Grundlagen Theoretischer Informatik I

SoSe 2011 in Trier

Henning Fernau
Universität Trier
fernau@uni-trier.de

Grundlagen Theoretischer Informatik I Gesamtübersicht

- Organisatorisches; Einführung
- Logik & Beweisverfahren
- Mengenlehre
- reguläre Sprachen

Organisatorisches

Vorlesungen FR 10.10-11.50 im HS 13
FR 12.30-13.50 im HS 13

Übungsbetrieb BEGINN: FR 29.04.2011

Dozentensprechstunde DO 13-14 in meinem Büro H 410 (4. Stock)

Mitarbeitersprechstunde (Daniel Meister) DO 13-14 H 413

Tutorensprechstunde MO 13-14 H 407

Besonderheit heute

Wir werden zwei Vorlesungseinheiten “en bloc” halten.

Da ich ab 13.15 mit anderen Verpflichtungen eingespannt bin, werde ich 10.15 bis 13.10 durchgehend Vorlesung machen.

Keine Angst: Es kommen hoffentlich genügend Auflockerungen dazwischen.

In der nächsten Woche fallen Vorlesungen und Übungen aus (Karfreitag).

Prüfungsvorleistungen

- Hausaufgabenpunkte $\geq$ 40% UND
- mind. einmal erfolgreich vorgerechnet UND
- Zwischenklausur bestanden.

Eine (und höchstens eine) der Prüfungsvorleistungen kann durch eine erfolgreich bestandene kleine mündliche Prüfung ersetzt werden.

Abgabe von Hausaufgaben

Diese sollte in Gruppen zu 1-3 Personen erfolgen.

Abgabeschluss ist im Allgemeinen mittwochs bis 16 Uhr, im mit GTI1 beschrifteten Kasten im 4. Stock vor dem gemeinsamen Sekretariat von Prof. Näher und mir.

Verspätete Abgaben gelten als nicht abgegeben und werden dementsprechend mit null Punkten bewertet.

Die Lösungen sind handschriftlich anzufertigen; weder Schreibmaschinen- noch Computerausdrucke werden akzeptiert, erst recht keine Kopien.

In der Regel werden die korrigierten Hausaufgaben kurz vor den passenden Übungen (freitags) zurückgegeben.

Vorrechnen von Hausaufgaben

Der Übungsleiter sucht sich Leute zum Vorrechnen heraus.
Erfolgreiches Vorrechnen ist eine der Prüfungsvorleistungen.
Erfolgloses Vorrechnen führt dazu, dass das gesamte Übungsblatt mit null Punkten für die betreffende Person in die Wertung gerechnet wird.
Wird eine nicht anwesende Person zum Vorrechnen bestimmt, so gilt ihr Vorrechnen als erfolglos.

Es werden keine Musterlösungen ausgegeben.
Bitte Mitschreiben und Mitdenken!

Probleme ? Fragen ?

Klären Sie bitte Schwierigkeiten mit Vorlesungen oder Übungen möglichst umgehend in den zur Verfügung gestellten Sprechzeiten.

In der Tutorensprechstunde stehen Ihnen (alternierend) Studenten höherer Semester zu Rückfragen bereit.

Wir helfen Ihnen gerne!
. . . wir sind aber keine Hellseher, die Ihnen Ihre Schwierigkeiten an der Nasenspitze ansehen. . .

Einführung 1: Historisches

Entstehung des Vorlesungszyklus GTI aus Teilen von:

—Diskrete Strukturen und Logik (DSL)

—Rechnerstrukturen I

—Automaten und Formale Sprachen (AFS)

—Berechenbarkeit und Komplexität (BK)

Einführung 1: Historisches

Entstehung der Vorlesung GTI-1 aus Teilen von:

—Diskrete Strukturen und Logik (DSL)

—Rechnerstrukturen I

—Automaten und Formale Sprachen (AFS)

—Berechenbarkeit und Komplexität (BK)

“Struktur”

Mathematik und Informatik als Strukturwissenschaften

“Softwareengineering” bedeutet immer auch:

- Beschreibe das Anwendungsproblem genau.
- Abstrahiere von unnötigen Einzelheiten.
- Erkenne statt dessen das Wesentliche (die Struktur).
- Formalisiere das Wesentliche, um die Aufgabe in ein Programm überführen zu können, das auch das leistet, was der Anwender möchte.

Grundverständnis der Vorlesung

Die Vorlesung beschäftigt sich mit den **absoluten Grundbegriffen der Informatik** als “Computer Science:”

- liefert Grundlagen für (fast) alle weiteren Vorlesungen Ihres Informatik- oder Informatik-nahen Studiums

- soll Ihnen die “Angst” vor mathematischen Formalismen nehmen: Mathematik als “Alltagssprache” und Bindeglied zwischen Informatikern und den meisten Anwendungsdisziplinen.

Einführung 2: Literatur

Aufgrund der Neukonzeption gibt es kein einzelnes Lehrbuch. Die DSL-Inhalte (und einiges mehr) finden Sie in:

C. Meinel, M. Mundhenk: Mathematische Grundlagen der Informatik. Mathematisches Denken und Beweisen, eine Einführung. Teubner, 2002.

Alternative: K. A. Ross, C. R. B. Wright: Discrete Mathematics. Prentice Hall, 1988.

Die automatentheoretischen Aspekte sind in “beliebigen” Lehrbüchern zu diesem Thema enthalten. Zwei Beispiele folgen:

E. Kinber / C. Smith: Theory of Computing. A Gentle Introduction. Prentice Hall.

U. Schöning: Theoretische Informatik kurz gefasst. BI / Spektrum.

Es gibt zahlreiche gute Skripten im Internet, z.B.: Skripten zur Vorlesung Informatik-B2 an der Universität Duisburg-Essen (Prof. Luther, Prof. Hertling)

Es gibt ungezählte Einführungen in die Thematik; finden Sie am besten eine für Sie am besten geeignete heraus !

$\rightsquigarrow$ nicht nachprüfbare Hausaufgabe:
Gehen Sie in die Bibliothek (leider zweigeteilt in Trier) oder in eine Buchhandlung und beginnen Sie, in verschiedenen Büchern zu lesen;
wenigstens zwei davon sollten Sie über die Vorlesung begleiten.

Einführung 3: Wie höre ich mathematische / theoretische Vorlesungen ?

Das Wichtigste: Bleiben Sie auf dem Laufenden !

Wie Sie schon bemerkt haben werden: Unimathematik ist “schneller” als Schulmathematik

Rechnen Sie mindestens die Zeit, die Sie in den Vorlesungen und Übungen verbringen (sollten), für die Nachbereitung der Vorlesungen und Übungen ein, zusätzlich zu der Zeit für die Übungsbearbeitung.

Stellen Sie Fragen, bevor wir das tun !

Grundlagen Theoretischer Informatik I Gesamtübersicht

- Organisatorisches; Einführung
- Logik & Beweisverfahren
- Mengenlehre
- reguläre Sprachen

Nochmals Einordnung

Logik und Mengenlehre dienen per se als die wesentlichen Grundlagen der Mathematik.

Wir werden in beiden Fällen einen eher naiven, sozusagen anwendungsorientierten Zugang anbieten.

Beweisverfahren lassen sich aus der Logik heraus begründen.

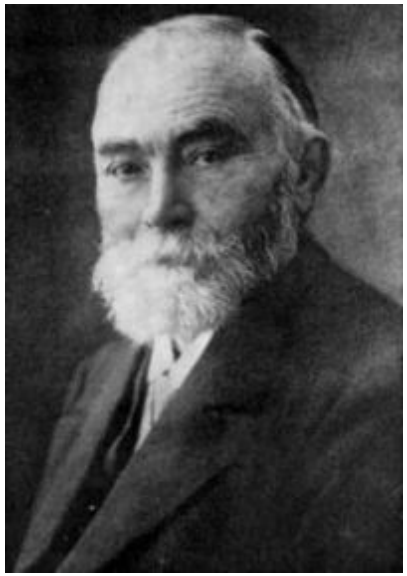
Mit der Trennung von Syntax und Semantik lernen Sie in der Logik bereits eines der Grundprinzipien der Informatik kennen.

Weiter sehen Sie, wie Rekursion und Induktion verschwistert sind und somit die Induktion das für die Informatik wesentliche Beweisprinzip ist.

Logische Ahnengalerie

Aristoteles

Leibniz



Frege

Boole

Grundlagen Logik

Aussagenlogik

Intuitive Festlegung: Eine *Aussage* ist ein Satz, der entweder **w**ahr oder **f**alsch ist. w und f heißen auch *Wahrheitswerte*.
„Halbwahrheiten“ gibt es nicht.

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 . . .

Syntax der Aussagenlogik

- *Atomare Aussagen* von der Form A_i, $i \in \mathbb{N}$, auch *Aussagevariablen* genannt, sind Aussagen.
- Sind p und q Aussagen, so auch (1) $(p \wedge q)$ (*Konjunktion*), (2) $(p \vee q)$ (*Disjunktion*) und (3) $\neg p$ (*Negation*).
- Nichts Weiteres sind Aussagen; Aussagen heißen manchmal auch *Formeln*.

Beispiele:
$((A_1 \wedge \neg A_3) \vee A_2)$ ist eine Aussage. Warum?
$(A_3 \wedge A_7$ ist keine Aussage (falsche Klammerstruktur).
$A_1 \vee A_4 \vee A_7$ ist keine Aussage (Klammern fehlen).
Wir werden noch sehen, dass wir im letzteren Fall (zum Beispiel) nicht so stringent sein müssen.

Bemerkungen zu Syntax und Semantik

- Hier haben wir ein erstes Beispiel einer rekursiven Definition:
  Entlang dieser formalen Definition müssen wir auch zeigen, dass gewisse Zeichenfolgen Aussagen bilden.
  Beispiel: A_1 und A_2 und A_3 sind atomare Aussagen.
  Daher ist auch $\neg A_3$ eine Aussage sowie dann auch $(A_1 \wedge \neg A_3)$.
  Also ist auch $((A_1 \wedge \neg A_3) \vee A_2)$ eine Aussage.

- Die *Syntax*, also der *Satzbau*, ist eine rein formale Angelegenheit, die den so geformten Zeichenfolgen (zunächst) keinerlei Sinn zuweist.
  Die Bedeutung ergibt sich erst durch eine (ebenso rekursive) Definition der *Semantik*.

- Diese Trennung von Syntax und Semantik ist ein Charakteristikum formaler Sprachen, eine der formalen Grundlagen der Informatik.

Noch mehr Bemerkungen zur Syntax

Anhand der rekursiven Definition der Syntax der Aussagenlogik kann man auch ein weiteres wichtiges theoretisches Konzept in der Informatik sehen: eine sogenannte *Baumstruktur*.

Dieses bildet die rekursive Definition graphisch ab.

Die Kreuzungspunkte dieses Objekts entsprechen den *inneren Knoten*, und die diese verbindenen Linien heißen *Kanten*.

(Beispiel $((A_1 \wedge \neg A_3) \vee A_2)$ an der Tafel.)

Beachte: “Baumstruktur”, wenn man das Objekt “auf den Kopf stellt”.

Die inneren Knoten entsprechen den Operatoren (evtl. Spezialfall “Wurzel”):

Die binären Operatoren $\vee$ und $\wedge$ haben zwei nach unten zeigende Kanten, der unäre Operator $\neg$ nur eine.

Die unteren Kantenenden, welche keine Fortsetzung haben, heißen auch *Blätter*; sie entsprechen den atomaren Aussagen.

Semantik der Aussagenlogik

. . . führt hin zum intuitiven Begriff (s.o.)

Es sei $\mathcal{A}$ eine Menge von atomaren Aussagen. Eine *Belegung(sfunktion)* $\beta_{\mathcal{A}}$ ordnet $A_i \in \mathcal{A}$ einen *Wahrheitswert* zu, i.Z.: $\beta_{\mathcal{A}} : \mathcal{A} \to \{w, f\}$.
Sei jetzt allgemeiner $\mathcal{F}(\mathcal{A})$ die Gesamtheit aller Formeln, in denen lediglich atomare Aussagen aus $\mathcal{A}$ vorkommen.

Wir erweitern $\beta_{\mathcal{A}}$ zu einer Belegung(sfunktion) $\beta : \mathcal{F}(\mathcal{A}) \to \{w, f\}$ wie folgt:
Ist $A \in \mathcal{F}(\mathcal{A})$ eine atomare Aussage, so ist $\mathcal{A} = \{A\}$; setze $\beta(A) = \beta_{\mathcal{A}}(A)$.
Ist $A = (B \vee C)$, so setze $\beta(A) = f$ genau dann, wenn sowohl $\beta(B) = f$ und $\beta(C) = f$. (Andernfalls sei $\beta(A) = w$.)
Ist $A = (B \wedge C)$, so setze $\beta(A) = w$ genau dann, wenn sowohl $\beta(B) = w$ und $\beta(C) = w$. (Andernfalls sei $\beta(A) = f$.)
Ist $A = \neg B$, so setze $\beta(A) = w$ genau dann, wenn $\beta(B) = f$.
Damit ist $\beta(A)$ für jede Aussage $A \in \mathcal{F}(\mathcal{A})$ definiert.

Die Berechnung lässt sich wieder gut am Rekursionsbaum nachvollziehen (Tafel).

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

Negation, *Verneinung*: "nicht p"

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

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

Wahrheitstafel:

p	$\neg p$
w	f
f	w

Beispiel: p sei: "3 ist eine Primzahl."
$\neg p$ 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: $p \vee q$, $p + q$, $p|q$ Achtung Römer: VEL.

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

Wahrheitstafel:

p	q	$p \vee q$
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."
$p \vee q$ bedeutet: "11 ist eine Primzahl oder durch drei teilbar."

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: $p \implies q$, $p \to q$ genau dann, wenn $(\neg p) \vee q$.
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: $p \Leftrightarrow q$, $p \leftrightarrow q$
Dies gilt genau dann, wenn $(p \implies q) \wedge (q \implies p)$.
Sprechweisen: “p genau dann, wenn q.”; “p ist äquivalent zu q.”

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.”)

Modelle und Erfüllbarkeit

Es sei F eine Formel mit $\mathcal{A}$ als zugehöriger Menge atomarer Aussagen.
Gibt es eine Belegung $\beta : \mathcal{F}(\mathcal{A}) \to \{w, f\}$ mit $\beta(F) = w$, so schreibt man auch: $\beta \models F$ und spricht: "F gilt unter der Belegung β." oder "β ist ein *Modell* für F."
Gibt es eine Belegung $\beta : \mathcal{F}(\mathcal{A}) \to \{w, f\}$ mit $\beta(F) = f$, so schreibt man auch: $\beta \not\models F$ und spricht: "F gilt nicht unter der Belegung β.", "β ist kein Modell für F."
F ist *erfüllbar*, wenn sie mindestens ein Modell besitzt; andernfalls heißt sie unerfüllbar.
F heißt *(allgemein)gültig* oder eine *Tautologie*, falls F unter jeder Belegung $\beta : \mathcal{F}(\mathcal{A}) \to \{w, f\}$ gilt. Wir notieren dies durch $\models F$.

Satz: F ist Tautologie genau dann, wenn $\neg\mathrm{F}$ unerfüllbar ist.

Beweis: Es sei F eine Formel mit $\mathcal{A}$ als zugehöriger Menge atomarer Aussagen.

F ist Tautologie gdw.

F gilt unter jeder Belegung $\beta : \mathcal{F}(\mathcal{A}) \to \{w, \mathrm{f}\}$ gdw.

Für jede Belegung $\beta : \mathcal{F}(\mathcal{A}) \to \{w, \mathrm{f}\}$ gilt: $\beta(\mathrm{F}) = w$ gdw.

Für jede Belegung $\beta : \mathcal{F}(\mathcal{A}) \to \{w, \mathrm{f}\}$ gilt: $\beta(\neg\mathrm{F}) = \mathrm{f}$ gdw.

Es gibt keine Belegung $\beta : \mathcal{F}(\mathcal{A}) \to \{w, \mathrm{f}\}$ mit $\beta(\neg\mathrm{F}) = w$ gdw.

$\neg\mathrm{F}$ besitzt kein Modell gdw.

$\neg\mathrm{F}$ ist unerfüllbar.

Das Erfüllbarkeitsproblem

Die folgende Frage ist eine der zentralen Fragen in der Informatik:
Wie schwierig ist es, bei einer vorgelegten Formel F herauszubekommen, ob sie erfüllbar ist?

Es sei F eine Formel mit $\mathcal{A}$ als zugehöriger Menge atomarer Aussagen.
Jede atomare Aussage kann (unabhängig voneinander) wahr oder falsch sein.
Enthält $\mathcal{A}$ n atomare Aussagen, so wäre ein naheliegendes Verfahren, all 2^n möglichen Belegungen β zu betrachten, um zu sehen, ob $\beta(F) = w$ gilt.
Das benötigt offensichtlich viel Zeit, selbst wenn n recht klein ist, z.B. $n = 1000$.

Man bekommt 1 Million Dollar für die Lösung der Frage, ob es einen Algorithmus gibt, dessen Laufzeit nur polynomiell von der Größe von F abhängt.

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.
  D.h.: $\models (\mathrm{p} \wedge \mathrm{q}) \iff (\mathrm{q} \wedge \mathrm{p})$.

- $\mathrm{p} \vee \mathrm{q}$ 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*.

Beispiele für stets wahre "Aussageverbindungen" (*Tautologien*)

1. $p \vee \neg p$ (Satz vom ausgeschlossenen Dritten)
2. $(p \Rightarrow \neg p) \Rightarrow \neg p$
3. $(p \wedge (p \Rightarrow q)) \Rightarrow q$ (Modus Ponens)
4. $(p \wedge q) \Rightarrow p$, $(p \wedge q) \Rightarrow q$
5. $p \Rightarrow (p \vee q)$, $q \Rightarrow (p \vee q)$
6. $(p \Rightarrow q) \Rightarrow [(q \Rightarrow r) \Rightarrow (p \Rightarrow r)]$ (Schlusskette)
7. $((p \Rightarrow q) \wedge (q \Rightarrow r) \wedge p) \Rightarrow r$
8. $(p \iff q) \iff (\neg p \iff \neg q)$.

Hinweis: Beweisverfahren fußen oft auf Tautologien.

Beweise für Tautologien Z.B. 6. mit (kombinierter, gekürzter) Wahrheitstafel

p	q	r	$A := p \Rightarrow q$	$B := q \Rightarrow r$	$C := p \Rightarrow r$	$D := B \Rightarrow C$	$A \Rightarrow D$
w	w	w	w	w	w	w	w
w	w	f	w	f	f	w	w
w	f	?	f	?	?	?	w
f	w	w	w	w	w	w	w
f	w	f	w	f	w	w	w
f	f	?	w	w	w	w	w

“Abkürzungsregel:” Ist die Prämisse falsch, so ist die Implikation wahr.

Logische Äquivalenz von Aussagen

Zwei Aussagen p und q mit demselben Vorrat an atomaren Aussagen heißen *logisch äquivalent*, falls die Aussage $\mathrm{p} \iff \mathrm{q}$ eine Tautologie ist.

Schreibweise: $\mathrm{p} \equiv \mathrm{q}$

Viele bislang gefundene Rechenregeln und Tautologien lassen sich so schreiben (s. nächste Folie).

Logische Äquivalenzen — Beispiele

1. Kommutativgesetze: $(p \wedge q) \equiv (q \wedge p)$; $(p \vee q) \equiv (q \vee p)$.
2. Assoziativgesetze: $(p \wedge (q \wedge r)) \equiv ((p \wedge q) \wedge r)$; $(p \vee (q \vee r)) \equiv ((p \vee q) \vee r)$.
3. Distributivgesetze: $(p \wedge (q \vee r)) \equiv ((p \wedge q) \vee (p \wedge r))$; $(p \vee (q \wedge r)) \equiv ((p \vee q) \wedge (p \vee r))$.
4. Gesetze von de Morgan: $\neg(p \wedge q) \equiv (\neg p \vee \neg q)$; $\neg(p \vee q) \equiv (\neg p \wedge \neg q)$.
5. Identitätsgesetze: $(p \vee f) \equiv p$, $(p \wedge w) \equiv p$.
6. Umkehrschluss: $(p \Rightarrow q) \equiv (\neg q \Rightarrow \neg p)$.
7. Auflösen der Implikation: $(p \Rightarrow q) \equiv (\neg p \vee q)$.
8. Auflösen der Äquivalenz: $(p \iff q) \equiv ((p \Rightarrow q) \wedge (q \Rightarrow p)) \equiv (p \wedge q) \vee (\neg p \wedge \neg q)$.

Logische Äquivalenzen — Beweise

1. Wahrheitstafelbeweis (vollständige Fallunterscheidung), z.B. für Umkehrschluss

2. Ausnutzung bekannter Tautologien (also durch Umformungen), z.B. für 8b:

$$
\begin{array}{lll}
(p \iff q) & \equiv^{8a} & ((p \Rightarrow q) \wedge (q \Rightarrow p)) \\
& \equiv^{7} & ((\neg p \vee q) \wedge (\neg q \vee p)) \\
& \equiv^{3} & ((\neg p \vee q) \wedge \neg q) \vee ((\neg p \vee q) \wedge p) \\
& \equiv^{1,3} & ((\neg p \wedge \neg q) \vee (q \wedge \neg q)) \vee ((\neg p \wedge p) \vee (q \wedge p)) \\
& \equiv^{1,2,5} & (p \wedge q) \vee (\neg p \wedge \neg q)
\end{array}
$$

Formal müssen wir das noch rechtfertigen!

Anmerkungen

1. Ein Wahrheitstafelbeweis für $\mathrm{p} \equiv \mathrm{q}$ ist nichts anderes als das Nachrechnen von $\models (\mathrm{p} \iff \mathrm{q})$ durch Betrachten sämtlicher Belegungen.

2. Umformungen bedeuten hingegen das Ersetzen von Teilformeln durch äquivalente. Dass dies eine korrekte Schlussweise ist, erscheint intuitiv offenkundig, bedarf aber eines Beweises, den wir im Folgenden nachliefern.

Diese zweite Methode legt ein mechanisches Schließen jenseits der Betrachtung von Belegungen nahe.
Tatsächlich gibt es eine Vielzahl solcher sogenannter *Kalküle*.
Dies führt (im Gegensatz zum semantischen Folgerungsoperator $\models$) auf einen eher syntaktischen Ableitungsbegriff $\vdash$.
So ein Kalkül heißt *korrekt*, falls $\vdash$ nur semantisch korrekte Folgerungen zulässt.
Ist jede semantische Folgerung auch ableitbar, so heißt der Kalkül *vollständig*.

Teilformel

Es sei p eine Aussage.
Ist p atomar, so ist p die einzige Teilformel von p.
Ist $\mathrm{p} = \neg \mathrm{q}$, so sind (neben p selbst) sämtliche Teilformeln von q auch Teilformeln von p.
Ist $\mathrm{p} = (\mathrm{q} \vee \mathrm{r})$ oder $\mathrm{p} = (\mathrm{q} \wedge \mathrm{r})$, so sind (neben p selbst) sämtliche Teilformeln von q und von r auch Teilformeln von p.
Nichts Anderes sind Teilformeln von p.

Betrachtet man den Rekursionsbaum zu p, so entsprechen Teilformeln in naheliegender Weise “Teilbäume”.

Zur Rechtfertigung von Umformungen

Satz: [Ersetzbarkeitstheorem] Es seien $\mathrm{p}, \mathrm{q}, \mathrm{r}$ Formeln.
q sei eine Teilformel von p und r sei äquivalent zu q.
Ersetzt man ein Vorkommen von q in p durch r, so erhält man eine Formel p'.
Es gilt: $\mathrm{p} \equiv \mathrm{p}'$.

Beweis: Der Beweis der behaupteten Aussage $\mathrm{E}_{\mathrm{q},\mathrm{r}}(\mathrm{p})$ erfolgt durch (strukturelle) Induktion über den rekursiven Formelaufbau von p.

Induktionsanker: Ist p atomar, so gilt $\mathrm{q} = \mathrm{p}$ (da q Teilformel von p) und die Ersetzung liefert $\mathrm{p}' = \mathrm{r}$. Nach Voraussetzung gilt $\mathrm{q} \equiv \mathrm{r}$ und mithin $\mathrm{p} \equiv \mathrm{p}'$.

Induktionsannahme: Es gelte $\mathrm{E}_{\mathrm{q},\mathrm{r}}(\mathrm{s})$ und $\mathrm{E}_{\mathrm{q},\mathrm{r}}(\mathrm{t})$.
Induktionsschritt: Wir betrachten die drei Fälle 1) $\mathrm{p} = \neg \mathrm{s}$, 2) $\mathrm{p} = (\mathrm{s} \vee \mathrm{t})$ und 3) $\mathrm{p} = (\mathrm{s} \wedge \mathrm{t})$.

Ist $q = p$, so gilt die Argumentation aus dem Induktionsanker.

Wir setzen im Folgenden $p \neq q$ voraus.

1) $p = \neg s$, so muss q eine Teilformel von s sein.

p' geht durch Ersetzen eines Vorkommens von q aus p hervor.

Die entsprechende Ersetzung formt s in s' um. Wegen $E_{q,r}(s)$ ist $s \equiv s'$.

Daher gilt $\neg s \equiv \neg s'$, also $p \equiv p'$.

2) $p = (s \vee t)$. p' geht durch Ersetzen eines Vorkommens von q aus p hervor.

O.E. liege dieses Vorkommen in s.

Die entsprechende Ersetzung formt s in s' um. Wegen $E_{q,r}(s)$ ist $s \equiv s'$.

Daher gilt $p = (s \vee t) \equiv (s' \vee t) = p'$.

3) $p = (s \wedge t)$ zeigt man analog.

Noch mehr logische Äquivalenzen

1. $((p \Rightarrow q) \wedge (r \Rightarrow q)) \equiv (p \vee r) \Rightarrow q$.
2. $(p \vee (q \wedge \neg q)) \equiv p$;
   $(p \wedge (q \vee \neg q)) \equiv p$.
3. Dominanz: $p \wedge f \equiv f$; $p \vee w \equiv w$.
4. Negationsgesetze: $p \wedge \neg p \equiv f$; $p \vee \neg p \equiv w$.
5. Doppelte Negation: $\neg\neg p \equiv p$.
6. Idempotenzgesetze: $p \wedge p \equiv p$; $p \vee p \equiv p$.
7. Absorptionsgesetze: $(p \wedge (p \vee q)) \equiv p$; $(p \vee (p \wedge q)) \equiv p$.

Hierbei haben wir das Falsum f und das Verum w in die Formeln eingebaut. Zur formalen Rechtfertigung könnte man $f = (p \wedge \neg p)$ und $w = (p \vee \neg p)$ setzen.

Abtrennungsregel *modus ponens* (erweitet zu Schlussketten)

Ist p wahr sowie $p \Rightarrow q$, so auch q.

In Zeichen: $(p \wedge (p \Rightarrow q)) \Rightarrow q$. (z.B.: Wahrheitstafelbeweis)

Sind allgemein $p_1, \ldots, p_n$ Aussagen und
sind p_1 und $p_i \Rightarrow p_{i+1}$ wahr für $i = 1, \ldots, n-1$,
so ist p_n wahr.

Hierfür schreibt man abkürzend (wenn auch nicht ganz korrekt):
$(p_1 \wedge (p_1 \Rightarrow p_2 \Rightarrow \ldots \Rightarrow p_n)) \Rightarrow p_n$

Ähnlich notiert man manchmal Ketten von Äquivalenzen (s.o.(!))

Logeleien, formalisiert mit Logik

Man betrachte folgendes Spiel 4-Doku:

Auf einem $4x4$ Spielfeld sind Zahlen $1, 2, 3, 4$ in die 16 Kästen so einzutragen, dass in jeder Zeile, Spalte und auch in jedem "Quadranten" je vier verschiedene Zahlen stehen. Ein mögliches 4-Doku ist also:

1	2	3	4
4	3	2	1
2	1	4	3
3	4	1	2

Nun seien einige Elemente bereits vorgegeben, und es ist ein 4-Doku zu ergänzen.
Betrachte

u	2	s	4
4	w	2	x
v	1	t	3
3	y	1	z

Da u, s in der ersten Zeile liegen, gilt: $\{u, s\} = \{1, 3\}$.
Da w, x in der zweiten Zeile liegen, gilt: $\{w, x\} = \{1, 3\}$.
Da v, t in der dritten Zeile liegen, gilt: $\{v, t\} = \{2, 4\}$.
Da y, z in der vierten Zeile liegen, gilt: $\{y, z\} = \{2, 4\}$.
Spaltenbedingungen: $\{u, v\} = \{1, 2\}$, $\{w, y\} = \{3, 4\}$, $\{s, t\} = \{3, 4\}$, $\{x, z\} = \{1, 2\}$.
Kästchenbedingungen: $\{u, w\} = \{1, 3\}$, $\{s, x\} = \{1, 3\}$, $\{v, y\} = \{2, 4\}$, $\{t, z\} = \{2, 4\}$.
Da wir mit der Mengenargumentation eigentlich noch nicht vertraut sind, erfolgt eine Übersetzung in die Aussagenlogik.

u	2	s	4
4	w	2	x
v	1	t	3
3	y	1	z

Beispielsweise sei $[u, 3]$ die atomare Aussage "$u = 3$."
Die Bedingung $\{u, s\} = \{1, 3\}$ liest sich nun so: $([u, 1] \wedge [s, 3]) \vee ([u, 3] \wedge [s, 1])$.
Die Bedingung $\{u, v\} = \{1, 2\}$ ergibt: $([u, 1] \wedge [v, 2]) \vee ([u, 2] \wedge [v, 1])$.
Außerdem dürfen wir u (z.B.) keine unterschiedlichen Zahlenwerte zuweisen, d.h.:
$([u, 1] \Rightarrow (\neg[u, 2] \wedge \neg[u, 3] \wedge \neg[u, 4])) \wedge ([u, 2] \Rightarrow \dots)$.
Alle diese Bedingungen werden mit einer großen Konjunktion verknüpft.
Daher bietet sich für die eigentlichen 4-Doku-Bedingungen an, das Distributivgesetz anzuwenden:
$(([u, 1] \wedge [s, 3]) \vee ([u, 3] \wedge [s, 1])) \equiv ((([u, 1] \wedge [s, 3]) \vee [u, 3]) \wedge (([u, 1] \wedge [s, 3]) \vee [s, 1]))$
$\equiv (([u, 1] \vee [u, 3]) \wedge \dots)$.
Ähnlich folgt: $(([u, 1] \wedge [v, 2]) \vee ([u, 2] \wedge [v, 1])) \equiv (([u, 1] \vee [u, 2]) \wedge \dots)$.
Da sich die möglichen u-Belegungen wechselseitig ausschließen, folgt aus $([u, 1] \vee [u, 3]) \wedge ([u, 1] \vee [u, 2])$, dass $u = 1$ gelten muss.
So ließe sich nach und nach dieses 4-Doku-Puzzle lösen.

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., $y = |x|$. }

Abschließende Hinweise

— “Rekursion” zentrales Konzept in der Informatik.
— Daher ist “Induktion” das zentrale Beweisprinzip in der Informatik.

— Als Programmiersprachenparadigma gestattet / erzwingt es “seiteneffektfreies Programmieren”.

— Trennung zwischen Syntax und Semantik.

— Erfüllbarkeitsproblem ist “schwierig” (mehr im nächsten Semester).
Beispiel 4-Doku hat potentiell bereits $64 = 4^3$ Aussagevariablen, das normale Sudoku sogar 9^3.