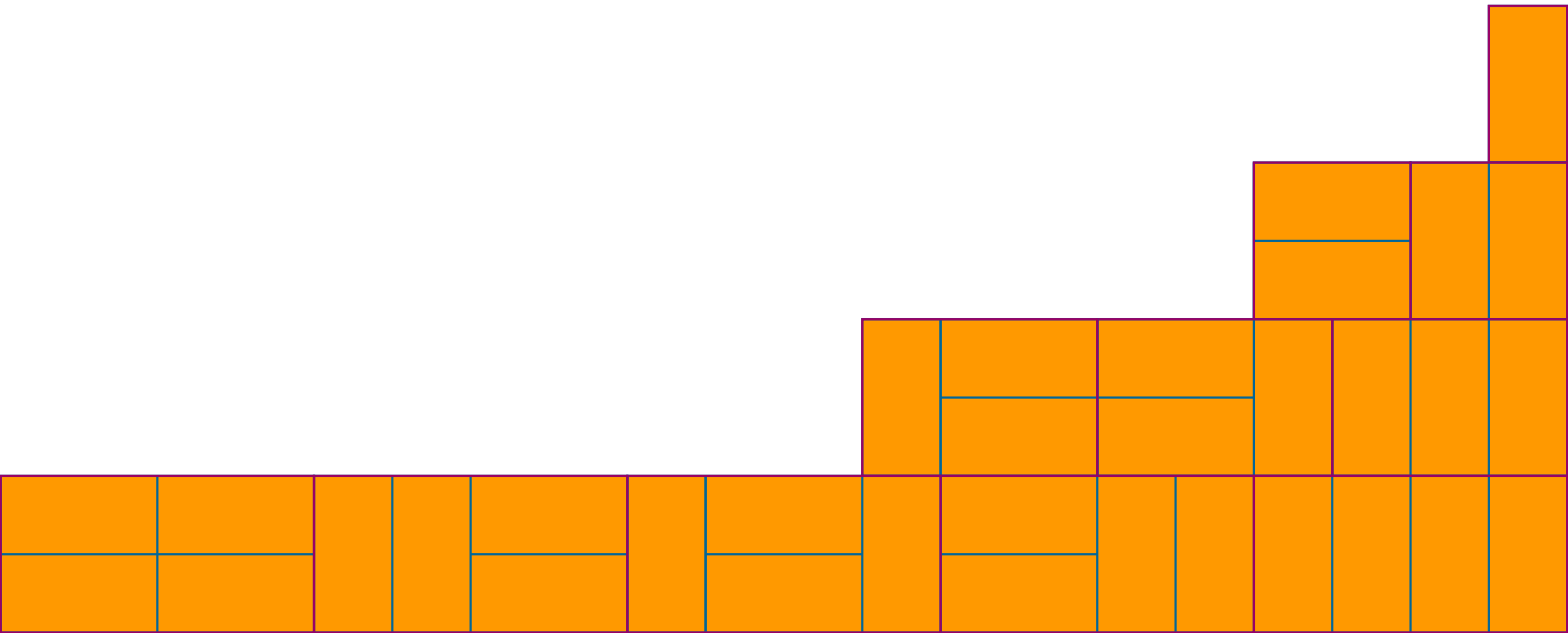
Grundlagen der Mathematik

Mathematik für Informatik/Data Science
Wintersemester 2023/24

Clara Löh



Version vom 27. November 2023
clara.loeh@mathematik.uni-regensburg.de
Fakultät für Mathematik, Universität Regensburg, 93040 Regensburg

Dungeon Map

7 Reelle und komplexe Zahlen

komplexe Zahlen	$\mathbb{C}$	algebraisch abgeschlossen
	↕	„imaginäre Einheit“
reelle Zahlen	$\mathbb{R}$	vollständig
	↕	„formale Steigungen“
rationale Zahlen	$\mathbb{Q}$	Körper
	↕	„formale Brüche“
ganze Zahlen	$\mathbb{Z}$	additive Gleichungen lösbar
	↕	„formale Differenzen“
natürliche Zahlen	$\mathbb{N}$	Induktionsprinzip

5 Graphen, Bäume, Modellierung

Graphen:
- ungerichtet: (V, E) mit $E \subset V[2]$
- gerichtet: (V, E) mit $E \subset V \times V$

Bäume:
- nicht-leere, zusammenhängende Graphen ohne Kreise
- alternative Charakterisierung durch Existenz und Eindeutigkeit von Wegen
- spezielle Bäume: binäre Wurzelbäume

Anwendungen: Modellierung von
- Beziehungen zwischen Objekten
- sozialen und anderen Netzwerken
- Färbungs-/Verteilungsproblemen
- Datenstrukturen
- ...

4 Induktion ...

Induktionsprinzip der natürlichen Zahlen:
Ist $A \subset \mathbb{N}$ mit
- Induktionsanfang: $0 \in A$ und
- Induktionsschritt: für alle $n \in A$ gilt: $n + 1 \in A$,

so folgt $A = \mathbb{N}$.

4 ... und Rekursion

Rekursionsprinzip der natürlichen Zahlen:
Ist A eine Menge, $a \in A$ und $g\colon A \longrightarrow A$ eine Abbildung,
so gibt es genau eine Abbildung $f\colon \mathbb{N} \longrightarrow A$ mit
- Rekursionsanfang: $f(0) = a$
- Rekursionsschritt: für alle $n \in A$ gilt: $f(n+1) = g\big(f(n)\big)$.

Zahlen | Graphen | Induktion | Logik | Mengen

1 Syntax und Semantik

Aussagenlogik/Quantorenlogik:

Syntax, Semantik } jeweils per „divide & conquer“

$\neg$	nicht
$\wedge$	und
$\vee$	oder
$\Longrightarrow$	impliziert; wenn ..., dann ...
$\Longleftrightarrow$	äquivalent zu; genau dann, wenn
$\forall_x$	Für alle x gilt ...
$\exists_x$	Es existiert ein x mit ...

Aussagenlogische Tautologie: erhält bei jeder w/f-Belegung aller Variablen den Wert w.

2 Beweise

Beweisschritte über eine Sprache/Theorie T:
- Axiome/Voraussetzungen
- quantorenlogische Axiome
- aussagenlogische Tautologien über T
- Modus Ponens
- Generalisierungsregel

Korrektheitssatz (und Vollständigkeit)

Beweisstruktur:
- modularisieren
- abstrahieren
- Baukastenprinzip: Intro/Elim
- Spezielle Beweisstrukturen: Äquivalenzen, Kontraposition, Reductio ad absurdum, Widerspruchsbeweis, ...

3 Mengen und Abbildungen

Mengen:
- sind genau dann gleich, wenn sie dieselben Elemente enthalten;
- wichtige Konstruktionen: $\cap \quad \cup \quad \setminus \quad P(\,\cdot\,) \quad \times \quad \{\ldots \mid \ldots\} \quad \emptyset$
- $\{x \mid x \text{ ist eine Menge und } x \notin x\}$ ist *keine* Menge!

Abbildungen:
- sind durch ihren Graphen definiert;
- wichtige Konstruktionen: Komposition, Restriktion

Wichtige Eigenschaften von Abbildungen
- Injektivität
- Surjektivität
- Bijektivität

6 Relationen

Relation:
- auf X: Teilmenge von $X \times X$
- zwischen X und Y: Teilmenge von $X \times Y$

Beispiele:
- gerichtete Graphen
- Abbildungen
- Übergangsrelationen
- Ordnungen

Spezielle Relationen:
- Äquivalenzrelationen
  - ⇝ Äquivalenzklassen
  - ⇝ Quotientenkonstruktionen (z.B. $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{Z}/n$)
- partielle/totale Ordnungen
  - ⇝ Such-/Sortierverfahren bzw. Induktions-/Rekursprinzipien

Inhaltsverzeichnis

Literaturhinweise

Die Vorlesung wird sich nicht an einer einzelnen Quelle orientieren – Sie sollten also individuell je nach Thema und eigenen Vorlieben die Literatur auswählen, die am besten zu Ihnen passt.

Mathematik für Informatiker

Alle für uns relevanten Begriffe und Methoden werden in der Vorlesung und im Skript behandelt – und sollen auch so in den Übungen eingesetzt werden. Zusätzliche Quellen können aber für das Gesamtverständnis hilfreich sein. Es gibt sehr viel Literatur mit den folgenden Titeln (oder kleinen Variationen davon):

- Mathematik für Informatiker
- Diskrete Mathematik für Informatiker
- Logik für Informatiker

Sie sollten mehrere solche Bücher anschauen und dann entscheiden, (ob) welche davon für Sie als Ergänzung zur Vorlesung geeignet sind. Das Material wird nicht immer in derselben Reihenfolge präsentiert und es gibt bei den Konventionen und Schwerpunkten Unterschiede.

Die nachfolgende Liste enthält Anregungen für vertiefende Literatur – dort wird deutlich mehr (und oft auch deutlich anders) behandelt als in dieser Vorlesung!

Logik und Mengenlehre

- G.S. Boolos, J.P. Burgess, R.C. Jeffrey. *Computability and Logic*, fifth edition, Cambridge University Press, 2007.
- P.J. Cameron. *Sets, Logic and Categories*, Universitext, Springer, 1998.
- H.-D. Ebbinghaus, J. Flum, W. Thomas. *Einführung in die mathematische Logik*, 5. Auflage, Spektrum Akademischer Verlag, 2007.
- U. Friedrichsdorf, A. Prestel. *Mengenlehre für den Mathematiker*, Vieweg, 1985.
- C.N. Delzell, A. Prestel. *Mathematical Logic and Model Theory: A Brief Introduction*, Universitext, Springer, 2011.
- W. Rautenberg. *Einführung in die mathematische Logik: Ein Lehrbuch*, Vieweg+Teubner, 2008.
- R.M. Smullyan, M. Fitting. *Set theory and the continuum problem*, überarbeitete Auflage, Dover, 2010.

Diskrete Mathematik

- M. Aigner. *Diskrete Mathematik*, sechste Auflage, Vieweg+Teubner, 2006.
- R. Diestel. *Graph Theory*, dritte Auflage, Graduate Texts in Mathematics, 173, Springer, 2005.
- R.L. Graham, D.E. Knuth, O. Patashnik. *Concrete Mathematics: A Foundation for Computer Science*, Addison–Wesley Publishing Company, 1994.
- J.M. Harris, J.L. Hirst, M.J. Mossinghoff. *Combinatorics and Graph Theory*, zweite Auflage, Undergraduate Texts in Mathematics, Springer, 2008.

Zahlen

- H.-D. Ebbinghaus et. al.. *Zahlen*, Springer, 1992.

Beweisen und Problemlösen

- J. Avigad, L. de Moura, S. Kong. *Theorem Proving in Lean*, https://lean-lang.org/theorem_proving_in_lean/, 2023.
- A. Beutelspacher. *Das ist o.B.d.A. trivial!*, neunte Auflage, Vieweg+Teubner, 2009.
- J. Cummings. *Proofs: A Long-Form Mathematics Textbook*, Indepdently published, 2021.
- A.G. Konforowitsch. *Logischen Katastrophen auf der Spur*, zweite Auflage, Fachbuchverlag Leipzig, 1994.
- L. Lamport. How to write a 21st century proof, *J. Fixed Point Theory Appl.*, 11(1), 43–63, 2012.
- C. Löh. *Exploring Formalisation. A Primer in Human-Readable Mathematics in Lean 3 with Examples from Simplicial Topology*, Surveys and Tutorials in the Applied Mathematical Sciences, 11, Springer, 2022.
- S. Mimram. *PROGRAM = PROOF*, Independently published, 2020.
- G. Polya, J.H. Conway (Hrsg.). *How to Solve it: A New Aspect of Mathematical Method*, Princeton Science Library, 2014.
- T. Tao. *Solving mathematical problems. A personal perspective*, Oxford University Press, 2006.

Weiterführende Literatur

- M. Aigner, G.M. Ziegler, *Proofs from The Book*, dritte Auflage, Springer, 2004.
- A. Doxiadis, C. Papadimitriou, A. Papadatos, A. Di Donna, *Logicomix: An epic search for truth*, Bloomsbury Publishing, 2009.
- D. Djukić, V. Janković, I. Matić, N. Petrović. *The IMO Compendium: A Collection of Problems Suggested for International Mathematical Olympiads 1959–2009*, Problem Books in Mathematics, zweite Auflage, Springer, 2011.
- A. Engel. *Problem Solving Strategies*, *Problem Books in Mathematics*, Springer, 1998.
- D.R. Hofstadter. *Gödel, Escher, Bach: An Eternal Golden Braid*, 20 anniversary edition, Basic Books, 1999.
- C. Löh, S. Krauss, N. Kilbertus. *Quod erat knobelandum*, zweite Auflage, Springer Spektrum, 2019.

0
Einführung

Was ist Mathematik?

Das Wort „Mathematik“ leitet sich vom altgriechischen Wort „$\mu\acute{\alpha}\vartheta\eta\mu\alpha$“ ab, das in etwa „Lernen, Wissen, Wissenschaft, . . . “ bedeutet.

Mathematik ist eine Wissenschaft des abstrakten Denkens, insbesondere des Studiums abstrakter Strukturen (wie z.B. Zahlen, Graphen, Geometrie, . . .) und des Studiums von formalen Methoden. Die klassische Mathematik baut sich von den grundlegenden logischen und mengentheoretischen Axiomen Schritt für Schritt aus den folgenden Bausteinen auf:

- **Axiome** legen die Spielregeln für das betrachtete Gebiet fest.
- **Definitionen** führen neue Begriffe ein.
- **Sätze, Lemmata, Korollare** formulieren Aussagen über mathematische Objekte.
- **Beweise** sind formale Begründungen für behauptete Aussagen. Man beachte dabei, dass auch der Begriff des Beweises mathematisch präzise definiert ist.
- **Beispiele** veranschaulichen die Bedeutung und Tragweite der betrachteten Begriffe und Sätze.

Man kann Mathematik als Programmiersprache für Axiome/Definitionen/Sätze/Beweise auffassen. Traditionell wird diese Programmiersprache in

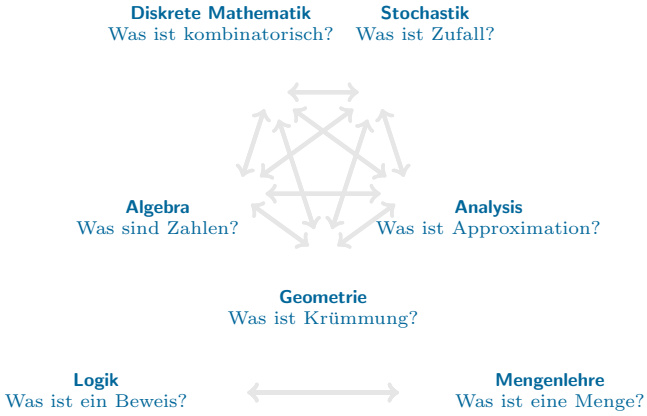


Abbildung 0.1.: Schematischer Aufbau der Mathematik, stark vereinfacht

natürlicher Sprache formuliert und im menschlichen Gehirn interpretiert. Beweisassistenten liefern eine technische Umsetzung des Konzepts

„proofs = programs“

als Programmiersprache im Sinne der Informatik; insbesondere können somit Beweise maschinell verifiziert werden.

Das Faszinierende an der Mathematik ist, dass sie einen exakten und eleganten Rahmen liefert, der aber auch in der Praxis (z.B. in den Naturwissenschaften, in der Informatik, in der Wirtschaft, in der Linguistik ...) erfolgreich eingesetzt und angewendet werden kann.

Ein grober schematischer Überblick über den Aufbau der Mathematik und ihrer Teilgebiete findet sich in Abbildung 0.1; stellvertretend ist jeweils eine zentrale Frage jedes Gebiets genannt, die andeutet, womit sich das Gebiet befasst. Zwischen den Gebieten gibt es vielfältige Verbindungen und Mischgebiete (z.B. algebraische Geometrie, diskrete Geometrie, stochastische Geometrie, geometrische Analysis, ...).

Wozu Mathematik in der Informatik?

Mathematik und Informatik interagieren auf viele Weisen und in manchen Teilgebieten sind die Bezüge so stark, dass eine genaue Zuordnung zur Mathematik oder Informatik nicht möglich oder sinnvoll ist. Dies betrifft insbesondere die diskrete Mathematik, logische Grundlagen und das maschinelle Lernen.

Mathematik wird in der Informatik sowohl als Sprache als auch als Werkzeug eingesetzt, zum Beispiel für die folgenden Aufgaben:

- präzise Spezifikation und Modellierung; z.B. Netzwerke (Graphentheorie), Computergraphik (lineare Algebra, Differentialgeometrie), maschinelles Lernen (Analysis, Stochastik, Numerik), Kryptographie (Algebra, Stochastik), ...
- präzise Beschreibung von Eigenschaften von Algorithmen/Systemen; insbesondere Korrektheit (relativ zur Spezifikation), Komplexitätsanalyse (Speicherplatz und Zeit)
- Beweis von Eigenschaften von Algorithmen/Systemen
- Einsatz geeigneter Abstraktionsmechanismen

Präzise Beschreibungen und formale Analysen sind die Grundlage für robuste, skalierbare, effiziente und transparente komplexe Systeme. Um komplexe Systeme auf einem hohen Qualitätsniveau zu entwerfen bzw. zu implementieren ist daher ein souveräner Umgang mit mathematischen Konzepten nicht nur förderlich, sondern unerlässlich.

Umgekehrt können Methoden aus der Informatik in der Mathematik eingesetzt werden:

- Beweisassistenten
- Experimentelle Erkundung von Strukturen oder Behauptungen; insbesondere Suche nach Gegenbeispielen oder Plausibilitätsüberprüfung
- Formulierung neuer Grundlagen (z.B. basierend auf Typtheorie statt Mengenlehre)

Über diese Vorlesung

In dieser Vorlesung werden wir uns mit grundlegenden mathematischen Begriffen und Techniken vertraut machen.

- Logik: Syntax, Semantik, Beweise
- Mengen, Abbildungen, Relationen
- Induktion/Rekursion
- Modellierung
- Graphen und Bäume
- Reelle und komplexe Zahlen

Wir werden diese Themen jeweils mit Beispielen aus der Informatik bzw. Data Science illustrieren.

Anmerkung zum Lernen (Mathematik $\neq$ Rechnen)**.** Die ersten Schritte in der Mathematik sind erfahrungsgemäß herausfordernd und verlaufen möglicherweise nicht wie erwartet.

Mathematik ist *nicht* dasselbe wie „Rechnen“. Der Umgang mit verschiedenen Zahlbereichen und konkrete Berechnungen sind nur ein Teil der Mathematik. Der Schwerpunkt der Mathematik liegt im präzisen Umgang mit abstrakten Strukturen aller Art und in den sorgfältigen Begründungen aller Behauptungen. Insbesondere sind Vorkenntnisse aus der Schulmathematik (wie z.B. das Beherrschen der in der Schule vermittelten Rechentechniken oder ein grundlegendes Verständnis von Funktionen und Geometrie) hilfreich, aber ausschlaggebend für den Erfolg sind eher andere Fähigkeiten:

Wie das Erlernen einer Fremdsprache, eines Instruments oder einer herausfordernden Passage in einem Computerspiel erfordert das Erlernen von Mathematik

- Offenheit für neue Begriffe und Methoden (insbesondere auch die Offenheit, zu akzeptieren, dass Mathematik und Schulmathematik nicht so eng verwandt sind wie evtl. erwartet),
- präzises Denken,
- Durchhaltevermögen und den Willen, regelmäßig zu üben.

Die Versuchung ist groß, das Üben durch externe Hilfen (z.B. Foren oder künstliche „Intelligenz“) abzukürzen oder gar zu eliminieren. Diese Vorlesung und der zugehörige Übungsbetrieb sind eine Gelegenheit für *Sie*, die Begriffe und Techniken zu erlernen – und nicht, ein anderes System oder andere Personen dafür zu trainieren, diese Herausforderungen für Sie zu lösen. Spätere Aufgaben im Studium und in der praktischen Umsetzung werden auf diesen *Fähigkeiten* aufbauen und nicht auf konkreten Lösungen zu einzelnen konkreten Übungsaufgaben.

Anmerkung zum Lernen (Skript)**.** Dieses Vorlesungsskript dokumentiert den Fortschritt dieser Vorlesung, die besprochenen Themen und zusätzliches optionales Material. Der Besuch der Vorlesung erleichtert es, die Entwicklung der Begriffe, Beweise, etc. nachzuvollziehen. Zusätzliches Material wird in GRIPS und auf der Vorlesungshomepage bereitgestellt:

https://loeh.app.ur.de/teaching/fids_ws2324

Die Anwendungsabschnitte und Ausblicke sind *nicht* prüfungsrelevant, sondern dienen der Allgemeinbildung.

Anmerkung zum Lernen (Fingerübungen)**.** Dieses Vorlesungsskript enthält einige Selbsttests und Fingerübungen, deren Feedback teilweise in die PDF-Datei integriert ist. Diese Funktionalität beruht auf PDF Layers (nicht auf

JavaScript) und wird von vielen PDF-Viewern unterstützt, z.B. Acrobat Reader, Evince, Foxit Reader, Okular, ... Einfacher Test, ob das funktioniert: Haben Sie auf "Nein" gedrückt?

Ja Nein Nein, Sie haben nicht getan, was Sie behauptet haben

Sie sollten natürlich erst dann die Hinweise und Antworten ansehen, wenn Sie bereits über das Problem nachgedacht haben; man kann schließlich nie wissen, was passiert, wenn man voreilig auf irgendeinen Knopf drückt.

GROAAARR!

Literaturaufgabe (Bibliothek)**.** Wie finden Sie im Regensburger Katalog und in der Teilbibliothek Mathematik Literatur über die Themen der Vorlesung? Welche weiteren Informationsquellen könnten nützlich sein?

Zusätzliche Anregungen finden Sie auf S. xi.

Konvention. Die Menge $\mathbb{N}$ der natürlichen Zahlen enthält 0.

1

Logik: Syntax und Semantik

Die mathematische Logik beschreibt die „Spielregeln“, auf denen die Mathematik basiert; die Mengenlehre beschreibt das „Spielfeld“ bzw. die grundlegenden Bausteine, aus denen mathematische Objekte konstruiert werden. Der stringente simultane Aufbau von Logik und Mengenlehre als Grundlage der modernen Mathematik ist zu aufwendig, um zu Beginn des Studiums im Detail ausgeführt zu werden. Wir werden uns daher auf ein paar Einblicke mit den für den mathematischen/informatischen Alltag wichtigsten Punkten beschränken.

Die mathematische Logik beschäftigt sich mit den folgenden (miteinander zusammenhängenden) Fragen:

- Wie kann man die mathematische Sprache formalisieren?
- Was ist eine „wahre“ mathematische Aussage?
- Was ist ein Beweis?
- Was kann man beweisen? Gibt es Grenzen der Beweisbarkeit?
- Kann man beweisen, dass die Mathematik widerspruchsfrei ist?

Wir folgen dem allgemeinen Prinzip, mit einfachen Teilaspekten zu beginnen und dann Schritt für Schritt daraus komplexere Strukturen und Theorien aufzubauen („divide and conquer“).

In diesem Kapitel führen wir die logische Sprache ein; in Kapitel 2 werden wir uns mit Beweisen beschäftigen. Mengen und Abbildungen werden in Kapitel 3 behandelt.

Überblick über dieses Kapitel.

1.1 Aussagenlogik

Die Aussagenlogik ist ein einfaches logisches System, das die Grundlagen des logischen Denkens formalisiert. Aussagenlogik besteht aus

- einer *syntaktischen Ebene* (Wie dürfen Aussagen aussehen?) und
- einer *semantischen Ebene* (Ist eine Aussage „wahr“?).

Wir beschreiben diese Ebenen im folgenden etwas detaillierter.

1.1.1 Syntax

Zunächst definieren wir aussagenlogische Formeln – nach dem „divide and conquer“-Prinzip. Ausgehend von *aussagenlogischen Variablen* (das sind einfach Symbole, z.B. Großbuchstaben) erklären wir, wie man Schritt für Schritt kompliziertere Formeln zusammensetzen kann:

Definition 1.1.1 (Syntax aussagenlogischer Formeln)**.**

- Aussagenlogische Variablen sind aussagenlogische Formeln.
- Sind A und B aussagenlogische Formeln, so auch

$$(\neg A), \quad (A \wedge B), \quad (A \vee B), \quad (A \Longrightarrow B), \quad (A \Longleftrightarrow B).$$

- Keine weiteren Symbolketten sind aussagenlogische Formeln.

[Falls keine Missverständnisse möglich sind, setzt man zur besseren Lesbarkeit manchmal Klammern etwas freizügiger.]

Beispiel 1.1.2. Sind A und B aussagenlogische Variablen, so ist die Symbolkette $\big((A \vee B) \wedge (\neg A)\big)$ eine aussagenlogische Formel. Die Struktur dieser Formel können wir in einem sogenannten Syntaxbaum (Abbildung 1.1) darstellen.

Fingerübung 1.1.3. Seien A, B, C aussagenlogische Variablen. Welche der folgenden Symbolketten sind aussagenlogische Formeln?

- $\big(\neg(\neg A)\big)$ Ja Nein Die korrekte Antwort ist „Ja“. ×
- $\wedge A$ Ja Nein Die korrekte Antwort ist „Nein“. ×
- $\big(A \Longrightarrow (\neg A)\big)$ Ja Nein Die korrekte Antwort ist „Ja“. ×
- $\big(A \Longrightarrow (B \Longrightarrow C)\big)$ Ja Nein Die korrekte Antwort ist „Ja“. ×
- $A \wedge \wedge B$ Ja Nein Die korrekte Antwort ist „Nein“. ×

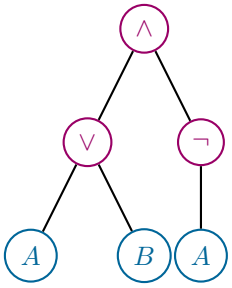


Abbildung 1.1.: Syntaxbaum zu $\big((A \vee B) \wedge (\neg A)\big)$

Anmerkung zum Lernen. Definitionen sind das Vokabular der Mathematik. Um mathematische Inhalte zu verstehen, ist es unerlässlich, sich alle in den Vorlesungen behandelten Definitionen genau zu merken. Sie sollten daher schon jetzt beginnen, sich systematisch während/nach jeder Vorlesung, die neu eingeführten Begriffe einzuprägen (notfalls mit Karteikarten)!

Anmerkung zum Lernen. Bei der Nachbereitung der Vorlesung helfen die folgenden Fragen:

- Verstehe ich den Wortlaut der Definition?
- Verstehe ich die Beispiele?
- Kann ich selbst weitere, ähnliche, Beispiele erzeugen?
- Verstehe ich den Zweck der Definition?
- Könnte ich die Definitionen/Beispiele erklären? (Ausprobieren!)

1.1.2 Semantik

Bisher haben wir nur geklärt, wie aussagenlogische Formeln aussehen dürfen. Als nächsten Schritt beschreiben wir, wie aussagenlogische Formeln interpretiert werden können:

Definition 1.1.4 (klassische Semantik aussagenlogischer Formeln)**.**

- Variablen können mit den Wahrheitswerten w („wahr“) bzw. f („falsch“) belegt werden.
- Belegen wir alle in einer aussagenlogischen Formel vorkommenden Variablen mit w bzw. f (wobei verschiedene Auftreten derselben Variablen in einer Formel denselben Wert erhalten müssen), so erhalten wir einen Wahrheitswert, indem wir Schritt für Schritt die semantischen Regeln (sogenannte *Wahrheitstafeln*) aus Abbildung 1.2 anwenden.

A	$\neg A$
w	f
f	w

A	B	$A \wedge B$	$A \vee B$	$A \Longrightarrow B$	$A \Longleftrightarrow B$
w	w	w	w	w	w
w	f	f	w	f	f
f	w	f	w	w	f
f	f	f	f	w	w

Abbildung 1.2.: Die grundlegenden Wahrheitstafeln

Bemerkung 1.1.5 (Anschauung der Wahrheitstafeln)**.** Die klassische Semantik kann man sich veranschaulichen, indem man aussagenlogische Variablen durch deutsche Sätze und die logischen Operatoren wie folgt ersetzt:

- $\neg$: „nicht“
- $\wedge$: „und“
- $\vee$: „oder“ (*kein* exklusives „oder“!)
- $\Longrightarrow$: „wenn ..., dann ...“; „impliziert“
- $\Longleftrightarrow$: „gilt genau dann, wenn“; „ist äquivalent zu“

Zum Beispiel erhalten wir durch

- A: Die Erde ist eine Scheibe.
- B: Regensburg liegt an der Donau.

die folgenden Aussagen und zugehörigen Übersetzungen:

- $A \Longrightarrow B$: Wenn die Erde eine Scheibe ist, dann liegt Regensburg an der Donau.
- $\neg B \Longrightarrow \neg A$: Wenn Regensburg nicht an der Donau liegt, dann ist die Erde keine Scheibe.

Den Wahrheitswert der zusammengesetzten Aussagen erhalten wir mit den Wahrheitstafeln aus den Wahrheitswerten der eingesetzten Aussagen.

Beispiel 1.1.6. Werden in der aussagenlogischen Formel $(A \vee B) \wedge (\neg A)$ die Variablen A und B mit f bzw. w belegt, so erhalten wir für $(A \vee B) \wedge (\neg A)$ Schritt für Schritt den folgenden Wahrheitswert:

- $A \vee B$: erhält den Wert w (nach der semantischen Regel für $\vee$)
- $\neg A$: erhält den Wert w (nach der semantischen Regel für $\neg$)
- $(A \vee B) \wedge (\neg A)$: erhält den Wert w (nach der semantischen Regel für $\wedge$ und den obigen Teilergebnissen für $A \vee B$ bzw. $\neg A$)

Man kann solche Berechnungen auch durchführen, indem man Syntaxbäume (Beispiel 1.1.2) von „unten“ nach „oben“ Schritt für Schritt mit den entsprechenden Werten dekoriert.

Definition 1.1.7 (Tautologie)**.** Eine aussagenlogische Formel A ist eine *(aussagenlogische) Tautologie*, wenn sich unter allen möglichen w/f-Belegungen aller auftretenden aussagenlogischen Variablen in A der Wert w ergibt.

Beispiel 1.1.8 (wichtige Tautologien)**.** Für alle aussagenlogischen Variablen A, B, C sind die folgenden Formeln aussagenlogische Tautologien (dies kann z.B. über entsprechende Wahrheitstafeln nachgewiesen werden; Übungsaufgabe):

$$
\begin{aligned}
A \vee (\neg A) &&& \text{tertium non datur}\\
(A \wedge B) &\iff (B \wedge A) && \wedge\text{-Kommutativität}\\
(A \vee B) &\iff (B \vee A) && \vee\text{-Kommutativität}\\
(A \iff B) &\iff (B \iff A) && \iff\text{-Kommutativität}\\
\neg(A \wedge B) &\iff \big((\neg A) \vee (\neg B)\big) && \text{de Morgansche Regeln}\\
\neg(A \vee B) &\iff \big((\neg A) \wedge (\neg B)\big) && \text{de Morgansche Regeln}\\
A &\implies \big(B \implies (A \wedge B)\big) && \wedge\text{-Introduktion}\\
(A \wedge B) &\implies A && \wedge\text{-Elimination}\\
A &\implies (A \vee B) && \vee\text{-Introduktion}\\
\big((A \implies C) \wedge (B \implies C)\big) &\implies \big((A \vee B) \implies C\big) && \vee\text{-Elimination}\\
\big((A \implies B) \wedge (B \implies A)\big) &\iff (A \iff B) && \iff\text{-Intro/-Elim}\\
\big((A \implies B) \wedge (B \implies C)\big) &\implies (A \implies C) && \implies\text{-Transitivität}\\
\big(A \implies (B \wedge \neg B)\big) &\implies (\neg A) && \text{reductio ad absurdum}\\
(A \implies B) &\iff (\neg B \implies \neg A) && \text{Kontraposition}\\
(A \implies B) &\iff \big((\neg A) \vee B\big) && \implies\text{-Elim/-Intro}\\
\big((A \wedge (\neg B) \implies (C \wedge (\neg C))\big) &\implies (A \implies B) && \text{Widerspruchsbeweis}
\end{aligned}
$$

Tautologien sind die Grundlage für viele Beweisstrukturen (Kapitel 2.2).

Caveat 1.1.9 (Umdrehen der Implikationsrichtung)**.** Sind A und B aussagenlogische Variablen, so ist $(A \implies B) \iff (B \implies A)$ *keine* aussangelogische Tautologie! Dies ist in umgangssprachlichen Interpretationen auch sofort ersichtlich – „wenn“ und „dann“ können im allgemeinen *nicht* vertauscht werden! Zum Beispiel kann man das mit den folgenden Sätzen ausprobieren:

- A: Der Mond fällt auf die Erde.
- B: Es gibt Straßenschäden.

Formal können wir dies einsehen, indem wir die zugehörigen Wahrheitstafeln betrachten: es genügt, eine Belegung der Variablen zu finden, die nicht zum Gesamtwert w führt (Abbildung 1.3).

A	B	$A \Longrightarrow B$	$B \Longrightarrow A$	$(A \Longrightarrow B) \Longleftrightarrow (B \Longrightarrow A)$
w	w			
w	f	f	w	f **(!)**
f	w			
f	f			

Abbildung 1.3.: „Wenn“ und „dann“ dürfen nicht vertauscht werden!

Caveat 1.1.10. Es gibt Zweige der Mathematik/Informatik, in denen andere Semantiken verwendet werden. In solchen Semantiken gilt insbesondere *tertium non datur* normalerweise *nicht*; insbesondere steht in solchen Kontexten der Widerspruchsbeweis (Kapitel 2.2) nicht in derselben Form als Beweistechnik zur Verfügung! Konkrete Beispiele sind:

- Dreiwertige Logiken, in denen der dritte Wert etwas „unbekanntes“ repräsentiert. Eine solche dreiwertige Logik wird etwa in der Datenbanksprache SQL verwendet, um mit NULL geeignet umgehen zu können [29].
- Konstruktive Logiken. Solche Logiken werden in manchen Beweisassistenten verwendet, z.B. in Coq [6].

1.2 Quantorenlogik

Wir erweitern und verfeinern die Aussagenlogik, um typische All- und Existenzaussagen besser formalisieren zu können. Zum Beispiel würden wir gerne Aussagen wie

> Für jede natürliche Zahl x gilt $x + 0 = x$.

formulieren und mit Wahrheitswerten belegen können.

Wie im Fall der Aussagenlogik besteht auch die Quantorenlogik aus einer syntaktischen und einer semantischen Ebene. Da die exakten Definitionen jedoch aufwendig sind [11, 8], begnügen wir uns hier mit einer vereinfachten, pragmatischen Darstellung.

1.2.1 Syntax

„Definition“ 1.2.1 (Syntax quantorenlogischer Aussagen)**.** Sei T eine mathematische Sprache/Theorie (z.B. die Sprache/Theorie der natürlichen Zahlen).

- „Atomare Aussagen“ aus der Theorie T sind quantorenlogische Aussagen über T; diese dürfen auch Variablen enthalten.

- Sind A und B quantorenlogische Aussagen über T, so auch

$$(\neg A), \quad (A \wedge B), \quad (A \vee B), \quad (A \Longrightarrow B), \quad (A \Longleftrightarrow B).$$

- Ist A eine quantorenlogische Aussage über T und ist x eine (in A „freie“) Variable, so sind auch

$$\big(\forall_x \quad A(x)\big) \qquad \text{bzw.} \qquad \big(\exists_x \quad A(x)\big)$$

  quantorenlogische Aussagen über T.

  Man bezeichnet $\forall$ als *Allquantor* und $\exists$ als *Existenzquantor*.

- Keine weiteren Symbolketten sind quantorenlogische Aussagen über T.

[Falls keine Missverständnisse möglich sind, setzt man zur besseren Lesbarkeit manchmal Klammern etwas freizügiger.]

Wir gehen kurz auf die hier nicht präzise eingführten Begriffe „atomare Aussagen“ bzw. „freie Variablen“ im Beispiel der Sprache/Theorie der natürlichen Zahlen ein:

Beispiel 1.2.2. In der Theorie der natürlichen Zahlen gibt es die Symbole 0, 1, $+$, $=$ und Variablen, aus denen man atomare Aussagen bilden kann; dabei ist zu berücksichtigen, dass diese Symbole nur auf bestimmte Weisen kombiniert werden können. Zum Beispiel sind $0+1=1$ und $x+0=x$ atomare Aussagen aus der Theorie der natürlichen Zahlen, aber $0++0$ und $x+0$ sind *keine* zulässigen atomaren Aussagen.

Die Variable x ist frei in der Aussage $x+0=x$, da sie durch keinen Quantor gebunden ist. Daher ist

$$\big(\forall_x \quad x+0=x\big)$$

eine quantorenlogische Aussage über T.

Im Gegensatz dazu ist die Variable x *nicht* frei in der quantorenlogischen Aussage $\big(\forall_x \quad x+0=x\big)$; somit ist $\big(\exists_x \quad \big(\forall_x \quad x+0=x\big)\big)$ *keine* zulässige quantorenlogische Aussage.

1.2.2 Semantik

„Definition“ 1.2.3 (klassische Semantik quantorenlogischer Aussagen)**.** Sei T eine mathematische Sprache/Theorie.

- Variablen können mit „Werten aus T“ belegt werden.

- Belegen wir alle in einer atomaren Aussage in T vorkommenden freien Variablen mit Werten aus T, so erhält die Aussage unter dieser Belegung genau dann den Wert w, wenn sie in T gilt.

- Die Semantik von $\neg, \wedge, \vee, \Longrightarrow, \Longleftrightarrow$ für quantorenlogische Aussagen über T wird analog zur Aussagenlogik definiert (sofern die Teilaussagen bereits w/f-Werte erhalten haben).
- Zusätzlich gelten die folgenden Interpretationen:
  - $\forall_x \quad A(x)$ erhält genau dann den Wert w, wenn: Für alle möglichen T-Belegungen für x hat $A(x)$ den Wert w.
  - $\exists_x \quad A(x)$ erhält genau dann den Wert w, wenn: Es existiert (mindestens) eine T-Belegung für x, für die $A(x)$ der Wert w hat.

Man kann quantorenlogische Aussagen über T im allgemeinen nur dann auf einen Wahrheitswert reduzieren, wenn sie keine freien Variablen enthalten.

Beispiel 1.2.4. Wir betrachten wie in Beispiel 1.2.2 die Sprache/Theorie der natürlichen Zahlen.

- Die Aussage $\forall_x \quad x + 0 = x$ ist ? wahr;
- im Gegensatz dazu ist die Aussage $\exists_x \quad \forall_y \quad x = y + 1$? falsch.
- Der Aussage $x + 0 = 1$ können wir *keinen* Wahrheitswert zuordnen, da diese Aussage eine freie Variable (nämlich x) enthält.

Caveat 1.2.5 (Reihenfolge der Quantoren)**.** Im allgemeinen darf die Reihenfolge von Quantoren *nicht* vertauscht werden! Man betrachte dazu zum Beispiel die quantorenlogischen Aussagen

$$\forall_x \quad \exists_y \quad A(x, y) \qquad \text{bzw.} \qquad \exists_y \quad \forall_x \quad A(x, y),$$

wobei $A(x, y)$ bedeute, dass x ein Mensch ist, y eine Blutgruppe ist und x Blutgruppe y hat.

Caveat 1.2.6 (Quantoren gehören nach vorne!)**.** Formeln wie „$A(x) \quad \forall_x$“ oder gar „$\exists_x \quad A(x, y) \quad \forall_y$“ ergeben keinen Sinn (selbst wenn es die deutsche Sprache manchmal nahelegt . . .)!

1.3 Anwendung*: Logische Kontrollstrukturen

Programmiersprachen besitzen im Normalfall einen Datentyp für Wahrheitswerte: „Booleans“. Die Wahrheitswerte werden dabei oft als true (bei uns: w) bzw. false (bei uns: f) bezeichnet. Zusätzlich werden logische Operatoren (wie “$\wedge$”, etc.) auf Booleans zur Verfügung gestellt. Konstrollstrukturen wie

$$\text{if } \langle \text{Boolean} \rangle \text{ then } \ldots \text{ else } \ldots$$

ermöglichen es, den Programmfluss zu steuern.

Beispiel 1.3.1 (Kontrollstrukturen im Linux-Kernel)**.** Wir betrachten den folgenden Auszug des Linux-Kernels (in C):

https://github.com/torvalds/linux/blob/master/kernel/debug/debug_core.c

Boolesche Werte treten hier sowohl als Ergebnisse von Vergleichsoperationen als auch als „Schalter“ (*flags*) auf.

```
int dbg_set_sw_break(unsigned long addr)
{
   int err = kgdb_validate_break_address(addr);
   int breakno = -1;
   int i;

   if (err)
      return err;

   for (i = 0; i < KGDB_MAX_BREAKPOINTS; i++) {
      if ((kgdb_break[i].state == BP_SET) &&
                (kgdb_break[i].bpt_addr == addr))
         return -EEXIST;
   }
   for (i = 0; i < KGDB_MAX_BREAKPOINTS; i++) {
      if (kgdb_break[i].state == BP_REMOVED &&
                kgdb_break[i].bpt_addr == addr) {
         breakno = i;
         break;
      }
   }

   if (breakno == -1) {
      for (i = 0; i < KGDB_MAX_BREAKPOINTS; i++) {
         if (kgdb_break[i].state == BP_UNDEFINED) {
            breakno = i;
            break;
         }
      }
   }

   if (breakno == -1)
      return -E2BIG;

   kgdb_break[breakno].state = BP_SET;
   kgdb_break[breakno].type = BP_BREAKPOINT;
   kgdb_break[breakno].bpt_addr = addr;

   return 0;
}
```

1.4 Ausblick*: Implementation von Semantiken

Semantiken für aussagenlogische Formeln lassen sich leicht implementieren. Wir verwenden Lean 4 [3, 20, 21] und nutzen induktive Datentypen sowie darauf definierte rekursive Funktionen, um Formeln und ihre Semantik „Schritt für Schritt“ zu definieren.

Syntax aussagenlogischer Formeln

```
inductive Expr where
| eVar : Nat -> Expr  -- we enumerate the different variables
    by natural numbers
| eNot : Expr -> Expr
| eAnd : Expr -> Expr -> Expr
| eOr  : Expr -> Expr -> Expr
| eImp : Expr -> Expr -> Expr
| eEqu : Expr -> Expr -> Expr
deriving Repr -- so that we can display values of this type
open Expr
```

Der Vorteil einer solchen Implementierung ist, dass wir sie an Beispielen austesten können:

```
def A := eVar 0
def B := eVar 1

#check eAnd (eOr A B) (eNot A) -- indeed an Expr
#check eAnd (eAnd A B)         -- (!)
-- #check And (And And)         -- uh-oh
```

Klassische Semantik aussagenlogischer Formeln

Für die klassische Semantik definieren wir klassische Booleans und darauf die gewöhnlichen logischen Operationen durch ihre Wahrheitstafeln. Selbstverständlich hätte man auch Lean-Booleans und die bereits implementierten logischen Operationen verwenden können. Die ausführlichere Vorgehensweise illustriert, wie man allgemein vorgehen kann.

```
inductive CBool where
| cTrue  : CBool
| cFalse : CBool
deriving Repr
open CBool
```

```
-- The standard boolean operations:
def lnot (a : CBool) : CBool
:= match a with
| cTrue  => cFalse
| cFalse => cTrue

def lor (a : CBool) (b : CBool) : CBool
:= match (a,b) with
| (cTrue,  cTrue)  => cTrue
| (cTrue,  cFalse) => cTrue
| (cFalse, cTrue)  => cTrue
| (cFalse, cFalse) => cFalse

def land (a : CBool) (b : CBool) : CBool
:= match (a,b) with
| (cTrue,  cTrue)  => cTrue
| (cTrue,  cFalse) => cFalse
| (cFalse, cTrue)  => cFalse
| (cFalse, cFalse) => cFalse

def limp (a : CBool) (b : CBool) : CBool
:= match (a,b) with
| (cTrue,  cTrue)  => cTrue
| (cTrue,  cFalse) => cFalse
| (cFalse, cTrue)  => cTrue
| (cFalse, cFalse) => cTrue

def lequ (a : CBool) (b : CBool) : CBool
:= match (a,b) with
| (cTrue,  cTrue)  => cTrue
| (cTrue,  cFalse) => cFalse
| (cFalse, cTrue)  => cFalse
| (cFalse, cFalse) => cTrue

-- A valuation assigns to each (index of a) variable a
    classical boolean
def cVal := Nat -> CBool

-- The classical semantics of propositional formulas
def cSemantics (v : cVal)  (e : Expr) : CBool
:= match e with
| eVar n   => v n
| eNot a   => lnot (cSemantics v a)
| eAnd a b => land (cSemantics v a) (cSemantics v b)
```

```
| eOr  a b => lor  (cSemantics v a) (cSemantics v b)
| eImp a b => limp (cSemantics v a) (cSemantics v b)
| eEqu a b => lequ (cSemantics v a) (cSemantics v b)

-- Examples
def ft : cVal
:= λ n
=> match n with
| 0 => cFalse
| 1 => cTrue
| _ => cTrue

def someExpr : Expr
:= (eAnd (eOr A B)
         (eNot A))

#check cSemantics ft someExpr
#eval cSemantics ft someExpr
```

Eine bizarre Semantik

Zusätzlich zur klassischen Semantik können wir auch bizarre Semantiken implementieren, z.B. mit Werten in den natürlichen Zahlen und semantischen Regeln, deren Sinn sich womöglich nie erschließen wird:

```
def sVal := Nat -> Nat

def sSemantics (v : sVal) (e : Expr) : Nat
:= match e with
| eVar n   => v n
| eNot a   => 42
| eAnd a b => (sSemantics v a) + (sSemantics v b)
| eOr  a b => (sSemantics v a) * 2023
| eImp a b => (sSemantics v a) + (sSemantics v b)
| eEqu a b => (sSemantics v a)

def incVal : sVal
:= λ n => n + 2

#check sSemantics incVal someExpr
#eval sSemantics incVal someExpr   -- 4088
```

2

Logik: Beweise

Wir beschreiben den klassischen Beweiskalkül der Mathematik; dieser ist eine Formalisierung der gängigen logischen Schlussweisen und erkälrt, welche Beweisschritte/Argumente zulässig sind.

Wir geben zunächst die formale Definition und zeigen dann an Beispielen, wie man in der Praxis damit umgehen kann. Insbesondere erklären wir, wie aussagenlogische Tautologien zu elementaren Beweisstrukturen führen.

Überblick über dieses Kapitel.

2.1 Der Beweisbegriff

2.1.1 Wozu Beweise?

Jede Behauptung ist mit der Pflicht, eine Begründung zu liefern, verbunden. In der Mathematik werden solche Begründungen durch Beweise gegeben; dabei ist auch der Begriff des Beweises mathematisch präzise definiert. Der Beweisbegriff ist so angelegt, dass bewiesene Aussagen in der entsprechenden Semantik „wahr“ sind (Satz 2.1.2).

2.1.2 Ein klassischer Beweiskalkül

Definition 2.1.1 (Beweis)**.** Sei T eine mathematische Sprache/Theorie (z.B. die Sprache der natürlichen Zahlen), seien V quantorenlogische Aussagen über T und sei B eine quantorenlogische Aussage über T. Ein *Beweis* von B („Behauptung“) aus V („Axiome/Voraussetzungen“) über T ist eine endliche Folge von quantorenlogischen Aussagen über T mit den folgenden Eigenschaften: Jede dieser Aussagen ist

- eine Aussage aus V,
- ein identitätslogisches Axiom (darauf wird hier nicht eingegangen),
- ein quantorenlogisches Axiom,
- eine aussagenlogische Tautologie über T

oder man erhält sie aus vorherigen Aussagen des Beweises mit Hilfe

- des *Modus Ponens* oder
- der Generalisierungsregel

und die letzte so entstandene Aussage ist B. Dabei gilt:

- Die *quantorenlogischen Axiome* sind (für alle quantorenlogischen Formeln A, A' über T):
  - $\big(\forall_x \quad A(x)\big) \Longrightarrow A(t)$,
    falls t ein „Term“ in T ohne ungewollte Variablenbindungen ist;
  - $\forall_x \quad \big(A \Longrightarrow A'(x)\big) \Longrightarrow \big(A \Longrightarrow (\forall_x \quad A'(x))\big)$,
    falls x nicht frei in A vorkommt;
  - $\big(\exists_x \quad A(x)\big) \Longleftrightarrow \big(\neg(\forall_x \quad \neg A(x))\big)$

- Eine *aussagenlogische Tautologie über T* ist eine aussagenlogische Tautologie, in der alle aussagenlogischen Variablen durch quantorenlogische Aussagen über T ersetzt werden.
- *Modus Ponens* ist die folgende Schlussregel: Enthalten die vorherigen Aussagen eine Aussage der Form $A \Longrightarrow A'$ und die Aussage A, so kann man A' zum Beweis hinzufügen.
- Die *Generalisierungsregel* besagt: Enthalten die vorherigen Aussagen eine Aussage A, in der die Variable x nicht gebunden auftritt, so kann man $\forall_x \quad A(x)$ zum Beweis hinzufügen.

2.1.3 Korrektheit und Vollständigkeit

Satz 2.1.2 (Korrektheit)**.** *Sei T eine mathematische Sprache/Theorie, seien $V_1, \ldots, V_m$ und B quantorenlogische Aussagen über T (ohne freie Variablen) und es gebe einen Beweis (im Sinne von Definition 2.1.1) von B aus $V_1, \ldots, V_m$ über T. Dann gibt es auch einen Beweis von $\big(V_1 \wedge (V_2 \wedge \cdots \wedge V_m)\big) \Longrightarrow B$ (ohne weitere Voraussetzungen) über T und*

$$\big(V_1 \wedge (V_2 \wedge \cdots \wedge V_m)\big) \Longrightarrow B$$

erhält den Wert w bezüglich der Semantik über T aus „Definition“ 1.2.3.

Den Korrektheitssatz kann man beweisen, indem man nachprüft, dass die Behauptung in den Basisfällen des Beweiskalküls erfüllt ist und dass diese Eigenschaft in allen Schlussregeln erhalten bleibt [8, Chapter 4].

Bemerkung 2.1.3 (Vollständigkeit)**.** In der sogenannten Logik erster Stufe gilt auch die Umkehrung des Korrektheitssatzes: Quantorenlogische Aussagen, die in „allen Modellen“ der gegebenen Axiome wahr sind, können aus den Axiomen im Sinne von Definition 2.1.1 bewiesen werden [8, Chapter 4].

2.1.4 Ein Beispiel

Beispiel 2.1.4 (ein erster Beweis)**.** Wir betrachten das folgende Axiomensystem über die Theorie der Pinguine:

- *Axiome/Voraussetzungen:*
  - ① Pinguine, die bellen, beißen nicht.
    D.h. es gilt $\forall_x \quad A(x)$, wobei

    $$A(x) := \big((x \text{ ist Pinguin}) \Longrightarrow \big((x \text{ bellt}) \Longrightarrow \neg(x \text{ beißt})\big)\big).$$

  - ② Tux ist ein Pinguin.
  - ③ Tux beißt.

- *Behauptung:* Tux bellt nicht.
- *Beweis.* Wir erhalten Schritt für Schritt:
  - $\forall_x \quad A(x)$
    wegen: Axiom ①
  - $\big(\forall_x \quad A(x)\big) \Longrightarrow A(\text{Tux})$
    wegen: quantorenlogisches Axiom
  - $A(\text{Tux})$
    wegen: Modus Ponens
    Ausformuliert bedeutet dies:

    $$(\text{Tux ist Pinguin}) \Longrightarrow \big((\text{Tux bellt}) \Longrightarrow \neg(\text{Tux beißt})\big)$$

  - Tux ist ein Pinguin
    wegen: Axiom ②
  - $(\text{Tux bellt}) \Longrightarrow \neg(\text{Tux beißt})$
    wegen: Modus Ponens
  - $\big((\text{Tux bellt}) \Longrightarrow \neg(\text{Tux beißt})\big) \Longrightarrow \big((\text{Tux beißt}) \Longrightarrow \neg(\text{Tux bellt})\big)$
    wegen: Tautologie: $(B \Longrightarrow \neg C) \Longrightarrow (C \Longrightarrow \neg B)$
  - $(\text{Tux beißt}) \Longrightarrow \neg(\text{Tux bellt})$
    wegen: Modus Ponens
  - Tux beißt
    wegen: Axiom ③
  - $\neg(\text{Tux bellt})$
    wegen: Modus Ponens □

Im Normalfall werden Beweise nicht in dieser Form aufgeschrieben, sondern sprachlich poliert, vereinfacht und strukturiert:

- *Beweis.* Da Tux nach Axiom ② ein Pinguin ist, erhalten wir mit Axiom ①:

  Wenn Tux bellt, beißt Tux nicht.

  Mit Kontraposition (und *tertium non datur*) folgt daraus:

  Wenn Tux beißt, bellt Tux nicht.

  Da Tux nach Axiom ③ beißt, liefert dies, dass Tux nicht bellt. □

Der formale Zugang zu Beweisen ist jedoch nötig, um überhaupt einen belastbaren Beweisbegriff einführen zu können, und hat den zusätzlichen Vorteil, dass explizit formalisierte Beweise maschinell überprüft werden können (Kapitel 2.4).

Anmerkung zum Lernen. Bei jedem Beweis, dem Sie begegnen, sollten Sie kritisch hinterfragen, ob Sie wirklich alle Beweisschritte verstehen, ob der Beweis vollständig ist, und was die grundlegende Idee dahinter ist. Sobald Sie mehr Beweise kennen, sollten Sie außerdem überlegen, ob es sich um eine Beweistechnik handelt, die in gleicher oder ähnlicher Form bereits in einer anderen Situation verwendet wurde.

Literaturaufgabe. Lesen Sie *„Das ist o.B.d.A. trivial!“* von Beutelspacher [5].

2.2 Grundlegende Beweisstrukturen

Das wichtigste Beweisen ist Korrektheit, d.h., dass wirklich die Behauptung gezeigt wird. Interessant an einem Beweis ist die unterliegende Idee. Man sollte bei Beweisen denselben Prinzipien wie beim Programmieren folgen:

- Beweise sollten in sinnvolle Einheiten strukturiert werden.
- Argumente, die mehrfach vorkommen, sollten nicht wiederholt, sondern abstrahiert werden.
- Transparenz und Robustheit sind im Normalfall wichtiger als Kürze.

Bei der Darstellung sollte darauf geachtet werden, dass Voraussetzungen, Behauptung und der eigentliche Beweis deutlich erkennbar sind. Voraussetzungen und Annahmen werden im Konjunktiv formuliert („Sei . . . “, „Es gelte . . . “); Behauptungen werden im Indikativ formuliert und zumeist mit „Dann . . . “ o.ä. eingeleitet. Es gibt aber viele sprachliche Varianten.

Beispiel 2.2.1. Was sind die Voraussetzungen bzw. was ist die Behauptung?

> Sei $(K, +, \cdot)$ ein Körper. Dann gilt für alle $x \in K$, dass $-x = (-1) \cdot x$.

Voraussetzung: Sei $(K, +, \cdot)$ ein Körper.
Behauptung: Für alle $x \in K$ gilt $-x = (-1) \cdot x$.

2.2.1 Introduktion und Elimination

Logische Aussagen werden Schritt für Schritt aus einfachen Bausteinen zusammengesetzt. Insbesondere trifft dies auf Voraussetzungen und Behauptungen (sowie alle Zwischenschritte eines Beweises) zu. Introduktions- bzw. Eliminationstautologien liefern elementare Beweisstrategien, um

- zusammengesetzte Behauptungen zu beweisen (Introduktion) oder
- zusammengesetzte Voraussetzungen zu zerlegen (Elimination).

Eine Analyse der Struktur der vorliegenden Voraussetzungen und Behauptungen kann daher oft wertvolle Ideen für Beweisansätze nach dem „Baukastenprinzip“ liefern.

Beispiel 2.2.2. Da es sich bei der Behauptung in Beispiel 2.2.1 um eine $\forall$-quantifizierte Aussage handelt, könnte ein möglicher Beweis mit „Sei $x \in K$. Dann ...“ beginnen.

2.2.2 Zwischenschritte

Zur Unterteilung eines Beweises in Zwischenschritte können wir die folgende aussagenlogische Tautologie verwenden (wobei A, B, C aussagenlogische Variablen sind):

$$\big((A \Longrightarrow B) \wedge (B \Longrightarrow C)\big) \Longrightarrow (A \Longrightarrow C)$$

Anmerkung zum Lernen. Überprüfen Sie anhand der semantischen Regeln der Aussagenlogik (Wahrheitstafeln), dass es sich tatsächlich um eine aussagenlogische Tautologie handelt.

Beweisschema 2.2.3 (für Zwischenschritte)**.** Seien A und C quantorenlogische Aussagen über einer Sprache/Theorie T.

- *Behauptung.* Aus A folgt C.
- *Beweis.* Wir zeigen zunächst die folgende Zwischenbehauptung:
  - *Behauptung.* Wenn A gilt, gilt auch B.
  - *Beweis.* ... □

  Wir zeigen nun, dass aus B die Behauptung C folgt:

  ... □

Warum funktioniert dieses Beweisschema?

- Das Beweisschema liefert zunächst einen Beweis von $A \Longrightarrow B$.
- Ausserdem liefert der zweite Teil des Beweisschemas einen Beweis für die Implikation $B \Longrightarrow C$.
- Zweimalige Anwendung des Modus Ponens und der Tautologie für $\wedge$-Introduktion ergibt daher einen Beweis von $(A \Longrightarrow B) \wedge (B \Longrightarrow C)$.
- Wir können die obige aussagenlogische Tautologie ($\Longrightarrow$-Transitivität) auf unsere quantorenlogischen Aussagen A, B, C anwenden und erhalten somit einen Beweis von:

$$\big((A \Longrightarrow B) \wedge (B \Longrightarrow C)\big) \Longrightarrow (A \Longrightarrow C)$$

- Anwenden des Modus Ponens ergibt einen Beweis von $A \Longrightarrow C$.

Analog können wir auch für die folgenden Beweisstrukturen zeigen, dass sie Beweise im Sinne von Definition 2.1.1 liefern.

2.2.3 Äquivalenzen

Äquivalenzen können gezeigt werden, indem man den Beweis in Beweise der einzelnen Implikationen aufteilt. Für den Beweis der Äquivalenz zweier Aussagen beruht dies auf der folgenden aussagenlogischen Tautologie (wobei A, B aussagenlogische Variablen sind):

$$(A \Longleftrightarrow B) \Longleftrightarrow \big((A \Longrightarrow B) \wedge (B \Longrightarrow A)\big)$$

Anmerkung zum Lernen. Überprüfen Sie anhand der semantischen Regeln der Aussagenlogik (Wahrheitstafeln), dass es sich tatsächlich um eine aussagenlogische Tautologie handelt.

Beweisschema 2.2.4 (für Äquivalenzen)**.** Seien A und B quantorenlogische Aussagen über einer Sprache/Theorie T.

- *Behauptung.* Es sind A und B äquivalent.
- *Beweis.* Wir zeigen die beiden Implikationen einzeln:
  - Es gelte A. Dann folgt B, denn:

    ...
  - Umgekehrt gelte B. Dann folgt A, denn:

    ... □

Beispiel 2.2.5. Details zu den ganzen Zahlen finden sich in Kapitel 6.2.3.

- *Voraussetzung.* Sei x eine ganze Zahl.
- *Behauptung.* Dann sind äquivalent:
  1. Es gilt $x^2 - 2 \cdot x + 1 = 0$.
  2. Es gilt $x = 1$.
- *Beweis.* Wir zeigen die beiden Implikationen einzeln:
  - Es gelte $x^2 - 2 \cdot x + 1 = 0$. Mit den binomischen Formeln erhalten wir
    $$0 = x^2 - 2 \cdot x + 1 = (x-1)^2.$$
    Da die ganzen Zahlen nullteilerfrei sind, folgt $x - 1 = 0$, und damit $x = 1$.
  - Sei umgekehrt $x = 1$. Dann ist
    $$x^2 - 2 \cdot x + 1 = 1^2 - 2 \cdot 1 + 1 = 1 - 2 + 1 = 0.$$ □

2.2.4 Kontraposition

Das Beweisprinzip der *Kontraposition* beruht auf der folgenden aussagenlogischen Tautologie (wobei A, B aussagenlogische Variablen sind):

$$(A \Longrightarrow B) \Longleftrightarrow (\neg B \Longrightarrow \neg A)$$

Anmerkung zum Lernen. Überprüfen Sie anhand der semantischen Regeln der Aussagenlogik (Wahrheitstafeln), dass es sich tatsächlich um eine aussagenlogische Tautologie handelt.

Beweisschema 2.2.6 (für Kontraposition)**.** Seien A und B quantorenlogische Aussagen über einer Sprache/Theorie T.

- *Voraussetzung.* Es gelte A.
- *Behauptung.* Dann gilt B.
- *Beweis.* Wir zeigen die Behauptung durch Kontraposition, d.h. wir zeigen, dass $(\neg B) \Longrightarrow (\neg A)$ gilt:

  Es gelte $\neg B$.

  ...

  Somit folgt $\neg A$. □

Beispiel 2.2.7 (ungerade Quadrate)**.** Details zu natürlichen Zahlen finden sich in Kapitel 4.

- *Voraussetzung.* Sei n eine natürliche Zahl und es sei n^2 ungerade.
- *Behauptung.* Dann ist n ungerade.
- *Beweis.* Wir zeigen die Behauptung durch Kontraposition:

  Sei also n gerade, d.h. es gibt ein $k \in \mathbb{N}$ mit $n = 2 \cdot k$. Dann folgt

  $$n^2 = (2 \cdot k)^2 = 2 \cdot (k \cdot 2 \cdot k).$$

  Insbesondere ist n^2 gerade, was zu zeigen war. □

2.2.5 Reductio ad absurdum

Das Beweisprinzip *Reductio ad absurdum* ist fundamental für den Beweis von negierten Aussagen. Es beruht auf der folgenden aussagenlogischen Tautologie (wobei A, B aussagenlogische Variablen sind):

$$\big(A \Longrightarrow (B \wedge (\neg B))\big) \Longrightarrow (\neg A)$$

Anmerkung zum Lernen. Überprüfen Sie anhand der semantischen Regeln der Aussagenlogik (Wahrheitstafeln), dass es sich tatsächlich um eine aussagenlogische Tautologie handelt.

Beweisschema 2.2.8 (für *Reductio ad absurdum*)**.** Sei A eine quantorenlogische Aussagen über einer Sprache/Theorie T.

- *Behauptung.* Es gilt $\neg A$.
- *Beweis. Angenommen*, A würde gelten.

  ...

  [Ableitung eines Widerspruchs „$B \wedge (\neg B)$“]

  Dies ist ein Widerspruch. Also gilt $\neg A$. □

Beispiel 2.2.9 (Irrationalität von $\sqrt{2}$)**.** Details zu den reellen Zahlen finden sich in Kapitel 7.

- *Voraussetzung.* Sei x eine reelle Zahl mit $x^2 = 2$.
- *Behauptung.* Dann ist x nicht rational.
- *Beweis. Angenommen,* x wäre rational. Dann gäbe es ganze Zahlen m und n mit

  $$x = \frac{m}{n}.$$

  Ohne Einschränkung können wir annehmen, dass dieser Bruch vollständig gekürzt ist, dass also m und n keine gemeinsamen Teiler besitzen. Wegen $x^2 = 2$ folgt somit

  $$2 \cdot n^2 = x^2 \cdot n^2 = \left(\frac{m}{n}\right)^2 \cdot n^2 = \frac{m^2}{n^2} \cdot n^2 = m^2.$$

  Da 2 prim ist, ist daher 2 ein Teiler von m. Also existiert eine ganze Zahl k mit $m = 2 \cdot k$. Wir erhalten

  $$2 \cdot n^2 = m^2 = (2 \cdot k)^2 = 2 \cdot (2 \cdot k^2),$$

  und damit $n^2 = 2 \cdot k^2$. Wir wenden wiederum an, dass 2 prim ist und schließen, dass 2 auch ein Teiler von n ist. Insgesamt folgt, dass sowohl m als auch n von 2 geteilt werden. Dies steht jedoch im Widerspruch dazu, dass m und n keine gemeinsamen Teiler besitzen.

  Also ist x nicht rational. □

Bemerkung 2.2.10. Es gibt unterschiedliche Konventionen, an welcher Stelle im Prinzip der Reductio ad absurdum die Negation auftritt. In der klassischen Semantik und im klassischen Beweiskalkül sind all diese Varianten äquivalent. In anderen Situationen (z.B. in konstruktiven Logiken) können sich aber Unterschiede ergeben!

Caveat 2.2.11. Aus dem Nicht-Auffinden eines Widerspruchs kann nichts gefolgert werden! [19]

2.2.6 Echter Widerspruchsbeweis

Der *echte Widerspruchsbeweis* beruht auf der folgenden aussagenlogischen Tautologie (wobei A, B, C aussagenlogische Variablen sind):

$$\big((A \wedge (\neg B)) \Longrightarrow (C \wedge (\neg C))\big) \Longrightarrow (A \Longrightarrow B)$$

Anmerkung zum Lernen. Überprüfen Sie anhand der semantischen Regeln der Aussagenlogik (Wahrheitstafeln), dass es sich tatsächlich um eine aussagenlogische Tautologie handelt.

Beweisschema 2.2.12 (für einen echten Widerspruchsbeweis)**.** Seien A und B quantorenlogische Aussagen über einer Sprache/Theorie T.

- *Voraussetzung.* Es gelte A.
- *Behauptung.* Dann gilt B.
- *Beweis. Angenommen*, B würde nicht gelten.

  ...

  [Ableitung eines Widerspruchs „$C \wedge (\neg C)$“]

  Dies ist ein Widerspruch. Also ist die Annahme falsch. Somit folgt B. □

Beispiel 2.2.13 (Brouwerscher Fixpunktsatz)**.** Prototypische Beispiele für einen echten Widerspruchsbeweis sind die Standardbeweise des Brouwerschen Fixpunktsatzes [23, Corollary 1.3.25].

Bemerkung 2.2.14. Wenn möglich sollten Widerspruchsbeweise vermieden werden: Direkte Beweise sind oft klarer und liefern Argumente, die auch in ähnlichen Situationen wiederverwendet werden könnten. Ein Widerspruchsbeweis mag zunächst „naheliegender“ oder „leichter zu finden“ sein. Viele Widerspruchsbeweise können aber ohne Schwierigkeiten in transparentere und kürzere direkte Beweise umgewandelt werden – dies sollte immer geprüft werden!

Literaturaufgabe (Negation, Kontraposition, Widerspruch)**.** Lesen Sie den Artikel *Proof of negation and proof by contradiction* von Bauer [4].

2.2.7 Induktion

Strukturen, die „Schritt für Schritt“ aus kleineren Bausteinen zusammengesetzt werden, können mit einem weiteren Beweisprinzip behandelt werden: Induktion. Wir werden dieses (vor allem in der Informatik zentrale) Prinzip am Beispiel der natürlichen Zahlen ausführlich diskutieren (Kapitel 4).

2.3 Anwendung*: Formale Verifikation

Unter *formaler Verifikation* fasst man verschiedene Verfahren zusammen, die maschinell überprüfbare Beweise dafür liefern, dass gewisse Softwaresysteme relativ zu ihrer Spezifikation korrekt sind. All diese Verfahren bauen auf entsprechenden formalen Beweisbegriffen auf. Die zwei wichtigsten Arten von Verfahren sind:

- *Model-Checking.* Dieses Verfahren beruht auf einer systematischen Überprüfung aller möglichen Zustände/Abläufe.
- *Deduktive Verifikation.* Dieses Verfahren beruht auf formalen Beweisen gewisser Eigenschaften in Beweisassistenten.

Ein *Beweisassistent* ist eine Programmiersprache sowie ein zugehöriger Interpreter/Compiler, die es erlauben, mathematische Objekte und Sachverhalte zu formalisieren; dazu gehören Definitionen, Sätze, Beweise und Beispiele für die Theorie. Die Hauptaufgabe eines Beweisassistenten ist dabei nicht, Beweise zu *finden*, sondern zu *überprüfen*, ob Beweise korrekt sind. Beweisassistenten können also eine Verifikation für Korrektheit liefern (basierend auf der Annahme, dass der Beweisassistent selbst korrekt arbeitet).

Es gibt diverse Beweisassistenten, die breit eingesetzt werden, z.B. Coq, HOL Light, Isabelle, Lean, ...

Zum Beispiel wurde mithilfe von Isabelle/HOL bewiesen, dass das Betriebssystem seL4 relativ zu seiner Spezifikation korrekt ist [17]. Mithilfe von Coq wurde bewiesen, dass das Multi-Core-Betriebssystem mC2 relativ zu seiner Spezifikation korrekt ist [15].

2.4 Ausblick*: Implementation eines Beweises

In Lean 4 [20, 3, 24, 21] können wir das Beispiel aus Kapitel 2.1.4 wie folgt formalisieren:

```
import Mathlib.Tactic.Contrapose -- for contrapose
open Mathlib.Tactic.Contrapose
```

Die Pinguinaxiome können wir als Typklasse implementieren:

```
structure penguins (a : Type) where
  is_penguin : a → Prop
  is_bark :    a → Prop
  is_bite :    a → Prop
  penguins_that_bark_dont_bite :
    ∀ x : a, is_penguin x → (is_bark x → ¬ is_bite x)
```

Die Kurzform des Beweises lässt sich dann direkt übertragen. In der Übersetzung „proof = program“ entsprechen Implikationen Funktionstypen; somit entspricht der *Modus Ponens* der Anwendung von Funktionen.

```
theorem tux_does_not_bark
  (p : penguins a)
  (Tux : a)
  (Tux_is_penguin : p.is_penguin Tux)
  (Tux_is_bite : p.is_bite Tux)
: (¬ p.is_bark Tux)
:=
by
  have if_Tux_barks_then_doesnt_bite
     : p.is_bark Tux → ¬ p.is_bite Tux
  := by exact p.penguins_that_bark_dont_bite Tux Tux_is_penguin

  have if_Tux_bites_then_doesnt_bark
     : p.is_bite Tux → ¬ p.is_bark Tux
  := by contrapose!
        exact if_Tux_barks_then_doesnt_bite

  show ¬ p.is_bark Tux
  exact if_Tux_bites_then_doesnt_bark Tux_is_bite
```

3

Mengen und Abbildungen

Die Mengenlehre beschreibt die grundlegenden Bausteine, aus denen alle mathematischen Objekte aufgebaut sind:

- Mengen und
- Abbildungen

Wir beschränken uns auf die sogenannte naive Mengenlehre. Eine kurze Einführung in eine rigorose axiomatische Mengenlehre findet sich in Kapitel 3.4. Wir erklären die grundlegenden Konstruktionen und Notationen für Mengen und Abbildungen; außerdem gehen wir kurz auf die Eigenschafaten Injektivität, Surjektivität und Bijektivität von Abbildungen ein, die zum Grundvokabular gehören und es erlauben, viele wiederkehrende Situationen kurz und knapp zu beschreiben.

Zusammen mit den logischen Grundlagen aus Kapitel 1 und Kapitel 2 ist damit unsere Einführung in das Fundament der Mathematik abgeschlossen. Darauf können wir im folgenden weitere Begriffe und Konzepte aufbauen: zum Beispiel die gängigen Zahlbereiche oder diskreten Strukturen.

Überblick über dieses Kapitel.

3.1 Mengen

3.1.1 Naive Mengen

Die naive Mengenlehre beruht auf folgender „Definition“:

„Definition“ 3.1.1 (Cantor, 1895)**.** Unter einer *Menge* verstehen wir jede Zusammenfassung M von bestimmten wohlunterschiedenen Objekten m unserer Anschauung oder unseres Denkens (welche *Elemente* von M genannt werden) zu einem Ganzen.

In der Sprache der Mengenlehre kann man also über Mengen sprechen und darüber, ob Objekte (die selbst letztendlich auch wieder Mengen sind ...) Element einer gegebenen Menge sind oder nicht.

Caveat 3.1.2. Obige „Definition“ ist *keine* Definition im mathematischen Sinne, da einige der auftretenden Begriffe nicht erklärt sind (bzw. nicht erklärbar sind). Wir werden trotzdem im Normalfall mit diesem naiven Mengenbegriff arbeiten. Exakte, axiomatische, Zugänge zur Mengenlehre beruhen auf der folgenden Beobachtung: Die naheliegende Frage

Was ist eine Menge?

sollte besser durch die Frage

Wie kann man mit Mengen umgehen?

ersetzt werden. Ähnliche Prinzipien greifen auch in der Programmierung: In vielen Fällen sollte durch ein API (Application Programming Interface) von der konkreten Implementierung abstrahiert werden und stattdessen spezifiziert werden, welche Konstruktoren, Operationen, und Destruktoren zur Verfügung stehen. Ein axiomatischer Zugang zur Mengenlehre ist in Kapitel 3.4 skizziert.

Gleichheit von Mengen ist die sogenannte *extensionale Gleichheit*:

Definition 3.1.3 (Gleichheit von Mengen)**.** Zwei Mengen sind genau dann *gleich*, wenn sie dieselben Elemente enthalten.

Bemerkung 3.1.4 (Beweis der Gleichheit von Mengen)**.** Um zu beweisen, dass Mengen A und B gleich sind, ist also zu zeigen, dass

- alle Elemente von A in B liegen und dass
- alle Elemente von B in A liegen.

In der Standardnotation führt dies zum Beweisschema 3.1.9.

3.1.2 Notationen und Konstruktionen

Definition 3.1.5 (grundlegende Notationen in der Mengenlehre)**.** Seien A und B Mengen. In Abbildung 3.1 sind die grundlegenden Notationen für Mengen und einfache Mengenkonstruktionen aufgelistet. Eine Veranschaulichung dieser Begriffe findet sich in Abbildung 3.2 bzw. 3.3.

Notation	Bedeutung/Definition
$x \in A$	x ist ein Element von A
$x \notin A$	x ist kein Element von A, d.h. $\neg(x \in A)$
$\emptyset$ oder $\{\}$	die *leere Menge*, d.h. die Menge, die keine Elemente enthält
$\{x \mid C(x)\}$	die Menge aller x, für die $C(x)$ gilt
$\{x, y, z, \dots\}$	die Menge mit den Elementen $x, y, z, \dots$
$A \subset B$	A ist eine *Teilmenge* von B, d.h. alle Elemente von A sind Elemente von B
$A \cap B$	die *Schnittmenge* von A und B, d.h. die Menge $A \cap B := \{x \mid (x \in A) \wedge (x \in B)\}$
$A \cup B$	die *Vereinigung* von A und B, d.h. die Menge $A \cup B := \{x \mid (x \in A) \vee (x \in B)\}$
$A \setminus B$	das *Komplement* von B in A (oder A *ohne* B), d.h. die Menge $A \setminus B := \{x \mid (x \in A) \wedge (x \notin B)\}$
$P(A)$	die *Potenzmenge* von A, d.h. die Menge aller Teilmengen von A: $P(A) := \{x \mid x \subset A\}$
$A \times B$	das *(kartesische) Produkt* von A und B, d.h. $A \times B := \{(x, y) \mid (x \in A) \wedge (y \in B)\}$ Dabei sind Paare (x, y) und (x', y') genau dann gleich, wenn $x = x'$ und $y = y'$ gilt.

Abbildung 3.1.: Grundlegende Notationen/Konstruktionen für Mengen. Hierbei bedeutet „$x := y$“, dass x durch y definiert wird.

Bemerkung 3.1.6. Die Definition der Schnittmenge, der Vereinigung und des Komplements von Mengen lässt erkennen, wie man eine Korrespondenz zwischen logischen Operationen und Mengenkonstruktionen herstellen kann. Insbesondere liefert dies auch Eselsbrücken für die logischen Symbole „$\wedge$“ bzw. „$\vee$“ über die geläufigeren Symbole „$\cap$“ bzw. „$\cup$“ aus der Mengenlehre.

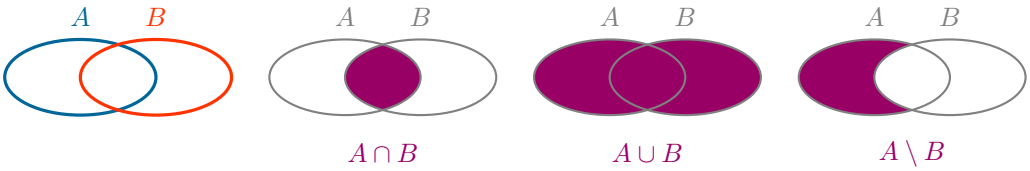


Abbildung 3.2.: Grundlegende Notationen in der Mengenlehre, schematisch

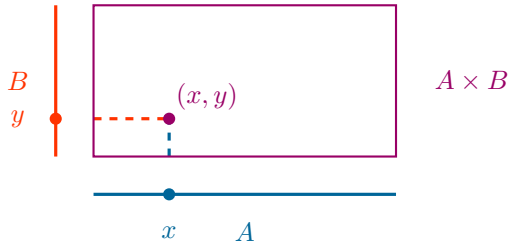


Abbildung 3.3.: Kartesisches Produkt, schematisch

Beispiel 3.1.7. Wir betrachten die Mengen

$$A := \{0, 1\} \quad \text{und} \quad B := \{0, 2\}.$$

Da Mengen genau dann gleich sind, wenn sie die gleichen Elemente enthalten, folgt

$$A = \{1, 0\} = \{1, 0, 1\} = \{0, 0, 0, 1, 1, 1, 0\} = \dots$$

Es kommt also weder auf die notierte Reihenfolge der Elemente an, noch auf die Vielfachheit, mit der Elemente aufgelistet werden.

Es gilt *nicht* $A \subset B$ (da $1 \in A$ aber $1 \notin B$); analog gilt auch *nicht* $B \subset A$. Nach Definition ist

$$\begin{aligned}
A \cap B &= \text{?} \quad \{0\} \\
A \cup B &= \text{?} \quad \{0, 1, 2\} \\
A \setminus B &= \text{?} \quad \{1\} \\
P(A) &= \text{?} \quad \{\emptyset, \{0\}, \{1\}, \{0, 1\}\} \\
P(B) &= \text{?} \quad \{\emptyset, \{0\}, \{2\}, \{0, 2\}\} \\
A \times B &= \text{?} \quad \{(0, 0), (0, 2), (1, 0), (1, 2)\}.
\end{aligned}$$

Caveat 3.1.8. Es ist $P(\emptyset) = \{\emptyset\} \neq \emptyset$, denn $\{\emptyset\}$ enthält ein Element (nämlich $\emptyset$), aber $\emptyset$ enthält keine Elemente.

Beweisschema 3.1.9 (Gleichheit von Mengen)**.** Bemerkung 3.1.4 liefert in die Notation aus Definition 3.1.5 das folgende Beweisschema:

- *Voraussetzung.* Seien A und B Mengen.
- *Behauptung.* Dann gilt $A = B$.
- *Beweis.*
  - Es gilt $A \subset B$, denn: Sei $x \in A$. Dann folgt

    ...

    Also ist $x \in B$.
  - Es gilt $B \subset A$, denn: Sei $x \in B$. Dann folgt

    ...

    Also ist $x \in A$. □

Proposition 3.1.10 (Eigenschaften der Mengenoperationen)**.** *Seien A, B, C Mengen.*

1. *Ist $A \subset B$ und $B \subset C$, so folgt $A \subset C$.*
2. *Es gilt (*Kommutativität *von „∪“ bzw. „∩“)*

$$A \cup B = B \cup A \quad \textit{und} \quad A \cap B = B \cap A.$$

3. *Es gilt (*Assoziativität *von „∪“ bzw. „∩“)*

$$(A \cup B) \cup C = A \cup (B \cup C) \quad \textit{und} \quad (A \cap B) \cap C = A \cap (B \cap C).$$

4. *Es gilt (*Distributivität *von „∪“ und „∩“)*

$$(A \cap B) \cup C = (A \cup C) \cap (B \cup C),$$
$$(A \cup B) \cap C = (A \cap C) \cup (B \cap C).$$

Beweis. Der Beweis dieser Proposition erfolgt, indem man die Behauptungen elementweise überprüft oder indem man geeignete aussagenlogische Tautologien auf die definierenden Eigenschaften der Mengen anwendet (nachrechnen!). Stellvertretend geben wir den Beweis des ersten Teils:

Zu 1. Es gelte $A \subset B$ und $B \subset C$. Sei $x \in A$. Wegen $A \subset B$ ist dann auch $x \in B$. Wegen $B \subset C$ ist somit $x \in C$.

Also gilt: Für alle $x \in A$ ist $x \in C$. D.h. es gilt $A \subset C$. □

Anmerkung zum Lernen. Ebenso wie die Definitionen, sollten Sie sich auch die Aussagen der Propositionen, Sätze, Lemmata, ... einprägen; wichtig ist dabei nicht der genaue Wortlaut, sondern der genaue mathematische Inhalt! Zusätzlich sollten Sie überprüfen, ob Sie die Aussage wirklich verstehen (z.B., indem Sie die Aussage an Beispielen ausprobieren), und ob Sie den Beweis nachvollziehen können.

Notation 3.1.11. Ist A eine Menge, so schreiben wir auch oft

$$\begin{aligned} &\text{„}\forall_{x\in A} \quad \dots\text{“} \quad \text{statt} \quad \text{„}\forall_x \quad \big((x \in A) \implies \dots\big)\text{“} \\ &\text{„}\exists_{x\in A} \quad \dots\text{“} \quad \text{statt} \quad \text{„}\exists_x \quad \big((x \in A) \wedge \dots\big)\text{“} \\ &\text{„}\{x \in A \mid \dots\}\text{“} \quad \text{statt} \quad \text{„}\big\{x \;\big|\; (x \in A) \wedge \dots\big\}\text{“}. \end{aligned}$$

Caveat 3.1.12 (Russellsches Paradoxon)**.** Die Betrachtung von

$$\{x \mid x \text{ ist eine Menge und } x \notin x\}$$

führt zu einem Widerspruch (!), denn:

Angenommen, $A := \{x \mid x \text{ ist eine Menge und } x \notin x\}$ wäre eine Menge. Dann gilt $A \in A$ oder $A \notin A$. Wir betrachten beide Fälle separat:

① Ist $A \in A$, so gilt nach Definition von A, dass $A \notin A$ ist, im Widerspruch zu $A \in A$.

② Ist $A \notin A$, so gilt nach Definition von A, dass $A \in A$ ist, im Widerspruch zu $A \notin A$.

Die Annahme, dass A eine Menge ist, führt somit zu einem Widerspruch. Also kann diese Annahme somit nicht zutreffen und es folgt, dass A *keine* Menge ist.

Dies ist ein ernsthaftes Problem, denn: Ein einziger Widerspruch genügt, um die gesamte Mathematik zusammenstürzen zu lassen! Nach der Wahrheitstafel für „$\implies$“ kann nämlich aus falschen Aussagen *alles* gefolgert werden.

Man darf also nicht wie in Cantors „Definition“ von Mengen *alle* Konstrukte als Mengen zulassen. Der Knackpunkt in Russells Paradoxon ist eine gewisse Selbstbezüglichkeit. Ein möglicher Ausweg ist, ein zweistufiges System einzuführen und Mengen wie in einem API durch ihre Operationen indirekt zu spezifizieren (Kapitel 3.4).

Ausblick 3.1.13 (sets, lists, arrays)**.** Viele Programmiersprachen stellen Datentypen/-strukturen für Mengen, Listen und Arrays zur Verfügung. Diese unterscheiden sich in mehreren Aspekten: Konstruktoren, Eliminatoren, Kombinatoren, sowie Laufzeiten/Speicherverbrauch (Übungsaufgabe). Sowohl semantisch als auch praktisch handelt es sich dabei also um unterschiedliche Konzepte, die passend zur Anwendung eingesetzt werden müssen.

3.2 Abbildungen

Es ist ein allgemeines Prinzip in der Mathematik, nicht nur Objekte zu betrachten, sondern auch zu studieren, wie gewisse Objekte zueinander in Beziehung stehen. Im Fall der Mengenlehre sind die

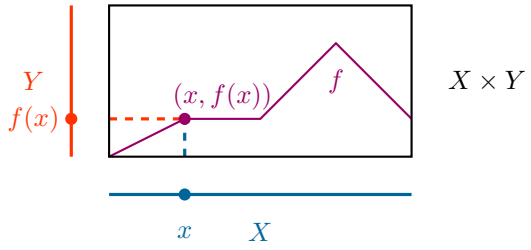


Abbildung 3.4.: Abbildungen, mengentheoretisch

- Objekte Mengen und die
- „Beziehungen“ sind Abbildungen (bzw. Relationen; Kapitel 6).

3.2.1 Abbildungen

„Definition“ 3.2.1 (Abbildung, naive Definition)**.** Seien X und Y Mengen. Eine *Abbildung* $X \longrightarrow Y$ ordnet jedem Element aus X genau ein Element aus Y zu.

Caveat 3.2.2. Dies ist *keine* mathematische Definition, denn „zuordnen“ besitzt keine mathematisch exakte Bedeutung!

Eine saubere Definition erhält man, indem man Abbildungen mengentheoretisch durch ihre Abbildungsgraphen beschreibt (Abbildung 3.4).

Definition 3.2.3 (Abbildung)**.** Seien X und Y Mengen.

- Eine *Abbildung* $X \longrightarrow Y$ ist eine Teilmenge $f \subset X \times Y$ mit folgender Eigenschaft: Für jedes $x \in X$ gibt es genau ein $y \in Y$ mit $(x, y) \in f$; man schreibt in diesem Fall $f(x) = y$ oder $x \longmapsto y$.
- Zwei Abbildungen $f,\ g\colon X \longrightarrow Y$ sind genau dann *gleich*, wenn die zugehörigen Teilmengen von $X \times Y$ gleich sind, d.h., wenn für alle $x \in X$ gilt, dass
$$f(x) = g(x).$$

Caveat 3.2.4. Wie bei Mengen verwenden wir also extensionale Gleichheit für Abbildungen. In Programmiersprachen ist die Lage komplizierter und je nach Situation ist extensionale Gleichheit nicht unbedingt der passende Begriff. Man beachte dabei außerdem, dass in Programmiersprachen mit Seiteneffekten im allgemeinen „Funktionen“ (Prozeduren, Methoden, ...) *keine* Funktionen im mathematischen Sinne sind: aufgrund des veränderlichen globalen Zustands können Funktionswerte nicht nur von ihren expliziten Argumenten, sondern auch vom (impliziten) globalen Zustand abhängen!

Definition 3.2.5 (Identität)**.** Sei X eine Menge. Die *Identität (auf X)* ist die wie folgt definierte Abbildung id_X:

$$\begin{aligned} \mathrm{id}_X \colon X &\longrightarrow X \\ x &\longmapsto x. \end{aligned}$$

(D.h. id_X ist durch die „diagonale“ Teilmenge $\{(x,x) \mid x \in X\} \subset X \times X$ gegeben.)

Beispiel 3.2.6 (einfache Abbildungen)**.** Sei $X := \{0,1,2\}$ und $Y := \{0,2\}$. Dann sind $f := \big\{(0,0),(1,2),(2,2)\big\}$ bzw. (in übersichtlicherer Notation)

$$\begin{aligned} f \colon X &\longrightarrow Y \\ 0 &\longmapsto 0 \\ 1 &\longmapsto 2 \\ 2 &\longmapsto 2 \end{aligned}$$

und $g := \big\{(0,0),(2,2)\big\}$ bzw.

$$\begin{aligned} g \colon Y &\longrightarrow X \\ 0 &\longmapsto 0 \\ 2 &\longmapsto 2 \end{aligned}$$

Abbildungen $X \longrightarrow Y$. Aber die Teilmenge $\{(0,2),(0,0)\}$ von $X \times Y$ beschreibt *keine* Abbildung $X \longrightarrow Y$.

3.2.2 Komposition und Restriktion

Elementare Techniken, um aus Abbildungen weitere Abbildungen zu konstruieren, sind Komposition und Restriktion:

Definition 3.2.7 (Komposition von Abbildungen)**.** Seien X, Y, Z Mengen und seien $f \colon X \longrightarrow Y$ und $g \colon Y \longrightarrow Z$ Abbildungen. Die *Komposition von g mit f* ist die Abbildung („g nach f“)

$$\begin{aligned} g \circ f \colon X &\longrightarrow Z \\ x &\longmapsto g\big(f(x)\big). \end{aligned}$$

D.h. $g \circ f$ entspricht der Teilmenge $\{(x,z) \mid \exists_{y \in Y} \quad ((x,y) \in f) \wedge ((y,z) \in g)\}$ von $X \times Z$.

Beispiel 3.2.8. Für die Abbildungen f und g aus Beispiel 3.2.6 erhalten wir die Kompositionen

$$\begin{aligned} g \circ f \colon X &\longrightarrow X \\ 0 &\longmapsto \boxed{?} \quad g(f(0)) = g(0) = 0 \\ 1 &\longmapsto \boxed{?} \quad \cdots = 2 \\ 2 &\longmapsto \boxed{?} \quad \cdots = 2 \end{aligned}$$

und

$$\begin{aligned} f \circ g \colon Y &\longrightarrow Y \\ 0 &\longmapsto \boxed{?} \quad \cdots = 0 \\ 2 &\longmapsto \boxed{?} \quad \cdots = 2. \end{aligned}$$

Insbesondere ist $f \circ g = \mathrm{id}_Y$.

Bemerkung 3.2.9. Die Komposition von Abbildungen ist assoziativ, d.h. für alle Abbildungen $f \colon X \longrightarrow Y$, $g \colon Y \longrightarrow Z$, $h \colon Z \longrightarrow U$ gilt

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

Außerdem gilt für alle Abbildungen $f \colon X \longrightarrow Y$, dass

$$f \circ \mathrm{id}_X = f \quad \text{und} \quad \mathrm{id}_Y \circ f = f.$$

Diese Eigenschaften der Abbildungskomposition lassen sich leicht elementweise überprüfen (nachrechnen!).

Definition 3.2.10 (Einschränkung von Abbildungen)**.** Seien X und Y Mengen, sei $f \colon X \longrightarrow Y$ eine Abbildung und sei $A \subset X$ eine Teilmenge. Die *Einschränkung von f auf A* ist die Abbildung $f|_A$, die wie folgt definiert ist:

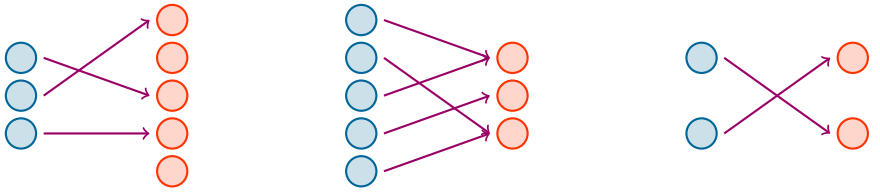
$$\begin{aligned} f|_A \colon A &\longrightarrow Y \\ x &\longmapsto f(x). \end{aligned}$$

Wir nennen in diesem Fall auch f eine *Fortsetzung von $f|_A$ auf X*.

3.2.3 Injektivität, Surjektivität, Bijektivität

Definition 3.2.11 (injektiv/surjektiv/bijektiv)**.** Seien X und Y Mengen.

- Eine Abbildung $f \colon X \longrightarrow Y$ ist *surjektiv*, wenn jedes Element aus Y mindestens ein Urbild besitzt, d.h., wenn es zu jedem $y \in Y$ ein $x \in X$ mit $f(x) = y$ gibt.
- Eine Abbildung $f \colon X \longrightarrow Y$ ist *injektiv*, wenn jedes Element aus Y höchstens ein Urbild unter f besitzt, d.h., wenn für alle x, $x' \in X$ gilt: Ist $f(x) = f(x')$, so ist $x = x'$.



injektiv surjektiv bijektiv

Abbildung 3.5.: Injektiv/surjektiv/bijektiv, schematisch

- Eine Abbildung $X \longrightarrow Y$ ist *bijektiv*, wenn sie injektiv und surjektiv ist (d.h., wenn jedes Element aus Y genau ein Urbild unter dieser Abbildung besitzt).

Caveat 3.2.12. Injektiv ist *nicht* das „Gegenteil“ von surjektiv!

Anmerkung zum Lernen. Machen Sie sich die Begriffe „injektiv“ „surjektiv“ „bijektiv“ an Alltagssituationen klar! (Z.B. „Sozialversicherungsnummer“, „Blutgruppe“, „Strümpfe anziehen (auch für den Oktopus!)“, ...; Abbildung 3.5)

Beispiel 3.2.13. Ist X eine Menge, so ist $\mathrm{id}_X \colon X \longrightarrow X$ sowohl injektiv als auch surjektiv und somit bijektiv.

Beispiel 3.2.14. Wir betrachten die Abbildungen f und g aus Beispiel 3.2.6. Die Abbildung f ist surjektiv, aber *nicht* injektiv (denn $f(1) = 2 = f(2)$). Die Abbildung g ist injektiv, aber *nicht* surjektiv (denn 1 besitzt kein Urbild unter g). Die Abbildung $g \circ f \colon X \longrightarrow X$ ist weder injektiv noch surjektiv.

Definition 3.2.15 (Umkehrabbildung/inverse Abbildung)**.** Seien X und Y Mengen und sei $f \colon X \longrightarrow Y$ eine Abbildung. Eine Abbildung $g \colon Y \longrightarrow X$ ist eine *Umkehrabbildung/inverse Abbildung* von f, wenn

$$g \circ f = \mathrm{id}_X \quad \text{und} \quad f \circ g = \mathrm{id}_Y \,.$$

Beispiel 3.2.16. Ist X eine Menge, so ist id_X eine (bzw. sogar die) Umkehrabbildung von $\mathrm{id}_X \colon X \longrightarrow X$.

Proposition 3.2.17 (Umkehrabbildungen und Bijektivität)**.** *Seien X und Y Mengen und sei $f \colon X \longrightarrow Y$ eine Abbildung.*

1. *Ist f bijektiv, so besitzt f eine Umkehrabbildung; außerdem ist die Umkehrabbildung von f eindeutig bestimmt. Man schreibt dann auch f^{-1} für die Umkehrabbildung von f.*
2. *Besitzt f eine Umkehrabbildung, so ist f bijektiv.*

Beweis. Zu 1. Sei f bijektiv.

Existenz einer Umkehrabbildung. Wir betrachten die Menge

$$g := \big\{(y,x) \;\big|\; (x,y) \in X \times Y \text{ und } f(x) = y\big\} \subset Y \times X.$$

Dann ist g eine Abbildung $Y \longrightarrow X$, denn: Sei $y \in Y$.

- Da $f\colon X \longrightarrow Y$ surjektiv ist, gibt es ein $x \in X$ mit $f(x) = y$, und daher mit $(y,x) \in g$.
- Da $f\colon X \longrightarrow Y$ injektiv ist, gibt es höchstens ein $x \in X$ mit $f(x) = y$ bzw. $(y,x) \in g$.

Also ist $g\colon Y \longrightarrow X$ eine Abbildung.

Nach Konstruktion gilt $g \circ f = \mathrm{id}_X$ und $f \circ g = \mathrm{id}_Y$ (nachrechnen!). Also ist g eine Umkehrabbildung von f.

Eindeutigkeit der Umkehrabbildung. Sei $g'\colon Y \longrightarrow X$ auch eine Umkehrabbildung von f. Wir zeigen, dass dann $g = g'$ gilt: Mit Bemerkung 3.2.9 erhalten wir

$$\begin{aligned}
g &= g \circ \mathrm{id}_Y && \text{(Bemerkung 3.2.9)}\\
&= g \circ (f \circ g') && \text{(da } g' \text{ zu } f \text{ invers ist)}\\
&= (g \circ f) \circ g' && \text{(Bemerkung 3.2.9)}\\
&= \mathrm{id}_X \circ g' && \text{(da } g \text{ zu } f \text{ invers ist)}\\
&= g', && \text{(Bemerkung 3.2.9)}
\end{aligned}$$

da g und g' invers zu f sind. Also ist $g = g'$.

Zu 2. Es gebe eine Umkehrabbildung $g\colon Y \longrightarrow X$ von f.

- Dann ist f surjektiv, denn: Sei $y \in Y$. Mit $x := g(y) \in X$ und $f \circ g = \mathrm{id}_Y$ folgt

$$\begin{aligned}
f(x) = f\big(g(y)\big) = f \circ g(y) &= \mathrm{id}_Y(y)\\
&= y
\end{aligned}$$

  Also ist f surjektiv.

- Und f ist injektiv, denn: Dies kann man anhand der Definitionen überprüfen (Übungsaufgabe).

Insgesamt folgt somit, dass f bijektiv ist. □

Bemerkung 3.2.18 (Eindeutigkeit und Artikel)**.** Da Umkehrabbildungen von bijektiven Abbildungen nach Proposition 3.2.17 eindeutig sind, werden wir im folgenden für Umkehrabbildungen immer den bestimmten Artikel (*die* Umkehrabbildung) verwenden statt den unbestimmten (*eine* Umkehrabbildung).

Ausblick 3.2.19 (andere Grundlagen)**.** Alternativ zur Mengenlehre kann man die Mathematik auch auf anderen Grundlagen aufbauen, z.B. via Typtheorie. Aber auch in diesen alternativen Zugängen ist es nützlich, mit mengenartigen Konstrukten arbeiten zu können; daher wird auch in solchen Zugängen oft ein Teil der Mengenlehre emuliert.

3.3 Anwendung*: Funktionale Programmierung

Imperative Programmierung beruht auf dem Berechnungsmodell der Turing-Maschinen. Im Gegensatz dazu beruht funktionale Programmierung auf dem Berechnungsmodell des λ-Kalküls. Beide Berechnungsmodelle sind äquivalent, liefern aber unterschiedliche Perspektiven.

In der funktionalen Programmierung sind die grundlegenden Objekte Funktionen und ihre Kompositionen. Dabei ist der Funktionsbegriff so allgemein, dass Funktionen mehrere Argumente haben können und Funktionen als Argumente und Werte auftreten können. Dies führt zu Funktionen höherer Ordnung. Im Normalfall erlauben solche Sprachen auch die partielle Auswertung von Funktionen und die Verwendung anonymer Funktionen (durch λ-Abstraktion).

Beispiel 3.3.1. Wir betrachten arithmetische Operationen natürlicher Zahlen in Lean 4 und definieren die folgende Funktion:

```
def f : Nat → Nat → Nat
:= λ x => (λ y => x + 2 * y)
-- λ x y => x + 2 * y
```

Dabei ist „λ“ die Syntax für anonyme Funktionen. Wenn wir die Funktion auf nur einem Argument auswerten, erhalten wir eine Funktion, die nur noch von einem Argument abhängt. Zum Beispiel handelt es sich bei

```
#check f 2                    -- Nat → Nat
```

um die Funktion

```
λ y => 2 + 2 * y : Nat → Nat
```

und wir erhalten:

```
#eval (f 2) 20                -- 42
```

Da es sich dabei um eine gewöhnliche Funktion handelt, kann diese auch wieder mit Funktionen (mit kompatiblen Typen) komponiert werden:

```
#check (f 3) ∘ (f 2)          -- Nat → Nat
#eval ((f 3) ∘ (f 2)) 504     -- 2023
```

In sogenannten *reinen* funktionalen Programmiersprachen erfüllen Funktionen referentielle Transparenz und verhalten sich daher in dem Sinne wie mathematische Funktionen, dass dieselben Argumente zum selben Wert führen (es gibt also keinen impliziten globalen Zustand, der die Werte beeinflusst; Caveat 3.2.4). Beispiele für reine funktionale Programmiersprachen sind Agda, Coq, Haskell, Lean. Sprachen wie C, Java, Python erfüllen dies jedoch *nicht*. Eine anschauliche Einführung in die funktionale Programmierung in Haskell ist *Learn You a Haskell for Great Good!* [25].

In reinen Programmiersprachen ist es einfacher, Aussagen über zu Programme zu beweisen. Dies ist nicht nur für Korrektheitsbeweise relevant, sondern eröffnet auch Optimierungsmöglichkeiten.

Beispiel 3.3.2. Wir wollen folgendes Verfahren implementieren:

- Gegeben einen String,
- wandele alle Buchstaben in Kleinbuchstaben um,
- ersetze dann alle Vorkommen von `u` durch `x`,
- ersetze dann alle Vorkommen von `x` durch `u`,
- entferne dann alle Vokale,
- drehe den resultierenden String um.

In Lean 4 können wir dies wie folgt umsetzen:

```
open List
open Char
open String

def g : String → String
:= asString
∘ reverse
∘ (filter (λ x => not (x ∈ ['a','e','i','o','u'])) )
∘ map (λ x => if x == 'x' then 'u' else x)
∘ map (λ x => if x == 'u' then 'x' else x)
∘ map toLower
∘ toList

#eval toString "Tux"
#eval asString ['T','u','x']
#eval g "Tux ist ein Pinguin"
```

Da die Funktion (aus der Bibliothek Data.List)

```
def map (f : α → β) : List α → List β
  | []    => []
  | a::as => f a :: map f as
```

mit der Komposition des Abbildungsarguments verträglich ist, können wir

```
  map (λ x => if x == 'x' then 'u' else x)
∘ map (λ x => if x == 'u' then 'x' else x)
```

durch

```
  map ( (λ x => if x == 'x' then 'u' else x)
      ∘ (λ x => if x == 'u' then 'x' else x) )
```

ersetzen. Die Komposition der inneren beiden Abbildungen können wir durch die Vereinfachung

```
  map (λ x => if x == 'x' then 'u' else x)
```

ersetzen. Zudem können wir

```
  (filter (λ x => not (x ∈ ['a','e','i','o','u'])) )
∘ map (λ x => if x == 'x' then 'u' else x)
```

zu

```
  (filter (λ x => not (x ∈ ['a','e','i','o','u','x'])) )
```

vereinfachen. Insgesamt erhalten wir somit die äquivalente Funktion:

```
def g' : String → String
:= asString
∘ reverse
∘ (filter (λ x => not (x ∈ ['a','e','i','o','u','x'])) )
∘ map toLower
∘ toList

#eval g' "Tux ist ein Pinguin"  -- "ngnp n ts t"
```

3.4 Ausblick*: Axiomatische Mengenlehre

Was ist axiomatische Mengenlehre? Statt wie in Cantors Definition anzugeben, *was* eine Menge ist, beschreibt man die Mengenlehre durch eine Liste von Axiomen, die angeben, *wie* man mit Mengen umgehen kann.

Wir geben im folgenden die *Axiome für die Mengenlehre nach von Neumann, Bernays und Gödel* [12, 28] an. (Eine andere weitverbreitete Axiomatisierung stammt von Zermelo und Fraenkel; es ergibt sich dabei dieselbe Mengenlehre, jedoch ohne Klassen.) Es gibt zwei Sorten von Objekten, *Mengen* und *Klassen*; man sollte sich dabei Mengen als „kleine Klassen“ vorstellen. Nach dem Komprehensionsaxiom darf man Klassen sehr freizügig zusammenstellen – aber nicht jede Klasse ist eine Menge!

Axiome 3.4.1 (Axiome der Mengenlehre nach von Neumann, Bernays, Gödel)**.**

- Es gibt zwei Sorten von Objekten: *Mengen* und *Klassen*.
- Jede Menge ist eine Klasse.
- Elemente von Klassen sind Mengen.
- *Extensionalität.* Zwei Klassen sind genau dann gleich, wenn sie dieselben Elemente enthalten.
- *Komprehension.* Ist C eine quantorenlogische Aussage „erster Stufe“ in einer mengenwertigen Variablen und wird in C *nicht* über Klassenvariablen quantifiziert, so ist

  $$\big\{x \mid x \text{ ist eine Menge und es gilt } C(x)\big\}$$

  eine Klasse.
- Die *leere Klasse* $\emptyset := \{x \mid x \text{ ist eine Menge und } x \neq x\}$ ist eine Menge.
- Jede Teilklasse einer Menge ist eine Menge; eine Klasse A ist eine *Teilklasse* einer Klasse B, wenn jedes Element von A ein Element von B ist.
- *Paarmengenaxiom.* Sind A und B Mengen, so ist auch $\{A, B\}$ eine Menge.
- *Vereinigungsaxiom.* Ist A eine Menge, so ist auch

  $$\bigcup A := \big\{x \mid \exists_y \quad ((x \in y) \wedge (y \in A))\big\}$$

  eine Menge, die *Vereinigungsmenge von A*.
- *Potenzmengenaxiom.* Ist A eine Menge, so ist auch $P(A) := \{x \mid x \subset A\}$ eine Menge, die *Potenzmenge von A*.
- *Ersetzungsaxiom.* Ist $F\colon X \longrightarrow Y$ eine Funktion zwischen den Klassen X und Y und ist $A \subset X$ eine Teil*menge*, so ist auch $F(A)$ eine Menge.
- *Unendlichkeitsaxiom.* Es gibt eine induktive Menge; eine Menge A heißt *induktiv*, wenn $\emptyset \in A$ und wenn für alle $x \in A$ auch $x \cup \{x\} \in A$ ist.
- *Auswahlaxiom.* Ist A eine Menge mit $\emptyset \notin A$, so gibt es eine *Auswahlfunktion für A*, d.h. eine Funktion $f\colon A \longrightarrow \bigcup A$ mit folgender Eigenschaft: Für alle $x \in A$ ist $f(x) \in x$.

Caveat 3.4.2 (Widerspruchsfreiheit?)**.** Man kann aus den Axiomen der Mengenlehre *nicht* folgern, dass die Mengenlehre widerspruchsfrei ist! (Zweiter Gödelscher Unvollständigkeitssatz). Die Mathematik beruht auf der *Annahme*, dass die Axiome der Mengenlehre widerspruchsfrei sind.

Proposition 3.4.3 (Existenz einer echten Klasse)**.** *Es gibt eine Klasse, die keine Menge ist, nämlich zum Beispiel die* Russellsche Klasse

$$\{x \mid x \text{ ist eine Menge und } x \notin x\}.$$

Beweis. Sei $R := \{x \mid x \text{ ist eine Menge und } x \notin x\}$. Nach dem Komprehensionsaxiom ist R eine Klasse.

Angenommen, R wäre eine Menge. Dann gilt $R \in R$ oder $R \notin R$. Wir betrachten beide Fälle separat.

① Ist $R \in R$, so gilt nach Definition von R, dass $R \notin R$, im Widerspruch zu $R \in R$.

② Ist $R \notin R$, so gilt nach Definition von R, dass $R \in R$ ist, im Widerspruch zu $R \notin R$.

Also ist R keine Menge, und damit eine echte Klasse. □

Eine naheliegende Frage ist, warum man das Russellsche Paradoxon nun nicht einfach eine Ebene höher, also auf Klassenebene, reproduzieren kann, indem man $\{x \mid x \text{ ist eine Klasse und } x \notin x\}$ betrachtet. Dabei ist jedoch zu beachten, dass diese Konstruktion mit unserem Axiomensystem *nicht* zulässig ist: Elemente von Klassen sind immer Mengen!

(Echte) Klassen, über die man in der Mathematik häufig spricht, sind zum Beispiel

- die Klasse aller Mengen,
- die Klasse aller Gruppen,
- die Klasse aller Vektorräume über einem gegebenen Grundkörper,
- die Klasse aller topologischen Räume.

Wir gehen kurz auf das Auswahlaxiom ein. Das Auswahlaxiom scheint zunächst irgendwie offensichtlich zu sein: Ist A eine Menge und gilt $\emptyset \notin A$, so bedeutet dies gerade, dass es zu jedem $x \in A$ ein $y_x \in x$ gibt. Daher scheint es naheliegend, dass man durch $x \mapsto y_x$ eine Auswahlfunktion für A definieren kann. Das Problem dabei ist, dass Abbildungen Mengen sind (nämlich Teilmengen des entsprechenden kartesischen Produktes) und daher den Anforderungen an die Konstruktion von Mengen genügen müssen (so wie sie durch die Axiome bereitgestellt werden). Dies ist in dem gerade betrachteten Beispiel aber nicht klar. Genauer gilt sogar:

Caveat 3.4.4 (Unabhängigkeit des Auswahlaxioms)**.** Das Auswahlaxiom ist „unabhängig“ von den anderen Axiomen der Mengenlehre (d.h. es kann weder aus den übrigen Axiomen gefolgert noch widerlegt werden). Aufgrund der nicht-konstruktiven Charakteristik und etwas ungewöhnlicher Konsequenzen wird im Normalfall explizit angegeben, wenn ein Beweis das Auswahlaxiom verwendet.

Als erste Anwendung des Auswahlaxioms geben wir eine nützliche Charakterisierung von Surjektivität:

Proposition 3.4.5 (Spalte und Surjektivität)**.** *Seien X, Y Mengen und sei $f\colon X \longrightarrow Y$ eine Abbildung. Dann sind folgende Aussagen äquivalent:*

1. *Die Abbildung $f\colon X \longrightarrow Y$ ist surjektiv.*
2. *Die Abbildung $f\colon X \longrightarrow Y$ besitzt einen (Rechts)Spalt. Eine* (Rechts)Spalt *von f ist dabei eine Abbildung $s\colon Y \longrightarrow X$ mit*

$$f \circ s = \mathrm{id}_Y\,.$$

Beweis (AC). Die Abkürzung „AC“ steht dabei für *Axiom of Choice* (Auswahlaxiom) und deutet an, dass der Beweis das Auswahlaxiom verwendet.

„$2 \Longrightarrow 1$“. Sei $s\colon Y \longrightarrow X$ eine Abbildung mit $f \circ s = \mathrm{id}_Y$. Dann ist f surjektiv, denn (s. Beweis von Proposition 3.2.17): Sei $y \in Y$. Für das Element $x := s(y) \in X$ erhalten wir dann

$$f(x) = f\big(s(y)\big) = f \circ s(y) = \mathrm{id}_Y(y) = y.$$

„$1 \Longrightarrow 2$“. Sei $f\colon X \longrightarrow Y$ surjektiv. Dann besitzt f einen Spalt, denn: Wir betrachten die Menge

$$A := \big\{ f^{-1}(\{y\}) \;\big|\; y \in Y \big\}.$$

Da f surjektiv ist, gilt für jedes $y \in Y$, dass $f^{-1}(\{y\}) \neq \emptyset$; also ist $\emptyset \notin A$. Nach dem Auswahlaxiom existiert somit eine Abbildung $g\colon A \longrightarrow \bigcup A$ mit

$$\forall_{z \in A} \quad g(z) \in z.$$

Nach Definition von A ist $\bigcup A = X$. Daher ist

$$\begin{aligned} s\colon Y &\longrightarrow X \\ y &\longmapsto g\big(f^{-1}(\{y\})\big) \end{aligned}$$

eine Abbildung. Nach Konstruktion von g gilt dabei für alle $y \in X$, dass

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

Also ist s ein Spalt von f. □

Ausblick 3.4.6 (Verwandte des Auswahlaxioms)**.** Wichtige, zum Auswahlaxiom äquivalente, Aussagen sind:

- der Wohlordnungssatz,
- das Zornsche Lemma,

- der Satz von Tychonoff.

D.h. diese Sätze sind äquivalent zum Auswahlaxiom, wenn man alle Axiome der Mengenlehre bis auf das Auswahlaxiom annimmt.

4

Induktion

Induktive Konstruktionen und Beweise sind zentral in der Mathematik und Informatik. Induktion formalisiert das „divide & conquer“-Prinzip, mit dem Objekte und Argumente „Schritt für Schritt“ aufgebaut werden.

Der grundlegendste induktive Prozess ist das Zählen. Wir erklären den Zugang zu den natürlichen Zahlen über die Peano-Axiome und das damit zusammenhängende Induktions- bzw. Rekursionsprinzip. Diese Konzepte werden an elementaren Beispielen illustriert, insbesondere an den Fibonacci-Zahlen.

Aus den natürlichen Zahlen können wir später mit weiteren Verfahren die ganzen, rationalen, reellen und komplexen Zahlen konstruieren.

Überblick über dieses Kapitel.

4.1 Peano-Axiome, Induktion, natürliche Zahlen

Die natürlichen Zahlen stellen eine Formalisierung des Zählens dar. Wir erklären zunächst diesen Zählbegriff und führen darauf aufbauend die grundlegenden arithmetischen Operationen (Addition und Multiplikation) auf den natürlichen Zahlen ein.

Was ist wichtig beim Zählen? Zum Zählen benötigen wir (Abbildung 4.1)

- einen Startpunkt („0“),
- das Weiterzählen („+1“).
- Zusätzlich ist es sinnvoll zu fordern, dass alle Anzahlen auf diese Weise eindeutig erreicht werden können.

Dieser Zählprozess wird durch die Peano-Axiome formalisiert.

Abbildung 4.1.: Zählen, schematisch

Axiome 4.1.1 (Peano-Axiome der natürlichen Zahlen)**.** Ein Tripel $(N, 0, s)$ *erfüllt die Peano-Axiome*, wenn N eine Menge ist, $0 \in N$ ist und $s\colon N \longrightarrow N$ eine Abbildung ist, die die folgenden Eigenschaften besitzen:

① Es gibt *kein* $n \in N$ mit $0 = s(n)$.

② Die Abbildung $s\colon N \longrightarrow N$ ist injektiv.

③ *Induktionsprinzip.* Ist $A \subset N$ eine Teilmenge mit den folgenden beiden Eigenschaften, so ist bereits $A = N$:

- *Induktionsanfang.* Es gilt $0 \in A$.
- *Induktionsschritt.* Für alle $n \in A$ ist $s(n) \in A$.

Ist $n \in N$, so nennt man $s(n)$ auch *Nachfolger von* n.

Tripel sind analog zu Paaren definiert, haben aber drei Komponenten. Analog erhält man Qaudrupel, Quintupel, ...

Mithilfe des Induktionsprinzips lassen sich auch Abbildungen induktiv bzw. rekursiv definieren:

Satz 4.1.2 (Rekursionssatz). *Sei $(N, 0, s)$ ein Tripel, das die Peano-Axiome erfüllt. Sei A eine Menge, sei $a \in A$ und sei $g\colon A \longrightarrow A$ eine Abbildung. Dann gibt es genau eine Abbildung $f\colon N \longrightarrow A$ mit der Eigenschaft, dass folgendes gilt:*

$$\begin{aligned} f(0) &= a, \text{ und} \\ f\big(s(n)\big) &= g\big(f(n)\big) \quad \text{für alle } n \in N. \end{aligned}$$

Beweisskizze. Mit dem Induktionsprinzip kann man sich vom vorgegebenen Startwert $0 \longmapsto a$ Nachfolgerschritt für Nachfolgerschritt durch ganz N „hochhangeln“. □

Es stellt sich heraus, dass es bis auf „kanonische, strukturerhaltende Bijektionen“ genau ein Tripel gibt, das die Peano-Axiome erfüllt:

Satz 4.1.3 (Modelle der Peano-Axiome).

1. *Es existiert ein Tripel, das die Peano-Axiome erfüllt.*
2. *Je zwei Tripel, die die Peano-Axiome erfüllen, sind kanonisch isomorph; genauer: Erfüllen $(N, 0, s)$ und $(N', 0', s')$ die Peano-Axiome, so gibt es genau eine Bijektion $f\colon N \longrightarrow N'$ mit $f(0) = 0'$ und der Eigenschaft, dass für alle $n \in N$ gilt, dass $f(s(n)) = s'(f(n))$.*

Beweisskizze. Der Beweis der ersten Aussage beruht auf dem Unendlichkeitsaxiom; der Beweis der zweiten Aussage beruht auf dem Rekursionssatz. Eine ausführlichere Beweisskizze findet sich in Kapitel 4.5. □

Definition 4.1.4 (natürliche Zahlen). Das (bis auf kanonische Isomorphie) eindeutige Tripel, das die Peano-Axiome erfüllt, bezeichnen wir mit $(\mathbb{N}, 0, \ldots + 1)$ und nennen es *natürliche Zahlen.*

Notation 4.1.5. Im Normalfall bezeichnen wir natürliche Zahlen durch ihre Dezimaldarstellung, d.h. $\mathbb{N} = \{0, 1, 2, \ldots\}$ (und ignorieren auch den mengentheoretischen Standpunkt, dass Elemente von Mengen Mengen sind).

Caveat 4.1.6 (beginnen die natürlichen Zahlen mit 0 oder mit 1 ?). In manchen Quellen wird die Konvention verwendet, dass die natürlichen Zahlen mit 1 beginnen. Achten Sie daher unbedingt darauf, welche Konvention jeweils verwendet wird! Beide Konventionen haben ihre Vorzüge:

- In der Analysis verwendet man oft Folgen wie $(1/n)_{n\in\mathbb{N}}$. An dieser Stelle ist es günstig, wenn 0 keine natürliche Zahl ist (was soll $1/0$ sein?!).
- In der Algebra und der Informatik ist es angenehmer, wenn 0 eine natürliche Zahl ist, denn auf diese Weise enthalten die natürlichen Zahlen ein neutrales Element bezüglich Addition.

4.2 Arithmetische Operationen auf den natürlichen Zahlen

Mithilfe des Rekursionssatzes können wir aus der Nachfolgerfunktion nacheinander Addition, Multiplikation und Potenzen natürlicher Zahlen definieren. Die Definition mag zunächst unnötig kompliziert erscheinen (denn das Rechnen mit den natürlichen Zahlen ist doch „offensichtlich“); es hilft an dieser Stelle vielleicht, sich zu überlegen, wie man einem Kind die Addition beibringen kann, wenn es nur Zählen kann ...

Definition 4.2.1 (Addition/Multiplikation/Potenzen)**.** Sei $m \in \mathbb{N}$. Wir definieren die Abbildungen $m + \cdot\colon \mathbb{N} \longrightarrow \mathbb{N}$, $m \cdot \cdot\colon \mathbb{N} \longrightarrow \mathbb{N}$ und $m^{\cdot}\colon \mathbb{N} \longrightarrow \mathbb{N}$ mit Hilfe des Rekursionssatzes wie folgt:

- *Addition.* Sei

$$\begin{aligned} m + 0 &:= m \\ m + (n+1) &:= (m+n) + 1 \quad \text{für alle } n \in \mathbb{N}. \end{aligned}$$

- *Multiplikation.* Sei

$$\begin{aligned} m \cdot 0 &:= 0 \\ m \cdot (n+1) &:= m \cdot n + m \quad \text{für alle } n \in \mathbb{N}. \end{aligned}$$

- *Potenzen.* Sei

$$\begin{aligned} m^0 &:= 1 \\ m^{n+1} &:= m^n \cdot m \quad \text{für alle } n \in \mathbb{N}. \end{aligned}$$

Proposition 4.2.2 (Eigenschaften von Addition/Multiplikation/Potenzen)**.**

1. Neutrale Elemente. *Für alle $n \in \mathbb{N}$ gilt*

$$n + 0 = n = 0 + n \quad \textit{und} \quad n \cdot 1 = n = 1 \cdot n.$$

2. Assoziativität. *Für alle $k, m, n \in \mathbb{N}$ gilt*

$$k + (m + n) = (k + m) + n \quad \textit{und} \quad k \cdot (m \cdot n) = (k \cdot m) \cdot n.$$

3. Kommutativität. *Für alle $m, n \in \mathbb{N}$ gilt*

$$m + n = n + m \quad \textit{und} \quad m \cdot n = n \cdot m.$$

4. Distributivität. *Für alle* $k, m, n \in N$ *gilt*

$$(k + m) \cdot n = k \cdot m + m \cdot n.$$

5. Potenzgesetze. *Für alle* $k, m, n \in N$ *gilt*

$$(k \cdot m)^n = k^n \cdot m^n, \qquad (k^m)^n = k^{m \cdot n}, \qquad k^m \cdot k^n = k^{m+n}.$$

Beweis. All diese Aussagen können per vollständiger Induktion gezeigt werden. Wir beweisen hier nur stellvertretend die Assoziativität der Addition. Genauer: Seien $k, m \in \mathbb{N}$. Wir zeigen per Induktion über $n \in \mathbb{N}$, dass $k + (m + n) = (k + m) + n$ gilt:

- *Induktionsanfang.* Da 0 nach dem ersten Teil (bzw. nach der Definition der Addition) das neutrale Element bezüglich Addition ist, folgt

$$k + (m + 0) = k + m = (k + m) + 0,$$

wie gewünscht.

- *Induktionsschritt.* Sei $n \in \mathbb{N}$ und die Behauptung gelte für n (d.h. es gelte $k + (m + n) = (k + m) + n$). Wir zeigen, dass die Behauptung dann auch für $n + 1$ gilt: Mit der Definition der Addition und der Induktionsvoraussetzung erhalten wir

$$\begin{aligned} k + (m + (n + 1)) &= k + ((m + n) + 1) && \text{(Definition der Addition)} \\ &= (k + (m + n)) + 1 && \text{(Definition der Addition)} \\ &= ((k + m) + n) + 1 && \text{(Induktionsvoraussetzung)} \\ &= (k + m) + (n + 1), && \text{(Definition der Addition)} \end{aligned}$$

wie gewünscht.

Mit dem Induktionsprinzip folgt: Für alle $n \in \mathbb{N}$ ist $k+(m+n) = (k+m)+n$. Damit ist die Assoziativität der Addition gezeigt. □

Bemerkung 4.2.3 (Varianten der vollständigen Induktion)**.** Sind m, $n \in \mathbb{N}$, so schreiben wir $m \leq n$, falls es ein $k \in \mathbb{N}$ mit $m+k = n$ gibt. Zu $n \in \mathbb{N}$ schreibt man auch kurz $\{0, \ldots, n\} := \{m \in \mathbb{N} \mid m \leq n\}$ und $\mathbb{N}_{\geq n} := \{m \in \mathbb{N} \mid n \leq m\}$. Damit können wir die folgenden Varianten des Induktionsprinzips formulieren:

- Sei $m \in \mathbb{N}$. Ist $A \subset \mathbb{N}$ mit $m \in A$ und der Eigenschaft

$$\forall_{n \in A} \quad n + 1 \in A,$$

so folgt bereits $\mathbb{N}_{\geq m} \subset A$.

- Ist $A \subset \mathbb{N}$ mit $0 \in A$ und der Eigenschaft

$$\forall_{n \in \mathbb{N}} \quad \big(\{0, \ldots, n\} \subset A \Longrightarrow n+1 \in A\big),$$

  so folgt bereits $A = \mathbb{N}$.

Beide Varianten lassen sich aus dem ursprünglichen Induktionsprinzip folgern (indem man die Mengen $\{n \in \mathbb{N} \mid m+n \in A\}$ bzw. $\{n \in \mathbb{N} \mid \{0, \ldots, n\} \subset A\}$ betrachtet). Analog gibt es auch die entsprechenden Varianten des Rekursionssatzes (Satz 4.1.2).

Per Induktion/Rekursion können wir Summen- und Produktnotationen einführen:

Notation 4.2.4 ($\sum$ / $\prod$)**.**

- Sei X eine Menge zusammen mit einer assoziativen und kommutativen Addition $+\colon X \times X \longrightarrow X$, die ein bezüglich Addition neutrales Element 0 enthält, und seien $x_0, x_1, \ldots \in X$. Dann definieren wir „$\sum_{j=0}^{n} x_j$“ für alle $n \in \mathbb{N}$ induktiv durch

$$\sum_{j=0}^{0} x_j := x_0$$
$$\sum_{j=0}^{n+1} x_j := \left(\sum_{j=0}^{n} x_j\right) + x_{n+1} \quad \text{für alle } n \in \mathbb{N}.$$

  D.h. $\sum_{j=0}^{n} x_j = „x_0 + \cdots + x_n“$.

  Ist $k \in \mathbb{N}$, so definieren wir analog $\sum_{j=k}^{k+n} x_j = „x_k + x_{k+1} + \cdots + x_{k+n}“$ (durch Induktion über n). Sind $k, k' \in \mathbb{N}$ und gibt es kein $n \in \mathbb{N}$ mit $k+n = k'$, so sei $\sum_{j=k}^{k'} x_j := 0$.

- Analog: Sei X eine Menge zusammen mit einer assoziativen und kommutativen Multiplikation $\cdot\colon X \times X \longrightarrow X$, die ein bezüglich Multiplikation neutrales Element 1 enthält, und seien $x_0, x_1, \ldots \in X$. Dann definieren wir „$\prod_{j=0}^{n} x_j$“ für alle $n \in \mathbb{N}$ induktiv durch

$$\prod_{j=0}^{0} x_j := x_0$$
$$\prod_{j=0}^{n+1} x_j := \left(\prod_{j=0}^{n} x_j\right) \cdot x_{n+1} \quad \text{für alle } n \in \mathbb{N}.$$

  D.h. $\prod_{j=0}^{n} x_j = „x_0 \cdot \cdots \cdot x_n“$.

Ist $k \in \mathbb{N}$, so definieren wir analog $\prod_{j=k}^{k+n} x_j = „x_k \cdot x_{k+1} \cdot \dots \cdot x_{k+n}“$ (durch Induktion über n). Sind $k, k' \in \mathbb{N}$ und gibt es kein $n \in \mathbb{N}$ mit $k+n = k'$, so sei $\prod_{j=k}^{k'} x_j := 1$.

Ist $n \in \mathbb{N}$, so schreibt man auch kurz (lies „n Fakultät“)

$$n! := \prod_{j=1}^{n} j.$$

Insbesondere ist $0! = 1$.

Wir werden in Zukunft den Vorgang des Addierens gerne „umkehren“ wollen. Als Vorbereitung dafür werfen wir noch einmal einen Blick auf die Eigenschaften der Addition natürlicher Zahlen.

Proposition 4.2.5 (Addition auf $\mathbb{N}$, Kürzungsregeln)**.**

1. *Es gilt die folgende Kürzungsregel: Für alle $k, m, n \in \mathbb{N}$ gilt: Ist $k+n = m+n$, so ist bereits $k = m$.*
2. *Jede natürliche Zahl $n \in \mathbb{N} \setminus \{0\}$ besitzt einen Vorgänger, d.h. es gibt ein $m \in \mathbb{N}$ mit $m + 1 = n$.*
3. *Für alle $n, n' \in \mathbb{N}$ gilt: Ist $n + n' = 0$, so ist $n = 0 = n'$.*
4. *Für alle $n, n' \in \mathbb{N}$ existiert ein $m \in \mathbb{N}$ mit $n + m = n'$ oder es existiert ein $m' \in \mathbb{N}$ mit $n' + m' = n$.*

   Für alle $n, n' \in \mathbb{N}$ gilt also $n \leq n'$ oder $n' \leq n$.

Beweis. Zu 1. Dies folgt durch vollständige Induktion über den zu kürzenden Summanden $n \in \mathbb{N}$.

Zu 2. Die Menge

$$\{0\} \cup \{n \in \mathbb{N} \mid \text{es gibt ein } m \in \mathbb{N} \text{ mit } m + 1 = n\}$$

erfüllt das Induktionsprinzip und stimmt somit mit $\mathbb{N}$ überein.

Zu 3. Dies folgt aus der Definition der Addition und der Tatsache, dass 0 keine Nachfolgerzahl ist.

Zu 4. Sei $n' \in \mathbb{N}$. Dann kann man die Behauptung

$$\forall_{n \in \mathbb{N}} \quad \left(\exists_{m \in \mathbb{N}} \quad n + m = n'\right) \vee \left(\exists_{m' \in \mathbb{N}} \quad n' + m' = n\right)$$

durch vollständige Induktion zeigen. □

Literaturaufgabe. Lesen Sie das Buch *Surreal numbers* von Knuth [18].

Literaturaufgabe. Spielen Sie die Spiele *Patrick's Parabox* [30], *Cocoon* [13] und das *Natural number game* [7].

4.3 Beispiele

4.3.1 Die Summe der ersten natürlichen Zahlen

Proposition 4.3.1. *Für alle $n \in \mathbb{N}$ gilt*

$$2 \cdot \sum_{j=1}^{n} j = n \cdot (n+1).$$

Beweis. Wir beweisen die Behauptung durch vollständige Induktion:

- *Induktionsanfang.* Es gilt (nach Definition von $\sum$)

$$2 \cdot \sum_{j=1}^{0} j = 0 = 0 \cdot (0+1).$$

- *Induktionsvoraussetzung.* Sei $n \in \mathbb{N}$ und es gelte $2 \cdot \sum_{j=1}^{n} j = n \cdot (n+1)$.
- *Induktionsschritt.* Wir zeigen, dass die Behauptung dann auch für $n+1$ gilt, d.h., dass $2 \cdot \sum_{j=1}^{n+1} j = (n+1) \cdot (n+2)$. Mit der Induktionsvoraussetzung und den elementaren Eigenschaften der Arithmetik auf $\mathbb{N}$ erhalten wir

$$\begin{aligned}
2 \cdot \sum_{j=1}^{n+1} j &= 2 \cdot \left(\sum_{j=1}^{n} j + (n+1) \right) && (? \text{ nach Definition von } \textstyle\sum) \\
&= 2 \cdot \sum_{j=1}^{n} j + 2 \cdot (n+1) && (? \text{ Ausmultiplizieren von } 2) \\
&= n \cdot (n+1) + 2 \cdot (n+1) && (? \text{ nach Induktionsvoraussetzung}) \\
&= (n+2) \cdot (n+1) && (? \text{ Ausklammern von } n+1) \\
&= (n+1) \cdot (n+2), && (? \text{ Kommutativität von „}\cdot\text{“})
\end{aligned}$$

wie gewünscht. □

4.3.2 Unfundierte Rekursion

Beispiel 4.3.2 (GNU)**.** Das rekursive Akronym GNU = GNU is Not Unix illustriert, dass sowohl Rekursionsanfang als auch Rekursionsschritt relevant sind, um einen terminierenden Rekursionsprozess zu erhalten.

4.3.3 Die Fibonacci-Rekursion

Definition 4.3.3 (Fibonacci-Zahlen)**.** Die Funktion $F\colon \mathbb{N} \longrightarrow \mathbb{N}$ sei rekursiv definiert durch

$$\begin{aligned} F(0) &:= 1 \\ F(1) &:= 1 \\ F(n+2) &:= F(n) + F(n+1) \quad \text{für alle } n \in \mathbb{N}. \end{aligned}$$

Ist $n \in \mathbb{N}$, so wird $F(n)$ als *n-te Fibonacci-Zahl* bezeichnet.

Für diese Rekursion wird ein verallgemeinertes Rekursionsprinzip verwendet; dieses verallgemeinert den Rekursionssatz (Satz 4.1.2) analog zum verallgemeinerten Induktionsprinzip aus Bemerkung 4.2.3.

Beispiel 4.3.4 (die ersten Fibonacci-Zahlen)**.** Mit der Rekursion aus Definition 4.3.3 erhalten wir für die ersten Werte:

$$\begin{aligned} &F(0) = 1, \quad F(1) = 1, \quad F(2) = \boxed{?} \quad 2, \quad F(3) = \boxed{?} \quad 3, \\ &F(4) = \boxed{?} \quad 5, \quad F(5) = \boxed{?} \quad 8, \quad F(6) = \boxed{?} \quad 13, \quad \ldots \end{aligned}$$

Wir können analog rekursiv viele weitere Werte berechnen. Es stellt sich aber die Frage, ob es nicht eine Möglichkeit gibt, die rekursive Beschreibung durch eine geschlossene Formel zu ersetzen. Erstaunlicherweise verwendet die folgende geschlossene Formel nicht nur natürliche Zahlen (obwohl alle Fibonacci-Zahlen natürliche Zahlen sind!), sondern auch den goldenen Schnitt (eine irrationale Zahl; Kapitel 7):

Definition 4.3.5 (goldener Schnitt)**.** Der *goldene Schnitt* ist

$$\varphi := \frac{1+\sqrt{5}}{2}.$$

Außerdem schreiben wir $\overline{\varphi} := \frac{1-\sqrt{5}}{2}$.

Proposition 4.3.6 (eine geschlossene Darstellung der Fibonacci-Zahlen)**.** *Für alle $n \in \mathbb{N}$ gilt (in $\mathbb{R}$; Kapitel 7)*

$$F(n) = \frac{\varphi^{n+1} - \overline{\varphi}^{n+1}}{\sqrt{5}}.$$

Beweis. Wir beweisen die Behauptung durch vollständige Induktion. Genauer gehen wir wie folgt vor: Zu $n \in \mathbb{N}$ sei

$$G(n) := \frac{\varphi^{n+1} - \overline{\varphi}^{n+1}}{\sqrt{5}}.$$

Wir zeigen induktiv, dass $F(n) = G(n)$ für alle $n \in \mathbb{N}$ gilt, indem wir nachweisen, dass G dieselbe verallgemeinerte Rekursion erfüllt wie F. Als Vorüberlegung berechnen wir, dass

$$\begin{aligned}\varphi^2 - \varphi - 1 &= \left(\frac{1+\sqrt{5}}{2}\right)^2 - \frac{1+\sqrt{5}}{2} - 1 = \frac{1 + 2\cdot\sqrt{5} + 5 - 2 - 2\cdot\sqrt{5} - 4}{4} \\ &= 0\end{aligned}$$

und analog $\overline{\varphi}^2 - \overline{\varphi} - 1 = \cdots = 0.$ gilt. Wir führen nun die eigentliche Induktion durch:

- *Rekursionsanfang.* Es gilt

  $$G(0) = \frac{\varphi^1 - \overline{\varphi}^1}{\sqrt{5}} = \frac{\varphi - \overline{\varphi}}{\sqrt{5}} = \frac{1 + \sqrt{5} - (1 - \sqrt{5})}{\sqrt{5}\cdot 2} = \frac{2\cdot\sqrt{5}}{\sqrt{5}\cdot 2} = 1 = F(0).$$

  Mit der obigen Vorüberlegung erhalten wir außerdem:

  $$G(1) = \frac{\varphi^2 - \overline{\varphi}^2}{\sqrt{5}} = \frac{1 + \varphi - (1 + \overline{\varphi})}{\sqrt{5}} = \frac{\varphi - \overline{\varphi}}{\sqrt{5}} = G(1) = 1 = F(1)$$

- *Rekursionsschritt.* Für alle $n \in \mathbb{N}$ ist

  $$\begin{aligned}G(n) + G(n+1) &= \frac{\varphi^{n+1} - \overline{\varphi}^{n+1}}{\sqrt{5}} + \frac{\varphi^{n+2} - \overline{\varphi}^{n+2}}{\sqrt{5}} \\ &= \frac{\varphi^{n+1}\cdot(1+\varphi) - \overline{\varphi}^{n+1}\cdot(1+\overline{\varphi})}{\sqrt{5}} \\ &= \frac{\varphi^{n+1}\cdot\varphi^2 - \overline{\varphi}^{n+1}\cdot\varphi^2}{\sqrt{5}} && \text{(nach der Vorüberlegung)} \\ &= G(n+2).\end{aligned}$$

Also erfüllt G dieselbe (verallgemeinerte) Rekursion wie F. Mit der Eindeutigkeitsaussage aus dem (verallgemeinerten) Rekursionssatz folgt $G = F$. □

Ausblick 4.3.7 (woher kommt $\sqrt{5}$?!)**.** Mit Methoden aus der Linearen Algebra kann man ein allgemeines Verfahren entwickeln, das es erlaubt, für verallgemeinerte lineare Rekursionen zugehörige geschlossene Darstellungen zu finden. Insbesondere wird dieser Zugang auch erklären, woher die Wurzeln kommen.

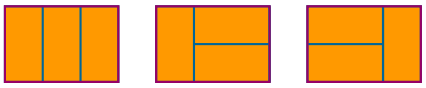
Ausblick 4.3.8 (OEIS)**.** Falls Hilfe für mysteriöse Zahlenfolgen gesucht ist, ist die *On-Line Encyclopedia of Integer Sequences* eine große Hilfe [26]. Insbesondere finden sich dort auch noch weitere Informationen zu den Fibonacci-Zahlen.

4.3.4 Anwendung: Dominozerlegungen zählen

Die Fibonacci-Folge besitzt zahlreiche Anwendungen: Sie tritt zum Beispiel im *Liber Abaci* aus dem Jahr 1202 von Leonardo Pisano (genannt Fibonacci) auf und wir dort verwendet, um ein (unrealistisches) Populationsmodell für Hasen zu studieren. Außerdem liefern die Fibonacci-Zahlen Lösungen von Zählproblemen in indischen Rhythmusfolgen. Wir betrachten das folgende, verwandte, Problem:

Problem 4.3.9. Sei $n \in \mathbb{N}$. Wieviele Möglichkeiten gibt es, ein $n \times 2$-Rechteck mit 1×2-Dominos zu pflastern? (Pflasterungen dürfen keine Lücken und keine Überlappungen besitzen; die Dominos dürfen sowohl „horizontal“ als auch „vertikal“ verwendet werden)

Beispiel 4.3.10. Alle Möglichkeiten, ein 3×2-Rechteck mit 1×2-Dominos zu pflastern, sind:



Proposition 4.3.11. *Sei $n \in \mathbb{N}$. Dann gibt es genau $F(n)$ Möglichkeiten, ein $n \times 2$-Rechteck mit 1×2-Dominos zu pflastern.*

Beweis. Zu $n \in \mathbb{N}$ sei $P(n)$ die Anzahl der Pflasterungen eines $n \times 2$-Rechtecks durch 1×2-Dominos. Wie im Beweis von Proposition 4.3.6 genügt es zu zeigen, dass P dieselbe verallgemeinerte Rekursion wie F erfüllt.

- *Rekursionsanfang.* Es ist $P(0) = 1$, da es genau eine Möglichkeit gibt, ein 0×2-Rechteck mit 1×2-Dominos zu pflastern (diese Pflasterung verwendet 0 Dominos).

  Es ist $P(1) = 1$, da es genau eine Möglichkeit gibt, ein 1×2-Rechteck mit 1×2-Dominos zu pflastern (diese Pflasterung verwendet genau ein 1×2-Domino und dieses wird „vertikal“ eingesetzt).

- *Rekursionsschritt.* Sei $n \in \mathbb{N}$. Jede Pflasterung eines $(n+2) \times 2$-Rechtecks hat entweder die Form

  oder die Form

Also ist $P(n+2) = P(n) + P(n+1)$.

Somit folgt $P = F$. □

4.4 Anwendung*: Induktion und Rekursion in der Programmierung

Induktion und Rekursion sind zentrale Elemente der Programmierung. Dabei wird im allgemeinen nicht nur Induktion über die natürlichen Zahlen, sondern über allgemeine (wohlfundierte) Strukturen verwendet.

4.4.1 Induktive Datentypen

Viele funktionale Programmiersprachen verfügen über Mechanismen, um induktive Datentypen und Funktionen darauf zu konstruieren und zu verwenden.

Beispiel 4.4.1 (natürliche Zahlen)**.** Natürliche Zahlen werden in vielen funktionalen Programmiersprachen induktiv durch den Startwert 0 und einen Nachfolgerkonstruktor definiert. Entsprechend können dann arithmetische und andere Funktionen rekursiv über diese Struktur definiert werden. Im Gegensatz zu natürlichen Zahlen mit fester Bitlänge sind dies (unter der Annahme, dass beliebig viel Speicherplatz zur Verfügung steht ...) exakte natürliche Zahlen ohne Überlauf!

Beispiel 4.4.2 (Listen)**.** Ähnlich werden auch Listen in vielen funktionalen Programmiersprachen induktiv durch den Startwert Nil (der leeren Liste) und den Konstruktor Cons (der ein Listenelement „vorne“ an eine Liste anhängt) definiert. Entsprechend können dann Funktionen auf Listen rekursiv über diese Struktur definiert werden und Aussagen über solche Funktionen induktiv bewiesen werden.

Allgemeiner kann man auch Bäume etc. induktiv repräsentieren.

Beispiel 4.4.3 (logische Ausdrücke)**.** In der Implementation von Semantiken in Kapitel 1.4 haben wir aussagenlogische Formeln (mit einem Basisfall und fünf Induktionsfällen) und Wahrheitswerte (mit zwei Basisfällen und ohne Induktionsfälle) durch induktive Datentypen modelliert. Entsprechend haben wir die Semantiken durch rekursive Funktionen auf diesen Datentypen definiert.

Auch der Beweis des Korrektheitssatzes (Satz 2.1.2) basiert auf einer analogen strukturellen Induktion über die Struktur logischer Aussagen.

4.4.2 Schleifen

In imperativen Sprachen gehören Schleifenkonstrukte (z.B. for, while, ...) und rekursive „Funktions"aufrufe zu den Standardkontrollstrukturen. Solche Schleifenkonstrukte erlauben es, rekursiv Prozesse zu modellieren. Diese sind im Normalfall jedoch noch weniger stark eingeschränkt als induktive/rekursive Definitionen entlang von induktiven Datentypen; insbesondere ist mehr Vorsicht geboten, um sicherzustellen, dass die Implementierung das gewünschte Terminierungsverhalten besitzt.

Beweise über Schleifenkonstrukte werden induktiv geführt, mithilfe sogenannter Schleifeninvarianten.

4.5 Ausblick*: Konstruktion der natürlichen Zahlen

Wir skizzieren kurz einen Beweis der Konstruktion bzw. Eindeutigkeit der natürlichen Zahlen, aufbauend auf der axiomatischen Mengenlehre und den Peano-Axiomen.

Satz 4.5.1 (Modelle der Peano-Axiome)**.**

1. *Es existiert ein Tripel, das die Peano-Axiome erfüllt.*
2. *Je zwei Tripel, die die Peano-Axiome erfüllen, sind kanonisch isomorph; genauer: Erfüllen $(N, 0, s)$ und $(N', 0', s')$ die Peano-Axiome, so gibt es genau eine Bijektion $f\colon N \longrightarrow N'$ mit $f(0) = 0'$ und der Eigenschaft, dass für alle $n \in N$ gilt, dass $f(s(n)) = s'(f(n))$.*

Beweis. Existenz. Wir betrachten

$$N := \bigcap \{A \mid A \text{ ist eine induktive Menge}\},$$

also sozusagen die „kleinste induktive Menge". Nach dem Unendlichkeitsaxiom gibt es mindestens eine induktive Menge; insbesondere ist $\emptyset \in N$ und somit ist N nicht-leer. Nach dem Teilmengenaxiom ist N eine Menge und aus der Definition von N folgt leicht, dass N induktiv ist.

Wir definieren nun 0 als das Element $\emptyset \in N$ und

$$\begin{aligned} s\colon N &\longrightarrow N \\ x &\longmapsto x \cup \{x\} \end{aligned}$$

(letztere Definition funktioniert, da N eine induktive Menge ist).

Durch sorgfältiges Nachrechnen kann man nun zeigen, dass dieses Tripel $(N, 0, s)$ tatsächlich die Peano-Axiome erfüllt.

Eindeutigkeit. Erfüllen $(N, 0, s)$ und $(N', 0', s')$ die Peano-Axiome, so definieren wir mithilfe des Rekursionssatzes Abbildungen $f\colon N \longrightarrow N'$ und $f'\colon N' \longrightarrow N$ durch

- $f(0) := 0'$ und $f(s(n)) := s'(f(n))$ für alle $n \in N$,
- $f'(0') := 0'$ und $f'(s'(n)) := s(f'(n))$ für alle $n \in N'$.

Dann erfüllen $f' \circ f$ bzw. $f \circ f'$ die Rekursion für id_N bzw. $\mathrm{id}_{N'}$. Aus der Eindeutigkeitsaussage im Rekursionssatz folgt daher $f' \circ f = \mathrm{id}_N$ und auch $f \circ f' = \mathrm{id}_{N'}$. Also ist $f\colon N \longrightarrow N'$ bijektiv und nach Konstruktion mit s bzw. s' verträglich. Außerdem folgt mit der Eindeutigkeitsaussage aus dem Rekursionssatz auch, dass es nur eine solche strukturerhaltende Bijektion geben kann. □

5

Graphen, Bäume, Modellierung

Beziehungen zwischen Objekten werden in der Informatik oft durch sogenannte Graphen modelliert. Die Objekte entsprechen dabei den Knoten des Graphen und die Beziehungen den Kanten des Graphen. Graphen werden dabei sowohl für die abstrakte Repräsentation (z.B. für Datenstrukturen) als auch für die Modellierung konkreter Probleme eingesetzt.

Bäume sind spezielle Graphen, die sich insbesondere eignen, um geordnete Daten effizient zu repräsentieren. Dazu gehören zum Beispiel Datenstrukturen für Sortier- und Suchprobleme.

Wir führen grundlegende Begriffe und Eigenschaften von Graphen bzw. Bäumen ein und illustrieren diese an elementaren Beispielen. Außerdem lernen wir das Modellierungsprinzip kennen.

In Kapitel 6 werden wir eine weitere Perspektive auf Graphen kennenlernen: Relationen.

Überblick über dieses Kapitel.

5.1 Graphen

Graphen bestehen aus „Knoten“ und „Kanten“; dabei verbindet jede Kante zwei Knoten. Dieses anschauliche Konzept kann man wie folgt formalisieren: Knoten und Kanten bilden jeweils eine Menge. Sind die Kanten ungerichtet, so modelliert man sie durch zwei-elementige Mengen (diese sind ungeordnet!); sind die Kanten gerichtet, so modelliert man sie durch Paare. Knotenmenge und Kantenmenge eines Graphen werden als Paar in einem Objekt zusammengefasst.

Caveat 5.1.1. Es gibt viele verschiedene Varianten von Graphen! Bei der Verwendung von Literatur ist also immer darauf zu achten, welche Definitionen/Konventionen verwendet werden.

5.1.1 Ungerichtete Graphen

Definition 5.1.2 (ungerichteter Graph)**.** Ein *ungerichteter Graph* ist ein Paar (V, E), bestehend aus einer Menge V und einer Menge E mit $E \subset V[2]$, wobei

$$V[2] := \big\{\{v, w\} \;\big|\; v, w \in V,\ v \neq w\big\}$$

die Menge der zwei-elementigen Teilmengen von V bezeichnet. Die Elemente von V heißen *Knoten*, die Elemente von E heißen *Kanten*. Ein Graph (V, E) ist *endlich*, wenn V und E endliche Mengen sind (Anhang A.2).

Beispiel 5.1.3. Wir betrachten die folgenden Graphen:

$$\begin{aligned}
X &:= \big(\{1,\dots,4\}, \{\{1,2\},\{1,3\},\{1,4\},\{2,3\},\{2,4\},\{3,4\}\}\big)\\
Y &:= \big(\{1,\dots,5\}, \{\{1,2\},\{1,3\},\{3,4\},\{3,5\}\}\big)\\
Q &:= \big(\{1,\dots,5\}, \{\{1,2\},\{1,4\}\{2,3\},\{3,4\},\{4,5\}\}\big)\\
Q' &:= \big(\{1,\dots,5\}, \{\{1,2\},\{1,5\},\{1,4\},\{2,3\},\{3,4\}\}\big)\\
P &:= \big(\{1,\dots,5\}, \{\{1,2\},\{1,4\},\{1,5\},\{2,3\},\{4,5\}\}\big)\\
L &:= \big(\{1,\dots,5\}, \{\{1,2\},\{3,4\},\{4,5\}\}\big)
\end{aligned}$$

Diese Graphen können wie in Abbildung 5.1 dargestellt werden. Es kommt dabei nur darauf an, welche Knoten mit welchen Knoten durch Kanten verbunden sind – die konkrete Anordnung der Knoten/Kanten ist nicht relevant für die kombinatorische Struktur (kann aber sehr unübersichtlich sein ...).

Definition 5.1.4 (vollständige Graphen)**.** Ein ungerichteter Graph (V, E) ist *vollständig*, wenn $E = V[2]$ ist. Ist $n \in \mathbb{N}$, so schreiben wir (Abbildung 5.2)

$$K_n := \big(\{1,\dots,n\}, \{\{j,k\} \mid j,k \in \{1,\dots,n\},\ j \neq k\}\big).$$

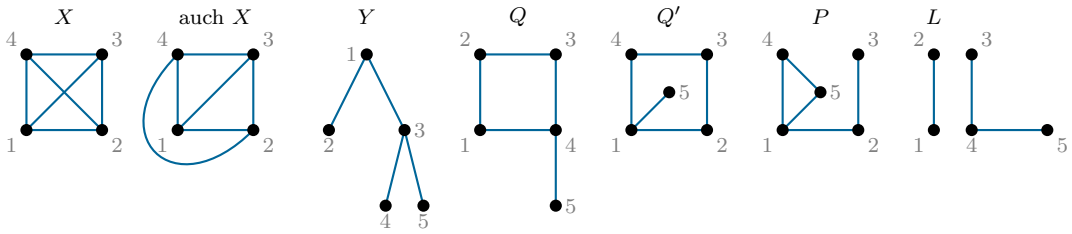


Abbildung 5.1.: Beispiele für Graphen (Beispiel 5.1.3)

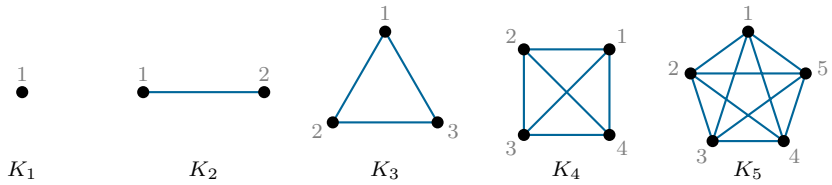


Abbildung 5.2.: vollständige Graphen

Definition 5.1.5 (benachbart, Grad eines Knotens)**.** Sei $X = (V, E)$ ein ungerichteter Graph und sei $v \in V$.

- Der Knoten v ist *(in X) benachbart* zu $w \in V$, wenn $\{v, w\} \in E$ ist.
- Ist die Menge $N_X(v)$ der Nachbarn von v in X endlich, so definiert man den *Grad von v (in X)* durch

$$\deg_X v := \#N_X(v).$$

Beispiel 5.1.6. In vollständigen Graphen sind je zwei verschiedene Knoten benachbart. Wir betrachten den Graphen Q aus Beispiel 5.1.3.

- Sind die Knoten 2 und 3 in Q benachbart?
  Ja Nein Die korrekte Antwort ist „Ja“. ×
- Sind die Knoten 2 und 4 in Q benachbart?
  Ja Nein Die korrekte Antwort ist „Nein“. ×
- Der Knoten 2 hat in Q Grad ? 2.
- Der Knoten 4 hat in Q Grad ? 3.

Ein klassisches Anwendungsfeld der Graphentheorie sind Netzwerke aller Art, z.B. soziale Netzwerke, kryptographische Vertrauensnetzwerke („web of

trust“) oder die Verlinkungsstruktur zwischen Webseiten. Ein gutes Verständnis der Eigenschaften solcher Graphen ist sowohl von akademischem als auch von wirtschaftlichem Interesse. Wir geben ein einfaches Beispiel:

Fingerübung 5.1.7 (Kontaktgraphen)**.** Sei M eine Menge von Personen, z.B. alle Einwohner eines Landkreises. Wir betrachten den folgenden ungerichteten Graphen:

- Knoten sind alle Personen in M.
- Zwei Knoten sind genau dann durch eine Kante verbunden, wenn die zugehörigen Personen in den letzten 14 Tagen einen Risikokontakt (z.B. bzgl. SARS-Cov-2) gebildet haben.

Es handelt sich dabei um einen endlichen ungerichteten Graphen.

- Welche praktische Bedeutung hat der Grad eines Knotens in diesem Graphen?
- Wie verändert sich der Graph für Regensburg bei Schließung/Öffnung von Kindergärten? Schulen? Hochschulen? Diskotheken?

Wir sehen Graphen als hinreichend ähnlich an, wenn sie „dieselbe Struktur“ besitzen und sich nur durch „Umbenennung der Knoten“ unterscheiden. Dies wird durch den Begriff des Isomorphismus (von Graphen) formalisiert; solche Isomorphismen müssen sowohl die Struktur der Knoten als auch die Struktur der Kanten ineinander überführen.

Definition 5.1.8 (Graphenisomorphismus)**.** Seien $X = (V, E)$ und $X' = (V', E')$ ungerichtete Graphen.

- Ein *Isomorphismus* $f\colon X \longrightarrow X'$ ist eine Abbildung $f\colon V \longrightarrow V'$ mit folgenden Eigenschaften:
  - Die Abbildung f ist bijektiv.
  - Für alle $v, w \in V$ gilt
$$\{v, w\} \in E \Longleftrightarrow \big\{f(v), f(w)\big\} \in E'.$$
- Wir nennen die Graphen X und X' *isomorph*, wenn es einen Isomorphismus $X \longrightarrow X'$ gibt.

Man beachte, dass in dieser Definition, die Notation $X \longrightarrow X'$ wirklich nur als Notation zu verstehen ist. Die unterliegende Abbildung ist eine Abbildung zwischen den Knotenmengen.

Bemerkung 5.1.9 (Invarianten)**.** Sind zwei Graphen isomorph, so können wir dies nachweisen, indem wir einen konkreten Isomorphismus angeben.

Aber wie können wir nachweisen, dass zwei Graphen *nicht* isomorph sind? Im Normalfall ist es nicht praktikabel, alle Abbildungen zwischen den Knotenmengen darauf zu überprüfen, ob sie bijektiv und mit der Kantenstruktur verträglich sind. In einem solchen Fall sind *Invarianten* hilfreich. Wir illustrieren dieses Prinzip an konkreten Beispielen: Seien $X = (V, E)$ und $X' = (V', E')$ isomorphe (endliche) ungerichtete Graphen. Dann gilt:

- Es ist $\#V = \#V'$, da die Mächtigkeit unter Bijektionen invariant ist.
- Es ist $\#E = \#E'$, da jeder Isomorphismus auch eine Bijektion zwischen den Kantenmengen induziert (nachrechnen!).
- Ist $f\colon X \longrightarrow X'$ ein Isomorphismus und ist $v \in V$, so gilt (da f eine Bijektion $N_X(v) \longrightarrow N_{X'}(f(v))$ induziert; nachrechnen!)

$$\deg_X v = \deg_{X'} f(v).$$

Die Umkehrung gilt im allgemeinen *nicht*!

Beispiel 5.1.10. Wir betrachten die Graphen in Beispiel 5.1.3.

- Sind die Graphen X und Q isomorph?

  Ja Nein Die korrekte Antwort ist „Nein“. Der Graph Q hat mehr Knoten als X. ×

- Sind die Graphen Y und Q isomorph?

  Ja Nein Die korrekte Antwort ist „Nein“. Der Graph Y hat weniger Kanten als Q. ×

- Sind die Graphen Q und Q' isomorph?

  Ja Nein Die korrekte Antwort ist „Ja“. Z.B. ist $1 \mapsto 4$, $2 \mapsto 3$, $3 \mapsto 2$, $4 \mapsto 1$, $5 \mapsto 5$ ein Isomorphismus. ×

- Sind die Graphen Q und P isomorph?

  Ja Nein Die korrekte Antwort ist „Nein“. Der eindeutige Knoten vom Grad 1 ist in P zu einem Knoten vom Grad 2 benachbart, aber in Q zu einem Knoten vom Grad 3. ×

5.1.2 Gerichtete Graphen

Definition 5.1.11 (gerichteter Graph)**.** Ein *gerichteter Graph* ist ein Paar (V, E), bestehend aus einer Menge V und einer Menge $E \subset V \times V$. Die Elemente von V heißen *Knoten*, die Elemente von E heißen *Kanten*.

Ein gerichteter Graph ist *endlich*, wenn V und E endliche Mengen sind.

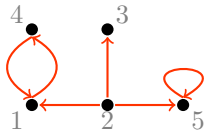


Abbildung 5.3.: Der gerichtete Graph aus Beispiel 5.1.12

Beispiel 5.1.12. Der gerichtete Graph

$$\big(\{1,\dots,5\},\{(1,4),(2,1),(2,3),(2,5),(4,1),(5,5)\}\big)$$

kann wie in Abbildung 5.3 veranschaulicht werden.

Ausblick 5.1.13 (Graphenzoo)**.** Es gibt viele weitere Varianten von Graphen, zum Beispiel:

- Werden mehrere (gerichtete oder ungerichtete) Kanten zwischen zwei Knoten ermöglicht, so spricht man auch von *Multigraphen*.
- Werden Knoten/Kanten mit Werten/Attributen versehen, so spricht man auch von *gewichteten* oder *dekorierten* Graphen.

Man sollte jeweils die Graphen wählen, die am besten zur betrachteten Anwendung passen.

Notation 5.1.14. Wir werden im folgenden zumeist nur ungerichtete Graphen betrachten und diese als „Graphen“ bezeichnen; gerichtete Graphen werden wir immer als „gerichtete Graphen“ bezeichnen.

5.1.3 Wege und Zusammenhang

Definition 5.1.15 (Weg, zusammenhängend)**.** Sei $X = (V, E)$ ein Graph.

- Ein *Weg in* X ist eine Folge $(v_0, \dots, v_n)$ von (paarweise) verschiedenen Knoten in V mit $n \in \mathbb{N}$ und

  $$\forall_{j\in\{0,\dots,n-1\}} \quad \{v_j, v_{j+1}\} \in E.$$

  In diesem Fall ist n die *Länge des Weges* $(v_0, \dots, v_n)$ und wir sagen, dass $(v_0, \dots, v_n)$ ein Weg von v_0 nach v_n ist.
- Ein *Kreis in* X ist ein Weg $(v_0, \dots, v_n)$ der Länge $n \geq 2$, der zusätzlich $\{v_n, v_0\} \in E$ erfüllt.
- Der Graph X ist *zusammenhängend*, wenn es für alle Knoten v, w von X mindestens einen Weg in X von v nach w gibt.

Caveat 5.1.16. Es gibt verschiedene Konventionen für Wege/Kreise in Graphen, abhängig davon, ob Knoten/Kanten wiederholt werden dürfen. Alle Varianten haben ihre eigenen Vor- und Nachteile.

Beispiel 5.1.17. Vollständige Graphen sind offensichtlich zusammenhängend. Wir betrachten die Graphen aus Beispiel 5.1.3. In Y ist $(2, 1, 3, 5)$ ein Weg; in Q ist $(1, 2, 3, 4)$ ein Kreis.

- Ist X zusammenhängend?

  Ja Nein Die korrekte Antwort ist „Ja“. ×

- Ist Y zusammenhängend?

  Ja Nein Die korrekte Antwort ist „Ja“. ×

- Ist Q zusammenhängend?

  Ja Nein Die korrekte Antwort ist „Ja“. ×

- Ist Q' zusammenhängend?

  Ja Nein Die korrekte Antwort ist „Ja“. ×

- Ist P zusammenhängend?

  Ja Nein Die korrekte Antwort ist „Ja“. ×

- Ist L zusammenhängend?

  Ja Nein Die korrekte Antwort ist „Nein“. Es gibt in Q keinen Weg von 2 nach 3. ×

Bemerkung 5.1.18 (gerichteter Weg)**.** In gerichteten Graphen verwenden wir die entsprechende gerichtete Version von Wegen: Ist $X = (V, E)$ ein gerichteter Graph, so ist ein *Weg in* X eine Folge $(v_0, \dots, v_n)$ (paarweise) verschiedener Knoten in V mit $n \in \mathbb{N}$ und

$$\forall_{j \in \{0,\dots,n-1\}} \quad (v_j, v_{j+1}) \in E.$$

Beispiel 5.1.19 (das Collatz-Problem)**.** Wir betrachten den folgenden als *Collatz-Graph* bezeichneten gerichteten Graphen [9]:

$$\big(\mathbb{N}, \{(x, 3 \cdot x + 1) \mid x \in \mathbb{N} \text{ ungerade}\} \cup \{(2 \cdot x, x) \mid x \in \mathbb{N}_{>0}\}\big)$$

Zum Beispiel gibt es einen Weg

- von 1 nach 1, nämlich: $1, 4, 2, 1$
- von 5 nach 1, nämlich: $5, 16, 8, 4, 2, 1$
- von 3 nach 1, nämlich: $3, 10, 5, 16, 8, 4, 2, 1$

- von 9 nach 1, nämlich:

$$9, 28, 14, 7, 22, 11, 34, 17, 52, 26, 13, 20, 20, 10, 5, 16, 8, 4, 2, 1 \text{ (puh!)}$$

Es ist ein offenes Problem (!), ob es von jeder natürlichen Zahl (ungleich 0) einen Weg im Collatz-Graphen zur 1 gibt.

Ausblick 5.1.20. Das effiziente, algorithmische Finden kürzester Wege in (gewichteten, gerichteten) Graphen hat viele praktische Anwendungen, z.B. in der Routenplanung für Züge, Autos, Fahrradfahrer, Fußgänger, Bergsteiger, ... und auch beim Vergleich von DNA-Sequenzen.

5.2 Bäume

Bäume sind Graphen, die „topologisch“ einfach sind: Sie sind zusammenhängend und enthalten keine Kreise. Bäume spielen in der Informatik eine tragende Rolle, insbesondere als Datenstrukturen.

Caveat 5.2.1. Um die Notation übersichtlich zu halten, betrachten wir im folgenden nur *ungerichtete Bäume*. Im Kontext von Datenstrukturen werden normalerweise gerichtete, geordnete Bäume verwendet (bei denen die „Kinder“ eine Reihenfolge besitzen), die außerdem noch mit weiteren Werten/Attributen dekoriert sind.

5.2.1 Bäume

Definition 5.2.2 (Baum, Blatt)**.**

- Ein *Baum* ist ein zusammenhängender nicht-leerer Graph ohne Kreise.
- Ein Knoten v eines Baumes T heißt *Blatt*, falls $\deg_T v = 1$ ist oder falls T nur diesen einen Knoten v besitzt.

Beispiel 5.2.3. Der einzige Baum unter den Graphen aus Beispiel 5.1.3 ist der Graph ? Y mit den Blättern ? 2, 4, 5. Alle anderen dort aufgeführten Graphen enthalten mindestens einen Kreis oder sind nicht zusammenhängend.

Proposition 5.2.4 (Charakterisierung von Bäumen)**.** *Sei X ein nicht-leerer Graph. Dann sind äquivalent:*

1. *Der Graph X ist ein Baum.*
2. *Für alle Knoten v, w von X gilt: Es gibt genau einen Weg in X von v nach w.*

Beweis. Zu 1. $\Longrightarrow$ 2.*:* Sei X ein Baum. Dann erfüllt X die zweite Bedingung, denn: Seien v und w Knoten von X.

- *Existenz von Wegen.* Da X als Baum zusammenhängend ist, gibt es mindestens einen Weg in X von v nach w.
- *Eindeutigkeit von Wegen.* Seien $(v_0, \ldots, v_n)$ und $(w_0, \ldots, w_m)$ Wege in X von v nach w. *Angenommen*, diese beiden Wege wären verschieden. Dann gibt es einen kleinsten Index j mit $v_j \neq w_j$. Wegen $v_n = w = w_m$ gibt es außerdem kleinste Indizes $k \in \{j+1, \ldots, n\}$, $\ell \in \{j+1, \ldots, m\}$ mit $v_k = w_\ell$. Dann bildet

  $$(v_j, v_{j+1}, \ldots, v_k = w_\ell, w_{\ell-1}, \ldots, w_{j+1})$$

  einen Kreis in X, im Widerspruch dazu, dass X ein Baum ist. Also stimmen die beiden Wege überein.

Zu 2. $\Longrightarrow$ 1.*:* Sei umgekehrt die zweite Bedingung für X erfüllt. Dann ist X ein Baum, denn X ist nicht-leer und besitzt die folgenden Eigenschaften:

- Der Graph X ist zusammenhängend, denn zwischen je zwei Knoten gibt es einen Weg (sogar genau einen).
- Der Graph X enthält keine Kreise, denn: *Angenommen*, $(v_0, \ldots, v_n)$ (mit $n \geq 2$) wäre ein Kreis in X. Dann wären $(v_0, v_n, v_{n-1}, \ldots, v_2, v_1)$ und (v_0, v_1) zwei verschiedene Wege von v_0 nach v_1, im Widerspruch zur Eindeutigkeit in der zweiten Bedingung. Also enthält X keine Kreise. □

5.2.2 Binäre Wurzelbäume

In den Anwendungen kommen aus diversen Gründen oft standardisierte Baumstrukturen zum Einsatz, zum Beispiel Binärbäume.

Definition 5.2.5 (binärer Wurzelbaum)**.** Ein *(ungerichteter, ungeordneter) binärer Wurzelbaum* ist ein Paar (T, v), bestehend aus einem Baum T und einem Knoten v (genannt *Wurzel*) von T mit folgender Eigenschaft:

- Es gilt $\deg_T v \in \{0, 2\}$ und
- für alle Knoten w von T mit $w \neq v$ gilt $\deg_T w \in \{1, 3\}$.

Bemerkung 5.2.6 (Vorgänger, Nachfolger)**.** Alle „inneren“ Knoten in einem binären Wurzelbaum haben also einen „Vorgänger“ (englisch: *parent*; das ist der eindeutig bestimmte Nachbar, der auf dem(!) Weg zur Wurzel liegt) und zwei „Nachfolger“ (englisch: *children*; das sind die Nachbarn, die nicht der Vorgänger sind).

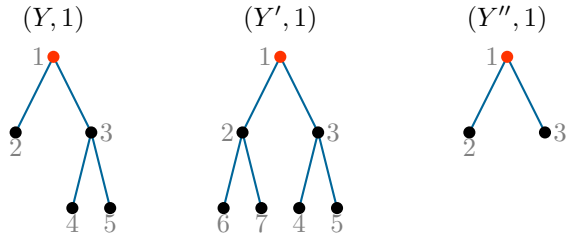


Abbildung 5.4.: binäre Wurzelbäume (Beispiel 5.2.7)

Beispiel 5.2.7. Die Bäume

$$Y := \big(\{1,\dots,5\}, \{\{1,2\},\{1,3\},\{3,4\},\{3,5\}\}\big)$$
$$Y' := \big(\{1,\dots,7\}, \{\{1,2\},\{1,3\},\{2,6\},\{2,7\},\{3,4\},\{3,5\}\}\big)$$
$$Y'' := \big(\{1,\dots,3\}, \{\{1,2\},\{1,3\}\}\big)$$

bilden jeweils mit der Wurzel 1 binäre Wurzelbäume $(Y,1)$, $(Y',1)$, $(Y'',1)$. Diese sind in Abbildung 5.4 veranschaulicht. Es ist dabei üblich, die Wurzel oben zu zeichnen und die Blätter unten.

Definition 5.2.8 (Höhe eines binären Wurzelbaums)**.** Sei (T,v) ein endlicher binärer Wurzelbaum. Die *Höhe* von (T,v) ist die größte Länge eines Weges in T zur Wurzel v.

Beispiel 5.2.9. Für die binären Wurzelbäume aus Beispiel 5.2.7 gilt:

- Der binäre Wurzelbaum Y hat die Höhe ? 2.
- Der binäre Wurzelbaum Y hat die Höhe ? 2.
- Der binäre Wurzelbaum Y hat die Höhe ? 1.

Proposition 5.2.10 (Höhe/Knoten binärer Wurzelbäume)**.** *Sei (T,v) ein endlicher binärer Wurzelbaum und sei n die Höhe von T. Dann besitzt T höchstens $2^{n+1}-1$ Knoten.*

Hierbei bezeichnen wir mit „-1“ den Vorgänger der entsprechenden natürlichen Zahl (welcher in den vorliegenden Fällen immer existiert).

Beweis (von Proposition 5.2.10). Wir beweisen die Behauptung induktiv über die Höhe endlicher binärer Wurzelbäume:

- *Induktionsanfang.* Ist (T,v) ein endlicher binärer Wurzelbaum der Höhe (höchstens) 0, so ist $T = (\{v\},\emptyset)$. Inbesondere hat T genau einen Knoten. Wegen $1 = 2^1 - 1$ zeigt dies die Behauptung für Höhe 0.

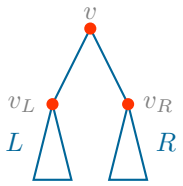


Abbildung 5.5.: Induktive Zerlegung von binären Wurzelbäumen

- *Induktionsvoraussetzung.* Sei $n \in \mathbb{N}$ und die Behauptung gelte für alle endlichen binären Wurzelbäume der Höhe höchstens n.
- *Induktionsschritt.* Wir zeigen, dass dann die Behauptung auch für alle binären Wurzelbäume der Höhe (höchstens) $n+1$ gilt: Sei (T, v) ein binärer Wurzelbaum der Höhe $n+1$ und sei $T = (V, E)$. Seien v_L und v_R die beiden Nachbarn von v (diese existieren, da T mindestens Höhe 1 besitzt). Dann können wir T wie folgt „disjunkt“ zerlegen

$$\begin{aligned} V &= V_L \cup \{v\} \cup V_R \\ E &= E_L \cup \big\{\{v, v_L\}, \{v, v_R\}\big\} \cup E_R, \end{aligned}$$

  wobei $(L := (V_L, E_L), v_L)$ und $(R := (E_R, V_R), v_R)$ binäre Wurzelbäume sind. Genauer enthält V_L genau die Knoten von T, bei denen v_L auf dem Weg zur Wurzel liegt (und analog für V_R); die Menge E_L enthält alle T-Kanten zwischen Knoten aus V_L (und analog für E_R). Diese Situation ist in Abbildung 5.5 dargestellt.

  Nach Konstruktion haben dabei die binären Wurzelbäume (L, v_L) und (R, v_R) höchstens die Höhe n (nachrechnen!). Mit der Induktionsvoraussetzung erhalten wir somit

$$\begin{aligned} \#V &= \#V_L + 1 + \#V_R && \text{(disjunkte Zerlegung)} \\ &\leq 2^n - 1 + 1 + 2^n - 1 && \text{(Induktionsvoraussetzung)} \\ &= 2^{n+1} - 1, && \text{(elementare Arithmetik in } \mathbb{N}) \end{aligned}$$

  wie behauptet. □

Ausblick 5.2.11. Viele Such- und Sortieralgorithmen verwenden als unterliegende kombinatorische Struktur für die Datenstrukturen binäre Bäume. Die Abschätzung aus Proposition 5.2.10 ist eine essentielle Beobachtung bei der Komplexitätsanalyse für solche Algorithmen.

Ausblick 5.2.12. In der Informatik bietet es sich an, (endliche) binäre Wurzelbäume induktiv zu definieren/konstruieren. Man kann dann Aussagen wie in Proposition 5.2.10 durch Induktion über solche Baumstrukturen beweisen.

5.3 Modellierung

Anwendungsprobleme werden untersucht, indem sie zunächst in mathematischer Sprache modelliert werden; dann wendet man mathematische Argumente auf diese Situation an und erhält mathematische Schlussfolgerungen; im letzten Schritt versucht man, diese Schlussfolgerungen wieder in die Anwendungen zurückzuübersetzen (Abbildung 5.6).

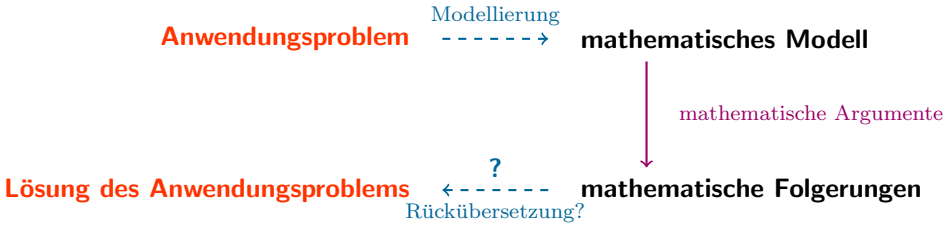


Abbildung 5.6.: Das Modellierungsprinzip, schematisch

Das Modell ist nur eine abstrakte und vereinfachte Approximation des eigentlichen Problems. Meistens kann man nicht *beweisen*, dass ein solches Modell „korrekt“ ist, da es im Normalfall keine formale und beweisbar korrekte Formalisierung des ursprünglichen Anwenundgsproblemes gibt. Insbesondere können auch die Schlussfolgerungen über das ursprüngliche Problem nicht stärker sein als das mathematische Modell. Man sollte sich auch dessen bewusst sein, dass das Modellierungsprinzip grundsätzlich besser dazu geeignet ist, zu zeigen, dass gewisse Dinge *nicht* möglich sind als zu zeigen, dass sie möglich sind.

Wir illustrieren diese Methode am folgenden Problem:

Problem 5.3.1. Kann jede Landkarte so mit vier Farben gefärbt werden, dass je zwei angrenzende Länder verschiedene Farben erhalten?

Dabei machen wir die folgenden Einschränkungen:

- Wir betrachten nur ebene Landkarten.
- Jedes Land ist zusammenhängend.
- „Angrenzend“ bedeutet, dass die beiden Länder eine gemeinsame Grenze positiver Länge besitzen müssen.

Wir modellieren diese Situation wie folgt:

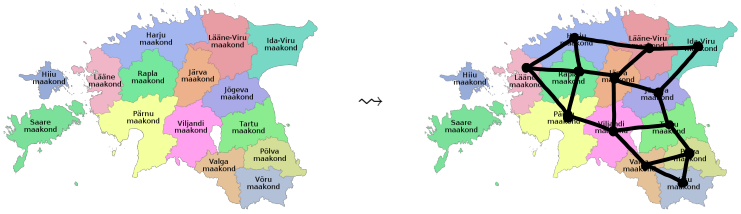


Abbildung 5.7.: Der duale Graph einer Landkarte (Estland; basierend auf Daten des Estonian Land Board (1.2006))

Modell 5.3.2. In der Situation von Problem 5.3.1 betrachten wir den folgenden Graphen (Abbildung 5.7):

- Knoten: Die Knoten entsprechen den Ländern.
- Kanten: Zwei Knoten sind genau dann durch eine Kante verbunden, wenn die zugehörigen Länder aneinander angrenzen (im Sinne von Problem 5.3.1.

Außerdem führen wir die folgenden Begriffe ein:

- Ist $X = (V, E)$ ein Graph, so ist eine 4-*Färbung* von X eine Abbildung $c\colon V \longrightarrow \{1, \ldots, 4\}$ mit folgender Eigenschaft:

$$\forall_{\{v,w\}\in E} \quad c(v) \neq c(w).$$

- Ein Graph heißt *planar*, wenn er „überkreuzungsfrei“ in der „Ebene“ „gezeichnet“ werden kann (dies kann mithilfe der Sprache der Topologie rigoros formalisiert werden). Man kann zeigen, dass folgendes gilt: Ist M ein Graph wie in Modell 5.3.2, so ist M planar.

Die Frage aus Problem 5.3.1 übersetzt sich somit in das folgende mathematische Problem: Besitzt jeder planare Graph eine 4-Färbung?

Satz 5.3.3 (Vierfarbensatz [1, 14])**.** *Jeder planare Graph besitzt eine* 4*-Färbung.*

Der Beweis des Vierfarbensatzes hat eine interessante Geschichte: Alle bekannten Beweise reduzieren das allgemeine Problem auf eine Fallunterscheidung mit viiiiiielen Fällen und beruhen dann auf einer computergestützten Behandlung dieser Fälle. Der ursprüngliche Beweis [1] war aufgrund dieser Computerunterstützung großer Skepsis ausgesetzt(!). Mittlerweile gibt es auch einen Coq-verifizierten formalisierten Beweis [14].

Insbesondere besitzt unser ursprüngliches Problem 5.3.1 eine positive Lösung: Jede Landkarte kann so mit vier Farben gefärbt werden, dass je zwei angrenzende Länder verschiedene Farben erhalten (mit den in Problem 5.3.1 formulierten Einschränkungen).

5.4 Anwendung*: Datenstrukturen

Die Entwicklung und Analyse von Algorithmen geht Hand in Hand mit der Entwicklung und Analyse geeigneter Datenstrukturen. In vielen Anwendungen bieten sich dabei Graphen (und all ihre Varianten) als unterliegendes kombinatorisches Modell an, siehe auch Fingerübung 5.1.7, Ausblick 5.1.20, Kapitel 5.3.

Bäume sind insbesondere bei effizienten Such- und Sortieralgorithmen hilfreich (Ausblick 5.2.11). Außerdem treten Bäume zum Beispiel im Compilerbau in Form von Syntaxbäumen auf (Beispiel 1.1.2).

5.5 Ausblick*: Implementation von Graphen

Es gibt verschiedene Möglichkeiten, Graphen zu implementieren. Die häufigsten sind:

- Pointerstrukturen und Adjazenzlisten (Jeder Knoten verweist auf (die Liste) seiner Nachbarn);
- Adjazenzmatrizen (Zeilen und Spalten der Matrix sind durch Knoten indiziert; die Nicht-0-Einträge der Matrix geben an, welche Knoten mit welchen Knoten durch Kanten verbunden sind);
- Inzidenzmatrizen (Zeilen/Spalten sind durch Knoten/Kanten indiziert; die Nicht-0-Einträge der Matrix geben an, welche Knoten in welchen Kanten enthalten sind).

Die verschiedenen Möglichkeiten haben unterschiedliche Komplexität in der Laufzeit und im Speicherverbrauch. In Anwendungen mit „großen“ sollte daher darauf geachtet werden, Implementationen zu wählen, die zu den auftretenden Operationen passen.

Bäume können als Spezialfälle von Graphen implementiert werden. In vielen Fällen ist es aber effizienter und übersichtlicher spezifischere Implementationen zu verwenden. Dafür eignen sich insbesondere entsprechende induktive Datentypen (Ausblick 5.2.12, Kapitel 4.4.1).

6

Relationen

Relationen sind ein abstraktes Konzept, das Abbildungen und (gerichtete) Graphen verallgemeinert. Wichtige Klassen von Relationen sind Äquivalenzrelationen und Ordnungen.

- Äquivalenzrelationen erlauben es, Objekte nach Ähnlichkeit ihrer Eigenschaften zusammenzufassen bzw. zu trennen. Insbesondere liefern sie ein wichtiges Hilfsmittel, um durch Quotientenbildung neue Strukturen zu konstruieren. Auf diese Weise kann man zum Beispiel die ganzen Zahlen, die rationalen Zahlen und die Modulo-Rechnung einführen.
- Ordnungen modellieren konsistente, gerichtete Vergleiche und treten daher insbesondere bei Sortier- und Suchverfahren auf. Wohlordnungen sind genau die Relationen, die Induktions- und Rekursionsprinzipien erlauben.

Wir führen die grundlegenden Begriffe und Konstruktionen ein und illustrieren diese an Beispielen.

Überblick über dieses Kapitel.

6.1 Relationen

Wir führen Relationen und das Standardvokabular für häufig auftretende Eigenschaften von Relationen ein.

Definition 6.1.1 (Relation)**.** Seien X und Y Mengen.

- Eine *Relation zwischen X und Y* ist eine Teilmenge von $X \times Y$.
- Eine *Relation auf X* ist eine Relation zwischen X und X, d.h. eine Teilmenge von $X \times X$.
- *Infix-Notation.* Ist $\square$ eine Relation zwischen X und Y, so schreibt man genau dann $x \mathrel{\square} y$, wenn $(x, y) \in \square$.

Beispiel 6.1.2 (triviale Relationen)**.** Sei X eine Menge.

- Die *leere Relation auf X* ist $\emptyset \subset X \times X$.
- Die *diagonale Relation auf X* ist $\Delta_X := \{(x, x) \mid x \in X\}$.
- Die *All-Relation auf X* ist $X \times X$.

Definition 6.1.3 (reflexiv, irreflexiv, symmetrisch, antisymmetrisch, transitiv)**.** Seien X eine Menge und sei $\square$ eine Relation auf X. Die Relation $\square$ heißt

- *reflexiv*, wenn gilt: $\forall_{x \in X} \quad x \mathrel{\square} x$
- *irreflexiv*, wenn gilt: $\forall_{x \in X} \quad \neg(x \mathrel{\square} x)$
- *symmetrisch*, wenn gilt: $\forall_{x,y \in X} \quad (x \mathrel{\square} y \implies y \mathrel{\square} x)$
- *antisymmetrisch*, wenn gilt: $\forall_{x,y \in X} \quad \big((x \mathrel{\square} y) \wedge (y \mathrel{\square} x) \implies x = y\big)$
- *transitiv*, wenn gilt: $\forall_{x,y,z \in X} \quad \big((x \mathrel{\square} y) \wedge (y \mathrel{\square} z) \implies x \mathrel{\square} z\big)$

Caveat 6.1.4. „Irreflexiv“ ist *nicht* dasselbe wie „nicht reflexiv“! „Antisymmetrisch“ ist *nicht* dasselbe wie „nicht symmetrisch“!

Anmerkung zum Lernen. Vergleichen Sie die Begriffsbildung für Relationen mit den Namen für Tautologien.

Fingerübung 6.1.5 (Nachbarschaft)**.** Welche Eigenschaft besitzt die Relation „ist Nachbar von“ auf der Menge aller Einwohner von Regensburg?

Beispiel 6.1.6 (Abbildungen als Relationen)**.** Seien X und Y Mengen und sei $f\colon X \longrightarrow Y$. Nach Definition (Definition 3.2.3) ist f eine Relation zwischen X und Y. Man kann Relationen also als verallgemeinerte Abbildungen auffassen oder Abbildungen als spezielle Relationen.

Beispiel 6.1.7 (gerichtete Graphen als Relationen)**.** Sei X eine Menge. Ist $\square$ eine Relation auf X, so ist $(X, \square)$ ein gerichteter Graph. Umgekehrt definiert die Kantenmenge eines gerichteten Graphen eine Relation auf der Knotenmenge. Gerichtete Graphen und Relationen sind also zwei verschiedene Blickwinkel auf „dasselbe“ Konzept. Je nach Anwendung und Eigenschaften kann eine der beiden Varianten naheliegender sein.

Ausblick 6.1.8 (relationale Datenbanken)**.** Relationale Datenbanken bestehen aus einer Menge an (mehrstelligen) Relationen (genannt *Tabellen*). Auf diesen Relationen wird durch Anfragen zugegriffen. Dabei wird zum Beispiel auf ausgewählte Komponenten der mehrstelligen Relationen projiziert oder es werden Tabellen entlang gewisser Komponenten fusioniert (sog. *Join*).

Weitere, konkrete, Beispiele für Relationen betrachten wir in Kapitel 6.2, Kapitel 6.3 und in Kapitel 6.4.

Proposition 6.1.9 (Abschlüsse)**.** *Sei X eine Menge und sei $\square$ eine Relation auf X. Dann gilt:*

1. *Die Relation*
$$\overline{\square} := \big\{(x,y) \;\big|\; (x \square y) \vee (y \square x)\big\}$$
auf X ist symmetrisch und $\square \subset \overline{\square}$. Außerdem gilt: Für alle symmetrischen Relationen R auf X mit $\square \subset R$ folgt $\overline{\square} \subset R$.

   D.h. $\overline{\square}$ ist die bezüglich Inklusion kleinste symmetrische Relation auf X, die $\square$ enthält. Wir bezeichnen $\overline{\square}$ als symmetrischen Abschluss *von $\square$.*

2. *Die Relation*
$$\square^* := \big\{(x,y) \;\big|\; \exists_{n\in\mathbb{N}\setminus\{0\}} \quad \exists_{x_0,\ldots,x_n\in X} \quad (x_0 = x) \wedge (x_n = y) \wedge (\forall_{j\in\{0,\ldots,n-1\}} \quad x_j \square x_{j+1})\big\}$$
auf X ist transitiv und $\square \subset \square^*$. Außerdem gilt: Für alle transitiven Relationen R auf X mit $\square \subset R$ folgt $\square^* \subset R$.

   D.h. $\square^*$ ist die bezüglich Inklusion kleinste transitive Relation auf X, die $\square$ enthält. Wir bezeichnen $\square^*$ als transitiven Abschluss *von $\square$.*

Beweis. Wir beweisen stellvertretend die zweite Aussage; der Beweis der ersten Aussage ist ähnlich und etwas übersichtlicher/einfacher.

- Es gilt $\square \subset \square^*$, denn: Seien $x, y \in X$ mit $x \square y$. Dann zeigt die Folge x, y, dass $x \square^* y$.
- Die Relation $\square^*$ ist transitiv, denn: Seien $x, y, z \in X$ mit $x \square^* y$ und $y \square^* z$. Insbesondere gibt es $n, m \in \mathbb{N}$ und Folgen $x_0, \ldots, x_n$ bzw. $y_0, \ldots, y_m$, die zeigen, dass $x \square^* y$ bzw. $y \square^* z$ gilt; dabei ist $x_n = y = y_0$. Dann zeigt die zusammengesetzte Folge $x_0, \ldots, x_n = y_0, \ldots, y_m$, dass $x \square^* z$.

- Sei R eine transitive Relation mit $\square \subset R$. Dann gilt $\square^* \subset R$, denn: Seien $x, y \in X$ mit $x \,\square^*\, y$, etwa bezeugt durch eine Folge $x_0, \ldots, x_n$. Wegen $\square \subset R$ gilt insbesondere

$$\forall_{j \in \{0,\ldots,n-1\}} \quad x_j \, R \, x_{j+1}.$$

  Induktiv (über die Länge n) folgt mit der Transitivität von R somit, dass $x_0 \, R \, x_n$. Wegen $x_0 = x$ und $x_n = y$ zeigt dies, dass $x \, R \, y$. Also ist $\square^* \subset R$. □

6.2 Äquivalenzrelationen

Äquivalenzrelationen erlauben es, Objekte nach Ähnlichkeit bezüglich gewisser Eigenschaften zusammenzufassen bzw. zu trennen. Mithilfe der Quotientenbildung können wir aus Äquivalenzrelationen neue Strukturen konstruieren. Wir führen zunächst die abstrakten Begriffe ein und erklären dann, wie man die ganzen und rationalen Zahlen konstruieren kann und was Modulo-Rechnung ist.

6.2.1 Definition

Definition 6.2.1 (Äquivalenzrelation, Äquivalenzklasse)**.** Sei X eine Menge und sei $\sim$ eine Relation auf X.

- Die Relation $\sim$ ist eine *Äquivalenzrelation*, wenn sie reflexiv, symmetrisch und transitiv ist.
- Ist $\sim$ eine Äquivalenzrelation und ist $x \in X$, so bezeichnet man

$$[x]_\sim := \{y \mid y \sim x\} \subset X$$

  als *Äquivalenzklasse von x (bezüglich $\sim$)*. Ist die Äquivalenzrelation aus dem Kontext klar, so schreiben wir manchmal auch $[x]$ statt $[x]_\sim$.

Beispiel 6.2.2 (Nachbarschaft)**.** Die Relation „ist Nachbar von“ auf der Menge aller Einwohner von Regensburg ist *keine* Äquivalenzrelation, da sie nicht transitiv ist.

Beispiel 6.2.3 (Paritätsrelation)**.** Wir betrachten auf $\mathbb{N}$ die Relation

$$\sim_P := \big\{(x, x + 2 \cdot k) \;\big|\; x, k \in \mathbb{N}\big\} \cup \big\{(x + 2 \cdot k, x) \;\big|\; x, k \in \mathbb{N}\big\}.$$

Die Relation $\sim_P$ ist eine Äquivalenzrelation (nachrechnen!). Es gilt

$$[0]_{\sim_P} = \text{?} \quad \{0, 2, 4, \dots\} = \{x \in \mathbb{N} \mid x \text{ ist gerade}\}$$
$$[1]_{\sim_P} = \text{?} \quad \{1, 3, 5, \dots\} = \{x \in \mathbb{N} \mid x \text{ ist ungerade}\}.$$

6.2.2 Quotienten

Äquivalenzklassen fassen jeweils die Elemente zusammen, die bezüglich der betrachteten Äquivalenzrelation „ähnlich“ sind. Die Menge aller Äquivalenzklassen bildet den Quotienten.

Definition 6.2.4 (Quotient einer Äquivalenzrelation)**.** Sei X eine Menge und sei $\sim$ eine Äquivalenzrelation auf X. Dann schreibt man

$$X/\sim := \left\{ [x]_\sim \;\middle|\; x \in X \right\} \subset P(X)$$

für die Menge aller $\sim$-Äquivalenzklassen und nennt diese Menge den *Quotienten von X nach $\sim$*.

Proposition 6.2.5 (Eigenschaften von Äquivalenzklassen)**.** *Sei X eine Menge und sei $\sim$ eine Äquivalenzrelation auf X. Dann gilt:*

1. *Sind $x, y \in X$, so folgt $[x]_\sim = [y]_\sim$ oder $[x]_\sim \cap [y]_\sim = \emptyset$.*
2. *Es gilt*
$$X = \bigcup (X/\sim) = \left\{ x \mid \exists_{c \in X/\sim} \quad x \in c \right\}.$$

Beweis. Zu 1. Seien $x, y \in X$ und es gelte $[x]_\sim \cap [y]_\sim \neq \emptyset$; dann gilt $[x]_\sim = [y]_\sim$, denn: Wegen $[x]_\sim \cap [y]_\sim \neq \emptyset$ gibt es ein $z \in X$ mit $z \in [x]_\sim \cap [y]_\sim$. Sei $x' \in [x]_\sim$. Wir zeigen, dass $x' \in [y]_\sim$:

Nach Definition gilt $z \sim x$ und $z \sim y$ sowie $x' \sim x$. Da $\sim$ als Äquivalenzrelation symmetrisch ist, folgt $x \sim z$. Da $\sim$ als Äquivalenzrelation transitiv ist, erhalten wir aus $x' \sim x$ und $x \sim z$, dass $x' \sim z$ gilt. Anwendung von Transitivität auf $x' \sim z$ und $z \sim y$ liefert $x' \sim y$. Also ist $x' \in [y]_\sim$.

Analog zeigt man, dass $[y]_\sim \subset [x]_\sim$ gilt.

Zu 2. Es ist $X \subset \bigcup (X/\sim)$, denn aufgrund der Reflexivität von $\sim$ gilt $x \in [x]_\sim$ für alle $x \in X$.

Umgekehrt gilt $\bigcup (X/\sim) \subset X$, da jede Äquivalenzklasse von $\sim$ eine Teilmenge von X ist. □

Beispiel 6.2.6 (Paritätsrelation)**.** Für die Paritätsrelation auf $\mathbb{N}$ (Beispiel 6.2.3) erhalten wir $[0]_{\sim_P} = [2]_{\sim_P} = \dots$ etc. und

$$\mathbb{N}/\sim_P = \text{?} \quad \left\{ [0]_{\sim_P}, [1]_{\sim_P} \right\}.$$

Fingerübung 6.2.7 (Mannschaftssport)**.** Was sind die Äquivalenzklassen der Äquivalenzrelation(!) „spielt im Moment für denselben Verein wie“ auf der Menge alle Profifußballer? Was ist der Quotient?

6.2.3 Die ganzen Zahlen

Im allgemeinen ist es nicht möglich, für natürliche Zahlen $a, b \in \mathbb{N}$, eine Lösung $x \in \mathbb{N}$ für die Gleichung

$$x + a = b$$

zu finden; zum Beispiel gibt es *kein* $x \in \mathbb{N}$ mit $x+1 = 0$. Wir werden daher $\mathbb{N}$ um Lösungen solcher Gleichungen erweitern. Dies führt zu den ganzen Zahlen.

Wir wollen die ganzen Zahlen konstruieren, indem wir Ausdrücke der Form $m - n$ mit $n, m \in \mathbb{N}$ betrachten. Dabei tritt die Schwierigkeit auf, dass mehrere solcher Ausdrücke dieselbe ganze Zahl repräsentieren sollen. Zum Beispiel würden wir erwarten, dass $0 - 1$ und $1 - 2$ dieselbe ganze Zahl beschreiben. Diese Herausforderung lässt sich mit einer geeigneten Äquivalenzrelation und der zugehörigen Quotientenkonstruktion meistern.

Wir definieren daher eine Relation auf $\mathbb{N} \times \mathbb{N}$. Gemäß der obigen Idee wollen wir dabei (m, n) und $(m', n') \in \mathbb{N} \times \mathbb{N}$ als „äquivalent“ ansehen, wenn „$m - n = m' - n'$“ gilt. Da wir aber strenggenommen noch nicht über diese Differenzen sprechen können, definieren wir die Relation indirekt durch die Beschreibung „$m + n' = m' + n$“.

Beispiel 6.2.8 (Summenrelation)**.** Wir betrachten auf $\mathbb{N} \times \mathbb{N}$ die Relation

$$\sim_+ := \big\{ \big((m, n), (m', n')\big) \mid m, n, m', n' \in \mathbb{N},\ m + n' = m' + n \big\}.$$

Die Relation $\sim_+$ ist eine Äquivalenzrelation, denn:

- *Reflexivität.* Für alle $(m, n) \in \mathbb{N} \times \mathbb{N}$ ist $m + n = m + n$, und damit $(m, n) \sim_+ (m, n)$.
- *Symmetrie.* Dies folgt direkt aus der Symmetrie von Gleichheit.
- *Transitivität.* Seien $(m, n), (m', n'), (m'', n'') \in \mathbb{N} \times \mathbb{N}$ mit $(m, n) \sim_+ (m', n')$ und $(m', n') \sim_+ (m'', n'')$. Dann gilt auch $(m, n) \sim_+ (m'', n'')$, denn: Es gilt

$$\begin{aligned}
m + n'' + n' &= m + n' + n'' && (? \text{ Kommutativität/Assoziativität}) \\
&= m' + n + n'' && (? \text{ da } (m, n) \sim_+ (m', n')) \\
&= m' + n'' + n && (? \text{ Kommutativität/Assoziativität}) \\
&= m'' + n' + n && (? \text{ da } (m'n') \sim_+ (m'', n'')) \\
&= m'' + n + n'; && (? \text{ Kommutativität/Assoziativität})
\end{aligned}$$

  mit den Kürzungsregeln (Proposition 4.2.5) folgt daraus $m + n'' = m'' + n$, wie gewünscht.

Proposition und Definition 6.2.9 (die ganzen Zahlen)**.** *Sei $\sim_+$ die Äquivalenzrelation aus Beispiel 6.2.8. Wir definieren die Menge der* ganzen Zahlen *durch*

$$\mathbb{Z} := (\mathbb{N} \times \mathbb{N})/\sim_+.$$

Dann gilt:

1. Einbettung der natürlichen Zahlen. *Die Abbildung $\mathbb{N} \longrightarrow \mathbb{Z}$, $n \longmapsto (n, 0)$ ist injektiv.*

   Ist $[(m, n)] \in \mathbb{Z}$, so gibt es ein $k \in \mathbb{N}$ mit $[(m, n)] = [(k, 0)]$ oder $[(m, n) = [(0, k)]$.

2. Addition und Multiplikation. *Die Abbildungen*

$$\begin{aligned} +\colon \mathbb{Z} \times \mathbb{Z} &\longrightarrow \mathbb{Z} \\ \big([(m,n)], [(m',n')]\big) &\longmapsto \big[(m+m', n+n')\big] \\ \cdot\colon \mathbb{Z} \times \mathbb{Z} &\longrightarrow \mathbb{Z} \\ \big([(m,n)], [(m',n')]\big) &\longmapsto \big[(m\cdot m' + n\cdot n', m\cdot n' + n\cdot m')\big] \end{aligned}$$

   (Multiplikation von Differenzen!)

   sind wohldefiniert.

3. *Die Addition auf $\mathbb{Z}$ ist assoziativ und kommutativ und hat $[(0, 0)]$ als neutrales Element.*

   Die Multiplikation auf $\mathbb{Z}$ ist assoziativ und kommutativ und hat $[(1, 0)]$ als neutrales Element.

   Es gilt das Distributivgesetz: Für alle $x, y, z \in \mathbb{Z}$ ist $x\cdot(y+z) = x\cdot y+x\cdot z$.

4. Lösbarkeit additiver Gleichungen. *Seien $a, b \in \mathbb{Z}$. Dann gibt es ein $x \in \mathbb{Z}$ mit*

$$x + a = b.$$

Beweis. All diese Eigenschaften folgen aus elementaren Rechnungen (nachrechnen!).

Wohldefiniertheit im zweiten Teil bedeutet das folgende: Wählt man auf der linken Seite verschiedene Repräsentanten *derselben* Äquivalenzklasse, so erhält man trotzdem dasselbe Ergebnis.

Zur Lösbarkeit additiver Gleichungen: Seien $(m_a, n_a), (m_b, n_b) \in \mathbb{N} \times \mathbb{N}$ Repräsentanten von a bzw. b. Dann ist

$$x = (m_b + n_a, m_a + n_b)$$

eine solche Lösung (nachrechnen). □

Notation 6.2.10. Da die obige Abbildung $\mathbb{N} \longrightarrow \mathbb{Z}$ injektiv ist, fassen wir $\mathbb{N}$ als Teilmenge von $\mathbb{Z}$ auf und schreiben ganze Zahlen in der Form n oder $-n$ mit $n \in \mathbb{N}$.

6.2.4 Die rationalen Zahlen

Analog zur Konstruktion der ganzen Zahlen aus den natürlichen Zahlen können wir die rationalen Zahlen aus den ganzen Zahlen als Äquivalenzklassen „formaler Brüche“ konstruieren, um multiplikative Gleichungen zu lösen.

Beispiel 6.2.11 (die Quotientenrelation)**.** Wir betrachten auf $\mathbb{Z} \times (\mathbb{Z} \setminus \{0\})$ die Relation

$$\sim_\bullet := \big\{ \big((x,y),(x',y')\big) \;\big|\; x,y,x',y' \in \mathbb{Z},\ x \cdot y' = x' \cdot y \big\}.$$

Diese Relation ist eine Äquivalenzrelation: Dies folgt aus einer Rechnung analog zu Beispiel 6.2.8; für den Beweis der Transitivität verwendet man entsprechende Kürzungsregeln für die Multiplikation auf $\mathbb{Z}$.

Proposition und Definition 6.2.12 (die rationalen Zahlen)**.** *Sei $\sim_\bullet$ die Äquivalenzrelation aus Beispiel 6.2.11. Wir definieren die Menge der* rationalen Zahlen *durch*

$$\mathbb{Q} := \big(\mathbb{Z} \times (\mathbb{Z} \setminus \{0\})\big)/{\sim_\bullet}.$$

Dann gilt:

1. Einbettung der ganzen Zahlen. *Die Abbildung $\mathbb{Z} \longrightarrow \mathbb{Q}$, $x \longmapsto (x,1)$ ist injektiv.*

2. Addition und Multiplikation. *Die Abbildungen*

$$\begin{aligned} +\colon \mathbb{Q} \times \mathbb{Q} &\longrightarrow \mathbb{Q} \\ \big([(x,y)],[(x',y')]\big) &\longmapsto \big[(x \cdot y' + x' \cdot y, y \cdot y')\big] \quad \textit{(Addition von Brüchen!)} \\ \cdot\colon \mathbb{Q} \times \mathbb{Q} &\longrightarrow \mathbb{Q} \\ \big([(x,y)],[(x',y')]\big) &\longmapsto \big[(x \cdot x', y \cdot y')\big] \end{aligned}$$

   sind wohldefiniert.

3. *Die Addition auf $\mathbb{Q}$ ist assoziativ und kommutativ und hat $[(0,1)]$ als neutrales Element.*

   Die Multiplikation auf $\mathbb{Q}$ ist assoziativ und kommutativ und hat $[(1,1)]$ als neutrales Element.

   Es gilt das Distributivgesetz: Für alle $x,y,z \in \mathbb{Q}$ ist $x \cdot (y+z) = x \cdot y + x \cdot z$.

4. Lösbarkeit additiver Gleichungen. *Seien $a,b \in \mathbb{Q}$. Dann gibt es ein $x \in \mathbb{Q}$ mit*

$$x + a = b.$$

5. Lösbarkeit multiplikativer Gleichungen. *Seien $a, b \in \mathbb{Q}$ mit $a \neq [(0,1)]$. Dann gibt es ein $x \in \mathbb{Q}$ mit*

$$a \cdot x = b.$$

Beweis. Die Argumente sind analog zu den entsprechenden Beweisen in der Konstruktion der ganzen Zahlen (Proposition 6.2.9). □

Notation 6.2.13. Da die obige Abbildung $\mathbb{Z} \longrightarrow \mathbb{Q}$ injektiv und mit der Addition bzw. Multiplikation auf $\mathbb{Z}$ bzw. $\mathbb{Q}$ verträglich ist, fassen wir $\mathbb{Z}$ im folgenden als Teilmenge von $\mathbb{Q}$ auf. Die Äquivalenzklasse $[(x,y)]_{\sim_\bullet}$ notieren wir dabei als x/y bzw. $\frac{x}{y}$.

6.2.5 Modulo-Rechnung

In vielen Anwendungen möchte man mit „periodischen“ ganzen Zahlen rechnen, z.B. bei Uhrzeiten (Periode 12 bzw. 24) oder Wochentagen (Periode 7). Dies lässt sich gut mit geeigneten Äquivalenzrelationen modellieren:

Definition 6.2.14 (Teilbarkeit)**.** Seien $x, y \in \mathbb{Z}$. Dann ist x *ein Teiler von* y, wenn es ein $k \in \mathbb{Z}$ mit $y = k \cdot x$ gibt. Man schreibt in diesem Fall auch $x \mid y$.

Beispiel 6.2.15. Sei $n \in \mathbb{N}$. Dann ist

$$\sim_n := \big\{(x,y) \in \mathbb{Z} \times \mathbb{Z} \;\big|\; n \text{ teilt } x - y\big\}$$

eine Äquivalenzrelation auf $\mathbb{Z}$ (Übungsaufgabe). Gilt $x \sim_n y$, so schreibt man auch $x \equiv y \pmod{n}$ (lies: „x ist y modulo n“).

Zum Beispiel ist $5 \sim_{12} 17$ bzw. $5 \equiv 17 \pmod{12}$.

Proposition und Definition 6.2.16 (Modulo-Rechnung)**.** *Sei $n \in \mathbb{N}$ mit $n \neq 0$ und sei $\sim_n$ die Äquivalenzrelation aus Beispiel 6.2.15. Wir definieren*

$$\mathbb{Z}/n := \mathbb{Z}/\sim_n.$$

Dann gilt:

1. *Es besteht $\mathbb{Z}/n$ genau aus den n Äquivalenzklassen $[0], \ldots, [n-1]$.*
2. Addition und Multiplikation. *Die Abbildungen*

$$\begin{aligned} +\colon \mathbb{Z} \times \mathbb{Z} &\longrightarrow \mathbb{Z} \\ \big([x],[y]\big) &\longmapsto [x+y] \\ \cdot\colon \mathbb{Z} \times \mathbb{Z} &\longrightarrow \mathbb{Z} \\ \big([x],[y]\big) &\longmapsto [x \cdot y] \end{aligned}$$

sind wohldefiniert.

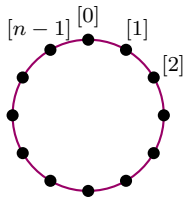


Abbildung 6.1.: Die Elemente von $\mathbb{Z}/n$, schematisch

3. *Die Addition auf $\mathbb{Z}/n$ ist assoziativ und kommutativ und hat $[0]$ als neutrales Element.*

   Die Multiplikation auf $\mathbb{Z}/n$ ist assoziativ und kommutativ und hat $[1]$ als neutrales Element.

   Es gilt das Distributivgesetz.

4. *Es gilt die Lösbarkeit additiver Gleichungen.*

Beweis. Die Argumente sind analog zu den entsprechenden Beweisen in der Konstruktion der ganzen Zahlen (Proposition 6.2.9). □

Beispiel 6.2.17 (eine Wurzel aus 2)**.** Es gibt ein $x \in \mathbb{Z}/7$ mit $x^2 = [2]$ (in $\mathbb{Z}/7$), denn: Für $x = ?$ $[3]$ gilt in $\mathbb{Z}/7$:

$$x^2 = x \cdot x = ? \quad [3] \cdot [3] = [3 \cdot 3] = [9] = [2].$$

Anmerkung zum Lernen. Es kann evtl. hilfreich sein, sich für $n \in \mathbb{N} \setminus \{0\}$ die Elemente von $\mathbb{Z}/n$, ähnlich zu einer Analoguhr, zyklisch angeordnet vorzustellen (Figure 6.1).

Beispiel 6.2.18 (Bits)**.** In der Informatik spielt für $n \in \mathbb{N} \setminus 0$ die Arithmetik in $\mathbb{Z}/2^n$ eine große Rolle. Insbesondere kann man $\mathbb{Z}/2$ als die Menge der Bits $[0]$ und $[1]$ auffassen.

Ausblick 6.2.19 (RSA-Verschlüsselung)**.** Das asymmetrische Verschlüsselungsverfahren RSA [27] beruht auf speziellene arithmetischen Eigenschaften von $\mathbb{Z}/m$, wobei $m = p \cdot q$ das Produkt zweier „großer“ (verschiedener) Primzahlen p und q ist.

6.3 Ordnungen

Ordnungen modellieren konsistente, gerichtete Vergleiche. Ordnungen erleichtern in der Arithmetik die Suche nach Lösungen und in der Informatik die systematische und effiziente Organisation von Daten und Abläufen.

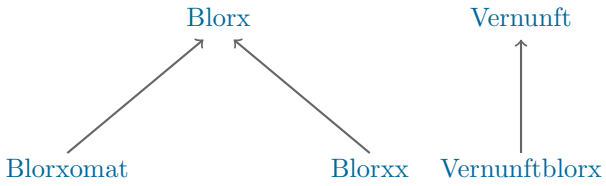


Abbildung 6.2.: ein Auschnitt der Ordnung aus Beispiel 6.3.5

Definition 6.3.1 (partielle Ordnung, totale Ordnung)**.** Sei X eine Menge.

- Eine *partielle Ordnung auf* X ist eine Relation auf X, die reflexiv, transitiv und anti-symmetrisch ist.
- Ist $\leq$ eine partielle Ordnung auf X, so ist $(X, \leq)$ eine *partiell geordnete Menge*.
- Eine partielle Ordnung $\leq$ auf X ist eine *totale Ordnung*, wenn

$$\forall_{x,y \in X} \quad (x \leq y) \vee (y \leq x).$$

- Ist $\leq$ eine totale Ordnung auf X, so ist $(X, \leq)$ eine *total geordnete Menge*.

Caveat 6.3.2. In der Literatur wird manchmal stattdessen die strikte (irreflexive) Version von partiellen/totalen Ordnungen betrachtet.

Beispiel 6.3.3 (die Ordnung der natürlichen Zahlen)**.** Die gewöhnliche Ordnung

$$\leq := \left\{(m, n) \in \mathbb{N} \times \mathbb{N} \;\middle|\; \exists_{k \in \mathbb{N}} \quad m + k = n\right\}$$

auf $\mathbb{N}$ (Bemerkung 4.2.3) ist eine totale Ordnung auf $\mathbb{N}$ (nachrechnen!).

Beispiel 6.3.4 (Inklusionsrelation)**.** Sei X eine Menge. Dann ist $P(X)$ durch $\subset$ partiell geordnet. Im allgemeinen ist diese partielle Ordnung *keine* totale Ordnung auf $P(X)$.

Beispiel 6.3.5 (Ordnungen auf Wörtern)**.** Die Relation „. . . enthält . . . als Präfix“ ist eine partielle Ordnung auf der Menge aller Wörter. Zum Beispiel ist

$$\text{Blorxomat} \leq \text{Blorx}.$$

Diese partielle Ordnung ist *keine* totale Ordnung, denn es gilt weder Blorx $\leq$ Vernunft noch Vernunft $\leq$ Blorx.

Beispiel 6.3.6 (Ordnungen auf Wörtern II)**.** Die lexikographische Ordnung ist eine totale Ordnung auf der Menge aller Wörter. Bezüglich der lexikographischen Ordnung gilt Blorx $\leq$ Blorxomat und Blorxomat $\leq$ Vernunft.

Definition 6.3.7 (minimale/maximale/kleinste/größte Elemente, untere/obere Schranke)**.** Sei $(X, \leq)$ eine partiell geordnete Menge.

- Ein Element $m \in X$ ist *maximal*, wenn folgendes gilt: Für alle $x \in X$ mit $m \leq x$ folgt bereits $x = m$.
- Sei $A \subset X$. Ein Element $m \in X$ ist eine *obere Schranke von* A, wenn folgendes gilt: Für alle $x \in A$ ist $x \leq m$.
- Ein Element $m \in X$ ist ein *größtes Element*, wenn es eine obere Schranke von X ist, d.h., wenn folgendes gilt: Für alle $x \in X$ ist $x \leq m$.

Analog definiert man *minimale* bzw. *kleinste* Elemente und *untere Schranken*.

Definition 6.3.8 (Wohlordnung)**.** Eine totale Ordnung $\leq$ auf einer Menge X ist eine *Wohlordnung*, wenn folgendes gilt: Für alle $A \subset X$ mit $A \neq \emptyset$ enthält A ein $\leq$-kleinstes Element.

Beispiel 6.3.9 (natürliche Zahlen)**.** Die gewöhnliche Ordnungsrelation auf den natürlichen Zahlen ist eine Wohlordnung. Diese Tatsache entspricht dem Induktionsprinzip der natürlichen Zahlen.

Beispiel 6.3.10 (die Ordnung der ganzen/rationalen Zahlen)**.** Die Relation

$$\leq := \left\{(x, y) \in \mathbb{Z} \times \mathbb{Z} \;\middle|\; \exists_{k \in \mathbb{N}} \quad x + k = y\right\}$$

auf $\mathbb{Z}$ ist eine totale Ordnung (nachrechnen!). Enthält $\mathbb{Z}$ ein $\leq$-kleinstes Element? Ja Nein Die korrekte Antwort ist „Nein“. denn für jedes $m \in \mathbb{Z}$ ist $m \not\leq m - 1$. × Insbesondere ist die gewöhnliche Ordnung auf $\mathbb{Z}$ *keine* Wohlordnung.

Analog ist auch

$$\leq := \left\{(x, y) \in \mathbb{Q} \times \mathbb{Q} \;\middle|\; \exists_{k \in \mathbb{N}} \quad \exists_{m \in \mathbb{N} \setminus \{0\}} \quad x + \frac{k}{m} = y\right\}$$

auf $\mathbb{Q}$ eine totale Ordnung, aber *keine* Wohlordnung (nachrechnen!).

Ausblick 6.3.11 (Wohlordnungen und Induktion)**.** Wohlordnungen sind genau die Ordnungen, die ein Induktions- und Rekursionsprinzip liefern [28].

Ausblick 6.3.12 (Galois-Korrespondenzen)**.** Galois-Korrespondenzen liefern eine Möglichkeit, zwischen verschiedenen Ordnungen hin- und herzuübersetzen. Historisch stammt dieser Begriff aus der Algebra, nämlich der Galois-Theorie. In der Informatik treten Galois-Korrespondenzen zum Beispiel bei der abstrakten Interpretation von Programmiersprachen und im Zusammenspiel zwischen Syntax und Semantik auf. In der modernen Mathematik spielt das verallgemeinerte Konzept der adjungierten Funktoren in vielen Gebieten eine wichtige Rolle.

6.4 Anwendung*: Reduktion und Evaluation

Relationen bieten sich insbesondere an, um Ableitungs- oder Transformationsprozesse zu beschreiben, zum Beispiel:

- Übergangsrelationen von endlichen Automaten, Kellerautomaten und Turingmaschinen;
- Reduktion- und Evaluationsrelationen für Semantiken von Programmiersprachen;
- Typregeln in Typsystemen;
- Schlussregeln in Beweiskalkülen
- ...

Oft wird in solchen Situationen auch zunächst nur eine Relation für einen einzelnen Schritt definiert und dann zum transitiven Abschluss (Proposition 6.1.9) übergegangen.

6.5 Ausblick*: Implementation geordneter Daten

Bei der Speicherung und Bearbeitung großer Datenmengen sind oft große Mengen an Daten der Form

$$(\text{Schlüssel}, \text{zugeordnete Daten})$$

zu verwalten. Falls die Menge der Schlüssel eine totale Ordnung besitzt, kann diese Ordnung verwendet werden, um Datenstrukturen zu konstruieren, die es erlauben, auf die Daten effizient zuzugreifen bzw. die Daten effizient zu manipulieren.

Solche Datenstrukturen sind insbesondere *B-Bäume* (und manche Varianten von *Hash-Tables*). Man macht sich dabei zunutze, dass man die geordneten Schlüssel in balancierten Bäumen arrangieren kann, so dass statt linearen Suchzeiten nur logarithmische Suchzeiten nötig sind. Dies beruht auf Beobachtungen, die ähnlich zu Proposition 5.2.10 sind.

7

Reelle und komplexe Zahlen

Die wesentlichen Eigenschaften der Grundrechenarten (Addition, Subtraktion, Multiplikation, Division) werden im Konzept des Körpers zusammengefasst. Zum Beispiel sind $\mathbb{Q}$ und $\mathbb{Z}/2$ Körper bezüglich der in Kapitel 6 definierten arithmetischen Operationen.

Wir werden zwei weitere wichtige Körper kennenlernen:

- Die reellen Zahlen bilden einen vollständigen angeordneten Körper, der die rationalen Zahlen umfasst.
- Die komplexen Zahlen bilden einen algebraisch abgeschlossenen Körper, der die reellen Zahlen umfasst.

Die reellen Zahlen treten in der Komplexitätsanalyse von Algorithmen und in der Modellierung (insbesondere bei geometrischen Problemen) auf; die komplexen Zahlen spielen in physikalischen Modellen und in der Stochastik eine wichtige Rolle.

Neben den abstrakten Eigenschaften werden wir auch kurz darauf eingehen, wie reelle und komplexe Zahlen in der Programmierung repräsentiert werden können.

Überblick über dieses Kapitel.

7.1 Körper

Die wesentlichen Eigenschaften der Grundrechenarten (Addition, Subtraktion, Multiplikation, Division) werden im Konzept des Körpers zusammengefasst:

Definition 7.1.1 (Körper)**.** Ein *Körper* ist ein Quintupel $(K, +, \cdot, 0, 1)$, bestehend aus

- einer Menge K,
- Abbildungen $+, \cdot : K \times K \longrightarrow K$,
- Elementen $0, 1 \in K$

mit folgenden Eigenschaften:

- *Addition.* Die Abbildung $+ \colon K \times K \longrightarrow K$ besitzt die folgenden Eigenschaften:
  - *Neutrales Element.* Für alle $x \in K$ gilt $x + 0 = x$ und $x = 0 + x$.
  - *Assoziativität.* Für alle $x, y, z \in K$ gilt $(x + y) + z = x + (y + z)$.
  - *Kommutativität.* Für alle $x, y \in K$ gilt $x + y = y + x$.
  - *Existenz von Inversen.* Für alle $x \in K$ gibt es ein $y \in K$ mit $x + y = 0$.
- *Multiplikation.* Die Abbildung $\cdot : K \times K \longrightarrow K$ besitzt die folgenden Eigenschaften:
  - *Neutrales Element.* Es ist $1 \neq 0$ und für alle $x \in K$ gilt $x \cdot 1 = x$ und $x = 1 \cdot x$.
  - *Assoziativität.* ? Für alle $x, y, z \in K$ gilt $(x \cdot y) \cdot z = x \cdot (y \cdot z)$.
  - *Kommutativität.* ? Für alle $x, y \in K$ gilt $x \cdot y = y \cdot x$.
  - *Existenz von Inversen.* Für alle $x \in K \backslash \{0\}$ gibt es ein $y \in K$ mit $x \cdot y = 1$.
- *Distributivgesetz.* Es gilt ?

$$\forall_{x,y,z \in K} \quad x \cdot (y + z) = x \cdot y + x \cdot z.$$

Falls die Daten $+$, $\cdot$, 0, 1 aus dem Kontext klar sind, sagt man auch „Sei K ein Körper ...“.

Beispiel 7.1.2 (die rationalen Zahlen)**.** Die rationalen Zahlen $\mathbb{Q}$ bilden bezüglich der in Proposition 6.2.12 definierten Addition und Multiplikation und den neutralen Elementen $0 = 0/1$ bzw. $1 = 1/1$ einen Körper (Proposition 6.2.12).

Beispiel 7.1.3 (die ganzen Zahlen)**.** Die ganzen Zahlen $\mathbb{Z}$ bilden bezüglich der in Proposition 6.2.9 definierten Addition und Multiplikation und den neutralen Elementen 0 bzw. 1 *keinen* Körper, denn es gibt *keine* ganze Zahl x mit $2 \cdot x = 1$.

Beispiel 7.1.4 (Bits)**.** Die ganzen Zahlen modulo 2 (also $\mathbb{Z}/2$) bilden bezüglich der in Proposition 6.2.16 definierten Addition und Multiplikation und den neutralen Elementen $[0]$ bzw. $[1]$ einen Körper (nachrechnen!).

Aber $\mathbb{Z}/4$ ist *kein* Körper, denn $[2]$ ist ungleich $[0]$ und besitzt *kein* multiplikatives Inverses (nachrechnen!).

Bemerkung 7.1.5 (Subtraktion/Division)**.** Sei $(K, +, \cdot, 0, 1)$ ein Körper. Dann gilt:

- Für alle $x \in K$ gibt es *genau ein* $y \in K$ mit $x + y = 0$. Man bezeichnet dann y als das *additive Inverse* von x und schreibt dafür „$-x$“. Außerdem schreiben wir für alle $x, y \in K$ kurz

  $$x - y := x + (-y).$$

  Diese Operation $-\colon K \times K \longrightarrow K$ wird als *Subtraktion* bezeichnet.

- Für alle $x \in K \setminus \{0\}$ gibt es *genau ein* $y \in K$ mit $x \cdot y = 1$. Man bezeichnet dann y als das *multiplikative Inverse* von y und schreibt dafür „y^{-1}“ oder „$1/y$“. Außerdem schreiben wir für alle $x, y \in K$ mit $y \neq 0$ kurz

  $$x/y := x \cdot y^{-1}.$$

  Diese Operation $/\colon K \times (K \setminus \{0\}) \longrightarrow K$ wird als *Division* bezeichnet.

Beweise für Eindeutigkeitsaussagen dieser Art werden wir in der Linearen Algebra I behandeln.

Ausblick 7.1.6 (Algebra)**.** In der (Linearen) Algebra beschäftigt man sich systematisch mit algebraischen Strukturen und den zugehörigen Methoden.

7.2 Die reellen Zahlen

Zusätzlich zur algebraischen Struktur eines Körpers besitzen die rationalen Zahlen auch eine kompatible Ordnungsstruktur (Beispiel 6.3.10); die rationalen Zahlen bilden auf diese Weise einen angeordneten Körper. Die rationalen Zahlen sind aber bezüglich ihrer Ordnung *nicht* „vollständig“ (Beispiel 7.2.7).

Insbesondere sind die rationalen Zahlen für viele analytische Probleme (Nullstellenberechnung, Extremwertprobleme, ...) *nicht* geeignet und müssen entsprechend erweitert werden. Dies führt zu den reellen Zahlen.

Die Frage, was man genau zu den rationalen Zahlen hinzufügen muss, um zu den reellen Zahlen zu gelangen, ist nicht so einfach zu beantworten wie bei den Übergängen $\mathbb{N} \rightsquigarrow \mathbb{Z}$ bzw. $\mathbb{Z} \rightsquigarrow \mathbb{Q}$. Der wesentliche Begriff in diesem Kontext ist „Vollständigkeit". Wir geben im folgenden einen kurzen Einblick; eine systematischere und detaillierter Untersuchung erfolgt im Rahmen der Analysis.

Definition 7.2.1 (angeordneter Körper)**.** Ein *angeordneter Körper* ist ein Paar $(K, \leq)$, bestehend aus einem Körper K und einer totalen Ordnung $\leq$ auf K, mit folgenden Eigenschaften:

- Für alle $x, x', y \in K$ gilt: Ist $x \leq x'$, so ist auch $x + y \leq x' + y$.
- Für alle $x, x', y \in K$ gilt: Ist $x \leq x'$ und $0 \leq y$, so ist auch $x \cdot y \leq x' \cdot y$.

Ein angeordneter Körper $(K, \leq)$ heißt *archimedisch*, wenn folgendes gilt (wobei die kanonische Abbildung $\mathbb{N} \longrightarrow K$ rekursiv via 0, 1 und + definiert ist):

$$\forall_{x \in K} \quad \exists_{n \in \mathbb{N}} \quad x \leq n.$$

Beispiel 7.2.2. Die rationalen Zahlen bilden bezüglich der gewöhnlichen Ordnung einen archimedischen angeordneten Körper (nachrechnen!). Man kann zeigen, dass es auf dem Körper $\mathbb{Z}/2$ keine Ordnung gibt, so dass ein angeordneter Körper entsteht (nachrechnen bzw. Bemerkung 7.2.10).

Notation 7.2.3. Ist $(X, \leq)$ eine partiell geordnete Menge, so führen wir die üblichen Abkürzungen ein: Für alle $x, y \in X$ schreiben wir

- $x \geq y$, falls $y \leq x$;
- $x < y$, falls $x \leq y$ und $x \neq y$;
- $x > y$, dalls $x \geq y$ und $x \neq y$.

Die Ordnungsstruktur ist für viele quantitative Betrachtungen beim Lösen von Gleichungen und in der Modellierung hilfreich.

Der Vollständigkeitsbegriff beruht auf Suprema, d.h. auf kleinsten oberen Schranken:

Definition 7.2.4 (Supremum/Infimum)**.** Sei $(X, \leq)$ eine partiell geordnete Menge und sei $A \subset X$ eine nicht-leere Teilmenge. Ein Element $s \in X$ ist ein *Supremum von* A, wenn folgendes gilt:

- Das Element s ist eine obere Schranke von A;
- für alle oberen Schranken t von A gilt $s \leq t$.

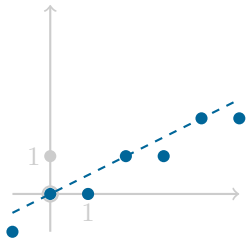


Abbildung 7.1.: Ein Quasimorphismus, der 1/2 repräsentiert

Analog definiert man *Infima* als größte untere Schranken.

Eine Teilmenge von X heißt *beschränkt*, wenn sie sowohl eine obere als auch eine untere Schranke in X besitzt.

Notation 7.2.5 (Supremum/Infimum)**.** Ist $(X, \leq)$ eine partiell geordnete Menge und besitzt $A \subset X$ ein Supremum bzw. Infimum in X, so sind Supremum bzw. Infimum eindeutig (nachrechnen!) und man schreibt dafür $\sup A$ bzw. $\inf A$.

Definition 7.2.6 (vollständiger angeordneter Körper)**.** Ein angeordneter Körper $(K, \leq)$ heißt *vollständig*, wenn folgendes gilt: Jede nicht-leere beschränkte Teilmenge von K besitzt ein Supremum in K.

Beispiel 7.2.7 (die rationalen Zahlen sind unvollständig)**.** Die Teilmenge $\{x \in \mathbb{Q} \mid x^2 < 5\}$ von $\mathbb{Q}$ ist nicht-leer und beschränkt (Übungsaufgabe). Sie besitzt jedoch *kein* Supremum in $\mathbb{Q}$ (denn dies würde zu einer rationalen Zahl z mit $z^2 = 5$ führen; vgl. Beispiel 7.2.9 und Beispiel 2.2.9).

Satz 7.2.8 (die reellen Zahlen)**.** *Bis auf „kanonische Isomorphie“ gibt es genau einen archimedischen vollständigen angeordneten Körper. Diesen bezeichnen wir mit* $\mathbb{R}$ *und nennen ihn Körper der* reellen Zahlen. *Es gibt eine kanonische injektive Abbildung* $\mathbb{Q} \longrightarrow \mathbb{R}$*, die mit Addition, Multiplikation und den totalen Ordnungen verträglich ist; außerdem liegt* $\mathbb{Q}$ *bezüglich dieser Abbildung* dicht *in* $\mathbb{R}$*, d.h.:*

$$\forall_{x \in \mathbb{R}} \quad \forall_{\varepsilon \in \mathbb{R}_{>0}} \quad \exists_{y \in \mathbb{Q}} \quad x - \varepsilon \leq y \leq x + \varepsilon.$$

Beweisskizze für die Existenz der reellen Zahlen. Es gibt mehrere Möglichkeiten, die reellen Zahlen aus den rationalen oder den ganzen Zahlen zu konstruieren [10, Kapitel 2]. Wir skizzieren eine Quotientenkonstruktion; eine Variante dieser Konstruktion wird verwendet, um reelle Zahlen im Beweisassistenten HOL Light [16] zu modellieren. Anschaulich handelt es sich dabei um die Konstruktion der reellen Zahlen als Menge aller „Steigungen“ von fast additiven Funktionen $\mathbb{Z} \longrightarrow \mathbb{Z}$ [2] (Abbildung 7.1).

Eine Abbildung $f \colon \mathbb{Z} \longrightarrow \mathbb{Z}$ ist ein *Quasimorphismus*, wenn sie „fast“ mit Addition verträglich ist, d.h., wenn die Menge

$$\big\{f(x+y) - f(x) - f(y) \;\big|\; x, y \in \mathbb{Z}\big\}$$

in $\mathbb{Z}$ beschränkt ist. Sei Q die Menge aller Quasimorphismen $\mathbb{Z} \longrightarrow \mathbb{Z}$. Dann ist

$$\sim := \big\{(f,g) \in Q \times Q \;\big|\; \{f(x) - g(x) \mid x \in \mathbb{Z}\} \text{ ist beschränkt}\big\}$$

eine Äquivalenzrelation auf Q (nachrechnen!). Wir definieren die Menge $\mathbb{R}$ als den Quotienten

$$\mathbb{R} := Q/\sim.$$

Es stellt sich heraus, dass $\mathbb{R}$ bezüglich der wohldefinierten (nachrechnen!) Operationen

$$\begin{aligned} +\colon \mathbb{R} \times \mathbb{R} &\longrightarrow \mathbb{R} \\ \big([f],[g]\big) &\longmapsto \big[x \mapsto f(x) + g(x)\big] \\ \cdot\colon \mathbb{R} \times \mathbb{R} &\longrightarrow \mathbb{R} \\ \big([f],[g]\big) &\longmapsto [f \circ g] \end{aligned}$$

einen Körper bildet [2]. Zusätzlich liefert

$$\leq := \big\{([f],[g]) \in \mathbb{R} \times \mathbb{R} \;\big|\; \forall_{m \in \mathbb{N}} \quad \exists_{n \in \mathbb{N}} \quad g(n) - f(n) \geq m\big\}$$

auf diesem Körper die Struktur eines angeordneten Körpers [2]. Man kann dann nachrechnen, dass $(\mathbb{R}, \leq)$ archimedisch und vollständig (durch ein geeignetes „Diagonalargument“) ist [2]. □

Der Beweis der Eindeutigkeit spielt „archimedisch“ und „vollständig“ geschickt gegeneinander aus [10, Kapitel 2.5].

Wir werden im folgenden die deklarative Beschreibung der reellen Zahlen aus Satz 7.2.8 verwenden und nicht die konkrete Konstruktion aus der Beweisskizze. Dies ist eine weitere Instanz des allgemeinen Prinzips, dass es in vielen Situationen besser ist, mit Konstruktoren/Eliminatoren/Eigenschaften zu arbeiten als mit konkreten Implementierungen.

Beispiel 7.2.9 (Wurzeln)**.** Die Teilmenge $A := \{x \in \mathbb{R} \mid x^2 < 5\}$ von $\mathbb{R}$ ist beschränkt und nicht-leer. Sei $z := \sup A$. Dann ist z eine reelle Zahl und (Übungsaufgabe)

$$z^2 = 5.$$

Außerdem ist $z \geq 0$ und man notiert z als $\sqrt{5}$ (siehe auch Proposition 4.3.6). Analog zu Beispiel 2.2.9 sieht man, dass z *keine* rationale Zahl ist.

Auf dieselbe Weise kann man zu jeder nicht-negativen reellen Zahl x eine reelle Wurzel $\sqrt{x}$ konstruieren (d.h. eine nicht-negative reelle Zahl, deren Quadrat x ist). Eine systematische Betrachtung dieser Konstruktion wird in der Analysis vorgenommen.

Bemerkung 7.2.10 (Quadrate sind nicht-negativ)**.** Ist $(K, \leq)$ ein angeordneter Körper, so gilt

$$\forall_{x \in K} \quad x^2 \geq 0,$$

denn: Sei $x \in K$. Dann ist $x \geq 0$ oder $x \leq 0$. Wir betrachten beide Fälle:

- Ist $x \geq 0$, so folgt mit den Axiomen angeordneter Körper, dass

$$0 = 0 \cdot x \leq x \cdot x = x^2.$$

- Ist $x \leq 0$, so folgt mit den Axiomen angeordneter Körper, dass $-x \geq 0$. Aus dem ersten Fall erhalten wir daher

$$0 \leq (-x)^2 = (-1)^2 \cdot x^2 = 1 \cdot x^2 = x^2.$$

Die implizit verwendeten Aussagen, dass $0 \cdot x = 0$ und $(-1)^2 = 1$ ist, werden wir im Rahmen der Linearen Algebra im Detail beweisen. Insbesondere folgt $1 = 1^2 > 0$ und $-1 < 0$.

Diese Nicht-Negativität von Quadraten ist eine der wichtigsten und hilfreichsten Ungleichungen: Einerseits kann man mit ihr z.B. in den reellen Zahlen viele Ungleichungen beweisen. Andererseits kann man damit zeigen, dass Körper, in denen es ein Element x mit $x^2 = -1$ gibt, keine Ordnung zulassen, die zu einem angeordneten Körper führt (z.B. $\mathbb{Z}/2$; nachrechnen!).

Die reellen Zahlen sind weit mehr als nur die Erweiterung der rationalen Zahlen um wurzelartige Zahlen:

Ausblick 7.2.11 (noch mehr reelle Zahlen)**.** Sei $A \subset \mathbb{N}$. Man kann zeigen (mithilfe von geeigneten geometrischen Summen), dass die Menge

$$D_A := \left\{ \sum_{j \in A \cap \{0,\dots,n\}} \frac{1}{10^j} \;\middle|\; n \in \mathbb{N} \right\}$$

nicht-leer und beschränkt ist. Insbesondere existiert die reelle Zahl $x_A := \sup D_A$. Man kann x_A als die reelle Zahl auffassen, deren Dezimaldarstellung die Form

$$a_0.a_1a_2a_3\dots$$

hat, wobei

$$a_n = \begin{cases} 0 & \text{falls } n \notin A \\ 1 & \text{falls } n \in A \end{cases}$$

für alle $n \in \mathbb{N}$ ist.

Ist A die Menge aller Zweierpotenzen oder die Menge aller Primzahlen oder „die“ Halteproblemmenge, so stellt sich heraus, dass die zugehörige Zahl x_A *nicht* rational ist (da die Dezimaldarstellung nicht periodisch ist).

Tatsächlich gibt es unvorstellbar viel mehr reelle Zahlen als rationale Zahlen: Es gibt eine Bijektion zwischen $\mathbb{Q}$ und $\mathbb{N}$. Es gibt jedoch *keine* Bijektion zwischen $\mathbb{R}$ und $\mathbb{N}$ (Kapitel 7.5). Eine Konsequenz dieser Beobachtung ist,

dass es (da es nur abzählbar unendlich viele Programme gibt ...) reelle Zahlen gibt, deren Dezimaldarstellungen nicht algorithmisch beschrieben werden können (!).

Ausblick 7.2.12 (elementare Funktionen)**.** Die Vollständigkeit der reellen Zahlen erlaubt es, nicht nur Wurzelfunktionen, sondern auch viele andere wichtige „elementare" Funktionen zu konstruieren und zu untersuchen: Sinus, Kosinus, Exponentialfunktion, Logarithmus, ... Dies ist Gegenstand der Analysis.

7.3 Die komplexen Zahlen

Die reellen Zahlen sind bezüglich der Ordnungsrelation vollständig. Aus algebraischer Sicht sind sie jedoch nicht vollständig genug: Nicht alle polynomialen Gleichungen mit reellen Koeffizienten besitzen eine reelle Nullstelle. Ein Körper ist algebraisch abgeschlossen, wenn jede (nicht-konstante) polynomiale Gleichung mindestens eine Lösung besitzt. Die komplexen Zahlen sind die „kleinste" Erweiterung der reellen Zahlen zu einem algebraisch abgeschlossenen Körper. Faszinierenderweise genügt es dabei, geeignet eine Zahl i hinzuzufügen, die $i^2 = -1$ erfüllt.

Definition 7.3.1 (algebraisch abgeschlossen)**.** Ein Körper K ist *algebraisch abgeschlossen*, wenn folgendes gilt: Für alle $n \in \mathbb{N}\backslash\{0\}$ und alle $a_0, \ldots, a_n \in K$ mit $a_n \neq 0$ gibt es (mindestens) ein $x \in K$ mit

$$\sum_{j=0}^{n} a_j \cdot x^k = 0.$$

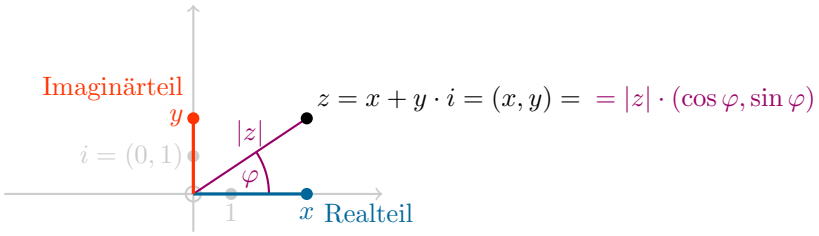
Beispiel 7.3.2.

- Die rationalen Zahlen sind *nicht* algebraisch abgeschlossen, da es zum Beispiel kein $x \in \mathbb{Q}$ mit $x^2 = 2$ gibt (Beispiel 2.2.9).
- Die reellen Zahlen sind *nicht* algebraisch abgeschlossen, da es kein $x \in \mathbb{R}$ mit $x^2 = -1$ gibt (Bemerkung 7.2.10).

Satz 7.3.3 (die komplexen Zahlen [10])**.** *Sei* $\mathbb{C} := \mathbb{R} \times \mathbb{R}$. *Wir bezeichnen* $\mathbb{C}$ *als Menge der* komplexen Zahlen*. Bezüglich den Operationen*

$$\begin{aligned} +\colon \mathbb{C} \times \mathbb{C} &\longrightarrow \mathbb{C} \\ \big((x,y),(x',y')\big) &\longmapsto (x+x', y+y') \\ \cdot\colon \mathbb{C} \times \mathbb{C} &\longrightarrow \mathbb{C} \\ \big((x,y),(x',y')\big) &\longmapsto (x \cdot x' - y \cdot y', x \cdot y' + x' \cdot y) \end{aligned}$$

bildet $\mathbb{C}$ *einen algebraisch abgeschlossenen Körper mit additiv neutralem Element* $(0,0)$ *und multiplikativ neutralem Element* $(1,0)$*. Die Abbildung*



Zerlegungen einer komplexen Zahl

Abbildung 7.2.: komplexe Zahlen, schematisch

$$\begin{aligned} \mathbb{R} &\longrightarrow \mathbb{C} \\ x &\longmapsto (x,0) \end{aligned}$$

ist injektiv und mit Addition bzw. Multiplikation verträglich. Bis auf „kanonischen Isomorphismus“ ist $\mathbb{C}$ *der „kleinste“ algebraisch abgeschlossene Körper, der* $\mathbb{R}$ *umfasst.*

Beispiel 7.3.4 (die imaginäre Einheit, Realteil/Imaginärteil)**.** Man bezeichnet

$$i := (0,1) \in \mathbb{C}$$

als *imaginäre Einheit*. Ist $z = (x,y) \in \mathbb{C}$ mit $x, y \in \mathbb{R}$, so bezeichnet man (Abbildung 7.2)

- x als *Realteil von* z und
- y als *Imaginärteil von* z.

Wir schreiben dann für $(x,y) \in \mathbb{C}$ mit $x, y \in \mathbb{R}$ auch

$$x + y \cdot i := (x,y) \in \mathbb{C};$$

diese Notation ist mit der kanonischen Inklusion $\mathbb{R} \longrightarrow \mathbb{C}$ und den Rechenoperationen verträglich (nachrechnen!).

Nach Konstruktion gilt

$$\begin{aligned} i^2 &= i \cdot i \\ &= (0,1) \cdot (0,1) = (0 \cdot 0 - 1 \cdot 1, 0 \cdot 1 + 1 \cdot 0) = (-1,0) = -(1,0) \\ &= -1. \end{aligned}$$

Insbesondere gibt es *keine* totale Ordnung auf $\mathbb{C}$, die $\mathbb{C}$ zu einem angeordneten Körper macht (Bemerkung 7.2.10).

Beispiel 7.3.5 (Summe von Quadraten)**.** Für alle $x, y \in \mathbb{R}$ gilt

$$\begin{aligned}(x+y\cdot i)\cdot(x-y\cdot i) &= x^2+y\cdot i\cdot x-x\cdot y\cdot i-y\cdot i\cdot y\cdot i\\ &= x^2+x\cdot y\cdot i-x\cdot y\cdot i-y^2\cdot i^2\\ &= x^2+0-y^2\cdot(-1)\\ &= x^2+y^2.\end{aligned}$$

Beispiel 7.3.6 (komplexe Division)**.** Die Berechnung aus Beispiel 7.3.5 ist insbesondere hilfreich, um komplexe Divisionen durch geeignetes Erweitern von Hand zu berechnen: Zum Beispiel gilt

$$\begin{aligned}\frac{1}{2+i} &= \frac{1\cdot(2-i)}{(2+i)\cdot(2-i)} && (? \text{ Erweitern mit } 2-i)\\ &= \frac{2-i}{4+1} && (? \text{ nach Beispiel 7.3.5})\\ &= \frac{2}{5}-\frac{1}{5}\cdot i.\end{aligned}$$

Also hat $1/(2+i)$ den Realteil ? $2/5$ und den Imaginärteil ? $-1/5$.

Ausblick 7.3.7 (Absolutbetrag)**.** Die „Größe“ von reellen bzw. komplexen Zahlen wird durch den (Absolut-)Betrag gemessen: Man definiert

$$\begin{aligned}&|\cdot|\colon \mathbb{R}\longrightarrow\mathbb{R}_{\geq 0}\\ &\qquad x\longmapsto \sqrt{x^2}=\begin{cases}x & \text{falls } x\geq 0\\ -x & \text{falls } x<0\end{cases}\\ &|\cdot|\colon \mathbb{C}\longrightarrow\mathbb{R}_{\geq 0}\\ &(x,y)\longmapsto \sqrt{x^2+y^2}. && (\text{„Pythagoras“})\end{aligned}$$

Diese Funktionen erfüllen insbesondere die *Dreiecksungleichung*

$$|x|-|y|\leq|x+y|\leq|x|+|y|$$

für alle reellen/komplexen Zahlen x und y. Die Körper $\mathbb{R}$ und $\mathbb{C}$ sind bezüglich diesen Betragsfunktionen im analytischen Sinne vollständig.

Ausblick 7.3.8 (Anschauung und Polardarstellung)**.** Wenn man $\mathbb{C}$ als reelle Ebene $\mathbb{R}^2$ veranschaulicht, entspricht die Addition auf $\mathbb{C}$ der Vektoraddition auf $\mathbb{R}^2$ und damit der „Verschiebung“ (Abbildung 7.3).

Die Multiplikation auf $\mathbb{C}$ entspricht in diesem Bild der „Streckung und Rotation“. Genauer lässt sich dies in Polarkoordinaten erklären: Mit analytischen Mitteln kann man zeigen, dass jede komplexe Zahl $z=x+y\cdot i$ mit $x,y\in\mathbb{R}$ in der *Polardarstellung*

$$z=|z|\cdot(\cos\varphi,\sin\varphi)=|z|\cdot e^{i\cdot\varphi}$$

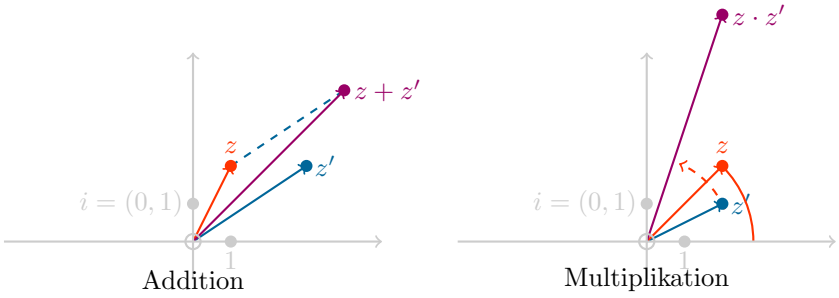


Abbildung 7.3.: Addition/Multiplikation komplexer Zahlen, schematisch

geschrieben werden kann, wobei $\varphi = \arctan(x/y)$; ist $y = 0$, so setzt man $\varphi = 0$. Dann gilt: Sind $r, r', \varphi, \varphi' \in \mathbb{R}$, so ist

$$(r \cdot e^{i \cdot \varphi}) \cdot (r' \cdot e^{i \cdot \varphi'}) = (r \cdot r') \cdot e^{i \cdot (\varphi + \varphi')};$$

der vordere Term ist dabei eine „Streckung“, der hintere eine „Rotation“.

Der Absolutbetrag einer komplexen Zahl stimmt dabei mit dem euklidischen Abstand überein (nach dem Satz des Pythagoras).

7.4 Anwendung*: Repräsentation von Zahlen

Zahlen aller Art treten in der Modellierung und Programmierung in vielen Rollen auf. Dabei sind auch reelle und komplexe Zahlen von großer Bedeutung. Zum Beispiel treten bei der Komplexitätsanalyse von Algorithmen auf natürliche Weise elementare Funktionen aus der Analysis auf (z.B. Logarithmen), die sich nur im Kontext der reellen bzw. Zahlen sinnvoll behandeln lassen. In der Modellierung treten insbesondere in der Computergraphik die reellen Zahlen als Grundbaustein der Vektorgeometrie auf. Das maschinelle Lernen beruht auf höhere Methoden aus der mehrdimensionalen Analysis und der Stochastik/Statistik, die essentiellen Gebrauch von reellen und komplexen Zahlen machen.

Während man in der abstrakten Modellierung und Analyse mit den in diesem Kapitel skizzierten mathematischen Konzepten umgehen kann, tritt bei der Implementierung von Modellen und Verfahren noch eine zusätzliche Schwierigkeit auf: Die mathematischen Konzepte von Zahlen müssen geeignet in der Programmierung modelliert und repräsentiert werden.

Die Konstruktionen der natürlichen, ganzen, rationalen, reellen und komplexen Zahlen geben Hinweise darauf, wie man solche Zahlsysteme in der

Programmierung repräsentieren kann. Es ist dabei grundsätzlich zwischen zwei verschiedenen Arten der Repräsentation zu unterscheiden:

- exakte Repräsentationen mit exakter Arithmetik;
- Repräsentationen fester Länge mit inexakter Arithmetik.

Exakte Repräsentationen haben den Vorteil, dass sie (unter der Annahme fehlerfreier Implementierung und Hardware) exakte, korrekte Ergebnisse liefern. Im allgemeinen haben exakte Repräsentationen aber den Nachteil, dass sie nicht sehr effizient sind und daher in vielen praktischen Situationen nicht ohne Weiteres eingesetzt werden können.

Repräsentationen fester Länge haben den Vorteil, dass sie sehr effizient umgesetzt werden können. Solche Repräsentationen haben aber den Nachteil, dass sie im Normalfall *keine* exakten, korrekten Ergebnisse liefern, da es zu Überläufen und Rundungsfehlern kommt. Bei der Verwendung von Repräsentationen fester Länge muss daher auch im Voraus überlegt werden, welche Ungenauigkeiten auftreten können, welche Größenordnung diese Effekte haben und welche Auswirkung dies auf den konkreten Einsatz hat.

Beispiel 7.4.1 (Zahlen in Haskell)**.** Zu den Standardtypen für Zahlen in der funktionalen Programmiersprache Haskell gehören unter anderem:

- Ganze Zahlen:
  - `Int`: Ganze Zahlen fester Länge (derzeit signed 64 Bit) mit Überlauf.
  - `Integer`: Exakte ganze Zahlen mit exakter Arithmetik. Intern handelt es sich dabei um eine geschickte Erweiterung von `Int`.

  Analog gibt es auch unsignierte Varianten `Word` (feste Länge) und `Natural` (exakt), die als natürliche Zahlen verwendet werden können; der Gebrauch ist aber nicht so weit verbreitet.
- Rationale Zahlen:
  - `Rational`: Exakte rationale Zahlen, die intern als Paare ganzer Zahlen repräsentiert werden.
- Reelle Zahlen:
  - `Double`: Fließkommarepräsentation mit fester Länge, gemäß IEEE-Standard.
  - `CReal`: Reelle Zahlen mit beliebiger Präzision, die intern als Fast Binary Cauchy Sequences repräsentiert werden (und Caching zur Effizienzsteigerung unterstützen).
- Komplexe Zahlen:
  - `Complex Double`: Fließkommaversion komplexer Zahlen mit fester Länge, repräsentiert als Paare von `Double`s (entsprechend dem Real- und Imaginärteil).

- `Complex CReal`: Eine Variante mit beliebiger Präzision.

Beispiel 7.4.2 (reelle Zahlen in HOL Light)**.** Im Beweisassistenten HOL Light gibt es eine exakte Repräsentation und Arithmetik von reellen Zahlen [16]; diese beruht auf der Konstruktion in der Beweisskizze von Satz 7.2.8.

Beispiel 7.4.3 (Zahlen in Lean)**.** Zu den Standardtypen für Zahlen in der Programmiersprache/in dem Beweisassistenten Lean gehören unter anderem die folgenden exakten Repräsentationen:

- Natürliche Zahlen: Der induktive Datentyp `nat` (Kapitel 4.4.1) mit den von den Peano-Axiomen inspirierten Konstruktoren `zero` und `succ`.
- Ganze Zahlen: Der Datentyp `int` repräsentiert ganze Zahlen als natürliche Zahlen oder negative natürliche Zahlen.
- Rationale Zahlen: Der Datentyp `rat` repräsentiert rationale Zahlen als vollständig gekürzte Brüche.
- Reelle Zahlen: Der Datentyp `real` repräsentiert reelle Zahlen als Cauchyfolgen.
- Komplexe Zahlen: Der Datentyp `complex` repräsentiert komplexe Zahlen als Paare reeller Zahlen (entsprechend dem Real- und Imaginärteil).

7.5 Ausblick*: Wie viele reelle Zahlen gibt es?

Satz 7.5.1 (Überabzählbarkeit der reellen Zahlen)**.** *Es gibt eine injektive Abbildung* $\mathbb{N} \longrightarrow \mathbb{R}$ *aber* keine *surjektive Abbildung* $\mathbb{N} \longrightarrow \mathbb{R}$*. Insbesondere sind* $\mathbb{N}$ *und* $\mathbb{R}$ nicht *gleichmächtig.*

Beweisskizze. Nach Satz 7.2.8 (und Proposition 6.2.9, Proposition 6.2.12) gibt es eine injektive Abbildung $\mathbb{N} \longrightarrow \mathbb{R}$.

Um zu zeigen, dass es keine surjektive Abbildung $\mathbb{N} \longrightarrow \mathbb{R}$ gibt, verwenden wir ein Diagonalargument: *Angenommen*, es gäbe eine surjektive Abbildung $f\colon \mathbb{N} \longrightarrow \mathbb{R}$. Aus f kann man rekursiv auch eine surjektive Abbildung

$$f'\colon \mathbb{N} \longrightarrow \left\{x_A \;\middle|\; A \in P(\mathbb{N})\right\}$$

konstruieren, wobei x_A wie in Ausblick 7.2.11 definiert ist. Außerdem ist es nicht schwer zu sehen, dass

$$\begin{aligned} g\colon P(\mathbb{N}) &\longrightarrow \left\{x_A \;\middle|\; A \in P(\mathbb{N})\right\} \\ A &\longmapsto x_A \end{aligned}$$

bijektiv ist. Somit ist die Komposition

$$g^{-1} \circ f' \colon \mathbb{N} \longrightarrow P(\mathbb{N})$$

surjektiv. Wir wissen aber bereits aus einem Diagonalargument, dass es eine solche Surjektion nicht geben kann (Übungsaufgabe). Dieser Widerspruch zeigt, dass f nicht surjektiv ist. □

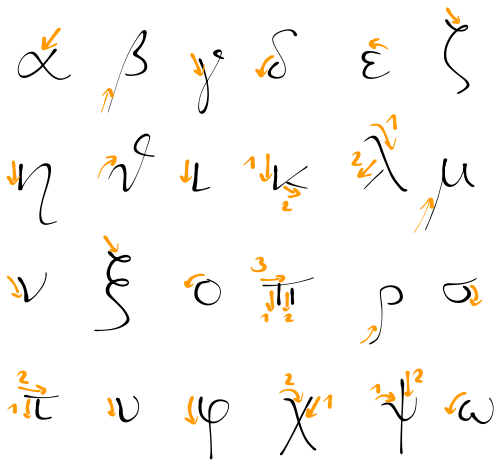
Man kann dieses Diagonalargument anschaulicher darstellen, wenn man die Dezimaldarstellung expliziter verwendet [22].

A
Anhang

Überblick über dieses Kapitel.

A.1 Das griechische Alphabet

Symbol		Name	TEX-/LATEX-Kommando	
A	α	alpha	A	\alpha
B	β	beta	B	\beta
Γ	γ	gamma	\Gamma	\gamma
Δ	δ	delta	\Delta	\delta
E	ε, ϵ	epsilon	E	\varepsilon , \epsilon
Z	ζ	zeta	Z	\zeta
H	η	eta	H	\eta
Θ	ϑ, θ	theta	\Theta	\vartheta , \theta
I	ι	iota	I	\iota
K	κ	kappa	K	\kappa
Λ	λ	lambda	\Lambda	\lambda
M	μ	my	M	\mu
N	ν	ny	N	\nu
Ξ	ξ	xi	\Xi	\xi
O	o	omikron	O	o
Π	π	pi	\Pi	\pi
P	ϱ, ρ	rho	P	\varrho , \rho
Σ	σ, ς	sigma	\Sigma	\sigma , \varsigma
T	τ	tau	T	\tau
Y	υ	ypsilon	Y	\upsilon
Φ	φ, ϕ	phi	\Phi	\varphi , \phi
X	χ	chi	X	\chi
Ψ	ψ	psi	\Psi	\psi
Ω	ω	omega	\Omega	\omega



A.2 Mächtigkeit von Mengen

Wir stellen im folgenden kurz vor, wie man die „Größe“ von Mengen angeben bzw. vergleichen kann:

A.2.1 Endliche Mengen

Definition A.2.1 (endliche Menge)**.** Eine Menge X ist *endlich*, wenn es ein $n \in \mathbb{N}$ und eine Bijektion $\{1, \dots, n\} \longrightarrow X$ gibt. Zur Erinnerung: Wir schreiben $\{1, \dots, n\} := \{k \in \mathbb{N} \mid (1 \leq k) \wedge (k \leq n)\}$.

Proposition und Definition A.2.2 (Mächtigkeit endlicher Mengen)**.** *Sei X eine endliche Menge.*

1. *Dann gibt es* genau *ein $n \in \mathbb{N}$, so dass eine Bijektion $\{1, \dots, n\} \longrightarrow X$ existiert. Wir definieren in diesem Fall die* Mächtigkeit von X *als*

$$\#X := n.$$

2. *Ist Y eine Menge und gibt es eine surjektive Abbildung $X \longrightarrow Y$, so ist auch Y endlich und $\#Y \leq \#X$.*

3. *Ist Y eine Menge und gibt es eine injektive Abbildung $Y \longrightarrow X$, so ist auch Y endlich und $\#Y \leq \#X$.*

4. Inklusions-/Exklusionsprinzip. *Ist Y eine endliche Menge, so ist $X \cup Y$ endlich und es gilt*

$$\#(X \cup Y) = \#X + \#Y - \#(X \cap Y).$$

5. *Ist Y eine endliche Menge, so ist $X \times Y$ endlich und es gilt*

$$\#(X \times Y) = \#X \cdot \#Y.$$

6. *Die Potenzmenge $P(X)$ ist endlich und es gilt $\#P(X) = 2^{\#X}$.*

Beweisskizze. Alle Aussagen können durch geeignete Induktionen gezeigt werden. □

Beispiel A.2.3 (leere Menge)**.** Die leere Menge ist endlich, da es eine Bijektion $\{1, \dots, 0\} = \emptyset \longrightarrow \emptyset$ gibt. Es gilt $\#\emptyset = 0$.

Ist eine Menge X nicht endlich, so schreiben wir auch kurz $\#X = \infty$.

A.2.2 Unendliche Mengen

Definition A.2.4 (gleichmächtig)**.** Mengen X und Y heißen *gleichmächtig*, wenn es eine Bijektion $X \longrightarrow Y$ gibt. Wenn X und Y gleichmächtig sind, sagt man auch, dass X und Y *dieselbe Kardinalität* haben und schreibt in diesem Fall $|X| = |Y|$.

Bemerkung A.2.5. Gleichmächtig zu sein ist reflexiv, symmetrisch und transitiv (wie man leicht an der Charakterisierung von Bijektivität über Umkehrabbildungen in Proposition 3.2.17 sehen kann).

Beispiel A.2.6 (Potenzmengen sind „groß“)**.** Ist X eine Menge, so ist

$$\begin{aligned} X &\longrightarrow P(X) \\ x &\longmapsto \{x\} \end{aligned}$$

eine injektive Abbildung. Aber es gibt *keine* bijektive Abbildung $X \longrightarrow P(X)$ (Übungsaufgabe). Insbesondere ist $|X| \neq |P(X)|$.

Satz A.2.7 (Satz von Schröder–Bernstein)**.** *Sind X und Y Mengen und gibt es injektive Abbildungen $X \longrightarrow Y$ und $Y \longrightarrow X$, so gilt bereits $|X| = |Y|$.*

Beweisskizze. Dies kann zum Beispiel mit einem geeigneten Fixpunktsatz für monotone mengenwertige Abbildungen gezeigt werden [28, Kapitel 9.2]. □

Definition A.2.8 (unendliche Menge)**.** Eine Menge ist *unendlich*, wenn sie nicht endlich ist.

Satz A.2.9 (Unendlichkeit und $\mathbb{N}$)**.** *Sei X eine Menge. Dann ist X genau dann unendlich, wenn es eine injektive Abbildung $\mathbb{N} \longrightarrow X$ gibt.*

Beweisskizze (AC). Falls es eine injektive Abbildung $\mathbb{N} \longrightarrow X$ gibt, so folgt mit Proposition A.2.2, dass X nicht endlich (und somit unendlich) ist.

Ist umgekehrt X eine unendliche Menge, so kann man induktiv (mithilfe des Rekursionssatzes und einer geeigneten Variante des Auswahlaxioms) eine injektive Abbildung $\mathbb{N} \longrightarrow X$ konstruieren. □

Definition A.2.10 (abzählbar unendlich, höchstens abzählbar, überabzählbar)**.**

- Eine Menge X heißt *abzählbar unendlich*, wenn $|X| = |\mathbb{N}|$ gilt.
- Eine Menge heißt *höchstens abzählbar*, wenn sie endlich oder abzählbar unendlich ist.
- Eine Menge heißt *überabzählbar unendlich*, wenn sie unendlich aber nicht abzählbar unendlich ist.

Beispiel A.2.11. Die Menge $\mathbb{N} \times \mathbb{N}$ ist abzählbar unendlich (diagonales Zickzack!), die Menge $\mathbb{Q}$ ist abzählbar unendlich (diagonales Zickzack!), aber die Menge $\mathbb{R}$ ist *nicht* abzählbar unendlich (Cantorsches Diagonalargument; Satz 7.5.1).

Caveat A.2.12 (Kontinuumshypothese)**.** Ob es eine Menge X gibt, die nicht abzählbar unendlich ist, und für die es keine injektive Abbildung $\mathbb{R} \longrightarrow X$ gibt, ist unabhängig von den Axiomen der Mengenlehre (!). Diese Aussage kann also weder aus den Axiomen der Mengenlehre gefolgert noch widerlegt werden [28].

B
Übungsblätter

Grundlagen der Mathematik$^{\text{FIDS}}$: Übungen

Prof. Dr. C. Löh/PD Dr. F. Strunk/M. Uschold Blatt 0, 16. Oktober 2023

Aufgabe 1 (tautologisch?!; 4 Punkte)**.** Seien A und B aussagenlogische Variablen. Welche der folgenden aussagenlogischen Formeln sind aussagenlogische Tautologien? Begründen Sie Ihre Antwort durch einen Beweis oder ein geeignetes Gegenbeispiel!

1. $A \implies \big(B \wedge (\neg B)\big)$
2. $\big((\neg A) \vee B\big) \iff (A \implies B)$

Aufgabe 2 (natürliche Wahrheiten?!; 4 Punkte)**.** Übersetzen Sie die folgende quantorenlogische Formel in der Sprache der natürlichen Zahlen in einen entsprechenden deutschen Satz (bzw. umgekehrt). Handelt es sich dabei jeweils um eine wahre quantorenlogische Aussage? Begründen Sie Ihre Antwort!

1. $\forall_x \quad \exists_y \quad \exists_z \quad x = z + y + 1 + y + z$
2. Es gibt eine natürliche Zahl z mit folgender Eigenschaft: Für alle natürlichen Zahlen x ist $x \cdot z + x = x$.

Hinweis. Wir werden die natürlichen Zahlen später präzise einführen. Sie können in dieser Aufgabe Ihr Schulwissen verwenden.

Aufgabe 3 (der Dreiecksoperator; 4 Punkte)**.** Wir ergänzen die aussagenlogische Syntax um folgendes:

- Sind A und B aussagenlogische Formeln, so ist auch $(A \triangle B)$ eine aussagenlogische Formel.

Wir ergänzen die klassische Semantik der Aussagenlogik um die folgende Interpretation:

A	B	$A \triangle B$
w	w	f
w	f	w
f	w	w
f	f	f

Bearbeiten Sie die folgenden Aufgaben in dieser erweiterten Aussagenlogik:

1. Zeigen Sie, dass $(A \triangle B) \implies (A \vee B)$ eine aussagenlogische Tautologie ist.
2. Wie könnte man „$\triangle$“ umgangssprachlich sinnvoll übersetzen? Begründen Sie Ihre Antwort!

Hinweis. Es handelt sich bei dem Symbol „$\triangle$“ *nicht* um die Standardnotation für diese logische Operation!

Keine Abgabe/Korrektur!

Hinweise zu Aufgabe 1

- Wie ist „aussagenlogische Tautologie“ definiert?
- Was muss man also überprüfen?

Hinweise zu Aufgabe 2

- Wie ist die Semantik von $\forall$ und $\exists$ definiert?
- Denken Sie auch an kleine natürliche Zahlen!
- Die natürlichen Zahlen enthalten 0.

Hinweise zu Aufgabe 3

- Wie ist „aussagenlogische Tautologie“ definiert?
- Vergleichen Sie die Wahrheitstafel für „$\triangle$“ mit umgangssprachlichen Satzkonstruktionen.

Lösung zu Aufgabe 1

Voraussetzung: Seien A und B aussagenlogische Variablen.

- Teilaufgabe 1:

  Behauptung. Es ist $A \Longrightarrow \big(B \wedge (\neg B)\big)$ *keine* aussagenlogische Tautologie.

  Beweis. Es genügt, eine Belegung für A und B anzugeben, für die angegebene Formel nicht den Wert w erhält.

  Wir belegen A mit w und B mit w. Dann erhalten wir Schritt für Schritt:

  - Die Formel $B \wedge (\neg B)$ erhält den Wert f.
  - Die Formel $A \longrightarrow \big(B \wedge (\neg B)\big)$ erhält somit den Wert f.

  Also ist $A \longrightarrow \big(B \wedge (\neg B)\big)$ keine aussagenlogische Tautologie. □

- Teilaufgabe 2:

  Behauptung. Es ist $\big((\neg A) \vee B\big) \Longleftrightarrow (A \Longrightarrow B)$ eine aussagenlogische Tautologie.

  Beweis. Wir überprüfen, dass die gegebene Formel unter allen möglichen Belegungen für A und B den Wert w erhält:

A	B	$\neg A$	$(\neg A) \vee B$	$A \Longrightarrow B$	$\big((\neg A) \vee B\big) \Longleftrightarrow (A \Longrightarrow B)$
w	w	f	w	w	w
w	f	f	f	f	w
f	w	w	w	w	w
f	f	w	w	w	w

  Also ist $\big((\neg A) \vee B\big) \Longleftrightarrow (A \Longrightarrow B)$ eine aussagenlogische Tautologie. □

Hinweis. Lösungen sollten im Normalfall in

- Voraussetzung
- Behauptung
- Beweis

gegliedert werden. Durch diese Struktur wird die Lösung übersichtlich und viele Fehler bzw. Lücken werden vermieden.

Den allgemeinen Beweisbegriff führen wir in der zweiten Vorlesung ein. Auf diesem Übungsblatt treten nur „Begründungen“ auf, die direkt auf den Definitionen beruhen. Es ist aber vielleicht nicht verkehrt, sich gleich das Wort „Beweis“ statt „Begründung“ anzugewöhnen.

Lösungsversuch zu Aufgabe 1.1

Ja, es ist eine Tautologie, denn, wenn man A mit f und B mit w belegt, ergibt sich für die gesamte Formel der Wert w.

Korrektur. Diese „Lösung“ ist nicht korrekt: Die behauptete Aussage stimmt nicht (s. obiger Beweis). Die Begründung ist nicht schlüssig, da für eine Tautologie *alle* möglichen Bewertungen überprüft werden müssen.

Lösungsversuch zu Aufgabe 1.1

Nein, es ist keine Tautologie. Dies ist offensichtlich.

Korrektur. „Offensichtlich“ ist ein gefährliches Wort, das zu vielen Fehlern und Missverständnissen führt. Lösungen zu Übungsaufgaben müssen so formuliert sein, dass dem Leser klar ist, dass Sie wirklich verstanden haben, wie Ihre Lösung funktioniert. Dies ist bei der obigen „Lösung“ nicht gegeben.

Lösungsversuch zu Aufgabe 1.2

Behauptung. Es ist $\big((\neg A) \vee B\big) \iff (A \implies B)$ eine aussagenlogische Tautologie.

Beweis. Ich habe ein Programm geschrieben, das alle möglichen Belegungen ausprobiert. Es ergab sich immer der Wert w. Also handelt es sich um eine aussagenlogische Tautologie. □

Korrektur. Man könnte diese Aufgabe tatsächlich mit einem entsprechenden Programm lösen. Dann muss aber unbedingt der Quellcode mit abgegeben werden und es muss nachvollziehbar (!) dokumentiert werden, warum das Programm korrekt ist und die Aufgabe löst (das ist oft schwieriger als man denkt . . .).

Lösungsversuch zu Aufgabe 1.2

A	B	$\neg A$	$(\neg A) \vee B$	$A \implies B$	$\big((\neg A) \vee B\big) \iff (A \implies B)$
w	w	f	w	w	w
w	f	f	f	f	w
f	w	w	w	w	w
f	f	w	w	w	w

Korrektur. Bei dieser „Lösung“ ist die Tabelle zwar korrekt, es ist aber nicht klar, was die Antwort auf Frage aus der Aufgabe ist und was diese Tabelle damit zu tun hat.

Falls man die Antwort auf die Fragen nicht sofort „sieht“, kann es natürlich sinnvoll sein, zunächst diese Tabelle vollständig auszufüllen und dann anhand dieser Tabelle die Frage zu beantworten. Die abgegebene Lösung muss aber auf jeden Fall die Antwort enthalten und nachvollziehbar formuliert sein.

Lösung zu Aufgabe 2

- Teilaufgabe 1:

  Voraussetzung. Sei A die quantorenlogische Formel

  $$\forall_x \quad \exists_y \quad \exists_z \quad x = z + y + 1 + y + z$$

  in der Sprache der natürlichen Zahlen.

  Umgangssprachliche Formulierung. Indem wir

  - $\forall_x$ mit „für alle natürlichen Zahlen x“,
  - $\exists_x$ mit „es existiert eine natürliche Zahl x“

  übersetzen, entspricht A dem folgenden deutschen Satz:

  Für alle natürlichen Zahlen x gibt es natürliche Zahlen y und z mit $x = z + y + 1 + y + z$.

  Behauptung. Die quantorenlogische Aussage A ist *nicht* wahr.

  Beweis. Es genügt, eine natürliche Zahl x zu finden, für die die Aussage $\exists_y \quad \exists_z \quad x = z + y + 1 + y + z$ in der Sprache der natürlichen Zahlen *nicht* wahr ist.

  Wir betrachten die natürliche Zahl $x = 0$. Für alle natürlichen Zahlen y und z ist

  $$z + y + 1 + y + z \geq 1 > 0 = x.$$

  Insbesondere ist $z + y + 1 + y + z \neq x$. Also gilt *nicht*, dass es natürlichen Zahlen y, z mit $x = z + y + 1 + y + z$ gibt.

  Somit ist $\exists_y \quad \exists_z \quad x = z + y + 1 + y + z$ für $x = 0$ *nicht* wahr. Also ist A *nicht* wahr. □

- Teilaufgabe 2:

  Voraussetzung. Wir betrachten den folgenden deutschen Satz: Es gibt eine natürliche Zahl z mit folgender Eigenschaft: Für alle natürlichen Zahlen x ist $x \cdot z + x = x$.

  Formulierung als quantorenlogische Formel in der Sprache der natürlichen Zahlen. Indem wir

  - „für alle natürlichen Zahlen x“ mit $\forall_x$,
  - „es existiert eine natürliche Zahl x“ mit $\exists_x$

übersetzen, entspricht der obige Satz der folgenden quantorenlogischen Formel B:

$$\exists_z \quad \forall_x \quad x \cdot z + x = x.$$

Behauptung. Die quantorenlogische Aussage B ist wahr.

Beweis. Es genügt, eine natürliche Zahl z zu finden, für die die quantorenlogische Aussage $\forall_x \quad x \cdot z + x = x$ in der Sprache der natürlichen Zahlen wahr ist.

Wir betrachten $z := 0$. Dann gilt für alle natürlichen Zahlen x, dass

$$x \cdot z + x = x \cdot 0 + x = 0 + x = x.$$

Also gilt $\forall_x \quad x \cdot z + x = x$ für $z = 0$. Somit ist B wahr. □

Lösung zu Aufgabe 3

- Teilaufgabe 1:

  Voraussetzung. Seien A und B aussagenlogische Variablen.

  Behauptung. Es ist $(A \triangle B) \Longrightarrow (A \vee B)$ eine aussagenlogische Tautologie.

  Beweis. Wir überprüfen, dass die gegebene Formel unter allen möglichen Belegungen für A und B den Wert w erhält:

A	B	$A \triangle B$	$A \vee B$	$(A \triangle B) \Longrightarrow (A \vee B)$
w	w	f	w	w
w	f	w	w	w
f	w	w	w	w
f	f	f	f	w

  Also ist $(A \triangle B) \Longrightarrow (A \vee B)$ eine aussagenlogische Tautologie. □

- Teilaufgabe 2:

  Behauptung. Umgangssprachlich entspricht „① △ ②“ der Formulierung „entweder ① oder ②“.

  Beweis. Das umgangssprachliche „entweder ... oder ...“ wird durch die folgende Wahrheitstafel modelliert:

A	B	entweder A oder B
w	w	f
w	f	w
f	w	w
f	f	f

  Die Werteverteilung dieser Wahrheitstabelle stimmt mit der für „△“ überein. □

  Hinweis. In der Informatik wird diese Operation normalerweise als „XOR“ (exclusive or) bezeichnet.

Grundlagen der Mathematik$^{\text{FIDS}}$: Übungen

Prof. Dr. C. Löh/PD Dr. F. Strunk/M. Uschold Blatt 1, 16. Oktober 2023

Fingerübung A (aussagenlogische Formeln)**.** Seien A und B aussagenlogische Variablen. Welche der folgenden Symbolketten sind aussagenlogische Formeln?

$$\big(A \wedge (A \wedge A)\big), \quad \big(B \implies (\vee A)\big), \quad (A \wedge / \vee B)$$

Fingerübung B (klassische Semantik)**.** Welche Werte erhält man in der klassischen Semantik der Aussagenlogik, wenn man in

$$\big(A \wedge (B \vee A)\big) \implies \big((\neg A) \vee B\big)$$

die Variable A mit w und die Variable B mit f belegt? Skizzieren Sie auch den Syntaxbaum und illustrieren Sie daran Ihre Berechnung!

Fingerübung C (StVO)**.** Formalisieren Sie die folgende Aussage im Stile der Quantorenlogik und negieren Sie die Aussage; versuchen Sie dabei, die Negationen auch wieder sprachlich sauber zu formulieren.

> Verkehrshindernisse sind, wenn nötig, mit eigener Lichtquelle zu beleuchten oder durch andere zugelassene lichttechnische Einrichtungen kenntlich zu machen.

Hinweis. Die Fingerübungen werden nicht abgegeben und nicht korrigiert. Normalerweise werden die Fingerübungen in der jeweiligen Übungsgruppe besprochen. Da die Übungsgruppen erst ab der zweiten Vorlesungswoche stattfinden, können Sie diese Woche alternativ das Frageforum in GRIPS verwenden, wenn Sie Fragen haben. Außerdem helfen evtl. die Hinweise zu Blatt 0.

Hinweis. Bitte lesen Sie vor dem Aufschreiben/vor der Abgabe auch die Lösungshinweise zu Blatt 0 und die allgemeinen Hinweise zum Bearbeiten von Übungsaufgaben!

Aufgabe 1 (tautologisch?!; 4 Punkte)**.** Seien A und B aussagenlogische Variablen. Welche der folgenden aussagenlogischen Formeln sind aussagenlogische Tautologien? Begründen Sie Ihre Antwort durch einen Beweis oder ein geeignetes Gegenbeispiel!

1. $\big(A \implies (B \wedge (\neg B))\big) \implies (\neg A)$
2. $(A \wedge B) \implies \big((\neg A) \wedge (\neg B)\big)$

Aufgabe 2 (natürliche Wahrheiten?!; 4 Punkte)**.** Übersetzen Sie die folgende quantorenlogische Formel in der Sprache der natürlichen Zahlen in einen entsprechenden deutschen Satz (bzw. umgekehrt). Handelt es sich dabei jeweils um eine wahre quantorenlogische Aussage? Begründen Sie Ihre Antwort!

1. $\forall_x \quad \big((\exists_y \quad x = y + 1) \implies (\exists_z \quad x = z + 1)\big)$
2. Für jede natürliche Zahl x gibt es eine natürliche Zahl y mit $x = y + 1$.

Hinweis. Wir werden die natürlichen Zahlen später präzise einführen. Sie können in dieser Aufgabe Ihr Schulwissen verwenden.

Bitte wenden

Aufgabe 3 (der Sternoperator; 4 Punkte)**.** Wir ergänzen die aussagenlogische Syntax um folgendes:

- Sind A und B aussagenlogische Formeln, so ist auch $(A * B)$ eine aussagenlogische Formel.

Wir ergänzen die klassische Semantik der Aussagenlogik um die folgende Interpretation:

A	B	$A * B$
w	w	f
w	f	f
f	w	f
f	f	w

Bearbeiten Sie die folgenden Aufgaben in dieser erweiterten Aussagenlogik:

1. Zeigen Sie, dass $(A * B) \iff \big((\neg A) \wedge (\neg B)\big)$ eine aussagenlogische Tautologie ist.
2. Wie könnte man „$*$“ umgangssprachlich sinnvoll übersetzen? Begründen Sie Ihre Antwort!

Bonusaufgabe (Blorxlogik; 4 Punkte)**.** Commander Blorx hat sich die untenstehende Logik mit dem Blorxoperator $\boxtimes$ ausgedacht. Erklären Sie, wie Commander Blorx die gewöhnlichen Operatoren $\neg$, $\wedge$, $\vee$, $\Longrightarrow$, $\Longleftrightarrow$ (mit ihrer klassischen Semantik) durch $\boxtimes$ ausdrücken kann!

- *Syntax.*
  - Aussagenlogische Variablen sind Blorxformeln.
  - Sind A und B Blorxformeln, so auch $(A \boxtimes B)$; lies: „A blorx B“.
  - Keine weiteren Symbolketten sind Blorxformeln.
- *Semantik.*
  - Variablen können mit den Wahrheitswerten w bzw. f belegt werden.
  - Belegen wir alle in einer aussagenlogischen Formel vorkommenden Variablen mit w bzw. f (wobei verschiedene Auftreten derselben Variablen in einer Formel denselben Wert erhalten müssen), so erhalten wir einen Wahrheitswert, indem wir Schritt für Schritt die folgende semantische Regel anwenden:

A	B	$A \boxtimes B$
w	w	f
w	f	w
f	w	w
f	f	w

Abgabe bis 23. Oktober 2023, 10:00, via GRIPS

Grundlagen der Mathematik$^{\mathsf{FIDS}}$: Übungen

Prof. Dr. C. Löh/PD Dr. F. Strunk/M. Uschold Blatt 2, 23. Oktober 2023

Hinweis. Zur Erinnerung: Die Fingerübungen werden nicht abgegeben/korrigiert, sondern in den Übungsgruppen bearbeitet.

Fingerübung A (Assoziativität?)**.** Seien A, B, C aussagenlogische Variablen. Handelt es sich bei den folgenden aussagenlogischen Formeln um aussagenlogische Tautologien?

$$\big(A \wedge (B \wedge C)\big) \iff \big((A \wedge B) \wedge C\big)$$
$$\big(A \implies (B \implies C)\big) \iff \big((A \implies B) \implies C\big)$$

Fingerübung B (Voraussetzung/Behauptung)**.** Zerlegen Sie die folgenden Textblöcke in Voraussetzungen und Behauptungen. Welche grundlegende Beweisstruktur könnte man jeweils erwarten?

- Sei K ein Körper. Dann gilt für alle $x \in K$, dass $x \cdot 0 = 0$.
- Sei K ein Körper, sei V ein K-Vektorraum mit einer Basis $(v_i)_{i \in I}$ und sei W ein K-Vektorraum. Dann gibt es zu jeder Abbildung $f \colon I \longrightarrow W$ genau eine K-lineare Abbildung $F \colon V \longrightarrow W$ mit
$$\forall_{i \in I} \quad F(v_i) = f(i).$$

Hinweis. Sie müssen diese Textblöcke nicht verstehen; es geht um die Struktur.

Fingerübung C (mehr Tautologien?)**.** Sei T eine Sprache/Theorie und sei A eine quantorenlogische Aussage über T, in der die Variable x nicht gebunden vorkommt. Welche der folgenden quantorenlogischen Aussagen sind aussagenlogische Tautologien über T ? Begründen Sie Ihre Antwort!

1. $\big(\forall_x \quad A(x)\big) \implies \big(\exists_x \quad A(x)\big)$
2. $\big(\exists_x \quad A(x)\big) \vee \big(\neg\big(\exists_x \quad A(x)\big)\big)$

Aufgabe 1 (noch mehr Tautologien?; 4 Punkte)**.** Sei T eine Sprache/Theorie und seien A und B quantorenlogische Aussagen über T, in denen die Variable x nicht gebunden vorkommt. Welche der folgenden quantorenlogischen Aussagen sind aussagenlogische Tautologien über T ? Begründen Sie Ihre Antwort!

1. $\big((\forall_x \quad A(x)) \wedge (\forall_x \quad B(x))\big) \implies \big((\forall_x \quad B(x)) \wedge (\forall_x \quad A(x))\big)$
2. $\big((\forall_x \quad A(x)) \wedge (\exists_x \quad B(x))\big) \implies (\exists_x \quad B(x))$

Aufgabe 2 (Implikationsumkehr; 4 Punkte)**.** Was ist falsch am nachfolgenden „Beweis“? Geben Sie genau an, an welcher Stelle etwas schiefgeht und erklären Sie den Fehler!

Behauptung. Wenn A und B quantorenlogische Aussagen sind und $A \implies B$ gilt, so gilt auch $B \implies A$

Beweis. Seien A und B quantorenlogische Aussagen und es gelte $A \implies B$. *Angenommen*, es gilt nicht $B \implies A$, d.h. es gilt $B \implies \neg A$. Wegen der Voraussetzung $A \implies B$ erhalten wir daraus aber auch $A \implies \neg A$, was nicht sein kann. Also muss die Annahme falsch gewesen sein, und damit gilt $B \implies A$. □

Bitte wenden

Aufgabe 3 (Wer war's?; 4 Punkte)**.** Professor Pirkheimer wird tot in der Bibliothek seines Anwesens aufgefunden.

① Als Täter kommen nur der Gärtner, der Frisör oder Blorx infrage.

② Nur der Gärtner und der Frisör haben eine Schere und es gibt keine Tasse, die nicht im Schrank ist.

③ Wenn die Frisur von Professor Pirkheimer wohlgeordnet ist, hatte der Frisör keine Zeit für den Mord oder der Frisör hat Blorx das Frisieren beigebracht.

④ Wenn Blorx den Professor erlegt hat, sind nicht mehr alle Tassen im Schrank.

⑤ Wenn alle Tassen im Schrank sind, ist auch Professor Pirkheimers Frisur wohlgeordnet.

⑥ Professor Pirkheimer wurde mit einer Schere erdolcht.

⑦ Blorx kann nicht Frisieren.

Wer war's? Formulieren Sie eine geeignete Behauptung und beweisen Sie diese logisch aus den obigen Axiomen!

Bonusaufgabe (Wer war's, reloaded; 4 Punkte)**.** Formalisieren Sie die Axiome und Ihren Beweis von Aufgabe 3 in Lean 4.
Hinweis. Geben Sie (auch) die Quelldatei ab (als Textdatei). Stellen Sie bitte sicher, dass die Datei fehlerfrei interpretiert/kompiliert werden kann und dokumentieren Sie Ihren Quellcode sinnvoll.

Abgabe bis 30. Oktober 2023, 10:00, via GRIPS

Grundlagen der Mathematik$^{\text{FIDS}}$: Übungen

Prof. Dr. C. Löh/PD Dr. F. Strunk/M. Uschold Blatt 3, 30. Oktober 2023

Fingerübung A (Mengenoperationen). Sei $A := \{0, 2, 4\}$, $B := \{0, 3, 4, 5\}$. Bestimmen Sie $A \cup B$, $A \cap B$, $A \setminus B$, $B \setminus A$, $A \times B$, $B \times A$. Skizzieren Sie diese Mengen!

Fingerübung B (Abbildungen). Die Abbildung $f \colon \{0, 1, 2\} \longrightarrow \{0, 1, 2\}$ sei gegeben durch $0 \longmapsto 2$, $1 \longmapsto 0$, $2 \longmapsto 0$. Ist $f \circ f \circ f$ injektiv? Ist $f \circ f \circ f$ surjektiv? Mussten Sie dafür $f \circ f \circ f$ wirklich ausrechnen?

Fingerübung C (surjektiv, injektiv). Seien X, Y Mengen und sei $f \colon X \longrightarrow Y$ eine Abbildung. Welche der folgenden Formeln sind äquivalent zur Injektivität bzw. Surjektivität von f ? Was bedeuten die anderen Formeln?

1. $\forall_{y \in Y} \quad \exists_{x \in X} \quad f(x) = y$
2. $\exists_{x \in X} \quad \forall_{y \in Y} \quad f(x) = y$
3. $\forall_{x \in X} \quad \forall_{x' \in X} \quad \big((f(x) = f(x')) \implies (x = x')\big)$
4. $\forall_{x \in X} \quad \forall_{x' \in X} \quad \big((f(x) \neq f(x')) \implies (x = x')\big)$

Aufgabe 1 (Mengenoperationen; 4 Punkte). Welche der folgenden Aussagen sind wahr? Begründen Sie Ihre Antwort mit einem Beweis oder Gegenbeispiel!

1. Für alle Mengen A und B gilt $(A \cup B) \cup A = A \cup B$.
2. Für alle Mengen A und B gilt $(A \cup B) \setminus B = A$.

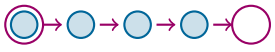
Hinweis. Wie beweist man eine $\forall$-Aussage? Wie widerlegt man eine $\forall$-Aussage durch ein (konkretes!) Gegenbeispiel?

Aufgabe 2 (Kompositionen; 4 Punkte).

1. Sei $f \colon \{0, 1, 2\} \longrightarrow \{0, 1, 2, 3\}$ gegeben durch $0 \longmapsto 2$, $1 \longmapsto 0$, $2 \longmapsto 3$. Geben Sie eine Abbildung $g \colon \{0, 1, 2, 3\} \longrightarrow \{0, 1, 2\}$ an, die $g \circ f = \mathrm{id}_{\{0,1,2\}}$ erfüllt (und begründen Sie Ihre Antwort).
2. Zeigen Sie: Sind A, B Mengen und sind $f \colon A \longrightarrow B$, $g \colon B \longrightarrow A$ Abbildungen mit $g \circ f = \mathrm{id}_A$, so ist f injektiv.

Aufgabe 3 (Potenzmengen sind „groß“; 4 Punkte). Sei X eine Menge und sei $f \colon X \longrightarrow P(X)$ eine Abbildung. Zeigen Sie, dass f *nicht* surjektiv ist.
Hinweis. Verwenden Sie *Reductio ad absurdum*. Liegt $\{x \in X \mid x \notin f(x)\}$ im Bild von f ?! Argumentieren Sie wie im Russellschen Paradoxon ...

Bonusaufgabe (sets, lists, arrays; 4 Punkte). Wählen Sie eine Programmiersprache, in der es Datentypen/-strukturen für Mengen, Listen und Arrays gibt. Vergleichen Sie diese! Achten Sie insbesondere darauf, welche Operationen zur Verfügung gestellt werden (Konstruktoren, Eliminatoren, Kombinatoren) und, wenn möglich, welche Laufzeiten bzw. welcher Speicherplatzverbrauch auftreten.
Hinweis. Belegen Sie Ihre Aussagen durch geeignete Quellen!


Abgabe bis 6. November 2023, 10:00, via GRIPS

Grundlagen der Mathematik$^{\text{FIDS}}$: Übungen

Prof. Dr. C. Löh/PD Dr. F. Strunk/M. Uschold Blatt 4, 6. November 2023

Fingerübung A (Summen). Bestimmen Sie $\sum_{j=1}^{2023}(2\cdot j)$ und $\sum_{j=1}^{2023}(2\cdot j+1)$ jeweils sowohl durch Rückführung auf bekannte Ergebnisse als auch per vollständiger Induktion.

Fingerübung B (Rekursion). Wir definieren $f\colon \mathbb{N} \longrightarrow \mathbb{N}$ rekursiv durch

$$\begin{aligned} f(0) &:= 42 \\ f(n+1) &:= 2023 \cdot f(n) \quad \text{für alle } n \in \mathbb{N}. \end{aligned}$$

Geben Sie eine geschlossene Darstellung für f an und beweisen Sie diese per Induktion. Was passiert, wenn man den Startwert von 42 auf 0 ändert?

Fingerübung C (Wiederholung). Begründen Sie jeweils Ihre Antwort!

1. Seien A, B aussagenlogische Variablen. Ist $(A \Longrightarrow B) \Longrightarrow ((\neg A) \vee B)$ eine aussagenlogische Tautologie?
2. Geben Sie ein Beispiel für eine Abbildung $\{0,2,4,6\} \longrightarrow \{0,3\}$, die nicht surjektiv ist. Wieviele Beispiele können Sie finden?

Aufgabe 1 (Quadratsummen; 4 Punkte). Welche der folgenden Aussagen sind wahr? Begründen Sie Ihre Antwort mit einem Beweis oder Gegenbeispiel!

1. Für alle $n \in \mathbb{N}$ gilt $6 \cdot \sum_{j=1}^{n} j^2 = n \cdot (n+1) \cdot (2 \cdot n + 1)$.
2. Für alle $n \in \mathbb{N}$ gilt $2 \cdot \sum_{j=1}^{n} j^2 = n^2 \cdot (n^2+1)$.

Aufgabe 2 (geometrische Summen; 4 Punkte). Zeigen Sie per vollständiger Induktion: Für alle $n \in \mathbb{N}$ gilt

$$1 + \sum_{j=0}^{n} 2^j = 2^{n+1}.$$

Aufgabe 3 (Blorxscher Gleichheitssatz; 4 Punkte). Was ist falsch am nachfolgenden „Beweis“? Geben Sie genau an, an welcher Stelle etwas schiefgeht und erklären Sie den Fehler!

Behauptung. Alle Außerirdischen haben dieselbe Anzahl an Füßen.

Beweis. Wir zeigen per vollständiger Induktion: Ist $n \in \mathbb{N}$, so besitzen in jeder Menge von genau n Außerirdischen alle dieselbe Anzahl an Füßen.

- *Induktionsanfang.* Ist $n = 0$ oder $n = 1$, so ist die Behauptung wahr.
- *Induktionsvoraussetzung.* Sei nun $n \in \mathbb{N}$ und die Behauptung sei für n Außerirdische bereits gezeigt.
- *Induktionsschritt.* Wir zeigen, dass die Behauptung auch für $n+1$ Außerirdische gilt: Wir numerieren die Außerirdischen als $A_0, \dots, A_n$. Nach Induktionsvoraussetzung haben $A_0, \dots, A_{n-1}$ bzw. $A_1, \dots, A_n$ jeweils dieselbe Anzahl an Füßen. Durch Betrachtung von A_1 folgt, dass dann auch $A_0, \dots, A_n$ alle dieselbe Anzahl an Füßen haben müssen. □

Bitte wenden

Bonusaufgabe (geometrische Summen, geometrisch; 4 Punkte)**.** Schreiben Sie ein LaTeX-Makro \geomsum mit einem Argument, so dass Bilder nach dem folgenden Schema erstellt werden:

\geomsum{1}

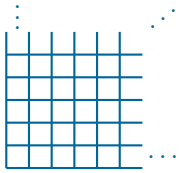
\geomsum{2}

\geomsum{3}

\geomsum{4}

Dokumentieren Sie Ihre Lösung und erklären Sie zusätzlich, was diese Bilder mit Aufgabe 2 zu tun haben!
Hinweis. Sie können mit der Vorlage squares.tex beginnen.

Bonusaufgabe (Infinisudoku; 4 Punkte)**.** Zeigen Sie: Man das nach rechts und oben unendliche Gitter



so mit natürlichen Zahlen füllen, dass in jeder Zeile und in jeder Spalte jede natürliche Zahl genau einmal auftritt.
Hinweis. Versuchen Sie zunächst, das Problem systematisch für Quadrate der Seitenlänge $1, 2, 4, \ldots$ zu lösen und kombinieren Sie diese Lösungen induktiv.

Bonusaufgabe (Neunomanie; 4 Punkte)**.** Wir definieren $f\colon \mathbb{N} \longrightarrow \mathbb{N}$ rekursiv durch

$$\begin{aligned} f(0) &:= 9 \\ f(n+1) &:= 3 \cdot f(n)^4 + 4 \cdot f(n)^3 \quad \text{für alle } n \in \mathbb{N}. \end{aligned}$$

Zeigen Sie, dass die Dezimaldarstellung von $f(11)$ mehr als 2023 Neunen enthält.
Hinweis. Es gilt $2^{11} > 2023$. Mit wievielen Neunen endet $10^m - 1$?!

Bonusaufgabe (Kommutativität der Addition; 4 Punkte)**.** Zeigen Sie wie folgt die Kommutativität der induktiv definierten Addition auf $\mathbb{N}$; Sie dürfen dabei die Assoziativität der Addition auf $\mathbb{N}$ verwenden.

1. Zeigen Sie: Für alle $n \in \mathbb{N}$ gilt $0 + n = n$.
2. Zeigen Sie: Für alle $n \in \mathbb{N}$ gilt $n + 1 = 1 + n$.
3. Zeigen Sie: Für alle $m, n \in \mathbb{N}$ gilt $m + n = n + m$.

Bonusaufgabe (Listen in Lean 4; 4 Punkte)**.** Listen (über einem gegebenen Basisdatentyp) sind in Lean 4 in Init/Prelude.lean wie folgt (induktiv!) definiert:

```
inductive List (α : Type u) where
/-- `[]` is the empty list. -/
| nil : List α
/-- If `a : α` and `l : List α`, then `cons a l`, or `a :: l`,
    is the list whose first element is `a` and with `l`
    as the rest of the list. -/
| cons (head : α) (tail : List α) : List α
```

Formulieren Sie (in Analogie zum Induktionsprinzip für die natürlichen Zahlen) in natürlicher Sprache ein Induktionsprinzip zum Beweis von Aussagen über Listen in Lean 4.

Abgabe bis 13. November 2023, 10:00, via GRIPS

Grundlagen der Mathematik$^{\text{FIDS}}$: Übungen

Prof. Dr. C. Löh/PD Dr. F. Strunk/M. Uschold Blatt 5, 13. November 2023

Fingerübung A (Graphenflora)**.** Stellen Sie die folgenden ungerichteten Graphen „graphisch“ dar. Welche dieser Graphen sind Bäume? Zusammenhängend? Sind diese Graphen isomorph?

$$\big(\{1,\dots,4\},\{\{1,2\},\{1,3\},\{2,3\}\}\big), \qquad \big(\{1,\dots,4\},\{\{1,4\},\{2,4\},\{3,4\}\}\big)$$

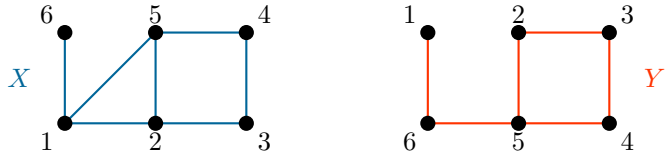
Fingerübung B (Graphen zählen)**.** Wieviele ungerichtete Graphen gibt es, deren Knotenmenge $\{1,\dots,83\}$ ist?

Hinweis. Sie können auch erstmal mit einer kleinen 83 experimentieren.

Fingerübung C (Wiederholung)**.** Berechnen Sie die folgenden Summen:

$$\sum_{j=0}^{n} j, \quad \sum_{k=0}^{42} k, \quad \sum_{n=0}^{42} n, \quad \sum_{j=0}^{41} 42, \quad \sum_{j=1}^{41} 42, \quad \sum_{n=0}^{41} (n+83)$$

Aufgabe 1 (isomorphe Graphen? 4 Punkte)**.** Wir betrachten die Graphen, die durch folgende Skizzen beschrieben werden:



Welche der folgenden Aussagen sind wahr? Geben Sie jeweils die formale Beschreibung dieser Graphen und begründen Sie Ihre Antwort!

1. Der Graph Y ist ein Baum.
2. Die Graphen X und Y sind isomorph.

Aufgabe 2 (Blätter von Binärbäumen; 4 Punkte)**.** Zeigen Sie per vollständiger Induktion: Ist $n \in \mathbb{N}$ und ist (T, v) ein binärer Wurzelbaum der Höhe n, so besitzt T höchstens 2^n Blätter.

Aufgabe 3 (doppelt gezählt hält besser; 4 Punkte)**.**

1. Sei $X = (V, E)$ ein endlicher Graph. Zeigen Sie:

$$\sum_{v \in V} \deg_X(v) = 2 \cdot \#E.$$

   Hinweis. Hierbei ist $\sum_{v\in V}$ analog zu $\sum_{j=1}^{n}$ definiert; wenn Sie möchten, können Sie auch annehmen, dass $V = \{1,\dots,n\}$ ist. Gehen Sie induktiv vor oder betrachten Sie die Menge $\{(v,w) \mid \{v,w\} \in E\}$.

2. Folgern Sie: Die Anzahl der Studenten der Universität Regensburg, die mit einer ungerade Anzahl anderer Studenten der UR eine gemeinsame Vorlesung besuchen, ist gerade.

   Hinweis. Sie werden Eigenschaften von geraden/ungeraden natürlichen Zahlen verwenden; Sie müssen diese nicht beweisen, aber Sie sollten klar formulieren, welche Eigenschaften Sie verwenden.

Bitte wenden

Bonusaufgabe (Blorxscher Geheimdienst; 4 Punkte)**.** Der Blorxsche Geheimdienst besteht aus mindestens sieben (mysteriös!) Agenten.

1. Zeigen Sie, dass folgendes gilt: Es gibt drei Agenten im Blorxschen Geheimdienst, die sich alle paarweise kennen, oder es gibt drei Agenten im Blorxschen Geheimdienst, die sich alle paarweise nicht kennen.

   Hinweis. Modellieren Sie die Situation durch einen geeigneten Graphen und vergessen Sie nicht, dass unter je drei Schuhen mindestens zwei linke oder mindestens zwei rechte sind. Beachten Sie außerdem, dass „oder“ nicht dasselbe ist wie „entweder . . . oder . . . “ !

2. Was passiert, wenn man „sieben“ durch „fünf“ ersetzt? Begründen Sie Ihre Antwort!

Abgabe bis 20. November 2023, 10:00, via GRIPS

Grundlagen der Mathematik$^{\mathsf{FIDS}}$: Übungen

Prof. Dr. C. Löh/PD Dr. F. Strunk/M. Uschold Blatt 6, 20. November 2023

Fingerübung A (eine studentische Relation)**.** Wir betrachten die Relation „ist im selben Studiengang eingeschrieben wie“ auf der Menge aller Studenten der UR. Ist diese Relation reflexiv? Irreflexiv? Symmetrisch? Antisymmetrisch? Transitiv? Eine Äquivalenzrelation? Eine partielle Ordnung? Eine totale Ordnung? Eine Abbildung?

Fingerübung B (Äquivalenzklassen)**.** Wir betrachten auf $\{0, \dots, 10\}$ die Relation $\{(x, y) \mid x - y \text{ ist gerade}\}$.

1. Zeigen Sie, dass diese Relation eine Äquivalenzrelation ist.
2. Bestimmen Sie die Äquivalenzklassen dieser Relation.

Fingerübung C (Wiederholung)**.**

1. Geben Sie ein Beispiel für einen zusammenhängenden Graphen, der genau fünf Knoten besitzt, wovon genau zwei den Grad 3 besitzen.
2. Geben Sie ein Beispiel für einen binären Wurzelbaum mit genau 42 Blättern. Welche Höhe hat Ihr Beispiel?

Aufgabe 1 (Relationship; 4 Punkte)**.** Wir betrachten die Relation „ist verheiratet mit“ auf der Menge aller Bürger in Deutschland. Welche der folgenden Aussagen sind wahr? Begründen Sie Ihre Antwort!

1. Diese Relation ist symmetrisch.
2. Diese Relation ist transitiv.

Aufgabe 2 (Teiler von 42; 4 Punkte)**.** Sei $Z \subset \{1, \dots, 42\}$ die Menge der Teiler von 42. Wir betrachten die partielle Ordnung $T := \big\{(x, y) \bigm| x, y \in Z,\ x \text{ teilt } y\big\}$ auf Z.

1. Skizzieren Sie den Graphen $X := (Z, T)$ in sinnvoller Weise.
2. Ist die partielle Ordnung T total?
3. Besitzt die partielle Ordnung T ein maximales Element?
4. Gibt es im Graphen X einen gerichteten Weg von 3 nach 14 ?

Begründen Sie Ihre Antworten!

Aufgabe 3 (Modulo-Rechnung; 4 Punkte)**.** Sei $n \in \mathbb{N}$ und sei $\sim_n$ die Relation $\big\{(x, y) \bigm| n \text{ teilt } x - y\big\}$ auf $\mathbb{Z}$. Wir betrachten auf $\mathbb{Z}/n := \mathbb{Z}/\sim_n$ die in der Vorlesung definierte Addition und Multiplikation.

1. Zeigen Sie, dass $\sim_n$ eine Äquivalenzrelation auf $\mathbb{Z}$ ist.
2. Gibt es ein $x \in \mathbb{Z}/42$ mit $[5] \cdot x = [3]$ (in $\mathbb{Z}/42$) ? Begründen Sie Ihre Antwort!

Bitte wenden

Bonusaufgabe (Wurzel aus 42; 4 Punkte)**.** Zeigen Sie, dass es $x \in \mathbb{Z}/2023$ mit $x^2 = [42]$ gibt. Gehen Sie dabei wie folgt vor:

1. Schreiben Sie ein Programm, das eine solche Lösung x findet und dokumentieren/erklären Sie Ihr Programm.
2. Finden Sie einen Beweis, dass x eine Lösung ist, den man von Hand (ohne „komplizierte“ Rechnung) nachvollziehen kann.

   Hinweis. Dies wird erfordern, dass Sie andere Dinge tun als in Ihrem Programm!

Abgabe bis 27. November 2023, 10:00, via GRIPS

Grundlagen der Mathematik$^{\mathsf{FIDS}}$: Übungen

Prof. Dr. C. Löh/PD Dr. F. Strunk/M. Uschold Blatt 7, 27. November 2023

Fingerübung A (Wurzel aus 5, Schritt 1)**.** Zeigen Sie, dass $\{x \in \mathbb{R} \mid x^2 < 5\}$ nicht-leer und beschränkt ist (ohne „$\sqrt{5}$“ zu verwenden!).

Fingerübung B (komplexe Zahlen)**.**

1. Bestimmen Sie den Real- und Imaginärteil von
$$\frac{-4 + 2 \cdot i}{9 + 6 \cdot i}.$$
2. Sei $\zeta := 1/2 \cdot (\sqrt{3} \cdot i - 1)$. Zeigen Sie, dass $\zeta^2 \neq 1$ und $\zeta^3 = 1$ ist. Skizzieren Sie die Elemente in der komplexen Ebene!

Fingerübung C (Bits)**.** Welchen logischen Operationen entsprechen Addition und Multiplikation auf $\mathbb{Z}/2$ unter den folgenden Übersetzungen?

- 0 entspricht w; 1 entspricht f.
- 0 entspricht f; 1 entspricht w.

Fingerübung D (Zusammenfassung)**.** Welche Themen wurden in der Vorlesung behandelt? Was sind die wichtigen Definitionen, Beispiele, Sätze, Beweistechniken? Was ist Ihr Lieblingsthema? Was ist Ihre Nemesis?

Aufgabe 1 (Ungleichungen; 4 Punkte)**.** Welche der folgenden Aussagen sind wahr? Begründen Sie Ihre Antwort!

1. $\forall_{x,y \in \mathbb{R}} \quad y \leq x + y$
2. $\forall_{x,y \in \mathbb{R}} \quad x \cdot y \leq x \cdot (x + y)$

Aufgabe 2 (ganz komplex; 4 Punkte)**.** Bestimmen Sie reelle Zahlen x und y mit der Eigenschaft, dass
$$(x + y \cdot i) \cdot \frac{2023 - i}{42 + 11 \cdot i}$$
eine ganze Zahl ungleich 0 ist. Begründen Sie Ihre Antwort!

Aufgabe 3 (Wurzel aus 5, Schritt 2; 4 Punkte)**.** Sei $A := \{x \in \mathbb{R} \mid x^2 < 5\}$. Nach Fingerübung A existiert aufgrund der Vollständigkeit von $\mathbb{R}$ die reelle Zahl $z := \sup A$. Die untenstehenden Aussagen zeigen kombiniert, dass $z^2 = 5$ ist. Lösen Sie einen der beiden Aufgabenteile (ohne „$\sqrt{5}$“ zu verwenden!):

1. Zeigen Sie, dass $z^2 \not< 5$ ist.

   Hinweis. Betrachten Sie $(z + \varepsilon)^2$, wobei $\varepsilon = (5 - z^2)/20$ ist und erinnern Sie sich an die Definition des Supremums.

2. Zeigen Sie, dass $z^2 \not> 5$ ist.

   Hinweis. Betrachten Sie $(z - \varepsilon)^2$, wobei $\varepsilon = (z^2 - 5)/2 \cdot z$ ist und erinnern Sie sich an die Definition des Supremums.

Bitte wenden

Bonusaufgabe (Fibonaccizahlen, reloaded; 4 Punkte)**.** Implementieren Sie in einer Programmiersprache Ihrer Wahl die Funktionen $F, G\colon \mathbb{N} \longrightarrow \mathbb{R}$ mit

$$\begin{aligned} F(0) &:= 1 \\ F(1) &:= 1 \\ F(n+2) &:= F(n) + F(n+1) \quad \text{für alle } n \in \mathbb{N} \end{aligned}$$

bzw. (wobei $\varphi := \frac{1+\sqrt{5}}{2}$ und $\overline{\varphi} := \frac{1-\sqrt{5}}{2}$)

$$G\colon n \longmapsto \frac{\varphi^{n+1} - \overline{\varphi}^{n+1}}{\sqrt{5}}.$$

Liefern $F(32)$ und $G(32)$ dasselbe Ergebnis? Wenn ja, warum? Wenn nein, warum nicht? Dokumentieren und erklären Sie Ihr Programm! Welche Datentypen verwenden Sie für die auftretenden Zahlen?

Die folgenden Bonusaufgaben bieten die Gelegenheit, einen etwaigen Punkterückstand aufzuholen und die Themen der Vorlesung nochmal zu üben.

Bonusaufgabe (tauto-logisch; 4 Punkte)**.** Seien A, B, C aussagenlogische Variablen. Sind die folgenden aussagenlogischen Formeln Tautologien? Begründen Sie jeweils Ihre Antwort!

1. $\big((A \wedge B) \vee C\big) \Longrightarrow \big((B \wedge (\neg C)) \vee A\big)$
2. $\big((A \Longrightarrow B) \Longrightarrow C\big) \vee (\neg A) \vee (\neg B) \vee (\neg C)$

Bonusaufgabe ($*$-jektiv; 4 Punkte)**.** Ist die folgende Abbildung injektiv? Surjektiv? Bijektiv? Begründen Sie jeweils Ihre Antwort!

$$\begin{aligned} \mathbb{Z} &\longrightarrow \mathbb{Z} \\ x &\longmapsto x^2 + 42 \end{aligned}$$

Bonusaufgabe (harmonische Summen; 4 Punkte)**.** Zeigen Sie per Induktion, dass folgende Ungleichung für alle $n \in \mathbb{N}$ gilt:

$$\sum_{j=1}^{2^n} \frac{1}{j} > \frac{n}{2}.$$

Hinweis. Es ist $\{1, \ldots, 2^{n+1}\} = \{1, \ldots, 2^n\} \cup \{2^n + 1, \ldots, 2^{n+1}\}$.

Bonusaufgabe (Relationsdesign; 4 Punkte)**.** Geben Sie ein Beispiel für eine Äquivalenzrelation mit genau 2023 Äquivalenzklassen, wobei jede Äquivalenzklasse genau 42 Elemente enthält. Begründen Sie Ihre Antwort!

Bonusaufgabe (Baumschule; 4 Punkte)**.** Geben Sie ein Beispiel für einen endlichen Graphen (V, E) mit folgender Eigenschaft: Es ist $E \neq \emptyset$ und für alle $e \in E$ ist $(V, E \setminus \{e\})$ ein Baum. Begründen Sie Ihre Antwort!

Bonusaufgabe (komplexe Bits; 4 Punkte)**.** Commander Blorx findet $\mathbb{C}$ zu unübersichtlich und $\mathbb{Z}/2$ zu winzig. Er überlegt sich daher, ob man nicht analog zur Konstruktion von $\mathbb{C}$ aus $\mathbb{R}$ auch einen Körper $\mathbb{B}$ (Blorx-Bits!) aus $\mathbb{Z}/2$ konstruieren könnte, der genau vier Elemente enthält. Funktioniert das? Begründen Sie Ihre Antwort!
Hinweis. Evtl. kann auch ein Programm bei der Beantwortung hilfreich sein!

Abgabe bis 4. Dezember 2023, 10:00, via GRIPS

Dies ist das letzte Übungsblatt der Vorlesung *Grundlagen der Mathematik*.

Probeklausur zu Grundlagen der Mathematik$^{\mathsf{FIDS}}$

Prof. Dr. C. Löh/PD Dr. F. Strunk/M. Uschold 27. November 2023

Name: Vorname:

Matrikelnummer: ☐☐☐☐☐☐☐ Übungsleiter:

- Diese Klausur besteht aus 7 Seiten. Bitte überprüfen Sie, ob Sie alle Seiten erhalten haben.
- Bitte versehen Sie *alle* Seiten mit Ihrem Namen/Ihrer Matrikelnummer.
- Bitte schreiben Sie Lösungen zu verschiedenen Aufgaben auf verschiedene Blätter.
- Beginn: 16:10. Sie haben 90 Minuten Zeit, um die Klausur zu bearbeiten; bitte legen Sie Ihren Studentenausweis oder Lichtbildausweis zu Beginn der Klausur vor sich auf den Tisch. Um Unruhe in den letzten Minuten zu vermeiden, geben Sie bitte entweder um 17:40 Uhr oder vor 17:20 Uhr ab.
- Die Klausur besteht aus 6 Aufgaben. Es können im Total 60 Punkte erreicht werden. Zum Bestehen genügen voraussichtlich 50% der Punkte.
- Es sind keinerlei Hilfsmittel wie Taschenrechner, Computer, Bücher, Vorlesungsmitschriften, Mobiltelephone etc. gestattet; Papier wird zur Verfügung gestellt. *Alle* Täuschungsversuche führen zum Ausschluss von der Klausur; die Klausur wird dann als nicht bestanden gewertet!

☐ Ich habe die Studienleistung für diese Vorlesung bestanden, werde diese aber *nicht* als Studienleistung einbringen und stattdessen max. 3 Bonuspunkte für die Klausur erhalten; Bonuspunkte werden nur vergeben, wenn die Klausur bestanden ist.

Falls ich die Studienleistung der Vorlesung Lineare Algebra I nicht bestehe, kann ich das Modul *nicht* im WS 23/24 abschließen. Im Rahmen des Moduls kann die Anrechnung von Bonuspunkten ausschließlich fr eine der beiden Modulteilprüfungen erfolgen. Wurde die Anrechnung von Bonuspunkten bereits für eine Modulteilprüfung beantragt, ist ein weiterer Antrag für eine andere Modulteilprüfung nicht zulässig und bleibt unberücksichtigt.

Unterschrift des Teilnehmers:

Viel Erfolg!

Aufgabe	1	2	3	4	5	6	Summe	Bonus?	Summe
Punkte maximal	10	10	15	10	10	5	60	max. 3	60
erreichte Punkte									

Note: Unterschrift:

Aufgabe 1 (3+2+5 = 10 Punkte)**.** Seien A und B aussagenlogische Variablen.

1. Welchen Wert erhält $(\neg B) \wedge (A \Longrightarrow B)$, wenn wir A mit w und B mit f belegen? Geben Sie alle Zwischenschritte an.

2. Zeichnen Sie den Syntaxbaum für $(\neg B) \wedge (A \Longrightarrow B)$.

3. Ist $(A \vee B) \Longrightarrow (A \wedge B)$ eine aussagenlogische Tautologie?

   Begründen Sie Ihre Antwort!

Aufgabe 2 $(2 + 3 + 5 = 10$ Punkte$)$**.**

1. Bestimmen Sie alle Elemente der Menge $\{x \in \mathbb{N} \mid x \leq 2\} \cap \{0, 42, 2023\}$. Geben Sie alle Zwischenschritte an.

2. Geben Sie ein Beispiel für eine Abbildung $f\colon \{0,1,2\} \longrightarrow \{0,1,2\}$ mit folgender Eigenschaft:

$$\exists_{y\in\{0,1,2\}} \quad \forall_{x\in\{0,1,2\}} \quad f(x) \neq y.$$

   Begründen Sie Ihre Antwort!

3. Sei $f\colon \mathbb{N} \longrightarrow \mathbb{N}$ surjektiv. Ist dann auch $f \circ f\colon \mathbb{N} \longrightarrow \mathbb{N}$ surjektiv?

   Begründen Sie Ihre Antwort!

Aufgabe 3 ($1 + 5 + 4 + 5 = 15$ Punkte)**.** Wir betrachten die rekursiv durch

$$\begin{aligned} f(0) &= 5 \\ f(n+1) &= f(n) + 3 \quad \text{für alle } n \in \mathbb{N} \end{aligned}$$

definierte Abbildung $f\colon \mathbb{N} \longrightarrow \mathbb{N}$.

1. Bestimmen Sie $f(2)$. Geben Sie alle Zwischenschritte an.
2. Beweisen Sie per Induktion, dass $f(n) = 5 + 3 \cdot n$ für alle $n \in \mathbb{N}$ gilt.
3. Gibt es ein $n \in \mathbb{N}$ mit $f(n) = 2023$? Begründen Sie Ihre Antwort!
4. Tausendfüßler Millo wird mit genau drei Füßen geboren. Am n-ten Tag nach seiner Geburt wachsen ihm jeweils genau n zusätzliche Füße. Wieviele Füße hat Millo nach dem 88.-ten Tag? Begründen Sie Ihre Antwort!

Aufgabe 4 (1+4+5 = 10 Punkte)**.** Wir betrachten den ungerichteten Graphen

$$X := \big(\{1, \ldots, 6\}, \{\{1,2\}, \{1,4\}, \{2,5\}, \{3,6\}, \{5,2\}\}\big).$$

1. Skizzieren Sie X.
2. Ist X ein Baum? Begründen Sie Ihre Antwort!
3. Geben Sie zwei nicht-isomorphe Bäume mit jeweils genau sechs Knoten an. Begründen Sie Ihre Antwort!

Aufgabe 5 ($2 + 4 + 4 = 10$ Punkte)**.**

1. Wann heißt eine Relation auf einer Menge *transitiv*?
2. Ist die Relation $\{(x, 2 \cdot x) \mid x \in \mathbb{Z}\}$ auf $\mathbb{Z}$ eine Äquivalenzrelation? Begründen Sie Ihre Antwort!
3. Geben Sie ein Beispiel für ein $x \in \mathbb{Z}/13$ mit $[5] \cdot x = [1]$ (in $\mathbb{Z}/13$). Begründen Sie Ihre Antwort!

Aufgabe 6 $(3 + 2 = 5$ Punkte$)$**.**

1. Bestimmen Sie den Imaginärteil von $\frac{2+3\cdot i}{3+2\cdot i}$.
   Geben Sie alle Zwischenschritte an.
2. Ist $\mathbb{Q}$ vollständig? Begründen Sie Ihre Antwort!

C
Quellcode

Die Quellen finden sich auf der Homepage zur Vorlesung:

https://loeh.app.ur.de/teaching/fids_ws2324

- expressions.lean: Eine Implementierung von Semantiken aussagenlogischer Formeln in Lean (Kapitel 1.4).
- penguins.lean: Eine Implementierung eines einfachen Beweises in Lean (Kapitel 2.4).
- functions.lean: Elementare funktionale Programmierung in Lean (Kapitel 3.3).

D
Organisatorisches

Grundlagen der Mathematik$^{\text{FIDS}}$ und Lineare Algebra I$^{\text{FIDS}}$ Organisatorisches

Prof. Dr. C. Löh/PD Dr. F.Strunk/M. Uschold Oktober 2023

Homepage. Alle aktuellen Informationen zur Vorlesung, zu den Übungen, zu Sprechstunden, Literaturangaben, sowie die Übungsblätter/Lesepläne finden Sie auf der Homepage zur Vorlesung bzw. in GRIPS:

https://loeh.app.ur.de/teaching/fids_ws2324

https://elearning.uni-regensburg.de

Vorlesung. Die Vorlesung findet jeweils montags (10:15–11:45; H 3) statt. Die erste Vorlesung ist am Montag, den 16. Oktober, um 10:15. Das Vorlesungsskript ist über die Vorlesungshomepage und GRIPS zugänglich; das Skript wird jeweils nach der Vorlesung im Verlauf des Tages aktualisiert.

- Der Kurs *Grundlagen der Mathematik (FIDS)* besteht aus sieben Vorlesungen;
- der Kurs *Lineare Algebra I (FIDS)* besteht aus acht Vorlesungen.

Ich möchte alle Teilnehmer dazu ermutigen, sich aktiv an der Vorlesung zu beteiligen und Fragen zu stellen bzw. zu beantworten. Deshalb möchte ich die Atmosphäre so locker, informell und unverbindlich wie möglich halten.

Übungen. Die neuen Übungsaufgaben werden wöchentlich montags spätestens um 10:00 Uhr auf den obigen Homepages online gestellt und sind bis zum darauffolgenden Montag um 10:00 Uhr abzugeben (via GRIPS).

Auf jedem Übungsblatt gibt es drei reguläre Aufgaben (je 4 Punkte) und herausforderndere Bonusaufgaben (je 4 Bonuspunkte).

Sie dürfen und sollen die Aufgaben in kleinen Gruppen bearbeiten; aber die Lösungen müssen individuell ausformuliert und aufgeschrieben werden, andernfalls werden die Punkte aberkannt. Sie dürfen (müssen aber nicht!) Lösungen zu zweit abgeben; in diesem Fall müssen selbstverständlich jeweils beide Autoren in der Lage sein, *alle* der Zweiergruppe abgegebenen Lösungen zu präsentieren, andernfalls werden die Punkte aberkannt.

Bitte geben Sie Ihre Lösungen als pdf-Datei ab (erzeugt von LaTeX & Co. oder Scans handschriftlicher Bearbeitungen; bitte *keine* Worddokumente, Markdown, ...); etwaigen Quellcode bitte als Textdatei abgeben.

Die Übungsgruppen beginnen in der zweiten Vorlesungswoche; in diesen ersten Übungen wird die Blatt 1 (Abgabe am 23. 10. 2023) besprochen.

Da aus organisatorischen Gründen in der ersten Vorlesungswoche noch keine Übungsgruppen stattfinden können, gibt es ein Blatt 0 mit Lösungsvorschlägen. Bitte beachten Sie dabei, dass dass es oft viele Möglichkeiten gibt, Aufgaben korrekt und vollständig zu lösen und dass dies nur eine Auswahl ist. Bei Fragen können Sie das Frageforum in GRIPS verwenden.

Zentralübung. Zusätzlich zur Vorlesung und den Übungen bietet die Zentralübung die Gelegenheit, Fragen zu stellen, den Stoff der Vorlesung zu wiederholen und weitere Beispiele zu behandeln. Die Zentralübung findet dienstags (16:00–18:00; H 4) statt und wird von Florian Strunk bzw. Matthias Uschold geleitet. Die Zentralübung beginnt in der *zweiten* Vorlesungswoche und ist ein freiwilliges Zusatzangebot.

Fingerübungen. Zusätzlich enthalten das Skript und die Übungsblätter Fingerübungen, die elementare Techniken und Begriffe trainieren. Diese Aufgaben sollten im Idealfall so einfach sein, dass sie innerhalb weniger Minuten gelöst werden können. Diese Aufgaben werden nicht abgegeben bzw. korrigiert. Die Fingerübungen auf den Übungsblättern werden ab Blatt 2 in den Übungsgruppen gemeinsam behandelt.

Einteilung in die Übungsgruppen. Die Einteilung in die Übungsgruppen erfolgt in der ersten Vorlesungswoche über die Studiengangskoordination der FIDS. Bei sonstigen Fragen zum Übungsbetrieb wenden Sie sich bitte an Matthias Uschold (matthias.uschold@ur.de) oder Florian Strunk (florian.strunk@ur.de).

Prüfungs-/Studienleistungen. Die Kurse *Grundlagen der Mathematik* und *Lineare Algebra I* können wie in den Modulkatalogen spezifiziert in die Studiengänge BSc Informatik bzw. BSc Data Science eingebracht werden.

- *Studienleistung*: Erfolgreiche Teilnahme an den Übungen: In
  - *Grundlagen der Mathematik* oder
  - *Lineare Algebra I*

  mindestens 50% der (in den regulären Aufgaben) möglichen Punkte, mindestens einmal zufriedenstellend vorrechnen.

  Es ist *sehr* empfehlenswert, die Studienleistung zu beiden Kursen abzulegen! Erfahrungsgemäß sind die Erfolgsaussichten in Klausuren in Mathematik sehr gering, wenn der Übungsbetrieb nicht erfolgreich absolviert wurde.
- *Prüfungsleistung*:
  - Klausur (90 Minuten) zu *Grundlagen der Mathematik* und
  - Klausur (90 Minuten) zu *Lineare Algebra I*.

Um das Modul *Mathematik I FIDS* erfolgreich zu belegen, müssen beide Klausuren bestanden sein (d.h. jeweils Note 4.0 oder besser). Die Gesamtnote des Moduls ist das arithmetische Mittel der beiden Einzelnoten.

Interessanter Twist: Wenn Sie *beide* Übungsbetriebe bestanden haben, wird bei dem Kurs, bei dem sie den Übungsbetrieb nicht als die verpflichtende Studienleistung für das Modul einbringen, die Teilnote (bei der Korrektur) wie folgt verbessert: Jeweils bei den Noten 4.0 bis 1.3 wird die Note um eine Stufe (0.3 bzw. 0.4) auf die nächstbessere Note verbessert. Insbesondere muss die Klausur dafür bestanden sein!

Sie müssen sich in FlexNow für die Studienleistung und die Prüfungsleistung anmelden. Bitte informieren Sie sich frühzeitig. Berücksichtigen Sie bitte auch (implizite) Fristen der entsprechenden Prüfungsordnungen bis wann (Wiederholungs-)Prüfungen abgelegt werden müssen.

Klausurtermine. Die Klausuren finden zu folgenden Terminen statt:

- *Grundlagen der Mathematik.* 04.12.2023, 16:00, H 1 (Audimax),
- *Lineare Algebra I.* 04.03.2024, 11:00, H 1 (Audimax).

Die Termine für die Wiederholungsklausuren werden rechtzeitig bekanntgegeben. Die Wiederholungsklausur für *Grundlagen der Mathematik* wird zu einem ähnlichen Termin wie die Klausur zur *Linearen Algebra I* stattfinden (nämlich am 04.03.2024, 13:15, H 1). Die Wiederholungsklausur zur *Linearen Algebra I* wird gegen Ende der Semesterferien stattfinden (voraussichtlich am 08.04.2024).

Wichtige Informationen im Krankheitsfall finden Sie unter:

https://www.uni-regensburg.de/studium/startseite/verhalten-krankheitsfall/index.html

Ansprechpartner.

- Bei Fragen zur Organisation des Übungsbetriebs wenden Sie sich bitte an die Studiengangskoordination der FIDS bzw. an Matthias Uschold oder Florian Strunk:

  matthias.uschold@ur.de, florian.strunk@ur.de

- Bei Fragen zu den Übungsaufgaben wenden Sie sich bitte an Ihren Übungsleiter oder fragen Sie in der Zentralübung.
- Bei mathematischen Fragen zur Vorlesung wenden Sie sich bitte an Ihren Übungsleiter, an Matthias Uschold, Florian Strunk oder an Clara Löh.
- Bei Fragen zur Planung Ihres Studiums bzw. zur Prüfungsordnung wenden Sie sich bitte an die zuständige Studienberatung oder das zuständige Prüfungsamt.

Grundlagen der Mathematik$^{\mathsf{FIDS}}$

Hinweise zu den Übungsaufgaben

Prof. Dr. C. Löh/PD Dr. F. Strunk/M. Uschold Oktober 2023

Ziel der Übungsaufgaben. Ziel der Übungsaufgaben ist, sich aktiv mit den behandelten Definitionen, Sätzen, Beispielen und Beweistechniken auseinanderzusetzen und zu lernen, damit umzugehen. Insbesondere ist der Weg das Ziel: Es ist wertvoller, eigenständig eine Aufgabe suboptimal zu bearbeiten als eine korrektere Lösung von anderen zu übernehmen. Nutzen Sie den Luxus, dass Sie für Ihre Abgaben *individuelle* Rückmeldung erhalten!

Das Punkteminimum für die Studienleistung ist das *Minimum*. Sie sollten versuchen, möglichst viele Punkte zu erreichen und nicht nach Erreichen dieser Minimalzahl die Übungen schleifen lassen!

Wie bearbeitet man eine Übungsaufgabe?

- Beginnen Sie mit der Bearbeitung an dem Tag, an dem das Übungsblatt erscheint – manche Dinge brauchen einfach ein paar Tage Zeit.
- Lesen Sie sich alle Aufgaben gründlich durch. Kennen Sie alle auftretenden Begriffe? Verstehen Sie, was in den Aufgaben verlangt wird?
- Was sind die Voraussetzungen? Was ist zu zeigen? Wie könnten diese Dinge zusammenhängen? Gibt es Sätze aus der Vorlesung, die auf diese Situation passen?
- Welche Lösungsstrategien bzw. Beweisstrategien passen auf die Aufgabe? Kann man einfach direkt mit den Definitionen arbeiten und so zum Ziel gelangen?
- Ist die Aufgabe plausibel? Versuchen Sie die behaupteten Aussagen, an einfachen Beispielen nachzuvollziehen!
- Falls Sie die Aufgabe unplausibel finden, können Sie versuchen, sie zu widerlegen und untersuchen, woran dieses Vorhaben scheitert.
- Kann man die Situation durch eine geeignete Skizze graphisch darstellen?
- Versuchen Sie, das Problem in kleinere Teilprobleme aufzuteilen. Können Sie diese Teilprobleme lösen?
- Verwenden Sie viel Schmierpapier und geben Sie sich genug Zeit, an der Aufgabe herumzuexperimentieren! Selbst wenn Sie die Aufgabe nicht vollständig lösen, werden Sie auf diese Weise viel lernen, da Sie sich aktiv mit den Begriffen und Sätzen auseinandersetzen.
- Wenn Sie nicht weiterwissen, diskutieren Sie die Aufgabe mit Kommilitonen. Lassen Sie sich aber auf keinen Fall dazu verleiten, einfach Lösungen irgendwo abzuschreiben oder ausschließlich in Gruppen zu arbeiten. Mathematik kann man nur lernen, wenn man aktiv damit arbeitet und seine Gedanken selbst formuliert!

Wie schreibt man eine Lösung auf?

- Gliedern Sie Ihre Lösung sauber in Voraussetzung, Behauptung und Beweis.
- Teilen Sie Ihre Beweise in sinnvolle Zwischenschritte auf.
- Achten Sie darauf, dass Sie verständlich formulieren und dass die Argumente logisch aufeinander aufbauen.
- Ist Ihre Argumentationskette wirklich lückenlos? Seien Sie misstrauisch gegenüber Ihrer eigenen Lösung und versuchen Sie, alle potentiellen Schwachpunkte ausfindig zu machen!
- Wenn Sie einzelne Beweisschritte nicht vollständig durchführen können, können Sie in Ihrer Lösung darauf hinweisen – die restliche Lösung kann trotzdem Punkte erhalten.
- Achten Sie darauf, dass Sie alle Bezeichner einführen und dass Sie mathematische Symbole und Fachbegriffe korrekt verwenden.
- Versuchen Sie, sich so präzise wie möglich auszudrücken.
- Versuchen Sie, indirekte Argumente so weit wie möglich zu vermeiden.
- Überprüfen Sie am Ende, ob Sie wirklich das bewiesen haben, was Sie ursprünglich behauptet haben.
- Oft ist es auch hilfreich zu überprüfen, ob/wie alle in der Aufgabe gegebenen Voraussetzungen verwendet wurden.
- Würden Sie Ihre Lösung verstehen, wenn Sie sie zum ersten Mal lesen würden?
- Alles, was Sie abgeben, müssen Sie eigenständig formuliert und auch verstanden haben.
- Geben Sie Literaturangaben an, wenn Sie zusätzliche Quellen verwendet haben.

Bewertungskriterien. Bei der Bewertung der abgegebenen Lösungen wird auf folgendes geachtet:

- Wurde die gestellte Aufgabe vollständig gelöst?
- Wurden Voraussetzung, Behauptung, Beweis deutlich voneinander getrennt?
- Stimmen die Voraussetzungen? Sind sie sauber formuliert?
- Stimmen die Behauptungen/Zwischenbehauptungen? Sind sie sauber formuliert?
- Ist die Argumentationskette der Beweisschritte vollständig?
- Sind die Beweisschritte präzise formuliert und verständlich?
- Sind alle Bezeichner eingeführt?
- Werden mathematische Symbole und Fachbegriffe korrekt eingesetzt?
- Ist an jeder Stelle des Beweises klar, was passiert?
- Werden die neu erlernten Begriffe und Techniken passend eingesetzt?

Viel Erfolg und viel Spass bei den Übungen!

Grundlagen der Mathematik$^{\text{FIDS}}$
Hinweise zur Prüfungsvorbereitung

Prof. Dr. C. Löh/PD Dr. F. Strunk/M. Uschold Oktober 2023

Ziel der Prüfungsvorbereitung. Hauptziel der Prüfungsvorbereitung ist die souveräne Beherrschung des behandelten Fachgebiets. Die Prüfung sichert ab, dass dies tatsächlich der Fall ist, ist aber nicht das eigentliche inhaltliche Ziel der Vorlesung.

Beherrscht werden sollten also:

- aktive Kenntnis der Fachbegriffe und Formalisierungsmethoden
- Verständnis der Ideen, die zu diesen Fachbegriffen und Formalisierungen führen
- wichtige Probleme und Fragestellungen, die das Gebiet maßgeblich beeinflusst haben bzw. die durch das Gebiet gelöst werden können
- wichtige Resultate und Zusammenhänge innerhalb des Gebiets
- wichtige Beweis- und Lösungsstrategien
- repräsentative Beispiele
- Anwendungen des Gebiets und Interaktion mit anderen Gebieten
- Fähigkeit, auf all diesen Kenntnissen weiter aufzubauen.

Erreichen dieses Ziels. Während der Vorlesungszeit:

- aktive Auseinandersetzung mit den Übungsaufgaben
- Erlernen des Fachwissens (Definitionen, Sätze), notfalls mit Karteikarten
- weiteres aktives Üben mit zusätzlichen Aufgaben und Vertiefung der Kenntnisse durch Selbststudium (Bibliothek und Computer-Werkzeuge)
- Bei Fragen: Betreuungsangebote nutzen!

Kurz vor der Prüfung:

- Kann ich mein Wissen präzise und verständlich präsentieren? (Das kann man einfach an anderen Kommilitonen ausprobieren ...)
- Was könnten typische Prüfungsfragen sein? Was sind gute Lösungen zu diesen Fragen?
- Wie belastbar sind meine Fähigkeiten? Was muss ich noch verbessern?

Bewertungskriterien. In der Prüfung werden folgende Fähigkeiten abgeprüft:

- Fachwissen (Definitionen, Sätze, Beweise, Beispiele, Anschauung, Zusammenhänge, Anwendungen, ...)
- präzises und korrektes, logisch schlüssiges, Formulieren und Argumentieren
- Lösen von Standardproblemen
- Kreativität bei der Lösung von Problemen
- Es werden keine Programmieraufgaben gestellt.

Viel Erfolg bei der Prüfung!

Literaturverzeichnis

Bitte beachten Sie, dass das Literaturverzeichnis im Laufe der Vorlesung wachsen wird und sich daher auch die Nummern der Quellen ändern werden!

[1] K. Appel, W. Haken, Every map is four colourable, *Bull. Amer. Math. Soc.*, 82, pp. 711–712, 1976. Cited on page: 75

[2] R.D. Arthan. The Eudoxus Real Numbers, preprint, arXiv:math.HO/0405454, 2004. Cited on page: 95, 96

[3] J. Avigad, L. de Moura, S. Kong. *Theorem Proving in Lean*, https://leanprover.github.io/theorem_proving_in_lean/, 2023. Cited on page: 16, 29

[4] A. Bauer. Proof of negation and proof by contradiction, https://math.andrej.com/2010/03/29/proof-of-negation-and-proof-by-contradiction/ Cited on page: 28

[5] A. Beutelspacher. *Das ist o.B.d.A. trivial!*, neunte Auflage, Vieweg+-Teubner, 2009. Cited on page: 23

[6] Y. Bertot, P. Castéran. *Interactive Theorem Proving and Program Development. Coq'Art: The Calculus of Inductive Constructions*, Texts in Theoretical Computer Science, An EATCS Series, Springer, 2004. Cited on page: 12

[7] K. Buzzard, J. Eugster. *The natural number game*, Lean 4-Version, 2023. https://adam.math.hhu.de/#/g/hhu-adam/NNG4 Cited on page: 55

[8] P.J. Cameron. *Sets, Logic and Categories*, Universitext, Springer, 1998. Cited on page: 12, 21

[9] J. Davies. Collatz Graph: All Numbers Lead to One, 2012. https://www.jasondavies.com/collatz-graph/ Cited on page: 69

[10] H.-D. Ebbinghaus et. al.. *Zahlen*, Springer, 1992. Cited on page: 95, 96, 98

[11] H.-D. Ebbinghaus, J. Flum, W. Thomas. *Einführung in die mathematische Logik*, 5. Auflage, Spektrum Akademischer Verlag, 2007. Cited on page: 12

[12] U. Friedrichsdorf, A. Prestel. *Mengenlehre für den Mathematiker*, Vieweg, 1985. Cited on page: 44

[13] Geometric Interactive, Annapurna Interactive. *Cocoon*, Computerspiel, 2023. https://www.cocoongame.com/ Cited on page: 55

[14] G. Gonthier. Formal proof–the four-color theorem, *Notices Amer. Math. Soc.*, 55(11), S. 1382–1393, 2008.
Implementierung in Coq: https://github.com/math-comp/fourcolor Cited on page: 75

[15] R. Gu, Z. Shao, H. Chen, J. Kim, J. Koenig, X. Wu, V. Sjöberg, D. Costanzo. Building Certified Concurrent OS Kernels, *Commun. ACM*, 09/2019, S. 89–99, 2019. Cited on page: 29

[16] J. Harrison. Theorem Proving with the Real Numbers. Technical report, University of Cambridge Computer Laboratory, 1996. Cited on page: 95, 103

[17] G. Klein, K. Elphinstone, G. Heiser, J. Andronick, D. Cock, P. Derrin, D. Elkaduwe, K. Engelhardt, R. Kolanski, M. Norrish, T. Sewell, H. Tuch, S. Winwood. seL4: Formal Verification of an OS Kernel, *Proceedings of the ACM SIGOPS 22nd Symposium on Operating Systems Principles*, ACM, 2009. Cited on page: 29

[18] D.E. Knuth. *Surreal numbers*, Addison–Wesley, 1974. Cited on page: 55

[19] A.G. Konforowitsch. *Logischen Katastrophen auf der Spur*, zweite Auflage, Fachbuchverlag Leipzig, 1994. Cited on page: 27

[20] Lean community. Learning Lean,
https://leanprover-community.github.io/learn.html Cited on page: 16, 29

[21] Lean community. Lean web editor,
https://lean.math.hhu.de/ Cited on page: 16, 29

[22] C. Löh. 1, 2, 3, ... zu viele. Was haben verschiedene Unendlichkeiten mit Computern zu tun? *Universität für Kinder*, Universität Regensburg, 2019.
https://loeh.app.uni-regensburg.de/seminars/einszweidrei_booklet.pdf
Cited on page: 104

[23] C. Löh. *Algebraic Topology*, Vorlesungsskript, WS 21/22, Universität Regensburg, 2022.
https://loeh.app.ur.de/teaching/topologie1_ws2122/lecture_notes.pdf
Cited on page: 28

[24] C. Löh. *Exploring Formalisation. A Primer in Human-Readable Mathematics in Lean 3 with Examples from Simplicial Topology*, Surveys and Tutorials in the Applied Mathematical Sciences, 11, Springer, 2022. Cited on page: 29

[25] M. Lipovača. *Learn You a Haskell for Great Good!: A Beginner's Guide*, No Starch Press, 2011. http://learnyouahaskell.com/ Cited on page: 43

[26] *The On-Line Encyclopedia of Integer Sequences*, https://oeis.org/ Cited on page: 58

[27] R.L. Rivest, A. Shamir, L.M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Comm. ACM*, 21(2), pp. 120–126, 1978. Cited on page: 86

[28] R.M. Smullyan, M. Fitting. *Set theory and the continuum problem*, überarbeitete Auflage, Dover, 2010. Cited on page: 44, 88, A.5, A.6

[29] Structured Query Language/NULLs and the Three Valued Logic.
https://en.wikibooks.org/wiki/Structured_Query_Language/NULLs_and_the_Three_Valued_Logic Cited on page: 12

[30] Patrick Traynor. *Patrick's Parabox*, Computerspiel, Audio/Musik von Priscilla Snow, 2022. https://www.patricksparabox.com/ Cited on page: 55

Deutsch → English

A

Abbildung	map	31
Abschluss	closure	79
Absolutbetrag	absolute value	100
abzählbar	countable	A.5
algebraisch abgeschlossen	algebraically closed	98
Allquantor	all quantifier	12
angeordneter Körper	ordered field	94
Äquivalenz	equivalence	10
Äquivalenzklasse	equivalence class	80
Äquivalenzrelation	equivalence relation	80
Auswahlaxiom	axiom of choice	45

B

Baum	tree	63, 70
beschränkt	bounded	94
Beweis	proof	19
Beweisassistent	proof assistant	29
beweisen	prove (verb)	19
Binärbaum	binary tree	71
binärer Wurzelbaum	rooted binary tree	71
Blatt	leaf	70

C

Collatz-Vermutung	Collatz conjecture	69

D

dicht	dense	95

E

echte Klasse	proper class	46
Einschränkung	restriction	39
endlich	finite	A.4
exakte Arithmetik	exact arithmetic	101
Existenzquantor	existential quantifier	12

F

Fakultätsfunction	factorial function	55
Fibonacci-Zahl	Fibonacci number	57
Fortsetzung	extension	39
Funktion	function	31
Funktion höherer Ordnung	higher order function	42
funktionale Programmierung	functional programming	42

G

Galois-Korrespondenz	Galois connection	88
ganze Zahl	integer	83
genau dann, wenn	if and only if; iff	10
Generalisierungsregel	generalisation rule	21
gerichteter Graph	directed graph	67
gleichmächtig	same cardinality	A.5
goldener Schnitt	golden ratio	57
Graph	graph	63
Graphenisomorphismus	graph isomorphism	66

H

höchstens abzählbar	at most countable	A.5
Höhe	height	72

I

Identitätsabbildung	identity map	38
imaginäre Einheit	imaginary unit	99
Imaginärteil	imaginary part	99
Implikation	implication	10
Induktion	induction	49
Infimum	infimum	94

K

Kante	edge	64, 67
Kardinalität	cardinality	A.4
Klasse	class	45
Knoten	vertex	64, 67
Komplement	complement	33
komplexe Zahl	complex number	91
Komposition	composition	38
Kontinuumshypothese	continuum hypothesis	A.6
Kontraposition	contraposition	11
Körper	field	92

Korrektheit	soundness	21
Kreis	cycle	68

L

leere Menge	empyt set	33
Logik	logic	7

M

Mächtigkeit	cardinality	A.4
Menge	set	31
Mengenlehre	set theory	31
Modell	model	63

O

obere Schranke	upper bound	88

P

partielle Ordnung	partial order	87
Peano-Axiome	Peano axioms	50
Potenzmenge	power set	33
Produkt	product	33

Q

Quantor	quantifier	12
Quasimorphismus	quasi-morphism	95
Quotient	quotient	81

R

rationale Zahl	rational number	84
Realteil	real part	99
reelle Zahl	real number	91
reine funktionale Programmierung	pure functional programming	43
Rekursion	recursion	51
Relation	relation	77
Repräsentation fester Länge	fixed length representation	101

S

Schnittmenge	intersection	33
Semantik	semantics	7
semantische Regel	semantic rule	9
Supremum	supremum	94
Syntax	syntax	7

T

Tautologie	tautology	11
Teiler	divisor	85
Teilmenge	subset	33

teilt	divides	85
totale Ordnung	total order	87

U

überabzählbar	uncountable	A.5
Umkehrabbildung	inverse map	40
unendlich	infinite	A.5
ungerichteter Graph	undirected graph	64
untere Schranke	lower bound	88
Unvollständigkeitssatz	incompleteness theorem	45

V

Vereinigung	union	33
vollständiger angeordneter Körper	complete ordered field	95
vollständiger Graph	complete graph	64
Vollständigkeit	completeness	21

W

Weg	path	68, 69
Widerspruchsbeweis	proof by contradiction	11
Wohlordnung	well-ordering	88
Wurzel	root	96
Wurzelbaum	rooted tree	71

Z

zusammenhängend	connected	68

English → Deutsch

A

absolute value	Absolutbetrag	100
algebraically closed	algebraisch abgeschlossen	98
all quantifier	Allquantor	12
at most countable	höchstens abzählbar	A.5
axiom of choice	Auswahlaxiom	45

B

binary tree	Binärbaum	71
bounded	beschränkt	94

C

cardinality	Mächtigkeit, Kardinalität	A.4
class	Klasse	45
closure	Abschluss	79
Collatz conjecture	Collatz-Vermutung	69
complement	Komplement	33
complete graph	vollständiger Graph	64
complete ordered field	vollständiger angeordneter Körper	95
completeness	Vollständigkeit	21
complex number	komplexe Zahl	91
composition	Komposition	38
connected	zusammenhängend	68
continuum hypothesis	Kontinuumshypothese	A.6
contraposition	Kontraposition	11
countable	abzählbar	A.5
cycle	Kreis	68

D

dense	dicht	95
directed graph	gerichteter Graph	67
divides	teilt	85
divisor	Teiler	85

E

edge	Kante	64, 67
empyt set	leere Menge	33
exact arithmetic	exakte Arithmetik	101
existential quantifier	Existenzquantor	12
extension	Fortsetzung	39

F

factorial function	Fakultätsfunction	55
Fibonacci number	Fibonacci-Zahl	57
finite	endlich	A.4
fixed length representation	Repräsentation fester Länge	101
function	Funktion	31
functional programming	funktionale Programmierung	42

G

Galois connection	Galois-Korrespondenz	88
generalisation rule	Generalisierungsregel	21
golden ratio	goldener Schnitt	57
graph	Graph	63
graph isomorphism	Graphenisomorphismus	66

H

height	Höhe	72
higher order function	Funktion höherer Ordnung	42

I

identity map	Identitätsabbildung	38
if and only if	genau dann, wenn	10
iff	genau dann, wenn	10
imaginary part	Imaginärteil	99
imaginary unit	imaginäre Einheit	99
implication	Implikation	10
incompleteness theorem	Unvollständigkeitssatz	45
induction	Induktion	49
infimum	Infimum	94
infinite	unendlich	A.5
integer	ganze Zahl	83
intersection	Schnittmenge	33
inverse map	inverse Abbildung/Umkehrabbildung	

L

leaf	Blatt	70
logic	Logik	7
lower bound	untere Schranke	88

M

map	Abbildung	31
model	Modell	63

O

ordered field	angeordneter Körper	94

P

partial order	partielle Ordnung	87
path	Weg	68, 69
Peano axioms	Peano-Axiome	50
power set	Potenzmenge	33
product	Produkt	33
proof	Beweis	19
proof assistant	Beweisassistent	29
proof by contradiction	Widerspruchsbeweis	11
proper class	echte Klasse	46
prove (verb)	beweisen	19
pure functional programming	reine funktionale Programmierung	43

Q

quantifier	Quantor	12
quasi-morphism	Quasimorphismus	95
quotient	Quotient	81

R

rational number	rationale Zahl	84
real number	reelle Zahl	91
real part	Realteil	99
recursion	Rekursion	51
relation	Relation	77
restriction	Einschränkung	39
root	Wurzel	96
rooted binary tree	binärer Wurzelbaum	71
rooted tree	Wurzelbaum	71

S

same cardinality	gleichmächtig	A.5
semantic rule	semantische Regel	9
semantics	Semantik	7
set	Menge	31
set theory	Mengenlehre	31
soundness	Korrektheit	21
subset	Teilmenge	33

supremum	Supremum	94
syntax	Syntax	7

T

tautology	Tautologie	11
total order	totale Ordnung	87
tree	Baum	63, 70

U

undirected graph	ungerichteter Graph	64
union	Vereinigung	33
upper bound	obere Schranke	88

V

vertex	Knoten	64, 67

W

well-ordering	Wohlordnung	88

Index

B

C

D

E

Q

R

S

T

Finger weg! Gaaaaa! . . . Das war der lila Knopf. Jetzt kann alles mögliche passieren! Aber fürs Drucken kann das durchaus nützlich sein . . .