Formale Grundlagen der Informatik

WiSe 2023/24

Universität Trier

Henning Fernau
Mitarbeiter: Kevin Goergen, Kevin Mann
Universität Trier
fernau@uni-trier.de

Formale Grundlagen der Informatik | Gesamtübersicht

1. Rechnen: Gesetze und Regeln

2. Modellieren und Formalisieren:
   Keine Angst vor Formalismen . . . und etwas Logik

3. Warum stimmt das eigentlich? Logik und Beweisverfahren

4. Herangehen an Aufgaben aus Informatik und Mathematik

Ziele der heutigen Veranstaltung

- Verstärktes Aufzeigen,
  was Mathematikunterricht an der Universität bedeuten kann.
- Unterbrechen Sie mich bitte, wenn Sie eine Folie nicht verstanden haben.
- . . . sonst haben Sie Schwierigkeiten, “dem Rest” zu folgen.
- Weiteres (wichtiges) Ziel:
  Rekapitulation der “Grundlagen der elementaren Logik”
  aus dem Sommersemester

Beweisverfahren—Warum stimmt das eigentlich?

Es gibt verschiedene Arten der Beweisführung:
direkte Beweise
indirekte Beweise / Beweise durch Widerspruch
konstruktive / algorithmische Beweise (sehr informatisch)
nicht-konstruktive Beweise (siehe Chomp)
Induktionsbeweise

Für alle diese Beweise haben Sie hoffentlich in der Schule Beispiele gesehen.

Für die Informatik ist Beweisen wichtig insbesondere im Zusammenhang mit der Korrektheit von Programmen und ihrer Laufzeit.

Logik für Informatiker

- Schaltkreislogik
- Also: Nähe zur elektrotechnischen Umsetzung
- Aber auch: Universelle mathematische Verkehrssprache
- Denkschule

Logik: Wahr oder falsch?

Eine *Aussage* ist ein sprachliches Konstrukt, das wahr oder falsch sein kann.

Manchmal ist es leicht, den Wahrheitsgehalt einer Aussage herauszufinden:

1. Paris ist die Hauptstadt von Frankreich.
2. Mäuse sind bekannt dafür, Kängurus zu jagen.

Die erste Aussage ist *offenkundig* wahr, die zweite ist falsch.
Die *Offenkundigkeit* hängt von unserem Weltwissen ab, wir wollen hiervon im Folgenden abstrahieren.
Beobachte: Die beiden Aussagen lassen sich nicht (syntaktisch) in Teilaussagen zerlegen, sie sind vielmehr *atomar*.

Mathematische Aussagen

Wir betrachten die folgenden atomaren Aussagen:

- 2 ist Teiler von 12.
- 3 ist Teiler von 12.
- 4 ist Teiler von 12.
- 5 ist Teiler von 12.
- 6 ist Teiler von 12.
- 7 ist Teiler von 12.

Zwei dieser atomaren Aussagen sind falsch, die übrigen sind wahr.

Von den folgenden zusammengesetzten Aussagen ist nur eine falsch.

- 2 ist Teiler von 12 **oder** 5 ist Teiler von 12.
- 2 ist Teiler von 12 **und** 5 ist Teiler von 12.
- 2 **und** 3 sind Teiler von 12.
- **Falls** 2 Teiler von 12 ist, **dann** ist 4 Teiler von 12.
- **Falls** 5 Teiler von 12 ist, **dann** ist 2 Teiler von 12.
- 2 ist Teiler von 12 **oder** **sowohl** 5 **als auch** 6 sind Teiler von 12.

Vorsicht Umgangssprache

Die folgenden zwei zusammengesetzten Aussagen meinen offenbar dasselbe:

- 5 ist Teiler von 12 **und** 2 ist Teiler von 12.
- 2 ist Teiler von 12 **und** 5 ist Teiler von 12.

Umgangssprachlich ist das aber bei der “UND-Verknüpfung” der folgenden beiden atomaren Aussagen nicht der Fall:

- Otto wird krank.
- Der Arzt verordnet Otto Medizin.

Das UND hat in der Allgemeinsprache auch noch einen zeitlichen Aspekt. Dieser wird in der Aussagenlogik vernachlässigt.

Rechengesetze für wahr und falsch

1 und 0 als Grundelemente.
Grundmenge ist also: $\{0, 1\}$.
Operatoren und Rechengesetze:

- Negation:

	$\neg$
0	1
1	0

- Konjunktion:

$$0 \wedge 0 = 0$$
$$0 \wedge 1 = 0$$
$$1 \wedge 0 = 0$$
$$1 \wedge 1 = 1$$

Die Verwandtschaft zur Schaltkreislogik ist offenbar.
http://www.dietrichgrude.de/informatik/schaltlogik.htm

Grundüberlegungen zur formalen Logik

Wir gehen davon aus, es gäbe eine Menge $\mathfrak{A}$ von *atomaren Formeln*.

Über einen Teil $\mathfrak{D} \subseteq \mathfrak{A}$ dieser Formeln "wissen wir Bescheid", d.h., wir können eine Abbildung $\beta : \mathfrak{D} \to \{0, 1\}$ angeben mit der Bedeutung:

- $\beta(\mathfrak{a}) = 0$, falls $\mathfrak{a}$ falsch ist;
- $\beta(\mathfrak{a}) = 1$, falls $\mathfrak{a}$ wahr ist.

$\mathfrak{D}$ ist also eine Menge definierter Aussagen, und β ist eine *Belegungsfunktion*.
$\mathfrak{A} \setminus \mathfrak{D}$: *logische Variablen* oder *Unbestimmte*.
Wir wollen dann *zusammengesetzte Aussagen* untersuchen.
Wir beschreiben nun, was das formal bedeutet.

Formalitäten: Die Syntax der Aussagenlogik

(Aussagenlogische) Formeln werden durch einen induktiven Prozess definiert:

- Jede atomare Formel ist eine Formel.
  Formaler: Ist $F \in \mathfrak{A}$, so ist F eine Formel.
- Ist F eine Formel, so auch $\neg F$. *Negation von* F
- Sind F und G Formeln, so auch $(F \wedge G)$. *Konjunktion von* F *und* G

Eine Formel, die als Teil einer Formel in F auftritt, heißt *Teilformel* von F.
$\mathfrak{F}$ bezeichne die Gesamtheit aller aussagenlogischen Formeln (bzgl. $\mathfrak{A}$).

Beispiel: Sind A, B, C Formeln, so auch $F = \neg((\neg(A \wedge B) \wedge C) \wedge \neg C)$.
$(A \wedge B)$ ist eine Teilformel von F. $(A \wedge B) \wedge C$ ist keine Teilformel von F, ebensowenig $(B \wedge A)$.
Überlegen Sie: Wie sieht die Menge aller Teilformeln von F aus?
Muss eine Teilformel notgedrungen als Formel im induktiven Aufbau einer Formel auftreten?

Übliche Abkürzungen

Wir werden im Folgenden zwei weitere Elemente von Formeln als Abkürzungen bekannter Formeln kennenlernen.

- $(F \vee G)$ steht für $\neg(\neg F \wedge \neg G)$. *Disjunktion von F und G*
- $(F \rightarrow G)$ steht für $(\neg F \vee G)$. *Implikation*
- $(F \leftrightarrow G)$ steht für $((F \rightarrow G) \wedge (G \rightarrow F))$. *Äquivalenz*

Die Objekte $\neg$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$ heißen auch *Junktoren*.
Sie dienen dazu, Teilformeln zu verbinden.
Aufgabe: Wofür steht $(F \leftrightarrow G)$ gemäß unserer ursprünglichen Formeldefinition?

Wir haben bislang nicht festgelegt, was diese Objekte bedeuten sollen.
Die Semantik der Aussagenlogik betrachten wir im Folgenden.

Wozu Formeln formal?

- Wir können einfach feststellen, ob eine Folge von Zeichen eine aussagenlogische Formel darstellt.
- Wir können dies auch programmieren.
- Daher kann ein Computer etwas mit Formeln als Eingabe anfangen.
- Ganz ähnlich kann man z.B. arithmetische Formeln (Ausdrücke, Terme) maschinell verarbeiten.
- Mehr dazu in Veranstaltungen wie “Automaten und Formale Sprachen” oder “Compilerbau”.

Die Semantik der Aussagenlogik: Was sollen die Formeln bedeuten?

$\{0, 1\}$ ist die Menge der *Wahrheitswerte*.
Ggb.: Menge $\mathfrak{A}$ atomarer Formeln, $\mathfrak{F}$ von Formeln (über den atomaren Formeln $\mathfrak{A}$);
Teilmenge $\mathfrak{D} \subseteq \mathfrak{A}$ mit Belegungsfunktion $\beta : \mathfrak{D} \to \{0, 1\}$.
$\mathfrak{E}$ bezeichne die aus $\mathfrak{D}$ aufbaubaren Formeln, also diejenigen Formeln aus $\mathfrak{F}$, die nur Elemente aus $\mathfrak{D}$ als atomare Formeln enthalten.
Wir erweitern β induktiv zu einer Belegungsfunktion $\widehat{\beta} : \mathfrak{E} \to \{0, 1\}$ wie folgt:
Ist $F \in \mathfrak{E}$ atomar, so setze $\widehat{\beta}(F) := \beta(F)$.
Andernfalls unterscheide zwei Fälle:

(a) $F = \neg G$. Setze $\widehat{\beta}(F) := \begin{cases} 0, & \widehat{\beta}(G) = 1 \\ 1, & \widehat{\beta}(G) = 0 \end{cases}$

(b) $F = (G \wedge H)$. Setze $\widehat{\beta}(F) := \begin{cases} 1, & \widehat{\beta}(G) = 1 \text{ und } \widehat{\beta}(H) = 1 \\ 0, & \text{sonst.} \end{cases}$

Nach dieser sauberen Definition werden wir auch für $\widehat{\beta}$ vereinfachend β schreiben.

Wahrheitstafeln für die Grund-Junktoren: Semantik im Überblick

- Negation:

$\beta(G)$	$\beta(\neg G)$
0	1
1	0

für $F = \neg G$

- Konjunktion:

$\beta(G)$	$\beta(H)$	$\beta(G \wedge H)$
0	0	0
0	1	0
1	0	0
1	1	1

für $F = (G \wedge H)$.

Wahrheitstafeln für abgeleitete Junktoren

- Disjunktion:

$\beta(F)$	$\beta(G)$	$\beta(F \vee G)$
0	0	0
0	1	1
1	0	1
1	1	1

, denn

$\beta(F)$	$\beta(G)$	$\beta(\neg F)$	$\beta(\neg G)$	$\beta((\neg F \wedge \neg G))$	$\beta(\neg(\neg F \wedge \neg G))$
0	0	1	1	1	0
0	1	1	0	0	1
1	0	0	1	0	1
1	1	0	0	0	1

Aufgabe: Leiten Sie entsprechend die Wahrheitstafeln für $\rightarrow$ und $\leftrightarrow$ her.

Ein Beispiel: Formelauswertung bei gegebener Belegung

Es sei $\mathfrak{D} = \{p, q\}$ vorgegeben mit $\beta(p) = 1$ und $\beta(q) = 0$.
Betrachte die Formel $F = ((p \to q) \vee p)$. Was liefert $\beta(F)$?
1. Rückführen auf die ursprüngliche Definition:
F steht für $((\neg p \vee q) \vee p)$ oder $\neg(\neg\neg(\neg\neg p \wedge \neg q) \wedge \neg p)$.
2. Benutze die induktive Definition als rekursive Berechnungsvorschrift:
Um $\beta(F)$ zu berechnen, benötigen wir nach (a) $\beta(F')$ für $F' = (\neg\neg(\neg\neg p \wedge \neg q) \wedge \neg p)$.
Dazu bestimme wegen (b): $\beta(G')$ und $\beta(H')$ mit $G' = \neg\neg(\neg\neg p \wedge \neg q)$ und $H' = \neg p$.
Mit (a) und wegen $\beta(p) = 1$ folgt: $\beta(H') = 0$.
Die Definition der Semantik der Konjunktion zeigt, dass dann (unabh. von $\beta(G')$) $\beta(F') = 0$ gilt.
Wegen (a) folgt also: $\beta(F) = 1$.

Aufgabe: Berechnen Sie $\beta(G')$ (was wir uns ja "gespart" hatten).
In einer Aufgabe (welcher genau?) hatten Sie im Grunde die Formel $J = (p \leftrightarrow q)$ "zurückgeführt".
Berechnen Sie mit obiger Belegungsfunktion $\beta(J)$.

Und weiter?

- Wie soeben begonnen, kann man die sogenannte *Aussagenlogik* entwickeln.
- Diese bildet die Grundlage für das Verständnis vieler Beweisverfahren, wie wir sehen werden.
- Erweiterung: Prädikatenlogik mit *Quantoren* $\forall$ und $\exists$.
- Als Nächstes: Anwendung für Korrektheitsüberlegungen für Programme.

Korrektheit von Programmstücken—ein einfaches Beispiel:

`swap(x,y)` soll die Werte der beiden Variablen `x` und `y` vertauschen.
Dies Verhalten wurde beim Sortieralgorithmus verwendet.
Oft gibt es `swap` nicht als elementaren Befehl.
$\rightsquigarrow$ `swap` muss erst durch eine Folge einfacherer *Zuweisungen* simuliert werden,
unter Zuhilfenahme einer *Hilfsvariablen* `h`:

```
h:=x; x:=y; y:=h
```

Warum arbeitet das Programmstück korrekt als Implementierung von `swap` ?
Betrachten wir einen direkten Beweis hierfür:
Eingangs gilt: `x` enthalte den Wert `a` und `y` enthalte den Wert `b`; der Wert von `h` ist beliebig.
Nach dem ersten Befehl enthält mit `x` auch `h` den Wert `a`, die übrigen Werte sind unverändert.
Nach dem zweiten Befehl enthält `h` den Wert `a`, während `x` und `y` den Wert `b` besitzen.
Am Ende enthalten `y` und auch `h` den Wert `a`, während `x` den Wert `b` enthält.
Tatsächlich sind also die Inhalte von `x` und `y` vertauscht worden.

Eine Aufgabe: Überprüfung der Korrektheit von Programmen
Frage: Was berechnet das folgende Programmstück (in Pseudo-Code) ?

1. Lies ganze Zahlen v, x, y, z ein.

2. Setze $u := v \cdot x$. (u ist eine Hilfsvariable für ganze Zahlen.)

3. Setze $u := (u + y) \cdot x$.

4. Setze $u := u + z^2$.

5. Gib u aus.

Beweisverfahren—Warum stimmt das eigentlich ?

Es gibt verschiedene Arten der Beweisführung:
Die voranstehende Argumentation mit *Vorbedingungen* und *Nachbedingungen* ist ein Beispiel für einen direkten Beweis.

Das Schema dieses Beweisverfahrens ist:
Ausgangslage: Es gibt eine Menge W von wahren Aussagen und eine Aussage a, die zu zeigen ist.
Schrittweise wird nun W erweitert:
• Man argumentiert, warum aus den Aussagen $a_1, \ldots, a_n \in W$ die (neue) Aussage a' folgt, die nunmehr in W aufgenommen werden kann.
• Diese Argumentationsweise bricht ab, sobald a in W aufgenommen wurde.

Querbezug für SoSe-Anfänger: *Resolutionskalkül*

Direkte Beweise und Korrektheit von Programmstücken

Korrektheitsbeweise für einfache Programmstücke passen wie folgt in diese Beweisstruktur:

Anfangs enthält W Aussagen über Variablenbelegungen vor dem Durchlauf des Programmstücks, etwa von der Form:

"Unmittelbar vor der Abarbeitung der Zeile 1 enthält die Variable x den Wert 2."

Aus Aussagen der Form: "Unmittelbar vor der Abarbeitung der Zeile i gilt: ..." (Vorbedingungen) werden nun, durch Nachvollziehen des Befehls in der Zeile i, Aussagen der Form "Unmittelbar vor der Abarbeitung der Zeile $\mathrm{i} + 1$ gilt: ..." gewonnen (Nachbedingungen).

Unter der Annahme, das Programm enthalte eine letzte Zeile n mit dem "leeren Befehl" **end**, sollten nun die Aussagen (oder einige der Aussagen) der Form "Unmittelbar vor der Abarbeitung der Zeile n gilt: ..." den Aussagen entsprechen, die man eigentlich über das Programmstück als Nachbedingungen insgesamt beweisen will.

Direkter Beweis—ein einfaches Beispiel

Behauptung: Das Quadrat jeder geraden natürlichen Zahl n ist gerade.

Beweis: Es sei n eine gerade natürliche Zahl.
Da n gerade, lässt sich n eindeutig als $n = 2k$ darstellen; hierbei ist k ist eine natürliche Zahl.
Darauf folgert man (mit Assoziativgesetz):

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

Da $k \cdot 2 \cdot k$ mit k eine natürliche Zahl ist, ist n^2 daher das Doppelte einer natürlichen Zahl und damit gerade. □

Beweis durch Widerspruch (indirekter Beweis, reductio ad absurdum)

Grundidee: Folgt aus einer Aussage p etwas offensichtlich Falsches, so muss das Gegenteil, formal $\neg\mathrm{p}$, richtig sein.

Logische Begründung: $(\mathrm{p} \rightarrow 0) \equiv (\neg\mathrm{p} \vee 0) \equiv \neg\mathrm{p}$.

Wie beim direkten Beweis können wir dabei (evtl. hypothetisch) eine Menge wahrer oder als wahr angenommener Aussagen W nach und nach erweitern, bis dass wir gezwungen wären, sowohl die Wahrheit von q als auch die Wahrheit der Verneinung $\neg\mathrm{q}$ einzuräumen.

Der indirekte Beweis—ein logisches Beispiel

Satz: $(\mathrm{p} \to \mathrm{q})$ ist logisch äquivalent zu $(\neg\mathrm{q} \to \neg\mathrm{p})$.
Bezug zum “Logikteil”: Wahrheitstafelbeweis.
Es folgt ein “kalkülbasierter” Beweis, der auch das “ex falso quodlibet” benutzt.

Beweis: Es seien p und q irgendwelche logischen Aussagen, denen mithin entweder der Wahrheitswert **wahr** oder **falsch** zukommt.
Wir zeigen: Gilt $\mathrm{p} \to \mathrm{q}$, so folgt $(\neg\mathrm{q} \to \neg\mathrm{p})$.
Wegen “ex falso quodlibet” können wir anfangen mit $W = \{\mathrm{p} \to \mathrm{q}, \neg\mathrm{q}\}$.
Wir müssen zeigen, dass (durch Erweiterung von W) schließlich $\neg\mathrm{p}$ in W liegt.
Nehmen wir einmal an, p wäre wahr. Wegen $\mathrm{p} \to \mathrm{q}$ wäre dann aber auch q wahr, im Widerspruch zu $\neg\mathrm{q} \in W$.
Daher ist unsere Annahme falsch, d.h., p ist falsch, also $\neg\mathrm{p}$ kann in W aufgenommen werden.

Die umgekehrte Richtung sieht man genauso (mit Doppelter Negation). □

Hinweis: “Kalküle” sind wesentlich für automatisierte Beweise.

Der vorige Satz ist die Grundlage für:
Beweis durch Umkehrschluss (Kontraposition)

Lemma: Ist eine Quadratzahl ungerade, so auch ihre Wurzel.

Was bedeutet dieser Satz ? Ausführliche Schreibweise:

Es sei a *eine natürliche Zahl.*
Wenn a^2 *eine ungerade Zahl ist, dann ist* a *ungerade.*

Wir haben also die Aussage(forme)n:
$p(a) := (a^2$ ist ungerade$)$ und $q(a) := (a$ ist ungerade$)$.
Unser Satz lautet also: $\forall a \in \mathbb{N}(p(a) \to q(a))$.

Kontraposition $\rightsquigarrow \forall a \in \mathbb{N}(\neg q(a) \to \neg p(a))$.

Beweis durch Umkehrschluss (Kontraposition) (Forts. des Bsp.)

Zu zeigen ist also: *Es sei* a *eine natürliche Zahl.*
Wenn a *keine ungerade Zahl ist, dann ist* a^2 *nicht ungerade.*
Dies ist offensichtlich eine komplizierte Formulierung für:
Ist a gerade, so auch a^2.
Eine Zahl a heißt *gerade* gdw. $2 \mid a$ (2 ist Teiler von a).

Beweis: (nochmal formalisierter aufgeschrieben)
a gerade $\rightarrow (\exists k(a = 2k)) \rightarrow (\exists k(a \cdot a = (2k) \cdot a)) \rightarrow (\exists k(a^2 = 2 \cdot (ka))) \rightarrow a^2$ gerade. □

Satz: Eine Quadratzahl ist genau dann gerade, wenn ihre Wurzel gerade ist.

Aufgabe: Beweisen Sie diesen Satz! Was ist noch zu zeigen, wenn wir die obigen Aussagen benutzen dürfen? Welche "Beweisfigur" haben Sie verwendet?

Beweis durch Widerspruch (indirekter Beweis, reductio ad absurdum)

Grundidee: Folgt aus einer Aussage p etwas offensichtlich Falsches, so muss das Gegenteil, formal $\neg p$, richtig sein.

Satz: $\sqrt{2}$ ist irrational.

Beweis: Wir führen die Annahme, $x = \sqrt{2}$ wäre rational, zum Widerspruch.

x rational und $x > 0 \rightsquigarrow$ es gibt *teilerfremde* $a, b \in \mathbb{N} : x = a/b$.

$\rightsquigarrow 2 = x^2 = a^2/b^2 \rightsquigarrow 2b^2 = a^2$

$\rightsquigarrow a^2$ ist gerade $\rightsquigarrow a$ ist gerade (s. obiger Satz)

$\rightsquigarrow a = 2c$ für eine ganze Zahl $c \rightsquigarrow 2b^2 = 4c^2$

$\rightsquigarrow b^2 = 2c^2 \rightsquigarrow b^2$ ist gerade $\rightsquigarrow b$ ist gerade

Also: 2 ist Teiler von a und von b, im Widerspruch zur Wahl von a und b. □

Alternativer 'modulo-3' bzw. geometrischer Widerspruchsbeweis in:
B. Polster & M. Ross: Putting Two and Two Together, Kapitel 2&3

Ein geometrischer Beweis

Satz: $\sum_{i=1}^{n} i = n + (n-1) + (n-2) + \cdots + 1 = \frac{n(n+1)}{2}$

Beweis: Betrachte ein $n \times n$ Quadrat, unterteilt in n^2 viele Kästchen.
Lege eine Diagonale von links oben bis rechts unten durch das Quadrat.
Von der ersten Spalte schneidet die Diagonale das erste Kästchen.
Von der zweiten Spalte schneidet sie das zweite Kästchen, usf.
Von der letzten Spalte, also der n-ten Spalte, schneidet die Diagonale das letzte Kästchen, also das n-te Kästchen.
Die Anzahl der Kästchen, die oberhalb von der Diagonalen liegen oder von ihr geschnitten werden, entspricht also gerade $\sum_{i=1}^{n} i$.
"Etwa" die Hälfte der Kästchen des Quadrates, also $n^2/2$, werden so gezählt.
Genauer sind es in jeder Spalte ein halbes Kästchen mehr.
Also: $\sum_{i=1}^{n} i = \frac{n^2}{2} + \frac{n}{2} = \frac{n(n+1)}{2}$ □

Peano-Axiome: ein klassisches Beispiel einer *induktiven Definition*

axiomatische Definition der Menge der natürlichen Zahlen $\mathbb{N}$ durch Giuseppe Peano (1889)
eigentlich von Richard Dedekind in “Was sind und was sollen die Zahlen ?” (1888)

1. 0 ist eine natürliche Zahl.
2. Zu jeder natürlichen Zahl n gibt es genau einen Nachfolger n', der ebenfalls eine natürliche Zahl ist.
3. Es gibt keine natürliche Zahl, deren Nachfolger 0 ist.
4. Zwei verschiedene natürliche Zahlen n und m (d.h.: $n \neq m$) besitzen stets verschiedene Nachfolger n' und m' (also: $n' \neq m'$).
5. Enthält eine Menge X die Zahl 0 und überdies mit jeder natürlichen Zahl n auch immer deren Nachfolger n', so enthält X bereits alle natürlichen Zahlen. *Induktionsaxiom*
(Ist X dabei selbst eine Teilmenge der natürlichen Zahlen, dann ist $X = \mathbb{N}$.)

Peano verwendet dabei die Begriffe *0*, *natürliche Zahl* und *Nachfolger*.

Wie sehen natürliche Zahlen aus ? (nach Dedekind / Peano)

$0, 0', 0'', 0''', 0'''', \ldots$

Es ist jedoch bequemer, bei der gewohnten Schreibweise zu bleiben:

$0, 1, 2, 3, 4, \ldots$

Diese ist überdies deutlich kürzer als die induktiv definierte.

Rekursion versus Induktion

Induktiv können wir "der Reihe nach" die definierten Objekte auflisten.

Den umgekehrten Weg geht die Rekursion: n' ist eine natürliche Zahl, wenn n eine ist, und das ist der Fall, wenn entweder $n = 0$ gilt oder aber n von der Form m' ist...

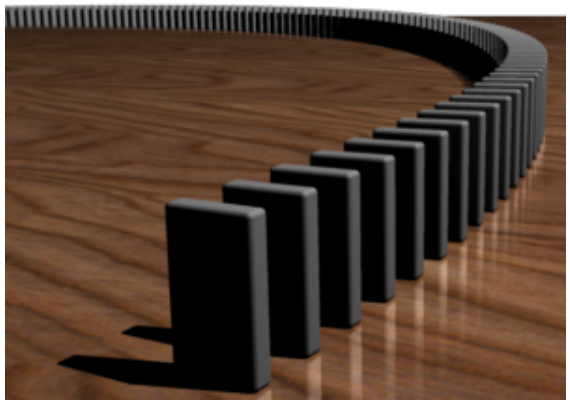
Grundgedanke der mathematischen Induktion

Es sei $p(n)$ eine Aussageform, die von $n \in \mathbb{N}$ abhängt.
Mathematische Induktion ist eine Beweistechnik, die auf dem Induktionsaxiom fußt und schematisch wie folgt arbeitet.

1. *Induktionsanfang* (IA) (auch *Anker* genannt): Zeige $p(0)$.
2. *Induktionsschritt* (IS) Es wird gezeigt, dass für alle $n \in \mathbb{N}$ gilt:
Aus $p(n)$ folgt $p(n+1)$.
$p(n)$ heißt hier auch *Induktionsannahme* oder *Induktionshypothese* (IH) oder *Induktionsvoraussetzung* (IV).
$p(n+1)$ ist die *Induktionsbehauptung* (IB)

Nach dem Prinzip der mathematischen Induktion folgt hieraus:
Für alle natürlichen Zahlen n gilt: $p(n)$.

Induktion veranschaulicht: Der Dominoeffekt:



Die Aufstellung gewährleistet:
Wenn der k-te Dominostein in der Reihe fällt, so auch der $k+1$-te.

Jetzt fällt der erste Dominostein.

Folgerung: Schließlich werden alle Steine umgefallen sein.

Ein gutes Einführungskapitel, das auch wieder unsere Bemerkungen zu den Fibonacci-Zahlen ergänzt, finden Sie hier.

Induktion am Beispiel.

Satz: $\forall n \in \mathbb{N}(n^2 = \sum_{i=1}^{n}(2i-1))$.

Beweis: IA: Die "leere Summe" ist gleich Null, d.h., die Behauptung gilt für $n = 0$.
IS: Angenommen, die Aussage gilt für $n \leq m$. Zu zeigen ist die Aussage für $n = m+1$.
Dann rechnen wir:

$$\begin{aligned}
(m+1)^2 &= m^2 + 2m + 1 && \text{binomischer Lehrsatz} \\
&= (\sum_{i=1}^{m}(2i-1)) + 2m + 1 && \text{IV} \\
&= (\sum_{i=1}^{m}(2i-1)) + 2(m+1) - 1 \\
&= \sum_{i=1}^{m+1}(2i-1)
\end{aligned}$$

Nach dem Prinzip der mathematischen Induktion folgt die Behauptung. □

Alternative: Geometrischer Beweis

Ein weiteres Beispiel: Money, Money, Money, ...

Satz: Jeder Cent-Betrag ≥ 4 Cent kann unter ausschließlicher Verwendung von 2- und 5-Cent-Münzen bezahlt werden.

Beweis: Klar für 4 und für 5 Cent.

Angenommen, wir wissen, wie $n > 4$ Cent bezahlt werden können, nämlich mit n_2 2-Cent-Münzen und mit n_5 5-Cent-Münzen. $\rightsquigarrow n = 2n_2 + 5n_5$.

Da $n > 4$, gilt $n_2 \geq 2$ oder $n_5 \geq 1$ (klar ?).

Wir geben jetzt zwei Regeln an, mit denen wir $n + 1$ Cent mit n_2' 2-Cent-Münzen und mit n_5' 5-Cent-Münzen bezahlen können:

Gilt $n_2 \geq 2$, so setze $n_2' := n_2 - 2$ und $n_5' := n_5 + 1$.

Andernfalls gilt $n_5 \geq 1$. $\rightsquigarrow$ Setze $n_2' = n_2 + 3$, $n_5' := n_5 - 1$.

Probe: $n + 1 = 2n_2 + 5n_5 + 1 = 2(n_2 - 2) + 5(n_5 + 1) = 2(n_2 + 3) + 5(n_5 - 1)$.

Die Behauptung folgt (wie genau ?) mit mathematischer Induktion.

Programmtexte als Formalismen—ein Weg zur Informatik

```
procedure BS(A)
//Die Eingabe A ist eine Liste zu sortierender Gegenstände
  for each i from 1 to length(A) do:
       for each j from length(A) downto i + 1 do:
         if A[ j ] < A[ j-1 ] then
           swap( A[ j ], A[ j-1 ] )
         end if
//A[j-1] ist ein kleinstes Element von A[j-1],...,A[length(A)]
       end for
//A[i] ist ein kleinstes Element von A[i],...,A[length(A)]
//Die Liste A[1],..,A[i] ist aufsteigend sortiert
  end for
end procedure
```

Programmtexte als Formalismen—ein Weg zur Informatik

```
procedure BS(A)
//Die Eingabe A ist eine Liste von n zu sortierenden Gegenständen
  for each i from 1 to n do:
       for each j from n downto i + 1 do:
         if A[ j ] < A[ j-1 ] then
           swap( A[ j ], A[ j-1 ] )
         end if
       end for
//swap wird höchstens n−i mal ausgeführt
  end for
//(Die äußere Schleife wird n mal ausgeführt.)
//Also wird swap insgesamt höchstens (n−1)+(n−2)+···+1 = n(n−1)/2 mal ausgeführt
end procedure
```

Aufgabe: Welche Teile müssten Sie noch beweisen?!
Wie könnte man das Programm noch vereinfachen aufgrund Ihrer / unserer Betrachtungen?

Übungsaufgaben Beweisen Sie mit Hilfe der vollständigen Induktion:

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

2. Arbeiten Sie das “money-money”-Beispiel formal durch.

3. $\sum_{k=1}^{n} \frac{1}{k(k+1)} = 1 - \frac{1}{n+1}$.

4. $1 + x + ... + x^{(n-1)} = \sum_{k=0}^{n-1} x^k = (1 - x^n)/(1 - x)$ für $x \neq 1$.

5. In VK1 wurde die Divisionsmethode zur Konversion einer Dezimal- in eine Dualzahl erklärt. Beweisen Sie, dass das Verfahren korrekt arbeitet.

Aufgabensammlung

1. Überlegungen zum Formelaufbau auf Folie Seite 11.
2. Schreiben von Formeln mit wenigen Operationen auf Folie Seite 12.
3. Aufstellen von Wahrheitstafeln auf Folie Seite 16.
4. Formelauswertung bei gegebener Belegung auf Folie Seite 17.
5. Überprüfung der Korrektheit von Programmen auf Folie Seite 20.
6. Zeigen Sie formal die “andere Richtung” auf Folie Seite 25.
7. Vervollständigen Sie den “Quadratzahlbeweis” von Folie Seite 27.
8. Einige Übungen zur vollständigen Induktion finden Sie auf Folie Seite 38.