Vorlesung Informatik 1 (Wintersemester 2020/2021)

Kapitel 10: Mathematische Konzepte in der Informatik

Martin Frieb
Johannes Metzger

Universität Augsburg
Fakultät für Angewandte Informatik

11. Januar 2021

Wozu braucht man Mathematik in der Informatik?

Die Informatik setzt sich aus vielen Gebieten zusammen, die Antworten auf verschiedene Fragen suchen, z.B.:

- **Rechnerarchitektur**: Wie kann man Rechner bauen?
- **Programmiersprachen und Compilerbau**: Wie kann man Rechner programmieren?
- **Software Engineering**: Wie kann man große und sehr komplexe Programme erstellen?
- **Künstliche Intelligenz**: Wie kann man Programme zu Problemen erstellen, die wir nicht gut verstehen?
- **Soziale Informatik**: Was muss man beachten, damit Rechner und Programme im Dienste des Menschen eingesetzt werden?
- **Sicherheitskritische Systeme**: Wie erstellt man Systeme aus Hard- und Software, von deren fehlerfreier Funktion Menschenleben abhängen?
- **Algorithmen und Datenstrukturen**: Wie erstellt und analysiert man besonders schnelle und speichereffiziente Programme?

Wozu braucht man Mathematik in der Informatik?

Das Ziel der Informatik ist, die in den verschiedenen Gebieten der Informatik betrachteten Fragestellungen und deren zugrundeliegenden Gegebenheiten der realen Welt so präzise zu modellieren, damit

- man mathematisch gesicherte Aussagen darüber treffen kann,
- und Rechner selbständig damit umgehen können.

Ein wichtiges Hilfsmittel dabei ist die Abstraktion, also das Weglassen von Details, die für die jeweils betrachtete Fragestellung unwichtig sind.

Zusammengefasst gründet die Informatik darauf, zu einer betrachteten Fragestellung ein passendes mathematisches Modell zu finden, mit dessen Hilfe die Fragestellung analysiert und rechnergestützt gelöst werden kann

Wozu braucht man Mathematik in der Informatik?

Mathematik ist also die Grundlage der Informatik.

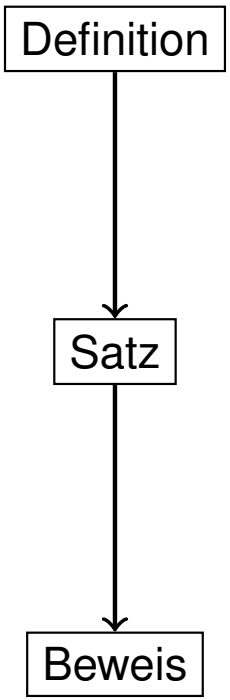
Aufgaben eines Informatikers

- Zu einer Fragestellung ein passendes bereits existierendes mathematisches Modell finden und anpassen, oder ein neues entwickeln
- Mathematische Modelle analysieren
- Mathematische Modelle implementieren

In Informatik-Vorlesungen lernen Sie dazu neben Programmiertechniken die für die Informatik grundlegenden mathematischen Strukturen und Methoden kennen und üben logisches Schließen und präzises und verständliches Formulieren

Wozu braucht man Mathematik in der Informatik?

Vorgehen in der Mathematik:



Anwendung in der Informatik:

- **Definiere** neue mathematische Objekte zur Modellierung einer praktischen Fragestellung
  Beispiel: Terme und Syntaxbäume
  Anwendung: Auswertung von Ausdrücken in Programmen
- Ein **mathematischer Satz** formuliert nützliche Aussagen über die definierten Objekte
- Ein **mathematischer Beweis** ist eine logische Herleitung der Richtigkeit (oder Unrichtigkeit) einer Aussage aus Definitionen und anderen (bereits bewiesenen) Aussagen

Operationen

Definition 10.1 (Operation)

Eine n **- stellige Operation** ist eine Abbildung

$$op : T_1 \times \ldots \times T_n \to T.$$

- Sie ordnet jedem Element $(t_1, \ldots, t_n) \in T_1 \times \ldots \times T_n$ durch eine **Rechenvorschrift** einen **Ergebniswert** $op(t_1, \ldots, t_n) \in T$ zu.
- n ist die **Stelligkeit** von op
- $t_1, \ldots, t_n$ heißen **Operanden** oder **Argumente**. Die Stelligkeit ist die Anzahl der Operanden.
- $T_1, ..., T_n$ heißen **Parametertypen** von op.
- T ist der **Resultattyp** von op.

Operationen

Unäre Operationen

1-stellige Operationen heißen **unär**

- Negatives Vorzeichen: $-: \mathbb{R} \to \mathbb{R}$
- Positives Vorzeichen: $+: \mathbb{R} \to \mathbb{R}$
- Abrundung: $\lfloor \cdot \rfloor : \mathbb{R} \to \mathbb{Z}$
- Aufrundung: $\lceil \cdot \rceil : \mathbb{R} \to \mathbb{Z}$
- Absolutbetrag: $|\cdot| : \mathbb{R} \to [0, \infty[$
- Wortlänge: $|\cdot| : A^* \to \mathbb{N}_0$
- Mächtigkeit von Mengen: $|\cdot| : P(A) \to \mathbb{N}_0$

Binäre Operationen

2-stellige Operationen heißen **binär**.

- Addition: $+ : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$
- Subtraktion: $- : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$
- Multiplikation: $\cdot : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$
- Division: $/ : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$
- Modulo: $\mathrm{mod} : \mathbb{Z} \times \mathbb{Z} \to \mathbb{Z}$
- Ganzzahlige Division: $\div : \mathbb{Z} \times \mathbb{Z} \to \mathbb{Z}$
- Mengenvereinigung: $\cup : P(A) \times P(A) \to P(A)$
- Mengendurchschnitt: $\cap : P(A) \times P(A) \to P(A)$
- Mengendifferenz: $\setminus : P(A) \times P(A) \to P(A)$

Terme: Grundelemente

Wir wollen nun formal mathematisch **Terme** zur Darstellung von Rechnungen definieren.

Dazu geben wir uns eine Menge von **Operationen**, eine Menge von **Funktionen** und eine Menge von **Variablen** vor.

- Ein Element eines Parameter- oder Resultattyps T nennen wir **Konstante**. Wir sagen, eine Konstante $t \in T$ ist **vom Typ** T.
- Eine Variable hat einen Typ T und dient als Platzhalter für Konstanten vom Typ T.

Anmerkung

Das entspricht dem typischen Vorgehen in der Mathematik:

- Man setzt einige Begriffe und mathematische Objekte als gegeben voraus.
- Darauf aufbauend konstruiert (definiert) man neue Begriffe und mathematische Objekte.

Auf diese Weise erstellt man Schritt für Schritt ein immer komplexeres mathematisches Modell.

Terme: Bildungsregeln

Definition 10.2 (Term)

1. Jede **Variable** oder **Konstante** eines Typs T ist ein **Term vom Typ** T.
2. Ist $op : T_1 \times \ldots \times T_n \to T$ eine Operation und A_i jeweils ein Term vom Typ T_i für $i \in \{1, ..., n\}$, so ist

$$op(A_1, \ldots, A_n)$$

   ein **Term vom Typ** T.
3. Ist $f : S \to T$ eine Funktion und A ein Term vom Typ S, so ist

$$f(A)$$

   ein **Term vom Typ** T.

Terme: Bildungsregeln

Beispiel 10.3

Mit obiger Definition können wir unter Benutzung der eingeführten Funktionen und arithmetischen Rechenoperationen der Reihe nach folgende Terme bilden (x und y sind hier Variablen vom Typ $\mathbb{Z}$):

- x, y, 5
- $-(x)$
- $+(-(x), 5)$
- $\cdot(y, y)$
- $\log(\cdot(y, y))$
- $\div(+(-(x), 5), \log(\cdot(y, y)))$

Strukturelle Induktion

Definition mit struktureller Induktion

Bei der Definition der **Menge der Terme** handelt es sich um eine sog. **strukturelle Induktion**:

- Zuerst werden **einfache Grundelemente** der Menge definiert (siehe vorige Definition, Unterpunkt 1)
- Dann werden **endliche viele Bildungsregeln** angegeben, mit denen aus existierenden Elementen der Menge zusätzliche Elemente konstruiert werden können (siehe vorige Definition, Unterpunkte 2 und 3)

Anwendung der Bildungsregeln

- Eine Bildungsregel dient als Regel zur Konstruktion eines neuen Elements aus bereits vorhandenen Elementen
- Nur solche Elemente gehören zur Menge, die sich aus Grundelementen durch (ggf. wiederholte) Anwendung der Bildungsregeln konstruieren lassen

Teilterme

Definition 10.4 (Teilterm)

- Jeder Term A ist ein **Teilterm** von sich selbst.
- Ist $op(A_1, \ldots, A_n)$ ein Term, so sind $A_1, \ldots, A_n$ und Teilterme von $A_1, \ldots, A_n$ **Teilterme** von $op(A_1, \ldots, A_n)$
- Ist $f(A)$ ein Term, so sind A und Teilterme von A **Teilterme** von $f(A)$

Beispiel 10.5

Der Term $+(-(x), 5)$ hat die folgenden Teilterme:

- $+(-(x), 5)$
- $-(x)$
- x
- 5

Präfixnotation

Präfixnotation (oder polnische Notation)

Ein Ausdruck in der **Präfixnotation** ist ein Ausdruck, in dem man

- $(op\ \ A_1\ \ A_2\ \ \ldots\ \ A_n)$ statt $op(A_1, \ldots, A_n)$ schreibt
- $(f\ \ A)$ statt $f(A)$ schreibt

Beispiel 10.6

- $-(5)$ schreibt man in der Präfixnotation $(-\ \ 5)$ (negatives Vorzeichen)
- $-(5,3)$ schreibt man in der Präfixnotation $(-\ \ 5\ \ 3)$ (Subtraktion)

(Verwenden wir nicht weiter, sieht man aber in manchen Programmiersprachen)

Infixnotation

Infixnotation

Ein Ausdruck in der **Infixnotation** ist ein Ausdruck, in dem man

- $(A\,op\,B)$ statt $op(A,B)$ schreibt für jede binäre Operation op.
- $(op\,A)$ statt $op(A)$ schreibt für jede unäre Operation op.

Beispiel 10.7

- $-(5)$ entspricht in der Infixnotation (-5)
- $\cdot(2,+(3,5))$ entspricht $(2\cdot(3+5))$
- $+(\cdot(2,3),5)$ entspricht $((2\cdot3)+5)$

(Das ist die übliche Schreibweise in mathematischen Formeln und die Schreibweise in C-Code)

Auswertung von Termen

Variablen-Belegung

Eine **Variablen-Belegung** ist eine Abbildung β, die jeder Variable x eines Terms vom Typ T eine Konstante $\beta(x) \in T$ zuordnet.

Definition 10.8 (Auswertung von Termen)

Sei β eine Variablen-Belegung. Wird definieren wie folgt induktiv den Wert A' eines Terms A bzgl. β:

- Der Wert k' einer Konstante k ist $k' := k$.
- Der Wert x' einer Variable x ist $x' := \beta(x)$.
- Der Wert $op(A_1, \ldots, A_n)'$ eines Terms $op(A_1, \ldots, A_n)$ ist $op(A_1, \ldots, A_n)' := op(A_1', \ldots, A_n')$
- Der Wert $f(A)'$ eines Terms $f(A)$ ist $f(A)' := f(A')$

Terme werden also von **innen nach außen** ausgewertet. Die Reihenfolge der Auswertung der Operanden einer Operation ist dabei nicht definiert.

Ein Beweisprinzip für induktiv definierte Mengen

Sei M eine induktiv definierte Menge und E eine Eigenschaft, für die wir beweisen wollen, dass sie für alle Elemente von M erfüllt ist.

Allgemeines Schema der **strukturellen Induktion**:

1. **Induktionsanfang**:
   Zu zeigen: E gilt für alle Grundelemente von M
2. **Induktionsschritt**:
   Zu zeigen: Angenommen E gilt die Elemente $m_1, \ldots, m_n$ (**Induktionsvoraussetzung**), so gilt E auch für jedes Element, das aus $m_1, \ldots, m_n$ mit einer der Bildungsregeln von M konstruiert wird.

Ein Beweisprinzip für induktiv definierte Mengen - Beispiel

Sei M die wie folgt definierte Menge:

1. 3 ist ein Element von M (d.h. das einzige Grundelement ist 3)
2. Ist x ein Element von M, dann ist auch $x+3$ ein Element von M (Bildungsregel für neue Elemente)

Wir wollen beweisen: **Jedes Element von M ist durch 3 teilbar**

Beweis mit struktureller Induktion

1. **Induktionsanfang**:
   Zu zeigen: 3 *ist durch* 3 *teilbar*
   Beweis: $(3 \bmod 3) = 0$
2. **Induktionsschritt**:
   Zu zeigen: Angenommen x ist durch 3 *teilbar (**Induktionsvoraussetzung**), dann ist auch $x+3$ durch* 3 *teilbar*
   Beweis:
   $(x+3) \bmod 3 = ((x \bmod 3)+(3 \bmod 3)) \bmod 3 = (0+0) \bmod 3 = 0$

Aussagen über Terme beweisen

Wir wollen beweisen:
Jeder Term A enthält eine Konstante oder eine Variable

Induktionsanfang (E gilt für die Grundelemente der Menge der Terme)

Das stimmt, denn jedes Grundelement ist entweder eine Variable oder eine Konstante

Induktionsschritt (Durch Anwendung einer Bildungsregel bleibt E erhalten)

Induktionsvoraussetzung: $A, A_1, \ldots, A_n$ sind Terme, die jeweils eine Variable oder eine Konstante enthalten
Zu zeigen: $op(A_1, \ldots, A_n)$ und $f(A)$ enthalten eine Variable oder eine Konstante
Beweis: Wenn jeder Term $A, A_1, \ldots, A_n$ eine Konstante oder Variable enthält, dann enthalten auch $op(A_1, \ldots, A_n)$ und $f(A)$ eine Variable oder Konstante, da $n \geq 1$ ist.

Aussagen über Terme beweisen

Wir wollen beweisen
Nicht jeder Term A enthält eine Operation

Beweis durch Angabe eines Gegenbeispiels

Jede Variable und jede Konstante ist ein Term, der keine Operation enthält

Möchte man zeigen, dass **nicht** alle Elemente einer Menge eine Eigenschaft E erfüllen, so reicht es ein **Gegenbeispiel** zu finden, d.h. ein Element aus der Menge zu finden, das E **nicht** erfüllt

Gerichtete Graphen

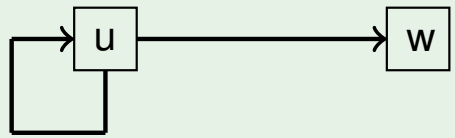
Definition 10.9 (Gerichteter Graph)

Ein **gerichteter Graph** ist ein Paar (V, E), wobei

- V eine endliche Menge von **Knoten** (engl. *vertices*) ist, und
- $E \subseteq V \times V$ eine Menge **gerichteter Kanten** (engl. *edges*) zwischen Knoten ist. Ein Element $(v, w) \in E$ bezeichnet eine **Kante von v nach w**.

Beispiel 10.10 (Graphische Darstellung)

$V = \{u, w\}$,
$E = \{(u, u), (u, w)\}$



Gerichtete Graphen

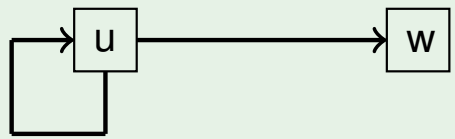
Definition 10.11 (Vorgänger und Nachfolger eines Knotens)

Sei (V, E) ein gerichteter Graph:

- Die **Vorgänger** eines Knotens $v \in V$ sind die Elemente der Menge $\{w \mid w \in V, (w, v) \in E\}$
- Die **Nachfolger** eines Knotens $v \in V$ sind die Elemente der Menge $\{w \mid w \in V, (v, w) \in E\}$

Beispiel 10.12

Vorgänger von u: $\{u\}$
Nachfolger u: $\{u, w\}$



Gerichtete Graphen

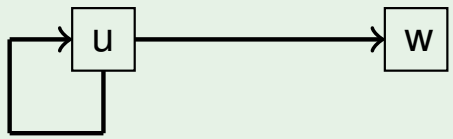
Definition 10.13 (azyklische gerichtete Graphen)

Sei (V, E) ein gerichteter Graph:

- Ein **Zyklus** ist eine Folge von Knoten $v_1, ..., v_n \in V$ mit $(v_1, v_2), ..., (v_{n-1}, v_n), (v_n, v_1) \in E$ und $n \in \mathbb{N}$.
- Ein gerichteter Graph heißt **azyklisch** falls er keine Zyklen enthält

Beispiel 10.14 (Graphen)

Der Graph enthält den Zyklus u



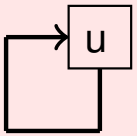
Gerichtete Bäume

Definition 10.15 (Baum)

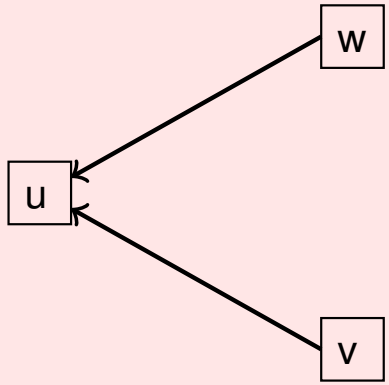
Ein **gerichteter Baum** ist ein gerichteter Graph (V, E) mit folgenden Eigenschaften:

1. der Graph ist azyklisch
2. es gibt genau einen Knoten ohne Vorgänger, genannt **Wurzel**
3. jeder Knoten hat höchstens einen Vorgänger

Das sind keine Bäume:



Eigenschaften (1) und (2) sind nicht erfüllt



Eigenschaften (2) und (3) sind nicht erfüllt

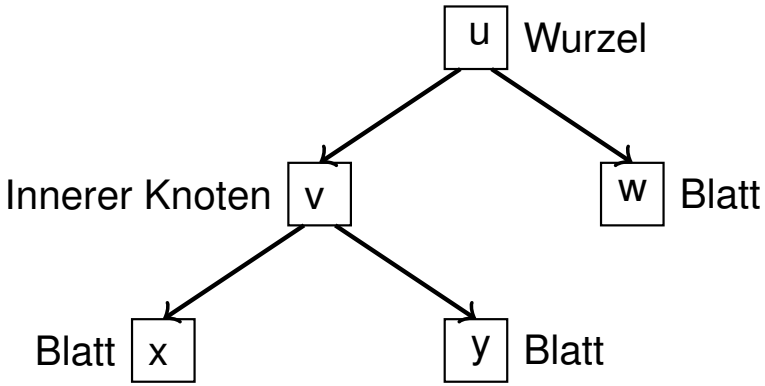
Gerichtete Bäume

Gerichtete Bäume zeichnet man, wie Stammbäume, oft von der Wurzel aus **von oben nach unten**, weil man nicht weiß, wie groß sie werden.

Definition 10.16 (Vater, Kind, Blatt)

Sei (V, E) ein gerichteter Baum und $v, w \in V$:

- w ist **Vater** von v, falls $(w, v) \in E$
- w ist **Kind** von v, falls $(v, w) \in E$
- w ist ein **Blatt**, falls w keine Kinder hat
- w ist ein **innerer Knoten**, falls w weder ein Blatt noch die Wurzel ist



- u ist Vater von v und w
- v und w sind Kinder von u
- v ist Vater von x und y
- x und y sind Kinder von v

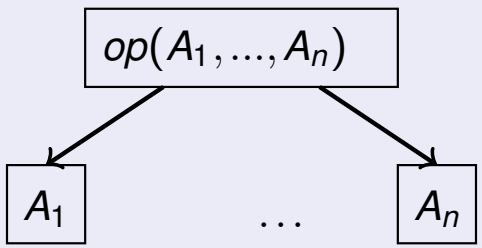
Was ist ein Syntaxbaum?

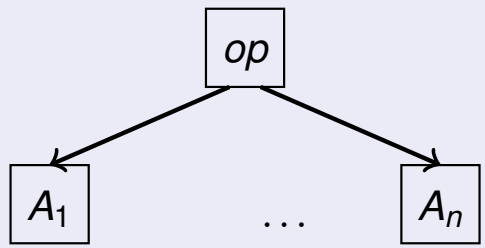
Definition 10.17 (Syntaxbaum)

Der **Syntaxbaum eines Terms** A ist der wie folgt definierte gerichtete Baum (V_A, E_A):

- V_A ist die Menge der Teilterme von A, wobei A die Wurzel ist
- Die Teilterme, die Variablen oder Konstanten entsprechen, sind die Blätter des Baumes
- Teilterme der Form $op(A_1, \ldots, A_n)$ haben die Kinder $A_1, \ldots, A_n$, also $(op(A_1, \ldots, A_n), A_i) \in E_A$ für jedes i.
- Teilterme der Form $f(A)$ haben ein Kind A, also $(f(A), A) \in E_A$.

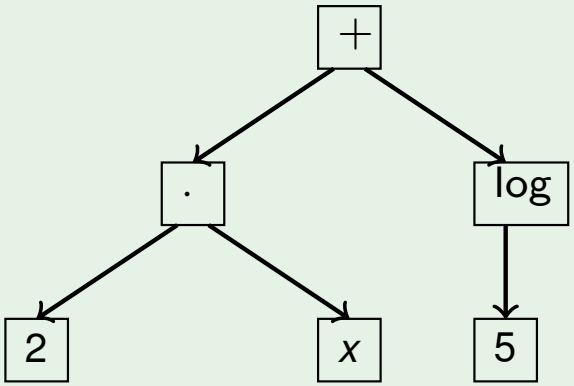
Oft schreibt man kurz nur op statt $op(A_1, \ldots, A_n)$ und f statt $f(A)$





Was ist ein Syntaxbaum?

Beispiel 10.18 (Syntaxbaum des Terms $((2 \cdot x) + \log(5))$)



- Klare Darstellung von Aufbau und Struktur eines Terms
- Klare Darstellung der Auswertungsreihenfolge in einem Term: Kinder müssen offenbar vor deren Vätern ausgewertet werden

Definition von aussagenlogischen Formeln

- Wir wollen nun formal mathematisch **aussagenlogischen Formeln** zur Darstellung von Aussagen, die erfüllt oder nicht erfüllt sein können, definieren.
- Dazu geben wir uns eine Menge von **Operationen** mit Parameter- und Resultattypen und eine Menge von **Variablen** zur Bildung von Termen vor.
- Darüber hinaus benötigen wir eine Menge von **Relationen** zur Bildung elementarer Aussagen.

Definition 10.19 (n-stellige Relation)

Seien $n \in \mathbb{N}$ und $T_1, \ldots, T_n$ Mengen.

- Die Menge

$$T_1 \times \ldots \times T_n := \{(t_1, \ldots, t_n) \mid t_i \in T_i, i \in \mathbb{N}, 1 \leq i \leq n\}$$

  heißt **kartesisches Produkt der Mengen** $T_1, \ldots, T_n$. Dessen Elemente heißen n**-Tupel**. t_i ist die i**-te Komponente** von $(t_1, \ldots, t_n)$.
- Eine n**-stellige Relation auf** $T_1 \times \ldots \times T_n$ ist eine Teilmenge $R \subseteq T_1 \times \ldots \times T_n$.

Relationen

Beispiel 10.20 (1-stellige Relationen)

- Eine 1-stellige Relation ist eine Menge einfacher Elemente ohne Komponenten.
- Beispiele für 1-stellige Relationen:
  $\mathbb{N}$, $\mathbb{Z}$, $[0,1]$, ...

Beispiel 10.21 (2-stellige Relationen)

- Eine 2-stellige Relation ist eine Menge von Elementen mit 2 Komponenten. Solche Relationen heißen auch **binär** und haben wir schon besprochen.
- Beispiele für 2-stellige Relationen auf $\mathbb{R} \times \mathbb{R}$:
  $<$, $\leq$, $>$, $\geq$, $=$, $\neq$, ...
- Beispiele für 2-stellige Relationen auf $P(A) \times P(A)$:
  $\subset$, $\subseteq$, $\supset$, $\supseteq$, $=$, $\neq$, ...

Aussagenlogische Formeln

Definition 10.22 (Aussagenlogische Formel)

Wir definieren mit struktureller Induktion:

- Seien $R \subseteq T_1 \times \ldots \times T_n$ eine n-stellige Relation und A_i ein Term vom Typ T_i für $i \in \mathbb{N}$, $1 \leq i \leq n$. Dann ist $(A_1, \ldots, A_n) \in R$ ist eine **aussagenlogische Formel**
- Sind $\mathtt{p}$ und $\mathtt{q}$ aussagenlogische Formeln, so lassen sich folgende weitere **aussagenlogische Formeln** bilden:
  - (Logisches Und) $p \wedge q$
  - (Logisches Oder) $p \vee q$
  - (Logisches Nicht) $\neg p$
  - (Implikation) $p \Rightarrow q$
  - (Äquivalenz) $p \Leftrightarrow q$

Erfüllung aussagenlogischer Formeln

Ob eine Formel erfüllt ist oder nicht, hängt von der Belegung der Variablen in den beteiligten Termen ab.

Definition 10.23 (Erfüllung)

Seien $R \subseteq T_1 \times \ldots \times T_n$ eine n-stellige Relation und A_i ein Term vom Typ T_i für $i \in \mathbb{N}$, $1 \leq i \leq n$. Wir definieren mit struktureller Induktion für eine Belegung β der Variablen in den Termen:

- $(A_1, \ldots, A_n) \in R$ ist **erfüllt bzgl.** β, falls $(A'_1, \ldots, A'_n) \in R$
- $p \wedge q$ ist **erfüllt bzgl.** β, falls p **und** q erfüllt sind
- $p \vee q$ ist **erfüllt bzgl.** β, falls p **oder** q erfüllt ist
- $\neg p$ ist **erfüllt bzgl.** β, falls p **nicht** erfüllt ist
- $p \Rightarrow q$ ist **erfüllt bzgl.** β, falls $(\neg p) \vee q$ erfüllt ist (**wenn** p**, dann** q)
- $p \Leftrightarrow q$ ist **erfüllt bzgl.** β, falls $(p \Rightarrow q) \wedge (q \Rightarrow p)$ erfüllt ist (p **genau dann wenn** q)

Beispiele aussagenlogischer Formeln

Ob eine Formel erfüllt ist oder nicht, hängt von der Belegung der Variablen in den beteiligten Termen ab.

Beispiel 10.24

- A ist Teilmenge von B: $A \subseteq B$
- x ist eine positive ganze Zahl: $x > 0 \wedge x \in \mathbb{Z}$
- x hat maximal 3 Nachkommastellen: $x \cdot 1000 \in \mathbb{Z}$
- x und y haben einen Abstand von höchstens 0.5: $|x - y| \leq 0.5$
- x ist nicht durch y teilbar: $\neg(x \bmod y = 0)$

Beispiele aussagenlogischer Formeln

Beispiel 10.25 (Implikation)

Die Formel $p \Rightarrow q$ ist gleichwertig zu der Formel $\neg p \vee q$. Diese ist in folgenden Fällen erfüllt:

- p und q sind erfüllt
- p und q sind nicht erfüllt
- p ist nicht erfüllt und q ist erfüllt

Allen Fällen gemeinsam ist: **Wenn p erfüllt ist, dann ist auch q erfüllt.**
Wenn p nicht erfüllt ist, dann kann q erfüllt sein oder nicht. Beispiel:

- Teilbarkeit: $(x \bmod 4 = 0) \Rightarrow (x \bmod 2 = 0)$

Beispiel 10.26 (Äquivalenz)

Die Formel $p \Leftrightarrow q$ ist gleichwertig zu der Formel $(p \wedge q) \vee (\neg p \wedge \neg q)$. Beispiel:

- Gleichungen: $(x + 5 = 10) \Leftrightarrow (x = 10 - 5)$

Besondere ausagenlogische Formeln

Definition 10.27 (Tautologie)

Eine Formel p heißt **Tautologie**, wenn sie für **jede Belegung** der Variablen erfüllt ist.
Beispiele:

- $(x < 0) \vee (x \geq 0)$
- $(x \bmod 4 = 0) \Rightarrow (x \bmod 2 = 0)$
- $(x + 5 = 10) \Leftrightarrow (x = 10 - 5)$

Definition 10.28 (Erfüllbarkeit)

Eine Formel p heißt **nicht erfüllbar**, wenn sie für **keine Belegung** der Variablen erfüllt ist.
Beispiel:

- $(x < 0) \wedge (x \geq 0)$

Motivation

Ob eine aussagenlogische Formel p, die eine Variable x enthält, wie z.B. $x < 5$, erfüllt ist, hängt ab von einer Belegung der Variablen.

Mit Hilfe der Prädikatenlogik lassen sich Aussagen der folgenden Art für eine solche aussagenlogische Formel p treffen

- **Es gibt** eine Belegung von x, s.d. p erfüllt ist: $\exists x(p)$
- **Für alle** Belegungen von x ist p erfüllt: $\forall x(p)$

Beispiel 10.29

Für alle Werte von x, y gilt: Falls x kleiner ist als y, dann gibt es ein z das größer ist als x und kleiner als y
$\forall x, y((x < y) \Rightarrow \exists z((x < z) \wedge (z < y)))$

- Diese Aussage ist **erfüllt**, falls x, y, z reelle Zahlen sind.
- Diese Aussage ist **nicht erfüllt**, falls x, y, z ganze Zahlen sind.

Definition von prädikatenlogischen Formeln

Definition 10.30 (Prädikatenlogische Formel)

Wir definieren mit struktureller Induktion:

- Seien $R \subseteq T_1 \times \ldots \times T_n$ eine n-stellige Relation und A_i ein Term vom Typ T_i für $i \in \mathbb{N}$, $1 \leq i \leq n$. Dann ist $(A_1, \ldots, A_n) \in R$ eine **prädikatenlogische Formel**
- Sind $\mathtt{p}$ und $\mathtt{q}$ prädikatenlogische Formeln, und ist x eine **freie Variable in** p, so lassen sich folgende weitere **prädikatenlogische Formeln** bilden:
  - $p \wedge q$, $p \vee q$, $\neg p$, $p \Rightarrow q$, $p \Leftrightarrow q$
  - (Allquantor) $\forall x(p)$
    Lies: “Für alle x”
  - (Existenzquantor) $\exists x(p)$
    Lies: “Es gibt ein x”

  In den Formeln $\forall x(p)$ und $\exists x(p)$ ist die Variable x **gebunden**. Nicht gebundene Variablen heißen **frei**.

Erfüllung prädikatenlogischer Formeln

Ob eine prädikatenlogische Formel erfüllt ist oder nicht, hängt von der **Belegung der freien Variablen** in den beteiligten Termen ab.

Definition 10.31 (Erfüllung)

Wir definieren mit struktureller Induktion für eine Belegung β der **freien Variablen** in den Termen:

- $(A_1, \ldots, A_n) \in R$ ist **erfüllt bzgl.** β, falls $(A'_1, \ldots, A'_n) \in R$
- $p \wedge q$ ist **erfüllt bzgl.** β, falls p **und** q erfüllt sind
- $p \vee q$ ist **erfüllt bzgl.** β, falls p **oder** q erfüllt ist
- $\neg p$ ist **erfüllt bzgl.** β, falls p **nicht** erfüllt ist
- $p \Rightarrow q$ ist **erfüllt bzgl.** β, falls $(\neg p) \vee q$ erfüllt ist (**wenn** p**, dann** q)
- $p \Leftrightarrow q$ ist **erfüllt bzgl.** β, falls $(p \Rightarrow q) \wedge (q \Rightarrow p)$ erfüllt ist (p **genau dann wenn** q)
- $\forall x(p)$ **erfüllt bzgl.** β, wenn p für **alle** Belegungen von x erfüllt ist.
- $\exists x(p)$ **erfüllt bzgl.** β, wenn p für **eine** Belegung von x erfüllt ist

Die Rolle der Variablentypen

- Die Quantoren beziehen sich immer auf den Typ der Variable
- Der Wahrheitswert der Formel hängt von diesem Typ ab

Beispiel 10.32

$\forall x, y((x < y) \Rightarrow \exists z((x < z) \wedge (z < y)))$

- ist erfüllt, falls x, y, z den Typ $\mathbb{R}$ haben (Satz aus der Mathematik)
- ist nicht erfüllt, falls x, y, z den Typ $\mathbb{Z}$ haben (Beweis per Gegenbeispiel: für $x = 3$ und $y = 4$ gibt es keine solche ganze Zahl z)

Einschränkung des Typs

Man kann den Typ einer Variable x wie folgt auf einen Typ M einschränken:

- $\forall x((x \in M) \Rightarrow p)$: Hier wird nur für Werte von x aus M gefordert, dass p erfüllt ist.
  Abkürzende Schreibweise: $\forall x \in M(p)$
- $\exists x((x \in M) \wedge p)$: Hier wird gefordert, dass es einen Wert von x aus M gibt, für den p erfüllt ist.
  Abkürzende Schreibweise: $\exists x \in M(p)$

Beispiele für prädikatenlogische Formeln

Beispiel 10.33 (Sortierung)

- Natürlichsprachliche Formulierung:
  Die Folge $a_1, \ldots, a_n$ ist aufsteigend sortiert
- Formulierung als prädikatenlogische Formel:
  $\forall i \in \{1, 2, ..., n-1\}(a_i < a_{i+1})$
  (i ist vom Typ $\mathbb{N}$, der Typ von $a_1, \ldots, a_n$ ist hier offen gelassen)

Beispiel 10.34 (Primzahlen)

- Natürlichsprachliche Formulierung:
  Die Zahl p ist eine Primzahl
- Formulierung als prädikatenlogische Formel:
  $\neg(\exists k((k > 1) \wedge (k < p) \wedge (p \ mod \ k = 0))$
  Oder: $\forall k \in \{1, 2, ..., p\}((p \ mod \ k = 0) \Rightarrow (k = 1) \vee (k = p))$
  (k und p sind vom Typ $\mathbb{N}$)

Beispiele für prädikatenlogische Formeln

Beispiel 10.35 (Palindrome)

- Natürlichsprachliche Formulierung:
  Das Wort $a_1 \dots a_n \in A^+$ *ist ein Palindrom*
- Formulierung als prädikatenlogische Formel:
  $\forall i \in \{1, 2, \dots, n \div 2\}(a_i = a_{n+1-i})$
  (i ist vom Typ $\mathbb{N}$)

Beispiel 10.36 (Mit *a* beginnende Wörter)

- Natürlichsprachliche Formulierung:
  Das Wort $w \in A^+$ *beginnt mit dem Buchstaben* $a \in A$
- Formulierung als prädikatenlogische Formel:
  $\exists u \in A^*(w = au)$

Grundlegende Forderung an C-Funktionen

- Eine grundlegende Forderung an eine C-Funktion ist, dass sie eine vorgegebene Problemstellung löst
- In der Praxis ist die Problemstellung aber oft sehr vage
- Deshalb präzisiert und formalisiert man die Problemstellung vor der Implementierung der C-Funktion. Eine solche Präzisierung nennt man **Problemspezifikation**
- Die Problemspezifikation hilft dann oft auch bei der korrekten Implementierung

Löst eine C-Funktion die durch eine Problemspezifikation gegebene Problemstellung, so nennt man sie **korrekt**.

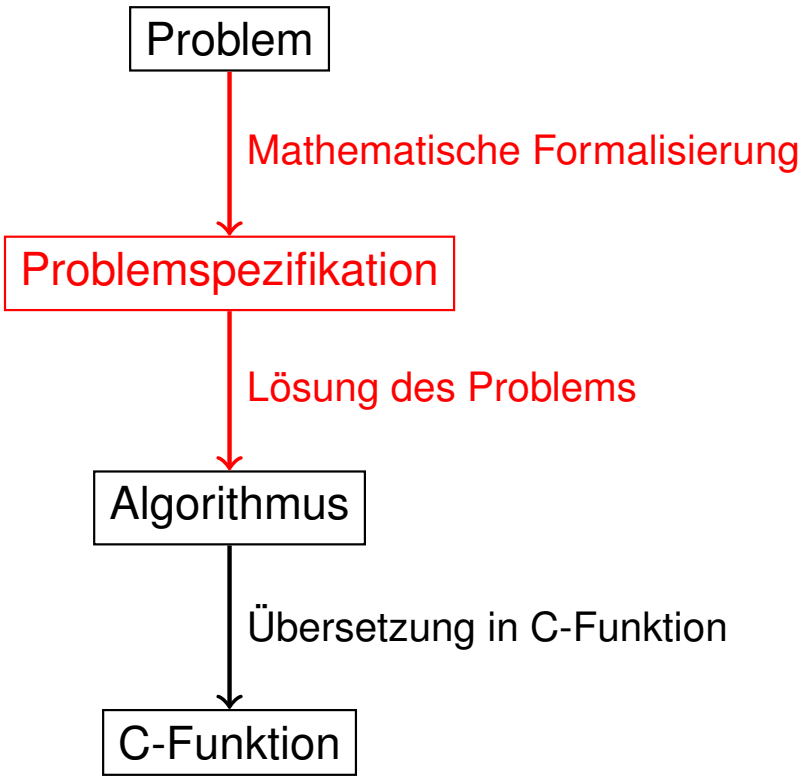
Definition 10.37 (Verifikationsverfahren)

Verifikationsverfahren dienen dem Nachweis der Korrektheit einer C-Funktion bzgl. einer Problemspezifikation.

Definition 10.38 (Validierungsverfahren)

Validierungsverfahren dienen dem Nachweis der Korrektheit einer Problemspezifikation bzgl. eines realen Problems.

Übersicht



Zur Lösung eines Problems entwirft man vor der Programmierung eine **mathematische Formulierung des Problems** und erstellt eine **programmiersprachen-unabhängige Lösung in Form eines sog. Algorithmus**:

- Zerlegung der Problemlösung in mehrere Schritte
- Präzisierung und Konkretisierung des Problems; Beseitigung von Widersprüchen und Unklarheiten

Was ist eine Problemspezifikation?

Definition 10.39 (Problemspezifikation)

Eine **Problemspezifikation** ist eine exakte (mathematische) Definition der Problemstellung durch folgende Festlegungen:

- **Eingabedaten**
  Art und Form der Daten, Wertebereiche, Einschränkungen
- **Ausgabedaten**
  Wie bei Eingabedaten
- **Funktionaler Zusammenhang zwischen Ein- und Ausgabedaten**
  Welche Eingabedaten sollen welche Ausgabedaten erzeugen?

Beispiel 10.40

Problemstellung: *Berechne das Produkt zweier ganzer Zahlen*

- **Eingabe**: $a, b \in \mathbb{Z}$
- **Ausgabe**: $x \in \mathbb{Z}$
- **Funktionaler Zusammenhang**: $x = a \cdot b$

Beispiel für eine unpräzise Problemstellung

Beispiel 10.41

Problemstellung: *Suche ein Wort in einer Liste von Wörtern*

Es ist noch Folgendes unklar und muss in der Problemspezifikation präzisiert werden:

- Welche Wörter sind erlaubt (z.B. was ist das Alphabet?,...)
- Ist die Liste sortiert oder nicht sortiert?
- Ist die Liste wiederholungsfrei oder nicht?
- Was ist die Ausgabe, wenn das Wort in der Liste vorkommt (ja/nein, Position des Treffers,...)
- Was ist die Ausgabe, wenn das Wort nicht in der Liste vorkommt?
- Was ist die Ausgabe, wenn das Wort öfter in der Liste vorkommt?

Beispiel für eine Problemspezifikation

Problemstellung: *Suche ein Wort in einer Liste von Wörtern*

Problemspezifikation: Variante 1

- **Eingabe**:
  $w_1, \ldots, w_n \in A^+$ unsortierte Folge und $w \in A^+$ für ein Alphabet A
- **Ausgabe**:
  $k \in \mathbb{N}$
- **Funktionaler Zusammenhang**:
  $[(\forall i \in \{1,2,...,n\}(w_i \neq w)) \wedge (k = n+1)]$
  $\vee [(w_k = w) \wedge (\forall i \in \{1,2,...,k-1\}(w_i \neq w))]$
  - Ausgabe der ersten Position k mit $w_k = w$
  - i ist vom Typ $\mathbb{N}$

Beispiel für eine Problemspezifikation

Problemstellung: *Suche ein Wort in einer Liste von Wörtern*

Problemspezifikation: Variante 2

- **Eingabe**:
  $w_1, \ldots, w_n \in A^+$ aufwärts sortierte Folge (d.h. es gilt $\forall i \in \{1,2,...,n-1\}(w_i \leq w_{i+1})$) und $w \in A^+$ für ein geordnetes Alphabet A und die lexikografische Ordnung $\leq$ auf A^+
- **Ausgabe**:
  $k \in \mathbb{N}$
- **Funktionaler Zusammenhang**:
  $[(\forall i \in \{1,2,...,n\}(w_i < w)) \wedge (k = n+1)]$
  $\vee [(w_k \geq w) \wedge (\forall i \in \{1,2,...,k-1\}(w_i < w))]$
  - Ausgabe der ersten Position k mit $w_k \geq w$
  - i ist vom Typ $\mathbb{N}$

Beispiel für eine Problemspezifikation

Problemstellung: *Teste, ob ein Wort ein Palindrom ist*

Problemspezifikation

- **Eingabe**:
  $a_1 \dots a_n \in A^+$ für ein Alphabet A
- **Ausgabe**:
  $k \in \{0,1\}$
- **Funktionaler Zusammenhang**:
  $[(\forall i \in \{1,2,\dots,n \div 2\}(a_i = a_{n+1-i})) \Rightarrow (k = 1)]$
  $\wedge [(\exists i \in \{1,2,\dots,n \div 2\}(a_i \neq a_{n+1-i})) \Rightarrow (k = 0)]$
  - Erinnerung: Ein Wort ist ein Palindrom, falls es vorwärts und rückwärts gelesen gleich ist
  - i ist vom Typ $\mathbb{N}$

10. Mathematische Konzepte in der Informatik

Beweistechniken: Implikation von Aussagen

Seien p und q Formeln für die wir $p \Rightarrow q$ zeigen wollen.

Implikationskette

Finde eine Kette von Formeln $p = t_1, t_2, \ldots, t_{n-1}, t_n = q$, so dass

- $t_1 \Rightarrow t_2 \Rightarrow \ldots \Rightarrow t_n$, wobei
- $t_i \Rightarrow t_{i+1}$ für jedes i aus einer Definition, einer bereits bewiesenen Aussage oder bekannten Rechenregeln folgt

Jedes $\Rightarrow$-Zeichen der Rechnung, das aus einer Definition oder Aussage folgt, wird mit darübergestelltem Verweis begründet

Beispiel 10.42

Sei A ein Alphabet. Beweise: Die *ist Präfix von*-Relation auf A^* ist transitiv.
Zu zeigen ist für $u, v, w \in A^*$: Ist u Präfix von v und v Präfix von w, dann ist u auch Präfix von w.

Es gilt: (u Präfix von v) $\wedge$ (v Präfix von w) $\overset{(1)}{\Rightarrow} \exists x \in A^*(v = ux) \wedge \exists y \in A^*(w = vy)$

$\Rightarrow \exists x, y \in A^*(w = uxy) \Rightarrow \exists z \in A^*(w = uz) \overset{(1)}{\Rightarrow}$ (u Präfix von w)

- (1): Definition von Präfixen

Beweistechniken: Implikation von Aussagen

Seien p und q Formeln für die wir $p \Rightarrow q$ zeigen wollen.

Kontraposition

Beweise $\neg q \Rightarrow \neg p$

Beispiel 10.43

Beweise $(x \bmod 2 \neq 0) \Rightarrow (x \bmod 4 \neq 0)$:

$$(x \bmod 4 = 0) \overset{(1)}{\Rightarrow} \exists y \in \mathbb{N}(x = 4 \cdot y) \Rightarrow \exists y \in \mathbb{N}(x = 2 \cdot (2 \cdot y)) \Rightarrow$$
$$\exists y \in \mathbb{N}(x = 2 \cdot y) \overset{(1)}{\Rightarrow} (x \bmod 2 = 0)$$

- (1): Definition der Operation mod

Beweistechniken: Äquivalenz von Aussagen

Seien p und q Formeln für die wir $p \Leftrightarrow q$ zeigen wollen.

Äquivalenzkette

Finde eine Kette von Formeln $p = t_1, t_2, \ldots, t_{n-1}, t_n = q$, so dass

- $t_1 \Leftrightarrow t_2 \Leftrightarrow \ldots \Leftrightarrow t_n$, wobei
- $t_i \Leftrightarrow t_{i+1}$ für jedes i aus einer Definition, einer bereits bewiesenen Aussage oder bekannten Rechenregeln folgt

Jedes $\Leftrightarrow$-Zeichen der Rechnung, das aus einer Definition oder Aussage folgt, wird mit darübergestelltem Verweis begründet.

Beispiel 10.44

Beweise: ($\forall x(p)$ ist erfüllt) $\Leftrightarrow$ ($\neg(\exists x(\neg p))$ ist erfüllt)

($\forall x(p)$ ist erfüllt) $\overset{(1)}{\Leftrightarrow}$ (p ist für jede Belegung von x erfüllt) $\Leftrightarrow$ ($\neg p$ ist für keine Belegung von x erfüllt) $\overset{(2)}{\Leftrightarrow}$ ($\neg(\exists x(\neg p))$ ist erfüllt)

- (1): Definition von Erfüllbar für den All-Quantor
- (2): Definition von Erfüllbar für den Existiert-Quantor

Anmerkung: Wir machen hier Formeln selbst zu mathematischen Objekten in Formeln, indem wir die einstellige 'ist erfüllt'-Relation auf der Menge der Formeln betrachten.

Beweistechniken: Äquivalenz von Aussagen

Seien p und q Formeln für die wir $p \Leftrightarrow q$ zeigen wollen.

Ringschluss

Finde Ketten von Formeln $p = t_1, t_2, \ldots, t_{n-1}, t_n = q$ und $q = s_1, s_2, \ldots, s_{m-1}, s_m = p$, so dass

- $p = t_1 \Rightarrow t_2 \Rightarrow \ldots \Rightarrow t_n = q$
- $q = s_1 \Rightarrow s_2 \Rightarrow \ldots \Rightarrow s_m = p$

Zerlegung in Implikationen

Beweise $p \Rightarrow q$ und $p \Leftarrow q$.

Negation

Beweise $\neg p \Leftrightarrow \neg q$

Beweistechniken: Strukturelle Induktion

Sei M eine induktiv definierte Menge:

1. **Induktionsanfang**: Angabe von *Grundelementen* von M
2. **Induktionsschritt**: Angabe von *Bildungsregeln*, mit denen aus vorhandenen Elementen von M neue Elemente von M erstellt werden können

M besteht genau aus den Elementen, die man aus den Grundelementen durch wiederholte Anwendung der Bildungsregeln erstellen kann

Strukturelle Induktion

Eine Aussage der Form $\forall x \in M(E)$ beweist man wie folgt mit **struktureller Induktion**:

1. **Induktionsanfang**: Zeige $\forall x \in M_G(E)$ für die Teilmenge der Grundelemente M_G von M
2. **Induktionsschritt**: Zeige ($x_1, \ldots, x_n$ erfüllen E) $\Rightarrow$ (jedes x, das aus $x_1, \ldots, x_n$ mit einer der Bildungsregeln erstellt werden kann, erfüllt E)
   Die Aussage ($x_1, \ldots, x_n$ erfüllen E) nennt man **Induktionsvoraussetzung (IV)**

Beweistechniken: Zahlenvergleich

Seien A und B Terme und p eine aussagenlogische Formel, für die wir $p \Rightarrow (A = B)$ zeigen wollen.

Gleichheitsketten

Erstelle eine Kette von Termen $A = T_1, T_2, \ldots, T_{n-1}, T_n = B$, so dass

- $T_1 = T_2 = \ldots = T_n$, wobei
- $T_i = T_{i+1}$ für jedes i aus einer Definition, aus einer bereits bewiesenen Aussage, aus p oder einer bekannten Rechenregel folgt

Jedes $=$-Zeichen der Rechnung, das aus einer Definition, einer Aussage oder p folgt, wird mit darübergestelltem Verweis begründet

Beispiel 10.45

Beweise $(x \geq 0 \wedge c_{1K,n}(x) = b_{n-1} \ldots b_0) \Rightarrow (c_{1K,n}(-x) = (1 - b_{n-1}) \ldots (1 - b_0))$:

$c_{1K,n}(-x) \overset{(1)}{=} c_{2,n}(2^n - 1) - c_{2,n}(x) \overset{(2)}{=} 1^n - b_{n-1} \ldots b_0 = (1 - b_{n-1}) \ldots (1 - b_0)$

- (1): $x \geq 0$ und Definition von $c_{1K,n}$
- (2): $c_{1K,n}(x) = b_{n-1} \ldots b_0$

Beweistechniken: Zahlenvergleich

Seien A und B Terme und p eine aussagenlogische Formel, für die wir $p \Rightarrow (A = B)$ zeigen wollen.

Zerlegung in Ungleichungen

- Zeige $p \Rightarrow (A \leq B)$ und $p \Rightarrow (A \geq B)$,
- und benutze ggf. für jede der beiden Aussagen eine Ungleichungskette von Termen $A = T_1 \leq T_2 \leq \ldots \leq T_n = B$ bzw. $A = S_1 \geq S_2 \geq \ldots \geq S_n = B$.

Jedes $\leq$- und $\geq$-Zeichen der Rechnung wird wieder mit darübergestelltem Verweis begründet, falls es nicht aus einer bekannten Rechenregel folgt

Beispiel 10.46

Beweise $(u < s \wedge s \leq o \wedge u = o - 1) \Rightarrow (s = o)$ für Variablen u, s, o vom Typ $\mathbb{N}$:

- $s \leq o$ gilt per Voraussetzung
- Es gilt $s \overset{u<s}{\geq} u + 1 \overset{u=o-1}{=} o$

Beweistechniken: Vollständige Induktion

Sei n eine Variable vom Typ $\mathbb{N}$, $k \in \mathbb{N}$ und $A(n)$ eine prädikatenlogische Formel mit freier Variable n (für jedes $n \in \mathbb{N}$)

Vollständige Induktion

Eine Aussage der Form $\forall n \in [k, \infty[\,(A(n))$ beweist man wie folgt mit **vollständiger Induktion**:

1. **Induktionsanfang**: Zeige $A(k)$
2. **Induktionsschritt**: Zeige $A(n) \Rightarrow A(n+1)$
   Die Aussage $A(n)$ nennt man **Induktionsvoraussetzung (IV)**

Beispiel 10.47

Beweise $\forall n \in [1, \infty[\,(\sum_{i=1}^{n} i = \frac{n \cdot (n+1)}{2})$

1. **Induktionsanfang**: Zeige $A(1)$: $\sum_{i=1}^{1} i = 1 = \frac{1 \cdot (1+1)}{2}$
2. **Induktionsschritt**: Zeige $A(n) \Rightarrow A(n+1)$:
   $\sum_{i=1}^{n+1} i = (n+1) + \sum_{i=1}^{n} i \overset{A(n)}{=} (n+1) + \frac{n \cdot (n+1)}{2} = \frac{(n+1) \cdot (n+2)}{2}$

Beweistechniken: Mengenvergleich

Seien X und Y Mengen, für die wir $X \subseteq Y$ zeigen wollen.

Elementprinzip

Zeige $\forall x((x \in X) \Rightarrow (x \in Y))$ oder $\forall x((x \notin Y) \Rightarrow (x \notin X))$

Beispiel 10.48

Beweise: $(x \text{ prim} \wedge x > 2) \Rightarrow (x \text{ ungerade})$

$\neg\ (x \text{ ungerade}) \overset{(1)}{\Rightarrow} x = 2 \vee \exists n \geq 2 (x = 2 \cdot n) \overset{(2)}{\Rightarrow} \neg(x > 2) \vee \neg(x \text{ prim })$

- (1): Definition von geraden Zahlen
- (2): Definition von Primzahlen

Anmerkung: Der Vergleich von Mengen wurde hier auf die Implikation von Formeln zurückgeführt, da die Elementbeziehung einer einstelligen Relation entspricht

Beweistechniken: Mengenvergleich

Seien X und Y Mengen, für die wir $X = Y$ zeigen wollen.

Elementprinzip

Zeige $\forall x((x \in X) \Leftrightarrow (x \in Y))$

Beispiel 10.49

Beweise das Distributivgesetz für Mengen X, Y, Z: $(X \cup Y) \cap Z = (Y \cap Z) \cup (X \cap Z)$

$(x \in (X \cup Y) \cap Z) \overset{(1)}{\Leftrightarrow} (x \in X \cup Y \wedge x \in Z) \overset{(2)}{\Leftrightarrow} ((x \in X \vee x \in Y) \wedge x \in Z) \overset{(W)}{\Leftrightarrow} (x \in X \wedge x \in Z) \vee (x \in Y \wedge x \in Z) \overset{(1),(2)}{\Leftrightarrow} (x \in (Y \cap Z) \cup (X \cap Z))$

- (1): Definition Mengenschnitt
- (2): Definition Mengenvereinigung
- (W): Wahrheitstafel

Anmerkung: Mengenoperationen kann man also auf aussagenlogische Formeln zurückführen