# Grundlagen der Informatik

**Vorlesungsskript**

Falk Jonatan Strube

Vorlesung von Dr. Boris Hollas

30. Oktober 2015

# Inhaltsverzeichnis

# Allgemeine Informationen

Zugelassene Hilfsmittel Klausur: A-4 Blatt (doppelseitig, handbeschrieben)
Prüfungsvorleistung: alle paar Woche eine Lernabfrage in der Vorlesung (Bestanden wenn insgesamt im Schnitt 50%)
Grundlage der Vorlesung: Grundkurs theoretische Informatik [1]
Lernkontrolle ab 23.10.2015 alle zwei Wochen.

# 1 Aussagenlogik

Mit der Aussagenlogik lassen sich Aussagen formulieren, die entweder wahr oder falsch sind. Aussagen sind atomare Aussagen wie „die Straße ist nass“ oder mit Hilfe von logischen Operatoren zusammengesetzte Aussagen.

## 1.1 Syntax und Semantik

**Definition:** Die *Formeln der Aussagenlogik* sind induktiv definiert.

- Jede atomare Aussage ist eine Formel der Aussagenlogik. Diese heißen Atomformeln oder Variablen. Atomformeln bezeichnen wir mit Kleinbuchstaben oder durch Wörter in Kleinbuchstaben.
- Wenn $F$, $G$ Formeln der Aussagenlogik sind, dann auch $(F \wedge G)$, $(F \vee G)$, $(\neg F)$.

**Bsp.:** Formeln der Aussagenlogik sind $x$, $y$, $x \wedge y$, $(x \wedge (y \wedge z)) \vee (\neg x \wedge (y \wedge \neg z))$, $regnet$, $regnet \wedge nass$, da sie jeweils aus atomaren Aussagen die nach der Definition zusammensetzen lassen bestehen. Keine Formeln der Aussagenlogik sind $x\wedge$, $\vee x$, $x \wedge \vee y$.

Um Klammern zu sparen, legen wir Prioritäten fest:

| Operator | Priorität |
|---|---|
| $\neg$ | höchste |
| $\wedge$, $\vee$ | |
| $\rightarrow$, $\leftrightarrow$ | niedrigste |

**Definition:** Eine *Belegung* einer Formel $F$ der Aussagenlogik ist eine Zuordnung von Wahrheitswerten „wahr“ (1) oder „falsch“ (0) zu den Atomarformeln in $F$. Daraus ergibt sich der *Wahrheitswert* einer Formel:

- Eine Atomformel ist genau dann wahr, wenn sie mit „wahr“ belegt ist.
- Die Formel $F \wedge G$ ist genau dann wahr, wenn $F$ „wahr“ ist und $G$ „wahr“ ist.
  $F \vee G$ ist wahr, wenn $F$ wahr ist oder $G$ wahr.
  $\neg F$ ist wahr, wenn $F$ falsch ist.

| $F$ | $G$ | $F \wedge G$ | $F \vee G$ | $\neg F$ | $F \rightarrow G$ | $F \leftrightarrow G$ |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 | 1 | 1 |

**Bsp.:** Wenn $regnet$ bedeutet: „Es regnet."
Wenn $nass$ bedeutet: „Die Straße ist nass."
Dann bedeutet $regnet \wedge nass$: „Es regnet und die Straße ist nass."

„Wenn es regnet, dann ist die Straße nass" ($regnet \rightarrow nass$). Es muss nur der Fall ausgeschlossen werden, der nicht eintreffen kann: $\neg(regnet \wedge \neg nass) \Rightarrow$ Folgendes darf nicht eintreffen: „Es regnet und die Straße ist nicht nass".

Alles andere („Es regnet nicht und die Straße ist nicht nass", „Es regnet nicht und die Straße ist nass" und „Es regnet und die Straße ist nass") darf eintreffen.

| $regnet$ | $nass$ | $\neg(regnet \wedge \neg nass) = \neg regnet \vee nass$ |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

**Definition:** Die Operatoren $\rightarrow$ (*Implikation*) und $\leftrightarrow$ (*Äquivalenz*) sind definiert durch:

- $F \rightarrow G = \neg F \vee G$
- $F \leftrightarrow G = (F \rightarrow G) \wedge (G \rightarrow F)$

(Siehe Tabelle oberhalb)

**Bsp.:** Berechnen des Betrags $y$ einer Zahl $x$:

```
if (x>= 0)
  y=x;
else
  y=-x;
```

Dargestellt als Formel der Assagenlogik: $((x \geq 0) \rightarrow y = x) \wedge (\neg(x \geq 0) \rightarrow y = -x)$

**Definition:** Eine Formel $F$ der Aussagenlogik heißt

- *erfüllbar*, wenn es eine Belegung gibt, sodass $F$ wahr ist, sonst *unerfüllbar*. Mit $\perp$ bezeichnen wir eine unerfüllbare Formel (Widerspruch).
- *Tautologie* oder *gültig*, wenn $F$ für jede Belegung wahr ist. Bezeichnung: $\top$

**Bsp.:**

- $x \wedge y$ ist erfüllbar.
- $((\neg x \wedge y) \vee (x \wedge \neg y)) \wedge \neg(x \vee y)$ ist unerfüllbar (linke Seite: entweder x oder y falsch - rechte Seite: x oder y falsch)
- $x \vee \neg x$ ist eine Tautologie

**Definition:** Wir schreiben $F \equiv G$ („$F$ ist äquivalent zu $G$"), wenn für jede Belegung gilt: $F \leftrightarrow G$ wahr (d.h., $F \leftrightarrow G$ ist gültig).

### 1.2 Rechenregeln

siehe Mathematik I

## 2 Beweistechniken

ABB21

#### Direkter Beweis

**Bsp.:** Wenn $a \in \mathbb{Z}$ gerade ist, dann ist auch $a^2$ gerade.
$(a \in \mathbb{Z}$ gerade $\Rightarrow a^2$ gerade$)$
*Beweis:*

- Wenn $a$ gerade ist, gibt es ein $n$ mit $a = 2 \cdot n$.
- Dann gilt $a^2 = 4 \cdot n^2 = 2 \cdot 2n^2$,
- woraus $a^2$ gerade folgt.

**Indirekter Beweis** Mit einem indirekten Beweis wird $A \Rightarrow B$ bewiesen, indem die äquivalente Aussage $\neg B \Rightarrow \neg A$ bewiesen wird.

**Bsp.:** Wenn $a^2$ gerade ist, dann auch $a$.
$(a^2$ gerade $\Rightarrow a$ gerade$)$
*Beweis:* Wir zeigen: Wenn $a$ ungerade ist, dann auch $a^2$.

- Aus $a$ ungerade folgt $a = 2n - 1$ für ein $n$.
- Dann ist $a^2 = 4n^2 - 4n + 1 = \underbrace{4(n^2 - n)}_{gerade} + \underbrace{1}_{ungerade}$,
- Aus gerade + ungerade folgt ungerade, woraus $a^2$ ungerade folgt.

**Beweis durch Widerspruch** Mit einem Beweis durch Widerspruch wird eine Aussage $A$ bewiesen, indem gezeigt wird, dass die Annahme „$A$ ist falsch“ zu einem Widerspruch führt.
(D.h., es wird $\neg A \rightarrow \perp$ gezeigt)

**Bsp.:** $\sqrt{2}$ ist irrational. Siehe Mathematik I.

**Vollständige Induktion** Mit einer vollständigen Induktion lassen sich Aussagen der Art „für alle $n \in \mathbb{N}$ gilt . . . “ beweisen.
Prinzip: Gegeben eine Aussage der Form „für alle $n \in \mathbb{N}$ gilt $A(n)$“

- *Induktionsanfang*: Man zeigt, die Wahrheit der Aussage für $n = 1$ (mit anderen Worten: Man zeigt, dass $A(1)$ wahr ist) [1: die kleinste mögliche Zahl $\Rightarrow$ kann auch 0 oder eine andere sein]
- *Induktionsvorussetzung*: Die Aussage ist für $n$ wahr.
- *Induktionsschritt*: Wenn IV wahr ist, dann ist die Aussage auch für $n + 1$ wahr.

In Formeln: Man zeigt

- IA: $A(1)$
- IV: $A(n)$
- IS: für alle $n$: $A(n) \Rightarrow A(n+1)$

ABB22

**Bsp.:** Für alle $n \geq 1$ gilt $\sum_{k=1}^{n} k = \frac{n(n+1)}{2}$

*Beweis (Induktion):*

IA $n = 1$: $1 = \frac{1 \cdot 2}{2}$ ist wahr.

IV Es gelte $\sum_{k=1}^{n} k = \frac{n(n+1)}{2}$ ist wahr.

IS $n \to n+1$: Zu zeigen: $\sum_{k=1}^{n+1} k = \frac{(n+1)(n+2)}{2}$ Es gilt:

$$\begin{aligned}\sum_{k=1}^{n+1} k &= (\sum_{k=1}^{n} k) + n + 1 \\ &\overset{IV}{=} \frac{n(n+1)}{2} + n + 1 \\ &= ... = \frac{(n+1)(n+2)}{2} \#\end{aligned}$$

# 3 Elementare Kombinatorik

Kreuzprodukt:
$A \times B = \{(a,b) | a \in A, b \in B\}$
$A^n = \underbrace{A \times ... \times A}_{n}$

Die *Potenzmenge* einer Menge $M$ ist die Menge aller Teilmengen von $M$: $\mathcal{P}(M) = \{A | A \subseteq M\}$

**Bsp.:** $\mathcal{P}(\{1,2\}) = \{\emptyset, \{1\}, \{2\}, \{1,2\}\}$

**Definition:** Die Mächtigkeit einer Menge $A$ ist die Anzahl ihrer Elemente. Notation: $|A|$

**Satz:** Es gilt $|A^n| = |A|^n$

**Beweis:** Nach Def. ist $A^n = \{(a_1, ..., a_n) | a_1, ..., a_n \in A\}$. Um das n-Tupel $(a_1, ..., a_n)$ zu erzeugen, gibt es $|A|$ viele Möglichkeiten. Insgesamt gibt es daher $|A|^n$ Möglichkeiten das n-Tupel $(a_1, ..., a_n)$ auszuwählen.

**Bsp.:** Eine PIN bestehe aus 6 Ziffern. Mit $A = \{0, ..., 9\}$ ist $A^6$ die Menge aller PINs. Mit obigen Satz folgt: Die Anzahl aller PINs ist $|A^6| = |A|^6 = 10^6$

**Bsp.:** In dem Programm

```
for (i=1 to n)
  for (j=1 to n)
    a[i][j]=i+j;
```

werden alle Paare (i,j) erzeugt. Die Anzahl der Paare ist $|\{1,...,n\}^2| = |\{1,...,n\}|^2 = n^2$. Es gibt daher $n^2$ Schleifendurchläufe.

**Satz:** $|\mathcal{P}(M)| = 2^{|M|}$

**Beweis:** Für $M = \{m_1,...,m_n\}$ identifizieren wir eine Teilmenge $A \subseteq M$ durch das n-Tupel $(a_1,...,a_n)$ mit $a_k \begin{cases} 0 \text{ für } M_k \notin A \\ 1 \text{ für } m_k \in A \end{cases}$. Nach obigen Satz gibt es $|\{0,1\}^n| = 2^n = 2^{|M|}$ derartige Tupel.

**Definition:** Für eine n-elementige Menge ist $\binom{n}{k}$ die Anzahl ihrer k-elementigen Teilmengen $(n \geq k \geq 0)$.

**Bsp.:**

$\binom{n}{0} = 1$, da $\emptyset$ die einzige 0-elementige Teilmenge ist.

$\binom{n}{n} = 1$, da es nur eine n-elementige Teilmenge gibt (die Menge selber).

$\binom{n}{1} = n$, da es $n$ 1-elementige Teilmengen gibt.

$\binom{n}{2} = \frac{n(n-1)}{2}$, denn für das 1. Element gibt es $n$ Möglichkeiten, für das 2. Element $n-1$ Möglichkeiten. Da das Element $\{a,b\} = \{b,a\}$ hierbei doppelt gezählt wird, müssen wir durch 2 teilen.

**Definition:** Eine Permutation der Folge $1,...,n$ ist eine neue Anordnung dieser Folge.

**Bsp.:** Alle Permutationen von $1,2,3$ sind $1,2,3$; $1,3,2$; $2,1,3$; $2,3,1$; $3,1,2$; $3,2,1$.

**Definition:** $n! = 1 \cdot ... \cdot n \qquad 0! = 1.$

**Satz:** Es gibt $n!$ Permutationen von $n$ Zahlen.

**Beweis:** Für die 1. Stelle gibt es $n$ Möglichkeiten, für die 2. Stelle $n-1$ usw. Für die letzte Stelle nur noch eine Möglichkeit. Insgesamt also $n \cdot ... \cdot 1 = n!$ Möglichkeiten.

**Satz:** $\binom{n}{k} = \frac{n!}{k!(n-k)!}$

**Beweis:** Um aus einer $n$-elementigen Menge $k$ Elemente auszuwählen, gibt es $n$ Möglichkeiten, um das erste Element auszuwählen, für das zweite Element $n-1$ Möglichkeiten, ..., für das $k$. Element $n-k+1$ Möglichkeiten, insgesamt daher $n \cdot ... \cdot (n-k+1)$ Möglichkeiten. Da die Reihenfolge, in der diese $k$ Elemente ausgewählt werden, keine Rolle spielt, muss dieses Produkt durch $k!$ geteilt werden.

Daher erhalten wir $\binom{n}{k} = \frac{n \cdot ... \cdot (n-k+1)}{k!} = \frac{n!}{k!(n-k)!}$

# 4 O-Notation

Mit Hilfe der O-Notation lassen sich obere Schranken für die Laufzeit eines Algorithmus angeben (Abschätzung mit $\leq$, die die maximale Laufzeit eines Algorithmus angibt, bspw. $\leq c \cdot n^2$). Um die Laufzeit eines Algorithmus zu messen, bestimmen wir die Anzahl Schritte und geben mit Hilfe der O-Notation deren Größenordnung in Abhängigkeit der Länge der Eingabe an.

**Beispiel:** lineare Suche

```
int lsearch (int a[], int n, int k) {
  int i;
  for (i=0; i<n; i++)
    if ( a[i] == k) return 1; //gefunden
  return 0; // nicht gefunden
}
```

Laufzeit dieser Funktion:

$$\leq \underbrace{c_1 + n \cdot c_2 + c_3}_{g(n)} \overset{\text{Abschätzung}}{\leq} (c_1 + c_2 + c_3) \cdot n = c \cdot n$$

$c_1$ ... Deklarierung von $i$
$c_2$ ... Vergleich der Werte in der Schleife (in $n$ Schleifedurchläufen)
$c_3$ ... Ausführung return
(Durch den Worst-Case von annähernd unendlich vielen Durchläufen spielen die Konstanten, egal wie groß, keine besonder Rolle mehr und können, wie in der Abschätzung zu sehen, zusammengefasst werden).
Die Laufzeit der linearen Suche liegt in $O(n)$

**Definition:** Für eine Funktion $f > 0$ ist $O(f)$ die Menge aller Funktionen $g$, für die gilt:
$g(n) \leq c \cdot f(n)$ für ein $c > 0$ für alle großen $n$.
ABB 31

**Bsp.:** $2n^3 - n + 5 \overset{1.)}{\leq} 2n^3 + 5 \leq 7n^3 \in O(n^3)$

1.) $-n$ ist kleiner Null, deswegen ist die rechte Seite ohne $-n$ nachgewiesener Maßen größer (Vorgehensweise Ungleichung aufstellen (siehe auch folgende): weg lassen, was kleiner Null ist; mit $n^x$ o.ä. erweitern, um auszuklammern).

**Bsp.:**

```
for (i=0; i<n-1; i++)
  for (j=i+1 ; j<n ; j++)
    if( a[i] == a[j] ) return 1;
return 0;
```

Die if-Anweisung wird höchstens $\binom{n}{2}$ mal durchlaufen. Die Laufzeit ist daher $\leq c_1 \cdot \binom{n}{2} + c_2 \leq (c_1+c_2)\cdot\binom{n}{2} = \frac{c_1+c_2}{2}\cdot n\cdot(n-1) \leq \frac{c_1+c_2}{2}\cdot n^2 \in O(n^2)$ (mit $c_i$ ... Zeiteinheiten für Rechenaufwand).

**Bsp.:** $2 \cdot log(n^2+1)$
$\leq 2 \cdot log(n^2(1+1))$
$= 2log(2n^2) = 2(log(2) + log(n^2))$
$\leq 2(log(n) + 2log(n))$
$\leq 6log(n) \in O(log(n))$
Schneller mit:
$n^2 + 1 \leq n^3 \Leftrightarrow$
$\frac{1}{n} + \frac{1}{n^3} \leq 1 \Rightarrow$
$0 \leq 1$ für $n \to \infty$
$2 \cdot log(n^2+1) \leq 2log(n^3) = 6log(n) \in O(log(n))$

## Literatur

[1] Boris Hollas. *Grundkurs Theoretische Informatik mit Aufgaben und Prüfungsfragen*. Spektrum Akademischer Verlag, 2007.