,\mathrm{P}(E \,|\, H_n)} \ . \end{aligned}$$

Wir nenen $\mathrm{P}(H_i)$ auch a priori *Wahrscheinlichkeit von H_i und $\mathrm{P}(H_i \mid E)$* a posteriori *Wahrscheinlichkeit von H_i, gegeben E.*

Beweis. Siehe [2] für den (nicht schweren) Beweis. □

Definition 6.9. Zwei Ereignisse A und B heißen *unabhängig*, wenn gilt $\mathrm{P}(A \,|\, B) = \mathrm{P}(A)\,\mathrm{P}(B)$.

Wenn A unabhänig von B ist, dann ist natürlich auch B unabhängig von A. Außerdem gilt, dass dann auch die komplementären Ereignisse $\sim A$ und $\sim B$ unabhängig sind.

Beispiel 6.11. Wir betrachten n Münzwürfe mit einer (nicht notwendigerweise fairen) Münze und wollen die Wahrscheinlichkeit berechnen mit der die Münze genau k-mal Kopf zeigt, wobei wir davon ausgehen, dass die Ergebnisse der Würfe unabhängig sind.

Dazu betrachten wir zunächst die Wahrscheinlichkeit, dass genau die ersten k Würfe Kopf zeigen und die restlichen Zahl. Wir bezeichnen mit A_i das Ereignis, dass beim $i$ten Wurf Kopf gewürfelt wird und nehmen an, dass die Münze mit Wahrscheinlichkeit p Kopf zeigt. Also gilt $\mathrm{P}(A_i) = p$ und $\mathrm{P}(\sim A_i) = 1 - p$. Somit können wir die Wahrscheinlichkeit, dass wir zunächst k-mal Kopf und $n - k$-mal Zahl würfeln, wir folgt berechnen:

$$\mathrm{P}(A_1) \cdots \mathrm{P}(A_k) \cdot \mathrm{P}(\sim A_{k+1}) \cdots \mathrm{P}(\sim A_n) = p^k (1-p)^{n-k} \ .$$

Für jede mögliche Anordnung k-mal Kopf und die restlichen Male Zahl zu würfeln, bekommen wir die selbe Wahrscheinlichkeit. Aus Satz 6.7 wissen wir, dass es $\binom{n}{k}$ viele solche Anordnungen gibt.

Somit ist die Wahrscheinlickeit genau k-mal Kopf zu würfeln wir folgt gegegeben:

$$\binom{n}{k} p^k (1-p)^{n-k} \ .$$

Eine solche Wahrscheinlichkeitsverteilung wird *Binomialverteilung* genannt.

Definition 6.10. Bezeichne $V\colon S \to \mathbb{R}$ eine Wertefunktion für den Ergebnisraum $S = \{x_1, \ldots, x_n\}$. Dann ist der *Erwartungswert* von V wie folgt definiert:

$$\mathrm{E}(V) := V(x_1)\,\mathrm{P}(x_1) + \cdots + V(x_n)\,\mathrm{P}(x_n) \ .$$

Die Wertefunktion V wird auch als *Zufallsvariable* bezeichnet.

Definition 6.11. Eine *Markowkette* ist eine endliche Abfolge von Zuständen, sodass jeder Zustandswechsel nur von dem vorherigen Zustand und einer vorausgesetzten Wahrscheinlichkeitsverteilung abhängt.

Beispiel 6.12. Wir betrachen zwei Zustände, bezeichnet mit 0 und 1. Die Wahrscheinlichkeiten der Zustandswechsel werden durch die folgende *Übergangsmatrix* P ausgedrückt:

$$\begin{pmatrix} 0.1 & 0.9 \\ 0.6 & 0.4 \end{pmatrix}$$

Dies bedeutet etwa dass die Wahrscheinlichkeit eines Überganges von Zustand 0 nach Zustand 1, 90% beträgt.

Die Wahrscheinlichkeiten nach n Schritten kann durch n-faches Potenzieren der Übergangsmatrix P berechnet werden. Genau gilt die folgende Eigenschaft für die hier betrachteten Markowketten.

Definition 6.12. Sei P eine Übergangsmatrix. Dann sind die Wahrscheinlichkeiten nach n Schritten durch P^n gegeben. Diese Eigenschaft heißt *Markoweigenschaft* oder auch *Gedächtnislosigkeit.*

Den nächsten Satz geben wir ohne Beweis an, siehe [2].

Satz 6.16. *Sei P eine Übergangsmatrix einer Markowkette, sodass für eine Potenz P' gilt, dass P' keine Nulleinträge besitzt. Dann gelten die folgenden Eigenschaften:*

1. *Es existitiert ein eindeutiger Wahrscheinlichsvektor $\vec{v}$, sodass $\vec{v}P = \vec{v}$ und $\vec{v}$ enthält keine Nulleinträge.*

2. *Bei wachsendem n nähert sich die Potenz P^n der Matrix an, die den Vektor $\vec{v}$ als Eintrag in jeder Zeile hat.*

3. *Bezeichne $\vec{v}_0$ die initial Wahrscheinlichkeitsverteilung der Zustände. Dann nähert sich der Vektor $\vec{v}_0 P^n$ dem Vektor $\vec{v}$ an.*

□

6.4 Lösen von Rekursionsformeln

Die Laufzeit oder der Speicherplatzbedarf eines Algorithmus hängt üblicherweise von der Eingabegröße n der Instanz ab. Im Allgemeinen ist es schwierig, für diese Funktion $f(n)$ eine explizite Formel zu finden. In vielen Fällen, besonders bei induktiv definierten Strukturen, ist es leichter, eine Rekursionsformel aufzustellen, die dann gelöst werden soll.

Definition 6.13 (erzeugende Funktion)**.** Für eine Folge $f\colon \mathbb{N} \to \mathbb{R}$ heißt die Potenzreihe

$$F(x) := \sum_{n=0}^{\infty} f(n)x^n$$

die *erzeugende Funktion* von f. Die Methode der erzeugenden Funktionen versucht, aus den Rekursionsformeln für $f(n)$ Gleichungen für $F(x)$ herzuleiten und diese mit algebraischen oder analytischen Mitteln zu lösen.

Beispiel 6.13. Die *Fibonacci-Zahlen* sind rekursiv definiert durch

$$f(n) = \begin{cases} 0 & \text{falls } n = 0 \\ 1 & \text{falls } n = 1 \\ f(n-1) + f(n-2) & \text{falls } n \geq 2. \end{cases}$$

Für die erzeugende Funktion $F(x)$ folgt aus obiger Rekursion die Gleichung

$$\begin{aligned} F(x) &= \sum_{n=0}^{\infty} f(n)x^n = f(0) + f(1)x + \sum_{n=2}^{\infty} \left(f(n-1) + f(n-2)\right) x^n \\ &= x + x \cdot F(x) + x^2 \cdot F(x) \end{aligned}$$

und durch Partialbruchzerlegung

$$F(x) = \frac{x}{1 - x - x^2} = \frac{1}{\sqrt{5}} \left(\frac{1}{1 - \frac{1+\sqrt{5}}{2} \cdot x} - \frac{1}{1 - \frac{1-\sqrt{5}}{2} \cdot x} \right).$$

Einsetzen der geometrischen Reihe

$$\frac{1}{1-x} = \sum_{n=0}^{\infty} x^n$$

liefert

$$F(x) = \frac{1}{\sqrt{5}} \left[\sum_{n=0}^{\infty} \left(\frac{1+\sqrt{5}}{2} \right)^n x^n - \sum_{n=0}^{\infty} \left(\frac{1-\sqrt{5}}{2} \right)^n x^n \right].$$

Koeffizientenvergleich ergibt die explizite Formel

$$f(n) = \frac{1}{\sqrt{5}} \left[\left(\frac{1+\sqrt{5}}{2} \right)^n - \left(\frac{1-\sqrt{5}}{2} \right)^n \right].$$

Beispiel 6.14. Die Menge der *binären Bäume* über der Menge M wird als formale Sprache mit Hilfe der Klammern „(“ und „)“ induktiv definiert:

(1) Die leere Zeichenkette ϵ ist ein binärer Baum.

(2) Wenn $x \in M$ und L, R binäre Bäume sind, dann ist (LxR) ein binärer Baum mit Knoten x.

Wir nennen binäre Bäume strukturell gleich, wenn sie durch Umbezeichnen der Elemente von M ineinander übergehen, z.B.

$$((a)b(c)) \text{ und } ((c)a(b))\,,$$

nicht aber

$$((a)b(c)) \text{ und } (a(b(c)))\,.$$

Offensichtlich ist strukturelle Gleichheit eine Äquivalenzrelation. Nach Konstruktion gilt für die Zahl $f(n)$ der Äquivalenzklassen binärer Bäume mit n Knoten die Rekursionsformel

$$f(n) = \begin{cases} 1 & \text{falls } n = 0 \\ \sum_{k=0}^{n-1} f(k) \cdot f(n-1-k) & \text{falls } n > 0. \end{cases}$$

Für die erzeugende Funktion $F(x)$ folgt

$$F(x) = \sum_{n=0}^{\infty} f(n)x^n = 1 + \sum_{n=1}^{\infty} \left(\sum_{k=0}^{n-1} f(k) \cdot f(n-1-k) \right) x^n = 1 + x \cdot F(x) \cdot F(x)$$

und

$$F(x)^2 - \frac{1}{x} \cdot F(x) + \frac{1}{x} = 0\,.$$

"Lösen der quadratischen Gleichung gibt

$$F(x) = (1 \pm \sqrt{1-4x})/2x\,.$$

Mit der Binomialreihe

$$\sqrt{1-4x} = (1-4x)^{1/2} = \sum_{n=0}^{\infty} \binom{1/2}{n} (-4x)^n = 1 - \sum_{n=1}^{\infty} \frac{2}{n} \binom{2n-2}{n-1} x^n$$

erhält man

$$F(x) = (1 - \sqrt{1-4x})/2x = \sum_{n=0}^{\infty} \frac{1}{n+1} \binom{2n}{n} x^n\,.$$

Koeffizientenvergleich liefert

$$f(n) = \frac{1}{n+1} \binom{2n}{n}\,.$$

Beispiel 6.15 (Divide-and-Conquer-Algorithmen)**.** Ein Algorithmus für ein bestimmtes Problem löst Instanzen bis zur Größe m direkt. Eine Instanz der Größe $n > m$ hingegen zerlegt der Algorithmus in a Teilinstanzen mit Größen

$$\lfloor n/b \rfloor \quad \text{oder} \quad \lceil n/b \rceil ,$$

löst diese rekursiv und setzt die Teillösungen zu einer Gesamtlösung zusammen. Dabei sind a und b Konstante mit $a \geq 1$ und $b > 1$, und $\lfloor\ \rfloor$ und $\lceil\ \rceil$ bezeichnen die Rundung nach unten bzw. oben. Die Zeit zum Aufteilen der Instanz und zum Zusammenfügen der Lösungen betrage $f(n)$, die Gesamtzeit sei $t(n)$. Eine natürliche Annahme ist, dass

$$t(n+1) \geq t(n)$$

für alle n gilt. Dann folgt

$$a \cdot t(\lfloor n/b \rfloor) + f(n) \leq t(n) \leq a \cdot t(\lceil n/b \rceil) + f(n) .$$

Mit den rekursiv definierten Funktionen

$$t^-(n) := \begin{cases} a \cdot t^-(\lfloor n/b \rfloor) + f(n) & \text{falls } n > m \\ t(n) & \text{falls } n \leq m \end{cases}$$

und

$$t^+(n) := \begin{cases} a \cdot t^+(\lceil n/b \rceil) + f(n) & \text{falls } n > m \\ t(n) & \text{falls } n \leq m \end{cases}$$

ist

$$t^-(n) \leq t(n) \leq t^+(n)$$

für alle n, sodass für die asymptotische Analyse der Laufzeit die Funktionen $t^{\pm}(n)$ an Stelle von $t(n)$ verwendet werden können.

Im Spezialfall $n = m \cdot b^k$ teilt der Algorithmus k-mal, bevor er auf Basisinstanzen der Größe m trifft. Die Gesamtzahl der Basisinstanzen ist

$$a^k = (b^r)^k = (b^k)^r = m^{-r} \cdot n^r ,$$

wobei

$$r := \log_b a .$$

Da jede Basisinstanz in konstanter Zeit gelöst werden kann, liegt die Zeit für die Lösung aller Basisinstanzen in

$$\Theta(n^r) ,$$

was auch allgemein für $t^{\pm}(n)$ gilt [1]. Im folgenden Satz wird diese Zeit mit der Zeit $f(n)$ zum Aufteilen und Zusammenfügen verglichen und entsprechend das asymptotische Wachstum von $t(n)$ angegeben.

Satz 6.17 (Master-Theorem)**.** *(1) Wenn $f \in \mathrm{O}(n^s)$ für eine reelle Zahl s mit $s < r$, dann ist*

$$t(n) \in \Theta(n^r) .$$

(2) Wenn $f \in \Theta(n^r)$ *, dann ist*

$$t(n) \in \Theta(n^r \cdot \log n)\,.$$

(3) Wenn eine reelle Zahl c *mit* $c < 1$ *und eine natürliche Zahl* k *existieren, sodass*

$$a \cdot f(\lceil n/b \rceil) \leq cf(n)$$

für alle n *mit* $n \geq k$ *, dann ist* $f \in \Omega(n^s)$ *für eine reelle Zahl* s *mit* $s > r$ *und*

$$t(n) \in \Theta(f)\,.$$

Beweis. [1]. □

Beispiel 6.16. Für

$$f(n) = n^r \log n$$

liegt das Wachstum von f zwischen Fall (2) und (3), somit ist das Master-Theorem nicht anwendbar.

Beispiel 6.17. Der Merge-Sort-Algorithmus zerlegt ein n-Tupel von Zahlen in zwei Tupel mit ungefähr $n/2$ Zahlen, sortiert getrennt und mischt die sortierten Tupel zusammen. Wegen $a = b = 2$ und $f \in \Theta(n)$ gibt das Master-Theorem die Laufzeitabschätzung

$$t(n) \in \Theta(n \cdot \log n)\,.$$

6.5 Aufgaben

Aufgabe 6.1. Wie lautet die Siebformel (spezialisiert) für drei Mengen? 80 Studierende der Informatik werden bezüglich Ihres Klausurverhaltens befragt. Von diesen nahmen 30 an der Klausur A teil, 26 an der Klausur B und 20 an der Klausur C. 10 Personen nahmen an A und B teil, 12 Personen an A und C und 9 Personen an B und C. Acht Studierende nahmen an keiner Klausur teil. Wie viele Studierende nahmen an genau einer bzw. zwei bzw. drei Klausuren teil? Können die Angaben der Studierenden stimmen?

Aufgabe 6.2. Wie lautet die Siebformel (spezialisiert) für drei Mengen? 80 Studierende der Informatik werden bezüglich Ihres Klausurverhaltens befragt. Von diesen nahmen 30 an der Klausur A teil, 26 an der Klausur B und 35 an der Klausur C. 10 Personen nahmen an A und B teil, 5 Personen an A und C und 9 Personen an B und C. Acht Studierende nahmen an keiner Klausur teil. Wie viele Studierende nahmen an genau einer bzw. zwei bzw. drei Klausuren teil?

Aufgabe 6.3. Eine Anzahl von Menschen wird gebeten 5 Gegenstände aus einer Liste von 8 möglichen Gegenständen auszuwählen, die sie auf eine einsame Insel mitnehmen würden. Die Nennungen der einzelnen Gegenstände werden mit

$$23, 20, 12, 10, 8, 5, 3, 1$$

angegeben. Visualisieren Sie die Zuteilung durch einen bipartiten Graphen. Welcher Schluss kann daraus gezogen werden?

Aufgabe 6.4. In einem Hörsaal mit 9 Sitzplätzen findet eine Klausur mit 5 Studierenden statt. Wie viele verschiedene Sitzordnungen gibt es?

Wie viele verschiedene Sitzordnungen gibt es, wenn diese 9 Sitzplätze in einer Reihe liegen und jeweils ein Platz zwischen zwei Studierenden frei bleiben soll?

Aufgabe 6.5. Für eine mündliche Prüfung haben sich 100 Studierende angemeldet. Der Vortragende möchte die Fragen an einer Gruppe von 5 Studierenden testen. Wie viele mögliche Zusammenstellungen gibt es? Vergleichen Sie die Anzahl mit den Möglichkeiten im Lotto (6 aus 45).

Aufgabe 6.6. Beweisen Sie. Die Menge $\mathbb{Q}$ der rationalen Zahlen ist abzählbar unendlich. Welche Sätze der Vorlesung sind nützlich?

Hinweis: Betrachten Sie die Abbildung

$$q\colon \mathbb{Z} \times (\mathbb{N} \setminus \{0\}) \to \mathbb{Q}\,,\ (a,b) \mapsto \frac{a}{b}\,.$$

Aufgabe 6.7. Beweisen Sie. Die Menge $\mathbb{R}$ der reellen Zahlen ist nicht abzählbar unendlich. Welche Sätze der Vorlesung sind nützlich?

Aufgabe 6.8. Bestimmen Sie die ersten 20 Elemente der Menge $\mathbb{N}^3$ bezüglich der graduiert-lexikographischen Ordnung auf Zahlentupeln. Ist die Menge $\mathbb{N}^3$ abzählbar unendlich?

Aufgabe 6.9. Beweisen oder widerlegen Sie: Sei A eine unendliche Menge, dann ist A^* abzählbar.

Aufgabe 6.10. Konstruieren Sie nach dem Beweis von Satz 6.11 für die injektiven Abbildungen

$$f\colon \{0,1,2,3\} \to \{0,1,2,3\} \text{ und } g\colon \{0,1,2,3\} \to \{0,1,2,3\}$$

mit $f(0) = 1, f(1) = 2, f(2) = 3, f(3) = 0$ und $g(0) = 2, g(1) = 3, g(2) = 0, g(3) = 1$ eine bijektive Abbildung $h\colon \{0,1,2,3\} \to \{0,1,2,3\}$.

Aufgabe 6.11. Konstruieren Sie nach dem Beweis von Satz 6.11 für die injektiven Abbildungen

$$f\colon \mathbb{N} \to \mathbb{N} \text{ mit } m \mapsto 2m$$

$$g\colon \mathbb{N} \to \mathbb{N} \text{ mit } n \mapsto 2n$$

die Mengen $M_0, N_0, M_1, N_1, M_2, N_2$ und skizzieren Sie M', N' sowie g'.

Aufgabe 6.12. Sei $G = (E, K, r)$ ein ungerichteter Graph mit Kantenbewertung b. Wie viele verschiedene spannende Wälder mit minimaler Kantenbewertung gibt es (maximal)?

Bonus: Wie viele verschiedene spannende Wälder gibt es, wenn G aus n Zusammenhangskomponenten besteht?

Aufgabe 6.13. Satz 6.7 liefert zwei Möglichkeiten, um den Binomialkoeffizienten $\binom{m}{k}$ zu berechnen. Welche bevorzugen Sie und warum? Implementieren Sie beide Funktionen und vergleichen Sie die Resultate.

Aufgabe 6.14. Welche Nummer bekommt das Paar $(7, 10)$ gemäß der Nummerierung von Satz 6.8? In welcher Diagonale liegt es?

7

Zahlentheorie

7.1 Rechnen mit ganzen Zahlen

Neben den üblichen Zifferndarstellungen zur Basis 10 (Mensch) oder Basis 2 (Hardware) werden in der Software Basen verwendet, bei denen die Ziffern im nativen Ganzzahlformat abgespeichert werden können. Zum Beispiel verwendet die Langzahl-Bibliothek

GNU Multiple Precision Arithmetic Library (`http://gmplib.org/`)

die Basen $2^{32} \approx 4 \cdot 10^9$ oder $2^{64} \approx 2 \cdot 10^{19}$, damit die Ziffern in die Datentypen 32- bzw. 64-bit unsigned integer von C passen.

Definition 7.1 (Ziffernoperationen)**.** Sei b eine natürliche Zahl ≥ 2. Für Zahlen $u, v \in \{0, 1, \ldots, b-1\}$ und $w \in \{0, 1\}$ heißt

$$C(u, v, w) := \begin{cases} 0 & \text{falls } u + v + w < b \\ 1 & \text{sonst} \end{cases}$$

der *Übertrag* (carry) modulo b und

$$S(u, v, w) := \begin{cases} u + v + w & \text{falls } u + v + w < b \\ u + v + w - b & \text{sonst} \end{cases}$$

die *Summe* modulo b. Offenbar gilt

$$S(u, v, w) \in \{0, 1, \ldots, b-1\}$$

und

$$u + v + w = C(u, v, w) \cdot b + S(u, v, w).$$

Satz 7.1 (Addition natürlicher Zahlen in Zifferndarstellung)**.** *Es seien x und y natürliche Zahlen mit Ziffern*

$$x_k x_{k-1} \cdots x_0 \quad \textit{bzw.} \quad y_\ell y_{\ell-1} \cdots y_0$$

zur Basis $b \geq 2$. Ohne Einschränkung der Allgemeinheit nehmen wir $k \geq \ell$ an und setzen

$$y_{\ell+1} := 0\,,\ y_{\ell+2} := 0\,, \ldots,\ y_k := 0\,.$$

Dann können die Ziffern der Summe $x + y$ durch den folgenden Algorithmus mit $\mathrm{O}(k)$ Ziffernoperationen berechnet werden:

Setze $c_0 = 0$.
Für i *von* 0 *bis* k *wiederhole:*
Setze $z_i = S(x_i, y_i, c_i)$.
Setze $c_{i+1} = C(x_i, y_i, c_i)$.
Falls $c_{k+1} \neq 0$, *setze* $z_{k+1} = c_{k+1}$.

Beweis. Wir führen eine Induktion nach k. *Um Fallunterscheidungen zu vermeiden, setzen wir im letzten Schritt des Algorithmus* $z_{k+1} = 0$, *falls* $c_{k+1} = 0$. *Für* $k = 0$ *ergibt sich*

$$z_1 b + z_0 = C(x_0, y_0, 0)b + S(x_0, y_0, 0) = x_0 + y_0 + 0 = x + y\,.$$

Für den Induktionsschluss seien

$$x = \sum_{i \le k+1} x_i b^i, \quad y = \sum_{i \le k+1} y_i b^i$$

und z_0, z_1,...,z_{k+2} *die Ziffern aus dem Algorithmus. Nach Induktionshypothese gilt*

$$\sum_{i \le k} x_i b^i + \sum_{i \le k} y_i b^i = \sum_{i \le k} z_i b^i + c_{k+1} b^{k+1}\,.$$

Damit folgt

$$\begin{aligned}\sum_{i \le k+1} x_i b^i + \sum_{i \le k+1} y_i b^i &= \sum_{i \le k} z_i b^i + (x_{k+1} + y_{k+1} + c_{k+1}) b^{k+1} = \\ &\sum_{i \le k} z_i b^i + (S(x_{k+1}, y_{k+1}, c_{k+1}) + C(x_{k+1}, y_{k+1}, c_{k+1}) b)\, b^{k+1} = \\ &\sum_{i \le k} z_i b^i + z_{k+1} b^{k+1} + z_{k+2} b^{k+2} = \sum_{i \le k+2} z_i b^i\,.\end{aligned}$$

□

Für die Subtraktion kann analog ein Algorithmus mit $\mathrm{O}(\max(k, \ell))$ Ziffernoperationen angegeben werden, für die Multiplikation oder die Division mit Rest Algorithmen mit $\mathrm{O}(k \cdot \ell)$ Ziffernoperationen [6]. Hier wird noch ein Algorithmus für das Potenzieren diskutiert.

Satz 7.2 (schnelles Potenzieren)**.** *Sei* x *eine ganze Zahl oder allgemeiner ein Element eines Ringes, und sei* e *eine positive ganze Zahl mit Binärziffern*

$$e_t e_{t-1} \cdots e_0\,,$$

wobei $e_t = 1$. *Dann kann die Potenz* $y := x^e$ *durch* t*-faches Quadrieren und maximal* t*-faches Multiplizieren berechnet werden:*

Setze $y = x$.
Für i *von* $t-1$ *hinab bis* 0 *wiederhole:*
Setze $y = y^2$.
Falls $e_i = 1$, *setze* $y = y * x$.

Beweis. Für $t = 0$ ist $e = 1$ und der Algorithmus liefert $x^1 = x$. Für $t > 0$ schreiben wir

$$e = \sum_{i=0}^{t} e_i 2^i = d \cdot 2 + e_0 \quad \textit{mit} \quad d = \sum_{i=1}^{t} e_i 2^{i-1}.$$

Nach Induktionshypothese hat y vor dem letzten Schleifendurchlauf den Wert x^d. Daher ist das Resultat

$$(x^d)^2 \cdot x^{e_0} = x^e \,.$$

□

Beispiel 7.1. Es gilt: $3^9 = 3^8 \cdot 3^1 = ((3^2)^2)^2 \cdot 3 = 19683$. Für die Berechnung sind 4 Multiplikationen nötig, davon 3 für das Quadrieren.

7.2 Der euklidische Algorithmus

Seien a, b, c ganze Zahlen mit $b \neq 0$ und $c \neq 0$. Dann ist

$$\frac{a}{b} = \frac{a \cdot c}{b \cdot c} \in \mathbb{Q}$$

eine rationale Zahl (siehe auch [9]). Der Übergang von der Darstellung durch das Zahlenpaar $(a \cdot c, b \cdot c)$ zu der durch das Zahlenpaar (a, b) heißt

durch c kürzen.

Rechnet man mit rationalen Zahlen, dann ist es empfehlenswert sofort durch möglichst große Zahlen zu kürzen, um Zähler und Nenner klein zu halten. In diesem Abschnitt wird ein Verfahren zum „optimalen Kürzen“ vorgestellt.

Definition 7.2 (Teiler, Vielfaches)**.** Sei a eine ganze Zahl. Eine ganze Zahl d heißt ein *Teiler* von a, wenn es eine ganz Zahl c gibt mit

$$a = c \cdot d \,.$$

Äquivalente Sprechweisen sind „d teilt a“ oder „a ist *Vielfaches* von d“. Die Teiler $\pm 1, \pm a$ nennt man *triviale Teiler*.

Definition 7.3 (größter gemeinsamer Teiler, kleinstes gemeinsames Vielfaches)**.** Seien a und b ganze Zahlen ungleich Null.

– Der *größte gemeinsame Teiler* von a und b ist die größte ganze Zahl, die sowohl a als auch b teilt, und wird mit

$$\mathrm{ggT}(a, b) \quad (\gcd(a, b), \text{ greatest common divisor})$$

bezeichnet.

– Das *kleinste gemeinsame Vielfache* von a und b ist die kleinste positive ganze Zahl, die sowohl Vielfaches von a als auch Vielfaches von b ist, und wird mit

$$\mathrm{kgV}(a,b) \quad (\mathrm{lcm}(a,b), \text{ least common multiple})$$

bezeichnet.

Satz 7.3 (Reduktion des größten gemeinsamen Teilers)**.** *Seien $a, b, c \in \mathbb{Z}$ mit $a \neq 0$, $b \neq 0$ und $a \neq c \cdot b$. Dann gilt*

$$\mathrm{ggT}(a,b) = \mathrm{ggT}(|a|,|b|) \quad \textit{und} \quad \mathrm{ggT}(a,b) = \mathrm{ggT}(a - c \cdot b, b)\,.$$

Beweis. Wenn eine ganze Zahl d die Zahl a teilt, dann auch die Zahl $-a$. Somit stimmen die gemeinsamen Teiler von a und b mit den gemeinsamen Teilern von $|a|$ und $|b|$ überein, und die größten gemeinsamen Teiler sind gleich.

Wenn eine ganze Zahl d die Zahlen a und b teilt, dann teilt sie auch die Zahl $a - c \cdot b$. Somit stimmen die gemeinsamen Teiler von a und b mit den gemeinsamen Teilern von $a - c \cdot b$ und b überein, und die größten gemeinsamen Teiler sind gleich. □

Satz 7.4 (euklidischer Algorithmus für ganze Zahlen)**.** *Der größte gemeinsame Teiler zweier ganzer Zahlen ungleich Null kann wie folgt berechnet werden:*

> *Ersetze die Zahlen durch ihre Beträge.*
>
> *Solange die Zahlen verschieden sind, wiederhole:*
>
> *Ersetze die größere der Zahlen durch die Differenz*
>
> *der größeren und der kleineren.*
>
> *Die resultierende Zahl ist dann der größte gemeinsame Teiler.*

Ersetzt man mehrfaches Abziehen derselben Zahl durch eine Division mit Rest, dann bekommt dieses Verfahren die folgende Form:

> *Ersetze die Zahlen durch ihre Beträge.*
>
> *Solange keine der zwei Zahlen ein Teiler der anderen ist, wiederhole:*
>
> *Ersetze die größere der Zahlen durch ihren Rest*
>
> *nach Division durch die kleinere.*
>
> *Der resultierende Teiler ist dann der größte gemeinsame Teiler.*

Beweis. Nach Satz 7.3 bleiben bei jedem Schritt die größten gemeinsamen Teiler der beiden Zahlen gleich. Da in jedem Schleifendurchlauf die Zahlen positiv bleiben, aber ihr Maximum um mindestens 1 sinkt, bricht der Algorithmus nach endlich vielen Schritten ab. □

Beispiel 7.2. Es gilt $\mathrm{ggT}(138, -48) = 6$.

Im euklidischen Algorithmus wird die folgende Strategie zur Lösung von Problemen verwendet: Wenn man ein gegebenes Problem nicht sofort lösen kann, reduziert man das Problem auf ein einfacheres mit derselben Lösungsmenge. Das wiederholt man solange, bis man ein Problem bekommt, dessen Lösungen man kennt. Diese Lösungen sind dann auch die Lösungen des ursprünglichen Problems.

Satz 7.5 (erweiterter euklidischer Algorithmus)**.** *Seien a und b ganze Zahlen ungleich Null. Dann gibt es ganze Zahlen u und v mit*

$$u \cdot a + v \cdot b = \mathrm{ggT}(a, b)$$

*(*Darstellung des größten gemeinsamen Teilers als Linearkombination*). Diese Zahlen u und v können durch folgenden Algorithmus berechnet werden:*

Setze $A = (\lvert a\rvert, 1, 0)$ *und* $B = (\lvert b\rvert, 0, 1)$.
Solange B_1 *die Zahl* A_1 *nicht teilt, wiederhole:*
Berechne den ganzzahligen Quotienten q *von* A_1 *und* B_1.
Setze $C = B$.
Setze $B = A - q \cdot C$.
Setze $A = C$.
Setze $u = \mathrm{vz}(a) \cdot B_2$ *und* $v = \mathrm{vz}(b) \cdot B_3$.

Dabei sind $A = (A_1, A_2, A_3)$, $B = (B_1, B_2, B_3)$, $C = (C_1, C_2, C_3)$ *Tripel ganzer Zahlen und* $A - q \cdot C = (A_1 - q \cdot C_1, A_2 - q \cdot C_2, A_3 - q \cdot C_3)$. *Weiters bezeichnet* $\mathrm{vz}(n)$ *das Vorzeichen einer ganzen Zahl* n, *i.e.,* $\mathrm{vz}(n) = 1$, *wenn* $n \geq 0$ *und* -1 *sonst.*

Beweis. Sei $T = (T_1, T_2, T_3)$ *ein Tripel ganzer Zahlen und* $(*)$ *die Eigenschaft*

$$T_1 = |a| \cdot T_2 + |b| \cdot T_3\,. \tag{*}$$

Wenn die Zahlentripel A *und* B *die Eigenschaft* $(*)$ *haben, dann auch alle Tripel* $A - q \cdot B$ *mit* $q \in \mathbb{Z}$. *Die ersten zwei Tripel im Algorithmus haben diese Eigenschaft, daher auch alle anderen auftretenden Tripel. In der ersten Komponente der Tripel wird der euklidische Algorithmus durchgeführt, für das letzte Tripel* B *gilt daher*

$$\mathrm{ggT}(a, b) = B_1 = |a| \cdot B_2 + |b| \cdot B_3 = (\underbrace{\mathrm{vz}(a) \cdot B_2}_{u}) \cdot a + (\underbrace{\mathrm{vz}(b) \cdot B_3}_{v}) \cdot b\,.$$

□

Beispiel 7.3. Für $a = 138$ und $b = -48$ liefert der erweiterte euklidische Algorithmus $u = -1$ und $v = -3$ und $\mathrm{ggT}(138, -48) = 6$.

Satz 7.6 (binärer erweiterter euklidischer Algorithmus)**.** *Seien a und b positive ganze Zahlen mit höchstens n Binärziffern. Dann können*

$$g := \mathrm{ggT}(a, b) \quad \textit{und} \quad u, v \in \mathbb{Z} \quad \textit{mit} \quad u \cdot a + v \cdot b = g$$

durch Additionen, Subtraktionen und Shifts mit $\mathrm{O}(n^2)$ *Bitoperationen berechnet werden:*

1. *Setze* $e = 0$.
2. *Solange* a *und* b *gerade sind, wiederhole:*
   Setze $a = a/2$, $b = b/2$ *und* $e = e + 1$.
3. *Setze* $A = (a, 1, 0)$ *und* $B = (b, 0, 1)$.
4. *Solange* A_1 *gerade ist, wiederhole:*
   Setze $A_1 = A_1/2$.
   Falls A_2 *und* A_3 *gerade sind,*
   setze $A_2 = A_2/2$ *und* $A_3 = A_3/2$,
   ansonsten falls $A_2 > 0$ *ist,*
   setze $A_2 = (A_2 - b)/2$ *und* $A_3 = (A_3 + a)/2$,
   ansonsten
   setze $A_2 = (A_2 + b)/2$ *und* $A_3 = (A_3 - a)/2$.
5. *Solange* B_1 *gerade ist, wiederhole:*
   Setze $B_1 = B_1/2$.
   Wenn B_2 *und* B_3 *gerade sind,*
   setze $B_2 = B_2/2$ *und* $B_3 = B_3/2$,
   ansonsten falls $B_2 > 0$ *ist,*
   setze $B_2 = (B_2 - b)/2$ *und* $B_3 = (B_3 + a)/2$,
   ansonsten
   setze $B_2 = (B_2 + b)/2$ *und* $B_3 = (B_3 - a)/2$.
6. *Falls* $A_1 > B_1$, *setze* $A = A - B$ *und gehe zu Schritt 4.*
7. *Falls* $A_1 < B_1$, *setze* $B = B - A$ *und gehe zu Schritt 5.*
8. *Setze* $g = B_1 \cdot 2^e$, $u = B_2$ *und* $v = B_3$.

Beweis. Bei den Reduktionen

$$\mathrm{ggT}(a,b) = \begin{cases} 2\,\mathrm{ggT}(a/2, b/2) & \text{falls } a \text{ und } b \text{ gerade sind} \\ \mathrm{ggT}(a/2, b) & \text{falls } a \text{ gerade und } b \text{ ungerade ist} \\ \mathrm{ggT}(a, b/2) & \text{falls } a \text{ ungerade und } b \text{ gerade ist} \\ \mathrm{ggT}((a-b)/2, b) & \text{falls } a \text{ und } b \text{ ungerade und } a > b \text{ ist} \\ \mathrm{ggT}(a, (b-a)/2) & \text{falls } a \text{ und } b \text{ ungerade und } a < b \text{ ist} \end{cases}$$

sinkt die Summe der Bitlängen um mindestens 1, sodass höchstens $2n - 2$ *Reduktionen ausgeführt werden. Bei den Operationen für* A_2, A_3, B_2, B_3 *wird die Bitlänge* $n + 1$ *(plus Vorzeichenbit) nicht überschritten. Wenn* c *die Binärdarstellung*

$$c_k c_{k-1} \cdots c_1 c_0$$

hat, dann haben $2c$ *und* $c/2$ *die Binärdarstellungen*

$$c_k c_{k-1} \cdots c_1 c_0 0 \quad \text{bzw.} \quad c_k c_{k-1} \cdots c_1\,,$$

sodass Multiplikationen mit 2 oder Divisionen durch 2 als Shifts ausgeführt werden können. Somit haben alle arithmetische Operationen lineare Komplexität, was quadratische Komplexität für den Algorithmus ergibt.

In den ersten zwei Schritten wird auf den Fall a oder b ungerade reduziert. Mit den neuen Werten a', b' und

$$g' := \mathrm{ggT}(a', b')$$

folgt dann in Schritt 8

$$g = \mathrm{ggT}(a, b) = \mathrm{ggT}(a' \cdot 2^e, b' \cdot 2^e) = g' \cdot 2^e \,.$$

Zur Analyse der Schritte 3-7 betrachten wir die Eigenschaft

$$T_1 = a' \cdot T_2 + b' \cdot T_3 \tag{*}$$

eines Tripels ganzer Zahlen $T = (T_1, T_2, T_3)$. Die Startwerte für A und B in Schritt 3 haben diese Eigenschaft. Wenn A und B die Eigenschaft $(*)$ besitzen, dann auch die Tripel

$$A - B \quad \text{bzw.} \quad B - A$$

in Schritt 6 bzw. 7. Wenn A_1, A_2 und A_3 gerade sind, dann hat auch das Tripel

$$(A_1/2, A_2/2, A_3/2)$$

in Schritt 4 die Eigenschaft $(*)$. Wenn A_1 gerade ist, A_2 oder A_3 ungerade sind und $A_2 > 0$ ist, dann ist $A_3 \leq 0$, sowohl $A_2 - b'$ als auch $A_3 + a'$ gerade und das Tripel

$$(A_1/2, (A_2 - b')/2, (A_3 + a')/2)$$

in Schritt 4 hat wieder die Eigenschaft $(*)$. Analoges gilt im Fall $A_2 \leq 0$, wo $A_3 > 0$ ist. Schließlich gilt für $A_1 = B_1$ in Schritt 8

$$g' = B_1 = a' \cdot B_2 + b' \cdot B_3$$

und

$$g = B_2 \cdot a + B_3 \cdot b \,.$$

□

Beispiel 7.4. Für $a = 138$ und $b = 48$ liefert der binäre erweiterte euklidische Algorithmus $e = 1$, $g = 3 \cdot 2^1$, $u = -9$, $v = 26$ und $\mathrm{ggT}(138, 48) = 6$.

Satz 7.7 (Berechnung des kleinsten gemeinsamen Vielfachen)**.** *Seien a und b ganze Zahlen ungleich Null. Dann gilt*

$$\mathrm{kgV}(a, b) = \frac{|a| \cdot |b|}{\mathrm{ggT}(a, b)} \,.$$

Beweis. Offensichtlich ist

$$m := \frac{|b|}{\mathrm{ggT}(a,b)} \cdot |a| = \frac{|a|}{\mathrm{ggT}(a,b)} \cdot |b|$$

ein Vielfaches sowohl von a als auch von b und somit ein gemeinsames Vielfaches. *Wir zeigen, dass m das* kleinste *gemeinsame Vielfache von a und b ist. Sei dazu eine positive ganze Zahl z gemeinsames Vielfaches von a und b. Dann gibt es ganze Zahlen c, d mit*

$$z = c \cdot a \quad \text{und} \quad z = d \cdot b\,.$$

Nach Satz 7.5 existieren Zahlen u, v mit

$$u \cdot a + v \cdot b = \mathrm{ggT}(a,b)\,.$$

Es folgt

$$z = \frac{u \cdot a + v \cdot b}{\mathrm{ggT}(a,b)} \cdot z = \frac{u \cdot a}{\mathrm{ggT}(a,b)} \cdot z + \frac{v \cdot b}{\mathrm{ggT}(a,b)} \cdot z = \frac{u \cdot a \cdot d \cdot b}{\mathrm{ggT}(a,b)} + \frac{v \cdot b \cdot c \cdot a}{\mathrm{ggT}(a,b)} = \frac{a \cdot b}{\mathrm{ggT}(a,b)} \cdot (u \cdot d + v \cdot c) = m \cdot \mathrm{vz}(a \cdot b) \cdot (u \cdot d + v \cdot c)$$

und damit ist z ein Vielfaches von m. Aus $z, m > 0$ folgt $\mathrm{vz}(a \cdot b) \cdot (u \cdot d + v \cdot c) > 0$, somit $z \geqslant m$ und wegen der Totalität von $>$ ist m das kleinste gemeinsame Vielfache von a und b. □

7.3 Primzahlen

Definition 7.4 (Primzahl)**.** Eine natürliche Zahl p heißt *Primzahl*, wenn $p \notin \{0, 1\}$ und p nur die trivialen Teiler besitzt.

Satz 7.8 (Primzahl teilt Faktor)**.** *Sei p eine Primzahl und seien $a, b \in \mathbb{Z}$. Wenn p das Produkt $a \cdot b$ teilt, dann teilt p auch einen der Faktoren a oder b.*

Beweis. Sei $c \in \mathbb{Z}$ mit

$$c \cdot p = a \cdot b\,.$$

Wenn p die Zahl a teilt, sind wir fertig. Wenn p die Zahl a nicht teilt, dann ist $\mathrm{ggT}(a,p) = 1$. Daher gibt es ganze Zahlen u und v mit

$$1 = u \cdot a + v \cdot p\,.$$

Es folgt

$$b = b \cdot u \cdot a + b \cdot v \cdot p = u \cdot c \cdot p + b \cdot v \cdot p = (u \cdot c + b \cdot v) \cdot p\,,$$

somit ist p ein Teiler von b. □

Satz 7.9 (Primfaktorzerlegung)**.** *Jede ganze Zahl größer als 1 kann als Produkt von Primzahlen geschrieben werden. Diese Primzahlen heißen* Primfaktoren *der Zahl und sind bis auf die Reihenfolge eindeutig bestimmt.*

Beweis. Es sei a eine ganze Zahl größer als 1. Wir zeigen die Existenz der Primfaktorzerlegung durch Induktion nach a. Wenn $a = 2$, dann ist a eine Primzahl. Wenn $a > 2$, dann ist a entweder eine Primzahl oder es gibt ganze Zahlen b, c mit

$$a = b \cdot c \quad \text{und} \quad 1 < b < a \quad \text{und} \quad 1 < c < a\,.$$

Nach Induktionsannahme sind die Zahlen b und c Produkte von Primzahlen, somit auch die Zahl a.

Wir beweisen noch die Eindeutigkeit der Primfaktorzerlegung. Seien

$$a = p_1 p_2 \cdots p_k = q_1 q_2 \cdots q_\ell$$

zwei Zerlegungen von a in Primfaktoren. Wir zeigen durch wohlfundierte Induktion, dass die Primfaktoren der zwei Zerlegungen bis auf die Reihenfolge gleich sind. Da p_1 das Produkt $q_1 q_2 \cdots q_\ell$ teilt, gibt es nach Satz 7.8 eine Zahl $j \in \{1, \ldots, \ell\}$ mit $p_1 = q_j$. Somit ist

$$p_2 \cdots p_k = \prod_{\substack{1 \le i \le \ell \\ i \ne j}} q_i\,,$$

und die Behauptung folgt aus der Induktionsannahme. □

Während der größte gemeinsame Teiler mit dem (binären) euklidischen Algorithmus schnell berechnet werden kann, ist die Berechnung der Primfaktorzerlegung im Allgemeinen aufwändig. Daher sollten Rechenverfahren, in denen Zahlen in Primfaktoren zerlegt werden müssen, nach Möglichkeit vermieden werden. Umgekehrt ist die RSA-Verschlüsselung sicher, weil (derzeit) keine effizienten Verfahren zur Faktorisierung von großen Zahlen bekannt sind.

Satz 7.10. *Es gibt unendlich viele Primzahlen.*

Beweis. Wir nehmen an, die Zahlen $p_1, p_2, \ldots, p_n$ wären sämtliche Primzahlen. Sei

$$q := \prod_{i=1}^{n} p_i\,.$$

Dann ist $q + 1$ größer als jede Primzahl und somit keine Primzahl. Nach Satz 7.9 gibt es eine Primzahl p_j, die $q + 1$ teilt. Da p_j auch q teilt, würde p_j auch 1 teilen, was im Widerspruch zur Primheit von p_j steht. □

Satz 7.11 (Berechnung von ggT und kgV aus der Primfaktorzerlegung)**.** *Seien a und b positive ganze Zahlen mit Primfaktorzerlegungen*

$$a = \prod_{i=1}^{n} p_i^{e_i} \quad \text{und} \quad b = \prod_{i=1}^{n} p_i^{f_i}\,,$$

wobei $p_1, \ldots, p_n$ paarweise verschiedene Primzahlen und $e_1, \ldots, e_n, f_1, \ldots, f_n$ natürliche Zahlen sind. Dann gilt

$$\mathrm{ggT}(a, b) = \prod_{i=1}^{n} p_i^{\min(e_i, f_i)} \quad \text{und} \quad \mathrm{kgV}(a, b) = \prod_{i=1}^{n} p_i^{\max(e_i, f_i)}\,,$$

wobei $\min(e_i, f_i)$ bzw. $\max(e_i, f_i)$ die kleinere bzw. die größere der zwei Zahlen e_i und f_i bezeichnet.

Beweis. Die Zahl

$$d := \prod_{i=1}^{n} p_i^{\min(e_i, f_i)}.$$

teilt offenbar die Zahlen a und b. Da nach Satz 7.9 die Primfaktorzerlegungen von a und b eindeutig sind, kann ihr größter gemeinsamer Teiler keine anderen Primfaktoren als $p_1, \ldots, p_n$ enthalten. Somit darf p_i in $\mathrm{ggT}(a, b)$ nur $\min(e_i, f_i)$-mal auftreten. Daher ist $d = \mathrm{ggT}(a, b)$. Die Behauptung für $\mathrm{kgV}(a, b)$ folgt aus Satz 7.7. □

7.4 Restklassen

Definition 7.5 (Kongruenz, Restklassen)**.** Sei n eine positive ganze Zahl. Zwei ganze Zahlen a, b heißen *kongruent modulo n*, in Zeichen

$$a \equiv_n b\,,$$

wenn ihre Reste nach Division durch n

$$a \bmod n \quad \text{und} \quad b \bmod n$$

gleich sind. Kongruenz modulo n ist eine Äquivalenzrelation (siehe Aufgabe 7.7). Die Äquivalenzklasse einer ganzen Zahl a ist

$$\overline{a} := \{a + z \cdot n \mid z \in \mathbb{Z}\}$$

und wird die *Restklasse* von a modulo n genannt. Die Menge aller Restklassen modulo n wird mit

$$\mathbb{Z}/n \qquad \text{(Sprechweise: } \mathbb{Z} \text{ modulo } n\text{)}$$

bezeichnet. Als Repräsentantensysteme verwenden wir üblicherweise die *kleinsten nichtnegativen Reste*

$$\{0, 1, 2, \ldots, n-1\}$$

oder die *absolut-kleinsten Reste*

$$\begin{cases} \{-n/2+1, \ldots, -1, 0, 1, \ldots, n/2\} & \text{falls } n \text{ gerade} \\ \{-(n-1)/2, \ldots, -1, 0, 1, \ldots, (n-1)/2\} & \text{falls } n \text{ ungerade.} \end{cases}$$

In der Literatur wird für $a \equiv_n b$ oft auch

$$a \equiv b \mod n$$

geschrieben.

Satz 7.12 (Restklassenarithmetik)**.** *Die Abbildungen*

$$+\colon \mathbb{Z}/n \times \mathbb{Z}/n \to \mathbb{Z}/n\,,\ (\overline{a}, \overline{b}) \mapsto \overline{a} + \overline{b} := \overline{a+b}\,,$$

und

$$\cdot\colon \mathbb{Z}/n \times \mathbb{Z}/n \to \mathbb{Z}/n\,,\ (\overline{a}, \overline{b}) \mapsto \overline{a} \cdot \overline{b} := \overline{a \cdot b}\,,$$

sind wohldefiniert. Mit diesen Rechenoperationen ist $\mathbb{Z}/n$ ein kommutativer Ring. Das Nullelement von $\mathbb{Z}/n$ ist $\overline{0}$, das Einselement ist $\overline{1}$.

Beweis. Wir zeigen zuerst, dass die Operationen wohldefiniert sind. Seien dazu $a, c, b, d \in \mathbb{Z}$. *Die Addition ist wohldefiniert, wenn aus*

$$\overline{a} = \overline{c} \quad \text{und} \quad \overline{b} = \overline{d} \quad \text{auch} \quad \overline{a+b} = \overline{c+d} \quad \text{folgt}\,.$$

Wenn $\overline{a} = \overline{c}$ *und* $\overline{b} = \overline{d}$, *dann sind* $a - c$ *und* $b - d$ *Vielfache von* n. *Wegen*

$$(a-c) + (b-d) = (a+b) - (c+d)$$

ist auch $(a+b) - (c+d)$ *ein Vielfaches von* n. *Somit gilt*

$$\overline{a+b} = \overline{c+d}\,,$$

was die Wohldefiniertheit von $+$ *beweist. Wegen*

$$ab - cd = a(b-d) + (a-c)d$$

ist

$$\overline{a \cdot b} = \overline{c \cdot d}\,,$$

was die Wohldefiniertheit von $\cdot$ *beweist. Da* $+$ *und* $\cdot$ *über die Addition und Multiplikation ganzer Zahlen definiert sind, erfüllen sie die Rechenregeln eines kommutativen Rings.* □

Beispiel 7.5. Es gilt $\mathbb{Z}/5 = \{\overline{0}, \overline{1}, \overline{2}, \overline{3}, \overline{4}\} = \{\overline{-2}, \overline{-1}, \overline{0}, \overline{1}, \overline{2}\}$. Weiters ist

$$\overline{0} = \{0, 5, 10, 15, \ldots\} = \overline{5}\,,$$

und $\overline{2} + \overline{4} = \overline{6} = \overline{1}$.

Beispiel 7.6. In der Programmiersprache C wird im Datentyp `unsigned int` im Restklassenring $\mathbb{Z}/n$ mit $n = 2^{32}$ bzw. $n = 2^{64}$ gerechnet. Auf einer 32-Bit Architektur ergibt die Summe von $2^{32} - 1$ und 1 dann 0.

Satz 7.13 (Invertierbarkeit von Restklassen)**.** *Sei* n *eine positive ganze Zahl und* a *eine ganze Zahl ungleich Null.*

(1) Die Restklasse von a *modulo* n *ist genau dann invertierbar, wenn*

$$\mathrm{ggT}(a, n) = 1\,.$$

In diesem Fall können mit dem erweiterten euklidischen Algorithmus ganze Zahlen u, v *mit* $u \cdot a + v \cdot n = 1$ *berechnet werden, und dann ist*

$$\overline{a}^{-1} = \overline{u}\,.$$

(2) Der Ring $\mathbb{Z}/n$ *ist genau dann ein Körper, wenn* n *eine Primzahl ist.*

Beweis. (1) Wenn $\mathrm{ggT}(a,n)=1$ *und* $u\cdot a+v\cdot n=1$ *ist, dann ist*

$$\overline{1}=\overline{u}\cdot\overline{a}+\overline{v}\cdot\overline{n}=\overline{u}\cdot\overline{a}+\overline{v}\cdot\overline{0}=\overline{u}\cdot\overline{a}\,.$$

Wenn umgekehrt $\overline{a}$ *invertierbar ist, dann gibt es eine ganze Zahl* b *mit*

$$\overline{a}\cdot\overline{b}=\overline{1}\quad\text{und}\quad\overline{ab-1}=\overline{0}\,.$$

Somit ist n *ein Teiler von* $ab-1$. *Da* $\mathrm{ggT}(a,n)$ *sowohl* a *als auch* $ab-1$ *teilt, ist* $\mathrm{ggT}(a,n)=1$. *(2) Sei* n *eine Primzahl und* a *eine ganze Zahl. Dann ist entweder* a *ein Vielfaches von* n *oder* $\mathrm{ggT}(a,n)=1$. *Somit folgt die Behauptung aus (1).* □

Beispiel 7.7. Die Zahl 6 ist nicht invertierbar modulo 26, das Inverse von 5 modulo 26 ist 21. Der Ring $\mathbb{Z}/2$ ist ein Körper mit zwei Elementen, der Ring $\mathbb{Z}/256$ ist kein Körper.

Satz 7.14 (der kleine Satz von Fermat)**.** *Sei* p *eine Primzahl und sei* a *eine ganze Zahl, die nicht von* p *geteilt wird. Dann gilt*

$$a^{p-1}\equiv_p 1\,.$$

Beweis. Die Restklassen

$$\overline{1a},\overline{2a},\ldots,\overline{(p-1)a}$$

sind alle von $\overline{0}$ *und untereinander verschieden und damit eine Permutation von*

$$\overline{1},\overline{2},\ldots,\overline{p-1}\,.$$

Somit ist

$$\overline{1\cdot 2\cdots(p-1)}\cdot\overline{1}=\overline{1a}\cdot\overline{2a}\cdots\overline{(p-1)a}=\overline{1\cdot 2\cdots(p-1)}\cdot\overline{a^{p-1}}\,.$$

Kürzen liefert das Ergebnis. □

Satz 7.15 (chinesischer Restsatz)**.** *Seien* p *und* q *positive ganze Zahlen mit* $\mathrm{ggT}(p,q)=1$, *und seien* a *und* b *beliebige ganze Zahlen. Dann hat das Kongruenzensystem*

$$\begin{aligned}x&\equiv_p a\\ x&\equiv_q b\end{aligned}$$

die eindeutige Lösung

$$x\equiv_{pq} vqa+upb\,,$$

wobei die ganzen Zahlen u *und* v *mit*

$$up+vq=1$$

durch den erweiterten euklidischen Algorithmus berechnet werden können.

Beweis. Nach Konstruktion gilt

$$x\equiv_p a\equiv_p 1a\equiv_p(vq+up)a\equiv_p(vqa+upa)\equiv_p vqa$$

und

$$x\equiv_q b\equiv_q 1b\equiv_q(vq+up)b\equiv_q(vqb+upb)\equiv_q upb\,.$$

□

Beispiel 7.8. Die eindeutige Lösung des Kongruenzensystems

$$x \equiv_5 1$$
$$x \equiv_7 2$$

ist

$$x \equiv_{35} 16\,.$$

Anwendung findet die modulare Arithmetik zum Beispiel bei Kryptosystemen, wo die Verschlüsselungsfunktion leicht zu berechnen aber schwer umzukehren sein soll.

7.5 Aufgaben

Aufgabe 7.1. Sei $b = 3$. Berechnen Sie $122 + 211$.

Aufgabe 7.2. Berechnen Sie 3^{12}. Wie viele Multiplikationen benötigt man mit dem Verfahren des schnellen Potenzierens (Satz 7.2)?

Aufgabe 7.3. Sei $a = 192$ und $b = 48$. Berechnen Sie mit dem erweiterten euklidischen Algorithmus ganze Zahlen u und v, sodass $u \cdot a + v \cdot b = \mathrm{ggT}(a, b)$.

Aufgabe 7.4. Sei $a = -12$ und $b = 75$. Berechnen Sie mit dem erweiterten euklidischen Algorithmus ganze Zahlen u und v, sodass $u \cdot a + v \cdot b = \mathrm{ggT}(a, b)$.

Aufgabe 7.5. Berechnen Sie mit dem binären erweiterten euklidischen Algorithmus ganze Zahlen u und v, sodass $u \cdot 192 + v \cdot 48 = \mathrm{ggT}(192, 48)$.

Aufgabe 7.6. Kann im binären erweiterten euklidischen Algorithmus in Schritt 8 auch

$$g = A_1 \cdot 2^e, u = A_2, v = A_3$$

verwendet werden?

Aufgabe 7.7. Zeigen Sie, dass Kongruenz modulo n eine Äquivalenzrelation ist.

Aufgabe 7.8. Berechnen Sie das inverse Element von $\overline{21}$ modulo 16. Welche Restklassen modulo 16 sind invertierbar?

Aufgabe 7.9. Lösen Sie das Kongruenzensystem

$$\begin{aligned} x &\equiv_8 1 \\ x &\equiv_{11} 2 \end{aligned}$$

Aufgabe 7.10. Ein alter Bauer hat 3 Söhne und 5 Töchter. Wenn er seine Kühe gleichmäßig auf die Söhne aufteilt, bleibt eine Kuh übrig, will er seine Kühe gleichmäßig auf die Töchter aufteilen, so bleiben 3 Kühe übrig. Wie viele Kühe hat der Bauer (mindestens)?

8

Reguläre Sprachen

Reguläre Sprachen sind eine Klasse von formalen Sprachen, für die viele Eigenschaften (effizient) überprüft werden können. Alle in diesem Kapitel verwendeten Alphabete sind endlich, sofern nicht anders angegeben.

8.1 Deterministische endliche Automaten

Definition 8.1 (deterministischer endlicher Automat)**.** Ein *deterministischer endlicher Automat* (*DEA*) ist gegeben durch

- eine endliche Menge Q, deren Elemente *Zustände* heißen,
- eine endliche Menge Σ, die *Eingabealphabet* heißt und deren Elemente *Eingabezeichen* genannt werden,
- eine Abbildung
$$\delta\colon Q \times \Sigma \to Q$$
die *Zustandsübergangsfunktion* heißt und angibt, wie sich der Zustand des Automaten bei einer Eingabe ändert.
- einen ausgezeichneten Zustand $s \in Q$, der *Startzustand* genannt wird,
- eine Teilmenge $F \subseteq Q$, deren Elemente *akzeptierende Zustände* genannt werden.

Ein DEA A kann als Quintupel $(Q, \Sigma, \delta, s, F)$ angegeben werden. Die Zustandsübergangsfunktion kann tabellarisch in der *Zustandstabelle* angegeben werden:

momentaner Zustand	Eingabe $e \in \Sigma$
$\vdots$	$\vdots$
$q \in Q$	$\cdots \quad \delta(q, e) \quad \cdots$
$\vdots$	$\vdots$

Visualisiert kann der Automat durch seinen *Zustandsgraphen* werden, der als gerichteter Graph wie folgt definiert ist:

– Die Ecken sind die Zustände.

– Für Zustände $p, q \in Q$ sind die Kanten von p nach q alle Tripel

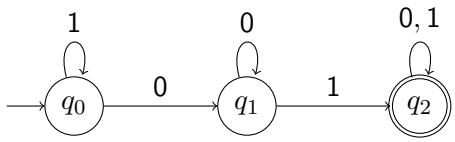
$$(p, a, q) \qquad \text{mit} \qquad a \in \Sigma \qquad \text{und} \qquad \delta(p, a) = q\,.$$

Üblicherweise schreibt man auf jede Kante (p, a, q) die Eingabe a, den Startzustand markiert man mit einem Pfeil und die akzeptierenden Zustände werden mit einem doppelten Kreis gekennzeichnet.

Beispiel 8.1. Der DEA $A = (\{q_0, q_1, q_2\}, \{\mathsf{0}, \mathsf{1}\}, \delta, q_0, \{q_2\})$ mit Zustandstabelle

	0	1
q_0	q_1	q_0
q_1	q_1	q_2
q_2	q_2	q_2

besitzt den folgenden Zustandsgraphen:



Gleichzeitig ist auch eine Darstellung durch die Zustandstabelle denkbar, wenn man zusätzlich den Startzustand mit $\rightarrow$ sowie die akzeptierenden Zustände mit $*$ markiert.

	0	1
$\rightarrow q_0$	q_1	q_0
q_1	q_1	q_2
$*q_2$	q_2	q_2

Definition 8.2 (erweiterte Zustandsübergangsfunktion)**.** Sei $A = (Q, \Sigma, \delta, s, F)$ ein DEA. Dann heißt $\hat{\delta}\colon Q \times \Sigma^* \rightarrow Q$ die *erweiterte Zustandsübergangsfunktion* und ist induktiv definiert:

1. Basis:
$$\hat{\delta}(q, \epsilon) := q \qquad \text{für alle } q \in Q\,.$$

2. Schritt:
$$\hat{\delta}(q, xa) := \delta(\hat{\delta}(q, x), a) \qquad \text{für alle } q \in Q,\ a \in \Sigma \text{ und } x \in \Sigma^*\,.$$

Beispiel 8.2. Für den DEA aus Beispiel 8.1 gilt $\hat{\delta}(q_0, \mathsf{0010}) = q_2$.

Definition 8.3 (Sprache eines DEA)**.** Für einen DEA $A = (Q, \Sigma, \delta, s, F)$ ist

$$\mathsf{L}(A) := \{x \in \Sigma^* \mid \hat{\delta}(s, x) \in F\}\,.$$

die *von A akzeptierte Sprache.*

Anschaulich interpretiert ist die von A akzeptierte Sprache die Menge der Wörter über Σ die im Zustandsgraphen von A vom Startzustand in einen akzeptierenden Zustand führen.

Beispiel 8.3. Für den DEA A aus Beispiel 8.1 gilt $\mathsf{L}(A) = \{x\mathsf{01}y \mid x, y \in \Sigma^*\}$. Die Sprache $\mathsf{L}(A)$ beschreibt die Menge aller Wörter, in welchen eine $\mathsf{1}$ direkt auf eine $\mathsf{0}$ folgt. Somit sind z.B. die Wörter $\mathsf{01}, \mathsf{11010}, \mathsf{100011}, \mathsf{0101}$ in $\mathsf{L}(A)$ enthalten.

Definition 8.4 (reguläre Sprache)**.** Eine formale Sprache L heißt *regulär*, wenn es einen DEA A gibt, sodass $\mathsf{L}(A) = L$.

Beispiel 8.4. Alle endlichen Sprachen sind regulär.

8.1.1 Minimierung

In der Praxis interessiert man sich für möglichst kleine Automaten, also solche die eine minimale Anzahl an Zuständen haben.

Definition 8.5 (erreichbarer Anteil)**.** Sei $A = (Q, \Sigma, \delta, s, F)$ ein DEA. Dann heißt der DEA $A' = (Q', \Sigma, \delta', s, F')$ der *erreichbare Anteil* von A, wobei

- $Q' = \{q \in Q \mid \text{ es gibt ein } x \in \Sigma^* \text{ mit } \hat{\delta}(s, x) = q\}$,
- $\delta' \colon Q' \times \Sigma \to Q'$ mit $\delta'(q, a) := \delta(q, a)$ für alle $q \in Q'$ und $a \in \Sigma$
- $F' = F \cap Q'$

Insbesondere können erreichbare Zustände mittels Nachfolgersuche (Satz 5.6) berechnet werden.

Beispiel 8.5. Der DEA aus Beispiel 8.1 stimmt mit seinem erreichbaren Anteil überein. Im DEA aus Beispiel 8.6 erhält man den erreichbaren Anteil durch *Weglassen* von Zustand q_3.

Satz 8.1 (Sprache des erreichbaren Anteils)**.** *Sei A ein DEA und A' der erreichbare Anteil von A. Dann ist $\mathsf{L}(A) = \mathsf{L}(A')$.*

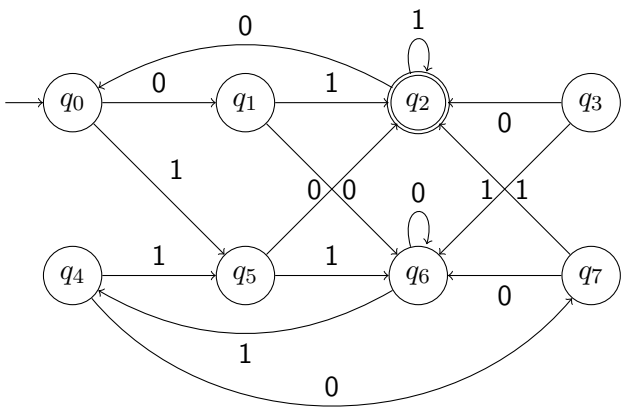
Beweis. Weil δ' wohldefiniert, $s \subseteq Q'$ und $F' \subseteq Q'$ ist $(Q', \Sigma, \delta', s, F')$ ein DEA. Es gilt

$$\begin{aligned}
\mathsf{L}(A) &= \{x \in \Sigma^* \mid \hat{\delta}(s, x) \in F\} && \text{(Definition 8.3)}\\
&= \{x \in \Sigma^* \mid \hat{\delta}'(s, x) \in F'\} && (\star)\\
&= \mathsf{L}(A') && \text{(Definition 8.3)}
\end{aligned}$$

Der Schritt $(\star)$ folgt direkt aus Aufgabe 8.4, welche zeigt, dass $\hat{\delta}(s, x) = \hat{\delta}'(s, x)$ für alle $x \in \Sigma^*$. □

Definition 8.6 (Äquivalenz von Zuständen)**.** Sei $A = (Q, \Sigma, \delta, s, F)$ ein DEA. Zwei Zustände $p \in Q$ und $q \in Q$ heißen *äquivalent*, wenn für jedes Wort $x \in \Sigma^*$, $\hat{\delta}(p, x) \in F$ genau dann, wenn $\hat{\delta}(q, x) \in F$.

Beispiel 8.6. Sei der folgende DEA gegeben durch den Zustandsgraphen:



Betrachte q_0 und q_6: Die Wörter ϵ, 0, und 1 sind nicht geeignet die beiden Zustände als nicht äquivalent festzustellen. Aber $\hat{\delta}(q_0, 01) = q_2$ und $\hat{\delta}(q_6, 01) = q_4$; also sind q_0 und q_6 nicht äquivalent.

Satz 8.2. *Äquivalenz von Zuständen ist eine Äquivalenzrelation.*

Beweis. Man prüft Reflexivität, Symmetrie und Transitivität nach. □

Definition 8.6 liefert ein Kriterium, wann zwei Zustände in einem DEA *zusammengefasst* werden können. Da für dieses Kriterium aber unendlich viele Wörter betrachtet werden müssen, führen wir eine alternative Definition ein.

Definition 8.7 (Unterscheidbarkeit von Zuständen)**.** Sei $A = (Q, \Sigma, \delta, s, F)$ ein DEA. Zwei Zustände $p \in Q$ und $q \in Q$ sind *unterscheidbar*,

1. Basis: wenn $p \in F$ genau dann, wenn $q \notin F$ oder
2. Schritt: wenn es unterscheidbare Zustände u und v sowie ein $a \in \Sigma$ gibt, sodass $\delta(p, a) = u$ und $\delta(q, a) = v$.

Aus Definition 8.7 erhalten wir einen Algorithmus (Definition 8.8) zum Auffinden von unterscheidbaren Zustandspaaren. Vorher widmen wir uns aber noch dem Zusammenhang von Äquivalenz und Nicht-Unterscheidbarkeit.

Satz 8.3 (Äquivalenz entspricht Nicht-Unterscheidbarkeit)**.** *Sei $A = (Q, \Sigma, \delta, s, F)$ ein DEA und seien $p, q \in Q$. Dann sind p und q äquivalent genau dann, wenn sie nicht unterscheidbar sind.*

Beweis.

– $\Rightarrow$ *Wir zeigen die Kontraposition der Aussage, also*

Wenn p und q unterscheidbar sind, sind sie nicht äquivalent.

Seien p und q unterscheidbar. Wir führen den Beweis über strukturelle Induktion bezüglich der Unterscheidbarkeits-Definition. *Im Basisfall gilt: $p \in F \Leftrightarrow q \notin F$. Somit ist $\hat{\delta}(p, \epsilon) \in F \Leftrightarrow \hat{\delta}(q, \epsilon) \notin F$ und damit sind p und q nicht äquivalent. Im Induktionsschritt folgt, weil p und q unterscheidbar sind, dass es ein $a \in \Sigma$ und $u, v \in Q$*

gibt, wobei $\delta(p,a) = u$, $\delta(q,a) = v$ und u und v unterscheidbar sind. Laut Induktionshypothese sind u und v nicht äquivalent und somit gibt es ein $x \in \Sigma^*$, sodass $\hat{\delta}(u,x) \in F \Leftrightarrow \hat{\delta}(v,x) \notin F$. Somit folgt aber $\hat{\delta}(p,ax) \in F \Leftrightarrow \hat{\delta}(q,ax) \notin F$. Also sind p und q nicht äquivalent.

– $\Leftarrow$ *Seien p und q nicht unterscheidbar. Für einen Widerspruchsbeweis nehmen wir an, dass p und q nicht äquivalent sind.*

  Im Beweis benutzen wir folgendes Minimalitätskriterium. Wenn p und q nicht äquivalente Zustände sind, die nicht unterscheidbar sind, dann muss es Zustände p', q' geben, die nicht unterscheidbar und nicht äquivalent sind und

$$\hat{\delta}(p',x) \in F \text{ und } \hat{\delta}(q',x) \notin F\,. \tag{8.1}$$

  wobei es kein kürzeres *$x \in \Sigma^*$ gibt, das zwei Zustände als nicht äquivalent zeigt.*

  Wenn $x = \epsilon$, dann folgt $p' \in F$ und $q' \notin F$, was einen Widerspruch zu der Annahme (dass p' und q' nicht unterscheidbar sind) gibt.

  Im anderen Fall ist $x = ay$. Sei $\delta(p',a) = u$ und $\delta(q',a) = v$. Klarerweise sind u und v nicht unterscheidbar (denn sonst wären entgegen unserer Annahme laut Definition 8.7 auch p' und q' unterscheidbar). Aus (8.1) folgt, dass $\hat{\delta}(u,y) \in F$ und $\hat{\delta}(v,y) \notin F$. Somit erhalten wir einen Widerspruch zur Annahme (dass es kein kürzeres Wort als x gibt, welches zwei nicht unterscheidbare Zustände als nicht äquivalent entlarvt).

□

Definition 8.8 (Markierungsalgorithmus)**.** Sei $A = (Q, \Sigma, \delta, s, F)$ ein DEA. Zur Initialisierung (Basis) markiert der Algorithmus alle Paare von Zuständen, sodass ein Zustand akzeptierend ist und der andere nicht. Solange (Schritt) es ein nicht markiertes Zustandspaar $(p,q) \in Q \times Q$ und ein $a \in \Sigma$ gibt, sodass $\delta(p,a) = u$ und $\delta(q,a) = v$, wobei (u,v) markiert ist, so wird auch (p,q) markiert.

Weil der Markierungsalgorithmus jene Zustandspaare markiert, die unterscheidbar sind, bleiben genau die äquivalenten Zustandspaare unmarkiert.

Beispiel 8.7. Für den DEA aus Beispiel 8.6 sind die Zustände q_0 und q_4, die Zustände q_1 und q_7 sowie die Zustände q_3 und q_5 äquivalent.

Der Markierungsalgorithmus bietet eine Methode um Automaten zu minimieren.

Definition 8.9 (Minimierungsalgorithmus)**.** Sei $A = (Q, \Sigma, \delta, s, F)$ ein DEA.

1. Bestimme den erreichbaren Anteil $A' = (Q', \Sigma, \delta', s, F')$ von A.
2. Bestimme die äquivalenten Zustände von A' mit dem Markierungsalgorithmus.
3. Konstruiere den minimalen DEA $B = (Q_B, \Sigma, \delta_B, s_B, F_B)$, wie folgt:
   – Q_B sind die Äquivalenzklassen von Q',
   – $\delta_B([q],a) := [p]$ wenn $\delta'(q,a) = p$ für $a \in \Sigma$ und $q \in Q'$,
   – der Startzustand ist $s_B := [s]$,

– die akzeptierenden Zustände sind $F_B = \{[f] \mid f \in F'\}$.

Beispiel 8.8. Für den Automaten aus Beispiel 8.6, liefert der Minimierungsalgorithmus die folgenden Äquivalenzklassen: $\{q_0, q_4\}$, $\{q_1, q_7\}$, $\{q_2\}$, $\{q_5\}$ und $\{q_6\}$.

Satz 8.4 (Korrektheit des Minimierungs-Algorithmus)**.** *Sei A ein DEA und B der DEA der durch Minimierung von A erhalten wurde. Dann ist $\mathsf{L}(A) = \mathsf{L}(B)$.*

Beweis. Man zeigt, dass der DEA B wohldefiniert ist (Aufgabe 8.5). Der Rest folgt aus Satz 8.1 und Satz 8.3. □

Satz 8.5 (Optimalität des Minimierungs-Algorithmus)**.** *Sei A ein DEA und B der DEA, der durch Minimierung von A erhalten wurde. Dann hat B eine minimale Anzahl an Zuständen, d.h. jeder DEA der die selbe Sprache wie A akzeptiert hat mindestens so viele Zustände wie B. Darüber hinaus ist der minimale DEA eindeutig bis auf Umbenennung der Zustände.*

Beweis. Siehe [3]. □

Satz 8.5 liefert eine Methode um zu entscheiden, ob zwei DEAs die selbe Sprache akzeptieren. Dazu vergleicht man die entsprechenden minimalen Automaten.

8.2 Nichtdeterministische endliche Automaten

Definition 8.10. Ein *nichtdeterministischer endlicher Automat* (*NEA*) ist gegeben durch

– eine endliche Menge Q, deren Elemente *Zustände* heißen,

– eine endliche Menge Σ, die *Eingabealphabet* heißt und deren Elemente *Eingabezeichen* genannt werden,

– eine Abbildung
$$\delta : Q \times \Sigma \to \mathcal{P}(Q)$$
die *Zustandsübergangsfunktion* heißt und angibt, wie sich der Zustand des Automaten bei einer Eingabe ändern kann.

– eine Teilmenge $S \subseteq Q$, deren Elemente *Startzustände* genannt werden,

– eine Teilmenge $F \subseteq Q$, deren Elemente *akzeptierende Zustände* genannt werden.

Somit kann ein NEA durch das Quintupel $(Q, \Sigma, \delta, S, F)$ angegeben werden. Die Zustandstabelle von DEAs beinhaltet für jeden Zustand und jede Eingabe genau einen Nachfolgezustand. Im Gegensatz dazu gibt die Zustandstabelle von NEAs eine Menge von möglichen Nachfolgezuständen an. Zudem kann ein NEA mehrere Startzustände haben.

Beispiel 8.9. Gegeben der NEA N durch die Zustandstabelle

	0	1
$\rightarrow q_0$	$\{q_0, q_1\}$	$\{q_0\}$
q_1	$\{q_1\}$	$\{q_2\}$
$*q_2$	$\{q_2\}$	$\{q_2\}$

In der Folge definieren wir die *erweiterte Zustandsübergangsfunktion* für NEAs.

Definition 8.11 (erweiterte Zustandsübergangsfunktion)**.** Sei $N = (Q, \Sigma, \delta, S, F)$ ein NEA. Dann heißt $\hat{\delta}\colon Q \times \Sigma^* \to \mathcal{P}(Q)$ die *erweiterte Zustandsübergangsfunktion* und ist wie folgt definiert:

1. Basis.
$$\hat{\delta}(q, \epsilon) := \{q\} \quad \text{für alle } q \in Q\,.$$

2. Schritt.
$$\hat{\delta}(q, xa) := \bigcup_{p \in \hat{\delta}(q,x)} \delta(p, a) \quad \text{für alle } q \in Q,\ x \in \Sigma^* \text{ und } a \in \Sigma\,.$$

Weniger formal ausgedrückt: Wir berechnen $\hat{\delta}(q, xa)$ indem wir zuerst $\hat{\delta}(q, x)$ berechnen und dann jedem Übergang von einem dieser Zustände folgen, der mit a markiert ist.

Beispiel 8.10. Sei N der NEA aus Beispiel 8.9. Dann ist $\hat{\delta}(q_0, \mathsf{00101}) = \{q_0, q_2\}$.

Definition 8.12 (Sprache eines NEA)**.** Sei $N = (Q, \Sigma, \delta, S, F)$ ein NEA. Dann wird

$$\mathsf{L}(N) := \{x \in \Sigma^* \mid \text{ es gibt einen Zustand } s \in S, \text{ sodass } \hat{\delta}(s, x) \cap F \neq \varnothing\}\,.$$

als die *von N akzeptierte Sprache* bezeichnet.

Beispiel 8.11. Für den NEA N aus Beispiel 8.9 gilt $\mathsf{L}(N) = \{x\mathsf{01}y \mid x, y \in \Sigma^*\}$.

Beispiele 8.3 und 8.11 zeigen, dass die Automaten A und N aus den Beispielen 8.1 und 8.9 die selben Sprachen akzeptieren. Das folgende Kapitel zeigt, dass dies kein Zufall ist, d.h. jeder NEA kann in einen DEA umgewandelt werden, der die selbe Sprache akzeptiert.

8.2.1 Teilmengenkonstruktion

Definition 8.13 (Teilmengenkonstruktion)**.** Sei $N = (Q_N, \Sigma, \delta_N, S, F_N)$ ein NEA. Dann ist $A = (Q_D, \Sigma, \delta_D, S, F_D)$ ein DEA, wobei

1. Q_D ist die Menge der Teilmengen von Q_N, also $Q_D = \mathcal{P}(Q_N)$.

2. Für alle $P \in Q_D$ und $a \in \Sigma$
$$\delta_D(P, a) = \bigcup_{p \in P} \delta_N(p, a)\,.$$

3. F_D ist die Menge $\{P \subseteq Q_N \mid P \cap F_N \neq \varnothing\}$.

Beispiel 8.12. Für den NEA N aus Beispiel 8.9 liefert die Teilmengenkonstruktion den DEA A mit der Zustandstabelle:

	0	1
$\varnothing$	$\varnothing$	$\varnothing$
$\rightarrow\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_1\}$	$\{q_1\}$	$\{q_2\}$
$*\{q_2\}$	$\{q_2\}$	$\{q_2\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$*\{q_0, q_2\}$	$\{q_0, q_1, q_2\}$	$\{q_0, q_2\}$
$*\{q_1, q_2\}$	$\{q_1, q_2\}$	$\{q_2\}$
$*\{q_0, q_1, q_2\}$	$\{q_0, q_1, q_2\}$	$\{q_0, q_2\}$

Wir zeigen, dass die Teilmengenkonstruktion korrekt ist, d.h. dass der erzeugte DEA die selbe Sprache akzeptiert wie der NEA.

Satz 8.6. *Sei $N = (Q_N, \Sigma, \delta_N, S, F_N)$ ein NEA und $A = (Q_D, \Sigma, \delta_D, S, F_D)$ der durch die Teilmengenkonstruktion berechnete DEA. Dann gilt $\mathsf{L}(A) = \mathsf{L}(N)$.*

Beweis. Aus den Definitionen von $\mathsf{L}(A)$ und $\mathsf{L}(N)$ erhalten wir:

$$\begin{aligned}
\mathsf{L}(A) &= \{x \in \Sigma^* \mid \hat{\delta}_D(S, x) \in F_D\} && \text{(Definition 8.3)}\\
&= \{x \in \Sigma^* \mid \hat{\delta}_D(S, x) \in \{P \subseteq Q_N \mid P \cap F_N \neq \varnothing\}\} && \text{(Definition von } F_D\text{)}\\
&= \{x \in \Sigma^* \mid \hat{\delta}_D(S, x) \cap F_N \neq \varnothing\}\\
&= \{x \in \Sigma^* \mid \bigcup_{s \in S} \hat{\delta}_N(s, x) \cap F_N \neq \varnothing\} && (8.2)\\
&= \mathsf{L}(N)\,. && \text{(Definition 8.12)}
\end{aligned}$$

Somit folgt das Resultat, wenn für alle Wörter $x \in \Sigma^*$ und $P \subseteq Q_N$:

$$\hat{\delta}_D(P, x) = \bigcup_{p \in P} \hat{\delta}_N(p, x)\,. \tag{8.2}$$

Wir zeigen (8.2) mittels struktureller Induktion über x.

1. Basis: *Sei $x = \epsilon$. Dann*

$$\hat{\delta}_D(P, \epsilon) = P = \bigcup_{p \in P} \{p\} = \bigcup_{p \in P} \hat{\delta}_N(p, \epsilon)\,. \tag{8.3}$$

2. Schritt: *Sei $x = ya$. Abkürzend schreiben wir*

$$\hat{\delta}_N(P, x) \text{ für } \bigcup_{p \in P} \hat{\delta}_N(p, x)\,.$$

Die Induktionshypothese lautet $\hat{\delta}_D(P, y) = \hat{\delta}_N(P, y)$. Wir müssen $\hat{\delta}_D(P, ya) = \hat{\delta}_N(P, ya)$ zeigen.

$$
\begin{aligned}
\hat{\delta}_D(P, ya) &= \delta_D(\hat{\delta}_D(P, y), a) && \text{(Definition 8.2)}\\
&= \delta_D(\hat{\delta}_N(P, y), a) && \text{(IH)}\\
&= \bigcup_{p \in \hat{\delta}_N(P,y)} \delta_N(p, a) && \text{(Definition von } \delta_D\text{)}\\
&= \delta_N(P, ya) && \text{(Definition 8.11)}
\end{aligned}
$$

□

Satz 8.7. *Eine formale Sprache L wird von einem DEA akzeptiert, genau dann wenn L von einem NEA akzeptiert wird.*

Beweis.

$\Leftarrow$: *Dieser Teil des Satzes folgt aus Satz 8.6.*

$\Rightarrow$: *Dieser Teil ist einfach, da für jeden DEA $A = (Q, \Sigma, \delta, s, F)$ der NEA $N = (Q, \Sigma, \delta_N, \{s\}, F)$ die selbe Sprache akzeptiert, wobei δ_N wie folgt definiert ist: $\delta_N(p, a) = \{q\} :\Leftrightarrow \delta(p, a) = q$.*

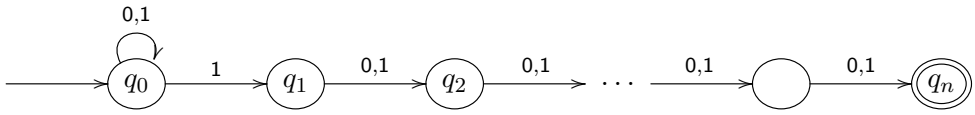
□

Es ist zu beachten, dass die Teilmengenkonstruktion in Beispiel 8.12 einen unnötig großen DEA liefert, da von den 8 angegebenen Zuständen nur 3 vom Startzustand aus *erreichbar* sind. Die nicht erreichbaren Zustände können laut Satz 8.1 aus dem Automaten entfernt werden, ohne dass sich dabei die akzeptierte Sprache ändert.

Um diesem Tatbestand Rechnung zu tragen, kann *nach* der Teilmengenkonstruktion der erreichbare Anteil (siehe Definition 8.5) berechnet werden. Effizienter hingegen ist es, die Zustandsübergangsfunktion δ_D nur für die erreichbaren Zustände zu berechnen. Es ist leicht einzusehen, dass die Zustände des DEA D auf die erreichbaren Teilmengen von Q_N eingeschränkt werden können. Folglich müssen wir nur für diese Teilmengen die Übergangsfunktion δ_D berechnen.

Die Teilmengenkonstruktion führt potentiell zu einer exponentiellen Explosion der betrachteten Zustände. In der Praxis tritt diese Explosion jedoch recht selten auf. Das nächste Beispiel zeigt jedoch, dass wir einen (relativ) einfachen NEA angeben können, sodass der assoziierte DEA die potentielle obere Schranke fast erreicht.

Beispiel 8.13. Betrachte den folgenden NEA N:



Zunächst sehen wir, dass N genau die Wörter über dem Alphabet $\{0, 1\}$ akzeptiert, sodass das n-te Zeichen vor Ende des Strings eine 1 ist. Es ist einfach einzusehen, dass jeder DEA D, der dieselbe Sprache wie der NEA in Beispiel 8.13 akzeptieren soll, sich alle n Zeichen merken muss, die vor dem Wortende gelesen werden. Angenommen D hätte nun weniger

als 2^n Zustände. Dann gäbe es aber einen Zustand q, der nicht zwischen zwei verschiedenen Zeichenreihen

$$a_1 a_2 \cdots a_n \qquad \text{und} \qquad b_1 b_2 \cdots b_n\,,$$

unterscheiden kann. Das führt wie folgt zum Widerspruch. Da die beiden Wörter verschieden sind, muss es zumindest eine Position i geben, wo sie sich unterscheiden. Sei $i = 1$ und o.B.d.A.[1]

können wir annehmen dass $a_1 = \mathsf{1}$ und $b_1 = \mathsf{0}$. Dann führt das Wort $a_1 a_2 \cdots a_n$ zu einem akzeptierenden Zustand, $b_1 b_2 \cdots b_n$ hingegen nicht. Nach Voraussetzung kann q aber nicht zwischen den beiden Worten unterscheiden. Widerspruch.

Sei nun $i > 1$, dann betrachten wir den Zustand p, in den D, nach dem Lesen von $a_1 a_2 \cdots a_n 0^{i-1}$ bzw. $b_1 b_2 \cdots b_n 0^{i-1}$, kommt. Nun kann das Argument für den Zustand p wiederholt werden und wir erhalten wiederum einen Widerspruch zur Annahme, dass D existiert.

Sei $N = (Q, \Sigma, \delta, S, F)$ ein NEA mit $S = \{s\}$ und $\#(\delta(q,a)) \leqslant 1$ für alle $q \in Q$ und alle $a \in \Sigma$. Durch Einführen eines sogenannten *Fangzustandes* $f \notin Q$ und Erweiterung der Zustandsübergangsfunktion wie folgt:

$$\delta_f(q,a) := \begin{cases} \delta(q,a) & \text{falls } q \in Q \ \text{ und } \ \#(\delta(q,a)) = 1 \\ \{f\} & \text{falls } (\ q \in Q \ \text{ und } \ \#(\delta(q,a)) = 0\ ) \ \text{ oder } \ q = f \end{cases}$$

erhält man einen NEA $N' = (Q \cup \{f\}, \Sigma, \delta_f, S, F)$, der dieselben Wörter wie N akzeptiert. Aus dem NEA N' konstruieren wir den folgenden DEA A:

$$A = (Q \cup \{f\}, \Sigma, \delta_D, s, F)\,.$$

Hier wird die Übergangsfunktion δ_D so definiert, dass für alle $p, q \in Q \cup \{f\}$ und alle $a \in \Sigma$ gilt: $\delta_D(p,a) = q :\Leftrightarrow \delta_f(p,a) = \{q\}$. Es ist leicht einzusehen, dass die Automaten A, N' und N die selbe Sprache akzeptieren. Der Automat A wird als DEA mit *Fangzustand* bezeichnet.

Aufgrund der Äquivalenz des NEA N mit dem DEA A werden in der Folge Automaten mit genau einem Startzustand und höchstens einem Folgezustand oft als *deterministisch* bezeichnet. Nur wenn dieser erweiterte Begriff nicht zulässig ist, wie etwa in der Minimierung von Automaten, werden DEAs nach Definition 8.1 und DEAs mit Fangzustand unterschieden.

8.3 Endliche Automaten mit Epsilon-Übergängen

Dieses Kapitel widmet sich ϵ-NEAs, welche spontan (ohne das Lesen einer Eingabe) den Zustand wechseln können.

Definition 8.14 (NEA mit ϵ-Übergängen)**.** Ein ϵ-NEA $N = (Q, \Sigma, \delta, S, F)$ ist gegeben durch

- eine endliche Menge Q, deren Elemente *Zustände* heißen,
- eine endliche Menge Σ, die *Eingabealphabet* heißt und ϵ nicht enthält,

[1] Ohne Beschränkung der Allgemeinheit

– eine Abbildung
$$\delta\colon Q \times (\Sigma \cup \{\epsilon\}) \to \mathcal{P}(Q)\,,$$
die *Zustandsübergangsfunktion* heißt, und angibt wie sich der Zustand eines Automaten bei einer Eingabe oder spontan ändern kann,

– eine Teilmenge $S \subseteq Q$, deren Elemente *Startzustände* genannt werden,

– eine Teilmenge $F \subseteq Q$, deren Elemente *akzeptierende Zustände* genannt werden.

Somit unterscheidet sich ein NEA von einem ϵ-NEA nur durch die Zustandsübergangsfunktion δ.

Beispiel 8.14. Sei der ϵ-NEA N gegeben durch die folgende Zustandstabelle:

	0	1	2	ϵ
$\to q_0$	$\{q_0\}$	$\{q_1\}$	$\{q_2\}$	$\varnothing$
q_1	$\{q_1\}$	$\{q_2\}$	$\varnothing$	$\{q_0\}$
$*q_2$	$\{q_2\}$	$\varnothing$	$\{q_0\}$	$\{q_1\}$

Um die erweiterte Zustandsübergangsfunktion für ϵ-NEAs zu definieren, benötigen wir die Definition der ϵ-Hülle.

Definition 8.15 (ϵ-Hülle)**.** Sei $N = (Q, \Sigma, \delta, S, F)$ ein ϵ-NEA. Die *ϵ-Hülle* einer Menge $P \subseteq Q$ ist induktiv definiert:

1. Basis: Für alle $p \in P$ gilt $p \in \epsilon\text{-Hülle}(P)$.
2. Schritt: Wenn $p \in \epsilon\text{-Hülle}(P)$ dann ist $\delta(p, \epsilon) \subseteq \epsilon\text{-Hülle}(P)$.

Die ϵ-Hülle kann mittels Nachfolgersuche (Satz 5.6) berechnet werden.

Beispiel 8.15. Für den ϵ-NEA aus Beispiel 8.14 ist
$$\epsilon\text{-Hülle}(\{q_0\}) = \{q_0\}, \quad \epsilon\text{-Hülle}(\{q_1\}) = \{q_0, q_1\} \quad \text{und} \quad \epsilon\text{-Hülle}(\{q_2\}) = \{q_0, q_1, q_2\}\,.$$

Definition 8.16 (erweiterte Zustandsübergangsfunktion)**.** Sei $(Q, \Sigma, \delta, S, F)$ ein ϵ-NEA. Die *erweiterte Zustandsübergangsfunktion*
$$\hat{\delta}\colon Q \times \Sigma^* \to \mathcal{P}(Q)$$
ist induktiv definiert:

1. Basis: $\hat{\delta}(q, \epsilon) = \epsilon\text{-Hülle}(\{q\})$.
2. Schritt: $\hat{\delta}(q, xa) = \epsilon\text{-Hülle}(\bigcup_{p \in \hat{\delta}(q,x)} \delta(p, a))\,.$

Beispiel 8.16. Für den ϵ-NEA aus Beispiel 8.14 ist
$$\hat{\delta}(q_0, 2) = \{q_0, q_1, q_2\} \quad \text{und} \quad \hat{\delta}(q_0, 21) = \{q_0, q_1, q_2\}\,.$$

Definition 8.17 (Sprache eines ϵ-NEA)**.** Sei $N = (Q, \Sigma, \delta, S, F)$ ein ϵ-NEA. Dann heißt

$$\mathsf{L}(N) := \{x \in \Sigma^* \mid \text{ es gibt einen Zustand } s \in S, \text{ sodass } \hat{\delta}(s, x) \cap F \neq \varnothing\}$$

die *von N akzeptierte Sprache.*

Beispiel 8.17. Wir konstruieren einen ϵ-NEA, der die Menge der Gleitkommazahlen akzeptiert. Gleitkommazahlen setzen sich wie folgt zusammen:

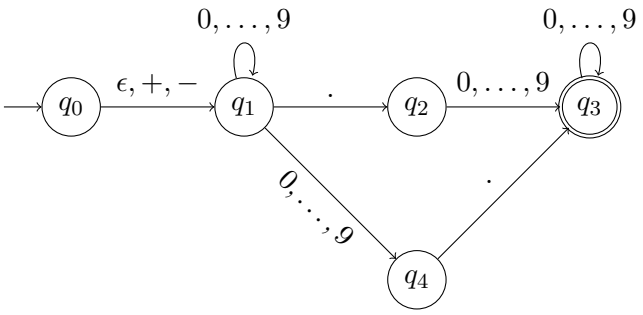
(a) einem optionalen Vorzeichen: $+$, $-$,

(b) einem Wort von Ziffern zwischen 0 und 9,

(c) einem Dezimalpunkt und

(d) einem weiteren Wort von Ziffern zwischen 0 und 9.

Die in den Punkten (b) und (d) beschriebenen Wörter können leer sein, aber zumindest eines davon muss nichtleer sein. Der folgende Automat akzeptiert nun genau die gesuchte Klasse von Gleitkommazahlen.



Satz 8.8 (Entfernen von ϵ-Übergängen)**.** *Sei $N = (Q, \Sigma, \delta, S, F)$ ein ϵ-NEA und $N' = (Q, \Sigma, \delta', S', F)$ ein NEA, wobei*

- $S' = \epsilon$-*Hülle*(S)
- $\delta'(q, a) := \epsilon$-*Hülle*$(\delta(q, a))$ *für alle $a \in \Sigma$ und $q \in Q$.*

Dann ist $\mathsf{L}(N) = \mathsf{L}(N')$.

Beweis. Die Eigenschaft

$$\hat{\delta}(q, x) = \bigcup_{p \in \epsilon\text{-Hülle}(\{q\})} \hat{\delta}'(p, x) \qquad \text{für alle } q \in Q,\ x \in \Sigma^* \tag{8.4}$$

kann mittels struktureller Induktion über x bewiesen werden (siehe Aufgabe 8.12). Somit folgt das Resultat

$$\begin{aligned}
\mathsf{L}(N) &= \{x \in \Sigma^* \mid \text{es gibt ein } s \in S \text{ mit } \hat{\delta}(s, x) \cap F \neq \varnothing\} && \text{(Definition 8.17)}\\
&= \{x \in \Sigma^* \mid \text{es gibt ein } s \in S \text{ mit } (\bigcup_{s' \in \epsilon\text{-Hülle}(\{s\})} \hat{\delta}'(s', x)) \cap F \neq \varnothing\} && (8.4)\\
&= \{x \in \Sigma^* \mid \text{es gibt ein } s \in S' \text{ mit } \hat{\delta}'(s, x) \cap F \neq \varnothing\} && \text{(Definition von } S')\\
&= \mathsf{L}(N')\,. && \text{(Definition 8.12)}
\end{aligned}$$

□

Satz 8.9. *Eine Sprache L wird genau dann von einem NEA akzeptiert, wenn sie von einem ϵ-NEA akzeptiert wird.*

Beweis.

1. *Wenn: Dieser Teil des Satzes folgt aus Satz 8.8.*

2. *Nur-dann-wenn: Dieser Teil ist einfach, da jeder NEA in einen ϵ-NEA übergeführt werden kann, indem die Zustandsübergangsfunktion δ für das Zeichen ϵ für alle $q \in Q$ wie folgt erweitert wird: $\delta(q, \epsilon) = \varnothing$.*

□

Beispiel 8.18. Für den ϵ-NEA aus Beispiel 8.14 liefert Satz 8.8 den NEA mit der Zustandstabelle:

	0	1	2
$\rightarrow q_0$	$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0, q_1, q_2\}$
q_1	$\{q_0, q_1\}$	$\{q_0, q_1, q_2\}$	$\varnothing$
$*q_2$	$\{q_0, q_1, q_2\}$	$\varnothing$	$\{q_0\}$

8.4 Reguläre Ausdrücke

Definition 8.18 (Reguläre Ausdrücke)**.** Sei Σ ein Alphabet, welches die Zeichen $\{\boldsymbol{\varnothing}, \boldsymbol{\epsilon}, \boldsymbol{*}, \boldsymbol{+}\}$ nicht enthält. Die formale Sprache der *regulären Ausdrücke* über Σ ist induktiv definiert:

1. Basis
   - $\boldsymbol{\varnothing}$ ist ein regulärer Ausdruck.
   - $\boldsymbol{\epsilon}$ ist ein regulärer Ausdruck.
   - Für jedes $a \in \Sigma$ ist $\boldsymbol{a}$ ein regulärer Ausdruck.

2. Schritt
   - Für jeden regulären Ausdruck E ist auch die *Iteration* (E^*) ein regulärer Ausdruck.
   - Für reguläre Ausdrücke E und F ist auch die *Konkatenation* (EF) ein regulärer Ausdruck.
   - Für reguläre Ausdrücke E und F ist auch die *Alternative* $(E + F)$ ein regulärer Ausdruck.

Beispiel 8.19. Sei $\Sigma = \{0, 1\}$ ein Alphabet. Dann sind

$$\boldsymbol{\varnothing}, \quad \boldsymbol{\epsilon}, \quad \mathbf{0}, \quad ((\mathbf{00})\mathbf{1}), \quad (\mathbf{0}(\mathbf{01})), \quad ((\mathbf{01})^*), \quad (((\mathbf{01}) + \boldsymbol{\epsilon})\mathbf{0})$$

reguläre Ausdrücke über Σ.

Um Klammern einzusparen, verwenden wir die folgenden Prioritäten für die Operatoren: Die Iteration * bindet stärker als Konkatenation und Alternative. Die Konkatenation bindet stärker als die Alternative. Sowohl Konkatenation als auch Alternative sind links-assoziativ.

Beispiel 8.20. Somit schreiben wir gemäß obiger Konvention für Beispiel 8.19 die regulären Ausdrücke als

$$\boldsymbol{\varnothing}, \quad \boldsymbol{\epsilon}, \quad \mathbf{0}, \quad \mathbf{001}, \quad \mathbf{0(01)}, \quad \mathbf{(01)^*}, \quad \mathbf{(01+\epsilon)0}\,.$$

Reguläre Ausdrücke können dazu verwendet werden um formale Sprachen zu beschreiben. Dazu wiederholen wir folgende Definitionen:

- $L \cup M := \{x \mid x \in L \text{ oder } x \in M\}$.
- $LM := \{xy \mid x \in L,\, y \in M\}$.
- $L^0 := \{\epsilon\}$, $L^{n+1} := LL^n$.
- $L^* := \bigcup_{n \geqslant 0} L^n$.

Definition 8.19 (Sprache eines regulären Ausdrucks)**.** Die *Sprache eines regulären Ausdrucks* ist induktiv definiert:

1. Basis
   - $\mathsf{L}(\boldsymbol{\varnothing}) := \varnothing$.
   - $\mathsf{L}(\boldsymbol{\epsilon}) := \{\epsilon\}$.
   - Für jedes $a \in \Sigma$ ist $\mathsf{L}(\boldsymbol{a}) := \{a\}$.
2. Schritt
   - $\mathsf{L}(E^*) := (\mathsf{L}(E))^*$.
   - $\mathsf{L}(EF) := \mathsf{L}(E)\,\mathsf{L}(F)$.
   - $\mathsf{L}(E+F) := \mathsf{L}(E) \cup \mathsf{L}(F)$.

Wir nennen reguläre Ausdrücke E und F *äquivalent* und schreiben $E \equiv F$, wenn $\mathsf{L}(E) = \mathsf{L}(F)$.

Beispiel 8.21. Die regulären Ausdrücke aus Beispiel 8.19 beschreiben die folgenden Sprachen:

$$\varnothing, \quad \{\epsilon\}, \quad \{\mathsf{0}\}, \quad \{\mathsf{001}\}, \quad \{\mathsf{001}\}, \quad \{\epsilon, \mathsf{01}, \mathsf{0101}, \ldots\}, \quad \{\mathsf{010}, \mathsf{0}\}\,.$$

Beispiel 8.22. Sei $\Sigma = \{\mathsf{0}, \mathsf{1}\}$. Für den regulären Ausdruck

$$E = \mathbf{(0+1)^*01(0+1)^*}$$

ist

$$\mathsf{L}(E) = \{x\mathsf{01}y \mid x, y \in \Sigma^*\}\,.$$

Wir geben einige algebraische Gesetze für reguläre Ausdrücke an.

Satz 8.10. *Seien D, E, F reguläre Ausdrücke, dann gilt:*

- $\mathsf{L}(E + F) = \mathsf{L}(F + E)$, *das* Kommutativgesetz *der Alternative.*
- $\mathsf{L}((D + E) + F) = \mathsf{L}(D + (E + F))$, *das* Assoziativitätsgesetz *der Alternative.*
- $\mathsf{L}((DE)F) = \mathsf{L}(D(EF))$, *das* Assoziativitätsgesetz *der Konkatenation.*

Beweis. Der erste Punkt folgt direkt aus Definition 8.19 und der Kommutativität der Mengenvereinigung, also $\mathsf{L}(E + F) = \mathsf{L}(E) \cup \mathsf{L}(F) = \mathsf{L}(F) \cup \mathsf{L}(E) = \mathsf{L}(F + E)$. *Die anderen Punkte sind analog.* □

Satz 8.11. *Sei E ein regulärer Ausdruck, dann gilt:*

- $\mathsf{L}(\varnothing + E) = \mathsf{L}(E + \varnothing) = \mathsf{L}(E)$, *d.h.* $\varnothing$ *ist das* neutrale Element *für die Alternative.*
- $\mathsf{L}(\epsilon E) = \mathsf{L}(E\epsilon) = \mathsf{L}(E)$, *d.h.* ϵ *ist das* neutrale Element *der Konkatenation.*
- $\mathsf{L}(\varnothing E) = \mathsf{L}(E\varnothing) = \varnothing$, *d.h.* $\varnothing$ *ist ein* Annihilator *der Konkatenation.*

Beweis. Übung. □

Satz 8.12. *Seien D, E, F reguläre Ausdrücke, dann gilt:*

- $\mathsf{L}(D(E + F)) = \mathsf{L}(DE + DF)$, linkes Distributivgesetz *der Konkatenation bezüglich der Alternative.*
- $\mathsf{L}((E + F)D) = \mathsf{L}(ED + FD)$, rechtes Distributivgesetz *der Konkatenation bezüglich der Alternative.*

Beweis. Übung. □

Satz 8.13. *Sei E ein regulärer Ausdruck, dann gilt:*

- $\mathsf{L}(E + E) = \mathsf{L}(E)$, Idempotenzgesetz.

Beweis. Übung. □

Satz 8.14. *Seien E und F regulärere Ausdrücke, dann gilt:*

- $\mathsf{L}(E^*) = \mathsf{L}(E^*E^*) = \mathsf{L}(E^* + E^*) = \mathsf{L}((E^*)^*)$.
- $\mathsf{L}(\varnothing^*) = \mathsf{L}(\epsilon^*) = \{\epsilon\}$.
- $\mathsf{L}((E + F)^*) = \mathsf{L}((E^* + F^*)^*) = \mathsf{L}((E^*F^*)^*) = \mathsf{L}((E^*F)^*E^*) = \mathsf{L}(E^*(FE^*)^*)$.

Beweis.

- *Um $L(E^*) = L((E^*)^*)$ zu zeigen, zeigen wir $L(E)^* = (L(E)^*)^*$. Wenn $x \in L(E)^*$, dann gibt es $x_1, \ldots, x_n$, sodass $x = x_1 \ldots x_n$ mit $x_i \in L(E)$ für $1 \leqslant i \leqslant n$. Aus $L(E) \subseteq L(E)^*$ erhalten wir $x_i \in L(E)^*$ für $1 \leqslant i \leqslant n$, was zu $x \in (L(E)^*)^*$ führt. Wenn umgekehrt $x \in (L(E)^*)^*$, dann gibt es $x_1, \ldots, x_n$ sodass $x = x_1 \ldots x_n$ wobei $x_i \in L(E)^*$ für alle $1 \leqslant i \leqslant n$. Wegen $x_i \in L(E)^*$ gibt es $y_{i1}, \ldots y_{im_i}$, sodass $x_i = y_{i1} \ldots y_{im_i}$ mit $y_{ij} \in L(E)$ für alle $1 \leqslant i \leqslant n$ und $1 \leqslant j \leqslant m_i$. Weil wir x wie folgt darstellen können $x = y_{11} \ldots y_{1m_1} y_{21} \ldots y_{2m_2} \ldots y_{n1} \ldots y_{nm_n}$ folgt somit $x \in L(E)^*$.*

- *Zunächst zeigen wir* $\mathsf{L}(E^*) = \mathsf{L}(E^*E^*)$. *Dazu reicht es zu zeigen, dass* $\mathsf{L}(E)^* = \mathsf{L}(E)^*\,\mathsf{L}(E)^*$. *Einerseits, wenn* $x \in \mathsf{L}(E)^*$, *dann* $x\epsilon \in \mathsf{L}(E)^*\,\mathsf{L}(E)^*$, *also* $\mathsf{L}(E)^* \subseteq \mathsf{L}(E)^*\,\mathsf{L}(E)^*$. *Andererseits wenn* $x \in \mathsf{L}(E)^*\,\mathsf{L}(E)^*$, *dann existieren Wörter* y, z *aus* $\mathsf{L}(E)^*$, *sodass* $x = yz$. *Also existieren* $k, l \in \mathbb{N}$ *sodass* $y \in \mathsf{L}(E)^k$, $z \in \mathsf{L}(E)^l$ *und somit* $x \in \mathsf{L}(E)^{k+l}$, *womit gezeigt wäre dass* $x \in \mathsf{L}(E)^*$. *Es folgt* $\mathsf{L}(E)^*\,\mathsf{L}(E)^* \subseteq \mathsf{L}(E)^*$ *und* $\mathsf{L}(E)^* = \mathsf{L}(E)^*\,\mathsf{L}(E)^*$ *ist gezeigt.*

- *Nun zeigen wir* $\mathsf{L}((E+F)^*) = \mathsf{L}((E^*+F^*)^*)$. *Einerseits wenn* $x \in \mathsf{L}((E+F)^*)$, *dann existiert* $n \in \mathbb{N}$, *sodass* $x = x_1 \cdots x_n$ *und für alle* $1 \leqslant i \leqslant n$: $x_i \in \mathsf{L}(E+F)$. *Somit gilt* $x_i \in \mathsf{L}(E^*+F^*)$ *und in Summe:* $x \in \mathsf{L}((E^*+F^*)^*)$, *also* $\mathsf{L}((E+F)^*) \subseteq \mathsf{L}((E^*+F^*)^*)$. *Andererseits, sei* $x \in \mathsf{L}((E^*+F^*)^*)$. *Dann existiert* $n \in \mathbb{N}$, *sodass* $x = x_1 \cdots x_n$ *und für alle* $1 \leqslant i \leqslant n$: $x_i \in \mathsf{L}(E^*+F^*)$. *Zunächst nehmen wir an, dass* $x_i \in \mathsf{L}(E^*)$, *dann gilt aber auch* $x_i \in \mathsf{L}((E+F)^*)$. *In gleicher Weise folgt* $x_i \in \mathsf{L}((E+F)^*)$, *wenn* $x_i \in \mathsf{L}(F^*)$. *Somit gilt* $x \in \mathsf{L}(((E+F)^*)^*) = \mathsf{L}((E+F)^*)$ *und* $\mathsf{L}((E^*+F^*)^*) \subseteq \mathsf{L}((E+F)^*)$, *also auch* $\mathsf{L}((E+F)^*) = \mathsf{L}((E^*+F^*)^*)$.

Rest: Übung. □

8.4.1 Endliche Automaten und reguläre Ausdrücke

Satz 8.15. *Sei* N *ein* ϵ*-NEA. Dann existiert ein regulärer Ausdruck* R *mit* $\mathsf{L}(N) = \mathsf{L}(R)$.

Beweis. Wir können annehmen, dass N *die* n *Zustände* $\{1, \ldots, n\}$ *besitzt (ansonsten benennen wir entsprechend um). Wir definieren reguläre Ausdrücke* $R_{ij}^{(k)}$ *induktiv. Informell beschreiben diese Ausdrücke die Symbole, die in* N *auf einem Weg (im Zustandsgraph) vom Zustand* i *zum Zustand* j *gelesen werden. Als Zusatzbedingung für den Weg von* i *nach* j *fordern wir, dass zwischen* i *und* j *nur Zustände* $\leqslant k$ *besucht werden. Wir definieren* $R_{ij}^{(k)}$ *induktiv:*

Basis. $k = 0$. *Wir betrachten nur die Wege der Länge maximal 1, also Kanten, die von* i *nach* j *führen und den leeren Weg. Wir unterscheiden die Fälle* $i \neq j$ *und* $i = j$.

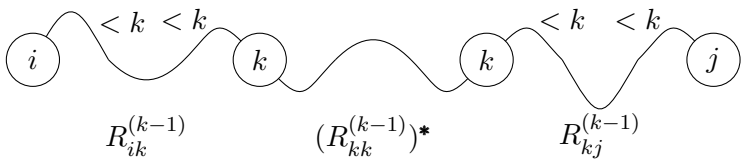
$i \neq j$
- *Wenn es keine solche Kante gibt, dann setzen wir* $R_{ij}^{(0)} := \boldsymbol{\varnothing}$.
- *Wenn es Kanten* $(i, a_1, j), \ldots, (i, a_l, j)$ *mit* $a_1, \ldots, a_l \in \Sigma \cup \{\epsilon\}$ *gibt, setzen wir* $R_{ij}^{(0)} := \boldsymbol{a_1 + \cdots + a_l}$.

$i = j$
- *Wenn es keine solche Kante gibt, dann setzen wir* $R_{ij}^{(0)} := \boldsymbol{\epsilon}$.
- *Wenn es Kanten* $(i, a_1, j), \ldots, (i, a_l, j)$ *mit* $a_1, \ldots, a_l \in \Sigma$ *gibt, setzen wir* $R_{ij}^{(0)} := \boldsymbol{\epsilon + a_1 + \cdots + a_l}$.

Schritt. $k > 0$. *Angenommen es gibt einen Weg von* i *nach* j, *der nur durch Zustände mit Nummer* $\leqslant k$ *führt. Wir betrachten zwei Fälle.*

- *Zustand* k *liegt nicht auf dem Weg. Dann können wir* $R_{ij}^{(k-1)}$ *verwenden.*
- *Zustand* k *liegt auf dem Weg. Wir spalten den Weg in mehrere Teilstücke. Der erste Teil führt von* i *nach* k *ohne* k *zu passieren, der letzte von* k *nach* j, *ebenfalls ohne* k *zu passieren. Die mittleren Stücke führen jeweils von* k *nach* k.

  Graphisch lässt sich die Zerlegung wie folgt darstellen:



Die Menge der gelesenen Wörter können wir also durch

$$R_{ik}^{(k-1)}(R_{kk}^{(k-1)})^* R_{kj}^{(k-1)}\,,$$

beschreiben. Das beendet die induktive Definition.

In Summe erhalten wir die folgende rekursive Definition:

$$R_{ij}^{(k)} := R_{ij}^{(k-1)} + R_{ik}^{(k-1)}(R_{kk}^{(k-1)})^* R_{kj}^{(k-1)}\,. \tag{8.5}$$

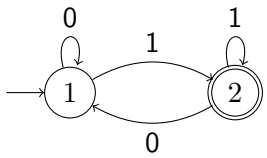
Der reguläre Ausdruck, der $\mathsf{L}(N)$ *beschreibt, ist durch die Alternative aller* $R_{sf}^{(n)}$ *gegeben, sodass* s *ein Startzustand und* f *akzeptierend ist. Wir definieren:*

$$R := R_{s_1 f_1}^{(n)} + \cdots + R_{s_1 f_l}^{(n)} + R_{s_2 f_1}^{(n)} + \cdots + R_{s_2 f_l}^{(n)} + \cdots + R_{s_m f_1}^{(n)} + \cdots + R_{s_m f_l}^{(n)}$$

wobei $s_1, \ldots, s_m$ *alle Startzustände und* $f_1, \ldots, f_l$ *alle akzeptierenden Zustände in* N *bezeichnen.* □

Beachten Sie die Ähnlichkeit des gegebenen Beweises zu den Korrektheitsbeweisen für die Algorithmen von Warshall und Floyd (siehe Satz 5.4 und 5.5). Im Folgenden wenden wir das Verfahren aus Satz 8.15 auch für DEAs und NEAs an.

Beispiel 8.23. Für den DEA A gegeben durch den Zustandsgraphen



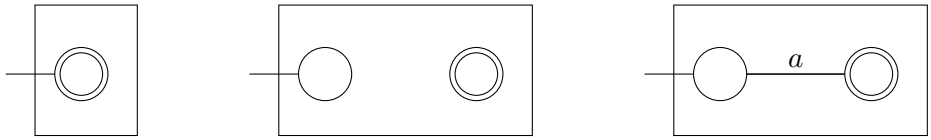
liefert Satz 8.15 (nach Vereinfachungen) den regulären Ausdruck $\mathbf{0^*1}(\mathbf{0^*1})^*$.

Satz 8.16. *Sei* R *ein regulärer Ausdruck. Dann existiert ein* ϵ*-NEA* N*, sodass* $\mathsf{L}(R) = \mathsf{L}(N)$.

Beweis. Sei R *ein regulärer Ausdruck. Im Beweis zeigen wir darüber hinaus, dass der* ϵ*-NEA* N

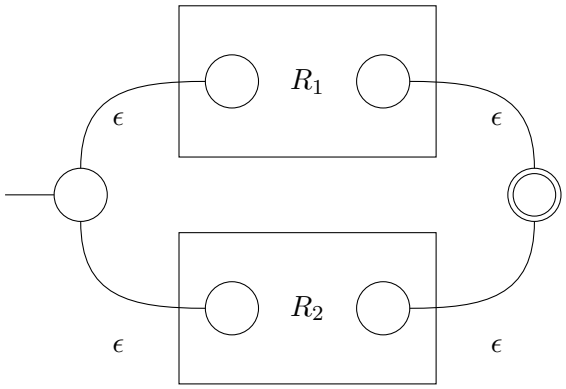
- *genau einen akzeptierenden Zustand,*
- *keine Kanten zum Startzustand und*
- *keine Kanten vom akzeptierenden Zustand*

besitzt. Wir verwenden strukturelle Induktion über reguläre Ausdrücke.
Basis. *Für die regulären Ausdrücke* $\boldsymbol{\epsilon}$, $\boldsymbol{\varnothing}$ *und* $\boldsymbol{a}$ *mit* $a \in \Sigma$ *betrachten wir die folgenden drei* ϵ*-NEAs.*

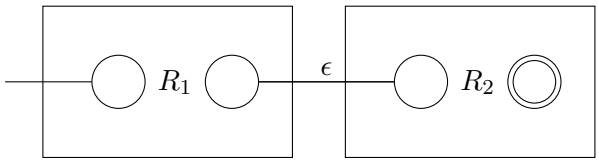


Schritt. *Wir führen eine Fallunterscheidung nach dem regulären Ausdruck* R *durch.*

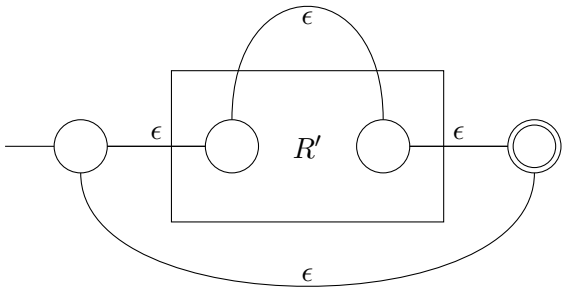
- $R = R_1 + R_2$. *Nach Induktionshypothese (IH) existieren* ϵ*-NEAs mit jeweils einem Start-/akzeptierenden Zustand für* R_1 *und* R_2. *Diese kombinieren wir wie folgt:*



- $R = R_1 R_2$. *Nach IH existieren* ϵ*-NEAs mit jeweils einem Start-/akzeptierenden Zustand für* R_1 *und* R_2. *Diese kombinieren wir wie folgt:*



- $R = (R')^*$. *Nach IH existiert ein* ϵ*-NEA mit einem Start-/akzeptierenden Zustand für* R'. *Wir transformieren diesen zu einem* ϵ*-NEA für* R.



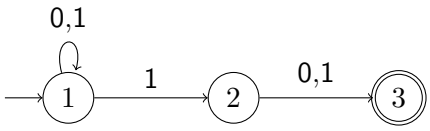
□

Beispiel 8.24. Für den regulären Ausdruck

$$(\mathbf{0}+\mathbf{1})^*\mathbf{1}(\mathbf{0}+\mathbf{1})\,,$$

liefert Satz 8.16 einen ϵ-NEA mit 16 Zuständen.

Die Methode von Satz 8.16 liefert im Allgemeinen keine minimalen Automaten. Der folgende NEA akzeptiert die selbe Sprache wie der reguläre Ausdruck aus Beispiel 8.24 und hat weniger Zustände:



Folgerung. *Die folgenden Mengen sind gleich:*

- *die Menge der durch reguläre Ausdrücke beschriebenen Sprachen,*
- *die Menge der von DEAs akzeptierten Sprachen,*
- *die Menge der von NEAs akzeptierten Sprachen,*
- *die Menge der von ϵ-NEAs akzeptierten Sprachen.*

Beweis. Folgt aus den Sätzen 8.7, 8.9, 8.15, und 8.16. □

Bemerkung. Reguläre Sprachen werden auch durch *rechts- oder linkslinear Grammatiken* beschrieben, siehe [7].

8.5 Abgeschlossenheit regulärer Sprachen

Satz 8.17. *Sind L und M regulär, dann ist auch die Vereinigung $L \cup M$ regulär.*

Beweis. Weil L und M regulär sind, existieren reguläre Ausdrücke E, F, sodass $L = \mathsf{L}(E)$ und $M = \mathsf{L}(F)$. Dann ist $E+F$ ein regulärer Ausdruck für die Sprache $\mathsf{L}(E) \cup \mathsf{L}(F) = L \cup M$ und somit ist $L \cup M$ regulär. □

Satz 8.18. *Wenn L (über dem Alphabet Σ) regulär ist, dann ist auch das Komplement $\sim L = \Sigma^* \setminus L$ regulär.*

Beweis. Da L regulär ist, existiert ein DEA $A = (Q, \Sigma, \delta, s, F)$, sodass $L = \mathsf{L}(A)$. Für den DEA $B = (Q, \Sigma, \delta, s, Q \setminus F)$ gilt $\sim L = \mathsf{L}(B)$ laut Konstruktion. □

Beispiel 8.25. Sei $L = \mathsf{L}((\mathbf{0}+\mathbf{1})^*\mathbf{01}(\mathbf{0}+\mathbf{1})^*)$. Dann ist $\sim L = \mathsf{L}(\mathbf{1}^*\mathbf{0}^*)$.

Satz 8.19. *Wenn L und M regulär sind, dann ist auch der Schnitt $L \cap M$ regulär.*

Beweis. Wir können das Gesetz von De Morgan anwenden.

$$L \cap M = \sim (\sim L \cup \sim M)\,.$$

Wir haben in den Sätzen 8.17 und 8.18 gezeigt, dass reguläre Sprachen unter Vereinigung und Komplement abgeschlossen sind. □

Satz 8.20. *Wenn L und M regulär sind, dann ist auch $L \setminus M$ regulär.*

Beweis. Verwende:

$$L \setminus M = L \cap (\sim M)\,.$$

□

8.6 Schleifen-Lemma

Satz 8.21. *Sei L eine reguläre Sprache über Σ. Dann existiert eine Konstante $n \in \mathbb{N}$, sodass für jedes Wort $w \in L$ mit $\ell(w) \geqslant n$ Wörter $x, y, z \in \Sigma^*$ existieren mit $w = xyz$ und*

- $y \neq \epsilon$,
- $\ell(xy) \leqslant n$,
- *für alle $k \geqslant 0$ gilt: $x(y)^k z \in L$.*

Beweis. Angenommen L ist regulär. Laut Folgerung 8.4.1 exisitiert ein DEA $A = (Q, \Sigma, \delta, s, F)$ sodass $L = \mathsf{L}(A)$. Sei $\#(Q) = n$ und

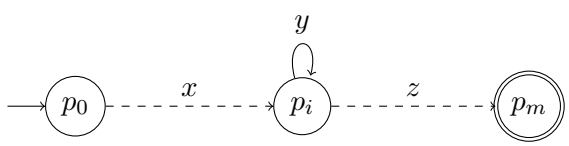
$$w = w_1 \cdots w_m \in L$$

mit $w_1, \ldots, w_m \in \Sigma$ und $m \geqslant n$. Definiere $p_l := \hat{\delta}(s, w_1 \cdots w_l)$ für $0 \leqslant l \leqslant m$; beachte für $l = 0$ ist $w_1 \cdots w_l = \epsilon$ und somit $p_0 = s$. Nach dem Schubfachprinzip (Satz 6.2) muss es $i, j \in \{0, \ldots, n\}$ mit $i < j$ und $p_i = p_j$ geben. Zerlege w:

$$\underbrace{w_1 \cdots w_i}_{x} \qquad \underbrace{w_{i+1} \cdots w_j}_{y \neq \epsilon} \qquad \underbrace{w_{j+1} \cdots w_m}_{z}\,.$$

Um das Wort $x(y)^k z$ zu akzeptieren, läuft der Automat k-mal durch den Weg, der p_i mit p_j verbindet. Da $p_i = p_j$ ist dies beliebig oft möglich. □

Die Situation stellt sich graphisch wie folgt dar:



Wir formulieren die Kontraposition des Schleifen-Lemmas.

Satz 8.22. *Sei L eine formale Sprache über Σ, sodass:*

1. Spielerin 1 wählt eine Sprache L, die als nicht regulär nachgewiesen werden soll.
2. Spieler 2 wählt eine natürliche Zahl $n \in \mathbb{N}$. Spieler 2 muss die Zahl nicht mitteilen.
3. Spielerin 1 wählt ein Wort $w \in L$, $\ell(w) \geqslant n$.
4. Spieler 2 zerlegt w beliebig in 3 Teile x,y,z, sodass $\ell(xy) \leqslant n$, $y \neq \epsilon$. Spieler 2 muss die Zerlegung nicht mitteilen.
5. Spielerin 1 gewinnt, wenn sie k wählen kann, sodass $x(y)^k z \notin L$ (unabhängig von der konkreten Zerlegung).

Wenn Spielerin 1 gewinnt, ist L nicht regulär. Wenn Spielerin 1 nicht gewinnt, kann keine Aussage über L getroffen werden.

Abbildung 8.1: Das Schleifen-Lemma als Spiel

– *für alle* $n \in \mathbb{N}$ *existiert ein Wort* $w \in L$ *mit* $\ell(w) \geqslant n$*, sodass*

– *für alle* $x, y, z \in \Sigma^*$ *mit* $w = xyz$*,* $y \neq \epsilon$ *und* $\ell(xy) \leqslant n$ *existiert* $k \in \mathbb{N}$ *mit* $x(y)^k z \notin L$

Dann ist L *nicht regulär.*

Beweis. Der Satz folgt unmittelbar aus Satz 8.21. □

Satz 8.22 können wir auch als Spiel formulieren, wie in Abbildung 8.1 zu sehen ist. Beachten Sie, wie die universellen Aussagen von Spieler 2, die existentiellen Aussagen von Spielerin 1 repräsentiert werden.

Beispiel 8.26. Sei $\Sigma = \{0, 1\}$. Dann ist die Sprache

$$L = \{w \in \Sigma^* \mid w \text{ enthält gleich viele } 0\text{en wie } 1\text{en}\}$$

nicht regulär.

Wir führen den Beweis als Spiel und nennen die Spieler Anna und Otto.

1. Anna wählt die Sprache L.
2. Otto wählt eine Zahl $n \in \mathbb{N}$. (Er muss Anna diese nicht mitteilen.)
3. Weil Anna nicht weiß, welches $n \in \mathbb{N}$ Otto gewählt hat, sucht sie ein Wort, welches von n abhängt, z.B. $w = 0^n 1^n$, welches in L ist.
4. Otto zerlegt w beliebig in x, y und z. Er muss dabei darauf achten, dass $\ell(xy) \leqslant n$ und $y \neq \epsilon$.
5. Anna kennt die Zerlegung nicht, kann aber aus den Bedingungen und der Kenntnis von w darauf schließen, dass $xy = 0^i$ für $i \leqslant n$. Sie wählt für $k = 0$.

Nun hat Anna gewonnen. Nach Voraussetzung gilt $y \neq \epsilon$, somit fehlt im Wort $x(y)^0 z = xz$ zumindest eine $\mathsf{0}$, wohingegen $xy^0 z$ aber noch dieselbe Anzahl an 1en enthält. Somit gilt $xz \notin L$. Also ist L nicht regulär.

Beispiel 8.27. Sei $\Sigma = \{\mathsf{1}\}$. Dann ist

$$L = \{w \in \Sigma^* \mid \ell(w) \text{ ist eine Primzahl}\}$$

nicht regulär.

Um zu zeigen, dass L nicht regulär ist, wenden wir Satz 8.22 an, wobei wir dem Schema des Satzes genau folgen. Wir müssen zeigen, dass für L gilt:

- für alle $n \in \mathbb{N}$ existiert ein Wort $w \in L$ mit $\ell(w) \geqslant n$, sodass
- für alle $x, y, z \in \Sigma^*$ mit $w = xyz$, $y \neq \epsilon$ und $\ell(xy) \leqslant n$ existiert $k \in \mathbb{N}$ mit $x(y)^k z \notin L$.

Um den ersten Punkt zu erfüllen, wählen wir das Wort $w = \mathsf{1}^p$, wobei p eine Primzahl größer oder gleich $n + 2$ ist. Somit ist $w \in L$ und $\ell(w) = p \geqslant n$.

Seien nun x, y und z beliebig, sodass $w = xyz$, $\ell(xy) \leqslant n$ und $y \neq \epsilon$. Setze $m := \ell(y)$; Wir wählen $k := \ell(xz) = p - m$. Betrachte $v := x(y)^{(p-m)} z$. Aber $v \notin L$, weil

$$\ell(v) = \ell(x(y)^{(p-m)} z) = (p - m) + m \cdot (p - m) = (p - m) \cdot (m + 1)\,.$$

Das heißt $\ell(v)$ ist das Produkt zweier Zahlen und somit keine Primzahl, wenn sowohl $(p - m) > 1$ als auch $(m + 1) > 1$. Dies zeigen wir durch direkte Rechnung:

1. $(p - m) > 1$, da $p \geqslant n + 2$ und $m = \ell(y) \leqslant \ell(xy) \leqslant n$; somit $p - m \geqslant n + 2 - n \geqslant 2$.
2. $(m + 1) > 1$, da $m = \ell(y)$ und $y \neq \epsilon$.

Da $\ell(v)$ keine Primzahl ist, folgt $v \notin L$ und aus Satz 8.22 folgt, dass L nicht regulär ist.

Beispiel 8.28. Sei $\Sigma = \{\mathsf{0}, \mathsf{1}\}$. Dann ist

$$L = \{w \in \Sigma^* \mid w \text{ enthält nicht gleich viele } \mathsf{0}\text{en wie } \mathsf{1}\text{en}\}$$

nicht regulär.

Eine Anwendung des Schleifen-Lemmas ist schwierig: Wir wählen z.B. das Wort $w = \mathsf{0}^n \mathsf{1}^{n+1}$, dann können wir für die Zerlegung $x = \mathsf{0}^{n-2}, y = \mathsf{0}^2$ und $z = \mathsf{1}^{n+1}$ kein k finden, sodass $xy^k z \notin L$.

Einfacher argumentieren wir mit Hilfe eines Widerspruchsbeweises. Angenommen L wäre regulär, dann wäre auch $\sim L$ regulär, unter Anwendung von Satz 8.18. Das Komplement von L ist jedoch die Sprache

$$\sim L = \{w \in \Sigma^* \mid w \text{ enthält gleich viele } \mathsf{0}\text{en wie } \mathsf{1}\text{en}\}$$

Die Nichtregularität von $\sim L$ haben wir in Beispiel 8.26 gezeigt. Widerspruch zu der Annahme, dass L regulär ist.

8.7 Aufgaben

Aufgabe 8.1. Was ist ein DEA? Was ist die *Sprache eines DEA*? Konstruieren Sie einen DEA, dessen Sprache alle Wörter über dem Alphabet $\{0, 1\}$ enthält, die eine gerade Anzahl von Nullen, aber eine ungerade Anzahl von Einsen haben. Wie viele Zustände hat dieser DEA mindestens?

Aufgabe 8.2. Betrachten Sie den DEA aus Beispiel 8.1. Berechnen Sie $\hat{\delta}(q_0, 0000)$ und $\hat{\delta}(q_0, 0101)$. Welches der beiden Wörter ist in der Sprache des Automaten?

Aufgabe 8.3. Sei $A = (Q, \Sigma, \delta, s, F)$ ein DEA. Beweisen Sie, dass

$$\hat{\delta}(q, yz) = \hat{\delta}(\hat{\delta}(q, y), z) \text{ für alle } q \in Q \text{ und } y, z \in \Sigma^* .$$

Hinweis: Verwenden Sie strukturelle Induktion über z, wobei Wörter induktiv definiert sind wie in Definition 3.3.

Aufgabe 8.4. Vervollständigen Sie den Beweis von Satz 8.1, d.h., zeigen Sie, dass

$$\hat{\delta}(s, x) = \hat{\delta}'(s, x) \text{ für alle } x \in \Sigma^* .$$

Aufgabe 8.5. Beweisen Sie, dass der DEA B aus Satz 8.4 wohldefiniert ist.

Aufgabe 8.6. Minimieren Sie den DEA mit der folgenden Zustandstabelle:

	0	1	2
$\rightarrow q_0$	q_2	q_1	q_0
q_1	q_3	q_0	q_1
q_2	q_0	q_3	q_2
$*q_3$	q_3	q_3	q_4
$*q_4$	q_4	q_4	q_4

Aufgabe 8.7. Beweisen oder widerlegen Sie: *Für jede reguläre Sprache gibt es einen DEA mit genau einem akzeptierenden Zustand.*

Aufgabe 8.8. Wie ist die erweiterte Zustandsübergangsfunktion für NEAs definiert? Wie ist die Sprache eines NEA definiert? Berechnen Sie $\hat{\delta}(q_0, 110001)$ für den NEA aus Beispiel 8.9.

Aufgabe 8.9. Wenden Sie die Teilmengenkonstruktion auf den NEA mit der folgenden Zustandstabelle an:

	0	1	2
$\rightarrow q_0$	$\{q_0, q_1\}$	$\{q_0\}$	$\{q_0\}$
q_1	$\{q_1\}$	$\{q_1, q_2\}$	$\varnothing$
$*q_2$	$\{q_0\}$	$\{q_0\}$	$\{q_0, q_1, q_2\}$

Aufgabe 8.10. Wenden Sie die Teilmengenkonstruktion auf den NEA mit der folgenden Zustandstabelle an:

	0	1	2
$\rightarrow q_0$	$\{q_0, q_1\}$	$\{q_0\}$	$\{q_0\}$
q_1	$\{q_1\}$	$\{q_0, q_1\}$	$\varnothing$
$*q_2$	$\{q_0\}$	$\{q_0\}$	$\{q_0, q_1, q_2\}$

Aufgabe 8.11. Konstruieren Sie einen NEA N', der die selbe Sprache wie der folgende ϵ-NEA N akzeptiert:

	$\mathsf{0}$	$\mathsf{1}$	ϵ
$\rightarrow q_0$	$\{q_0, q_1\}$	$\{q_0\}$	$\{q_0\}$
q_1	$\{q_1\}$	$\{q_1, q_2\}$	$\varnothing$
$*q_2$	$\{q_0\}$	$\{q_0\}$	$\{q_0, q_1, q_2\}$

Aufgabe 8.12. Beweisen Sie Eigenschaft (8.4) im Beweis von Satz 8.8.

Hinweis: Verwenden Sie zur einfacheren Lesbarkeit $\hat{\delta}(P, x)$ als Abkürzung für $\bigcup_{p \in P} \hat{\delta}(p, x)$ und $\delta(P, x)$ als Abkürzung für $\bigcup_{p \in P} \delta(p, x)$. Ist das Resultat von Aufgabe 8.31 nützlich?

Aufgabe 8.13. Geben Sie einen regulären Ausdruck E an, der die Sprache aller Wörter gerader Länge über dem Alphabet $\Sigma = \{\mathsf{0}, \mathsf{1}\}$ beschreibt.

Aufgabe 8.14. Sei $\Sigma = \{\mathsf{0}, \mathsf{1}\}$ und $E = (\mathbf{0} + \mathbf{1}^*)^*\mathbf{1}$. Testen Sie für jedes Wort $w \in \Sigma^*$ mit $\ell(w) \leq 3$, ob $w \in \mathsf{L}(E)$.

Aufgabe 8.15. Seien $E_1 = (\mathbf{0}+\mathbf{1})^*$, $E_2 = (\mathbf{01})^*$, $E_3 = \mathbf{0}^* + \mathbf{1}^*$, $E_4 = (\mathbf{0}^* + \mathbf{1})^*$, $E_5 = (\mathbf{0}^*\mathbf{1}^*)^*$ und $E_5 = (\mathbf{1}^*\mathbf{0}^*)^*$. Beschreiben Sie $\mathsf{L}(E_i)$ in Worten. Bestimmen Sie für $1 \leq i < j \leq 6$, ob $\mathsf{L}(E_i) = \mathsf{L}(E_j)$.

Aufgabe 8.16. Beweisen Sie Satz 8.11.

Aufgabe 8.17. Beweisen Sie Satz 8.12.

Aufgabe 8.18. Beweisen Sie Satz 8.13.

Aufgabe 8.19. Beweisen Sie, dass $\mathsf{L}(EE^*) = \mathsf{L}(E^*E)$ für jeden regulären Ausdruck E.

Aufgabe 8.20. Konstruieren Sie für den ϵ-NEA N aus Aufgabe 8.11 einen regulären Ausdruck E, sodass $\mathsf{L}(N) = \mathsf{L}(E)$.

Aufgabe 8.21. Konstruieren Sie (gemäß der Konstruktion aus Satz 8.15) für den regulären Ausdruck $E = (\mathbf{0} + \mathbf{1}^*)\mathbf{0} + \boldsymbol{\epsilon}$ einen ϵ-NEA N, sodass $\mathsf{L}(N) = \mathsf{L}(E)$.

Bonus: Wie viele Zustände hat ein äquivalenter ϵ-NEA mindestens?

Aufgabe 8.22. Sei $\Sigma = \{\mathsf{0}, \mathsf{1}\}$ und $E = \mathbf{0} + \mathbf{1}$. Berechnen Sie einen regulären Ausdruck F, mit $\mathsf{L}(F) = \sim(\mathsf{L}(E))$.

Aufgabe 8.23. Beweisen Sie, dass die Sprache

$$L = \{\mathsf{0}^n\mathsf{1}^n \mid n \geqslant 0\}$$

über $\Sigma = \{\mathsf{0}, \mathsf{1}\}$ nicht regulär ist.

Aufgabe 8.24. Beweisen Sie, dass die Sprache

$$L = \{\mathsf{0}^n\mathsf{1}^m \mid 0 < n < m\}$$

nicht regulär ist.

Aufgabe 8.25. Beweisen Sie, dass die Sprache

$$L = \{0^n 0^m \mid 0 < n < m\}$$

über $\Sigma = \{0\}$ regulär ist.

Aufgabe 8.26. Beweisen Sie, dass die Sprache

$$L = \{0^n 1^m \mid n, m \geqslant 0\}$$

über $\Sigma = \{0, 1\}$ regulär ist.

Aufgabe 8.27. Beweisen Sie, dass die formale Sprache der Palindrome über $\{a, b\}$ (siehe Beispiel 2.19) nicht regulär ist.

Aufgabe 8.28. Beweisen Sie Satz 8.2.

Aufgabe 8.29. Berechnen Sie die folgenden Mengen:

$$\varnothing^* \quad \{\epsilon\}^* \quad \{a, b\}\varnothing \quad \{a, b\}\{\epsilon\} \quad \{a, b\}\{\epsilon, a, b\} \quad \{\epsilon, a, b\}\{\epsilon, a, b\} \quad \{a, b\}^* \,.$$

Aufgabe 8.30. Beweisen oder widerlegen Sie:

1. *Für jede reguläre Sprache gibt es einen ϵ-NEA mit genau einem akzeptierenden Zustand.*
2. *Für jede reguläre Sprache gibt es einen NEA mit genau einem akzeptierenden Zustand.*

Aufgabe 8.31. Sei $N = (Q, \Sigma, \delta, S, F)$ ein NEA. Beweisen Sie, dass für alle $P \subseteq Q$ und $x \in \Sigma^*$

$$\epsilon\text{-Hülle}\Big(\bigcup_{q \in \hat{\delta}(P,x)} \delta(q, a)\Big) = \bigcup_{q \in \hat{\delta}(P,x)} \epsilon\text{-Hülle}(\delta(q, a)) \,.$$

Aufgabe 8.32. Die Konstruktion von Satz 8.8 liefert die Gleichheit (siehe Aufgabe 8.12)

$$\bigcup_{p \in P} \hat{\delta}(p, x) = \bigcup_{p \in \epsilon\text{-Hülle}(P)} \hat{\delta}'(p, x) \,.$$

Skizzieren Sie eine Konstruktion, welche einen ϵ-NEA $N = (Q, \Sigma, \delta, S, F)$ in einen NEA $N'' = (Q'', \Sigma, \delta'', S'', F'')$ umwandelt, sodass

$$\bigcup_{p \in P} \hat{\delta}(p, x) = \bigcup_{p \in P} \hat{\delta}''(p, x) \,.$$

Nennen Sie Vor- und Nachteile Ihrer Konstruktion im Vergleich zu jener aus Satz 8.8. Demonstrieren Sie beide Konstruktionen am ϵ-NEA N gegeben durch die folgende Zustandstabelle:

	0	1	ϵ
$\rightarrow q_0$	$\varnothing$	$\varnothing$	$\{q_1\}$
q_1	$\{q_2\}$	$\varnothing$	$\varnothing$
q_2	$\varnothing$	$\varnothing$	$\{q_3\}$
q_3	$\varnothing$	$\varnothing$	$\{q_4\}$
q_4	$\varnothing$	$\{q_5\}$	$\varnothing$
q_5	$\varnothing$	$\varnothing$	$\{q_6\}$
$*q_6$	$\varnothing$	$\varnothing$	$\varnothing$

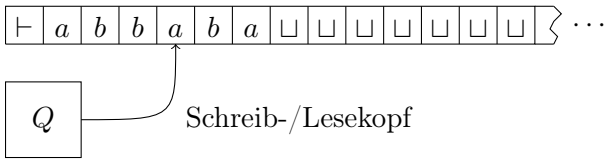
9

Berechenbarkeit

Alle in diesem Kapitel verwendeten Alphabete sind endlich. Turingmaschinen stellen ein einfaches Konzept dar, um die Klasse der *berechenbaren Funktionen* abstrakt zu beschreiben.

9.1 Deterministische Turingmaschinen

Eine *(deterministische einbändige) Turingmaschine* besteht aus einer endlichen Menge an Zuständen, einem einseitig unendlich langem Band und einem Schreib- bzw. Lesekopf, der ein Symbol am Band liest, ein Symbol auf das Band schreibt und eine Position nach links oder rechts wechselt. Schematisch kann eine Turingmaschine wie folgt dargestellt werden:



Definition 9.1. Eine *(deterministische einbändige) Turingmaschine* M (kurz TM oder DTM) ist ein 9-Tupel

$$M = (Q, \Sigma, \Gamma, \vdash, \sqcup, \delta, s, t, r)\,,$$

sodass

1. Q eine endliche Menge von *Zuständen*,
2. Σ eine endliche Menge von *Eingabesymbolen*,
3. $\Gamma \supseteq \Sigma$ eine endliche Menge von *Bandsymbolen*,
4. $\vdash \in \Gamma \setminus \Sigma$ der *linke Endmarker*,
5. $\sqcup \in \Gamma \setminus \Sigma$ das *Leerzeichen*,
6. $\delta\colon Q \times \Gamma \to Q \times \Gamma \times \{\mathrm{L}, \mathrm{R}\}$ die *Übergangsfunktion*,
7. $s \in Q$ der *Startzustand*,
8. $t \in Q$ der *akzeptierende Zustand* und

9. $r \in Q$ der *verwerfende Zustand* mit $t \neq r$.

Informell bedeutet $\delta(p, a) = (q, b, d)$: "Wenn die TM M im Zustand p das Zeichen a liest, dann wechselt M in den Zustand q, ersetzt am Band das Zeichen a durch das Zeichen b und der Schreib-/Lesekopf bewegt sich einen Schritt in die Richtung d." Wir verlangen, dass das Symbol $\vdash$ niemals (durch ein anderes Zeichen) überschrieben werden kann und die Turingmaschine niemals über die linke Begrenzung hinaus fährt. Dies wird formal durch die folgende Bedingung festgelegt: Für alle $p \in Q$ existiert ein $q \in Q$ mit:

$$\delta(p, \vdash) = (q, \vdash, \mathrm{R})\,. \tag{9.1}$$

Außerdem verlangen wir, dass die Turingmaschine, sollte sie den akzeptierenden bzw. verwerfenden Zustand erreicht haben, diesen nicht mehr verlassen kann. Das heißt für alle $a \in \Gamma$ existieren $b, b' \in \Gamma$ und $d, d' \in \{\mathrm{L}, \mathrm{R}\}$ sodass gilt:

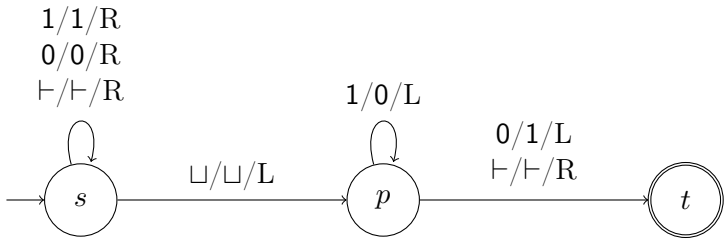
$$\delta(t, a) = (t, b, d) \tag{9.2}$$
$$\delta(r, a) = (r, b', d') \tag{9.3}$$

Die Zustandsmenge Q und die Übergangsfunktion δ einer TM M werden auch als die *endliche Kontrolle* von M bezeichnet.

Beispiel 9.1. Sei $M = (\{s, p, t, r\}, \{0, 1\}, \{\vdash, \sqcup, 0, 1\}, \vdash, \sqcup, \delta, s, t, r)$ eine Turingmaschine wobei die Übergangsfunktion δ durch die *Zustandstabelle*

	$\vdash$	0	1	$\sqcup$
s	$(s, \vdash, \mathrm{R})$	$(s, 0, \mathrm{R})$	$(s, 1, \mathrm{R})$	$(p, \sqcup, \mathrm{L})$
p	$(t, \vdash, \mathrm{R})$	$(t, 1, \mathrm{L})$	$(p, 0, \mathrm{L})$	$\cdot$

oder durch das *Zustandsübergangsdiagramm*



angegeben werden kann.

In der Tabelle sowie im Zustandsübergangsdiagramm wurde auf die Übergänge für die Zustände t und r verzichtet. Diese können so gewählt werden, dass der Zustand nicht geändert wird, das gelesene Zeichen wieder geschrieben wird und sich der Schreib-/Lesekopf nach rechts bewegt. Trivialerweise sind die Bedingungen (9.2)–(9.3) dann erfüllt. Ebenso kann der fehlende Übergang für $\delta(p, \sqcup)$ gewählt werden.

Um eine TM M auf der Eingabe $x \in \Sigma^*$ laufen zu lassen, wird

$$\vdash x \sqcup^\infty$$

auf das Band geschrieben. Dabei dient $\vdash$ dazu, dass die Maschine nicht über das linke Bandende hinausläuft (siehe Bedingung (9.1)), x ist die eigentliche Eingabe und rechts von der Eingabe stehen lauter $\sqcup$ am Band. Zu jedem Zeitpunkt enthält das Band einer TM M ein (unendliches) Wort der Form $y\sqcup^\infty$ mit $y_0 = \vdash$ und $y \in \Gamma^*$.

Um Berechnungsschritte einer Turingmaschine formal zu beschreiben, verwenden wir Konfigurationen.

Definition 9.2. Eine *Konfiguration* einer TM M ist ein Tripel $(p, y\sqcup^\infty, n)$, sodass

- $p \in Q$ der aktuelle Zustand,
- $y\sqcup^\infty$ der aktuelle Bandinhalt ($y \in \Gamma^*$) und
- $n \in \mathbb{N}$ die Position des Schreib-/Lesekopfes am Band sind.

Die *Startkonfiguration* bei Eingabe $x \in \Sigma^*$ ist die Konfiguration

$$(s, \vdash x\sqcup^\infty, 0)\,.$$

Wir definieren eine binäre Relation zwischen Konfigurationen, um einen Rechenschritt einer Turingmaschine zu formalisieren.

Definition 9.3. Sei $n \in \mathbb{N}$ und $y = y_0 \cdots y_{m-1} \in \Gamma^*$ mit $m > n$.[1] Die Relation $\xrightarrow[M]{1}$ ist wie folgt definiert:

$$(p, y\sqcup^\infty, n) \xrightarrow[M]{1} \begin{cases} (q, y_0 \ldots y_{n-1} b y_{n+1} \ldots y_{m-1}\sqcup^\infty, n-1) & \text{wenn } \delta(p, y_n) = (q, b, \mathrm{L}) \\ (q, y_0 \ldots y_{n-1} b y_{n+1} \ldots y_{m-1}\sqcup^\infty, n+1) & \text{wenn } \delta(p, y_n) = (q, b, \mathrm{R}) \end{cases}$$

In der Folge verwenden wir griechische Buchstaben vom Anfang des Alphabets um Konfigurationen zu bezeichnen.

Definition 9.4. Wir definieren die reflexive, transitive Hülle $\xrightarrow[M]{*}$ von $\xrightarrow[M]{1}$ induktiv:

1. $\alpha \xrightarrow[M]{0} \alpha$ für jede Konfiguration α
2. $\alpha \xrightarrow[M]{n+1} \beta$, wenn $\alpha \xrightarrow[M]{n} \gamma \xrightarrow[M]{1} \beta$ für eine Konfiguration γ und
3. $\alpha \xrightarrow[M]{*} \beta$, wenn $\alpha \xrightarrow[M]{n} \beta$ für ein $n \geqslant 0$.

Beispiel 9.2. Für die TM M aus Beispiel 9.1 gilt

$$(s, \vdash\mathsf{0010}\sqcup^\infty, 0) \xrightarrow[M]{*} (t, \vdash\mathsf{0011}\sqcup^\infty, 3)\,.$$

Definition 9.5 (Sprache einer TM)**.** Eine TM $M = (Q, \Sigma, \Gamma, \vdash, \sqcup, \delta, s, t, r)$ *akzeptiert* die Eingabe $x \in \Sigma^*$, wenn

$$(s, \vdash x\sqcup^\infty, 0) \xrightarrow[M]{*} (t, y\sqcup^\infty, n)\,,$$

für beliebige $y \in \Gamma^*$ und $n \in \mathbb{N}$ und *verwirft* die Eingabe $x \in \Sigma^*$, wenn

$$(s, \vdash x\sqcup^\infty, 0) \xrightarrow[M]{*} (r, y\sqcup^\infty, n)\,,$$

für beliebige $y \in \Gamma^*$ und $n \in \mathbb{N}$. Die Menge $\mathsf{L}(M)$ bezeichnet die Menge aller von M akzeptierten Wörter, und heißt die von M *akzeptierte Sprache*.

[1] Weil $\sqcup \in \Gamma$, ist $m > n$ keine Einschränkung, weil wir y am Ende einfach mit $\sqcup$ auffüllen können.

Beispiel 9.3. Für die TM

$$M = (\{s, q_0, q_1, q_0', q_1', q, t, r\}, \{0, 1\}, \{\vdash, \sqcup, 0, 1\}, \vdash, \sqcup, \delta, s, t, r)$$

mit δ gegeben durch die Zustandstabelle

	$\vdash$	0	1	$\sqcup$
s	$(s, \vdash, R)$	$(q_0, \vdash, R)$	$(q_1, \vdash, R)$	$(t, \sqcup, L)$
q_0	$\cdot$	$(q_0, 0, R)$	$(q_0, 1, R)$	$(q_0', \sqcup, L)$
q_1	$\cdot$	$(q_1, 0, R)$	$(q_1, 1, R)$	$(q_1', \sqcup, L)$
q_0'	$(r, \vdash, R)$	$(q, \sqcup, L)$	$(r, \sqcup, L)$	$\cdot$
q_1'	$(r, \vdash, R)$	$(r, \sqcup, L)$	$(q, \sqcup, L)$	$\cdot$
q	$(s, \vdash, R)$	$(q, 0, L)$	$(q, 1, L)$	$\cdot$

ist $\mathsf{L}(M) = \{w \in \{0, 1\}^* \mid w \text{ ist ein Palindrom gerader Länge}\}$.

Eine TM M (nach Definition 9.1) *kommt niemals zur Ruhe*. Da M den akzeptierenden bzw. verwerfenden Zustand (aufgrund der Bedingungen (9.2),(9.3)) nicht mehr verlassen kann, sprechen wir dennoch vom *Halten* der TM M.

Definition 9.6 (Totale Turingmaschine)**.** Eine TM M *hält* bei Eingabe x, wenn M die Eingabe x akzeptiert oder verwirft. Andernfalls *hält* M auf x *nicht*. Eine TM M heißt *total*, wenn sie auf allen Eingaben hält.

Beispiel 9.4. Die Turingmaschinen aus den Beispielen 9.1 und 9.3 sind total.

Definition 9.7 (rekursiv, rekursiv aufzählbar, co-rekursiv aufzählbar)**.** Eine Sprache L (oder allgemein eine Menge) heißt *rekursiv aufzählbar*, wenn eine TM M existiert, sodass $L = \mathsf{L}(M)$. Eine Sprache L heißt *co-rekursiv aufzählbar* wenn sie das Komplement einer rekursiv aufzählbaren Sprache ist. Eine Sprache L heißt *rekursiv*, wenn es eine *totale* TM M gibt, sodass $L = \mathsf{L}(M)$.

In der Programmierung nennt man Algorithmen rekursiv, die sich selbst aufrufen. Hier wird die selbe Bezeichnung für ein anderes Konzept verwendet.[2]

Rekursive Mengen sind unter Komplementbildung abgeschlossen.

Satz 9.1. *Sei Σ ein Alphabet und $L \subseteq \Sigma^*$ rekursiv. Dann ist $\sim L$ rekursiv.*

Beweis. Da L rekursiv ist, gibt es eine totale TM M mit $L = \mathsf{L}(M)$. Wir definieren eine TM M', wobei der akzeptierende und der verwerfende Zustand von M vertauscht werden. Weil M total ist, ist auch M' total. Somit akzeptiert M' ein Wort genau dann, wenn M es verwirft und es folgt $\sim L = \mathsf{L}(M')$, d.h. $\sim L$ ist rekursiv. □

Satz 9.2. *Jede rekursive Menge ist rekursiv aufzählbar. Andererseits ist nicht jede rekursiv aufzählbare Menge rekursiv.*

Beweis. Der erste Teil des Satzes ist eine triviale Konsequenz der Definitionen (jede totale TM ist eine TM). Den Beweis des zweiten Teiles holen wir in Sektion 9.3 nach. □

[2] Dennoch gibt es einen Zusammenhang. Rekursive Funktionen können nämlich ebenfalls genau die Klasse von rekursiven Mengen, d.h., entscheidbare Probleme repräsentieren.

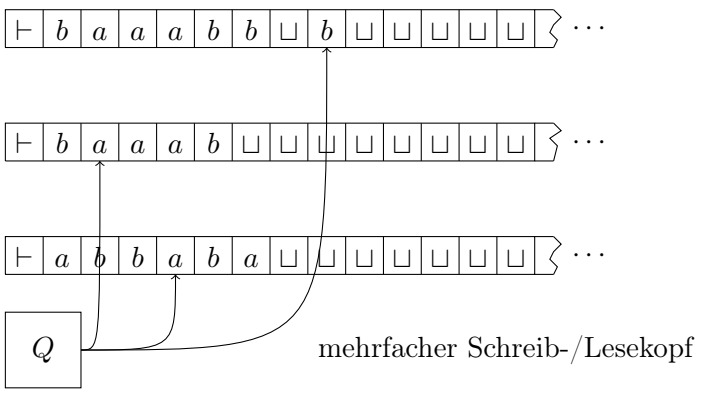


Abbildung 9.1: 3-bändige Turingmaschine

Satz 9.3. *Wenn A und $\sim A$ rekursiv aufzählbar sind, dann ist A rekursiv.*

Beweis. *Angenommen es existieren Turingmaschinen M und M', sodass $A = \mathsf{L}(M)$ und $\sim(A) = \mathsf{L}(M')$. Wir definieren eine neue TM N, die bei Eingabe x die Maschinen M und M' simultan auf x ausführt. Dazu teilen wir das Band von N in eine obere und untere Hälfte, sodass wir M auf der oberen und M' auf der unteren Hälfte ausführen können. Formal erreichen wir das, indem das Bandalphabet von N die folgenden Symbole enthält:*

$$\boxed{\begin{matrix} a \\ c \end{matrix}} \qquad \boxed{\begin{matrix} \hat{a} \\ c \end{matrix}} \qquad \boxed{\begin{matrix} a \\ \hat{c} \end{matrix}} \qquad \boxed{\begin{matrix} \hat{a} \\ \hat{c} \end{matrix}}$$

Hier ist a ein Bandsymbol von M und c ein Bandsymbol von M'. Die Zusatzmarkierung $\hat{\cdot}$ wird verwendet, um die Position des Schreib-/Lesekopfes von M bzw. M' zu markieren. Das Band von N kann also folgende Gestalt haben:

$$\left|\begin{matrix} b \\ c \end{matrix}\right|\begin{matrix} \hat{b} \\ c \end{matrix}\left|\begin{matrix} a \\ c \end{matrix}\right|\begin{matrix} b \\ d \end{matrix}\left|\begin{matrix} a \\ d \end{matrix}\right|\begin{matrix} a \\ d \end{matrix}\left|\begin{matrix} a \\ c \end{matrix}\right|\begin{matrix} a \\ \hat{c} \end{matrix}\left|\begin{matrix} b \\ d \end{matrix}\right|\begin{matrix} a \\ c \end{matrix}\left|\begin{matrix} a \\ d \end{matrix}\right|\begin{matrix} a \\ c \end{matrix}\right\rangle \ldots$$

Die Maschine N führt abwechselnd einen Schritt von M oder einen Schritt von M' aus. Dazu werden die jeweiligen Zustände von M und M' durch zusätzliche Zustände in N gespeichert. Wenn M jemals akzeptiert, hält N und akzeptiert. Andererseits wenn M' akzeptiert, dann verwirft N. Nach Voraussetzung muss einer der Fälle für jede Eingabe auf jeden Fall eintreten. Somit hält N auf allen Eingaben und ist total. □

Im Beweis von Satz 9.3 haben wir bereits gesehen, dass das Band einer Turingmaschine geteilt werden kann. Im Folgenden untersuchen wir die Erweiterung von Turingmaschinen auf mehrere Bänder und Schreib-/Leseköpfe, wie in Abbildung 9.1 dargestellt. Allgemein hat die Übergangsfunktion einer k-bändigen Turingmaschine den folgenden Typ:

$$\delta\colon Q \times \Gamma^k \to Q \times \Gamma^k \times \{\mathrm{L}, \mathrm{R}\}^k\,.$$

Die formale Definition einer *k-bändigen deterministischen Turingmaschine* überlassen wir als Übung (Aufgabe 9.5).

Satz 9.4. *Sei M eine k-bändige TM. Dann existiert eine (einbändige) TM M', sodass $\mathsf{L}(M) = \mathsf{L}(M')$.*

Beweis.[Skizze] In der Literatur finden sich zwei unterschiedliche Ansätze, eine mehrbändige Turingmaschine durch eine einbändige zu simulieren. Entweder werden die k Bänder von M hintereinander, durch Sonderzeichen getrennt, auf dem Band von M' simuliert oder das Bandalphabet von M' besteht aus k-Tupeln von Bandsymbolen in M (mit möglichen Markierungen für die aktuelle Position des Schreib-/Lesekopfes). Dabei folgt man der Konstruktion im Beweis von Satz 9.3. Die formale Ausarbeitung dieser Beweisskizze überlassen wir als Übung (Aufgabe 9.6). □

9.2 Nichtdeterministische Turingmaschinen

Nichtdeterministische Turingmaschinen erweitert deterministische Turingmaschinen um die Möglichkeit, dass die Berechnung zu jedem Zeitpunkt anhand von mehreren Möglichkeiten voranschreiten kann.

Definition 9.8 (Nichtdeterministische Turingmaschine)**.** Eine *nichtdeterministische (einbändige) Turingmaschine N* (kurz NTM) ist ein 9-Tupel

$$N = (Q, \Sigma, \Gamma, \vdash, \sqcup, \delta, s, t, r)\,,$$

sodass

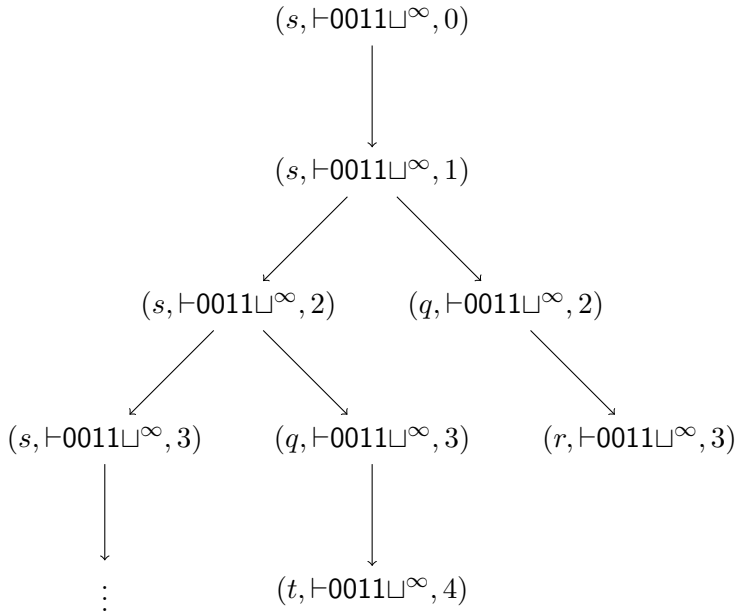
1. Q eine endliche Menge von *Zuständen*,
2. Σ eine endliche Menge von *Eingabesymbolen*,
3. $\Gamma \supseteq \Sigma$ eine endliche Menge von *Bandsymbolen*,
4. $\vdash \in \Gamma \setminus \Sigma$, der *linke Endmarker*,
5. $\sqcup \in \Gamma \setminus \Sigma$, das *Leerzeichen*,
6. $\delta\colon Q \times \Gamma \to \mathcal{P}(Q \times \Gamma \times \{\mathrm{L}, \mathrm{R}\})$ die *Übergangsfunktion*,
7. $s \in Q$, der *Startzustand*,
8. $t \in Q$, der *akzeptierende Zustand* und
9. $r \in Q$, der *verwerfende Zustand* mit $t \neq r$.

Die Rechenschritte einer NTM (für eine Eingabe $x \in \Sigma^*$) können als Baum aufgefasst werden. Dabei sind die Ecken des Baumes Konfigurationen der NTM und es gibt eine Kante zwischen zwei Konfigurationen α und β, wenn die NTM von der Konfiguration α in einem Schritt in die Konfiguration β wechseln kann. Die Wurzel des Baumes ist die Startkonfiguration $(s, \vdash x \sqcup^\infty, 0)$ und ein Weg im Berechnungsbaum repräsentiert somit eine konkrete Möglichkeit, wie die NTM ihre Berechnung durchführt. Damit eine NTM N ihre Eingabe akzeptiert, muss nur eine Möglichkeit, also ein Weg, existieren, sodass N einen akzeptierenden Zustand erreicht.

Beispiel 9.5. Für die NTM $N = (\{s, q, r, t\}, \{\mathsf{0}, \mathsf{1}\}, \{\vdash, \sqcup, \mathsf{0}, \mathsf{1}\}, \vdash, \sqcup, \delta, s, t, r)$ mit δ gegeben durch

	$\vdash$	$\mathsf{0}$	$\mathsf{1}$	$\sqcup$
s	$\{(s, \vdash, \mathrm{R})\}$	$\{(s, \mathsf{0}, \mathrm{R}), (q, \mathsf{0}, \mathrm{R})\}$	$\{(s, \mathsf{1}, \mathrm{R})\}$	$\{(r, \sqcup, \mathrm{R})\}$
q	$\cdot$	$\{(r, \mathsf{0}, \mathrm{R})\}$	$\{(t, \mathsf{1}, \mathrm{R})\}$	$\{(r, \sqcup, \mathrm{R})\}$

ergibt sich für das Wort $\mathsf{0011}$ der folgende Berechnungsbaum:



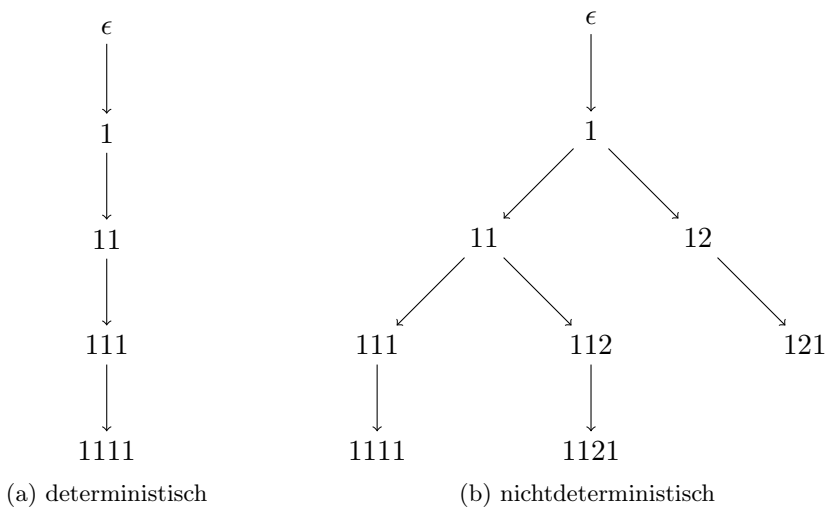
Somit gilt $\mathsf{0011} \in \mathsf{L}(N)$.

Schematisch lässt sich der Unterschied zwischen Determinismus und Nichtdeterminismus wie in Abbildung 9.2 darstellen. Zur Adressierung der Konfigurationen im Berechnungsbaum werden Wörter über dem Alphabet $\Sigma_b = \{1, 2, \ldots, b\}$ verwendet, wobei b den maximalen Ausgangsgrad einer Ecke im Berechnungsbaum darstellt. Man nennt b auch den *Grad des Nichtdeterminismus* von N. Offensichtlich repräsentiert jedes Wort $w \in \Sigma_b^*$ entweder eine eindeutige Position in diesem Berechnungsbaum oder ist ungültig (wenn an einer Stelle im Baum zu wenig Möglichkeiten bestehen). Das leere Wort ϵ adressiert die Wurzel des Berechnungsbaumes, das heißt den Beginn der Berechnung. Für eine Konfiguration α mit Adresse π und Nachfolgekonfigurationen $\alpha_1, \ldots, \alpha_n$ (d.h. $\alpha \xrightarrow[M]{1} \alpha_i$ für $1 \leq i \leq n$) erhält die Konfiguration α_i die Adresse πi für $1 \leq i \leq n$.

Satz 9.5. *Sei N eine NTM. Dann existiert eine DTM M, sodass $\mathsf{L}(M) = \mathsf{L}(N)$. Umgekehrt wird jede von einer DTM akzeptierte Sprache auch von einer NTM akzeptiert.*

Beweis. Wir zeigen den ersten Teil des Satzes; der zweite Teil ist einfach. Sei N eine NTM. Wir konstruieren eine dreibändige DTM M mit $\mathsf{L}(N) = \mathsf{L}(M)$.

Sei x das Eingabewort für N. Das erste Band von M wird immer nur diese Eingabe x enthalten. Auf dem zweiten Band simulieren wir die Rechenschritte von N bezüglich eines *Weges im Berechnungsbaum. Schließlich dient das dritte Band dazu, den aktuellen Weg zu adressieren.*



(a) deterministisch (b) nichtdeterministisch

Abbildung 9.2: Berechnungsbaum mit Adressen

Mithilfe der Adressierung von Wegen können wir das Verhalten von N wie folgt in M simulieren:

1. *Anfangs enthält Band 1 von M das Eingabewort x, die Bänder 2 und 3 sind leer.*
2. *Kopiere den Inhalt von Band 1 auf Band 2.*
3. *Verwende Band 2, um die Rechenschritte von N auf x zu simulieren. Bei jeder Stelle in der Berechnung, in der die Übergangsfunktion δ mehrere Möglichkeiten zulässt, sieht M auf Band 3 nach, welche Möglichkeit gewählt werden soll. Es wird also genau ein Weg im Berechnungsbaum auf Band 2 simuliert. Wenn die Adresse auf Band 3 ungültig ist, oder keine weiteren Rechenschritte mehr betrachtet werden können, gehe zu Punkt (4). Andernfalls, wenn die Simulation von N akzeptiert, dann akzeptiere.*
4. *Ersetze das Wort auf Band 3 durch seinen unmittelbaren Nachfolger in der graduiert-lexikographischen Ordnung auf Wörtern $\leq_{gradlex}$ (Satz 2.10). Gehe zu Schritt (2).*

□

9.3 Unentscheidbarkeit

Die folgende Definition (in Kombination mit Definition 9.7) erlaubt uns Entscheidbarkeit mittels Turingmaschinen formal zu beschreiben.

Definition 9.9 (Entscheidbarkeit, Semi-Entscheidbarkeit)**.** Sei Σ ein Alphabet. Eine Eigenschaft P von Wörtern über Σ heißt

– *entscheidbar* genau dann, wenn die Menge $\{x \in \Sigma^* \mid x \text{ hat Eigenschaft } P\}$ rekursiv ist,

– *semi-entscheidbar* genau dann, wenn die Menge $\{x \in \Sigma^* \mid x \text{ hat Eigenschaft } P\}$ rekursiv aufzählbar ist.

Beispiel 9.6. Sei $P(x) := x$ ist ein Palindrom gerader Länge. Dann ist P entscheidbar.

Beispiel 9.7. Jedes entscheidbare Problem ist semi-entscheidbar.

Gemäß Definitionen 9.9 und 9.7 ist ein Problem P genau dann semi-entscheidbar, wenn es eine TM M gibt, deren Sprache alle Wörter sind, welche die Eigenschaft P haben. Weiters ist ein Problem P entscheidbar, wenn es eine totale TM M gibt, sodass M genau jene Wörter akzeptiert, welche die Eigenschaft P haben.

Nun widmen wir uns dem zweiten Teil des Beweises von Satz 9.2. Dazu werden wir zeigen, dass die Menge der Turingmaschinen, die auf einer Eingabe halten, zwar rekursiv aufzählbar, aber nicht rekursiv ist. Eine wichtige Grundlage für dieses Resultat ist die Tatsache, dass Turingmaschinen ihren eigenen Code interpretieren können.

9.3.1 Universelle Turingmaschine

Um eine Turingmaschine als Eingabe für eine Turingmaschine zu verwenden, müssen wir uns auf eine sinnvolle Codierung von Turingmaschinen einigen. Diese Codierung sollte einfach sein und alle notwendigen Informationen wie etwa Anzahl der Zustände, Eingabe- und Bandalphabet, linken Endmarker, Leerzeichen, Übergangsfunktion, Startzustand, akzeptierenden und verwerfenden Zustand enthalten. Wir stellen alle nötigen Informationen als Zahlen dar, die wir unär codieren.

Als Beispiel könnte die Codierung mit dem folgenden Wort beginnen

$$0^n 1 0^m 1 0^k 1 0^u 1 0^v 1 0^s 1 0^t 1 0^r 1\,,$$

wobei dieses Wort eine Turingmaschine mit den n Zuständen $\{0, \ldots, n-1\}$, den m Eingabezeichen $\{0, \ldots, m-1\}$, den $m+k$ Bandzeichen $\{0, \ldots, m-1, m, \ldots, m+k-1\}$, dem linken Endmarker u $(m \leqslant u < m+k)$, dem Leerzeichen v $(m \leqslant v < m+k)$, dem Startzustand s $(0 \leqslant s < n)$, dem akzeptierenden Zustand t $(0 \leqslant t < n)$ und dem verwerfenden Zustand r $(0 \leqslant r < n)$ codiert. Das Zeichen 1 dient als Trennzeichen. Der Rest des Codesegments kann nun aus einer Sequenz von Wörtern bestehen, welche die Übergangsfunktion codieren, etwa könnte $\delta(p, a) = (q, b, d)$ wie folgt dargestellt werden:

$$0^p 1 0^a 1 0^q 1 0^b 1 0^c 1\,,$$

wobei $c = 0$ wenn $d = \mathrm{L}$ und $c = 1$ wenn $d = \mathrm{R}$.

Definition 9.10. Aufbauend auf diese Codierung von Turingmaschinen, konstruieren wir eine *universelle Turingmaschine* U (kurz UTM) sodass

$$\mathsf{L}(U) = \{\ulcorner M \urcorner \# \ulcorner x \urcorner \mid x \in \mathsf{L}(M)\}\,.$$

Das Symbol # ist ein Hilfssymbol im Eingabealphabet von U, um den Code $\ulcorner M \urcorner$ der TM M vom Code $\ulcorner x \urcorner$ der Eingabe für M abzugrenzen. Die UTM U operiert wie folgt:

1. Zunächst prüft U, ob $\ulcorner M \urcorner$ tatsächlich der Code einer TM M ist und $\ulcorner x \urcorner$ die Codierung eines Eingabewortes über dem Eingabealphabet von M darstellt. Falls nicht, wird U sofort verwerfen.

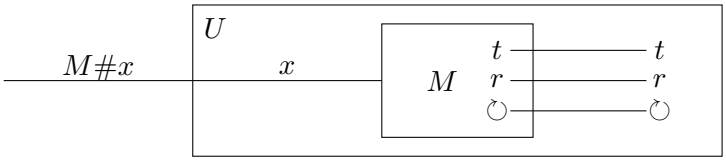


Abbildung 9.3: Universelle Turingmaschine

2. Wenn die Codierung gültig ist, simuliert U die TM M auf der Eingabe x. Dazu verwenden wir drei Bänder: Auf Band 1 steht während der ganzen Simulation die Beschreibung von M. Auf Band 2 wird zu Beginn das Eingabewort x geschrieben. Das dritte Band speichert den aktuellen Zustand von M und die Position des Schreib-/Lesekopfes. Nun simuliert die TM U Schritt für Schritt die TM M, wobei auf Band 1 der entsprechende Übergang gesucht wird, der Inhalt von Band 2 entsprechend geändert wird und auf Band 3 der neue Zustand bzw. die neue Position des Schreib-/Lesekopfes geschrieben werden.

3. Wenn M akzeptiert, so akzeptiert U ebenfalls. Wenn M verwirft, so verwirft U ebenfalls.

Im Folgenden lassen wir die expliziten Hinweise auf die Codierung weg und schreiben M für $\ulcorner M \urcorner$ und x für $\ulcorner x \urcorner$. Somit ist $\mathsf{L}(U) = \{M\#x \mid x \in \mathsf{L}(M)\}$ bzw. $M\#x \in \mathsf{L}(U) \Leftrightarrow x \in \mathsf{L}(M)$. Weiters hält U auf $M\#x$ genau dann, wenn M auf x hält. Schematisch kann U wie in Abbildung 9.3 dargestellt werden.

9.3.2 Diagonalisierung

Universelle Turingmaschinen zusammen mit der *Diagonalisierungsmethode* können verwendet werden, um bestimmte Sprachen als nicht rekursiv nachzuweisen. Eine Anwendung der Diagonalisierungsmethode wurde bereits in Satz 6.12 benutzt.

Definition 9.11. Wir definieren die dem *Halteproblem* (engl. halting problem) und *Zugehörigkeitsproblem* (engl. membership problem) von Turingmaschinen entsprechenden Mengen:

$$\begin{aligned} \mathsf{HP} &:= \{M\#x \mid M \text{ hält bei Eingabe } x\} \\ \mathsf{MP} &:= \{M\#x \mid x \in \mathsf{L}(M)\}\,. \end{aligned}$$

Wir machen uns die Codierungsmöglichkeit von Turingmaschinen zunutze, um eine Aufzählung aller Turingmaschinen anzugeben.

Definition 9.12. Sei M_x eine TM mit *Eingabealphabet* $\{\mathtt{0}, \mathtt{1}\}$, deren Code x ist. Wenn x keine gültige Beschreibung einer Turingmaschine darstellt, definieren wir M_x als eine beliebige aber fixe TM über dem Eingabealphabet $\{\mathtt{0}, \mathtt{1}\}$.

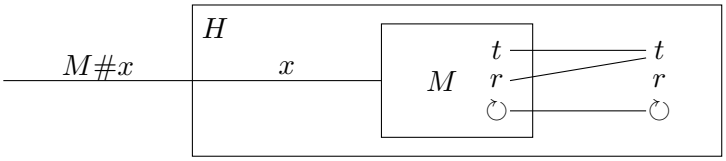


Abbildung 9.4: Eine Turingmaschine für das Halteproblem ($\mathsf{L}(H) = \mathsf{HP}$)

Als Konsequenz von Definition 9.12 erhalten wir eine unendliche Liste von Turingmaschinen, deren Programmcodes in aufsteigender Reihenfolge (bezüglich der graduiert lexikographischen Ordnung auf Wörtern) geordnet sind

$$M_\epsilon, M_0, M_1, M_{00}, M_{01}, M_{10}, M_{11}, M_{000}, M_{001}, \ldots . \tag{9.4}$$

Diese Liste enthält alle möglichen Turingmaschinen mit dem Eingabealphabet $\{0, 1\}$. In weiterer Folge betrachten wir eine zweidimensionale Matrix, einerseits indiziert mit Wörtern $w \in \{0, 1\}^*$ und andererseits mit den Turingmaschinen aus der Liste (9.4).

	ϵ	0	1	00	01	10	11	000	001	010	...
M_ϵ	!	↻	↻	!	!	↻	!	↻	!	!	
M_0	↻	↻	!	!	↻	!	!	↻	↻	!	
M_1	↻	!	↻	!	↻	!	!	↻	↻	!	
M_{00}	!	↻	↻	!	!	!	!	↻	↻	!	
M_{01}	!	!	!	!	↻	↻	↻	!	!	↻	...
M_{10}	!	!	↻	!	!	↻	!	!	↻	!	
M_{11}	!	!	↻	↻	!	↻	!	↻	!	↻	
M_{000}	!	!	!	!	↻	!	!	↻	!	↻	
M_{001}	↻	!	!	!	!	↻	!	!	!	!	
⋮						⋮					⋱

Eine Zeile dieser Matrix beschreibt für jedes Eingabewort y, ob M_x auf y hält oder nicht. Hier wird das Halten von M_x durch ! und das Nicht-halten durch ↻ ausgedrückt.

Satz 9.6. *Die Menge* HP *ist rekursiv aufzählbar, aber nicht rekursiv.*

Beweis. Die Menge HP *ist rekursiv aufzählbar, da es eine TM H gibt, die für die Eingabe $M\#x$ feststellen kann, ob M die Codierung einer Turingmaschine ist und x ein Wort aus dem Eingabealphabet von M. Nun simuliert H die Maschine M auf der Eingabe x und akzeptiert, wenn M hält. Graphisch lässt sich H wie in Abbildung 9.4 darstellen.*

Um zu zeigen, dass HP *nicht rekursiv ist, verfahren wir indirekt: Angenommen* HP *ist rekursiv. Dann existiert eine totale TM K, sodass* $\mathsf{HP} = \mathsf{L}(K)$ *(siehe Abbildung 9.5). Somit akzeptiert K die Eingabe $M\#x$ wenn M auf x hält und verwirft $M\#x$ wenn M auf x nicht hält. Basierend auf K definieren wir eine* Diagonaliserungsmaschine *für das Halteproblem D:*

1. *Die TM D erhält als Eingabe ein Wort $x \in \{0, 1\}^*$. Mit Hilfe von x konstruiert D die TM M_x und schreibt $M_x\#x$ auf ihr Eingabeband.*

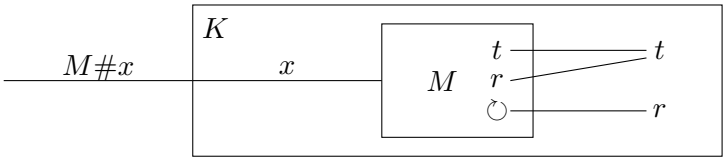


Abbildung 9.5: Eine totale Turingmaschine für das Halteproblem ($\mathsf{L}(K) = \mathsf{HP}$)

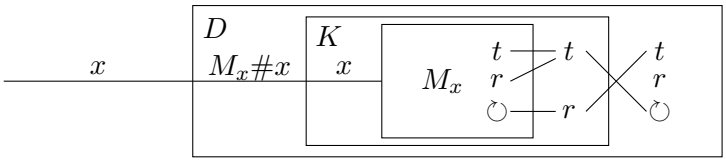


Abbildung 9.6: Diagonalisierungsmaschine für das Halteproblem

2. *Dann simuliert D die TM K auf der Eingabe $M_x\#x$, allerdings startet D eine nicht-terminierende Schleife, wenn K akzeptiert und akzeptiert, wenn K verwirft.*

Abbildung 9.6 stellt die TM D graphisch dar. Laut Konstruktion erhalten wir

$$\begin{aligned} D \text{ hält auf } x &\Leftrightarrow K \text{ verwirft } M_x\#x && \text{Definition von } D \\ &\Leftrightarrow M_x \text{ hält nicht auf } x && \text{Definition von } K \end{aligned}$$

Damit ist D verschieden von jeder der Turingmaschinen M_x aus (9.4) (auf dem Eingabewort x). Genauer erhalten wir

	ϵ	0	1	00	01	10	11	000	001	...
D	↻	!	!	↻	!	!	↻	!	↻	...

Weil die Liste (9.4) laut Annahme alle Turingmaschinen enthält, D darin aber fehlt, erhalten wir einen Widerspruch zur Annahme, dass HP rekursiv ist. Also ist HP nicht rekursiv. □

Ganz nebenbei erhalten wir eine Sprache, die nicht rekursiv aufzählbar ist.

Folgerung. *Die Menge $\sim\mathsf{HP}$ ist nicht rekursiv aufzählbar.*

Beweis. Wir verfahren indirekt. Angenommen $\sim\mathsf{HP}$ sei rekursiv aufzählbar. Aus Satz 9.6 wissen wir, dass HP rekursiv aufzählbar ist. Somit ist laut Satz 9.3 die Menge HP auch rekursiv, was im Widerspruch zu Satz 9.6 ist. □

Satz 9.6 besagt, dass das Halteproblem für Turingmaschinen im Allgemeinen unentscheidbar ist. Dennoch kann für bestimmte Turingmaschinen das Halteproblem entscheidbar sein (z.B. wenn die Übergangsfunktion bei allen Eingaben vom Startzustand aus sofort in den akzeptierenden Zustand wechselt).

Es ist möglich beliebige Programme in C, Java, etc. in Turingmaschinen umzuwandeln. Darüber hinaus gilt sogar, dass jedes im Laufe der letzten Dekaden vorgeschlagene abstrakte Rechenmodell (siehe etwa Sektion 9.4) äquivalent zu Turingmaschinen ist.

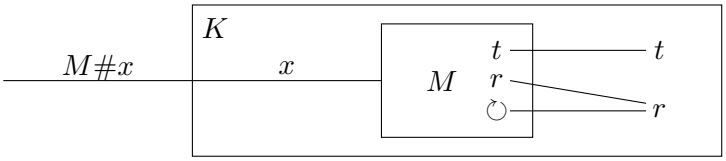


Abbildung 9.7: Eine totale Turingmaschine für das Zugehörigkeitsproblem ($\mathsf{L}(K) = \mathsf{MP}$)

Diese Beobachtung hat schon in den 1930er Jahren die Grundlage für die sogenannte *Church-Turing These* geliefert:

These. *Jedes algorithmisch lösbare Problem ist auch mit Hilfe einer Turingmaschine lösbar.* □

Es gibt auch Probleme, die einfacher zu formulieren sind als das Halteproblem, aber dennoch nicht entscheidbar sind. Die folgenden Sätze geben wir ohne Beweis an und verweisen die interessierte Leserin auf [4] oder [3].

Definition 9.13. Wir definieren das Postsche Korrespondenzproblem (*PCP*). Gegeben zwei Listen von Wörtern

$$w_1, w_2, \ldots, w_n \quad \text{und} \quad x_1, x_2, \ldots, x_n\,.$$

Gesucht sind Indizes $i_1, i_2, \ldots, i_m$ mit $m \geq 1$, sodass

$$w_{i_1} w_{i_2} \ldots w_{i_m} = x_{i_1} x_{i_2} \ldots x_{i_m}\,.$$

Wir nennen die Indexliste $i := i_1 i_2 \ldots i_m$ dann eine *Lösung*.

Satz 9.7. *Das Problem ob ein PCP eine Lösung hat ist semi-entscheidbar, aber nicht entscheidbar.* □

Beispiel 9.8. Das PCP mit den Listen 0, 01, 110 und 100, 00, 11 hat die Lösung $3, 2, 3, 1$.

Satz 9.8. *Ob eine formale Sprache regulär ist, ist nicht semi-entscheidbar.* □

Folgerung. *Ob eine formale Sprache regulär ist, ist nicht entscheidbar.* □

9.3.3 Reduktion

Satz 9.9. *Die Menge* MP *ist rekursiv aufzählbar, aber nicht rekursiv.*

Beweis. Um zu zeigen, dass MP *rekursiv aufzählbar ist, verwenden wir die universelle Turingmaschine U. Weil $\mathsf{L}(U) = \mathsf{MP}$, sind wir fertig.*

Um zu zeigen, dass MP *nicht rekursiv ist, argumentieren wir wieder indirekt. Angenommen* MP *wäre rekursiv, dann existiert eine totale TM K (siehe Abbildung 9.7), sodass* $\mathsf{MP} = \mathsf{L}(K)$.

Für den gewünschten Widerspruch zeigen wir, dass wir aus K eine totale Turingmaschine konstruieren können, deren Sprache HP *ist (Widerspruch zu Satz 9.13).*

Um für eine beliebige TM M festzustellen, ob M auf x hält, transformieren wir M wie folgt: Die TM N, ist genau wie M definiert, aber mit der Ausnahme, dass wir jeweils vom

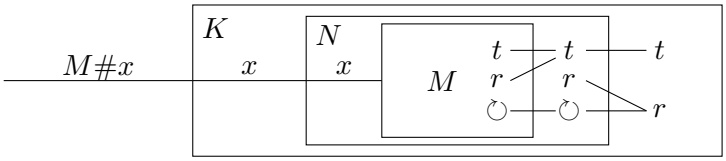


Abbildung 9.8: Reduktion des Halteproblems auf das Zugehörigkeitsproblem

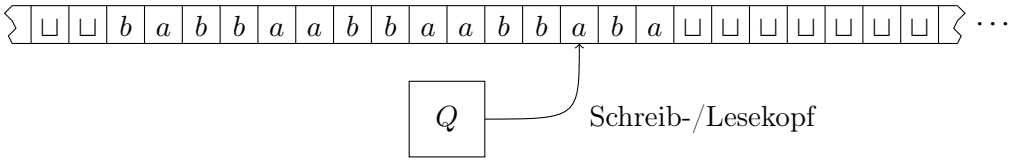


Abbildung 9.9: Beidseitig unendliches Band

akzeptierenden und verwerfenden Zustand von M in den akzeptierenden Zustand von N wechseln. Wenn M auf x nicht hält, so hält auch N auf x nicht. Also akzeptiert N das Wort x genau dann, wenn M auf x hält. Nun können wir eine totale Turingmaschine angeben, deren Sprache HP *ist (siehe Abbildung 9.8).*

Nach Satz 9.13 kann es solch eine totale Turingmaschine aber nicht geben. Somit ist die Annahme, dass MP *rekursiv ist falsch und folglich ist* MP *nicht rekursiv.* □

Im Beweis von Satz 9.9 haben wir eine Standardtechnik angewandt, um die Unentscheidbarkeit bestimmter Eigenschaften zu zeigen: die *Reduktion* eines Problems auf ein anderes. In diesem Fall haben wir das Halteproblem auf das Zugehörigkeitsproblem reduziert. Weil wir wissen, dass das Halteproblem nicht entscheidbar ist, kann somit auch das Zugehörigkeitsproblem nicht entscheidbar sein.

9.4 Äquivalente Formulierungen

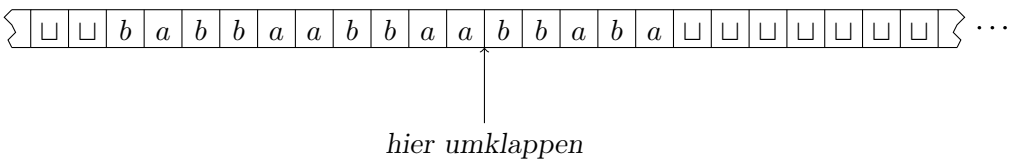
Zweiseitig Unbeschränkte Bänder

Eine andere Möglichkeit Definition 9.1 zu erweitern ist es, das Band sowohl nach links als auch nach rechts unbeschränkt zu definieren, wie in Abbildung 9.9 dargestellt. Auch diese Erweiterung erhöht die Ausdrucksfähigkeit des Turingmaschinenmodells nicht.

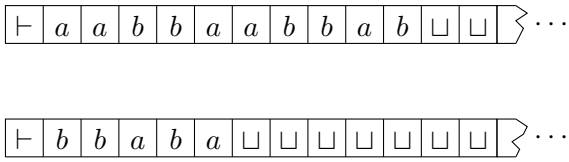
Satz 9.10. *Sei M eine einbändige DTM, dessen Band in beide Richtungen unbeschränkt ist. Dann existiert eine einbändige DTM M', sodass $\mathsf{L}(M) = \mathsf{L}(M')$.*

Beweis.[Skizze] Wir zeigen den Satz, indem wir M durch eine 2-bändige Turingmaschine simulieren. Die Behauptung folgt dann aus Satz 9.4.

In der Simulation wird das zweiseitig unendliche Band an einer beliebigen Stelle geknickt, wie die folgende Grafik verdeutlicht:



Wir erhalten die folgende Repräsentation des ursprünglichen Bandes, wobei das obere Band verwendet wird, um den Bandinhalt links von der Knickstelle darzustellen und das untere Band für den Bandinhalt rechts von der Knickstelle.



□

Zwei Keller

Ein *2-Kellerautomat* ist ein endlicher Automat, der zusätzlich noch zwei Keller (oder *Stapel*) zur Verfügung hat. Wir begnügen uns mit der genannten informellen Definition und überlassen es der Leserin 2-Kellerautomaten, sowie den Begriff der *Sprache* $\mathsf{L}(K)$ eines Kellerautomaten K formal einzuführen. Der folgende Satz zeigt, dass 2-Kellerautomaten und Turingmaschinen äquivalent sind.

Satz 9.11. *Sei K ein Kellerautomat, dann existiert eine TM M, sodass $\mathsf{L}(M) = \mathsf{L}(K)$. Umgekehrt gibt es zu jeder TM M' einen Kellerautomaten K', sodass $\mathsf{L}(K') = \mathsf{L}(M')$.*

Beweis.[Skizze] Die Simulation eines 2-Kellerautomaten durch eine Turingmaschine mit drei Bändern ist trivial: Das obere Band dient der Eingabe und die unteren beiden Bänder der Darstellung der Keller.

Andererseits kann jede TM durch einen 2-Kellerautomaten wie folgt simuliert werden: Der Bandinhalt links vom Schreib-/Lesekopf wird auf dem ersten Keller, der Bandinhalt rechts vom Schreib-/Lesekopf auf dem zweiten Keller dargestellt. Dabei ist nur darauf zu achten, dass die Symbole, die dem Schreib-/Lesekopf am nächsten sind, auf den beiden Stapeln am höchsten liegen. Dann kann das Lesen und Schreiben durch Kopieren von einem Stapel auf den anderen simuliert werden. □

Registermaschinen

Eine *k-Registermaschine* ist eine Maschine, die mit einem Leseband und zusätzlich k Registern, die natürliche Zahlen aufnehmen können ausgestattet ist. Jedes Register kann eine beliebige natürliche Zahl speichern. In jedem Schritt kann der Automat unabhängig jeden Registerinhalt um eins erhöhen und verkleinern. Außerdem erlauben die Instruktionen den Test der Register auf 0 und das Verschieben des Lesekopfs (in beide Richtungen) um eine Zelle.

Satz 9.12. *Sei R eine k-Registermaschine, dann existiert eine Turingmaschine M, sodass $\mathsf{L}(M) = \mathsf{L}(R)$. Umgekehrt gibt es zu jeder TM M' eine 2-Registermaschine R', sodass $\mathsf{L}(R') = \mathsf{L}(M')$.*

Beweis. Sei R eine k-Registermaschine. Diese kann am einfachsten durch eine $(k+1)$-bändige Turingmaschine M'' simuliert werden, indem jedem Register ein eigenes Arbeitsband entspricht. Als Bandalphabet für die zusätzlichen Bänder können wir $\{0, 1\}$ verwenden und den Inhalt der Register binär codieren. Dann ist es nicht schwer entsprechende Turingmaschinenprogramme zu schreiben, die das Inkrementieren und Dekrementieren von Registern codieren. Um abschließend den ersten Teil des Satzes zu beweisen, muss nur noch die Maschine M'' durch eine einbändige TM M simuliert werden.

Andererseits sei M' die Turingmaschine, die wir durch eine 2-Registermaschine simulieren wollen. Dazu wandeln wir zunächst M' in einen 2-Kellerautomaten um und zeigen, wie wir einen Stapel durch zwei Register simulieren können. Anschließend skizzieren wir, wie die erhaltene 4-Registermaschine durch eine 2-Registermaschine simuliert werden kann.

Wir zeigen die Simulierung eines Kellers durch zwei Register. Dazu können wir annehmen, dass das Kelleralphabet nur die Symbole 0 und 1 enthält: Sollte das Alphabet mehr Symbole enthalten, können wir diese leicht durch verschiedene Binärstrings codieren. Wir gehen also davon aus, dass der Kellerinhalt als Binärzahl dargestellt werden kann, dessen niedrigwertigste Ziffer ganz oben am Stapel liegt. Die Simulation speichert die Zahl im ersten Register und verwendet das zweite Register, um Stapeloperationen zu simulieren. Etwa um eine 0 oben auf den Stapel zu legen, muss der Stapelinhalt (als Zahl gelesen) verdoppelt werden. Dazu wird eine Schleife aufgerufen, die iterativ das erste Register dekrementiert und in jedem Schritt 2 zum Inhalt des zweiten Registers dazu zählt. In einer ähnlichen Weise wird simuliert, dass eine 1 auf den Stapel gelegt werden soll. Um das Entfernen des obersten Elements zu simulieren, müssen wir nur den Registerinhalt durch 2 dividieren und durch einen mod2 Test feststellen, ob der Stapel von einer 0 oder einer 1 gekrönt war. Damit haben wir die Simulation eines 2-Kellerautomaten durch eine 4-Registermaschine vollständig beschrieben.

Wir skizzieren die Simulation einer 4-Registermaschine R'' durch eine 2-Registermaschine R'. Angenommen die Register von R'' enthalten die Werte i, j, k und l. Dann schreiben wir die Zahl $2^i \cdot 3^j \cdot 5^k \cdot 7^l$ in das erste Register von R'. Das zweite Register dient zur Simulation der Rechenoperationen von R''. □

Aufzählmaschinen

Wir haben die rekursiv aufzählbaren Mengen als jene Sprachen definiert, die durch eine Turingmaschine akzeptiert werden. Der Ausdruck *rekursiv aufzählbar* kommt allerdings von einem alternativen Maschinenmodell, den *Aufzählmaschinen*.

Wir verstehen unter einer Aufzählmaschine einen Automaten mit einer endlichen Kontrolle und zwei Bändern. Das erste Band ist ein Schreib-/Leseband, das auch *Arbeitsband* genannt wird. Das zweite Band ist das *Ausgabeband*, auf das nur geschrieben werden kann. Eine Aufzählmaschine startet immer mit dem leeren Arbeitsband (erhält somit also keine Eingabe) und besitzt keinen akzeptierenden oder verwerfenden Zustand. Wie in einer Turingmaschine werden die Rechenschritte durch eine Übergangsfunktion codiert. Von Zeit zu Zeit gelangt die Aufzählmaschine in einen speziellen Zustand, den *Aufzählungszustand*. Wenn das passiert, wird das aktuell am Ausgabeband stehende Wort als *aufgezählt* bezeichnet.

Das Wort wird gelöscht und die Maschine generiert das nächste aufzuzählende Wort. Eine Aufzählmaschine terminiert nicht. Es ist möglich, dass eine Aufzählmaschine E niemals in den Aufzählungszustand gelangt, dann gilt $\mathsf{L}(E) = \varnothing$. Es gibt auch Aufzählungsmaschinen, die unendlich viele Wörter aufzählen. Dabei ist es erlaubt das selbe Wort mehrmals aufzuzählen.

Satz 9.13. *Die Familie der Mengen, die von einer Aufzählungsmaschine aufgezählt werden können, entspricht genau der Familie von rekursiv aufzählbaren Mengen. In anderen Worten, eine Menge ist* $\mathsf{L}(E)$ *für eine Aufzählungsmaschine* E *genau dann, wenn diese Menge* $\mathsf{L}(M)$ *einer Turingmaschine* M *ist.*

Beweis. Der vollständige Beweis kann in [4] nachgelesen werden. □

Weitere äquivalente Formalismen

Berechenbarkeit kann nicht nur durch Maschinenmodelle beschrieben werden. Ebenso sind mathematische Konzepte wie

- Partielle rekursive Funktionen
- Lambda Kalkül
- Kombinatorische Logik
- (Term-)Ersetzungssysteme

äquivalent zu Turingmaschinen. Insbesondere sind auch bisherige Modelle für den Quantencomputer in Bezug auf die Berechenbarkeit nicht mächtiger als die Turingmaschine; allerdings *effizienter* für einige bestimmte Probleme (Suchen in unsortierten Listen, Faktorisieren von Zahlen). Zudem können Quantencomputer echte Zufallszahlen generieren (im Gegensatz zu *Pseudozufallszahlen* wie bei einem herkömmlichen Computer), was für Verschlüsselungsalgorithmen wichtig ist.

9.5 Aufgaben

Aufgabe 9.1. Welche Konfigurationen erreicht die TM M aus Beispiel 9.1 beginnend mit der Startkonfiguration

1. $(s, \vdash 001 \sqcup^\infty, 0)$
2. $(s, \vdash 111 \sqcup^\infty, 0)$

Bestimmen Sie $\mathsf{L}(M)$. Was steht am Band von M, wenn M bei Eingabe x in den akzeptierenden Zustand wechselt?

Aufgabe 9.2. Sei die TM $M = (\{s, q, t, r\}, \{0, 1\}, \{\vdash, \sqcup, 0, 1\}, \vdash, \sqcup, \delta, s, t, r)$ mit δ gegeben durch

	$\vdash$	0	1	$\sqcup$
s	$(s, \vdash, \mathrm{R})$	$(q, 0, \mathrm{R})$	$(t, 1, \mathrm{R})$	$(r, \sqcup, \mathrm{R})$
q	$\cdot$	$(s, 0, \mathrm{L})$	$(r, 1, \mathrm{R})$	$(r, \sqcup, \mathrm{R})$

Zeichnen Sie das Zustandsübergangsdiagramm für M. Welche Wörter akzeptiert bzw. verwirft M? Auf welchen Wörtern hält M, auf welchen nicht? Ist M total? Bestimmen Sie $\mathsf{L}(M)$. Ist $\mathsf{L}(M)$ rekursiv bzw. rekursiv aufzählbar?

Aufgabe 9.3. Ändern/Erweitern Sie die TM M aus Beispiel 9.1, sodass der Übertrag bei der binären Addition berücksichtigt wird.

Hinweis: Die neue TM muss zuerst Platz für einen möglichen Übertrag schaffen.

Aufgabe 9.4. Ändern Sie die TM M aus Beispiel 9.3, sodass sie alle Palindrome (auch jene ungerader Länge) über dem Alphabet $\Sigma = \{0, 1\}$ akzeptiert.

Aufgabe 9.5. Geben Sie eine formale Definition für eine k-bändige deterministische Turingmaschine an. Wie sieht die Startkonfiguration aus? Wie ist die akzeptierte Sprache definiert?

Aufgabe 9.6. Beweisen Sie Satz 9.4.

Aufgabe 9.7. Geben Sie eine TM an, welche die Sprache

$$L = \{0^n 1^n \mid n \geqslant 0\}$$

akzeptiert. Ist diese Turingmaschine total?

Aufgabe 9.8. Geben Sie eine TM an, welche die Sprache

$$L = \{0^n 1^n 2^n \mid n \geqslant 0\}$$

akzeptiert. Ist diese Turingmaschine total?

Aufgabe 9.9. Geben Sie eine einbändige TM M über dem Eingabealphabet $\{0\}$ an, sodass sich der Schreib-/Lesekopf in der Mitte des Eingabewortes befindet, wenn M akzeptiert, d.h. $(s, \vdash x \sqcup^\infty, 0) \xrightarrow[M]{*} (t, \vdash x \sqcup^\infty, \frac{\ell(x)}{2} + 1)$.

Hinweis: Markieren Sie das erste sowie letzte Zeichen der Eingabe und schieben Sie die Markierungen schrittweise Richtung Mitte.

Aufgabe 9.10. Geben Sie eine mehrbändige TM M mit der Eigenschaft aus Aufgabe 9.9 an. Sie können annehmen, dass die Schreib-/Leseköpfe der TM auch stehen bleiben können.

Aufgabe 9.11. Beweisen Sie, dass jede reguläre Sprache rekursiv ist.
Hinweis: Konstruieren Sie eine totale Turingmaschine für einen gegebenen Automaten (z.B. einen DEA).

Aufgabe 9.12. Was bedeuten die Begriffe *rekursiv*, *rekursiv aufzählbar*, *entscheidbar*, *semi-entscheidbar*? Welche Zusammenhänge kennen Sie zwischen diesen Begriffen? Angenommen $L \subseteq \Sigma^*$ sei eine formale Sprache und sie wollen wissen, ob ein Wort $w \in \Sigma^*$ in L enthalten ist. Macht es für diese Frage einen Unterschied, ob L rekursiv bzw. rekursiv aufzählbar ist?

Aufgabe 9.13. Welche der folgenden Sprachen ist (i) rekursiv, (ii) rekursiv aufzählbar, (iii) nicht rekursiv aufzählbar?

1. $L_1 = \{\mathsf{0}^n \mid n \text{ ist eine Primzahl}\}$
2. $L_2 = \{\ulcorner M \urcorner \# \ulcorner x \urcorner \mid M \text{ ist eine TM und } x \text{ ein Wort über deren Eingabealphabet}\}$
3. $L_3 = \{\ulcorner M \urcorner \mid M \text{ ist eine TM mit } \mathsf{L}(M) \neq \varnothing\}$
4. $L_4 = \{x \in \{\mathsf{0}, \mathsf{1}\}^* \mid x \text{ enthält den Substring } \mathsf{0011}\}$

Hinweis: Keine formalen Beweise notwendig. Informelle Begründung reicht aus.

Aufgabe 9.14. Sei NTM $N = (\{s, q, r, t\}, \{\mathsf{0}, \mathsf{1}\}, \{\vdash, \sqcup, \mathsf{0}, \mathsf{1}\}, \vdash, \sqcup, \delta, s, t, r)$ mit δ gegeben durch

	$\vdash$	$\mathsf{0}$	$\mathsf{1}$	$\sqcup$
s	$\{(s, \vdash, \mathrm{R})\}$	$\{(s, \mathsf{0}, \mathrm{R}), (q, \mathsf{0}, \mathrm{R})\}$	$\{(s, \mathsf{1}, \mathrm{R})\}$	$\{(r, \sqcup, \mathrm{L})\}$
q	$\cdot$	$\{(s, \mathsf{0}, \mathrm{L}), (r, \mathsf{0}, \mathrm{L})\}$	$\{(t, \mathsf{1}, \mathrm{L})\}$	$\{(r, \sqcup, \mathrm{L})\}$

Skizzieren Sie den Berechnungsbaum für die Wörter ϵ und $\mathsf{0011}$. Was ist der Grad des Nichtdeterminismus von N? Wie kann man die einzelnen Konfigurationen im Berechnungsbaum adressieren?

Aufgabe 9.15. Was ist die *universelle Turingmaschine* U? Wofür wird sie verwendet? Welche Sprache akzeptiert U?

Aufgabe 9.16. Codieren Sie die TM M aus Beispiel 9.1 über dem Alphabet $\{\mathsf{0}, \mathsf{1}\}$.

Aufgabe 9.17. Beweisen Sie mittels Diagonalisierung, dass MP nicht rekursiv ist. Skizzieren Sie Ihre Diagonalisierungsmaschine.

Aufgabe 9.18. Ist die Menge $\sim\mathsf{MP}$ rekursiv / rekursiv aufzählbar / nicht rekursiv aufzählbar? Begründen Sie Ihre Antwort.

Aufgabe 9.19. Welche der folgenden PCPs haben eine Lösung?

1. $\mathsf{111}, \mathsf{1}, \mathsf{010}$ und $\mathsf{1}, \mathsf{11}, \mathsf{01}$
2. $\mathsf{111}, \mathsf{1}, \mathsf{010}$ und $\mathsf{1}, \mathsf{0}, \mathsf{01}$

Aufgabe 9.20. Sei $i = i_1 \dots i_m$ eine Indexliste und p eine Instanz von PCP. Sei $P(i, p)$ die Eigenschaft, i ist eine Lösung von p. Zeigen Sie, dass die Eigenschaft P entscheidbar ist.

Hinweis: Skizzieren Sie eine TM M mit der Eigenschaft „i ist Lösung von p genau dann, wenn $\ulcorner i \urcorner \# \ulcorner p \urcorner \in \mathsf{L}(M)$“.

Aufgabe 9.21. Zeigen Sie, dass das Finden einer Lösung für ein PCP semi-entscheidbar ist.

Hinweis: Konstruieren Sie eine TM, die mögliche Lösungen gemäß einer Ordnung durchläuft. Auf was muss geachtet werden?

Aufgabe 9.22. Finden Sie eine TM M, sodass $\ulcorner M \urcorner$ minimal (in der graduiert-lexikographischen Ordnung) ist.

Hinweis: Beachten Sie die Bedingungen in Definition 9.1.

Aufgabe 9.23. Zeigen Sie, dass die Sprache der Palindrome über dem Alphabet $\{\mathsf{0}, \mathsf{1}\}$ rekursiv ist. Verwenden Sie eine Reduktion auf die TM aus Beispiel 9.3.

Hinweis: Verwenden Sie Aufgabe 3.10.

10

Komplexitätstheorie

Alle in diesem Kapitel verwendeten Alphabete sind endlich, alle Turingmaschinen total und alle Eigenschaften/Probleme entscheidbar.

10.1 Einführung in die Komplexitätstheorie

Komplexitätsabschätzungen haben wir bereits kennen gelernt, z.B. in Satz 5.6, dass die Nachfolgersuche in polynomiell vielen Schritten (in der Anzahl der Ecken und Kanten) alle von einer Startmenge S aus erreichbaren Ecken in einem gerichteten Multigraphen G markiert. Die Abschätzungen, die wir jetzt anstellen werden, sind etwas verschieden. Hat uns vorher die Komplexität eines bestimmten *Algorithmus* interessiert, wird nun ein bestimmtes *Problem* in den Vordergrund rücken. Die verwendeten Algorithmen rücken dabei in den Hintergrund.

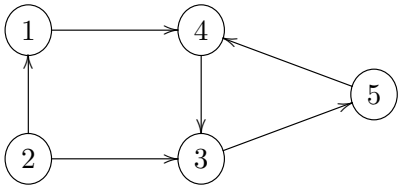
Die *Komplexitätstheorie* analysiert Algorithmen und Probleme hinsichtlich der von ihnen benötigten Ressourcen wie etwa Rechenzeit oder Speicherplatz. Hierbei bezeichnet der Ausdruck *Problem* eine allgemeine, mit Ja oder Nein zu beantwortende Frage; ein Verfahren zur Beantwortung dieser Frage heißt *Algorithmus*. Beide Begrifflichkeiten haben wir bereits verwendet. Zu jedem Problem gibt es verschiedene Algorithmen unterschiedlichster Komplexität. Unter der Komplexität eines Algorithmus verstehen wir die notwendigerweise aufgewandten Ressourcen (in Relation zur Eingabe), um den Algorithmus auszuführen. Im Besonderen sind wir daran interessiert, *wie lange* ein Algorithmus braucht, um ein Problem zu lösen (*Laufzeitkomplexität*) oder *wie viel Platz* benötigt wird (*Speicherplatzkomplexität*). Die Komplexität eines Problems ist die Komplexität des effizientesten Algorithmus, um das Problem zu lösen. Wir betrachten beispielhaft zwei Probleme.

Erreichbarkeit:

Problem. *Gegeben ein gerichteter Graph G und Ecken $s, t \in E$, wobei E die Eckenmenge von G bezeichnet. Gibt es einen Weg von s nach t? Dieses Problem nennt man* Maze *bzw.* Erreichbarkeit.

Zur Verdeutlichung betrachten wir ein einfaches Beispiel.

Beispiel 10.1. Im gerichteten Graphen



gibt es einen Weg von Ecke 1 zu Ecke 5, aber keinen Weg von Ecke 5 zu Ecke 1.

Ein naiver Algorithmus könnte das Problem Maze wie folgt Lösen:

– Überprüfe für jedes Tupel aus $K^{\#(E)-1}$, ob es ein Weg in G mit Startecke s ist, der t als Zwischen- oder Endecke hat.

Sei $n = \#(E) + \#(K)$. Für ein Tupel kann dieser Test in Zeit $\mathrm{O}(n^2)$ durchgeführt werden. Weil es $\mathrm{O}(n^n)$ verschiedene Tupel gibt, beträgt die Zeitkomplexität dieses Algorithmus $\mathrm{O}(n^2 \cdot n^n) = \mathrm{O}(n^{n+2})$. Die Platzkomplexität hingegen beträgt $\mathrm{O}(n)$, weil man sich immer nur ein Tupel merken muss.

Das Problem Maze kann auch mittels Nachfolgersuche (Satz 5.6) gelöst werden:

1. Setze $S = \{s\}$ und markiere s.
2. Solange S nicht leer ist, wiederhole:
   – Wähle eine Ecke $e \in S$ und entferne sie aus S.
   – Bestimme alle unmarkierten unmittelbaren Nachfolger von e, markiere sie und gebe sie zu S hinzu.
3. Antwort "ja" wenn t markiert, sonst "nein".

Sei $n = \#(E) + \#(K)$. Dann folgt aus Satz 5.6, dass die Nachfolgersuche in Zeit $\mathrm{O}(n^2)$ läuft. Der Platzbedarf des Algorithmus ist $\mathrm{O}(n)$.

Man kann zeigen, dass die Nachfolgersuche von der Zeitkomplexität her ein *optimaler* Algorithmus für das Erreichbarkeitsproblem ist, d.h. die Zeitkomplexität ist notwendigerweise quadratisch. Bezüglich der Platzkomplexität gibt es Algorithmen, die mit $\mathrm{O}(\log^2 n)$ Platz auskommen (siehe [8]), dann aber eine exponentielle Zeitkomplexität aufweisen.

Hamiltonscher Kreis:

Problem. *Gegeben ein ungerichteter Graph G. Gibt es einen Zykel, der jede Ecke genau einmal enthält (mit Ausnahme der Start-/Endecke)? Solch ein Zykel wird* Hamiltonscher Kreis *genannt.*

Beispiel 10.2. In Beispiel 5.11 (Seite 41) ist $(1, 2, 6, 7)$ ein Hamiltonscher Kreis.

Ein (naiver) Algorithmus könnte für einen ungerichteten Graphen mit Eckenmenge E und Kantenmenge K wie folgt verfahren:

- Überprüfe für jedes Tupel aus $K^{\#(E)}$, ob es ein Hamiltonscher Kreis ist.

Sei $n := \#(E)+\#(K)$. Dann kann für ein gegebenes Tupel in Zeit $\mathrm{O}(n^2)$ überprüft werden, ob es ein Hamiltonscher Kreis ist. Allerdings gibt es $\mathrm{O}(n^n)$ viele dieser Tupel und somit ist die Zeitkomplexität für den skizzierten Algorithmus $\mathrm{O}(n^{n+2})$. Die Speicherplatzkomplexität ist $\mathrm{O}(n)$, weil man sich immer nur das aktuell betrachtete Tupel merken muss. Die Zeitkomplexitätsschranke lässt sich verbessern, z.B. sind Algorithmen mit *exponentieller* Laufzeit bekannt. Das heißt, das Problem ist nur für recht kleine Probleminstanzen mit verträglichem Zeitaufwand lösbar.

Bisher ist nicht bekannt, ob es einen Algorithmus gibt, der das Hamiltonsche Kreis Problem in *polynomieller* Zeit lösen kann.

Als letztes Problem betrachten wir das Problem festzustellen, ob eine Spielerin das Spiel *verallgemeinerte Länderkunde* immer gewinnen kann. Hierbei ist das Spiel *verallgemeinerte Länderkunde* wie folgt definiert. Gegeben seien ein gerichteter Graph G und ein ausgewählter Startknoten s. Das Spiel *verallgemeinerte Länderkunde* wird von zwei Spielern gespielt. Spielerin I beginnt in der Ecke (dem "Land") s und wählt eine in G erreichbare Ecke ("Land"), das markiert wird. Dann zieht Spieler II und markiert das erreichte Land wieder und so weiter. Derjenige Spieler, der kein (unmarkiertes) Land mehr erreichen kann, verliert.

Problem. *Seien ein gerichteter Graph und ein ausgewählter Startknoten s gegeben. Das* verallgemeinerte Länderkundeproblem *(*generalised geography problem*, kurz GEO) ist das Problem, ob Spielerin I das Spiel immer gewinnen kann, wenn sie im Land s beginnt. Es wird also nach einer Gewinnstrategie für Spielerin I gesucht.*

Die folgende rekursive Entscheidungsprozedur A stellt fest, ob Spielerin I eine Gewinnstrategie hat. Dabei bezeichnet G den Graphen, E, K Eckenmenge und Kantenmenge und b das aktuell besuchte Land.

1. Wenn der Ausgangsgrad von b, 0 ist, dann verliert Spielerin I immer, deshalb verwerfen wir die Eingabe.

2. Eliminiere Ecke b und alle wegführenden Kanten. Der erhaltene Graph wird mit G' bezeichnet.

3. Für jede der Ecken b' aus der Menge der unmittelbaren Nachfolger von b (in G), rufe A rekursiv mit (G', b') auf.

4. Wenn alle Aufrufe akzeptieren, dann hat Spieler II eine Gewinnstrategie, also verwerfe. Andererseits hat Spieler II keine Gewinnstrategie, dafür Spielerin I, also akzeptiere.

Wir schätzen nur den Speicherplatzbedarf des Algorithmus ab. Der einzige Platz den A benötigt, ist der Platz für die rekursiven Aufrufe. In jedem rekursiven Aufruf wird eine Ecke gespeichert. Die Anzahl der Ecken ist durch $\#(E)$ beschränkt, also ist der Platzbedarf linear in Größe des Graphen G.

Die Komplexität eines Problems kann vom Berechnungsmodell abhängen. Für die Komplexitätsanalyse wählen wir ein möglichst einfaches (und formales) Berechnungsmodell wie etwa Turingmaschinen. Weil man zeigen kann, dass ein konventioneller Computer (in polynomieller Zeit) auf einer Turingmaschine simuliert werden kann, sind unsere Analysen auch für die Praxis relevant.

10.2 Laufzeitkomplexität

Die folgende Definition gilt für deterministische und nichtdeterministische Turingmaschinen.

Definition 10.1 (Laufzeitkomplexität)**.** Sei M eine totale Turingmaschine. Die *Laufzeit* oder *Laufzeitkomplexität* von M ist eine Abbildung $T\colon \mathbb{N} \to \mathbb{N}$, wobei $T(n)$ die maximale Anzahl von Schritten von M auf allen Eingaben der Länge n bezeichnet, also

$$T(n) := \max\{m \mid M \text{ hält bei Eingabe } x \text{ nach } m \text{ Schritten und } \ell(x) = n\}$$

Wir sagen auch, dass T *die Laufzeit von* M ist und, dass M *eine* T*-Zeit Turingmaschine* ist.

Beispiel 10.3. Für die TM M aus Beispiel 9.1 ergibt sich z.B. $T(0) = 3$ und $T(1) = 5$. Weiters ist $T(n) \in \mathrm{O}(n)$.

Als nächstes führen wir *Komplexitätsklassen* ein. In dieser Vorlesung beschränken wir uns auf die Komplexität von *entscheidbaren Problemen.*

Definition 10.2. Sei $T\colon \mathbb{N} \to \mathbb{N}$ eine Abbildung. Die deterministische Zeitkomplexitätsklasse $\mathsf{DTIME}(T)$ ist wie folgt definiert:

$$\mathsf{DTIME}(T) := \{\mathsf{L}(M) \mid M \text{ ist eine mehrbändige DTM mit Laufzeit } \mathrm{O}(T)\}$$

Die nichtdeterministische Zeitkomplexitätsklasse $\mathsf{NTIME}(T)$ definiert man analog:

$$\mathsf{NTIME}(T) := \{\mathsf{L}(M) \mid M \text{ ist eine mehrbändige NTM mit Laufzeit } \mathrm{O}(T)\}$$

Die Klasse $\mathsf{DTIME}(T)$ enthält also alle Sprachen, deren Zugehörigkeitstest von einer mehrbändigen DTM in Zeit $\mathrm{O}(T)$ entschieden werden kann.

Wir können das Problem Maze auch als Sprache betrachten, wenn wir ein geeignetes Alphabet zur Kodierung von Graphen voraussetzen.[1]

$$\text{Maze} = \{(G, s, t) \mid G \text{ ist ein gerichteter Graph mit einem Weg von } s \text{ nach } t\}$$

Auch das Problem des Hamiltonschen Kreises kann als Sprache definiert werden:

$$\text{HK} = \{(G, k) \mid G \text{ ist ein ungerichteter Graph mit Hamiltonschem Kreis } k\}$$

Beispiel 10.4. Wir haben gesehen, dass Maze mit einer deterministischen Turingmaschine in Laufzeit $\mathrm{O}(n^2)$ entschieden werden kann. Also gilt Maze $\in \mathsf{DTIME}(n^2)$.

Beispiel 10.5. Wir haben gesehen, dass HK $\in \mathsf{DTIME}(n^{n+2})$. Allerdings ist HK $\in \mathsf{NTIME}(n^2)$, weil eine NTM einen Hamiltonschen Kreis *raten* und diesen in $\mathrm{O}(n^2)$ Zeit verifizieren kann.

[1] In diesem Kapitel verwenden wir die Begriffe *Sprache* und *Problem* synonym, da man jedes algorithmische Problem geeignet als formale Sprache darstellen kann und umgekehrt (siehe Definition 9.9).

10.2.1 Die Klassen P und NP

Wir interessieren uns für das asymptotische Wachstum von Komplexitätsschranken. Dabei sind Laufzeitkomplexitätsfunktionen, die polynomiell sind, solchen die exponentiell wachsen (im Normalfall) deutlich vorzuziehen.

Im Besonderen interessieren wir uns für diejenigen Probleme, die durch einen Algorithmus, der in polynomieller Zeit läuft, berechnet werden können. Diese Probleme werden in der Komplexitätsklasse P zusammengefasst. Hier steht das P für *P*olynomielle Zeit. Die Klasse P umfasst jene Probleme, für welche eine Lösung in polynomieller Zeit *gefunden* werden kann.

Die Möglichkeit von nichtdeterministischen Algorithmen erklärt das Interesse an einer zweiten Komplexitätsklasse, der Klasse NP. Hier steht NP für *N*ichtdeterministische *P*olynomielle Zeit. Die Klasse NP umfasst jene Probleme, für welche eine Lösung in polynomieller Zeit *verifiziert* werden kann. Die Komplexität der *Suche* einer geeigneten Lösung bleibt hier jedoch verborgen.

In der Folge definieren wir die Klassen P und NP formal: Die Komplexitätsklasse P umfasst alle Sprachen, deren Zugehörigkeitstest in polynomieller Zeit durch eine DTM entschieden werden kann. Analog können wir NP definieren als die Klasse von Sprachen, deren Zugehörigkeitstest in polynomieller Zeit durch eine NTM entschieden werden kann.

Definition 10.3.

$$\mathsf{P} := \bigcup_{k \geq 0} \mathsf{DTIME}(n^k)$$
$$\mathsf{NP} := \bigcup_{k \geq 0} \mathsf{NTIME}(n^k)$$

Wir geben zwei unterschiedliche, aber äquivalente Definitionen der Klasse NP. Zunächst formalisieren wir die eingangs erwähnte Intuition. Dies Definition erleichtert auch das Feststellen, ob eine bestimmte Sprache in NP ist oder nicht.

Definition 10.4. Ein *Verifikator* einer Sprache L ist ein Algorithmus V, sodass

$$L = \{x \mid \text{es existiert ein String } c\text{, sodass } V \text{ akzeptiert } (x, c)\}\,.$$

Ein *polytime Verifikator* ist ein Verifikator mit Laufzeit $\mathrm{O}(n^k)$ (k beliebig), wobei $n = \ell(x)$. Das Wort c wird auch als *Zertifikat* bezeichnet.

Die Klasse NP' ist definiert als die Klasse der Sprachen L, die einen polytime Verifikator haben.

Beispiel 10.6. Es gilt TSP $\in \mathsf{NP}'$. Das ist am leichtesten dadurch einzusehen, indem die optimale Tour als Zertifikat c betrachtet wird. Gegeben eine Tour c ist es leicht möglich mit Hilfe eines deterministischen Algorithmus nachzuprüfen, dass diese Tour wirklich eine Bewertung kleiner gleich B hat.

Satz 10.1. *Es gilt* $\mathsf{NP} = \mathsf{NP}'$*, das heißt, die Klasse der Sprachen, die einen polytime Verifikator haben, ist gleich der Klasse von Sprachen, die durch eine NTM entschieden werden.*

Beweis. Sei $L \in \mathsf{NP}'$*. Dann existiert eine nichtdeterministische TM,* N*, sodass* $L = \mathsf{L}(N)$*. Wir definieren eine polytime Verifikator* V *mit Eingabe* (x, c)*, wobei* x *und* c *Wörter sind.*

1. *Simuliere auf der Eingabe x die NTM N, wobei die Symbole in c als Auswahlsequenz verwendet werden, um die Nichtdeterminismuspunkte aufzulösen. (Vergleiche dazu mit dem Beweis von Satz 9.5.)*

2. *Wenn N akzeptiert, akzeptiert V, ansonsten verwerfe.*

Andererseits sei $L \in \mathsf{NP}$. Das heißt, es existiert ein polytime Verifikator V für L. Dann konstruieren wir eine nichtdeterministische Maschine N aus V mit Eingabe x. Wir nehmen an V läuft in Zeit n^k.

1. *Wähle (nichtdeterministisch) einen String c der Länge n^k.*

2. *Simuliere V auf (x, c).*

3. *Wenn V akzeptiert, dann akzeptiere, sonst verwerfe.*

□

Mit dem Lemma gilt also TSP $\in \mathsf{NP}$, was natürlich auch direkt über TMs leicht zu argumentieren ist.

Beispiel 10.7. Es gilt Maze $\in \mathsf{P}$. Wegen $\mathsf{P} \subseteq \mathsf{NP}$ ist auch Maze $\in \mathsf{NP}$.

Beispiel 10.8. Es gilt HK $\in \mathsf{NP}$. Derzeit ist unbekannt, ob HK $\in \mathsf{P}$.

Offensichtlich ist $\mathsf{P} \subseteq \mathsf{NP}$. Die andere Inklusion ist ein offenes Problem in der Informatik.

> Die Direktoren von CMI [Clay Mathematics Institute] haben 7 Millionen Dollar Preisgeld für die sogenannten *Millennium Probleme* ausgeschrieben. Eines der Millennium Probleme ist die Frage nach $\mathsf{P} = \mathsf{NP}$.

10.2.2 Polynomielle Reduktion

Am Ende von Kapitel 9.3.2 haben wir den Begriff der *Reduktion* informell eingeführt. Nun werden wir diesen Begriff, genauer den Begriff der *polynomiellen Reduktion*, formal einführen.

Definition 10.5. Seien Σ und Δ Alphabete. Eine Abbildung $R\colon \Sigma^* \to \Delta^*$ heißt in *polynomieller Zeit* berechenbar, wenn es eine k-bändige totale DTM M mit Eingabealphabet Σ gibt, sodass die folgenden Bedingungen erfüllt sind:

1. M läuft in polynomieller Zeit, das heißt, es existiert ein $k \in \mathbb{N}$, sodass M eine $\mathrm{O}(n^k)$-Zeit Turingmaschine ist.

2. Bei Eingabe $x \in \Sigma^*$ steht $R(x) \in \Delta^*$ am k-ten Band, wenn M hält.

Definition 10.6 (Reduktion in polynomieller Zeit)**.** Seien $A \subseteq \Sigma^*$ und $B \subseteq \Delta^*$ formale Sprachen. Dann ist A in *polynomieller Zeit reduzierbar* auf B ($A \leqslant^{\mathrm{p}} B$), wenn eine in polynomieller Zeit berechenbare Abbildung $R\colon \Sigma^* \to \Delta^*$ existiert, sodass für alle $x \in \Sigma^*$

$$x \in A \;\Leftrightarrow\; R(x) \in B\,.$$

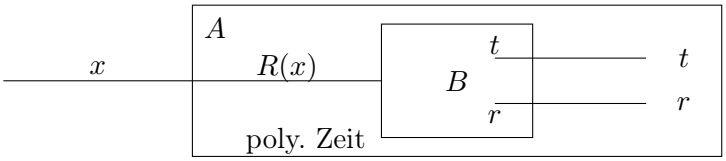


Abbildung 10.1: Polynomielle Reduktion: $x \in A \Leftrightarrow R(x) \in B$

Wir verwenden Reduktionen, wenn wir zeigen wollen, dass ein Problem A nicht schwieriger zu lösen ist als ein Problem B. Sei etwa für das Problem B schon eine Turingmaschine bekannt, die B entscheidet. Gelte nun $A \leqslant^{\mathrm{p}} B$, wobei wir die Reduktion R verwendet haben. Dann können wir mit Hilfe von R und der Turingmaschine für B das Problem A entscheiden.

In Abbildung 10.1 wird der Zusammenhang anschaulich beschrieben. Da es sich bei R um eine in polynomieller Zeit berechenbare Funktion handelt, wissen wir auch, dass sich das Laufzeitverhalten von der Turingmaschine für A nur um einen polynomiellen Faktor von der Turingmaschine für B unterscheidet.

Beispiel 10.9. Seien

$$A = \{x \in \{a,b\}^* \mid \ell(x) \text{ ist gerade}\} \text{ und}$$
$$B = \{x \in \{0,1\}^* \mid x \text{ ist ein Palindrom gerader Länge}\}\,.$$

Dann gilt $A \leqslant^{\mathrm{p}} B$.

Satz 10.2. *Seien A und B formale Sprachen mit $A \leqslant^p B$. Dann gilt*

- *Wenn $B \in \mathsf{P}$, dann $A \in \mathsf{P}$.*
- *Wenn $B \in \mathsf{NP}$, dann $A \in \mathsf{NP}$.*

Beweis. Direkte Konsequenz aus Definition 10.6. □

Definition 10.7 ($\mathbb{C}$-hart, $\mathbb{C}$-vollständig)**.** Sei $\mathbb{C}$ eine beliebige Komplexitätsklasse (z.B. P oder NP). Eine formale Sprache B heißt

- $\mathbb{C}$*-hart (bezüglich $\leqslant^p$)*, wenn $A \leqslant^{\mathrm{p}} B$ für alle Sprachen $A \in \mathbb{C}$ und
- $\mathbb{C}$*-vollständig (bezüglich $\leqslant^p$)*, wenn $B \in \mathbb{C}$ und B zudem $\mathbb{C}$-hart ist.

Laut Definition 10.7 ist jedes Problem in P auch P-vollständig (bezüglich $\leqslant^{\mathrm{p}}$), weil man das Problem in der Reduktion lösen kann. Die folgenden Resultate geben wir ohne Beweis.

Satz 10.3. *Das Problem HK ist NP-vollständig.* □

Als einfache Konsequenz von Satz 10.3 sehen wir, dass $\mathsf{P} = \mathsf{NP}$, wenn $\mathrm{HK} \in \mathsf{P}$. Somit könnten Sie mit einem deterministischen Algorithmus für HK, der in polynomieller Zeit läuft eine Million Dollar verdienen.

Satz 10.4. *Das Erfüllbarkeitsproblem für aussagenlogische Formeln (SAT) ist NP-vollständig (siehe [7]).* □

10.3 Speicherplatzkomplexität

Analog zur Laufzeitkomplexität führen wir die *Speicherplatzkomplexität* formal ein. Weil wir die Eingabe nicht als Speicherplatz betrachten, nehmen wir an, dass die Turingmaschine auf das Eingabeband nur lesend zugreift, auf den Arbeitsbändern aber sowohl lesen als auch schreiben kann.

Definition 10.8 (Speicherplatzkomplexität)**.** Sei M eine totale k-bändige Turingmaschine. Der *Speicherplatz* oder die *Speicherplatzkomplexität* von M ist die Abbildung $S\colon \mathbb{N} \to \mathbb{N}$, wobei $S(n)$ die maximale Anzahl von Bandfeldern bezeichnet, die M auf allen Eingaben der Länge n liest, wobei aber nur die Zeichen auf den Arbeitsbändern betrachtet werden. Wir sagen auch, dass M eine S-Platz Turingmaschine ist.

Definition 10.9. Sei $S\colon \mathbb{N} \to \mathbb{N}$ eine Abbildung. Die deterministischen und nichtdeterministischen Platzkomplexitätsklassen sind wie folgt definiert:

$$\begin{aligned}
\mathsf{DSPACE}(S) &:= \{\mathsf{L}(M) \mid M \text{ ist eine mehrbändige DTM mit Speicherplatz } \mathrm{O}(S)\} \\
\mathsf{NSPACE}(S) &:= \{\mathsf{L}(M) \mid M \text{ ist eine mehrbändige NTM mit Speicherplatz } \mathrm{O}(S)\}
\end{aligned}$$

Wir definieren einige gängige Platzkomplexitätsklassen.

Definition 10.10.

$$\begin{aligned}
\mathsf{LOGSPACE} &:= \mathsf{DSPACE}(\log n) & \mathsf{NLOGSPACE} &:= \mathsf{NSPACE}(\log n) \\
\mathsf{PSPACE} &:= \bigcup_{k \geqslant 1} \mathsf{DSPACE}(n^k) & \mathsf{NPSPACE} &:= \bigcup_{k \geqslant 1} \mathsf{NSPACE}(n^k)
\end{aligned}$$

Abschließend wollen wir noch einen Überblick über die Zusammenhänge der betrachteten Komplexitätsklassen geben:

$$\mathsf{LOGSPACE} \subseteq \mathsf{NLOGSPACE} \subseteq \mathsf{P} \subseteq \mathsf{NP} \subseteq \mathsf{PSPACE} = \mathsf{NPSPACE}$$

Wir sehen also unter anderem, dass die polynomielle Platzkomplexitätsklasse unter Nichtdeterminismus abgeschlossen ist (d.h. $\mathsf{PSPACE} = \mathsf{NPSPACE}$). Wie schon berichtet, ist die selbe Frage für die polynomiellen Zeitkomplexitätsklassen P und NP ein offenes Problem. Zudem weiß man, dass $\mathsf{LOGSPACE} \neq \mathsf{PSPACE}$. Hingegen ist unbekannt welche (und wie viele) der obigen Inklusionen strikt sind [8].

10.3.1 Logarithmisch Platzbeschränkte Reduktion

In Sektion 10.2.2 haben wir Reduktionen betrachtet, die in polynomieller Zeit berechnet werden können. Für (kleine Platz)komplexitätsklassen, ist diese Definition mitunter zu grob. Deshalb definieren wir eine feinere Reduktionsrelation.

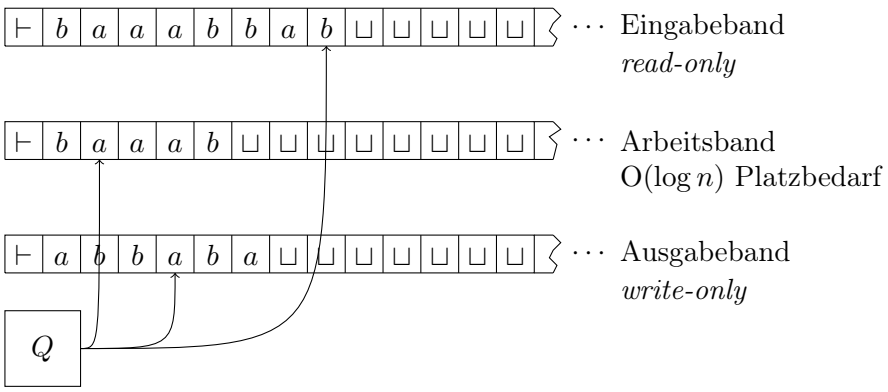
Definition 10.11. Ein *logarithmischer Umwandler* T ist ein 10-Tupel

$$T = (Q, \Sigma, \Gamma, \Delta, \vdash, \sqcup, \delta, s, t, r)\,,$$

sodass Δ eine endliche Menge von *Ausgabesymbolen* ist und Q, Σ, Γ, $\vdash$, $\sqcup$, δ, s, t, r wie bei einer totalen dreibändigen DTM definiert sind.

Zusätzlich gilt, dass das erste Band nur gelesen werden darf, das dritte Band nur beschrieben werden darf und am zweiten Band bei jeder Eingabe der Länge n maximal $\log n$ Zeichen gelesen werden dürfen.

Schematisch kann ein logarithmischer Umwandler wie folgt dargestellt werden.



Definition 10.12. Seien Σ und Δ Alphabete. Eine Abbildung $R\colon \Sigma^* \to \Delta^*$ heißt *mit logarithmischem Platz berechenbar*, wenn ein logarithmischer Umwandler existiert, der bei Eingabe von $x \in \Sigma^*$ das Wort $R(x) \in \Delta^*$ auf seinem Ausgabeband stehen hat, wenn er hält.

Definition 10.13 (Reduktion mit logarithmischem Platz)**.** Seien $A \subseteq \Sigma^*$ und $B \subseteq \Delta^*$ formale Sprachen. Dann ist A *reduzierbar mit logarithmischem Platz* auf B ($A \leqslant^{\log} B$), wenn eine mit logarithmischem Platz berechenbare Abbildung $R\colon \Sigma^* \to \Delta^*$ existiert, sodass für alle $x \in \Sigma^*$

$$x \in A \;\Leftrightarrow\; R(x) \in B\,.$$

Der nächste Satz zeigt, dass die Relation $\leqslant^{\log}$ die Relation $\leqslant^{\mathrm{p}}$ verfeinert.

Satz 10.5. *Seien A und B formale Sprachen. Wenn $A \leqslant^{log} B$, dann $A \leqslant^{p} B$.*

Beweis. Sei T ein logarithmischer Umwandler mit logarithmischem Platz, d.h. es gibt eine Konstante $c \in \mathbb{N}$, sodass T maximal $c \log n$ Felder am Arbeitsband liest. Es gibt also $\#(\Gamma)^{c \cdot \log n}$ verschiedene Möglichkeiten für den Arbeitsbandinhalt von T. Wir wählen ein $d \in \mathbb{N}$ mit $\#(\Gamma) \leq 10^d$. Dann ist

$$\#(\Gamma)^{c \log n} \leq (10^d)^{c \log n} = 10^{c \cdot d \log n} = (10^{\log n})^{c \cdot d} = n^{c \cdot d}$$

und somit können nur polynomiell viele Konfigurationen

$$\underbrace{\#(Q)}_{aktuellerZustand} \cdot \underbrace{n^{c \cdot d}}_{Bandinhalt} \cdot \underbrace{n}_{PositionSchreib-/Lesekopf}$$

auftreten. Weil T total ist, hat T eine polynomielle Laufzeit. Für Details verweisen wir auf [8]. □

Das folgende Resultat ist eine direkte Konsequenz aus den Sätzen 10.2 und 10.5.

Folgerung. *Seien A und B formale Sprachen.*

- *Wenn* $A \leqslant^{log} B$ *und* $B \in \mathsf{P}$*, dann* $A \in \mathsf{P}$*.*
- *Wenn* $A \leqslant^{log} B$ *und* $B \in \mathsf{NP}$*, dann* $A \in \mathsf{NP}$*.*

Beispiel 10.10. Für die Sprachen A und B aus Beispiel 10.9 gilt $A \leqslant^{\log} B$. (Die Abbildung R kann sogar ohne Platz berechnet werden.)

Wie für $\leqslant^{\mathrm{p}}$ kann man für $\leqslant^{\log}$ die Konzepte $\mathbb{C}$-hart und $\mathbb{C}$-vollständig definieren.

Satz 10.6. *Ob ein gerichteter Graph stark zusammenhängend ist, ist* NLOGSPACE*-vollständig (bezüglich* $\leqslant^{log}$*).* □

Ohne Beweis geben wir den folgenden Satz an; der Beweis kann etwa in [10] oder [5] nachgelesen werden.

Satz 10.7.

1. *Das Problem Maze ist* NLOGSPACE*-vollständig, das heißt vollständig für die Klasse* NLOGSPACE *in Bezug auf* $\leqslant^{log}$*.*
2. *Das Problem GEO ist* PSPACE*-vollständig, das heißt vollständig für die Klasse* PSPACE *in Bezug auf* $\leqslant^{log}$*.*

10.4 Aufgaben

Aufgabe 10.1. Betrachten Sie das Problem Maze. Berechnen Sie für den naiven Algorithmus bzw. für die Nachfolgersuche jeweils (asymptotische) Werte für die Laufzeitkomplexität bzw. Speicherplatzkomplexität für $n = 1$, $n = 10$, $n = 100$, $n = 1000$.

Aufgabe 10.2. Was ist die *Laufzeitkomplexität* einer Turingmaschine? Bestimmen Sie die Laufzeitkomplexität der Turingmaschinen aus den Beispielen 9.1 und 9.3.

Hinweis: Wenn Sie die Funktion $T(n)$ nicht genau angeben können, bestimmen Sie das asymptotische Wachstum mittels O Notation.

Aufgabe 10.3. Wie sind die Komplexitätsklassen P und NP definiert? In welcher Relation stehen sie zueinander? Scheint die Komplexitätsklasse $\mathsf{LOGTIME} := \mathsf{DTIME}(\log n)$ sinnvoll? Bestimmen Sie welche Probleme in welcher Komplexitätsklasse liegen.

	P	NP
Maze	✓	
HK	×	
PCP		
SAT		

Hinweis: Einige Kombinationen sind offene Probleme.

Aufgabe 10.4. Skizzieren Sie eine zweibändige DTM M mit linearer Laufzeitkomplexität, deren Sprache die Menge aller Palindrome ungerader Länge über dem Alphabet $\{0, 1\}$ ist. Warum wurde die Einschränkung auf Palindrome ungerader Länge getroffen?

Aufgabe 10.5. Seien A, B und C formale Sprachen. Zeigen Sie, dass die Reduktion mit polynomieller Zeit transitiv ist:

$$\text{Wenn } A \leqslant^{\mathrm{p}} B \text{ und } B \leqslant^{\mathrm{p}} C, \text{ dann } A \leqslant^{\mathrm{p}} C.$$

Aufgabe 10.6. Seien A und B formale Sprachen und $\mathbb{C}$ eine Komplexitätsklasse. Zeigen Sie:

$$\text{Wenn } A \ \mathbb{C}\text{-hart ist und } A \leqslant^{\mathrm{p}} B, \text{ dann ist } B \ \mathbb{C}\text{-hart.}$$

Gilt auch

$$\text{Wenn } A \ \mathbb{C}\text{-vollständig ist und } A \leqslant^{\mathrm{p}} B, \text{ dann ist } B \ \mathbb{C}\text{-vollständig.}$$

Hinweis: Verwenden Sie das Resultat von Aufgabe 10.5.

Aufgabe 10.7. Gegeben ein (Un-)Gleichungssystem $A\vec{x} \geq \vec{b}$, wobei $A \in \mathbb{Z}^{m \times n}$, $\vec{x}$ ein n-Vektor von Unbekannten und b ein m-Vektor mit Einträgen aus $\mathbb{Z}$. Das finden einer Lösung $\vec{x}$ nennt man *integer programming*. Zeigen Sie, dass integer programming NP-vollständig ist.

Hinweis: Reduzieren Sie von SAT, dh. geben Sie eine in polynomieller Zeit berechenbare Funktion r an, sodass

- $r(\varphi) = A\vec{x} \geq \vec{b}$ und
- φ erfüllbar genau dann, wenn $A\vec{x} \geq b$ eine Lösung besitzt.

Sie können annehmen, dass die aussagenlogische Formel in CNF gegeben ist.

Literaturverzeichnis

[1] Cormen, T. H. ; Leiserson, C. E. ; Rivest, R. L. ; Stein, C.: *Introduction to Algorithms.* MIT Press, Cambridge, 2001

[2] Hein, J. L.: *Discrete Structures, Logic, and Computability.* 3te Auflage. Jones and Bartlett Publishers, 2010

[3] Hopcroft, J. E. ; Motwani, R. ; Ullman, J. D.: *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, 2001. – 2te Auflage

[4] Kozen, D.: *Automata and Computability.* Springer Verlag, 1997

[5] Kozen, D.: *Theory of Computation.* Springer Verlag, 2006

[6] Menezes, A. J. (Hrsg.) ; van Oorschot, P. C. (Hrsg.) ; Vanstone, S. A. (Hrsg.): *Handbook of Applied Cryptography.* CRC Press, Boca Raton, 1997. – http://www.cacr.math.uwaterloo.ca/hac/

[7] Moser, G.: *Einführung in die Theoretische Informatik.* 5te Auflage. Institut für Informatik, 2015. – Skriptum zur Vorlesung

[8] Papadimitriou, C.H.: *Computational Complexity.* Addison-Wesley, 1995

[9] Pauer, F. (Hrsg.): *Einführung in die Mathematik 1.* Institut für Mathematik, 2008. – Skriptum zur Vorlesung

[10] Sipser, M.: *Introduction to the Theory of Computation.* PWS Publishing Company, 1997