HANSER

Leseprobe

Rolf Socher

Mathematik für Informatiker

Mit Anwendungen in der Computergrafik und Codierungstheorie

ISBN: 978-3-446-42254-4

Weitere Informationen oder Bestellungen unter

http://www.hanser.de/978-3-446-42254-4

sowie im Buchhandel.

3 Funktionen und Abzählbarkeit

3.1 Funktionen

Sie, liebe Leserin, lieber Leser, kennen sicherlich den *Cäsar-Code*. Er wurde vom römischen Feldherrn Gaius Julius Caesar (100 v. Chr. – 44 v. Chr.) in seiner militärischen Korrespondenz verwendet, um Botschaften geheim zu halten, sodass sie nur derjenige lesen konnte, der den Schlüssel zum Entziffern hatte.

Die Idee ist sehr einfach: Jeder Buchstabe des Alphabets wird um 3 Stellen verschoben. Wir beschränken uns im Folgenden auf das kleine lateinische Alphabet aus 26 Buchstaben. Aus a wird D, aus b wird E, aus c wird F usw. Moment, was heißt „usw."? Wie geht es am Schluss des Alphabets weiter? Na klar, dann fängt man einfach wieder von vorne an: Aus w wird Z, aus x wird A, aus y wird B und aus z wird C. Wir sprechen auch von einer *zyklischen Verschiebung*.

Klartext	a	b	c	d	...	v	w	x	y	z
Geheimtext	D	E	F	G	...	Y	Z	A	B	C

Dabei halten wir uns an die branchenübliche Konvention, Klartextbuchstaben klein- und Geheimtextbuchstaben großzuschreiben.

Als kryptografische Methode ist das Verfahren des römischen Feldherrn allenfalls von historischem Interesse. Nichtsdestominder lohnt sich ein näherer Blick darauf, denn an diesem simplen Beispiel lassen sich viele mathematische Begriffe und Methoden erläutern. Der erste grundlegende Begriff ist der der Funktion. Die obige Tabelle ordnet jedem Kleinbuchstaben des lateinischen Alphabets genau einen Großbuchstaben zu. Eine solche Zuordnung heißt *Funktion* oder *Abbildung*. Um eine konkrete Funktion zu definieren, müssen wir stets die zugeordneten Mengen (die Urbildmenge und die Bildmenge) angeben. In unserem Fall ist dies einmal das kleine und zum anderen das große lateinische Alphabet. Wir bezeichnen Funktionen meistens mit den Buchstaben f, g und h. Unsere Cäsarfunktion – nennen wir sie f_C – ist eine Funktion von der Menge der Kleinbuchstaben in die Menge der Großbuchstaben. Wir schreiben:

$$f_C : \{\mathrm{a,b,\ldots,z}\} \to \{\mathrm{A,B,\ldots,Z}\}\,.$$

Diese Schreibweise stellt eine abstrakte Spezifikation der Funktion dar, die besagt: Das Urbild (oder das Argument) muss ein Kleinbuchstabe sein, der Funktionswert ist ein Großbuchstabe. Sie sagt jedoch nichts darüber aus, wie man für einen konkreten Buchstaben x den Funktionswert $f_C(x)$ bestimmt[1]. Diese Funktionsvorschrift lautet: Verschiebe den Buchstaben x um 3 Positionen.

1 Dieses x ist nicht der Buchstabe x, sondern eine Variable, die für einen beliebigen Buchstaben stehen kann. Woran man das erkennt? Variablen werden stets *kursiv* gesetzt.

Definition
Funktion

Eine *Funktion* oder *Abbildung* $f : D \to M$ ist eine Vorschrift, die jedem Element $x \in D$ genau ein Element $f(x) \in M$ zuordnet. Man nennt $f(x)$ den *Funktionswert* von x. Die Funktionsvorschrift wird oft in der Form $x \mapsto f(x)$ angegeben.

Die Menge D heißt auch *Definitionsmenge*, M heißt auch *Wertemenge* von f. Die Menge

$$\{f(x) | x \in D\}$$

aller Funktionswerte heißt auch *Wertebereich* von f und wird oft $f(D)$ geschrieben. Der Wertebereich einer Funktion ist stets eine Teilmenge ihrer Wertemenge.

Die Begriffe *Funktion* und *Abbildung* werden in der Mathematik synonym verwendet. Welchen der beiden Begriffe man verwendet, hängt vom Kontext ab: In der Analysis, wo es hauptsächlich um reellwertige oder komplexe Funktionen geht, spricht man von Funktionen, während in der Algebra oder linearen Algebra von Abbildungen gesprochen wird.

Aufgabe Schreiben Sie ein Java-Interface (nur das Interface!) für eine Klasse Caesar. Diese Klasse enthält (zunächst) nur eine einzige Methode `caesarEncode`, die die Cäsar-Verschlüsselung für einen einzelnen Buchstaben realisiert: Gibt man einen Klartextbuchstaben ein, so gibt die Methode den Geheimtextbuchstaben zurück. Nehmen Sie für den Moment an, es gäbe in Java eine Klasse `letter` für Kleinbuchstaben und eine Klasse `Letter` für Großbuchstaben.

Lösung Im Interface wird die Methode lediglich deklariert, jedoch nicht implementiert. Das Interface sieht so aus:

```
public Letter caesarEncode(letter c);
```

Diese Deklaration entspricht in etwa der obigen abstrakten Kennzeichnung der Funktion f_C. ■

Entsprechende Funktionen kann man für jeden anderen Verschiebungswert $k \in \{0, 1, \ldots, 25\}$ definieren, ja sogar für $k = 0$. Zu Geheimhaltungszwecken ist der Wert $k = 0$ sicherlich sinnlos, aber eine Funktion ist es auf jeden Fall, nämlich die sogenannte identische Funktion:

Definition
identische
Funktion

Die *identische Funktion* $id_M : M \to M$ auf einer Menge M ist definiert durch

$$id_M(x) = x$$

für alle $x \in M$. Meistens wird der Index M weggelassen.

Beispiel 3.1 Mehr Funktionen

a) Die ASCII-Codierung ist eine Funktion, die jedem Zeichen (Buchstaben, Ziffern, Sonderzeichen) des ASCII-Zeichensatzes eine natürliche Zahl zuordnet. Wenn wir den ASCII-Zeichensatz Z nennen, so können wir schreiben

$$f_{\text{ASCII}} : Z \to \mathbb{N}_0.$$

In Java würde man deklarieren:

```
public int fAscii (char c);
```

Da es in Java keine Klasse gibt, die der Menge $\mathbb{N}_0$ entspricht, muss man die Klasse `int` nehmen. Selbstverständlich hätte man auch statt $f_{\text{ASCII}} : \text{Z} \to \mathbb{N}_0$ schreiben können $f_{\text{ASCII}} : \text{Z} \to \mathbb{Z}$. Das wäre nicht falsch, sondern bloß weniger exakt – und vielleicht auch ein wenig verwirrend („können da auch negative Zahlen herauskommen?“). Im Grunde genommen ist schon die Wertemenge $\mathbb{N}_0$ zu weit gefasst: $f_{\text{ASCII}} : \text{Z} \to \{0, 1, 2, \ldots, 127\}$ hätte es auch getan.

In der Mathematik findet man naturgemäß viele Funktionen auf Zahlenmengen:

b) Die Funktion $f : \mathbb{Z} \to \mathbb{Z}$, $x \mapsto x^2$ ordnet jeder natürlichen Zahl n ihr Quadrat zu, also: $f(0) = 0$, $f(1) = 1$, $f(-1) = 1$, $f(2) = 4$, $f(-2) = 4$ usw.

c) Die Funktion $f : \mathbb{R} \to \mathbb{R}$ mit $x \mapsto x^2$ hat zwar dieselbe Berechnungsvorschrift, jedoch eine andere Definitions- und Wertemenge als die Funktion in b). Es handelt sich definitiv um zwei verschiedene Funktionen! In Java ist das auch der Fall: Bei den beiden Methoden `public int square(int x)` und `public double square(double x)` handelt es sich um verschiedene Methoden.

d) Eine Funktion $f : \mathbb{R} \to \mathbb{R}$ mit $x \mapsto \frac{1}{x}$ gibt es nicht! Für jedes x aus der Definitionsmenge muss ein Funktionswert definiert sein, aber $\frac{1}{0}$ ist nicht definiert. Man behilft sich in solchen Fällen damit, dass man die Definitionsmenge entsprechend anpasst: $f : \mathbb{R}-\{0\} \to \mathbb{R}$ mit $f(x) = \frac{1}{x}$.

In Java ist es jedoch nicht möglich, die Definitionsmenge anzupassen. Es gibt keine Klasse, die der Menge $\mathbb{R}-\{0\}$ entspricht. In einem solchen Fall, in dem ein bestimmter Funktionswert undefiniert ist, **muss** die Methode **eine Exception werfen**! Und genau das passiert ja auch, wenn Sie durch 0 dividieren.

Darstellung von Funktionen

Funktionen lassen sich auf unterschiedliche Weisen darstellen. Welche Darstellungsform man wählt, hängt zum Teil von der Art der Funktion ab, insbesondere von Definitions- und Wertemenge, aber auch von gewissen Aspekten der Funktion, die man betonen möchte. Wenn es um die Darstellung eines Begriffes geht – und das trifft für die Mathematik genauso wie für die Informatik und alle Naturwissenschaften zu – geht es nicht um richtig oder falsch, sondern um sinnvoll oder brauchbar versus sinnlos bzw. nutzlos. Denken Sie an Netzpläne des öffentlichen Personen-Nahverkehrs, etwa die U- und S-Bahn-Pläne größerer Städte[1]. Die

1 Siehe etwa http://www.bvg.de/index.php/de/3713/name/Liniennetz.html für das Liniennetz der BVG in Berlin

Lage der eingezeichneten Haltestellen und ihre Entfernungen zueinander sind im Allgemeinen überhaupt nicht maßstabsgerecht eingezeichnet. Diese Pläne sind dennoch nicht falsch, sondern sie sind mehr oder weniger brauchbar für bestimmte Zwecke. Wenn Sie beispielsweise wissen wollen, mit welchen Linien Sie vom Ernst-Reuter-Platz zum Brandenburger Tor kommen, ist der Netzplan sehr nützlich. Wenn Sie dagegen die Strecke mit dem Auto fahren möchten, ist der Plan völlig unbrauchbar.

a) Die Funktionsdarstellung, an die Sie sicherlich zuerst denken, ist die Angabe einer Berechnungsformel, etwa in der Form $f : \mathbb{Z} \to \mathbb{Z}$, $x \mapsto x^2$. Nicht für alle Funktionen lässt sich jedoch eine Berechnungsformel angeben – wer etwa eine „Berechnungsformel" für Lottozahlen wüsste, der könnte schnell reich werden.
b) Eine Funktion mit einer endlichen Definitionsmenge lässt sich mithilfe einer Wertetabelle darstellen, so wie Sie das sicher noch aus der Schule kennen.
c) Funktionen auf endlichen Mengen lassen sich auch durch Pfeildiagramme darstellen.

Arno	Britta	Carl	Dörte
27	21	23	21

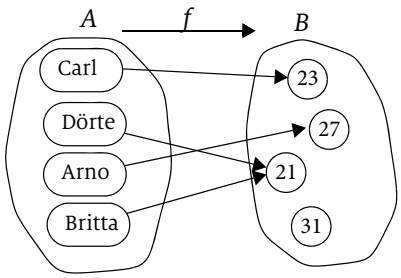


Abb. 3-1
Wertetabelle und Pfeildiagramm einer Funktion

d) Weitere Möglichkeiten der Darstellung von Funktionen mit endlichem Definitionsbereich sind Balken- oder Tortendiagramme.
e) Reelle Funktionen (d. h. Funktionen auf den reellen Zahlen) kann man mittels Funktionsgraphen darstellen.
f) Für viele Funktionen des Alltags wie die Berechnung des Briefportos aus Größe und Gewicht eines Briefes oder der KFZ-Steuer aus Hubraum und Abgaswerten gibt es keine mathematische Formel, sondern nur einen Algorithmus zur Berechnung.

Funktionen mit mehreren Argumenten

Hätte Cäsar seine geheimen Botschaften immer um 3 Stellen verschoben, so wären sie vermutlich nicht allzu lange geheim geblieben. Wäre es jemand gelungen, auch nur einen einzigen Geheimtext zu entziffern, so hätte er damit die feste Zuordnung f_C gefunden und somit alle Geheimtexte lesen können. Sicherlich hat schon Cäsar damals mit unterschiedlichen Verschiebungen gearbeitet. Eine grundlegende Maxime der Kryptografie lautet daher: *Nicht die Methode*, mit der ein Klartext verschlüsselt wurde, sorgt für die Sicherheit der Übermittlung, *sondern der Schlüssel*, der dabei verwendet wurde. Die Methode ist in unserem Fall die zyklische Verschiebung der Buchstaben an sich. Der Schlüssel k ist die Anzahl der

Stellen, um die verschoben wird. Auf diese Weise lässt sich der Schlüssel häufiger wechseln und damit wird die Gefahr, dass der Code gebrochen wird, geringer.

In unserem Kontext bedeutet dies, dass die Cäsar-Funktion einen zweiten Parameter k braucht. Nennen wir die Funktion diesmal nur f, so ordnet f einem Paar (x, k), bestehend aus einem Kleinbuchstaben x und einem Schlüssel k (also einer Zahl zwischen 0 und 25) einen Großbuchstaben zu. Wir schreiben:

$$f : \{a, b, \ldots, z\} \times \{0, 1, \ldots, 25\} \to \{A, B, \ldots, Z\}.$$

Dabei bezeichnet $\times$ das kartesische Produkt (▶ Abschnitt 2.2) der beiden Mengen. Die Java-Methode müsste dann folgendermaßen angepasst werden:

```
public Letter caesarEncode(letter c, int k);
```

Entsprechend können wir die Addition auf den ganzen Zahlen folgendermaßen als Funktion beschreiben:

$$f : \mathbb{Z} \times \mathbb{Z} \to \mathbb{Z}$$
$$f(x, y) = x + y\,.$$

Komposition von Funktionen

Bisher haben wir die Java-Methode nur deklariert. Wie könnte man sie konkret implementieren? Es liegt auf der Hand, den Kleinbuchstaben x zunächst in eine Zahl zwischen 0 und 25 umzuwandeln (etwa mithilfe der ASCII-Codierung), anschließend den Schlüssel k zu addieren und schließlich diese Zahl wieder zurück in einen Großbuchstaben (mithilfe der ASCII-Decodierung) verwandeln. Dabei muss man lediglich aufpassen, dass man bei der Addition von k im Bereich $\{0, \ldots, 25\}$ bleibt. Die konkrete Realisierung überlasse ich Ihnen (▶ Aufgabe 3.1).

Hier werden nacheinander mehrere Funktionen angewandt: ASCII-Codierung, Addition des Schlüssels, ASCII-Decodierung. Die gesamte Funktion, die (bei festem Schlüsselwert $k = 3$) aus dem Buchstaben a den Buchstaben D macht, aus b ein E usw., wird als Komposition von 3 einzelnen Funktionen dargestellt (▶ Abbildung 3-2).

Definition
Komposition von Funktionen

Sind $f : A \to B$ und $g : B \to C$ Funktionen, so ist die *Komposition* (oder *Verkettung*)

$$g \circ f : A \to C$$

definiert durch die Funktionsvorschrift

$$(g \circ f)(x) = g(f(x))\,.$$

Beachten Sie dabei:

- $g \circ f$ bedeutet: erst f, dann g!
- Die Verkettung $g \circ f$ ist nur definiert, wenn die Wertemenge von f mit der Definitionsmenge von g übereinstimmt.

Für jede Funktion $f : A \to B$ gilt offenbar $id \circ f = f \circ id = f$. Beachten Sie, dass es sich hierbei genaugenommen um zwei verschiedene identische Funktionen handelt – um welche?

Außerdem gilt für die Funktionskomposition das Assoziativgesetz, das heißt, sind $f : A \to B$, $g : B \to C$, $h : C \to D$ Funktionen, so gilt:

$$(h \circ g) \circ f = h \circ (g \circ f).$$

Dies lässt sich einfach nachrechnen:

$$((h \circ g) \circ f)(x) = (h \circ g)(f(x)) = h(g(f(x)))$$

und

$$(h \circ (g \circ f))(x) = h((g \circ f)(x)) = h(g(f(x))).$$

Beachten Sie jedoch, dass im Allgemeinen $f \circ g$ und $g \circ f$ nicht identisch sind. Das fängt schon damit an, dass eine der beiden Funktionen vielleicht gar nicht definiert ist. Aber selbst dann, wenn beide Kompositionen definiert sind, sind sie im Allgemeinen nicht identisch (▶ Aufgabe 3.3).

Aufgaben zu 3.1

3.1 Vervollständigen Sie den Methodenrumpf der Methode `caesarEncode`. Verwenden Sie dabei die modulo-Operation, die in Java mit dem %-Zeichen geschrieben wird, um die Zahl im Bereich von 0 bis 25 zu halten.

3.2 Welche der folgenden Vorschriften sind zulässige Funktionsvorschriften?

a) $f : \mathbb{N} \to \mathbb{N}$ mit $x \mapsto \sqrt{x}$

b) $f : \mathbb{R} \to \mathbb{R}$ mit $x \mapsto \sqrt{x}$

c) $f : \mathbb{N} \to \mathbb{R}$ mit $x \mapsto \sqrt{x}$

3.3 Seien $f : \mathbb{Z} \to \mathbb{Z}$ und $g : \mathbb{Z} \to \mathbb{Z}$ definiert durch

$$f(x) = x + 1$$

$$g(x) = x^2.$$

Zeigen Sie, dass $f \circ g \neq g \circ f$ ist.

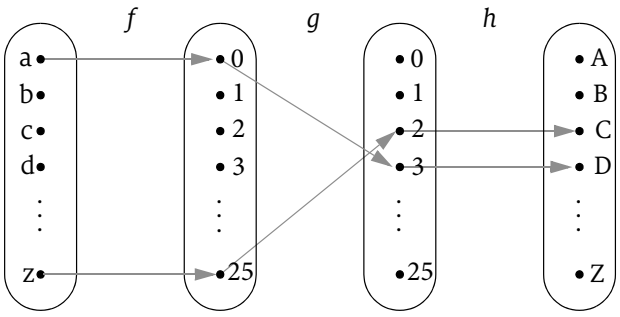


Abb. 3-2
Verkettung von Funktionen

3.4 Geben Sie ein einfaches Beispiel für eine Funktion $f: M \to M$, $f \neq id$, für die
a) $f \circ f = id$ gilt,
b) $f \circ f = f$ gilt.

Projekt

Projekt 1: Kryptoanalyse des Cäsar-Codes

Informieren Sie sich über die mathematischen Möglichkeiten, den Cäsar-Code zu knacken.

Programmierprojekt

Der Cäsar-Code

Schreiben Sie ein Java-Programm, das die Ver- und Entschlüsselung mit dem Cäsar-Code realisiert. Schön wäre natürlich eine grafische Oberfläche, in der Sie

- einen Klartext eingeben (oder aus einer Datei laden) können,
- einen Geheimtext eingeben (oder aus einer Datei laden) können,
- einen Schlüssel eingeben können,
- einen Klartext verschlüsseln und einen Geheimtext entschlüsseln können.

3.2 Injektive, surjektive und bijektive Funktionen und die Umkehrfunktion

Bleiben wir beim Thema Codierung. Um die Darstellung möglichst einfach zu halten, betrachten wir nur den Funktionsanteil, der die Zahlenmenge $\{0,\ldots,25\}$ auf sich selbst abbildet – das ist der eigentlich interessante Teil. Der Rest besteht immer nur aus ASCII-Codierung und -Decodierung.

Was sagen Sie zu folgender Codierung?

$$f: \{0,\ldots,25\} \to \{0,\ldots,25\}$$

$$x \mapsto (2x) \,\%\, 26$$

Dabei bezeichnet $a \,\%\, b$ den Rest bei der ganzzahligen Division von a durch b. Diese Codierung resultiert in folgender Tabelle:

Klartext	a	b	c	d	...	n	o	p	q	...
Geheimtext	A	C	E	G	...	A	C	E	G	...

Diese Funktion ist offenbar als Codierung ungeeignet, denn es ist keine eindeutige Decodierung möglich. Ein „A“ kann im Klartext sowohl ein „a“ als auch ein „n“ sein. Um die Decodierung zu ermöglichen, dürfen unterschiedliche Klartextbuchstaben nicht auf denselben Geheimtextbuchstaben abgebildet werden. Man kann es auch so ausdrücken: Ein Geheimtextbuchstabe darf nicht mehr als ein „Urbild“ unter der Decodierfunktion haben. Eine Funktion, die diese Eigenschaft hat, heißt

injektiv. Die obige Funktion ist nicht injektiv. Betrachten Sie als weiteres Beispiel die Altersfunktion aus Abbildung 3-1: Wer ist der/die 21-jährige Student(in)? Sie sehen: Die Altersfunktion ist ebenfalls nicht injektiv, denn zu der Zahl 21 gibt es zwei verschiedene „Urbilder".

Definition
injektive Funktion

Die Funktion $f : A \to B$ heißt *injektiv*, wenn es zu jedem $y \in B$ **höchstens** ein $x \in A$ gibt mit $f(x) = y$.

Die Funktion f ist genau dann injektiv, wenn aus $f(x) = f(x')$ folgt, dass $x = x'$ ist.

Mit Matrikelnummern kann dies nicht passieren: Verschiedene Studierende (an derselben Hochschule) haben auch unterschiedliche Matrikelnummern. Die „Matrikelnummerfunktion" ist injektiv. Bei der Rückverfolgung kann jedoch ein anderes Problem entstehen: Wenn ich etwa den/die Student(in) mit der Nummer 20099876 suche, kann es durchaus sein, dass diese Matrikelnummer gar nicht existiert. Eine Funktion, die dieses Problem nicht hat, heißt surjektiv.

Definition
surjektive Funktion

Die Funktion $f : A \to B$ heißt *surjektiv*, wenn es zu jedem $y \in B$ **mindestens** ein $x \in A$ gibt mit $f(x) = y$.

Die Funktion f ist genau dann surjektiv, wenn der Wertebereich von f gleich der Wertemenge B ist.

Das Problem, das mit nicht surjektiven Funktionen entstehen kann, ist offenbar ein Problem der genauen Kenntnis des Wertebereichs der Funktion. Wenn ich beispielsweise genau weiß, welche Matrikelnummern tatsächlich auftreten, dann kann ich diese (im Prinzip wenigstens) rückverfolgen.

Definition
bijektive Funktion

Die Funktion $f : A \to B$ heißt *bijektiv*, wenn es zu jedem $y \in B$ **genau** ein $x \in A$ gibt mit $f(x) = y$.

Die Funktion f ist genau dann bijektiv, wenn sie injektiv und surjektiv ist.

Abbildung 3-3 zeigt die Eigenschaften injektiv, surjektiv und bijektiv anhand von Pfeildiagrammen.

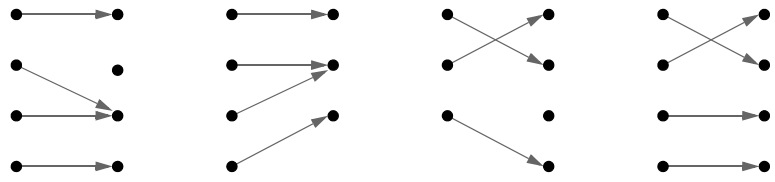


Abb. 3-3
Injektive, surjektive, bijektive Funktionen

Beispiel 3.2 Wir betrachten die Funktion $f(x) = x^2$ mit unterschiedlichen Definitions- und Wertemengen.

a) $f : \mathbb{N} \to \mathbb{N}$, $n \mapsto n^2$: Injektiv, denn zu jeder natürlichen Zahl m gibt es höchstens eine natürliche Zahl n mit $n^2 = m$. Nicht surjektiv, denn es gibt keine natürliche Zahl n mit $n^2 = 2$.

b) $f : \mathbb{Z} \to \mathbb{Z}$, $x \mapsto x^2$: Nicht injektiv, denn $(-1)^2 = 1^2$; nicht surjektiv (siehe a)).

c) $f : \mathbb{R} \to \mathbb{R}$, $x \mapsto x^2$: Nicht injektiv (siehe b)). Surjektiv, denn jede reelle Zahl y hat mindestens eine Wurzel.

d) $f : \mathbb{R}^+ \to \mathbb{R}^+$, $x \mapsto x^2$: Bijektiv (also injektiv und surjektiv), denn jede positive reelle Zahl y hat genau eine positive Wurzel. ■

Das Beispiel verdeutlicht ein weiteres Mal, wie wichtig es ist, Definitions- und Wertemenge einer Funktion anzugeben.

Die Cäsar-Verschiebung und die ASCII-Codierung $f_{\text{ASCII}} : \mathrm{Z} \to \{0, 1, 2, \ldots, 127\}$ sind Beispiele für bijektive Funktionen. Beide Funktionen sind umkehrbar, das heißt, sie können rückgängig gemacht werden: Die Umkehrung der Codierung ist die Decodierung. Das bedeutet: Wenn ich einen Klartextbuchstaben erst codiere, dann das Ergebnis decodiere, so muss wieder der ursprüngliche Buchstabe herauskommen. Wenn ich umgekehrt eine Zahl $n \in \{0,1,2,\ldots,127\}$ erst zu einem ASCII-Zeichen decodiere, dann das Ergebnis wieder codiere, muss das Ergebnis die Zahl n sein.

Definition
Umkehrfunktion

Die Funktion $f : A \to B$ heißt *umkehrbar* (oder *invertierbar*), wenn es eine Funktion $g : B \to A$ gibt mit

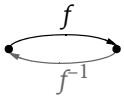
$$g(f(x)) = x \text{ für alle } x \in A$$

und

$$f(g(x)) = x \text{ für alle } x \in B.$$

In diesem Fall heißt g die *Umkehrfunktion* (oder *Inverse*) von f. Wir schreiben f^{-1} für die Umkehrfunktion von f.

In Kurzschreibweise: Die Funktion f heißt umkehrbar, wenn es eine Funktion g gibt, sodass $f \circ g$ und $g \circ f$ beide definiert sind und $f \circ g = g \circ f = id$ gilt.



In sehr vielen Anwendungen ist es wichtig, durchgeführte Aktionen oder Operationen wieder rückgängig machen zu können. Dies erklärt die besondere Bedeutung der Umkehrabbildung. In einem Pfeildiagramm erhalten Sie die Umkehrfunktion von f (falls sie existiert), indem Sie die Pfeile von f umdrehen.

Nicht jede Funktion ist umkehrbar. Ist die Funktion f jedoch invertierbar, so ist ihre Umkehrfunktion eindeutig bestimmt: Sind nämlich g und h beide Inverse von f, so ist $f \circ g = id$ und $h \circ f = id$, und es folgt:

$$h \circ f \circ g = h \circ (f \circ g) = h \circ id = h.$$

Aber andererseits ist

$$h \circ f \circ g = (h \circ f) \circ g = id \circ g = g.$$

Also ist $h = g$.

In einem Pfeildiagramm erhalten Sie aus f die Inverse f^{-1} (falls sie existiert), indem Sie alle Pfeile umdrehen. Drehen Sie ein zweites Mal um, so erhalten Sie wieder die Ausgangsfunktion f. Das heißt, die Inverse von f^{-1} ist wieder f:

$$(f^{-1})^{-1} = f.$$

Satz

Eine Funktion ist genau dann umkehrbar, wenn sie bijektiv ist.

Beweis: Sei $f : A \to B$ bijektiv. Wir definieren eine Funktion $g : B \to A$ folgendermaßen: Für $x \in B$ sei $g(x)$ das eindeutig bestimmte Urbild von x unter f. Dann ist g die Umkehrfunktion von f.

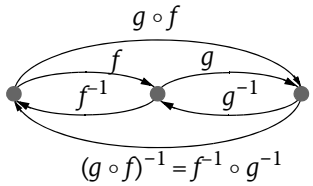
Sei umgekehrt $f : A \to B$ umkehrbar. Dann existiert die Umkehrfunktion $f^{-1} : B \to A$. Wir zeigen zunächst, dass f injektiv ist: Sei $f(x) = f(x')$. Wir wenden f^{-1} auf beiden Seiten der Gleichung an und erhalten: $f^{-1}(f(x)) = f^{-1}(f(x'))$, also $x = x'$. Wir zeigen nun, dass f surjektiv ist: Sei $y \in B$ und sei $x = f^{-1}(y)$. Dann ist $f(x) = f(f^{-1}(y)) = y$. Dies zeigt, dass jedes Element von B ein Urbild unter f hat, das heißt, dass f surjektiv ist. ■

Satz

Sind die Funktionen $f : A \to B$ und $g : B \to C$ beide bijektiv (also invertierbar), so ist auch die *Komposition* $g \circ f : A \to C$ bijektiv (also auch invertierbar) und es gilt:

$$(g \circ f)^{-1} = f^{-1} \circ g^{-1}.$$

Beweis: Wir zeigen zunächst, dass die Komposition zweier bijektiver Funktionen wieder bijektiv ist. Dazu zeigen wir, dass es zu jedem $c \in C$ genau ein $a \in A$ gibt mit $(g \circ f)(a) = c$. Sei $c \in C$. Dann gibt es genau ein $b \in B$ mit $g(b) = c$, denn g ist bijektiv. Für dieses b gibt es genau ein $a \in A$ mit $f(a) = b$, denn f ist bijektiv. Also gibt es genau ein $a \in A$ mit



$$(g \circ f)(a) = g(f(a)) = g(b) = c.$$

Es gilt:

$$(g \circ f) \circ (f^{-1} \circ g^{-1}) = g \circ f \circ f^{-1} \circ g^{-1} = g \circ id \circ g^{-1} = g \circ g^{-1} = id.$$

Daraus folgt, dass $f^{-1} \circ g^{-1}$ eine Inverse von $g \circ f$ ist. Weiterhin wissen wir, dass die Inverse einer bijektiven Funktion eindeutig ist. Daraus folgt die Aussage des Satzes. ■

Aufgaben zu 3.2

3.5 Welche der Eigenschaften injektiv, surjektiv und bijektiv trifft auf die folgenden Funktionen zu?

a) $f : \mathbb{Z} \to \mathbb{Z}$, $x \mapsto 3x - 2$

b) $f : \mathbb{R} \to \mathbb{R}$, $x \mapsto 3x - 2$

c) $f : \mathbb{R} \to \mathbb{R}$, $x \mapsto 2^x$

3.6 Gegeben sei die Menge P = {Arno, Bettina, Carl, Dagmar, Emil, Franziska} von Personen. Bestimmen Sie für jede der folgenden Funktionen f eine sinnvolle Wertemenge B. Welche der Eigenschaften injektiv, surjektiv und bijektiv trifft auf die Funktion f zu?

a) $f : P \to B$, $x \mapsto$ Anzahl der Buchstaben des Vornamens von x

b) $f : P \to B$, $x \mapsto$ Erster Buchstabe des Vornamens von x

c) $f : P \to B$, $x \mapsto$ Geschlecht von x

3.7 Drehen Sie jeweils die Pfeile in den ersten drei Pfeildiagrammen von Abbildung 3-3 um und erklären Sie in jedem Fall, warum das Ergebnis keine Funktion sein kann.

3.8 a) Finden Sie ein Beispiel für zwei Funktionen f und g, sodass die Kompositionen $f \circ g$ und $g \circ f$ beide definiert sind und $g \circ f = id$ und $f \circ g \neq id$ gilt.

b) Welche Eigenschaften (injektiv, surjektiv, bijektiv) müssen f und g haben, damit a) überhaupt möglich ist?

3.9 a) Finden Sie ein Beispiel für eine Funktion $f \neq id$, sodass die Komposition $f \circ f$ definiert ist und $f \circ f = id$ gilt.

b) Welche Eigenschaften (injektiv, surjektiv, bijektiv) muss f haben, damit a) überhaupt möglich ist?

3.10 Bestimmen Sie jeweils die Umkehrfunktion folgender Funktionen.

a) $f : \mathbb{R} \to \mathbb{R}$, $x \mapsto 3x - 2$

b) $f : \mathbb{R}^+ \to \mathbb{R}$, $x \mapsto x^2 - 2$

3.11 Sei $f : A \to B$ eine Funktion und sei $\equiv$ die durch f induzierte Äquivalenzrelation (▶ S. 61).

a) Wie viele Äquivalenzklassen hat die Relation $\equiv$, wenn f injektiv ist?

b) Wie viele Äquivalenzklassen hat die Relation $\equiv$, wenn f surjektiv ist?

3.3 Endliche und unendliche Mengen

Das Schubfachprinzip

Das *Schubfachprinzip* (auch *Taubenschlagprinzip* genannt, engl. *pigeon hole principle*) lautet: