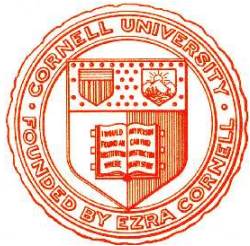


Theoretische Informatik I

Einheit 1

Mathematische Beweisführung



1. Formale und informale Beweise
 2. Verkettung einfacher Argumente
 3. Widerlegungsbeweise
 4. Induktion und Datenstrukturen
 5. Grundsätzliche Methodik
- Anhang: Wichtige Grundbegriffe

BEWEISFÜHRUNG IN DER INFORMATIK – WARUM?

- **Testen von Programmen ist unzureichend**
 - Nur hilfreich zur Entdeckung grober Fehler
 - Viele **kleine**, aber **gravierende** Fehler fallen durch das Testraster
z.B. Pentium Bug (Zeile vergessen), Ariane 5 (Arithmetik-Overflow),
Mars Polar Lander (Einheitenfehler), Heatbleed (Zeile vergessen), ...
 - Selbst “Hochqualitätssoftware” hat 2-3 Fehler pro 100 Zeilen Code
 - Nur Amateure glauben noch an Fehlerlosigkeit getesteter Programme
- **Programme müssen “bewiesen” werden**
 - **Erfolgreicher Beweis** zeigt genau, wie das Programm arbeitet
 - **Erfolgloser Beweisversuch** deutet auf mögliche Fehler im Programm
 - In sicherheitskritischen Anwendungen werden seit einigen Jahren
Programme ohne Korrektheitsbeweis nicht akzeptiert
 - Gute Programme entstehen aus dem Beweis einer Lösungsidee

Entwickler müssen die Korrektheit von Programmen beweisen können

Aber was ist eigentlich ein Beweis?

- **Eine Begründung, warum eine Behauptung gelten soll**

- Intuitiv einsichtige, logisch schlüssige und lückenlose Erklärung
- Braucht klare Formulierung der Behauptung und der Annahmen

- *Alle Primzahlen sind ungerade*

- Unausgesprochene Annahme: es geht um Primzahlen größer als 2
- Begründung: *jede gerade Zahl ist durch 2 teilbar*
jede Primzahl ist nur durch 1 und sich selbst teilbar
jede Primzahl größer als 2 ist nicht durch 2 teilbar

- **Gute Beweise sind genau und übersichtlich**

- Klar erkennbarer roter Faden in der Argumentation
- Knapp genug, um verständlich zu sein
- Genau genug, um fehlende Details rekonstruieren zu können

Keine Gedankensprünge, außer wenn sie leicht nachvollziehbar sind

MANCHMAL KANN MAN VERBLÜFFENDES BEWEISEN

- *Es gibt genauso viele gerade Zahlen wie natürliche Zahlen*
 - Kann das stimmen? Die geraden Zahlen sind doch nur die Hälfte?
 - Was heißt das überhaupt “gleich viele” bei unendlichen Mengen?
 - Die Anzahl der Elemente ist kein guter Vergleichsmaßstab
deswegen verwendet man Bijektionen zwischen Mengen als Kriterium
 A und B sind gleichmächtig, wenn es eine Bijektion $f:A\rightarrow B$ gibt
 - Wie könnte eine Bijektion von \mathbb{N} nach \mathbb{N}_2 (gerade Zahlen) aussehen?
Definiere $f:\mathbb{N}\rightarrow\mathbb{N}_2$ durch $f(n) = 2n$, dann ist f injektiv und surjektiv.

MANCHMAL KANN MAN VERBLÜFFENDES BEWEISEN

- *Es gibt genauso viele gerade Zahlen wie natürliche Zahlen*
 - Kann das stimmen? Die geraden Zahlen sind doch nur die Hälfte?
 - Was heißt das überhaupt “gleich viele” bei unendlichen Mengen?
 - Die Anzahl der Elemente ist kein guter Vergleichsmaßstab
deswegen verwendet man Bijektionen zwischen Mengen als Kriterium
 A und B sind gleichmächtig, wenn es eine Bijektion $f:A\rightarrow B$ gibt
 - Wie könnte eine Bijektion von \mathbb{N} nach \mathbb{N}_2 (gerade Zahlen) aussehen?
Definiere $f:\mathbb{N}\rightarrow\mathbb{N}_2$ durch $f(n) = 2n$, dann ist f injektiv und surjektiv.
- *\mathbb{N} , \mathbb{Z} , $\mathbb{N}\times\mathbb{N}$ und \mathbb{Q} sind gleichmächtig*
 - Bijektion von $\mathbb{N}\rightarrow\mathbb{Z}$ beschreibbar als Aufzählung $0,1,-1,2,-2,3,\dots$
 - $f:\mathbb{N}\times\mathbb{N}\rightarrow\mathbb{N}$ zählt Zahlenpaare diagonal durch $f(x,y) = (x+y)(x+y+1)\div 2 + y$
 - Rationale Zahlen sind wie Paare von ganzen und natürlichen Zahlen

MANCHMAL KANN MAN VERBLÜFFENDES BEWEISEN

- *Es gibt genauso viele gerade Zahlen wie natürliche Zahlen*
 - Kann das stimmen? Die geraden Zahlen sind doch nur die Hälfte?
 - Was heißt das überhaupt “gleich viele” bei unendlichen Mengen?
 - Die Anzahl der Elemente ist kein guter Vergleichsmaßstab
deswegen verwendet man Bijektionen zwischen Mengen als Kriterium
 A und B sind gleichmächtig, wenn es eine Bijektion $f:A\rightarrow B$ gibt
 - Wie könnte eine Bijektion von \mathbb{N} nach \mathbb{N}_2 (gerade Zahlen) aussehen?
Definiere $f:\mathbb{N}\rightarrow\mathbb{N}_2$ durch $f(n) = 2n$, dann ist f injektiv und surjektiv.
- *\mathbb{N} , \mathbb{Z} , $\mathbb{N}\times\mathbb{N}$ und \mathbb{Q} sind gleichmächtig*
 - Bijektion von $\mathbb{N}\rightarrow\mathbb{Z}$ beschreibbar als Aufzählung $0,1,-1,2,-2,3,\dots$
 - $f:\mathbb{N}\times\mathbb{N}\rightarrow\mathbb{N}$ zählt Zahlenpaare diagonal durch $f(x,y) = (x+y)(x+y+1)\div 2 + y$
 - Rationale Zahlen sind wie Paare von ganzen und natürlichen Zahlen
- **Wie sieht es aus mit reellen Zahlen?**
 - *\mathbb{R} ist genauso mächtig wie das offene Intervall $(0,1)$*
Definiere $f:(0,1)\rightarrow\mathbb{R}$ durch $f(x) = 1/(1-x) - 1/x$.
 - Aber $(0,1)$ und \mathbb{R} sind mächtiger als \mathbb{N} , \mathbb{Z} , $\mathbb{N}\times\mathbb{N}$ und \mathbb{Q} .

(Folie 22)

Es gibt zwei irrationale Zahlen x und y , deren Potenz rational ist

- **Standard-Beweis aus Mathematik-Lehrbuch**

- Wir wissen, daß $\sqrt{2}$ irrational ist. Betrachte $\sqrt{2}^{\sqrt{2}}$. Diese Zahl ist rational oder irrational. Ist $\sqrt{2}^{\sqrt{2}}$ rational, dann wähle $x=y=\sqrt{2}$. Sonst wähle $x=\sqrt{2}^{\sqrt{2}}$ und $y=\sqrt{2}$. Dann ist $x^y = \sqrt{2}^{(\sqrt{2}*\sqrt{2})} = 2$.
- Beweis ist korrekt, aber für viele etwas verwirrend. Wir wissen am Ende des Beweises nicht, welchen Wert x und y haben

MAN KANN DASSELBE SEHR UNTERSCHIEDLICH BEWEISEN

Es gibt zwei irrationale Zahlen x und y , deren Potenz rational ist

- **Standard-Beweis aus Mathematik-Lehrbuch**

- Wir wissen, daß $\sqrt{2}$ irrational ist. Betrachte $\sqrt{2}^{\sqrt{2}}$. Diese Zahl ist rational oder irrational. Ist $\sqrt{2}^{\sqrt{2}}$ rational, dann wähle $x=y=\sqrt{2}$. Sonst wähle $x=\sqrt{2}^{\sqrt{2}}$ und $y=\sqrt{2}$. Dann ist $x^y = \sqrt{2}^{(\sqrt{2}*\sqrt{2})} = 2$.
- Beweis ist korrekt, aber für viele etwas verwirrend. Wir wissen am Ende des Beweises nicht, welchen Wert x und y haben

- **Informatik-Beweise sollten “konstruktiv” sein, wo möglich**

- Wähle $x = \sqrt{2}$ und $y = 2 * \log_2 3$. Wir wissen daß beide Zahlen irrational sind und es gilt $x^y = 2^{1/2 * 2 * \log_2 3} = 3$.

MAN KANN DASSELBE SEHR UNTERSCHIEDLICH BEWEISEN

Es gibt zwei irrationale Zahlen x und y , deren Potenz rational ist

- **Standard-Beweis aus Mathematik-Lehrbuch**

- Wir wissen, daß $\sqrt{2}$ irrational ist. Betrachte $\sqrt{2}^{\sqrt{2}}$. Diese Zahl ist rational oder irrational. Ist $\sqrt{2}^{\sqrt{2}}$ rational, dann wähle $x=y=\sqrt{2}$. Sonst wähle $x=\sqrt{2}^{\sqrt{2}}$ und $y=\sqrt{2}$. Dann ist $x^y = \sqrt{2}^{(\sqrt{2}*\sqrt{2})} = 2$.
- Beweis ist korrekt, aber für viele etwas verwirrend. Wir wissen am Ende des Beweises nicht, welchen Wert x und y haben

- **Informatik-Beweise sollten “konstruktiv” sein, wo möglich**

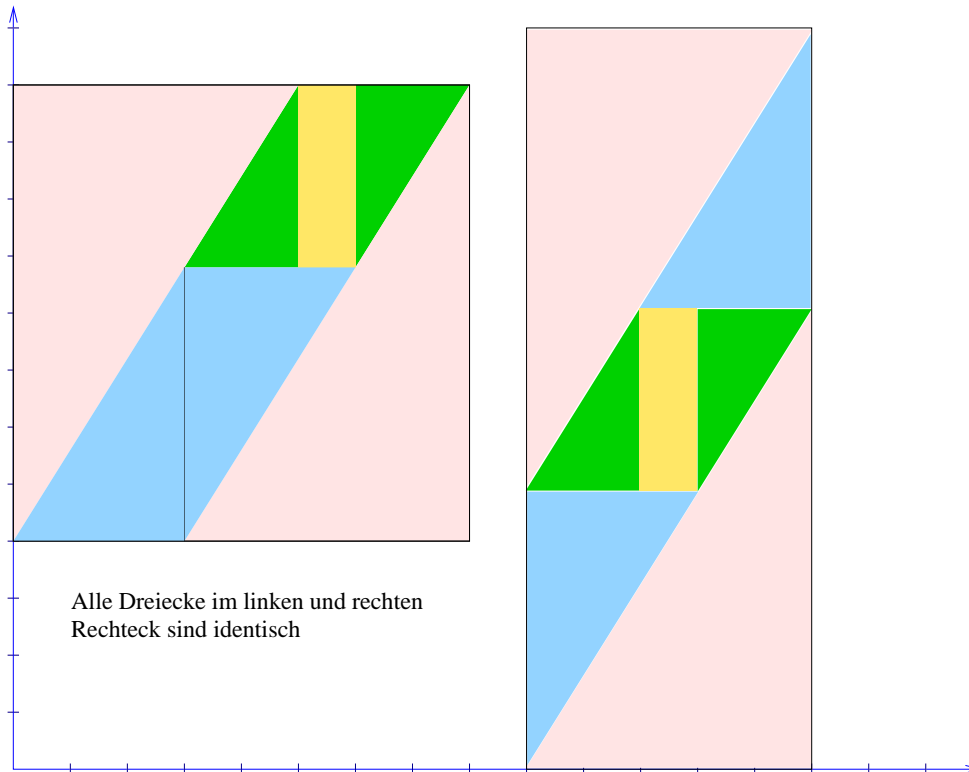
- Wähle $x = \sqrt{2}$ und $y = 2 * \log_2 3$. Wir wissen daß beide Zahlen irrational sind und es gilt $x^y = 2^{1/2 * 2 * \log_2 3} = 3$.

- **Es bleibt eine Frage offen**

- *Wie können wir sicher sein, daß $\sqrt{2}$ und $\log_2 3$ irrational sind?* (Folie 21)

BEWEISE KÖNNEN AUCH TRUGSCHLÜSSE ENTHALTEN

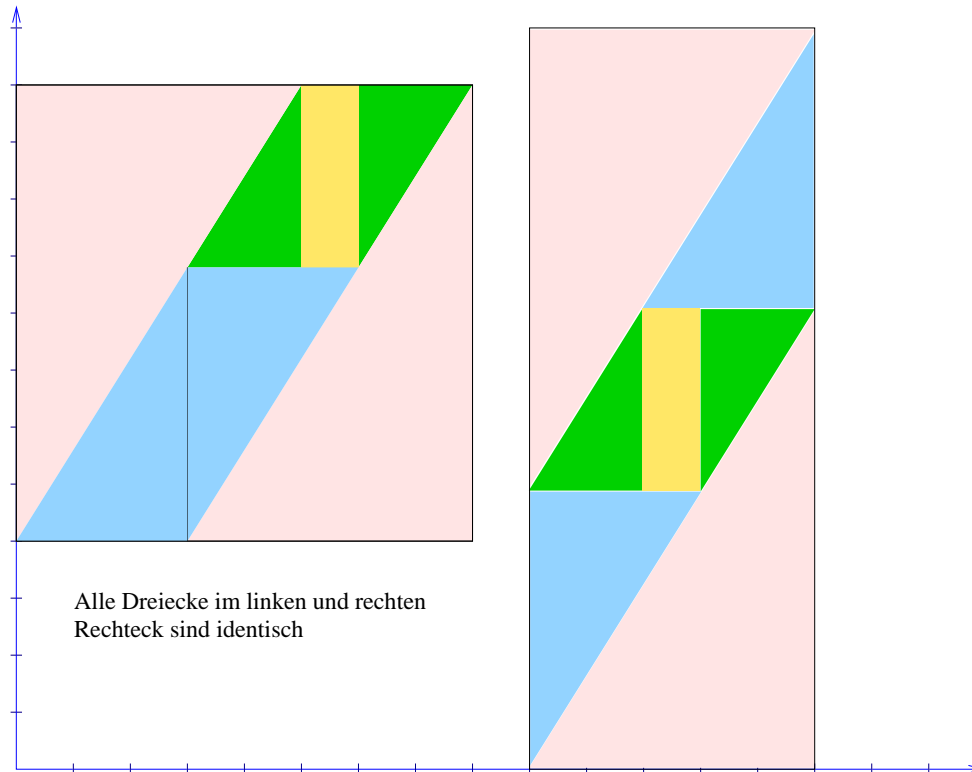
- **Geometrischer Beweis für Gleichheit von Flächen**



– Beweis wirkt einleuchtend, aber das Ergebnis scheint unsinnig

BEWEISE KÖNNEN AUCH TRUGSCHLÜSSE ENTHALTEN

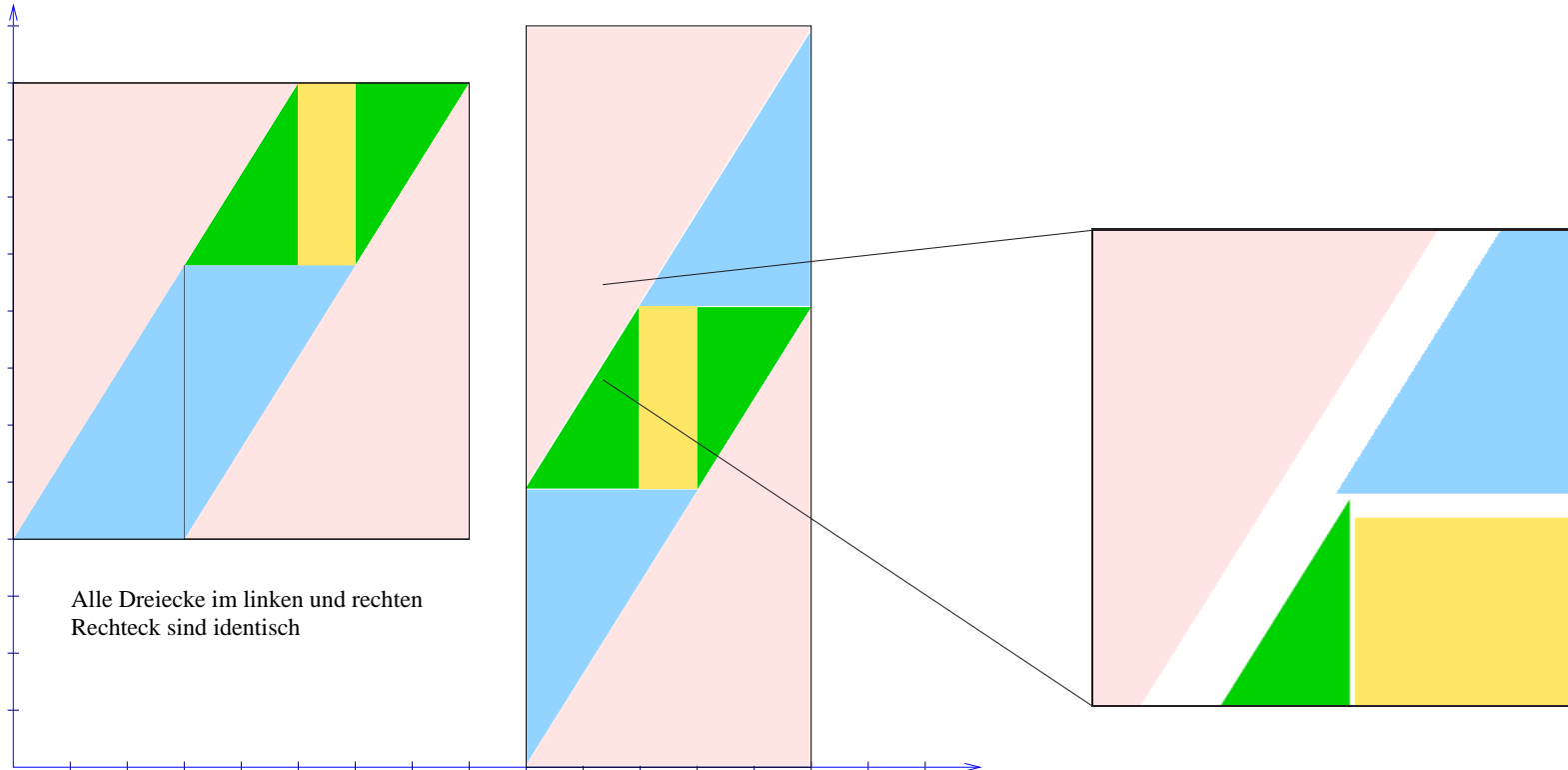
● Geometrischer Beweis für Gleichheit von Flächen



- Beweis wirkt einleuchtend, aber das Ergebnis scheint unsinnig $64=65??$
- Wo ist der Fehler im Beweis?

BEWEISE KÖNNEN AUCH TRUGSCHLÜSSE ENTHALTEN

● Geometrischer Beweis für Gleichheit von Flächen



– Beweis wirkt einleuchtend, aber das Ergebnis scheint unsinnig $64=65??$

– Wo ist der Fehler im Beweis?

● Anschauung alleine ist kein Beweis

– Es muß auch gezeigt werden, daß die Dreiecke lückenlos gleich sind

– Für “Kongruenzbeweise” gibt es klare Kriterien

- **Umgangssprache ist oft mißverständlich**

- Fachspezifisches Vokabular und symbolische Notationen sind hilfreich
- In der Informatik muß man etwas formaler sein, da Computer nur formale Objekte (Programme, Schaltungen) verarbeiten können
- In sicherheitskritischen Anwendungen werden nur noch Beweise akzeptiert, die man auch mit Computern überprüfen kann

- *Quadrieren ist kleiner als Potenzieren*

- Gemeint ist: Für jede natürliche Zahl x gilt $x^2 \leq 2^x$
- Kürzere mathematische Schreibweise: $\forall x \in \mathbb{N}. x^2 \leq 2^x$
- Die Behauptung läßt sich prüfen für $x=0, 1, 2, 4, 5, 6, \dots$
Aber sie gilt **nicht** für $x=3$, denn $3^2 = 9 > 8 = 2^3$
- Korrekte Formulierung der Behauptung $\forall x \in \mathbb{N}. x \geq 4 \Rightarrow x^2 \leq 2^x$
- Ein (rein) umgangssprachlicher Beweis hierfür ist schwer formulierbar
Beweis benötigt mathematische Notation und Induktion (siehe Folie 25)

FORMALE BEWEISE SIND NICHT IMMER VERSTÄNDLICHER

- **Formale Notation ist nur eine Notationshilfe**

- Sie soll helfen, eine gute Idee klar zu formulieren
- Formale Notation alleine macht einen Beweis noch nicht gut

- *Wenn drei ganze Zahlen sich jeweils um maximal 1 unterscheiden, dann sind zwei von ihnen gleich*

- Informaler Beweis zeigt den wesentlichen Grund auf:

Wenn alle drei Zahlen unterschiedlich wären, dann gäbe es eine kleinste, eine mittlere und eine größte und alle drei Zahlen unterschieden sich jeweils um mindestens 1. Damit wäre der Unterschied zwischen der kleinsten und der größten Zahl mindestens 2.

FORMALE BEWEISE SIND NICHT IMMER VERSTÄNDLICHER

- **Formale Notation ist nur eine Notationshilfe**

- Sie soll helfen, eine gute Idee klar zu formulieren
- Formale Notation alleine macht einen Beweis noch nicht gut

- *Wenn drei ganze Zahlen sich jeweils um maximal 1 unterscheiden, dann sind zwei von ihnen gleich*

- Informaler Beweis zeigt den wesentlichen Grund auf:

Wenn alle drei Zahlen unterschiedlich wären, dann gäbe es eine kleinste, eine mittlere und eine größte und alle drei Zahlen unterschieden sich jeweils um mindestens 1. Damit wäre der Unterschied zwischen der kleinsten und der größten Zahl mindestens 2.

- Formal: $\forall x, y, z \in \mathbb{N}. (|x-y| \leq 1 \wedge |x-z| \leq 1 \wedge |z-y| \leq 1) \Rightarrow x=y \vee x=z \vee z=y$

*Sei $x, y, z \in \mathbb{N}$ beliebig. Es gelte $|x-y| \leq 1 \wedge |x-z| \leq 1 \wedge |z-y| \leq 1$ und o.B.d.A. $x \leq y \leq z$. Wenn $x=y \vee x=z \vee z=y$ nicht gilt, dann müsste $|x-y|=1$, $|x-z|=1$ und $|z-y|=1$ gelten. Es folgt $z=y+1=x+2$, also $|x-z|=2$. Dies widerspricht aber der Annahme $|x-z| \leq 1$.
Deswegen muß $x=y \vee x=z \vee z=y$ gelten.*

Beweis zeigt mehr Details, ist aber nicht unbedingt klarer

TROTZDEM MÜSSEN WIR FORMALE BEWEISE EINÜBEN

- **Sie sind das Handwerkzeug, daß jeder kennen muß**
 - Wenn man nicht sicher ist, ob ein Argument wirklich korrekt ist, zeigen formale Beweise die nötigen Details
 - Wenn einen die Intuition im Stich läßt, kann man alles genau aufschreiben und “durchrechnen”
 - Wenn man Aussagen ganz genau formalisiert, kann man die Beweise sogar mit Hilfe von Computern führen (Theorembeweiser)
- **In der Informatik dreht sich alles um Formales**
 - Programme und Hardware sind formale Objekte mit festen Regeln
 - Umgangsprachliche Beschreibungen des Verhaltens von Computern sind oft umständlicher als präzise mathematische Beschreibungen
 - Formale Beweise von Programmeigenschaften sind leichter als informale
 - Computer können die Korrektheit von sicherheitskritischer Software nur dann überprüfen, wenn Beweise vollständig formal geführt wurden
- **Hier zunächst nur verhältnismäßig einfache Probleme**
 - Beweisführung läßt sich leichter trainieren, wenn die Materie vertraut ist und die Menge der Details überschaubar bleibt

BEWEIS DURCH VERKETTUNG EINFACHER ARGUMENTE

Ziel: Zeige, daß eine Behauptung B aus Annahmen A folgt

- **Methodik: Kette von Aussagen $A_1, \dots, A_n = B$**
 - Zwischenaussagen A_i müssen schlüssig aus dem Vorhergehenden folgen
 - Dabei dürfen nur Annahmen aus A , Definitionen, bewiesene Aussagen, mathematische Gesetze, logische Schlußfolgerungen verwendet werden
 - Einfachste Form der Beweisführung, auch **Deduktiver Beweis** genannt

BEWEIS DURCH VERKETTUNG EINFACHER ARGUMENTE

Ziel: Zeige, daß eine Behauptung B aus Annahmen A folgt

- **Methodik: Kette von Aussagen $A_1, \dots, A_n = B$**

- Zwischenaussagen A_i müssen schlüssig aus dem Vorhergehenden folgen
- Dabei dürfen nur Annahmen aus A , Definitionen, bewiesene Aussagen, mathematische Gesetze, logische Schlußfolgerungen verwendet werden
- Einfachste Form der Beweisführung, auch **Deduktiver Beweis** genannt

- **Beispiel: Die Summe zweier ungerader Zahlen ist gerade**

- **Informaler Beweis:** *Es seien a und b zwei ungerade Zahlen.*

Per Definition ist $a = 2x + 1$ und $b = 2y + 1$ für gewisse x und y und die Summe ist $2(x + y + 1)$, also eine gerade Zahl.

- **Schematische Darstellung** ist oft kürzer und präziser

Aussage	Begründung
1. $a = 2x + 1$	Gegeben (Auflösung des Begriffs “ungerade”)
2. $b = 2y + 1$	Gegeben (Auflösung des Begriffs “ungerade”)
3. $a + b = 2(x + y + 1)$	(1,2) und Gesetze der Arithmetik

Die schematische Form ist eine Vorstufe maschinenprüfbarer Beweise, aber nicht “besser”

OFT REICHT ES, DEFINITIONEN AUFZULÖSEN

- **De Morgan Regel für Mengen:** $A \cap B = \overline{\overline{A} \cup \overline{B}}$

– Beweis: Sei x beliebig. Dann gilt

$$x \in \overline{\overline{A} \cup \overline{B}} \Leftrightarrow x \notin \overline{A} \cup \overline{B} \Leftrightarrow x \notin \overline{A} \wedge x \notin \overline{B} \Leftrightarrow x \in A \wedge x \in B \Leftrightarrow x \in A \cap B$$

– Verwendete Definitionen:

$$A=B \Leftrightarrow \forall x. x \in A \Leftrightarrow x \in B \quad (\text{Extensionalitätsregel für Mengen})$$

(Zwei Mengen sind gleich, wenn sie die gleichen Elemente haben)

$$x \in A \cap B \Leftrightarrow x \in A \wedge x \in B \quad (\text{Durchschnitt von Mengen})$$

$$x \in A \cup B \Leftrightarrow x \in A \vee x \in B \quad (\text{Vereinigung von Mengen})$$

$$x \in \overline{A} \Leftrightarrow x \notin A \quad (\text{Komplement von Mengen})$$

OFT REICHT ES, DEFINITIONEN AUFZULÖSEN

● De Morgan Regel für Mengen: $A \cap B = \overline{\overline{A} \cup \overline{B}}$

– Beweis: Sei x beliebig. Dann gilt

$$x \in \overline{\overline{A} \cup \overline{B}} \Leftrightarrow x \notin \overline{A} \cup \overline{B} \Leftrightarrow x \notin \overline{A} \wedge x \notin \overline{B} \Leftrightarrow x \in A \wedge x \in B \Leftrightarrow x \in A \cap B$$

– Verwendete Definitionen:

$$A=B \Leftrightarrow \forall x. x \in A \Leftrightarrow x \in B \quad (\text{Extensionalitätsregel für Mengen})$$

(Zwei Mengen sind gleich, wenn sie die gleichen Elemente haben)

$$x \in A \cap B \Leftrightarrow x \in A \wedge x \in B \quad (\text{Durchschnitt von Mengen})$$

$$x \in A \cup B \Leftrightarrow x \in A \vee x \in B \quad (\text{Vereinigung von Mengen})$$

$$x \in \overline{A} \Leftrightarrow x \notin A \quad (\text{Komplement von Mengen})$$

● Anwendung der Umkehrfunktion

Ist $f:A \rightarrow B$ injektiv, dann ist $f^{-1}(f(x))=x$ für alle $x \in A$

– Die Umkehrfunktion f^{-1} ist definiert durch $f^{-1}(y)=z \Leftrightarrow f(z)=y$.

Für $y=f(x)$ gilt also $f^{-1}(f(x))=z \Leftrightarrow f(z)=f(x)$.

Da f injektiv ist, gilt $f(z)=f(x) \Leftrightarrow x=z$ und damit ist $f^{-1}(f(x))=x$

Gilt auch $f(f^{-1}(y))=y$ für alle $y \in B$?

OFT REICHT ES, DEFINITIONEN AUFZULÖSEN

• De Morgan Regel für Mengen: $A \cap B = \overline{\overline{A} \cup \overline{B}}$

– Beweis: Sei x beliebig. Dann gilt

$$x \in \overline{\overline{A} \cup \overline{B}} \Leftrightarrow x \notin \overline{A} \cup \overline{B} \Leftrightarrow x \notin \overline{A} \wedge x \notin \overline{B} \Leftrightarrow x \in A \wedge x \in B \Leftrightarrow x \in A \cap B$$

– Verwendete Definitionen:

$$A=B \Leftrightarrow \forall x. x \in A \Leftrightarrow x \in B \quad (\text{Extensionalitätsregel für Mengen})$$

(Zwei Mengen sind gleich, wenn sie die gleichen Elemente haben)

$$x \in A \cap B \Leftrightarrow x \in A \wedge x \in B \quad (\text{Durchschnitt von Mengen})$$

$$x \in A \cup B \Leftrightarrow x \in A \vee x \in B \quad (\text{Vereinigung von Mengen})$$

$$x \in \overline{A} \Leftrightarrow x \notin A \quad (\text{Komplement von Mengen})$$

• Anwendung der Umkehrfunktion

Ist $f:A \rightarrow B$ injektiv, dann ist $f^{-1}(f(x))=x$ für alle $x \in A$

– Die Umkehrfunktion f^{-1} ist definiert durch $f^{-1}(y)=z \Leftrightarrow f(z)=y$.

Für $y=f(x)$ gilt also $f^{-1}(f(x))=z \Leftrightarrow f(z)=f(x)$.

Da f injektiv ist, gilt $f(z)=f(x) \Leftrightarrow x=z$ und damit ist $f^{-1}(f(x))=x$

Gilt auch $f(f^{-1}(y))=y$ für alle $y \in B$?

Nein! f^{-1} ist nicht für alle $y \in B$ definiert, wenn f nicht auch surjektiv ist

VIELE PRÄZISIERUNGSGRAD E KÖNNEN SINNVOLL SEIN

Wenn S endliche Teilmenge einer Menge U ist und das Komplement von S bezüglich U endlich ist, dann ist U endlich

1. Informales Argument beschreibt Kernidee des Beweises

“ U ist die Vereinigung zweier endlicher Mengen, also endlich”

- Leser versteht den Grundgedanken, warum die Behauptung gilt
- Oft die verständlichste Form, aber anfällig für Trugschlüsse

VIELE PRÄZISIERUNGSGRAD E KÖNNEN SINNVOLL SEIN

Wenn S endliche Teilmenge einer Menge U ist und das Komplement von S bezüglich U endlich ist, dann ist U endlich

1. Informales Argument beschreibt Kernidee des Beweises

“ U ist die Vereinigung zweier endlicher Mengen, also endlich”

- Leser versteht den Grundgedanken, warum die Behauptung gilt
- Oft die verständlichste Form, aber anfällig für Trugschlüsse

2. Präzises Argument, teilweise mit formaler Notation

“Es sei S endlich, T das Komplement von S bezüglich U und T endlich. Dann ist $U = T \cup S$ und $T \cap S = \emptyset$. Somit ist die Kardinalität $|U|$ von U die Summe der Kardinalitäten von S und T . Da $|S|$ und $|T|$ natürliche Zahlen sind, gilt dies auch für $|U|$ und somit ist U endlich.”

- Kompromiss zwischen Genauigkeit und Verständlichkeit
- Formulierung von Annahmen, Zielaussage und Zwischenaussagen
- Klare Begründung, warum eine Aussage aus dem Vorhergehenden folgt

Weitere Stufen auf Folgefolien

PRÄZISIERUNGSGRADE FÜR BEWEISE II

Wenn S endliche Teilmenge einer Menge U ist und das Komplement von S bezüglich U endlich ist, dann ist U endlich

3. Schematische Darstellung der Argumentation

Definition: S endlich \equiv Es gibt eine ganze Zahl n mit $|S| = n$
 T Komplement von S $\equiv T \cup S = U$ und $T \cap S = \emptyset$

Aussage	Begründung
1. S endlich	Gegeben
2. T Komplement von S	Gegeben
3. T endlich	Gegeben
4. $ S = n$ für ein $n \in \mathbb{N}$	Auflösen der Definition in (1)
5. $ T = m$ für ein $m \in \mathbb{N}$	Auflösen der Definition in (3)
6. $T \cup S = U$	Auflösen der Definition in (2)
7. $T \cap S = \emptyset$	Auflösen der Definition in (2)
8. $ U = m + n$ für $n, m \in \mathbb{N}$	(4),(5),(6), (7) und Gesetze der Kardinalität
9. U endlich	Einsetzen der Definition in (8)

- Höhere Genauigkeit hat Vorrang vor Verständlichkeit
- Kleinteiligere Beweisführung mit größerer Anzahl von Einzelschritten
- Formulierung der Begründung ist sehr genau

PRÄZISIERUNGSGRADE FÜR BEWEISE III

Ist S endliche Teilmenge einer Menge U und das Komplement von S bezüglich U endlich, dann ist U endlich

4. Vollständig formale Beweisführung

Definition: X endlich $\equiv \exists n:\mathbb{N}. |X|=n$ (*endlich*)
 $Y = X^c$ bzgl. Z $\equiv Y \cup X = Z \wedge Y \cap X = \emptyset$ (*Komplement*)
 Lemma: $\forall X, Y, Z:\text{Set}. (Y \cup X = Z \wedge Y \cap X = \emptyset) \Rightarrow |Z| = |X| + |Y|$ (*Kardinalität*)

Aussage	Begründung
1. S endlich	Annahme
2. $T = S^c$ bzgl. U	Annahme
3. T endlich	Annahme
4. $T \cup S = U \wedge T \cap S = \emptyset$	unfold <i>Komplement</i> (2)
5. $(T \cup S = U \wedge T \cap S = \emptyset) \Rightarrow U = S + T $	Lemma <i>Kardinalität</i> für S, T, U
6. $ U = S + T $	modus ponens (4) (5)
7. $\exists n:\mathbb{N}. S =n$	unfold <i>endlich</i> (1)
8. $n:\mathbb{N}$	\exists -elim (7)
9. $ S =n$	\exists -elim zweiter Teil
10. $m:\mathbb{N}$	unfold <i>endlich</i> (3), dann \exists -elim
11. $ T =m$	\exists -elim zweiter Teil
12. $ U = n+m$	subst (9,11) in (6)
13. $\exists n:\mathbb{N}. U =n$	\exists -intro (12) für $n+m$
14. U endlich	fold <i>endlich</i> (13)

- Alle Aussagen bestehen ausschließlich aus formaler Notation
- Begründungen sind Anwendungen schematischer Beweisregeln
- Auch auf dem Computer durchführbar

VOLLSTÄNDIG FORMALE BEWEISFÜHRUNG

- **Regeln erzeugen logische Konsequenzen von Formeln**

- z.B. **Modus Ponens**: wenn Aussage A gilt und Aussage B aus Aussage A folgt, dann gilt auch Aussage B

- Kurz geschrieben als $\frac{A \quad A \Rightarrow B}{B}$ **modus ponens**

- A und B sind Platzhalter für die tatsächlichen Aussagen

- Instantiierte Form im Beispiel

$$\frac{TUS=U \wedge T \cap S=\emptyset \quad (TUS=U \wedge T \cap S=\emptyset) \Rightarrow |U|=|S|+|T|}{|U|=|S|+|T|}$$

- **Es gibt Regeln für alle logischen Junktoren**

- Zerlegen (“**elim**”) und Zusammensetzen (“**intro**”) von $\forall, \exists, \wedge, \vee, \Rightarrow, \neg$

- Modus Ponens ist ein anderer (historisch älterer) Name für \Rightarrow -elim

- **Regeln für Verwendung von Definitionen und Lemmata**

- Eine Definition $lhs \equiv rhs$ (*name*) kann auf- und zurückgefaltet werden

- Kurz geschrieben als $\frac{lhs}{rhs}$ **unfold** bzw. $\frac{rhs}{lhs}$ **fold**

- Die **Lemma** Regel fügt das genannte Lemma ein und wendet dann für jeden genannten Wert die Regel \forall -elim an

REGELN FÜR AUSSAGENLOGISCHE OPERATOREN

Je zwei Arten von Regeln für jedes logisches Symbol

- **Einführungsregel:** Welche Voraussetzungen machen eine Formel gültig?
- **Eliminationsregel:** Was folgt aus einer gegebenen Formel?

$\frac{A \quad B}{A \wedge B} \wedge\text{-intro}$	$\frac{A \wedge B}{A} \wedge\text{-elim1}$	$\frac{A \wedge B}{B} \wedge\text{-elim2}$
$\frac{A}{A \vee B} \vee\text{-intro1}$	$\frac{B}{A \vee B} \vee\text{-intro2}$	$\frac{A \vee B \quad \begin{array}{c} [A] \\ C \end{array} \quad \begin{array}{c} [B] \\ C \end{array}}{C} \vee\text{-elim}$
$\frac{\begin{array}{c} [A] \\ B \end{array}}{A \Rightarrow B} \Rightarrow\text{-intro}$	$\frac{A \quad A \Rightarrow B}{B} \Rightarrow\text{-elim}$	
$\frac{\begin{array}{c} [A] \\ \text{ff} \end{array}}{\neg A} \neg\text{-intro}$	$\frac{\neg A \quad A}{\text{ff}} \neg\text{-elim}$	
$\overline{A \vee \neg A} \text{excl-m}$	$\frac{\text{ff}}{A} \text{ff-elim}$	

$\Rightarrow\text{-intro}$: Wenn B aus der Annahme A folgt, dann gilt $A \Rightarrow B$

$\neg\text{-intro}$: $\neg A$ gilt, wenn aus der Annahme A ein Widerspruch (ff) folgt

$\vee\text{-elim}$: Wenn $A \vee B$ gilt und Behauptung C aus der Annahme A folgt und C auch aus der Annahme B folgt, dann gilt C (Fallunterscheidung)

EIN REIN LOGISCHER FORMALER BEWEIS

Zeige $(x \geq 5 \wedge x \neq 5) \Rightarrow x > 5$

Definition: $x \geq y \equiv x > y \vee x = y$ (*geq*)

Definition: $x \neq y \equiv \neg(x = y)$ (*neq*)

Aussage	Begründung
1. $x \geq 5 \wedge x \neq 5$	<i>Annahme</i> entfällt in Schritt 12
2. $x \geq 5$	\wedge -elim1 (1)
3. $x \neq 5$	\wedge -elim2 (1)
4. $x > 5 \vee x = 5$	unfold <i>leq</i> (2)
5. $\neg(x = 5)$	unfold <i>neq</i> (3)
6. $x > 5$	<i>Annahme</i> entfällt in Schritt 11
7. $x > 5$	folgt identisch aus (6)
8. $x = 5$	<i>Annahme</i> entfällt in Schritt 11
9. ff	\neg -elim (5) (8)
10. $x > 5$	ff-elim (9)
11. $x > 5$	\vee -elim (4) ([6] 7) ([8] 10)
12. $(x \geq 5 \wedge x \neq 5) \Rightarrow x > 5$	\Rightarrow -intro ([1] 11)

- Annahme (1) mußte gemacht werden um die Implikation zu zeigen
- Annahmen (6) und (8) wurden für die Fallanalyse gebraucht
- Alle drei Annahmen sind während des Beweises “weggefallen”
- Die Aussage $(x \geq 5 \wedge x \neq 5) \Rightarrow x > 5$ gilt ohne jede weitere Annahme

FORMALE BEWEISFÜHRUNG – BEWEISE FINDEN

- **Im wesentlichen Rückwärtsanwendung von Regeln**
 - Zerlege Behauptungen mit *intro*-Regeln, Annahmen können entstehen
 - Zerlege Annahmen mit *elim*-Regeln
 - Fülle Lücke zwischen Annahmen und (Zwischen-)Behauptungen
 - Entstandener Beweis wird dann vorwärts gelesen
- **Entwickle Beweis für $(x \geq 5 \wedge x \neq 5) \Rightarrow x > 5$**

Definition: $x \geq y \equiv x > y \vee x = y$ (*geq*)

Definition: $x \neq y \equiv \neg(x = y)$ (*neq*)

Aussage	Begründung
1.	
2.	
3.	
4.	
5.	
6.	
7.	
8.	
9.	
10.	
11.	
12. $(x \geq 5 \wedge x \neq 5) \Rightarrow x > 5$	<i>Zielbehauptung – noch zu zeigen</i>

FORMALE BEWEISFÜHRUNG – BEWEISE FINDEN

- **Im wesentlichen Rückwärtsanwendung von Regeln**
 - Zerlege Behauptungen mit *intro*-Regeln, Annahmen können entstehen
 - Zerlege Annahmen mit *elim*-Regeln
 - Fülle Lücke zwischen Annahmen und (Zwischen-)Behauptungen
 - Entstandener Beweis wird dann vorwärts gelesen
- **Entwickle Beweis für $(x \geq 5 \wedge x \neq 5) \Rightarrow x > 5$**

Definition: $x \geq y \equiv x > y \vee x = y$ (*geq*)

Definition: $x \neq y \equiv \neg(x = y)$ (*neq*)

Aussage	Begründung
1. $x \geq 5 \wedge x \neq 5$	<i>Annahme</i> entfällt in Schritt 12
2.	
3.	
4.	
5.	
6.	
7.	
8.	
9.	
10.	
11. $x > 5$	<i>noch zu zeigen</i>
12. $(x \geq 5 \wedge x \neq 5) \Rightarrow x > 5$	\Rightarrow -intro ([1] 11)

- **Im wesentlichen Rückwärtsanwendung von Regeln**
 - Zerlege Behauptungen mit *intro*-Regeln, Annahmen können entstehen
 - Zerlege Annahmen mit *elim*-Regeln
 - Fülle Lücke zwischen Annahmen und (Zwischen-)Behauptungen
 - Entstandener Beweis wird dann vorwärts gelesen
- **Entwickle Beweis für $(x \geq 5 \wedge x \neq 5) \Rightarrow x > 5$**

Definition: $x \geq y \equiv x > y \vee x = y$ (*geq*)

Definition: $x \neq y \equiv \neg(x = y)$ (*neq*)

Aussage	Begründung
1. $x \geq 5 \wedge x \neq 5$	<i>Annahme</i> entfällt in Schritt 12
2. $x \geq 5$	\wedge -elim1 (1)
3. $x \neq 5$	\wedge -elim2 (1)
4.	
5.	
6.	
7.	
8.	
9.	
10.	
11. $x > 5$	<i>noch zu zeigen</i>
12. $(x \geq 5 \wedge x \neq 5) \Rightarrow x > 5$	\Rightarrow -intro ([1] 11)

FORMALE BEWEISFÜHRUNG – BEWEISE FINDEN

- **Im wesentlichen Rückwärtsanwendung von Regeln**
 - Zerlege Behauptungen mit *intro*-Regeln, Annahmen können entstehen
 - Zerlege Annahmen mit *elim*-Regeln
 - Fülle Lücke zwischen Annahmen und (Zwischen-)Behauptungen
 - Entstandener Beweis wird dann vorwärts gelesen
- **Entwickle Beweis für $(x \geq 5 \wedge x \neq 5) \Rightarrow x > 5$**

Definition: $x \geq y \equiv x > y \vee x = y$ (*geq*)

Definition: $x \neq y \equiv \neg(x = y)$ (*neq*)

Aussage	Begründung
1. $x \geq 5 \wedge x \neq 5$	<i>Annahme</i> entfällt in Schritt 12
2. $x \geq 5$	\wedge -elim1 (1)
3. $x \neq 5$	\wedge -elim2 (1)
4. $x > 5 \vee x = 5$	unfold <i>leq</i> (2)
5. $\neg(x = 5)$	unfold <i>neq</i> (3)
6.	
7.	
8.	
9.	
10.	
11. $x > 5$	<i>noch zu zeigen</i>
12. $(x \geq 5 \wedge x \neq 5) \Rightarrow x > 5$	\Rightarrow -intro ([1] 11)

- **Im wesentlichen Rückwärtsanwendung von Regeln**
 - Zerlege Behauptungen mit *intro*-Regeln, Annahmen können entstehen
 - Zerlege Annahmen mit *elim*-Regeln
 - Fülle Lücke zwischen Annahmen und (Zwischen-)Behauptungen
 - Entstandener Beweis wird dann vorwärts gelesen
- **Entwickle Beweis für $(x \geq 5 \wedge x \neq 5) \Rightarrow x > 5$**

Definition: $x \geq y \equiv x > y \vee x = y$ (*geq*)

Definition: $x \neq y \equiv \neg(x = y)$ (*neq*)

Aussage	Begründung
1. $x \geq 5 \wedge x \neq 5$	<i>Annahme</i> entfällt in Schritt 12
2. $x \geq 5$	\wedge -elim1 (1)
3. $x \neq 5$	\wedge -elim2 (1)
4. $x > 5 \vee x = 5$	unfold <i>leq</i> (2)
5. $\neg(x = 5)$	unfold <i>neq</i> (3)
6. $x > 5$	<i>Annahme</i> entfällt in Schritt 11
7. $x > 5$	<i>noch zu zeigen</i>
8. $x = 5$	<i>Annahme</i> entfällt in Schritt 11
9.	
10. $x > 5$	<i>noch zu zeigen</i>
11. $x > 5$	\vee -elim (4) ([6] 7) ([8] 10)
12. $(x \geq 5 \wedge x \neq 5) \Rightarrow x > 5$	\Rightarrow -intro ([1] 11)

- **Im wesentlichen Rückwärtsanwendung von Regeln**
 - Zerlege Behauptungen mit *intro*-Regeln, Annahmen können entstehen
 - Zerlege Annahmen mit *elim*-Regeln
 - Fülle Lücke zwischen Annahmen und (Zwischen-)Behauptungen
 - Entstandener Beweis wird dann vorwärts gelesen
- **Entwickle Beweis für $(x \geq 5 \wedge x \neq 5) \Rightarrow x > 5$**

Definition: $x \geq y \equiv x > y \vee x = y$ (*geq*)

Definition: $x \neq y \equiv \neg(x = y)$ (*neq*)

Aussage	Begründung
1. $x \geq 5 \wedge x \neq 5$	<i>Annahme</i> entfällt in Schritt 12
2. $x \geq 5$	\wedge -elim1 (1)
3. $x \neq 5$	\wedge -elim2 (1)
4. $x > 5 \vee x = 5$	unfold <i>leq</i> (2)
5. $\neg(x = 5)$	unfold <i>neq</i> (3)
6. $x > 5$	<i>Annahme</i> entfällt in Schritt 11
7. $x > 5$	folgt identisch aus (6)
8. $x = 5$	<i>Annahme</i> entfällt in Schritt 11
9.	
10. $x > 5$	<i>noch zu zeigen</i>
11. $x > 5$	\vee -elim (4) ([6] 7) ([8] 10)
12. $(x \geq 5 \wedge x \neq 5) \Rightarrow x > 5$	\Rightarrow -intro ([1] 11)

- **Im wesentlichen Rückwärtsanwendung von Regeln**

- Zerlege Behauptungen mit *intro*-Regeln, Annahmen können entstehen
- Zerlege Annahmen mit *elim*-Regeln
- Fülle Lücke zwischen Annahmen und (Zwischen-)Behauptungen
- Entstandener Beweis wird dann vorwärts gelesen

- **Entwickle Beweis für $(x \geq 5 \wedge x \neq 5) \Rightarrow x > 5$**

Definition: $x \geq y \equiv x > y \vee x = y$ (*geq*)

Definition: $x \neq y \equiv \neg(x = y)$ (*neq*)

Aussage	Begründung
1. $x \geq 5 \wedge x \neq 5$	<i>Annahme</i> entfällt in Schritt 12
2. $x \geq 5$	\wedge -elim1 (1)
3. $x \neq 5$	\wedge -elim2 (1)
4. $x > 5 \vee x = 5$	unfold <i>leq</i> (2)
5. $\neg(x = 5)$	unfold <i>neq</i> (3)
6. $x > 5$	<i>Annahme</i> entfällt in Schritt 11
7. $x > 5$	folgt identisch aus (6)
8. $x = 5$	<i>Annahme</i> entfällt in Schritt 11
9. ff	\neg -elim (5) (8)
10. $x > 5$	<i>noch zu zeigen</i>
11. $x > 5$	\vee -elim (4) ([6] 7) ([8] 10)
12. $(x \geq 5 \wedge x \neq 5) \Rightarrow x > 5$	\Rightarrow -intro ([1] 11)

FORMALE BEWEISFÜHRUNG – BEWEISE FINDEN

- **Im wesentlichen Rückwärtsanwendung von Regeln**
 - Zerlege Behauptungen mit *intro*-Regeln, Annahmen können entstehen
 - Zerlege Annahmen mit *elim*-Regeln
 - Fülle Lücke zwischen Annahmen und (Zwischen-)Behauptungen
 - Entstandener Beweis wird dann vorwärts gelesen
- **Entwickle Beweis für $(x \geq 5 \wedge x \neq 5) \Rightarrow x > 5$**

Definition: $x \geq y \equiv x > y \vee x = y$ (*geq*)

Definition: $x \neq y \equiv \neg(x = y)$ (*neq*)

Aussage	Begründung
1. $x \geq 5 \wedge x \neq 5$	<i>Annahme</i> entfällt in Schritt 12
2. $x \geq 5$	\wedge -elim1 (1)
3. $x \neq 5$	\wedge -elim2 (1)
4. $x > 5 \vee x = 5$	unfold <i>leq</i> (2)
5. $\neg(x = 5)$	unfold <i>neq</i> (3)
6. $x > 5$	<i>Annahme</i> entfällt in Schritt 11
7. $x > 5$	folgt identisch aus (6)
8. $x = 5$	<i>Annahme</i> entfällt in Schritt 11
9. ff	\neg -elim (5) (8)
10. $x > 5$	ff-elim (9)
11. $x > 5$	\vee -elim (4) ([6] 7) ([8] 10)
12. $(x \geq 5 \wedge x \neq 5) \Rightarrow x > 5$	\Rightarrow -intro ([1] 11)

Interaktive Theorembeweiser bieten maschinelle Unterstützung für diesen Prozeß

BEWEISFÜHRUNG DURCH “UMKEHRUNG”

- **Kontraposition**

- Anstatt zu zeigen, daß die Behauptung B aus den Annahmen A folgt, beweise, daß **nicht** A aus der Annahme **nicht** B folgt
- Aussagenlogisch ist $\neg B \Rightarrow \neg A$ äquivalent zu $A \Rightarrow B$

BEWEISFÜHRUNG DURCH “UMKEHRUNG”

- **Kontraposition**

- Anstatt zu zeigen, daß die Behauptung B aus den Annahmen A folgt, beweise, daß **nicht** A aus der Annahme **nicht** B folgt
- Aussagenlogisch ist $\neg B \Rightarrow \neg A$ äquivalent zu $A \Rightarrow B$

- **Häufigste Anwendung: Indirekte Beweisführung**

- Zeige, daß aus **nicht** B und A ein Widerspruch (also $\neg A$) folgt
- Aussagenlogisch ist $\neg(\neg B \wedge A)$ äquivalent zu $A \Rightarrow B$
- Beispiel: *Wenn für eine natürliche Zahl x gilt $x^2 > 1$, dann ist $x \geq 2$*

Beweis: *Sei $x^2 > 1$.*

Wenn $x \geq 2$ nicht gilt, dann ist $x=1$ oder $x=0$.

Wegen $1^2=1$ und $0^2=0$ ist $x^2 > 1$ in beiden Fällen falsch.

Also muss $x \geq 2$ sein

*Diese Art Beweisführung erscheint zunächst unnatürlich und erfordert etwas Training
Formalisieren Sie diesen Beweis als Übung*

WIDERLEGUNGSBEWEISE

ZEIGE, DASS EINE BEHAUPTUNG B NICHT GILT

- **Widerspruchsbeweise zeigen, daß B niemals gelten kann**
 - Zeige, daß aus Annahme B ein Widerspruch folgt ($\hat{=}$ Regel \neg -intro)
 - Beispiel: *Ist S endliche Teilmenge einer unendlichen Menge U , dann ist das Komplement von S bezüglich U unendlich*
 - Beweisidee: *Wenn \bar{S} endlich wäre, dann müsste auch U endlich sein.**
 - Ausführlich: *Wir nehmen an \bar{S} sei endlich. Da S ebenfalls endlich ist, muß U aufgrund des Satzes auf Folie 11 endlich sein. Dies ist ein Widerspruch, da U unendlich ist. Also ist die Annahme falsch und \bar{S} ist nicht endlich*

WIDERLEGUNGSBEWEISE

ZEIGE, DASS EINE BEHAUPTUNG B NICHT GILT

- **Widerspruchsbeweise zeigen, daß B niemals gelten kann**
 - Zeige, daß aus Annahme B ein Widerspruch folgt ($\hat{=}$ Regel \neg -intro)
 - Beispiel: *Ist S endliche Teilmenge einer unendlichen Menge U , dann ist das Komplement von S bezüglich U unendlich*
 - Beweisidee: *Wenn \bar{S} endlich wäre, dann müsste auch U endlich sein.**
 - Ausführlich: *Wir nehmen an \bar{S} sei endlich. Da S ebenfalls endlich ist, muß U aufgrund des Satzes auf Folie 11 endlich sein. Dies ist ein Widerspruch, da U unendlich ist. Also ist die Annahme falsch und \bar{S} ist nicht endlich*
- **Gegenbeispiele zeigen, daß B nicht immer wahr sein kann**
 - B ist nicht *allgemeingültig*, wenn es ein einziges Gegenbeispiel gibt
 - Beispiel: *Wenn x eine Primzahl ist, dann ist x ungerade* ist falsch
 - Gegenbeispiel: *2 ist eine gerade Zahl, die eine Primzahl ist*

* Wir haben zugunsten einer einfacheren Schreibweise die Notation \bar{S} als Abkürzung für “das Komplement von S bezüglich U ” verwendet

EIN KLASSISCHER WIDERSPRUCHSBEWEIS

- *Es gibt unendlich viele Primzahlen*

– *Wir nehmen an, es gäbe nur endlich viele Primzahlen p_1, \dots, p_{max} .*

Dann ist jede natürliche Zahl $n \in \mathbb{N}$, die größer als 1 ist, durch mindestens eine dieser Primzahlen teilbar.

Sei $x = 1 + \prod_{i=1}^{max} p_i$ das Produkt all dieser Primzahlen plus 1.

Dann ist x durch keine der Zahlen p_i teilbar

(Beweis unten)

Dies ist ein Widerspruch. Also ist die Annahme falsch und somit gibt es unendlich viele Primzahlen.

EIN KLASSISCHER WIDERSPRUCHSBEWeis

- *Es gibt unendlich viele Primzahlen*

- *Wir nehmen an, es gäbe nur endlich viele Primzahlen p_1, \dots, p_{max} .*

Dann ist jede natürliche Zahl $n \in \mathbb{N}$, die größer als 1 ist, durch mindestens eine dieser Primzahlen teilbar.

Sei $x = 1 + \prod_{i=1}^{max} p_i$ das Produkt all dieser Primzahlen plus 1.

Dann ist x durch keine der Zahlen p_i teilbar (Beweis unten)

Dies ist ein Widerspruch. Also ist die Annahme falsch und somit gibt es unendlich viele Primzahlen.

- *Für alle $p \geq 2$ und $k \in \mathbb{N}$ ist $x = k * p + 1$ nicht durch p teilbar*

- *Wir nehmen an x sei durch p teilbar, also $x = j * p$ für ein $j \in \mathbb{N}$. Dann muß $j > k$ sein, also $j - k \geq 1$, und es gilt $1 = x - k * p = (j - k) * p \geq 1 * p \geq 2$*

Dies ist ein Widerspruch, also kann x nicht durch p teilbar sein.

BEWEIS DER IRRATIONALITÄT VON ZAHLEN

- *Die Wurzel von 2 ist keine rationale Zahl*

- *Wir nehmen an $\sqrt{2}$ sei rational, d.h. $\sqrt{2} = p/q$ für zwei teilerfremde Zahlen $p, q \in \mathbb{Z}$. Dann ist $p^2 = 2q^2$, also eine gerade Zahl.*

- Damit muß auch p gerade sein, also $p = 2r$ für ein $r \in \mathbb{Z}$.*

- Dann ist aber auch $q^2 = 2r^2$, also ist q ebenfalls gerade.*

- Damit sind p und q gerade, also nicht teilerfremd.*

- Dies ist ein Widerspruch und damit kann $\sqrt{2}$ nicht rational sein.*

BEWEIS DER IRRATIONALITÄT VON ZAHLEN

- *Die Wurzel von 2 ist keine rationale Zahl*

- Wir nehmen an $\sqrt{2}$ sei rational, d.h. $\sqrt{2} = p/q$ für zwei teilerfremde Zahlen $p, q \in \mathbb{Z}$. Dann ist $p^2 = 2q^2$, also eine gerade Zahl.

Damit muß auch p gerade sein, also $p = 2r$ für ein $r \in \mathbb{Z}$.

Dann ist aber auch $q^2 = 2r^2$, also ist q ebenfalls gerade.

Damit sind p und q gerade, also nicht teilerfremd.

Dies ist ein Widerspruch und damit kann $\sqrt{2}$ nicht rational sein.

- *Der Zweierlogarithmus von 3 ist keine rationale Zahl*

- Wir nehmen an $\log_2 3$ sei rational, d.h. $\log_2 3 = p/q$ für zwei Zahlen $p, q \in \mathbb{Z}$. Dann ist $3^q = 2^p$, also eine gerade Zahl.

Dies ist ein Widerspruch, und damit kann $\log_2 3$ nicht rational sein.

\mathbb{R} IST ÜBERABZÄHLBAR (MÄCHTIGER ALS \mathbb{N})

- **Beweis verkettet mehrere Argumente**

- Wir zeigen, daß es keine surjektive Funktion $f:\mathbb{N}\rightarrow(0,1)$ geben kann
- Dann kann es auch keine Bijektion $f:\mathbb{N}\rightarrow(0,1)$ geben
- Da $(0,1)$ und \mathbb{R} gleichmächtig sind, gibt es keine Bijektion $f:\mathbb{N}\rightarrow\mathbb{R}$
- Damit kann \mathbb{R} nicht abzählbar sein

\mathbb{R} IST ÜBERABZÄHLBAR (MÄCHTIGER ALS \mathbb{N})

- **Beweis verkettet mehrere Argumente**

- Wir zeigen, daß es keine surjektive Funktion $f:\mathbb{N}\rightarrow(0,1)$ geben kann
- Dann kann es auch keine Bijektion $f:\mathbb{N}\rightarrow(0,1)$ geben
- Da $(0,1)$ und \mathbb{R} gleichmächtig sind, gibt es keine Bijektion $f:\mathbb{N}\rightarrow\mathbb{R}$
- Damit kann \mathbb{R} nicht abzählbar sein

- *Es gibt keine surjektive Funktion $f:\mathbb{N}\rightarrow(0,1)$*

- *Wir nehmen an, f existiert, d.h. für alle $x\in(0,1)$ gibt es ein $i\in\mathbb{N}$ mit $x=f(i)$. Wir konstruieren ein $z\in(0,1)$ das von allen $f(i)$ verschieden ist. Dazu bezeichnen wir die j -te Dezimalstelle einer Zahl $x\in(0,1)$ mit x_j und definieren $z_j=4$, falls $f(j)_j=5$ ist, und sonst $z_j=5$. Damit ist $z_j\neq f(j)_j$ für alle j , also kann es kein $i\in\mathbb{N}$ geben mit $z=f(i)$.*
- *Dies ist ein Widerspruch und somit gibt es kein surjektives $f:\mathbb{N}\rightarrow(0,1)$.*

\mathbb{R} IST ÜBERABZÄHLBAR (MÄCHTIGER ALS \mathbb{N})

- **Beweis verkettet mehrere Argumente**

- Wir zeigen, daß es keine surjektive Funktion $f:\mathbb{N}\rightarrow(0,1)$ geben kann
- Dann kann es auch keine Bijektion $f:\mathbb{N}\rightarrow(0,1)$ geben
- Da $(0,1)$ und \mathbb{R} gleichmächtig sind, gibt es keine Bijektion $f:\mathbb{N}\rightarrow\mathbb{R}$
- Damit kann \mathbb{R} nicht abzählbar sein

- *Es gibt keine surjektive Funktion $f:\mathbb{N}\rightarrow(0,1)$*

- Wir nehmen an, f existiert, d.h. für alle $x\in(0,1)$ gibt es ein $i\in\mathbb{N}$ mit $x=f(i)$. Wir konstruieren ein $z\in(0,1)$ das von allen $f(i)$ verschieden ist. Dazu bezeichnen wir die j -te Dezimalstelle einer Zahl $x\in(0,1)$ mit x_j und definieren $z_j=4$, falls $f(j)_j=5$ ist, und sonst $z_j=5$. Damit ist $z_j\neq f(j)_j$ für alle j , also kann es kein $i\in\mathbb{N}$ geben mit $z=f(i)$.
- Dies ist ein Widerspruch und somit gibt es kein surjektives $f:\mathbb{N}\rightarrow(0,1)$.

- **Zusatzfrage: warum nicht einfach $z_j=9-f(j)_j$?**

\mathbb{R} IST ÜBERABZÄHLBAR (MÄCHTIGER ALS \mathbb{N})

- **Beweis verkettet mehrere Argumente**

- Wir zeigen, daß es keine surjektive Funktion $f:\mathbb{N}\rightarrow(0,1)$ geben kann
- Dann kann es auch keine Bijektion $f:\mathbb{N}\rightarrow(0,1)$ geben
- Da $(0,1)$ und \mathbb{R} gleichmächtig sind, gibt es keine Bijektion $f:\mathbb{N}\rightarrow\mathbb{R}$
- Damit kann \mathbb{R} nicht abzählbar sein

- *Es gibt keine surjektive Funktion $f:\mathbb{N}\rightarrow(0,1)$*

- Wir nehmen an, f existiert, d.h. für alle $x\in(0,1)$ gibt es ein $i\in\mathbb{N}$ mit $x=f(i)$. Wir konstruieren ein $z\in(0,1)$ das von allen $f(i)$ verschieden ist. Dazu bezeichnen wir die j -te Dezimalstelle einer Zahl $x\in(0,1)$ mit x_j und definieren $z_j=4$, falls $f(j)_j=5$ ist, und sonst $z_j=5$. Damit ist $z_j\neq f(j)_j$ für alle j , also kann es kein $i\in\mathbb{N}$ geben mit $z=f(i)$.
- Dies ist ein Widerspruch und somit gibt es kein surjektives $f:\mathbb{N}\rightarrow(0,1)$.

- **Zusatzfrage: warum nicht einfach $z_j=9-f(j)_j$?**

- Für $z=0.4999999\dots=0.50000000\dots$ wäre $z_j=9-z_j$ kein Widerspruch
- Auch wenn es unwahrscheinlich ist, daß wir genau diese Zahl erreichen, wäre der Beweis nicht mehr überzeugend

Spezielle Form von Widerlegungsbeweisen

Konstruktion von Gegenbeispielen für Aussagen über unendliche Objekte

Wichtig für Unmöglichkeitsbeweise in der Informatik

- **z.B. Terminierung von Programmen ist unentscheidbar**

Es gibt kein Programm, das testen kann, ob ein beliebiges Programm bei einer bestimmten Eingabe überhaupt anhält

- **Beweis stützt sich auf wenige Grundannahmen**

1. Programme und ihre Daten sind **als Zahlen codierbar**

Schreibweise: $p_i(j) \hat{=}$ Anwendung des i -ten Programms auf die Zahl j

2. Computer sind **universelle Maschinen**

Bei Eingabe von Programm und Daten berechnen sie das Ergebnis

3. Man kann Programme **beliebig zu neuen Programmen zusammensetzen**

... und die Nummer des neuen Programms berechnen

PROGRAMMTERMINIERUNG IST UNENTSCHEIDBAR

- **Annahme:** es gibt ein Programm für den Terminierungstest

– $\text{Term}(i, j) = 1$ falls $p_i(j)$ anhält (sonst 0)

	0	1	2	3	4	...
p_0	×	×	×	⊥	×	...
p_1	⊥	⊥	×	×	×	...
p_2	×	×	⊥	×	×	...
p_3	⊥	×	⊥	×	⊥	...
⋮	⋮	⋮	⋮	⋮	⋮	...

× $\hat{=}$ Terminierung, ⊥ $\hat{=}$ hält nicht

PROGRAMMTERMINIERUNG IST UNENTSCHEIDBAR

- **Annahme:** es gibt ein Programm für den Terminierungstest

– $\text{Term}(i, j) = 1$ falls $p_i(j)$ anhält (sonst 0)

- **Konstruiere ein Programm Unsinn**

wie folgt:

$$\text{Unsinn}(i) = \begin{cases} 0 & \text{wenn } \text{Term}(i, i) = 0 \\ \perp & \text{sonst} \end{cases}$$

	0	1	2	3	4	...
p_0	×	×	×	⊥	×	...
p_1	⊥	⊥	×	×	×	...
p_2	×	×	⊥	×	×	...
p_3	⊥	×	⊥	×	⊥	...
⋮	⋮	⋮	⋮	⋮	⋮	...

× $\hat{=}$ Terminierung, ⊥ $\hat{=}$ hält nicht

PROGRAMMTERMINIERUNG IST UNENTSCHEIDBAR

- **Annahme:** es gibt ein Programm für den Terminierungstest

– $\text{Term}(i, j) = 1$ falls $p_i(j)$ anhält (sonst 0)

- **Konstruiere ein Programm Unsinn**

wie folgt:

$$\text{Unsinn}(i) = \begin{cases} 0 & \text{wenn } \text{Term}(i, i) = 0 \\ \perp & \text{sonst} \end{cases}$$

	0	1	2	3	4	...
p_0	\perp	\times	\times	\perp	\times	...
p_1	\perp	\times	\times	\times	\times	...
p_2	\times	\times	\times	\times	\times	...
p_3	\perp	\times	\perp	\perp	\perp	...
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	...

$\times \hat{=}$ Terminierung, $\perp \hat{=}$ hält nicht

PROGRAMMTERMINIERUNG IST UNENTSCHEIDBAR

- **Annahme:** es gibt ein Programm für den Terminierungstest

– $\text{Term}(i, j) = 1$ falls $p_i(j)$ anhält (sonst 0)

- **Konstruiere ein Programm Unsinn**

wie folgt:

$$\text{Unsinn}(i) = \begin{cases} 0 & \text{wenn } \text{Term}(i, i) = 0 \\ \perp & \text{sonst} \end{cases}$$

- Weil **Unsinn** ein Programm ist, muß es eine **Nummer k** haben

	0	1	2	3	4	...
p_0	\perp	\times	\times	\perp	\times	...
p_1	\perp	\times	\times	\times	\times	...
p_2	\times	\times	\times	\times	\times	...
p_3	\perp	\times	\perp	\perp	\perp	...
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	...

$\times \hat{=}$ Terminierung, $\perp \hat{=}$ hält nicht

PROGRAMMTERMINIERUNG IST UNENTSCHEIDBAR

- **Annahme:** es gibt ein Programm für den Terminierungstest

– $\text{Term}(i, j) = 1$ falls $p_i(j)$ anhält (sonst 0)

- **Konstruiere ein Programm Unsinn**

wie folgt:

$$\text{Unsinn}(i) = \begin{cases} 0 & \text{wenn } \text{Term}(i, i) = 0 \\ \perp & \text{sonst} \end{cases}$$

- Weil **Unsinn** ein Programm ist, muß es eine **Nummer k** haben

- **Was macht $\text{Unsinn} = p_k$ bei Eingabe der eigenen Nummer als Daten?**

– Wenn $p_k(k)$ hält, dann $\text{Term}(k, k) = 1$, also hält **Unsinn**(k) nicht an **???**

– Wenn $p_k(k)$ nicht hält, dann $\text{Term}(k, k) = 0$, also hält **Unsinn**(k) an **???**

	0	1	2	3	4	...
p_0	\perp	\times	\times	\perp	\times	...
p_1	\perp	\times	\times	\times	\times	...
p_2	\times	\times	\times	\times	\times	...
p_3	\perp	\times	\perp	\times	\perp	...
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	...

$\times \hat{=}$ Terminierung, $\perp \hat{=}$ hält nicht

PROGRAMMTERMINIERUNG IST UNENTSCHEIDBAR

- **Annahme:** es gibt ein Programm für den Terminierungstest

– $\text{Term}(i, j) = 1$ falls $p_i(j)$ anhält (sonst 0)

- **Konstruiere ein Programm Unsinn**

wie folgt:

$$\text{Unsinn}(i) = \begin{cases} 0 & \text{wenn } \text{Term}(i, i) = 0 \\ \perp & \text{sonst} \end{cases}$$

- Weil **Unsinn** ein Programm ist, muß es eine **Nummer k** haben

- **Was macht $\text{Unsinn} = p_k$ bei Eingabe der eigenen Nummer als Daten?**

– Wenn $p_k(k)$ hält, dann $\text{Term}(k, k) = 1$, also hält **Unsinn**(k) nicht an **???**

– Wenn $p_k(k)$ nicht hält, dann $\text{Term}(k, k) = 0$, also hält **Unsinn**(k) an **???**

- **Dies ist ein Widerspruch,**

Also kann es den Test auf Terminierung nicht geben

	0	1	2	3	4	...
p_0	\perp	\times	\times	\perp	\times	...
p_1	\perp	\times	\times	\times	\times	...
p_2	\times	\times	\times	\times	\times	...
p_3	\perp	\times	\perp	\times	\perp	...
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	...

$\times \hat{=}$ Terminierung, $\perp \hat{=}$ hält nicht

INDUKTIVE BEWEISE

BEWEISE EINE BEHAUPTUNG B FÜR ALLE NATÜRLICHEN ZAHLEN

- **Standardinduktion (beginnend ab i)**
 - “Gilt B für i und B für $n+1$, wenn B für n gilt, dann gilt B für alle $n \geq i$ ”

INDUKTIVE BEWEISE

BEWEISE EINE BEHAUPTUNG B FÜR ALLE NATÜRLICHEN ZAHLEN

- **Standardinduktion (beginnend ab i)**

- “Gilt B für i und B für $n+1$, wenn B für n gilt, dann gilt B für alle $n \geq i$ ”

- Beispiel: *Für alle $x \geq 4$ gilt $2^x \geq x^2$*

Induktionsanfang $x=4$: *Es ist $2^x = 16 \geq 16 = x^2$*

Induktionsannahme: *Es gelte $2^n \geq n^2$ für ein beliebiges $n \geq 4$*

Induktionsschritt: *Es ist $2^{n+1} = 2 \cdot 2^n \geq 2n^2$ (aufgrund der Induktionsannahme)*

und $(n+1)^2 = n^2 + 2n + 1 = n(n + 2 + \frac{1}{n}) \leq n(n+n) = 2n^2$ (wegen $n \geq 4$)

also gilt $2^{n+1} \geq (n+1)^2$

Aufgrund des Induktionsprinzips gilt damit $2^n \geq n^2$ für alle $n \geq 4$

INDUKTIVE BEWEISE

BEWEISE EINE BEHAUPTUNG B FÜR ALLE NATÜRLICHEN ZAHLEN

- **Standardinduktion (beginnend ab i)**

- “Gilt B für i und B für $n+1$, wenn B für n gilt, dann gilt B für alle $n \geq i$ ”

- Beispiel: *Für alle $x \geq 4$ gilt $2^x \geq x^2$*

Induktionsanfang $x=4$: *Es ist $2^x = 16 \geq 16 = x^2$*

Induktionsannahme: *Es gelte $2^n \geq n^2$ für ein beliebiges $n \geq 4$*

Induktionsschritt: *Es ist $2^{n+1} = 2 \cdot 2^n \geq 2n^2$ (aufgrund der Induktionsannahme)*

und $(n+1)^2 = n^2 + 2n + 1 = n(n + 2 + \frac{1}{n}) \leq n(n+n) = 2n^2$ (wegen $n \geq 4$)

also gilt $2^{n+1} \geq (n+1)^2$

Aufgrund des Induktionsprinzips gilt damit $2^n \geq n^2$ für alle $n \geq 4$

- **Vollständige Induktion**

- “Folgt B für n , wenn B für alle $i \leq j < n$ gilt, dann gilt B für alle $n \geq i$ ”

- Mächtiger, da man nicht den unmittelbaren Vorgänger benutzen muss

ZWEI WICHTIGE SUMMENFORMELN

- **Gauß'sche Summenformel:** $\sum_{i=1}^n i = n(n+1)/2$

Induktionsanfang $n = 0$: *Es ist* $\sum_{i=1}^0 i = 0 = 0(0+1)/2$

Induktionsannahme: *Es gelte* $\sum_{i=1}^m i = m(m+1)/2$ für ein $m \in \mathbb{N}$

Induktionsschritt: *Es sei* $n = m+1$.

$$\begin{aligned} \text{Dann ist } \sum_{i=1}^n i &= \sum_{i=1}^m i + (m+1) = m(m+1)/2 + (m+1) \\ &= (m+1)(m+2)/2 = n(n+1)/2 \end{aligned}$$

Aufgrund des Induktionsprinzips gilt $\sum_{i=1}^n i = n(n+1)/2$ für alle $n \in \mathbb{N}$

ZWEI WICHTIGE SUMMENFORMELN

- **Gauß'sche Summenformel:** $\sum_{i=1}^n i = n(n+1)/2$

Induktionsanfang $n = 0$: *Es ist* $\sum_{i=1}^0 i = 0 = 0(0+1)/2$

Induktionsannahme: *Es gelte* $\sum_{i=1}^m i = m(m+1)/2$ für ein $m \in \mathbb{N}$

Induktionsschritt: *Es sei* $n = m+1$.

$$\begin{aligned} \text{Dann ist } \sum_{i=1}^n i &= \sum_{i=1}^m i + (m+1) = m(m+1)/2 + (m+1) \\ &= (m+1)(m+2)/2 = n(n+1)/2 \end{aligned}$$

Aufgrund des Induktionsprinzips gilt $\sum_{i=1}^n i = n(n+1)/2$ für alle $n \in \mathbb{N}$

- **Die "Informatik-Formel":** $\sum_{i=1}^n 2^{i-1} = 2^n - 1$

(Beschreibt die größte mit n Bit darstellbare Zahl)

Induktionsanfang $n = 0$: *Es ist* $\sum_{i=1}^0 2^{i-1} = 0 = 2^0 - 1$

Induktionsannahme: *Es gelte* $\sum_{i=1}^m 2^{i-1} = 2^m - 1$ für ein $m \in \mathbb{N}$

Induktionsschritt: *Es sei* $n = m+1$.

$$\begin{aligned} \text{Dann ist } \sum_{i=1}^n 2^{i-1} &= \sum_{i=1}^m 2^{i-1} + 2^{m+1-1} = 2^m - 1 + 2^m \\ &= 2^{m+1} - 1 = 2^n - 1 \end{aligned}$$

Aufgrund des Induktionsprinzips gilt $\sum_{i=1}^n i = n(n+1)/2$ für alle $n \in \mathbb{N}$

DAS BRIEFMARKEN PROBLEM



Ist es möglich, jedes Porto ab 8 Cent nur mit 3c und 5c Briefmarken zu erzeugen?

DAS BRIEFMARKEN PROBLEM



Ist es möglich, jedes Porto ab 8 Cent nur mit 3c und 5c Briefmarken zu erzeugen?











● **Lösungsidee**

$$8c = \begin{array}{c} \text{5 Cent} \\ \text{Deutschland} \end{array} \begin{array}{c} \text{Deutschland} \\ 3 \text{ Cent} \end{array}, \quad 9c = \begin{array}{c} \text{Deutschland} \\ 3 \text{ Cent} \end{array} \begin{array}{c} \text{Deutschland} \\ 3 \text{ Cent} \end{array} \begin{array}{c} \text{Deutschland} \\ 3 \text{ Cent} \end{array}, \quad 10c = \begin{array}{c} \text{5 Cent} \\ \text{Deutschland} \end{array} \begin{array}{c} \text{5 Cent} \\ \text{Deutschland} \end{array},$$
$$11c = \begin{array}{c} \text{5 Cent} \\ \text{Deutschland} \end{array} \begin{array}{c} \text{Deutschland} \\ 3 \text{ Cent} \end{array} \begin{array}{c} \text{Deutschland} \\ 3 \text{ Cent} \end{array}, \dots$$

DAS BRIEFMARKEN PROBLEM



Ist es möglich, jedes Porto ab 8 Cent nur mit 3c und 5c Briefmarken zu erzeugen?











- **Lösungsidee** $8c =$  , $9c =$   , $10c =$  ,
- $11c =$   , ...

- **Informales Argument ist leicht einzusehen**
 - Basisfälle 8, 9, 10 sind lösbar wie oben illustriert.
 - Lösung für größere n wird aus der für $n-3$ mit weiteren 3c erzeugt

DAS BRIEFMARKEN PROBLEM



Ist es möglich, jedes Porto ab 8 Cent nur mit 3c und 5c Briefmarken zu erzeugen?

- **Lösungsidee** $8c =$  , $9c =$   , $10c =$  ,
 $11c =$   , \dots

- **Informales Argument ist leicht einzusehen**

- Basisfälle 8, 9, 10 sind lösbar wie oben illustriert.
- Lösung für größere n wird aus der für $n-3$ mit weiteren 3c erzeugt

- **Induktionsbeweis benötigt vollständige Induktion**

Behauptung $B(n)$ (für $n \geq 8$): *Es gibt $a, b \in \mathbb{N}$ mit $n = 3 \cdot a + 5 \cdot b$*

Induktionsannahme: *Sei $n \geq 8$ beliebig. Es gelte $B(j)$ für alle $8 \leq j < n$*

Induktionsschritt: *Falls $n = 8, 9, 10$ wähle $(a, b) = (1, 1), (3, 0)$, bzw. $(0, 2)$.*

Ansonsten gibt es für $j = n - 3 \geq 8$ Zahlen a', b' mit $j = 3 \cdot a' + 5 \cdot b'$. (IA)

Es folgt $n = j + 3 = 3 \cdot (a' + 1) + 5 \cdot b'$. Wähle $(a, b) = (a' + 1, b')$

Aufgrund des Prinzips der vollständigen Induktion folgt $B(n)$ für alle $n \geq 8$

EXISTENZ EINER GANZZAHLIGEN QUADRATWURZEL

Für alle $n \in \mathbb{N}$ existiert ein $r \in \mathbb{N}$ mit $r^2 \leq n < (r+1)^2$

● Beweis mit Standardinduktion

Behauptung $B(n)$: *es existiert ein $r \in \mathbb{N}$ mit $r^2 \leq n < (r+1)^2$*

Induktionsanfang $n = 0$: wähle $r = 0$, dann gilt $0^2 \leq 0 < (0+1)^2$, also $B(0)$.

Induktionsannahme: *Es gelte $B(m)$ für ein $m \in \mathbb{N}$*

Induktionsschritt: *Es sei $n = m+1$.*

Wegen $B(m)$ gibt es ein r_m mit $r_m^2 \leq m < (r_m+1)^2$

Falls $(r_m+1)^2 \leq m+1 = n$ dann wähle $r = r_m+1$ und sonst $r = r_m$

In beiden Fällen gilt $r^2 \leq n < (r+1)^2$ (ausrechnen!), also $B(n)$

Aufgrund des Induktionsprinzips gilt $B(n)$ für alle $n \in \mathbb{N}$

EXISTENZ EINER GANZZAHLIGEN QUADRATWURZEL

Für alle $n \in \mathbb{N}$ existiert ein $r \in \mathbb{N}$ mit $r^2 \leq n < (r+1)^2$

● Beweis mit Standardinduktion

Behauptung $B(n)$: *es existiert ein $r \in \mathbb{N}$ mit $r^2 \leq n < (r+1)^2$*

Induktionsanfang $n = 0$: wähle $r = 0$, dann gilt $0^2 \leq 0 < (0+1)^2$, also $B(0)$.

Induktionsannahme: *Es gelte $B(m)$ für ein $m \in \mathbb{N}$*

Induktionsschritt: *Es sei $n = m+1$.*

Wegen $B(m)$ gibt es ein r_m mit $r_m^2 \leq m < (r_m+1)^2$

Falls $(r_m+1)^2 \leq m+1 = n$ dann wähle $r = r_m+1$ und sonst $r = r_m$

In beiden Fällen gilt $r^2 \leq n < (r+1)^2$ (ausrechnen!), also $B(n)$

Aufgrund des Induktionsprinzips gilt $B(n)$ für alle $n \in \mathbb{N}$

● Beweis mit vollständiger Induktion

Induktionsannahme: *Es gelte $B(m)$ für alle $m < n$*

Induktionsschritt: *Wegen $B(n \div 4)$ gibt es ein r_0 mit $r_0^2 \leq n \div 4 < (r_0+1)^2$*

Falls $(2r_0+1)^2 \leq n$ dann wähle $r = 2r_0+1$ und sonst $r = 2r_0$

In beiden Fällen gilt $r^2 \leq n < (r+1)^2$ (ausrechnen!), also $B(n)$

Der Beweis hat eine ähnliche Struktur, springt aber weiter zurück. Er enthält implizit einen sehr effizienten Algorithmus zur Berechnung von $\lfloor \sqrt{n} \rfloor$. Wie sieht dieser aus??

SIMULTANE (GLEICHZEITIGE) INDUKTION

.. WENN MEHRERE AUSSAGEN VONEINANDER ABHÄNGEN

- **Häufig bei rekursiv definierten Funktionen**

- Beispiel: Sei $f(0) = 0$ und $f(n+1) = 1 - f(n)$
- Zeige: $f(n) = 0$, falls n gerade und sonst 1

SIMULTANE (GLEICHZEITIGE) INDUKTION

.. WENN MEHRERE AUSSAGEN VONEINANDER ABHÄNGEN

- **Häufig bei rekursiv definierten Funktionen**
 - Beispiel: Sei $f(0) = 0$ und $f(n+1) = 1 - f(n)$
 - Zeige: $f(n) = 0$, falls n gerade und sonst 1
- **Problem beinhaltet zwei zusammengehörige Aussagen**

$B_1(n)$: Für gerade n ist $f(n)=0$, $B_2(n)$: Für ungerade n ist $f(n)=1$

SIMULTANE (GLEICHZEITIGE) INDUKTION

.. WENN MEHRERE AUSSAGEN VONEINANDER ABHÄNGEN

- **Häufig bei rekursiv definierten Funktionen**

- Beispiel: Sei $f(0) = 0$ und $f(n+1) = 1 - f(n)$
- Zeige: $f(n) = 0$, falls n gerade und sonst 1

- **Problem beinhaltet zwei zusammengehörige Aussagen**

$B_1(n)$: Für gerade n ist $f(n)=0$, $B_2(n)$: Für ungerade n ist $f(n)=1$

- **Induktionsbeweis zeigt beide Aussagen gleichzeitig**

Induktionsanfang $n=0$: n ist gerade und es ist $f(0)=0$, also gilt $B_1(0)$
 n ist nicht ungerade, also gilt $B_2(0)$

Induktionsannahme: *Es gelte $B_1(n)$ und $B_2(n)$.* (Voraussetzung nicht erfüllt)

Induktionsschritt: *Falls $n+1$ gerade ist, dann ist n ungerade*

und es gilt $f(n)=1$ wegen $B_2(n)$ also $f(n+1)=0$ und damit $B_1(n+1)$

Ausserdem gilt $B_2(n+1)$, da $n+1$ nicht ungerade ist

Ansonsten ist n gerade und $f(n+1)=1$ wegen $B_1(n)$, also gilt $B_2(n+1)$

Ausserdem gilt $B_1(n+1)$, da $n+1$ nicht gerade ist

Aufgrund des Induktionsprinzips folgen beide Behauptungen für alle n

Um zu zeigen, daß $f(n) = 0$, falls n gerade und sonst 1 ist, zeigen wir durch Induktion, daß für alle $n \in \mathbb{N}$ die folgenden beiden Aussagen gelten

$$B_1(n): n \text{ gerade} \Leftrightarrow f(n)=0$$

$$B_2(n): n \text{ ungerade} \Leftrightarrow f(n)=1$$

Induktionsanfang $n=0$:

$B_1(n)$: 0 ist gerade und nach Definition ist $f(0) = 0$, also gilt Aussage $B_1(n)$.

$B_2(n)$: 0 ist nicht ungerade und nach Definition ist $f(0) \neq 1$, also gilt die

Äquivalenz $B_2(n)$, da jeweils die rechte und linke Seite falsch ist.

Induktionsannahme: Es gelte $B_1(m)$ und $B_2(m)$ für ein beliebiges $m \in \mathbb{N}$.

Induktionsschritt: Es sei $n = m+1$.

$B_1(n)$: Es ist $n = m+1$ gerade $\Leftrightarrow m$ ist ungerade

$$\Leftrightarrow f(m) = 1 \quad (\text{Induktionsannahme } B_2(m))$$

$$\Leftrightarrow f(n) = 1 - f(m) = 0$$

$B_2(n)$: Es ist $n = m+1$ ungerade $\Leftrightarrow m$ ist gerade

$$\Leftrightarrow f(m) = 0 \quad (\text{Induktionsannahme } B_1(m))$$

$$\Leftrightarrow f(n) = 1 - f(m) = 1$$

Aufgrund des Induktionsprinzips gilt $B_1(n)$ und $B_2(n)$ für alle $n \in \mathbb{N}$

EXISTENZ EINES GRÖSSTEN GEMEINSAMEN TEILERS

Für alle $a, b \in \mathbb{N}$ existiert ein größter gemeinsamer Teiler $x \in \mathbb{N}$

- **Wann ist x der größte gemeinsame Teiler von a und b**
 - x teilt a , x teilt b , und für jedes y , das a und b teilt, gilt $y \leq x$

EXISTENZ EINES GRÖSSTEN GEMEINSAMEN TEILERS

Für alle $a, b \in \mathbb{N}$ existiert ein größter gemeinsamer Teiler $x \in \mathbb{N}$

- **Wann ist x der größte gemeinsame Teiler von a und b**

– x teilt a , x teilt b , und für jedes y , das a und b teilt, gilt $y \leq x$

- **Beweis benötigt vollständige Induktion auf Paaren**

Behauptung $B(a, b)$: *es gibt einen größten gemeinsamen Teiler x von a und b*

Induktionsannahme: Sei a, b beliebig. $B(a', b')$ gelte für alle $(a', b') < (a, b)^*$

Induktionsschritt: Falls $a = b$ dann wähle $x = a$.

Falls $a > b$ dann gibt es wegen $B(a-b, b)$ einen größten gemeinsamen Teiler x_1 von $a-b$ und b . Wähle $x = x_1$.

Falls $a < b$ dann gibt es wegen $B(a, b-a)$ einen größten gemeinsamen Teiler x_2 von a und $b-a$. Wähle $x = x_2$.

In allen drei Fällen ist x der größte gemeinsame Teiler von a und b .

Aufgrund des Induktionsprinzips folgt $B(a, b)$ für alle $a, b \in \mathbb{N}$

Der Beweis enthält implizit einen sehr effizienten Algorithmus zur Berechnung von $ggT(a, b)$

* Es ist $(a', b') < (a, b)$ genau dann wenn $a' < a$ oder $a' = a$ und $b' < b$

Viele Datenstrukturen lassen sich induktiv repräsentieren

- **\mathbb{N} : Natürliche Zahlen**

- $0 \in \mathbb{N}$ (Null ist eine natürliche Zahl)
- $n \in \mathbb{N} \Rightarrow n+1 \in \mathbb{N}$ (jeder Nachfolger einer natürlichen Zahl ist eine natürliche Zahl)

- **List T : Listen über einem Datentyp T**

- $[] \in \text{List } T$ (leere Liste - ohne Elemente)
- $l \in \text{List } T \wedge x \in T \Rightarrow x :: l \in \text{List } T$ (Vorstellen eines Elements aus T)

- **Σ^* : Wörter (Strings) über einem Alphabet Σ**

- $\epsilon \in \Sigma^*$ (leeres Wort - ohne Symbole)
- $w \in \Sigma^* \wedge a \in \Sigma \Rightarrow w a \in \Sigma^*$ (Anhängen eines Symbols aus Σ an ein Wort)

- **Tree T : Bäume mit Markierungen aus T**

- $x \in T \Rightarrow x \in \text{Tree } T$ (einzelner Knoten mit Markierung x)
- $x \in T \wedge t_1..t_n \in \text{Tree } T \Rightarrow (x, [t_1, \dots, t_n]) \in \text{Tree } T$
(Baum mit Wurzel x und Unterbäumen $t_1..t_n$)

Induktionsbeweise folgen der Struktur dieses Aufbaus

FORMALE BEWEISFÜHRUNG MIT INDUKTIVEN STRUKTUREN

- **Regeln zur Erzeugung von Elementen des Typs**

$$\frac{}{\text{Zero} \in \text{nat}} \text{Zero-intro} \quad \frac{n \in \text{nat}}{\text{Suc } n \in \text{nat}} \text{Suc-intro}$$

Regeln beschreiben den Typ `nat`, werden meist nur für Verifikation von Software gebraucht

- **Regeln zur Zerlegung von Elementen eines Typs**

- Per Konstruktion gilt: $n \in \text{nat} \Leftrightarrow n = \text{Zero} \vee \exists m : \text{nat}. n = \text{Suc } m$
- Also gilt eine Behauptung B für alle $n \in \text{nat}$ genau dann, wenn sie für `Zero` gilt und wenn sie für `Suc m` gilt, wann immer sie für m gilt
- Das ist die bekannte Induktionsregel, formuliert für den Typ `nat` statt \mathbb{N}

$$\text{Formal: } \frac{B[\text{Zero}] \quad \forall m : \text{nat}. B[m] \Rightarrow B[\text{Suc } m]}{\forall n : \text{nat}. B[n]} \text{nat-elim}$$

- **Regeln sind analog für den mathematischen Typ \mathbb{N}**

- Informal: Gilt $B(0)$ und $B(m) \Rightarrow B(m+1)$, dann gilt $\forall m \in \mathbb{N}. B(n)$

- Formal:
$$\frac{B(0) \quad \forall m \in \mathbb{N}. B(m) \Rightarrow B(m+1)}{\forall n \in \mathbb{N}. B(n)} \mathbb{N}\text{-elim}$$

Alle Induktionsbeweise lassen sich damit formalisieren

STRUKTURELLE INDUKTION

BEWEIS AUF BASIS DER STRUKTUR EINES INDUKTIVEN DATENTYPS

● **Listen** (mathematische Notation)

– Informal: Gilt $B([])$ und $B(l') \Rightarrow \forall x \in T. B(x :: l')$, dann $\forall l \in \text{List } T. B(l)$

– Formal:
$$\frac{B([]) \quad \forall l' \in \text{List } T. B(l') \Rightarrow \forall x \in T. B(x :: l')}{\forall l \in \text{List } T. B(l)} \quad \text{List-elim}$$

● **Wörter**

– Informal: Gilt $B(\epsilon)$ und $B(w') \Rightarrow \forall a \in \Sigma. B(w' a)$, dann $\forall w \in \Sigma^*. B(w)$

– Formal:
$$\frac{B(\epsilon) \quad \forall w' \in \Sigma^*. B(w') \Rightarrow \forall a \in \Sigma. B(w' a)}{\forall w \in \Sigma^*. B(w)} \quad \text{String-elim}$$

● **Bäume** (formale regel **Tree-elim**)

$$\frac{\forall x \in T. B(x) \quad \forall t_1..t_n \in \text{Tree } T. B(t_1) \wedge \dots \wedge B(t_n) \Rightarrow \forall x \in T. B((x, [t_1, \dots, t_n]))}{\forall t \in \text{Tree } T. B(t)}$$

● **Allgemeines Prinzip:** Gilt Behauptung B für die Basiselemente des Typs und für zusammengesetzte Elemente, wenn B für die entsprechenden Unterelemente gilt, dann gilt B für alle Elemente des Typs.

Häufig eingesetzt für Analyse von Listen- und Baumstrukturen (Suchen, Sortieren, ...) oder von syntaktischen Strukturen (Formeln, Programmiersprachen, ...)

STRUKTURELLE INDUKTION AUF LISTEN

- *Die Summe einer Liste l von positiven ganzen Zahlen ist mindestens so groß wie ihre Länge*

Induktionsanfang $l = []$: *Die Summe und die Länge von l sind 0*

Induktionsannahme: *Es gelte $sum(l') \geq |l'|$*

Induktionsschritt: *Sei $x \in \mathbb{Z}^+$ und $l = x :: l'$*

$$\text{Dann gilt } sum(l) = sum(l') + x \geq sum(l') + 1 \geq |l'| + 1 = |l| \quad (\text{IA})$$

Aufgrund des Induktionsprinzips folgt die Behauptung für alle Listen

STRUKTURELLE INDUKTION AUF LISTEN

- *Die Summe einer Liste l von positiven ganzen Zahlen ist mindestens so groß wie ihre Länge*

Induktionsanfang $l = []$: *Die Summe und die Länge von l sind 0*

Induktionsannahme: *Es gelte $sum(l') \geq |l'|$*

Induktionsschritt: *Sei $x \in \mathbb{Z}^+$ und $l = x :: l'$*

$$\text{Dann gilt } sum(l) = sum(l') + x \geq sum(l') + 1 \geq |l'| + 1 = |l| \quad (\text{IA})$$

Aufgrund des Induktionsprinzips folgt die Behauptung für alle Listen

- **Formale Beweisskizze für $\forall l \in \text{List } \mathbb{Z}^+. sum(l) \geq |l|$**

Aussage	Begründung
1.	
2.	
3.	
4.	
5.	
6.	
7.	
8.	
9.	
10.	
11 $\forall l \in \text{List } \mathbb{Z}^+. sum(l) \geq l $	<i>Zielbehauptung</i>

STRUKTURELLE INDUKTION AUF LISTEN

- *Die Summe einer Liste l von positiven ganzen Zahlen ist mindestens so groß wie ihre Länge*

Induktionsanfang $l = []$: *Die Summe und die Länge von l sind 0*

Induktionsannahme: *Es gelte $sum(l') \geq |l'|$*

Induktionsschritt: *Sei $x \in \mathbb{Z}^+$ und $l = x :: l'$*

$$\text{Dann gilt } sum(l) = sum(l') + x \geq sum(l') + 1 \geq |l'| + 1 = |l| \quad (\text{IA})$$

Aufgrund des Induktionsprinzips folgt die Behauptung für alle Listen

- **Formale Beweisskizze für $\forall l \in \text{List } \mathbb{Z}^+. sum(l) \geq |l|$**

Aussage	Begründung
1.	
2. $sum([]) \geq [] $	<i>zu zeigen</i>
3.	
4.	
5.	
6.	
7.	
8.	
9.	
10 $\forall l' \in \text{List } \mathbb{Z}^+. sum(l') \geq l' \Rightarrow \forall x \in \mathbb{Z}^+. sum(x :: l') \geq x :: l' $	<i>zu zeigen</i>
11 $\forall l \in \text{List } \mathbb{Z}^+. sum(l) \geq l $	List-elim (2) (10)

STRUKTURELLE INDUKTION AUF LISTEN

- *Die Summe einer Liste l von positiven ganzen Zahlen ist mindestens so groß wie ihre Länge*

Induktionsanfang $l = []$: *Die Summe und die Länge von l sind 0*

Induktionsannahme: *Es gelte $sum(l') \geq |l'|$*

Induktionsschritt: *Sei $x \in \mathbb{Z}^+$ und $l = x :: l'$*

$$\text{Dann gilt } sum(l) = sum(l') + x \geq sum(l') + 1 \geq |l'| + 1 = |l| \quad (\text{IA})$$

Aufgrund des Induktionsprinzips folgt die Behauptung für alle Listen

- **Formale Beweisskizze für $\forall l \in \text{List } \mathbb{Z}^+. sum(l) \geq |l|$**

Aussage	Begründung
1. $0 \geq 0$	<i>zu zeigen</i>
2. $sum([]) \geq [] $	fold sum , $length$ (1)
3.	
4.	
5.	
6.	
7.	
8.	
9.	
10 $\forall l' \in \text{List } \mathbb{Z}^+. sum(l') \geq l' \Rightarrow \forall x \in \mathbb{Z}^+. sum(x :: l') \geq x :: l' $	<i>zu zeigen</i>
11 $\forall l \in \text{List } \mathbb{Z}^+. sum(l) \geq l $	List-elim (2) (10)

STRUKTURELLE INDUKTION AUF LISTEN

- *Die Summe einer Liste l von positiven ganzen Zahlen ist mindestens so groß wie ihre Länge*

Induktionsanfang $l = []$: *Die Summe und die Länge von l sind 0*

Induktionsannahme: *Es gelte $sum(l') \geq |l'|$*

Induktionsschritt: *Sei $x \in \mathbb{Z}^+$ und $l = x :: l'$*

$$\text{Dann gilt } sum(l) = sum(l') + x \geq sum(l') + 1 \geq |l'| + 1 = |l| \quad (\text{IA})$$

Aufgrund des Induktionsprinzips folgt die Behauptung für alle Listen

- **Formale Beweisskizze für $\forall l \in \text{List } \mathbb{Z}^+. sum(l) \geq |l|$**

Aussage	Begründung
1. $0 \geq 0$	arithmetic
2. $sum([]) \geq [] $	fold sum , $length$ (1)
3.	
4.	
5.	
6.	
7.	
8.	
9.	
10 $\forall l' \in \text{List } \mathbb{Z}^+. sum(l') \geq l' \Rightarrow \forall x \in \mathbb{Z}^+. sum(x :: l') \geq x :: l' $	<i>zu zeigen</i>
11 $\forall l \in \text{List } \mathbb{Z}^+. sum(l) \geq l $	List-elim (2) (10)

STRUKTURELLE INDUKTION AUF LISTEN

- *Die Summe einer Liste l von positiven ganzen Zahlen ist mindestens so groß wie ihre Länge*

Induktionsanfang $l = []$: *Die Summe und die Länge von l sind 0*

Induktionsannahme: *Es gelte $sum(l') \geq |l'|$*

Induktionsschritt: *Sei $x \in \mathbb{Z}^+$ und $l = x :: l'$*

$$\text{Dann gilt } sum(l) = sum(l') + x \geq sum(l') + 1 \geq |l'| + 1 = |l| \quad (\text{IA})$$

Aufgrund des Induktionsprinzips folgt die Behauptung für alle Listen

- **Formale Beweisskizze für $\forall l \in \text{List } \mathbb{Z}^+. sum(l) \geq |l|$**

Aussage	Begründung
1. $0 \geq 0$	arithmetic
2. $sum([]) \geq [] $	fold sum , $length$ (1)
3. $l' : \text{List } \mathbb{Z}^+$	Neue Variable
4.	
5.	
6.	
7.	
8.	
9. $sum(l') \geq l' \Rightarrow \forall x \in \mathbb{Z}^+. sum(x :: l') \geq x :: l' $	<i>zu zeigen</i>
10. $\forall l' \in \text{List } \mathbb{Z}^+. sum(l') \geq l' \Rightarrow \forall x \in \mathbb{Z}^+. sum(x :: l') \geq x :: l' $	\forall -intro (3) (9)
11. $\forall l \in \text{List } \mathbb{Z}^+. sum(l) \geq l $	List-elim (2) (10)

STRUKTURELLE INDUKTION AUF LISTEN

- *Die Summe einer Liste l von positiven ganzen Zahlen ist mindestens so groß wie ihre Länge*

Induktionsanfang $l = []$: *Die Summe und die Länge von l sind 0*

Induktionsannahme: *Es gelte $sum(l') \geq |l'|$*

Induktionsschritt: *Sei $x \in \mathbb{Z}^+$ und $l = x :: l'$*

$$\text{Dann gilt } sum(l) = sum(l') + x \geq sum(l') + 1 \geq |l'| + 1 = |l| \quad (\text{IA})$$

Aufgrund des Induktionsprinzips folgt die Behauptung für alle Listen

- **Formale Beweisskizze für $\forall l \in \text{List } \mathbb{Z}^+. sum(l) \geq |l|$**

Aussage	Begründung
1. $0 \geq 0$	arithmetic
2. $sum([]) \geq [] $	fold sum , $length$ (1)
3. $l' : \text{List } \mathbb{Z}^+$	Neue Variable
4. $sum(l') \geq l' $	“Induktionsannahme”
5.	
6.	
7.	
8. $\forall x \in \mathbb{Z}^+. sum(x :: l') \geq x :: l' $	<i>zu zeigen</i>
9. $sum(l') \geq l' \Rightarrow \forall x \in \mathbb{Z}^+. sum(x :: l') \geq x :: l' $	\Rightarrow -intro
10. $\forall l' \in \text{List } \mathbb{Z}^+. sum(l') \geq l' \Rightarrow \forall x \in \mathbb{Z}^+. sum(x :: l') \geq x :: l' $	\forall -intro (3) (9)
11. $\forall l \in \text{List } \mathbb{Z}^+. sum(l) \geq l $	List-elim (2) (10)

STRUKTURELLE INDUKTION AUF LISTEN

- *Die Summe einer Liste l von positiven ganzen Zahlen ist mindestens so groß wie ihre Länge*

Induktionsanfang $l = []$: *Die Summe und die Länge von l sind 0*

Induktionsannahme: *Es gelte $sum(l') \geq |l'|$*

Induktionsschritt: *Sei $x \in \mathbb{Z}^+$ und $l = x :: l'$*

$$\text{Dann gilt } sum(l) = sum(l') + x \geq sum(l') + 1 \geq |l'| + 1 = |l| \quad (\text{IA})$$

Aufgrund des Induktionsprinzips folgt die Behauptung für alle Listen

- **Formale Beweisskizze für $\forall l \in \text{List } \mathbb{Z}^+. sum(l) \geq |l|$**

Aussage	Begründung
1. $0 \geq 0$	arithmetic
2. $sum([]) \geq [] $	fold $sum, length$ (1)
3. $l' : \text{List } \mathbb{Z}^+$	Neue Variable
4. $sum(l') \geq l' $	“Induktionsannahme”
5.	
6. $x : \mathbb{Z}^+$	Neue Variable
7. $sum(x :: l') \geq x :: l' $	<i>zu zeigen</i>
8. $\forall x \in \mathbb{Z}^+. sum(x :: l') \geq x :: l' $	\forall -intro (6) (7)
9. $sum(l') \geq l' \Rightarrow \forall x \in \mathbb{Z}^+. sum(x :: l') \geq x :: l' $	\Rightarrow -intro
10. $\forall l' \in \text{List } \mathbb{Z}^+. sum(l') \geq l' \Rightarrow \forall x \in \mathbb{Z}^+. sum(x :: l') \geq x :: l' $	\forall -intro (3) (9)
11. $\forall l \in \text{List } \mathbb{Z}^+. sum(l) \geq l $	List-elim (2) (10)

STRUKTURELLE INDUKTION AUF LISTEN

- *Die Summe einer Liste l von positiven ganzen Zahlen ist mindestens so groß wie ihre Länge*

Induktionsanfang $l = []$: *Die Summe und die Länge von l sind 0*

Induktionsannahme: *Es gelte $sum(l') \geq |l'|$*

Induktionsschritt: *Sei $x \in \mathbb{Z}^+$ und $l = x :: l'$*

$$\text{Dann gilt } sum(l) = sum(l') + x \geq sum(l') + 1 \geq |l'| + 1 = |l| \quad (\text{IA})$$

Aufgrund des Induktionsprinzips folgt die Behauptung für alle Listen

- **Formale Beweisskizze für $\forall l \in \text{List } \mathbb{Z}^+. sum(l) \geq |l|$**

Aussage	Begründung
1. $0 \geq 0$	arithmetic
2. $sum([]) \geq [] $	fold $sum, length$ (1)
3. $l' : \text{List } \mathbb{Z}^+$	Neue Variable
4. $sum(l') \geq l' $	“Induktionsannahme”
5. $sum(l') + 1 \geq l' + 1$	<i>zu zeigen</i>
6. $x : \mathbb{Z}^+$	Neue Variable
7. $sum(x :: l') \geq x :: l' $	fold $sum, length$ (5)
8. $\forall x \in \mathbb{Z}^+. sum(x :: l') \geq x :: l' $	\forall -intro (6) (7)
9. $sum(l') \geq l' \Rightarrow \forall x \in \mathbb{Z}^+. sum(x :: l') \geq x :: l' $	\Rightarrow -intro
10. $\forall l' \in \text{List } \mathbb{Z}^+. sum(l') \geq l' \Rightarrow \forall x \in \mathbb{Z}^+. sum(x :: l') \geq x :: l' $	\forall -intro (3) (9)
11. $\forall l \in \text{List } \mathbb{Z}^+. sum(l) \geq l $	List-elim (2) (10)

STRUKTURELLE INDUKTION AUF LISTEN

- *Die Summe einer Liste l von positiven ganzen Zahlen ist mindestens so groß wie ihre Länge*

Induktionsanfang $l = []$: *Die Summe und die Länge von l sind 0*

Induktionsannahme: *Es gelte $sum(l') \geq |l'|$*

Induktionsschritt: *Sei $x \in \mathbb{Z}^+$ und $l = x :: l'$*

$$\text{Dann gilt } sum(l) = sum(l') + x \geq sum(l') + 1 \geq |l'| + 1 = |l| \quad (\text{IA})$$

Aufgrund des Induktionsprinzips folgt die Behauptung für alle Listen

- **Formale Beweisskizze für $\forall l \in \text{List } \mathbb{Z}^+. sum(l) \geq |l|$**

Aussage	Begründung
1. $0 \geq 0$	arithmetic
2. $sum([]) \geq [] $	fold $sum, length$ (1)
3. $l' : \text{List } \mathbb{Z}^+$	Neue Variable
4. $sum(l') \geq l' $	“Induktionsannahme”
5. $sum(l') + 1 \geq l' + 1$	arithmetic (4)
6. $x : \mathbb{Z}^+$	Neue Variable
7. $sum(x :: l') \geq x :: l' $	fold $sum, length$ (5)
8. $\forall x \in \mathbb{Z}^+. sum(x :: l') \geq x :: l' $	\forall -intro (6) (7)
9. $sum(l') \geq l' \Rightarrow \forall x \in \mathbb{Z}^+. sum(x :: l') \geq x :: l' $	\Rightarrow -intro
10. $\forall l' \in \text{List } \mathbb{Z}^+. sum(l') \geq l' \Rightarrow \forall x \in \mathbb{Z}^+. sum(x :: l') \geq x :: l' $	\forall -intro (3) (9)
11. $\forall l \in \text{List } \mathbb{Z}^+. sum(l) \geq l $	List-elim (2) (10)

STRUKTURELLE INDUKTION AUF STRINGS

- *Für alle Strings $u, v \in \Sigma^*$ gilt $|uv| = |u| + |v|$*

Die Länge der Konkatination uv zweier Strings u und v ist die Summe der Einzellängen $|u|$ und $|v|$

Wir führen eine strukturelle Induktion über den zweiten String v

Behauptung $B(v)$: *Für alle $u \in \Sigma^*$ gilt $|uv| = |u| + |v|$*

Induktionsanfang $v = \epsilon$: *Es ist $|uv| = |u\epsilon| = |u| = |u| + 0 = |u| + |\epsilon| = |u| + |v|$*

also gilt $(B(\epsilon))$

Induktionsannahme: *Es gelte $B(v')$ für ein $v' \in \Sigma^*$*

Induktionsschritt: *Es sei $a \in \Sigma$ beliebig und $v = v'a$.*

Dann ist $|uv| = |uv'a| = |uv'| + 1 \stackrel{IA}{=} |u| + |v'| + 1 = |u| + |v'a| = |u| + |v|$

also gilt $(B(v))$

Wegen des Induktionsprinzips folgt die Behauptung $B(v)$ für alle $v \in \Sigma^*$

- **Binärbäume haben zwei Nachfolger pro Knoten**

- $x \in T \Rightarrow x \in \text{BTree } T$

- $x \in T \wedge t_1, t_2 \in \text{BTree } T \Rightarrow (x, [t_1, t_2]) \in \text{BTree } T$

- Anzahl der Knoten: $\#(x) = 1$, $\#(x, [t_1, t_2]) = \#(t_1) + \#(t_2) + 1$

- Tiefe eines Baums: $d(x) = 1$, $d(x, [t_1, t_2]) = \max(d(t_1), d(t_2)) + 1$

STRUKTURELLE INDUKTION AUF BINÄRBÄUMEN

- **Binärbäume haben zwei Nachfolger pro Knoten**

- $x \in T \Rightarrow x \in \text{BTree } T$

- $x \in T \wedge t_1, t_2 \in \text{BTree } T \Rightarrow (x, [t_1, t_2]) \in \text{BTree } T$

- Anzahl der Knoten: $\#(x) = 1$, $\#(x, [t_1, t_2]) = \#(t_1) + \#(t_2) + 1$

- Tiefe eines Baums: $d(x) = 1$, $d(x, [t_1, t_2]) = \max(d(t_1), d(t_2)) + 1$

- **Für jeden Binärbaum t gilt $\#(t) \leq 2^{d(t)} - 1$**

Induktionsanfang $t = x$: *Tiefe und Anzahl der Knoten ist $1 \leq 2^1 - 1$*

Induktionsannahme: *Es gelte $\#(t_1) \leq 2^{d(t_1)} - 1$ und $\#(t_2) \leq 2^{d(t_2)} - 1$*

Induktionsschritt: *Sei $t = (x, [t_1, t_2])$. Dann gilt*

$$\#(t) = \#(t_1) + \#(t_2) + 1$$

$$\leq (2^{d(t_1)} - 1) + (2^{d(t_2)} - 1) + 1 \quad (\text{IA})$$

$$\leq (2^{\max(d(t_1), d(t_2))} - 1) + (2^{\max(d(t_1), d(t_2))} - 1) + 1 \quad d(t_1) \leq \max(d(t_1), d(t_2))$$

$$= 2 * 2^{\max(d(t_1), d(t_2))} - 1 = 2^{d(t)} - 1$$

Wegen des Induktionsprinzips folgt die Behauptung für alle Binärbäume

WIE GENAU/FORMAL MUSS EIN BEWEIS SEIN?

- **“Ein Beweis ist ein Argument, das den Leser überzeugt”**

Beweisidee muß klar erkennbar und einsichtig sein

Beweis muß **genau** genug, um Details rekonstruieren zu können

Formal und detailliert ist nicht immer besser (vgl. Briefmarkenproblem)

- Text muß **präzise Sprache** verwenden, **lesbar** und **klar verständlich** sein

Formeln / Textfragmente ohne erkennbaren Sinn aneinanderzureihen ist unakzeptabel

Zwischenschritte müssen **mit “üblichen” Vorkenntnissen erklärbar** sein

- Gedankensprünge sind erlaubt, wenn Sie die Materie gut genug verstehen, dass Sie **nichts mehr falsch machen können**

... **es reicht nicht, dass Sie es einmal richtig gemacht haben**

- Tip: **ausführliche Lösungen entwickeln**, bis Sie genug Erfahrung haben.

Bei Präsentation für Andere: **zentrale Gedanken** aus Lösung **extrahieren**

- Test: **verstehen Kommilitonen Ihre Lösung** und **warum** sie funktioniert?

Mehr dazu in den Übungen

- **Klärung der Voraussetzungen**

- Welche **Begriffe** sind zum Verständnis des Problems erforderlich?
- Erstellung eines **präzisen Modells**: abstrahiere von Details
- **Formulierung des Problems im Modell**: was genau ist zu tun?

- **Lösungsweg konkretisieren**

- Welche **Einzelschritte** benötigt man, um das Problem zu lösen?
- Welches **Gesamtergebnis** ergibt sich aus den Einzelschritten?
- Wie **beweist** man die Korrektheit des Gesamtergebnisses?

- **Lösung zusammenfassen**

- **Kurz und prägnant**: Argumente auf das Wesentliche beschränken
- Wo möglich **mathematisch präzise Formulierungen** verwenden

ANHANG

• Notation für logische Operationen

- P, Q, R oder $P(x), Q(x)$ sind Platzhalter für elementare Aussagen
- Aussagenlogische Formeln: $\neg P, P \wedge Q, P \vee Q, P \Rightarrow Q, P \Leftrightarrow Q$
- Quantoren: $\forall x.P(x), \exists x.P(x)$

• Notation für mengentheoretische Konzepte

- S, M sind typische Platzhalter für Mengen
- Mengenzugehörigkeit: $x \in S$
- Beschreibung einer Menge durch Eigenschaften: $\{x \mid P(x)\}$
- Konstruktion durch Eigenschaften und Funktionen: $\{f(x) \mid P(x)\}$
... mit Auswahl aus einer Menge: $\{x \in S \mid P(x)\}, \{f(x) \mid x \in S\}, \dots$
- Quantifizierung über Elemente einer Menge: $\forall x \in S.P(x), \exists x \in S.P(x)$
- Vereinigung: $S_1 \cup S_2 = \{x \mid x \in S_1 \vee x \in S_2\}$
- Durchschnitt: $S_1 \cap S_2 = \{x \mid x \in S_1 \wedge x \in S_2\}$
- Differenz: $S_1 - S_2 = \{x \mid x \in S_1 \wedge x \notin S_2\}$
- Komplement: $\overline{S} = \{x \mid x \notin S\}$
- Teilmenge $S_1 \subseteq S_2$, echte Teilmenge $S_1 \subset S_2$, Obermenge $S_1 \supseteq S_2, \dots$

MATHEMATISCHES VOKABULAR: FUNKTIONEN

- **Funktion** $f : S \rightarrow S'$: Abbildung zwischen Grundmengen S und S'
nicht unbedingt auf allen Elementen von S definiert (\perp = undefiniert)
- **Domain von f** : $domain(f) = \{x \in S \mid f(x) \text{ definiert}\}$ (Definitionsbereich)
- **Range von f** : $range(f) = \{y \in S' \mid \exists x \in S. f(x) = y\}$ (Wertebereich)
- **f total**: $domain(f) = S$ (andernfalls ist f **partiell**)
- **f injektiv**: $x \neq y \Rightarrow f(x) \neq f(y)$
- **f surjektiv**: $range(f) = S'$
- **f bijektiv**: f injektiv und surjektiv
- **Umkehrfunktion** $f^{-1}: S' \rightarrow S$: $f^{-1}(y) = x \Leftrightarrow f(x) = y$ (f muß injektiv sein)
- **Urbild** $f^{-1}(M) = \{x \in S \mid f(x) \in M\}$ (eine Menge, f muß nicht injektiv sein)
- **Charakteristische Funktion** χ_M von $M \subseteq S$:
$$\chi_M(x) = \begin{cases} 1 & \text{falls } x \in M, \\ 0 & \text{sonst} \end{cases}$$
- **Partiell-charakteristische Funktion** ψ_M :
$$\psi_M(x) = \begin{cases} 1 & \text{falls } x \in M, \\ \perp & \text{sonst} \end{cases}$$

Mehr Vokabular wird bei Bedarf vorgestellt

List T : Datenstruktur mit Elementen aus T

- $[x_1, \dots, x_n]$: geordnete Folge mit Elementen $x_1, \dots, x_n \in T$
- $[]$: leere Liste (ohne Elemente)
- $x :: l$: Voranstellen eines Elements aus T vor Liste l
- $hd(l)$: Kopf (erstes Element) der Liste l $(hd(x :: l) = x)$
- $tl(l)$: Rest der Liste l (nach Entfernen des Kopfes) $(tl(x :: l) = l)$
- $rev(l)$: Umkehrung der Liste l $(rev([x_1, \dots, x_n]) = [x_n, \dots, x_1])$
- $l_1 \circ l_2$: Konkatenation zweier Listen $([x_1, \dots, x_n] \circ [y_1, \dots, y_m] = [x_1, \dots, x_n, y_1, \dots, y_m])$
- $|l|$: Länge der Liste l $(|[x_1, \dots, x_n]| = n)$
- $map(f, l)$: Anwendung von f auf alle Listenelemente $(map(f, [x_1, \dots, x_n]) = [f(x_1), \dots, f(x_n)])$
- $l[n]$ (oder l_n): n -tes Element von l $([x_1, \dots, x_n][m] = x_m)$

Viele weitere Konzepte definierbar

- **Alphabet** Σ : endliche Menge von Symbolen,
z.B. $\Sigma = \{0, 1\}$, $\Sigma = \{0, \dots, 9\}$, $\Sigma = \{A, \dots, Z, a, \dots, z, \ , ?, !, \dots\}$
- **Wort** (String, Zeichenreihe): endliche Folge w von Symbolen eines Alphabets
- Σ^* : Menge aller Wörter über Σ
- Σ^+ : Menge aller nichtleeren Wörter über Σ
- Σ^k : Menge der Wörter der Länge k mit Symbolen aus Σ
- ϵ : **Leeres Wort** (ohne jedes Symbol)
- $w a$: **Anhängen** eines Symbols a an das Wort w
- $w v$: **Konkatenation** (Aneinanderhängung) der Wörter w und v
- u^i : i -fache Konkatenation des Wortes (oder Symbols) u
- $|w|$: **Länge** des Wortes w (Anzahl der Symbole)
- $v \sqsubseteq w$: v **Präfix** von w , wenn $w = v u$ für ein Wort u
- **Sprache** L : Beliebige Menge von Wörtern über einem Alphabet Σ
Üblicherweise in abstrakter Mengennotation gegeben
z.B. $\{w \in \{0, 1\}^* \mid |w| \text{ ist gerade}\}$ $\{0^n 1^n \mid n \in \mathbb{N}\}$
- Eine Sprachen kann auch als **Problem** P bezeichnet werden
(Das “Problem” ist in diesem Fall, die Zugehörigkeit zur Menge P zu testen)

● **Gerichteter Graph: Datenstruktur** $G = (V, E)$

- V endliche Menge von **Knoten**, $E \subseteq \{ (v, v') \in V \times V \mid v \neq v' \}$ (“**Kanten**”)
- **Nachfolger** von v : durch Kanten verbundene Knoten aus $\{v' \in V \mid (v, v') \in E\}$
- **Nachkommen**: transitive Hülle der Nachfolgerrelation
- **Grad** eines Knotens: Anzahl der direkten Nachfolger
- **Pfad** (von v_1 nach v_n): Folge $[v_1, \dots, v_n]$ durch Kanten verbundener Knoten
- v' **erreichbar** von v : es gibt einen Pfad von u nach v
- **Kreis**: Pfad mit identischen Start- und Endknoten (alle anderen sind verschieden)
- H **Subgraph** von G ($H \sqsubseteq G$): Knoten und Kanten von H gehören zu G
- H **isomorph** zu G ($H \cong G$): Umbenennung der Knoten von H erzeugt G

Konzepte analog für ungerichtete Graphen (Kanten haben keine Richtung)

● **Lohnenswerte Literatur**

- S. Krumke, H. Noltemeier: *Graphentheoretische Konzepte und Algorithmen*, Teubner 2005.
- C. Meinel, M. Mundhenk: *Mathematische Grundlagen der Informatik*, Teubner 2002.
- K. Denecke: *Algebra und Diskrete Mathematik für Informatiker*, Teubner 2003.

- **Tree T : Datenstruktur mit Elementen aus T**
 - x : Einzelknoten mit Wurzel $x \in T$
 - $(x, [t_1, \dots, t_n])$: Baum mit **Wurzel** x und **Unterbäumen** $t_1 \dots t_n$
- **Darstellbar als kreisfreier Graph mit markierten Knoten**
 - Elemente von T sind Knoten des Baums
 - Kanten verbinden Knoten mit Wurzeln der Unterbäume
 - “**Nachfolger**” eines Knotens sind geordnet
 - **Wurzel**: Knoten ohne Vorgänger
 - **Blatt / Innerer Knoten**: Knoten ohne/mit Nachfolger
 - **Binärbaum**: Baum, in dem jeder innere Knoten Grad 2 hat
 - **Tiefe**: Länge des längsten Pfades im Baum
- **Wichtige Eigenschaften**
 - Von der Wurzel gibt es zu jedem Knoten genau einen Pfad
 - Ein Baum mit n Knoten hat $n-1$ Kanten
 - Ein Binärbaum mit n Knoten hat mindestens Tiefe $\lceil \log_2(n+1) \rceil$

