

Theoretische Informatik I

Wintersemester 2006/07



Christoph Kreitz / Kirstin Peters

Theoretische Informatik

kreitz@cs.uni-potsdam.de

peters.kirstin@web.de

<http://www.cs.uni-potsdam.de/ti/lehre/06-Theorie-I>



1. Lehrziele und Lernformen
2. Lehrinhalte
3. Das Team
4. Organisatorisches
5. Gedanken zur Arbeitsethik

Lernen ist wie eine Bergbesteigung

- **Schule entspricht einem Wanderweg bis zur Alm**
 - Breit, gut beschildert, langsamer Anstieg
 - Etwas anstrengend aber ohne nennenswerte Schwierigkeiten
 - Sie sehen den Gipfel in der Ferne, erreichen ihn aber nicht
- **Universität ist wie Kletterpfade zum Gipfel**
 - Ein Gewirr von Wegen, steil, anstrengend und z.T. risikoreich
 - Bergführer können Ihnen mit Rat und Ausrüstung helfen
 - Bevor es losgeht, zeigt man Ihnen die wichtigsten Klettertechniken
 - Für den eigentlichen Aufstieg wählen Sie Weg und Begleiter selbst aus
 - Das anstrengende Klettern kann Ihnen niemand abnehmen
 - Manchmal übersteigt ein Pfad Ihre Kräfte und Sie müssen neu planen
 - Am Ende stehen Sie oben auf dem Gipfel ... ein echtes Erfolgserlebnis

UNTERSCHIEDE ZUR SCHULE

- Ihnen begegnen ständig neue Denkweisen
- Zu Beginn sehr grundlagenorientiert
- Erheblich steilerer Anstieg, höheres Niveau
- Angebote – statt Zwang und Anwesenheitspflicht
- Sie entscheiden allein, was Sie tun

Eigenverantwortung und Selbstdisziplin erforderlich

LEHRZIELE DES UNIVERSITÄTSSTUDIUMS

- Selbständigkeit
- Verantwortungsbewusstsein
- Berufsqualifikation
- Teamfähigkeit
- Qualifikation zur wissenschaftlichen Arbeit
- ... am Ende besser als wir!

Eigenständiges Lernen unter Anleitung

● **Selbststudium** ist das wichtigste

- Lernen durch Bearbeitung **verschiedener Quellen** (Literatur, Web,...)
- Trainieren durch Lösung von leichten und schweren **Beispielaufgaben** alleine und im Team mit anderen
- Nachweis von Fähigkeiten in Prüfungen und Projekten
- Ziel ist **Verständnis eines Themengebiets** (nicht nur der Vorlesung)
- Unsere Aufgabe ist, Ihnen dabei zu helfen

● **Vorlesung**

Was soll ich lernen ?

- **Vorstellung und Illustration** zentraler Konzepte und Zusammenhänge
- Knapp und “unvollständig” – nur als Heranführung gedacht
- Die **Idee (Verstehen)** zählt mehr als das Detail (Aufschreiben)
- Es hilft, schon etwas über das Thema **im Voraus** zu **lesen**
- **Stellen Sie Fragen**, wenn Ihnen etwas unklar ist !!
- Nutzen Sie das optionale Tutorium **zweiwöchentlich Do 11:00–12:30**

LEHR- UND LERNFORMEN (II)

● Übungen

Vertiefung und Anwendung

- Kurzquiz als Selbsttest – verstehe ich die Konzepte?
- Betreutes Üben in Gruppen: Bearbeitung von Aufgaben unter Anleitung
- Klärung von Fragen allgemeinen Interesses
- Bearbeitung von aufwändigeren Hausaufgaben: Feedback & Korrektur
 - Ziel ist verständliches Aufschreiben einer vollständigen Lösung
 - Arbeit in Gruppen sehr zu empfehlen
- Selbst aktiv werden ist notwendig für erfolgreiches Lernen
- Kommen Sie vorbereitet – Sie lernen mehr dabei

● Sprechstunden

Persönliche Beratung

- Fachberatung zur Optimierung des individuellen Lernstils
- Klärung von Schwierigkeiten mit der Thematik
 - ... aber nicht Lösung der Hausaufgaben
- Auch sinnvoll für bessere Studenten, die Herausforderungen suchen

● Problemstellungen

- Präzisierung: Wie beschreibt man Probleme?
- Berechenbarkeit: Ist ein Problem überhaupt lösbar?
- Effizienz: Ist ein Problem schwer oder leicht lösbar?

● Algorithmen: abstrakte Lösungsmethoden

- Was ist ein Algorithmus und welche Beschreibungsformen gibt es?
- Welche Merkmale und Eigenschaften haben Algorithmen?
- Wie kann man sicherstellen, daß eine Lösung korrekt ist?

● Programme: konkrete Lösungsvorschriften

- Syntax: Wie kann man Sprachen beschreiben, erkennen und erzeugen?
- Semantik: wie beschreibt man die Bedeutung von Programmen
- Wie flexibel kann man eine Programmiersprache gestalten?

● Maschinen: Ausführung von “Berechnungen”

- Welche grundsätzlichen Typen und Merkmale von Maschinen gibt es?
- Was können bestimmte Maschinentypen leisten?

● **Automatentheorie und Formale Sprachen** TI-1

- Endliche Automaten und Reguläre Sprachen – Lexikalische Analyse
- Kontextfreie Sprachen und Pushdown Automaten – Syntaxanalyse
- Turingmaschinen und allgemeine formale Sprachen

● **Theorie der Berechenbarkeit** TI-2

- Berechenbarkeitsmodelle
- Aufzählbarkeit, Entscheidbarkeit, Unlösbare Probleme

● **Komplexitätstheorie** TI-2

- Komplexitätsmaße und -klassen für Algorithmen und Probleme
- Nicht handhabbare Probleme ($\mathcal{NP}$ -Vollständigkeit)
- Effiziente Alternativen zu konventionellen Verfahren

● Reihenfolge und Notation folgt Leittext

- J. Hopcroft, R. Motwani, J. Ullman: *Einführung in die Automatentheorie, Formale Sprachen und Komplexitätstheorie*, Pearson 2002
- Vorlesungsfolien sind im Voraus auf dem Webserver erhältlich

● Lesenswerte Zusatzliteratur

- G. Vossen, K.-U. Witt: *Grundkurs Theoretische Informatik*. Vieweg 2004
- P. Leypold: *Schneller Studieren*. Pearson 2005
- M. Sipser: *Introduction to the Theory of Computation*. PWS 1997
- A. Asteroth, C. Baier: *Theoretische Informatik*, Pearson 2002
- I. Wegener: *Theoretische Informatik*, Teubner Verlag 1993
- U. Schöning: *Theoretische Informatik - kurzgefaßt*, Spektrum-Verlag 1994
- K. Erk, L. Priese: *Theoretische Informatik*, Springer Verlag 2000
- H. Lewis, C. Papadimitriou: *Elements of the Theory of Computation*, PHL 1998
- Mitschriften aus früheren Semestern benutzen andere Notationen

DAS TEAM



Christoph Kreitz

Raum 1.18, Telephon 3060
kreitz@cs.uni-potsdam.de



Kirstin Peters

Raum 1.22, Telephon 3068
peters.kirstin@web.de

Tutoren

Marcel Goehring

Jan Schwarz

Ellen König

Jens Steinborn

Marius Schneider

Holger Trölenberg

- **Zielgruppe: ab 3. Semester (!)**
 - Bei guten mathematischen Vorkenntnissen auch ab dem 1. Semester
 - Oft ist es sinnvoller erst an Mathematikveranstaltungen teilzunehmen
- **Vorlesung** (Wird aufgezeichnet und ins Internet gestellt)
 - Wöchentlich **Fr 11:00–12:30**
- **Tutorium** (Besprechung allgemeiner Fragen)
 - Jede zweite Woche **Do 11:00–12:30**
- **Übungen**
 - 6 **Gruppen**, wöchentlich (Montags & Dienstags) je 2 Stunden
- **Sprechstunden**
 - C. Kreitz: **Mi 9:30–10:30** ..., und immer wenn die Türe offen ist
 - K. Peters: **Do 12:30–13:30** und nach Vereinbarung
 - Tutoren: individuell in Übungsgruppen vereinbaren

- **Eine Klausur entscheidet die Note**

- Hauptklausur Anfang April (Ende vorlesungsfreie Zeit)
- Probeklausur Ende Dezember (geht nicht in Bewertung ein)

- **Zulassung zur Klausur**

- 50% der Punkte in den Hausaufgaben
 - Gruppen bis 4 Studenten dürfen gemeinsame Lösungen abgeben
 - Gruppen dürfen sich nur nach Rücksprache mit Frau Peters ändern
- Quiz und Probeklausur zählen jeweils wie ein Hausaufgabenblatt

- **Vorbereitung auf die Klausur**

- Kurzquiz in jeder Übungsstunde ernsthaft bearbeiten
- Eigenständige Lösung von Haus- und Übungsaufgaben
- Feedback durch Korrektur der Hausaufgaben und der Probeklausur
- Klärung von Fragen in Übung und Sprechstunden

Fangen Sie frühzeitig mit der Vorbereitung an

WELCHE VORKENNTNISSE SOLLTEN SIE MITBRINGEN ?

Eine gute Oberstufenmathematik reicht aus

● **Verständnis mathematischer Konzepte**

- Elementare Mengentheorie und die Gesetze von $\{x|P(x)\}$, $\cup$, $\cap$
- Bezug zwischen Mengen, Relationen und Funktionen
- **Datenstrukturen** wie Listen, Wörter, Graphen, Bäume ...
- Elementare Gesetze der **Algebra** und **Logik**
- Elementare **Wahrscheinlichkeitsrechnung**
- Zusammenhang zwischen formaler und informaler Beschreibung

Nötiges Vokabular wird bei Bedarf kurz vorgestellt/wiederholt/eingeübt

● **Verständnis mathematischer Beweismethoden**

Informatiker müssen **Korrektheit von Programmen beweisen können**

- Deduktive Beweise für Analyse von Befehlssequenzen
- Induktionsbeweise für Analyse von Rekursion / Schleifen
- Widerlegungsbeweise und Gegenbeispiele für Unmöglichkeitsaussagen

Zeige, daß Behauptung B aus Annahmen A folgt

● Deduktiver Beweis:

- Aneinanderkettung von Argumenten / Aussagen $A_1, A_2, \dots, A_n = B$
- Zwischenaussagen A_i müssen schlüssig aus dem Vorhergehenden folgen
- Verwendet werden dürfen nur Annahmen aus A , mathematische Grundgesetze, bereits bewiesene Aussagen und logische Schlußfolgerungen

Beispiel: *“Die Summe zweier ungerader Zahlen ist gerade”*

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

● Beweis durch Kontraposition

- Beweise, daß **nicht** A aus der Annahme **nicht** B folgt
- $\neg B \Rightarrow \neg A$ ist aussagenlogisch äquivalent zu $A \Rightarrow B$

● Indirekte Beweisführung

- Aus A und nicht B folgt ein Widerspruch (äquivalent zu $A \Rightarrow B$)

WICHTIGE BEWEISMETHODEN (II)

- **Widerlegungsbeweise:** Zeige, daß A nicht gilt

- Widerspruchsbeweis: Zeige, daß aus Annahme A ein Widerspruch folgt
- Gegenbeispiele beweisen, daß A nicht allgemeingültig ist

- **Induktionsbeweise:** Zeige, daß A für alle x gilt

Standardinduktion (auf natürlichen Zahlen):

Gilt A für i und folgt A für $n+1$, wenn A für n gilt, so gilt A für alle $n \geq i$

Vollständige Induktion:

Folgt A für n , wenn A für alle $j < n$ mit $j \geq i$ gilt, dann gilt A für alle $n \geq i$

Strukturelle Induktion (auf Datentypen wie Listen, Bäumen, Wörtern):

Gilt A für das Basiselement und folgt A für ein zusammengesetztes Element, wenn A für seine Unterelemente gilt, dann gilt A für alle Elemente

Gegenseitige oder simultane Induktion

Zeige mehrere zusammengehörige Aussagen gleichzeitig in einer Induktion

Mehr dazu im Anhang

WAS IST EIGENTLICH EIN BEWEIS?

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

- Genau genug, um Details rekonstruieren zu können
 - Knapp genug, um übersichtlich und merkbar zu sein
 - Text muß lesbar und klar verständlich sein und präzise Sprache verwenden
Formeln und Textfragmente ohne erkennbaren Sinn aneinanderzureihen ist unakzeptabel
 - Zwischenschritte müssen mit “üblichen” Vorkenntnissen erklärbar sein
 - Also nicht notwendig formal oder mit allen Details
 - 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 Ihre Kommilitonen Ihre Lösung und warum sie funktioniert?

NUTZEN SIE IHRE CHANCEN!

- **Theorie ist bedeutender als viele glauben**

- Ist Theorie langweilig? überflüssig? unverständlich? ...eine Plage?
- Alle großen Softwareprojekte benutzten theoretische Modelle
- Ohne theoretische Kenntnisse begehen Sie viele elementare Fehler
- Theorie kann **durchaus sehr interessant** sein

- **Es geht um mehr als nur bestehen**

- Das wichtige ist **Verstehen**
- Sie können jetzt umsonst lernen, was später teure Lehrgänge benötigt
- Wann kommen Sie je wieder mit den Besten des Gebietes in Kontakt?

- **Die Türe steht offen**

- Lernfrust und mangelnder Durchblick sind normal aber heilbar
- Kommen Sie in die Sprechstunden und stellen Sie Fragen

VERTRAUEN IST EIN KOSTBARES GUT

... missbrauchen Sie es nicht

- **Abschreiben fremder Lösungen bringt nichts**

- Sie lernen nichts dabei – weder Inhalt noch Durchhaltevermögen
- Sie erkennen Ihre Lücken nicht und nehmen Hilfe zu spät wahr
- Sie werden nie ein echtes Erfolgserlebnis haben
- Es schadet Ihrer persönlichen Entwicklung

- **Wir vertrauen Ihrer Ehrlichkeit**

- Benutzen Sie externe Ideen (Bücher/Internet) nur mit Quellenangabe
- Benutzen Sie keine Lösungen von Kommilitonen
- Geben Sie keine Lösungen an Kommilitonen weiter

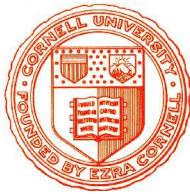
Klausurlösungen sollten ausschließlich Ihre eigenen sein

Keine “Überwachung”, aber wenn es dennoch auffliegt ...

- **Mehr zur Arbeitsethik auf unseren Webseiten**

ANHANG

Theoretische Informatik I



Lektion 1

Mathematische Methodik



1. Problemlösen
2. Beweistechniken
3. Wichtige Grundbegriffe

● 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

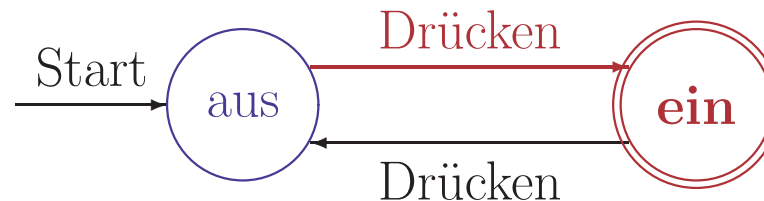
- Welche **Einzelschritte** benötigt man, um das Problem zu lösen?
- Welches **Gesamtergebnis** ergibt sich aus den Einzelschritten?
- Wie **beweist** man die Korrektheit des Gesamtergebnisses?

● Lösung zusammenfassen

- **Kurz und prägnant**: Argumente auf das Wesentliche beschränken
- Umgangssprache durch mathematisch präzise Formulierungen ersetzen

● **Automaten:** Abarbeitung von Eingaben

- z.B. Wechselschalter: Verarbeitung von “Drücken”-Eingaben



- 2 Zustände: aus, ein – 1 Startzustand: aus
- 1 Eingabesymbol: Drücken
- 1 Endzustand: ein — wird erreicht bei ungerader Anzahl von Drücken

● **Grammatiken:** Vorschriften für Spracherzeugung

- z.B.: $S \rightarrow \text{Drücken}$, $S \rightarrow S\text{DrückenDrücken}$
- Erzeugt nur ungerade Anzahl von Drücken-Symbolen

● **Reguläre Ausdrücke:** algebraische Strukturen

- z.B.: $(\text{DrückenDrücken})^*\text{Drücken}$

- **Testen von Programmen ist unzureichend**
 - Nur hilfreich zur Entdeckung grober Fehler
 - Viele kleine, aber gravierende Fehler fallen durch das Testraster
 - Pentium Bug (1994), Ariane 5 (1996), Mars Polar Lander (1999), ...
- **Kritische Programme muss man “beweisen”**
 - Erfolgreicher Beweis zeigt genau, wie das Programm arbeitet
 - Erfolgloser Beweisversuch deutet auf mögliche Fehler im Programm
 - Jeder Informatiker sollte die eigenen Programme beweisen
- **Jeder Informatiker muss Beweise verstehen**
 - Deduktive Beweise für sequentielle Verarbeitung
 - Induktionsbeweise für Rekursion / Schleifen
 - Widerlegungsbeweise und Gegenbeispiele für Unmöglichkeitsaussagen

Wie führt man stichhaltige Beweise?

BEHAUPTUNGEN: AUSGANGSPUNKT JEDES BEWEISES

● Wenn–Dann Aussagen:

- Eine **Konklusion** folgt aus einer oder mehreren **Hypothesen** (Annahmen)
- z.B. “*Wenn $x \geq 4$, dann $2^x \geq x^2$* ”
- Auch: H impliziert K , aus H folgt K , K wenn H , $H \Rightarrow K$
- Achtung: wenn K gilt, muss H nicht der Grund sein

Fast alle Behauptungen haben diese Form

- Hypothesen sind zuweilen implizit oder ergeben sich aus dem Kontext
- z.B. “ $\sin^2\theta + \cos^2\theta = 1$ ” hat implizite Hypothese “ θ ist ein Winkel”

● Genau dann, wenn Aussagen

- Aussagen A und B sind äquivalent ($A \Leftrightarrow B$, $A \equiv B$, A iff B (engl.))
- z.B. “ $x^2 = 1$ genau dann, wenn $x = 1$ ”
- Gleichwertig mit $A \Rightarrow B$ und $B \Rightarrow A$

BEISPIEL EINES DEDUKTIVEN BEWEISES

Wenn x die Summe der Quadrate von vier positiven ganzen Zahlen ist, dann gilt $2^x \geq x^2$

● Informaler Beweis

- Es sei x die Summe der Quadrate von vier positiven ganzen Zahlen
- Das Quadrat jeder positiven ganzen Zahl ist mindestens 1
- Aus der Annahme folgt damit, dass $x \geq 4$ sein muss
- Wir benutzen den Satz “*Wenn $x \geq 4$, dann $2^x \geq x^2$* ” und schließen daraus, dass $2^x \geq x^2$ gilt

HMU Satz 1.3, Folie 14

● Beweis in schematischer Darstellung

Aussage	Begründung
1. $x = a^2 + b^2 + c^2 + d^2$	Gegeben
2. $a \geq 1, b \geq 1, c \geq 1, d \geq 1$	Gegeben
3. $a^2 \geq 1, b^2 \geq 1, c^2 \geq 1, d^2 \geq 1$	(2) und Gesetze der Arithmetik
4. $x \geq 4$	(1), (3) und Gesetze der Arithmetik
5. $2^x \geq x^2$	(4) und HMU Satz 1.3, Folie 14

Logische Schritte von Annahmen zur Konklusion

- **Beweis $\hat{=}$ Folge von Zwischenaussagen**

- Beginne mit Menge der Annahmen
- Jede Zwischenaussage folgt schlüssig aus vorhergehenden Aussagen
- Konklusion ergibt sich als letzter Beweisschritt

- **Zulässige Argumente in Beweisschritten**

- Logischer Schluss: Sind A und $A \Rightarrow B$ bekannt, kann B gefolgert werden
- Bekannte mathematische Grundgesetze, z.B. aus der Arithmetik
- Bereits bewiesene Sätze
- Auflösung von Definitionen
- Extensionalität von Mengen: $M=M' \Leftrightarrow M \subseteq M' \wedge M' \subseteq M$
 $M \subseteq M' \Leftrightarrow (\forall x) x \in M \Rightarrow x \in M'$
- Gleichheit von Zahlen: $x=y \Leftrightarrow$ weder $x < y$ noch $x > y$

BEISPIEL FÜR AUFLÖSUNG VON DEFINITIONEN

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

● Definitionen

S endlich $\equiv$ Es gibt eine ganze Zahl n mit $||S|| = n$

T Komplement von S $\equiv T \cup S = U$ und $T \cap S = \emptyset$

● Beweis

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)

BEWEIS EINER MENGENÄQUIVALENZ

Für beliebige Mengen R und S gilt $R \cup S = S \cup R$

● Definitionen

$$x \in R \cup S \equiv x \in R \text{ oder } x \in S$$

● Zu zeigen:

- $R \cup S = S \cup R$ also
- $R \cup S \subseteq S \cup R$ und $S \cup R \subseteq R \cup S$ also
- Wenn $x \in R \cup S$, dann $x \in S \cup R$ und wenn $x \in S \cup R$, dann $x \in R \cup S$

● Beweis der ersten Implikation

Aussage	Begründung
1. $x \in R \cup S$	Gegeben
2. $x \in R$ oder $x \in S$	Auflösen der Definition in (1)
3. $x \in S$ oder $x \in R$	Logische Umstellung von (2)
4. $x \in S \cup R$	Einsetzen der Definition in (3)

Beweis der zweiten Implikation genauso

WIE GENAU/FORMAL MUSS EIN BEWEIS SEIN?

Ein Beweis ist ein Argument, das den Leser überzeugt

- Präzise genug, um Details rekonstruieren zu können
 - Knapp genug, um übersichtlich und merkbar zu sein
 - Zwischenschritte müssen mit “üblichen” Vorkenntnissen erklärbar sein
 - Also nicht notwendig formal oder mit allen Details
 - 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 Ihre Kommilitonen Ihre Lösung und warum sie funktioniert?

Zeige, dass eine Aussage A nicht gilt

● Beweis durch Widerspruch

- A gilt nicht, wenn aus der Annahme von A ein Widerspruch folgt
- z.B. *Wenn S endliche Teilmenge einer unendlichen Menge U ist, dann ist das Komplement von S (bezüglich U) nicht endlich*

Beweis	Aussage	Begründung
	1. S endlich	Gegeben
	2. T Komplement von S	Gegeben
	3. U unendlich	Gegeben
	4. T endlich	Annahme
	5. U endlich	(1), (4) mit Satz auf Folie 7
	6. Widerspruch	(3), (5)
	7. T nicht endlich	Annahme (4) muss falsch sein

● Beweis durch Gegenbeispiel

- A ist nicht allgemeingültig, wenn es ein einziges Gegenbeispiel gibt
- z.B. *Wenn x eine Primzahl ist, dann ist x ungerade* ist falsch
- Beweis: 2 ist eine gerade Zahl, die eine Primzahl ist

● Beweis durch Kontraposition

- Statt *wenn H , dann K* zeige *wenn nicht K , dann nicht H*
- Behauptungen sind aussagenlogisch äquivalent

● Spezielle Anwendung: Indirekte Beweisführung

- Zeige, dass aus *H und nicht K* ein Widerspruch folgt
Aussagenlogisch äquivalent zu *wenn H , dann K*
- z.B. *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

Gegenbeispielkonstruktion für unendliche Objekte

- **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
2. Computer sind universelle Maschinen
 - Bei Eingabe von Programm und Daten berechnen sie das Ergebnis
 - Schreibweise: $p_i(j) \hat{=}$ Anwendung des i -ten Programms auf die Zahl j
3. Man kann Programme beliebig zu neuen Programmen zusammensetzen
... und die Nummer des neuen Programms berechnen

TERMINIERUNG VON PROGRAMMEN IST UNENTSCHEIDBAR

- **Annahme:** es gibt ein Programm für den Terminierungstest
 - $\text{Term}(i,j)=1$ falls $p_i(j)$ hält (sonst 0)

- **Konstruiere ein neues 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	$\perp$	$\times$	$\perp$	$\times$	$\perp$	...
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	...

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

- **Unsinn** ist ein Programm

Also muss **Unsinn** eine **Nummer k** haben

- **Was macht $p_k = \text{Unsinn}$ auf seiner eigenen Nummer?**

- 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

Beweise eine Aussage A für alle natürlichen Zahlen

● Standardinduktion

- Gilt A für i und folgt A für $n+1$, wenn A für n gilt, dann gilt A für alle $n \geq i$
- z.B. *Wenn $x \geq 4$, dann $2^x \geq x^2$*

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

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

Dann ist $2^{n+1} = 2 \cdot 2^n \geq 2n^2$ aufgrund der Induktionsannahme

und $(n+1)^2 = n^2 + 2n + 1 = n(n+2+1/n) \leq n(n+n) = 2n^2$ wegen $n \geq 4$

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

● Vollständige Induktion

- Folgt A für n , wenn A für alle $j < n$ mit $j \geq i$ gilt, dann gilt A für alle $n \geq i$
- Mächtiger, da man nicht den unmittelbaren Vorgänger benutzen muss

STRUKTURELLE INDUKTION

Zeige A für alle Elemente eines rekursiven Datentyps

Gilt A für das Basiselement und folgt A für ein zusammengesetztes Element, wenn A für seine Unterelemente gilt, dann gilt A für alle Elemente

– z.B. *Die Summe einer Liste L von positiven ganzen Zahlen ist mindestens so groß wie ihre Länge*

Induktionsanfang L ist leer: Die Summe und die Länge von L sind 0

Induktionsschritt: Es gelte $sum(L) \geq |L|$

Betrachte die Liste $L \circ x$, die durch Anhängen von x and L entsteht

Dann gilt $sum(L \circ x) = sum(L) + x \geq sum(L) + 1 \geq |L| + 1 = |L \circ x|$

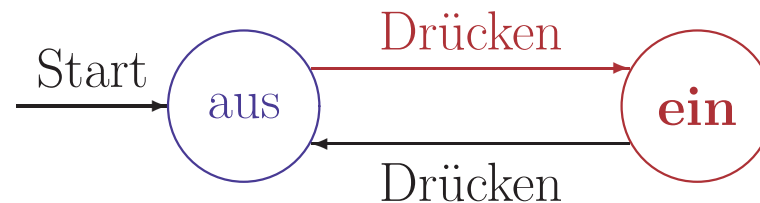
Häufig eingesetzt für Analyse von

- **Baumstrukturen** (Suchen, Sortieren, ...)
- **Syntaktische Strukturen** (Formeln, Programmiersprachen, ...)

⋮

GEGENSEITIGE INDUKTION

Zeige mehrere zusammengehörige Aussagen simultan



Zeige: Automat ist ein Wechselschalter

$S_1(n)$: Ist n gerade, so ist der Automat nach n -fachem Drücken ausgeschaltet

$S_2(n)$: Ist n ungerade, so ist der Automat nach n -fachem Drücken eingeschaltet

Induktionsanfang $n=0$: n ist gerade also gilt $S_2(0)$

der Automat ist ausgeschaltet, also gilt $S_1(0)$

Induktionsschritt: Es gelte $S_1(n)$ und $S_2(n)$. Betrachte $n+1$

– Falls $n+1$ ungerade, dann gilt $S_1(n+1)$ und n ist gerade.

Wegen $S_1(n)$ war der Automat “aus” und wechselt auf “ein”. Es gilt $S_2(n+1)$

– Falls $n+1$ gerade, dann gilt $S_2(n+1)$ und n ist ungerade.

Wegen $S_2(n)$ war der Automat “ein” und wechselt auf “aus”. Es gilt $S_1(n+1)$

- **Alphabet Σ** : endliche Menge von Symbolen,
z.B. $\Sigma = \{0, 1\}$, $\Sigma = \{0, \dots, 9\}$, $\Sigma = \{A, \dots, Z, a, \dots, z, \ , ?, !, \dots\}$
- **Wörter**: endliche Folge w von Symbolen eines Alphabets
Auch **Zeichenreihen** oder **Strings** genannt
- **ϵ** : Leeres Wort (ohne jedes Symbol)
- **wv** : **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 $w = vu$ für ein Wort u
- **Σ^k** : Menge der Wörter der Länge k mit Symbolen aus Σ
- **Σ^*** : Menge aller Wörter über Σ
- **Σ^+** : Menge aller nichtleeren Wörter über Σ
- **Sprache L** : Beliebige Menge von Wörtern über einem Alphabet Σ
Üblicherweise in abstrakter Mengennotation gegeben
z.B. $\{w \in \{0, 1\}^* \mid |w| \text{ ist gerade}\} \quad \{0^n 1^n \mid n \in \mathbb{N}\}$
- **Problem P** : Menge von Wörtern über einem Alphabet Σ
Das “Problem” ist, Zugehörigkeit zur Menge P zu testen

- **Funktion $f : S \rightarrow S'$** : Abbildung zwischen den Grundmengen S und S'
nicht unbedingt auf allen Elementen von S definiert
- **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 injektiv!)
- **Urbild $f^{-1}(L)$** : Die Menge $\{x \in S \mid f(x) \in L\}$
- **Charakteristische Funktion χ_L von $L \subseteq S$** : $\chi_L(w) = \begin{cases} 1 & \text{falls } w \in L, \\ 0 & \text{sonst} \end{cases}$
- **Partiell-charakteristische Funktion ψ_L** : $\psi_L(w) = \begin{cases} 1 & \text{falls } w \in L, \\ \perp & \text{sonst} \end{cases}$

Mehr Vokabular wird bei Bedarf vorgestellt