Diskrete Strukturen und Logik

WiSe 2007/08 in Trier

Henning Fernau
Universität Trier
fernau@uni-trier.de

Diskrete Strukturen und Logik

Gesamtübersicht

- Organisatorisches
- Einführung
- Logik & Mengenlehre
- Beweisverfahren
- Kombinatorik: Die Kunst des Zählens
- algebraische Strukturen

Organisatorisches

Vorlesungen MO 10.15-11.45 im HS 12
DI 12.25-13.55 im HS 13

Übungsbetrieb in Form von zwei Übungsgruppen
BEGINN: in der zweiten Semesterwoche MO 16-18, HZ 202; MI 14-16, HZ 204

Dozentensprechstunde DO 9-10 in meinem Büro H 410 (4. Stock)

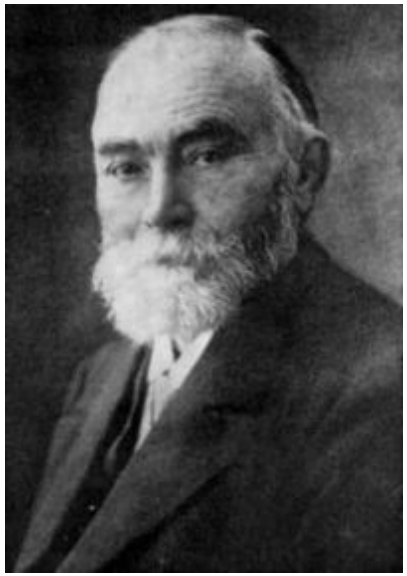
Mitarbeitersprechstunde (Stefan Gulan) DO 16-17 H 413

Tutorensprechstunde DO 9-10 & 16-17 H 412

Logische Ahnengalerie

Aristoteles

Leibniz



Frege

Boole

Grundlagen Logik

Aussagenlogik

Eine *Aussage* ist ein Satz, der entweder **w**ahr oder **f**alsch ist.
w und f heißen auch *Wahrheitswerte*.

Beispiel: 5 ist eine Primzahl.

Beispiel: $\sqrt{7}$ ist rational.

Beispiel: Dieser Satz ist falsch. (*Antinomie* von Russel)

Aussagenlogik

Eine *Aussageform* (oder *Prädikat*) ist ein Satz mit wenigstens einer *Leerstelle* (oder *Variable*n), welcher durch Ersetzen in die Leerstelle zu einer Aussage wird. Die Gesamtheit X der Objekte, die in eine Leerstelle x eingesetzt werden dürfen, muss (vorher) bekannt sein; X heißt auch *Universum*.

Beispiel: "x ist durch drei teilbar." ist eine Aussageform:
Für jede ganze Zahl x ergibt sich eine Aussage.
Abhängig von der *Belegung* ist die Aussage wahr oder falsch.

Beispiel: "$(x+y)^2 = x^2 + 2xy + y^2$" ist eine Aussageform:
Für beliebige reelle x, y ergibt sich eine wahre Aussage.

Prädikatenlogik (ein erster Hinweis)

Ist $A(x)$ eine Aussageform (mit Universum X und Leerstelle x), so kann man auch Aussagen gewinnen durch Konstruktionen wie:

- Für alle x gilt: $A(x)$.
- Für irgendein x gilt: $A(x)$.
- Für mindestens zwölf x gilt: $A(x)$.

Dat kriege mer später . . .

Aussagenlogik: Verknüpfung von Aussagen. Es sei p eine Aussage.

Negation, *Verneinung*: "nicht p"

Schreibweise: $\neg\mathrm{p}$, $\bar{\mathrm{p}}$, $\tilde{\mathrm{p}}$

$\neg\mathrm{p}$ ist wahr genau dann, wenn p falsch ist.

Wahrheitstafel:

p	$\neg\mathrm{p}$
w	f
f	w

Beispiel: p sei: "3 ist eine Primzahl."
$\neg\mathrm{p}$ bedeutet: "Es gilt nicht, dass 3 eine Primzahl ist:"
oder kürzer: "3 ist keine Primzahl."

Aussagenlogik: Verknüpfung von Aussagen. Es seien p, q Aussagen.

Konjunktion, *Und-Verknüpfung*: "p und q"

Schreibweise: $p \wedge q$, $p \cdot q$, pq, $p\&q$

$p \wedge q$ ist wahr genau dann, wenn p und q wahr sind.

Wahrheitstafel:

p	q	$p \wedge q$
w	w	w
w	f	f
f	w	f
f	f	f

Beispiel: p sei: "11 ist eine Primzahl."
q sei: "11 ist durch drei teilbar."
$p \wedge q$ bedeutet: "11 ist eine durch drei teilbare Primzahl."

Aussagenlogik: Verknüpfung von Aussagen. Es seien p, q Aussagen.

Disjunktion, *Oder-Verknüpfung*: "p oder (auch) q"

Schreibweise: $\mathrm{p} \vee \mathrm{q}$, $\mathrm{p} + \mathrm{q}$, $\mathrm{p}|\mathrm{q}$ Achtung Römer: VEL.

$\mathrm{p} \vee \mathrm{q}$ ist wahr genau dann, wenn p oder auch q wahr ist.

Wahrheitstafel:

p	q	$\mathrm{p} \vee \mathrm{q}$
w	w	w
w	f	w
f	w	w
f	f	f

Beispiel: p sei: "11 ist eine Primzahl."
q sei: "11 ist durch drei teilbar."
$\mathrm{p} \vee \mathrm{q}$ bedeutet: "11 ist eine Primzahl oder durch drei teilbar."

Rechengesetze der Aussagenlogik

Satz: Konjunktion und Disjunktion sind kommutativ.

Dies bedeutet für beliebige Aussagen p, q:

- $\mathrm{p} \wedge \mathrm{q}$ ist wahr genau dann, wenn $\mathrm{q} \wedge \mathrm{p}$ wahr ist.

- $\mathrm{p} \vee \mathrm{q}$ ist wahr genau dann, wenn $\mathrm{q} \vee \mathrm{p}$ wahr ist.

Wie beweist man eine solche Aussage ?
$\rightsquigarrow$ vollständige Fallunterscheidung $\rightsquigarrow$ Wahrheitstafelmethode

Rechengesetze der Aussagenlogik

Satz: Konjunktion und Disjunktion sind assoziativ.

Dies bedeutet für beliebige Aussagen $\mathrm{p}, \mathrm{q}, \mathrm{r}$:

- $(\mathrm{p} \wedge \mathrm{q}) \wedge \mathrm{r}$ ist wahr genau dann, wenn $\mathrm{p} \wedge (\mathrm{q} \wedge \mathrm{r})$ wahr ist.

- $(\mathrm{p} \vee \mathrm{q}) \vee \mathrm{r}$ ist wahr genau dann, wenn $\mathrm{p} \vee (\mathrm{q} \vee \mathrm{r})$ wahr ist.

Bei “längeren” $\wedge$ (oder auch $\vee$) Ausdrücken können wir Klammern “einsparen”.

Beweis wiederum durch Wahrheitstafelmethode.

Rechengesetze der Aussagenlogik: Kombination verschiedener Verknüpfungen.

Satz: Es gelten folgende Distributivgesetze für Konjunktion und Disjunktion:
(1) $(p \wedge q) \vee r$ und $(p \vee r) \wedge (q \vee r)$ haben stets denselben Wahrheitswert.
(2) $(p \vee q) \wedge r$ und $(p \wedge r) \vee (q \wedge r)$ haben stets denselben Wahrheitswert.

Satz: Regeln zur Behandlung der Negation:
(1) $(\neg\neg p)$ und p haben stets denselben Wahrheitswert.
(2) $\neg(p \vee q)$ und $(\neg p) \wedge (\neg q)$ haben stets denselben Wahrheitswert.
(3) $\neg(p \wedge q)$ und $(\neg p) \vee (\neg q)$ haben stets denselben Wahrheitswert.

(2) und (3) heißen auch *Gesetze von de Morgan*.

Logische Operationen in Programmiersprachen

Frage: Sind die *aussagenlogischen Verknüpfungen* (auch *Junktoren* genannt) "dasselbe" wie die die *logischen Operatoren* in Programmiersprachen ?

JEIN. . .
JA bei seiteneffektfreien Programmen
NEIN im Allgemeinen z.B. in C-Dialekten: aus Effizienzgründen wird bei $x||y$ y nicht "ausgeführt", falls x schon als wahr bekannt ist.
Damit ist die Wirkung von $x||y$ möglicherweise von der von $y||x$ verschieden.

Aussagenlogik: Weitere *Junktoren*

Implikation, Schreibweise: $\mathrm{p} \implies \mathrm{q}$, $\mathrm{p} \rightarrow \mathrm{q}$ genau dann, wenn $(\neg \mathrm{p}) \vee \mathrm{q}$.
Sprechweisen: p ist *hinreichende Bedingung* oder *Prämisse* für q, q ist *notwendige Bedingung* oder *Konklusion* für p; oder kurz: “Wenn p, dann q.” oder “Nur wenn q, dann p.”

(Logische) Äquivalenz, Schreibweise: $\mathrm{p} \Leftrightarrow \mathrm{q}$, $\mathrm{p} \leftrightarrow \mathrm{q}$
Dies gilt genau dann, wenn $(\mathrm{p} \implies \mathrm{q}) \wedge (\mathrm{q} \implies \mathrm{p})$.

Wie lauten also die Wahrheitstafeln ?

Die *(materiale)* **Implikation** — ein schwieriger Junktor...

Beispiel:

- Dass es regnet, ist eine hinreichende Bedingung dafür, dass die Straße nass ist.
- Schon wenn es regnet, ist die Straße nass.
- Wenn es regnet, dann ist die Straße nass.
- Nur wenn die Straße nass ist, regnet es.

Die Lesart "wenn...dann" ist insofern problematisch, als man in der natürlichen Sprache vor allem *inhaltliche Zusammenhänge* oder *zeitliche Nähe* dadurch ausdrückt.
All das macht die materiale Implikation **nicht**, sie nennt nur den *formalen Zusammenhang*.
Zur Frage, warum das eine hinreichende Bedingung ist — ob auf Grund eines kausalen Zusammenhangs oder auch nur rein zufällig —, nimmt die materiale Implikation nicht Stellung.

Als *Umkehrschluss* (oder *Kontraposition*) bezeichnet man
den Schluss von $\mathrm{p} \Rightarrow \mathrm{q}$ auf $(\neg \mathrm{q}) \Rightarrow (\neg \mathrm{p})$. Für das Beispiel bedeutet das:
Wenn die Straße nicht nass ist, dann regnet es nicht.
Nur wenn es nicht regnet, ist die Straße nicht nass.

Umgangssprachlich lässt man sich gelegentlich zu weiteren — falschen — Aussagen verleiten, z.B.:
Weil es nicht regnete, kann die Straße nicht nass sein.
Diese Folgerung ist falsch, da die Straße auch aus anderen Gründen nass werden kann (Rohrbruch, Übung der Feuerwehr ...).

Also: Wenn die Folgerung $\mathrm{p} \Rightarrow \mathrm{q}$ wahr ist, dann erhält man aus der Aussage $\neg \mathrm{p}$
keine Aussage über q; q kann wahr oder falsch sein.
(“Ex falso (sequitur) quodlibet.” — “Aus Falschem folgt Beliebiges.”)

Eine Anwendung: Überprüfung der Korrektheit von Programmen

Frage: Was berechnet das folgende Programmstück (in Pseudo-Code) ?

1. Lies ganze Zahlen v, x, y, z ein.

2. Setze $u := vx$. (u ist eine Hilfsvariable für ganze Zahlen.)

3. Setze $u := (u + y)x$.

4. Setze $u := u + z$.

5. Gib u aus.

Eine Anwendung: Überprüfung der Korrektheit von Programmen

Methode: Einfügen von *Eigenschaften* (auch *Prädikate* genannt), die zu gewissen Zeitpunkten wahr sind.

1. Lies ein ganze Zahlen v, x, y, z ein.
   { v, x, y, z sind ganze Zahlen. } *Vorbedingung* P

2. Setze $u := vx$. (u ist eine Hilfsvariable für ganze Zahlen.)
   { u enthält die ganze Zahl vx. }

3. Setze $u := (u + y)x$.
   { u enthält die ganze Zahl $vx^2 + yx$.}

4. Setze $u := u + z$.
   { u enthält die ganze Zahl $vx^2 + yx + z$. } *Nachbedingung* Q

5. Gib u aus.

Eine Anwendung: Überprüfung der Korrektheit von Programmen

Verzweigungen enthalten weitere Bedingungen.

1. Lies eine ganze Zahl x ein. y ist eine ganzzahlige Variable.

2. **Wenn** $x \geq 0$, **so** setze $y := x$.

3. **Andernfalls** setze $y := -x$.

4. Gib y aus.

Eine Anwendung: Überprüfung der Korrektheit von Programmen

1. Lies eine ganze Zahl x ein. y ist eine ganzzahlige Variable.
   P: { x ist eine ganze Zahl, y eine Variable für ganze Zahlen. }

2. **Wenn** $x \geq 0$, **so** setze $y := x$.
   $\{ (P \wedge (x \geq 0)) \Rightarrow y = x. \}$

3. **Andernfalls** setze $y := -x$.
   $\{ (P \wedge (x < 0)) \Rightarrow y = -x. \}$

4. Gib y aus.
   Q: { Der Betrag von x wird ausgegeben, d.h., $y = |x|$. }