Theoretische Informatik I
Modellierungskonzepte der Informatik

Wintersemester 2013/14

Christoph Kreitz / Nuria Brede

Theoretische Informatik

{kreitz,brede}@cs.uni-potsdam.de

http://cs.uni-potsdam.de/ti1-ws1314

1. Lehrziele und Lernformen
2. Organisatorisches
3. Lernerfolg und Arbeitsethik

Theoretische Informatik: Worum geht es?

- **Analyse von Grundsatzfragen der Informatik**
  - Für welche Probleme gibt es gute, allgemeingültige Verfahren?
  - Wie kann man Systeme programmiersprachenunabhängig modellieren
  - Wie flexibel kann man Programmiersprachen gestalten?
  - Wie hängen verschiedenartige Computerarchitekturen zusammen?
  - Was kann man prinzipiell mit Computern nicht lösen, was nicht?
  - Welche Probleme sind effizient lösbar – welche nicht?
- **Mathematische Vorgehensweise**
  - Abstraktion von irrelevanten Details (Hardware/Programmiersprache)
  - Erkenntnisse sind beweisbar und haben weitreichende Gültigkeit
- **Der älteste Zweig der Informatik**
  - Theoretische Erkenntnisse gab es lange vor den ersten Computern
  - Aussagen sind langlebiger als in den anderen Informatikzweigen
  - Aber: Erkenntnisse sind selten “unmittelbar” anwendbar

Wozu soll das gut sein?

Denken ist besser als Hacken

- **Verständnis allgemeiner Zusammenhänge**
  - Details ändern sich schnell, Modelle und Methoden nicht
- **Effizientere Arbeitsweise**
  - Abstraktion richtet den Blick auf das Wesentliche
    Wenn Sie zu sehr auf Details schauen, verlieren Sie die Übersicht
  - Abstrakte Lösungen sind auf spezifische Situationen übertragbar
- **Vermeidung der gröbsten Fehler**
  - z.B. Korrektheit von Software kann nicht getestet werden
    Optimale Navigation ist nicht effizient möglich
    Flexible Programmiersprachen sind nicht (effizient) compilierbar
  - Viele Programmierer haben sich schon in unlösbare Probleme verbissen, weil sie das nicht wußten oder nicht geglaubt haben

Konkrete Themen der Theoretischen Informatik

- **Mathematische Methodik in der Informatik** TI-1
- **Automatentheorie und Formale Sprachen** TI-1
  - Endliche Automaten und Reguläre Sprachen – Lexikalische Analyse
  - Kontextfreie Sprachen und Pushdown Automaten – Syntaxanalyse
  - Turingmaschinen, kontextsensitive und allgemeine formale Sprachen

---

- **Theorie der Berechenbarkeit** TI-2
- **Komplexitätstheorie** TI-2

Der Lehrstoff

- **Reihenfolge und Notation folgt Leittext**
  - J. Hopcroft, R. Motwani, J. Ullman: *Einführung in die Automatentheorie, Formale Sprachen und Komplexitätstheorie, Pearson 2002* (nicht 2011)
  - Vorlesungsfolien sind im Voraus auf dem Webserver erhältlich
  - Videomitschnitte der Vorlesungen von 2005/2006/2011 online verfügbar

- **Lesenswerte Zusatzliteratur**
  - G. Vossen, K.-U. Witt: *Grundkurs Theoretische Informatik. Vieweg 2011*
  - M. Sipser: *Introduction to the Theory of Computation. PWS 2012*
  - A. Asteroth, C. Baier: *Theoretische Informatik, Pearson 2008*
  - J. Hromkovic: *Sieben Wunder der Informatik, Teubner Verlag 2006*
  - I. Wegener: *Theoretische Informatik, Teubner Verlag 2005*
  - U. Schöning: *Theoretische Informatik - kurzgefaßt, Spektrum-Verlag 2008*
  - K. Erk, L. Priese: *Theoretische Informatik, Springer Verlag 2009*
  - H. Lewis, C. Papadimitriou: *Elements of the Theory of Computation, PH 1998*
  - P. Leypold: *Schneller Studieren. Pearson 2005*

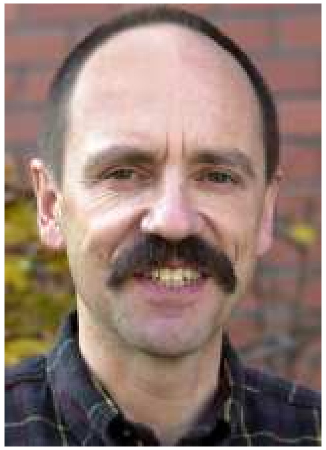
Lehr- und Lernformen

- **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 bewußt “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 — wöchentlich Do 14:15–15:45

Lehr- und Lernformen (II)

- **Übungen** — Vertiefung und Anwendung
  - Kurzquiz als Selbsttest – verstehe ich bisher besprochene Konzepte?
  - Betreutes Üben in Gruppen: Lösung von Problemen unter Anleitung
  - Klärung von Fragen allgemeinen Interesses
  - Eigenständige Einarbeitung in neue Konzepte und Fragestellungen
  - Bearbeitung von aufwändigeren Hausaufgaben: Feedback & Korrektur
    - Ziel ist verständliches Aufschreiben einer vollständigen Lösung
    - Arbeit in Gruppen sehr zu empfehlen
    - Lösungen schwieriger Aufgaben werden im Tutorium besprochen
  - 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
  - Hilfe, wenn der Lernfrust überhand nimmt

Das Team



Christoph Kreitz

Raum 1.18, Telephon 3060

kreitz@cs.uni-potsdam.de

Nuria Brede

Raum 1.23, Telephon 3071

brede@cs.uni-potsdam.de

Tutoren

Thorsten Alten	talten@uni-potsdam.de
Tina Beigel	tbeigel@uni-potsdam.de
Hannes Schröder	hschroed@uni-potsdam.de
Jan Splett	mail@jan-splett.de

Organisatorisches

- **Zielgruppe: ab 1. Semester**
  - Bei geringen mathematischen Vorkenntnissen besser ab 3. Semester
  - Oft ist es sinnvoll erst an Mathematikveranstaltungen teilzunehmen
- **Vorlesung**
  - Wöchentlich Fr 8:15–9:45
- **Tutorium** (optional, aber sehr zu empfehlen)
  - Besprechung von allgemeinen Fragen und schwierigen Hausaufgaben
  - Wöchentlich Do 14:15–15:45, ab 24. Oktober
- **Übungen**
  - 4 Gruppen, wöchentlich (Montags – Mittwochs) je 2 Stunden
- **Sprechstunden**
  - C. Kreitz: Fr 10:30–11:30 . . . und immer wenn die Türe offen ist
  - N. Brede: Do 10:00–11:30??????????????????????????????
  - Tutoren: individuell in Übungsgruppen vereinbaren

Leistungserfassung

- **Eine Klausur entscheidet die Note** (Anmeldung erforderlich!)
  - Hauptklausur 14. Februar 2014, 12:30–15:30 Uhr
  - Probeklausur 20. Dezember 2013, 8:15–9:45 Uhr
- **Zulassung zur Klausur**
  - 50% der Punkte in den Hausaufgaben
    - Gruppen bis 4 Studenten dürfen gemeinsame Lösungen abgeben
    - Gruppen dürfen sich nur nach Rücksprache ändern
    - Klausurzulassungen aus Vorjahren sind nicht gültig
  - Probeklausur zählt 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 den Vorbereitungen an

Welche Vorkenntnisse sollten Sie mitbringen?

Eine gute Oberstufenmathematik reicht aus

- **Verständnis mathematischer Konzepte und Notationen**
  - 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**
  - Wichtig für Analyse von Befehlssequenzen, Schleifen und Rekursion
  - Essentiell, um zeigen zu können, daß etwas nicht möglich ist

Mehr dazu nach der Pause

Lernerfolg

- **kommt nicht durch passive Teilnahme**
  - Sammeln von Folien, Übungen, Videos, ...
  - Ansehen von Vorlesungsvideos, Folien und Musterlösungen
  - Rein physische Anwesenheit in Vorlesung und Übung
  - "Bulimielernen" kurz vor der Klausur
- **sondern nur durch eigene Aktivität**
  - Durcharbeiten von Notizen/Folien/Buch/Webseiten vor der Vorlesung
  - Erstellen und Überarbeiten von Notizen zur Vorlesung *(aktives Lesen)*
  - "Weiterdenken": welche Fragen/Ideen folgen aus den Vorlesungsthemen?
  - Fragen stellen während, vor, nach der Vorlesung/Übung/Sprechstunde
    "wer nicht fragt bleibt dumm" – Sie können sich nicht blamieren
  - Erarbeitung von Lösungsideen zu Aufgaben vor der Übungsstunde
- **... und Ehrlichkeit**
  - Sie lernen nichts, wenn Sie Lösungen von anderen übernehmen
  - Erfolgserlebnisse entstehen, wenn Sie Schwierigkeiten selbst überwinden

Wir helfen Ihnen gerne dabei

Theoretische Informatik I

Einheit 1

Mathematische Beweisführung

1. Deduktive Beweise
2. Widerlegungsbeweise
3. Induktion und Datenstrukturen
4. 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
    - · Pentium Bug (1994), Ariane 5 (1996), Mars Polar Lander (1999), ...
- **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

  $\mapsto$ Entwickler müssen die Korrektheit ihrer Programme beweisen
- **Informatiker müssen Beweise verstehen / führen können**

Wie führt man stichhaltige Beweise?

Was ist eigentlich ein Beweis?

"Ein Beweis ist ein Argument, das den Leser überzeugt"

- **Begründung, warum eine Behauptung aus Annahmen folgt**
  - Intuitiv einsichtig, logisch schlüssig und lückenlos
- **Genau, aber nicht notwendig formal oder mit allen Details**
  - Knapp genug, um verständlich (und merkbar) zu sein
    Übersichtlich und mit erkennbarem roten Faden
  - Genau genug, um Details rekonstruieren zu können
    Gedankensprünge sind nur erlaubt, wenn sie leicht nachvollziehbar sind
- **Eine präzise Sprache ist wichtig**
  - Umgangssprachliche Formulierungen sind 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
  - Für Hochsicherheitsprobleme sind nur maschinelle Beweise akzeptabel

Zu dieser Thematik gibt es Spezialvorlesungen wie "Automatisierte Logik und Programmierung"

Die einfachste Beweisform: Deduktive Beweise

- **Ziel: Zeige, daß Behauptung B aus Annahmen A folgt**
- **Beweis ist Kette von Argumenten $A_1, .., A_n = B$**
  - Zwischenaussagen A_i müssen schlüssig aus dem Vorhergehenden folgen
  - Dabei dürfen nur Annahmen aus A, Definitionen, bewiesene Aussagen, mathematische Gesetze, logische Schlußfolgerungen verwendet werden
- **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"

Ausführlicher schematischer Beweis

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

- **Informales Argument beschreibt Kernidee des Beweises**
  - U ist die Vereinigung zweier endlicher Mengen, also endlich
- **Schematischer Beweis präsentiert alle Details**

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)

In einfachen Beweisen ist das informale Argument oft einsichtiger

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

WIDERLEGUNGSBEWEISE
ZEIGE, DASS EINE BEHAUPTUNG B NICHT GILT

- **Widerspruchsbeweise zeigen, daß B niemals gelten kann**
  - Zeige, daß aus Annahme B ein Widerspruch folgt
  - 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 4 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

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 $2^{n+1} = 2*2^n \geq 2n^2$ (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

Beispiel: 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
- **Induktionsbeweis benötigt vollständige Induktion**

Behauptung $B(n)$: Für alle $n{\geq}8$ gibt es $a, b \in \mathbb{N}$ 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 $n{\geq}8$

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 $B_2(0)$

  (Voraussetzung nicht erfüllt)

  Induktionsannahme: Es gelte $B_1(n)$ und $B_2(n)$.

  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

Simultane Induktion ausformuliert $f(0)=0,\ f(n+1)=1-f(n)$

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 gerade $\Leftrightarrow f(n)=0$

$B_2(n)$: n 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

$\Leftrightarrow f(m) = 1$ (Induktionsannahme $B_2(m)$)

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

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

$\Leftrightarrow f(m) = 0$ (Induktionsannahme $B_1(m)$)

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

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

Induktionsbeweise für andere Datenstrukturen

Viele Datenstrukturen lassen sich induktiv beschreiben

- **$\mathbb{N}$: 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)
  - $l \in \text{List } T \wedge x \in T \Rightarrow x :: l \in \text{List } T$ (Voranstellen 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)
  - $x \in T \wedge t_1..t_n \in \text{Tree } T \Rightarrow [x; t_1, .., t_n] \in \text{Tree } T$
    (Baum mit Wurzel x und Unterbäumen $t_1..t_n$)

Induktionsbeweise folgen der Struktur dieses Aufbaus

STRUKTURELLE INDUKTION

BEWEIS AUF BASIS DER STRUKTUR EINES INDUKTIVEN DATENTYPS

Gilt Behauptung B für das Basiselement des Typs und für ein zusammengesetztes Element, wenn B für seine Unterelemente gilt, dann gilt B für alle Elemente des Typs

– Zahlen: Gilt $B(0)$ und $B(n{+}1)$, wenn $B(n)$ gilt, dann gilt $\forall x{\in}\mathbb{N}.B(x)$

– Listen: $B([\,]) \wedge (B(l) \Rightarrow \forall x{\in}T.B(x :: l)) \Rightarrow \forall l{\in}\text{List } T.B(l)$

– Wörter: $B(\epsilon) \wedge (B(w) \Rightarrow \forall a{\in}\Sigma.B(w\,a)) \Rightarrow \forall w{\in}\Sigma^*.B(w)$

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

- **Beispiel:** *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 $l' = x :: l$

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

Aufgrund des Induktionsprinzips folgt die Behauptung für alle Listen

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

$$\begin{aligned}
\#(t) &= \#(t_1) + \#(t_2) + 1 \\
&\leq 2^{d(t_1)} - 1 + 2^{d(t_2)} - 1 + 1 && \text{(IA)} \\
&\leq 2^{max(d(t_1), d(t_2))} - 1 + 2^{max(d(t_1), d(t_2))} - 1 + 1 \\
&= 2(*) 2^{max(d(t_1), d(t_2))} - 1 \;=\; 2^{d(t)} - 1
\end{aligned}$$

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

Methodik des Problemlösens

- **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: $\{x \in S \mid P(x)\}$, $\{f(x) \mid x \in S\}$, ...
  - 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 $S_1 \supseteq S_2$, ...

Mathematisches Vokabular: Funktionen

– Funktion $f : S \to S'$: Abbildung zwischen Grundmengen S und S'
  nicht unbedingt auf allen Elementen von S definiert (⊥ = undefiniert)

– Domain von f: *domain*(f) = $\{x \in S \mid f(x)$ 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 $f^{-1}: S' \to S$: $f^{-1}(y) = x \Leftrightarrow f(x) = y$ (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: $\psi_M(x) = \begin{cases} 1 & \text{falls } x \in M, \\ \perp & \text{sonst} \end{cases}$

$\boxed{\text{Mehr Vokabular wird bei Bedarf vorgestellt}}$

Mathematisches Vokabular: Listen

List T: Datenstruktur mit Elementen aus T

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

  $(map(f, [x_1, .., x_n]) = [f(x_1), .., f(x_n)])$
- $l[n]$ (oder l_n): n-tes Element von l $\quad([x_1, .., x_n][m] = x_m)$

Viele weitere Konzepte definierbar

MATHEMATISCHES VOKABULAR: WÖRTER UND SPRACHEN

– Alphabet Σ: endliche Menge von Symbolen,
  z.B. $\Sigma = \{0,1\}$, $\Sigma = \{0,..,9\}$, $\Sigma = \{A,..,Z,a,..,z,\ \ ,?,!,..\}$
– 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. $\{w \in \{0,1\}^* \mid |w| \text{ ist gerade}\}$ $\{0^n1^n \mid n \in \mathbb{N}\}$
– Eine Sprachen kann auch als Problem P bezeichnet werden
(Das “Problem” ist in diesem Fall, die Zugehörigkeit zur Menge P zu testen)

Mathematisches Vokabular: Graphen

- **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 $[v_1, .., v_n]$ 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 vn H erzeugt G

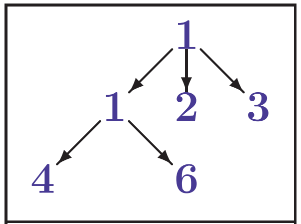
  Konzepte analog für ungerichtete Graphen (Kanten haben keine Richtung)

- **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.

Mathematisches Vokabular: Bäume

- **Tree T: Datenstruktur mit Elementen aus T**
  - x: Einzelknoten wit Wurzel $x \in T$
  - $[x; t_1, .., t_n]$: Baum mit Wurzel x und Unterbäumen $t_1..t_n$
- **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



$[1; [1; 4, 6], 2, 3]$

- **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 $\lceil \log_2(n+1) \rceil$