Theoretische Informatik
Mengentheoretisch-algebraische Grundlagen
Gerhard Brewka

1. Motivation: Teilgebiete der Informatik und Theoretische Informatik

Die Informatik wird meist in folgende 4 Gebiete eingeteilt: Technische, Praktische, Angewandte und Theoretische Informatik. Die Theoretische Informatik untersucht grundlegende Konzepte, die für das gesamte Gebiet von Bedeutung sind. Die Vorlesung Mengentheoretisch-algebraische Grundlagen ist die erste in einem Zyklus von 4 Vorlesungen des Grundstudiums. Wir wollen zunächst einen kurzen Überblick über relevante Themen dieser Vorlesungen geben:

1. Mengentheoretisch-algebraische Grundlagen:

   In dieser Vorlesung wird das begriffliche mathematische/informatische Rüstzeug vermittelt, das für jede Beschäftigung mit der Informatik auf einem wissenschaftlichen Niveau unerlässlich ist.

2. Logik:

   Logik untersucht Folgerungsbeziehungen. Sie entwickelt dazu präzise Sprachen, in denen exakte Spezifikationen vorgenommen werden können. Von besonderem Interesse sind auch automatisierbare Schlussverfahren.

3. Formale Sprachen:

   Formale Sprachen sind Kunstsprachen, wie sie z.B. zur Kommunikation mit Rechner entwickelt werden (Programmiersprachen). Untersucht werden verschiedene Klassen solcher Sprachen, ihre Eigenschaften, Beschreibungsmöglichkeiten (Grammatiken) und entsprechende Automatenmodelle, durch die diese Sprachen erkannt werden können.

4. Berechenbarkeit und Komplexität:

   Die Berechenbarkeitstheorie untersucht, wie der Begriff der Berechenbarkeit präzisiert werden kann. Verschiedene Modelle (Turing-Berechenbarkeit, While-Berechenbarkeit, ...) erweisen sich als äquivalent. Man kann zeigen, dass es nicht berechenbare, mathematisch präzise beschreibbare Funktionen gibt.

   Die Komplexitätstheorie untersucht, wie die Komplexität von Algorithmen (Rechenvorschriften) beschrieben werden kann. Dazu werden verschiedene Komplexitätsklassen und Beschreibungsmittel eingeführt.

Zur Motivation sollen noch zwei Beispiele für typische Fragestellungen der theoretischen Informatik gegeben werden.

Beispiel 1: Nicht berechenbare Funktionen über den natürlichen Zahlen

Gibt es Funktionen (also Zuordnungen von einem Ausgabe- zu Eingabewerten), die nicht berechenbar sind? Das ist tatsächlich der Fall. Hier soll kurz eine Argumentationsskizze geliefert werden. Eine der verschiedenen Präzisierungen des Berechenbarkeitsbegriffs ist die der While-Berechenbarkeit. While-Programme enthalten als zulässige Konstrukte Zuweisungen (Werte werden in Variablen gespeichert) und While-Schleifen (ein Programmstück wird so lange ausgeführt, wie eine Testbedingung erfüllt ist). Jede in irgendeinem der formalen Berechenbarkeitsmodelle berechenbare Funktion kann auch durch ein While-Programm berechnet werden.

Jedes While-Programm lässt sich eindeutig als natürliche Zahl codieren. Betrachte nun die folgende Frage (auch spezielles Halteproblem genannt): terminiert das While-Programm mit Code n bei Eingabe der Zahl n, oder geht es in eine Endlosschleife? Wir können eine Funktion f definieren, die bei Eingabe von n den Wert 1 liefert, falls das Programm mit Code n bei Eingabe n terminiert. Ansonsten soll f den Wert 0 liefern.

Wäre f berechenbar, so müsste es ein While-Programm P geben, das f berechnet. P ließe sich auf einfache Weise zu einem Programm P' erweitern, das folgendermaßen aufgebaut ist: zunächst wird P ausgeführt, dann wird getestet, ob die Ausgabe von P 1 ist. Ist das der Fall so soll P' in eine Endlosschleife gehen, ansonsten soll P' 0 liefern. Es sei m der Code von P'.

P' hält bei Eingabe m $\Leftrightarrow$ P liefert 0 bei Eingabe von m
$\Leftrightarrow$ das Programm mit Code m hält nicht bei Eingabe m
$\Leftrightarrow$ P' hält nicht bei Eingabe m

Wir haben also ais der Annahme, f sei berechenbar, einen Widerspruch abgeleitet. Damit kann f nicht berechenbar sein.

Beispiel 2: Sortieren von Listen natürlicher Zahlen (aufsteigend)

Wir betrachten zwei Sortierverfahren und ihre Komplexität, gemessen an der Zahl der im schlechtesten Fall für eine Liste der Länge n notwendigen Vergleiche:

a) Bubblesort:
gehe zu sortierende Liste von links nach rechts durch und vergleiche jeweils Nachbarn;
wenn größeres Element vor kleinerem steht, vertausche die Elemente;
wiederhole diesen Prozess, bis keine Vertauschung mehr stattfindet

Anzahl der Vergleiche im schlechtesten Fall:
Länge der Liste n, im ersten Durchlauf n-1 Vergleiche
das größte Element gelangt jeweils nach hinten, deshalb
$(n-1) + (n-2) + ... + 1 = n * (n-1) / 2 = (n^2 – n)/2$Vergleiche
da der größte Exponent das Wachstum wiedergibt, sagt man: $O(n^2)$

b) Heapsort:
verwende Binärbaum (maximal 2 Nachfolger pro Knoten), für den Heap-Eigenschaft gilt:
1. der Baum ist vollständig bis zur Ebene k-1, wobei k die Tiefe des Baumes ist
2. die Knoten in der untersten Ebene sind linksbündig angeordnet
3. das Element an jedem Knoten ist größer als die Elemente an den Nachfolgerknoten

Sortiere nun folgendermaßen:
a) Entferne das (größte) Element an der Wurzel des Baumes und füge es in Liste ein
b) Nimm am weitesten rechts stehendes Element der untersten Ebene aus dem Baum, füge es an der Wurzel ein und lass es „versickern“ bis die Heap-Eigenschaft wieder hergestellt ist: wenn es größeren Nachfolger gibt, vertausche es jeweils mit dem größten Nachfolger
c) falls der Baum vollständig abgearbeitet ist, terminiere, sonst gehe zu a)

Anzahl Vergleiche beim Versickern im schlechtesten Fall?
Tiefe des Baums (Anzahl Kanten) mit n Knoten $\leq \log_2 n$
Vergleiche für Versickern also maximal: $2 * \log_2 n$
das Ganze höchstens n mal => Anzahl Vergleiche O(n log n)

Betrachten wir eine Liste der Länge $n = 2^{10} = 1024$. n^2 ist in diesem Fall bereits größer als 1 Million, während n log n = 10240.

Frage: ist nicht die Erzeugung des Heaps selbst zu komplex?

Idee:
Füge Zahlen unsortiert in Baum ein. Sei k die Tiefe des Baums.
lasse alle Knoten auf Ebene k-1 versickern (Blätter erfüllen sowieso die Heap-Eigenschaft)
mache weiter bei Ebene k-2, k-3 etc, bis Wurzel erreicht.
Anzahl Vergleiche: O(n log n)

Insgesamt ergibt die Komplexitätsanalyse also, dass ein auf den ersten Blick umständliches Verfahren ein wesentlich besseres Verhalten zeigen kann als ein einfaches Verfahren. Ziel der Komplexitätstheorie ist es, formale Hilfsmittel und Konzepte bereit zu stellen, die Komplexitätsuntersuchungen ähnlicher Art möglich machen.

Wir beenden diese motivierende Einführung mit einer kurzen Übersicht über die (geplanten) Themen dieser Vorlesung:

1. Mengenbegriff und Mengenalgebra
2. Aussagenlogik
3. Relationen
4. Korrespondenzen, Funktionen, Unendlichkeit
5. Algebraische Strukturen
6. Graphen und Verbände
7. Ordinal- und Kardinalzahlen
8. Induktion und Rekursion
9. Halbgruppen und Sprachen

2. Mengenbegriff

Cantor (1845-1918) gilt als Begründer der Mengenlehre. Er charakterisiert Mengen folgendermaßen:

Eine Menge ist eine Zusammenfassung bestimmter, wohlunterschiedener Objekte unserer Anschauung oder unseres Denkens zu einem Ganzen; diese Objekte heißen die Elemente der Menge

Notation: a Element von M: $a \in M$
a nicht Element von M: $a \notin M$

endliche Mengen lassen sich aufzählen: {2,8,12,13}, {Peter, Hans, Fritz}

unendliche Mengen können durch Angabe der Eigenschaften ihrer Elemente definiert werden

$$\{x \mid x \text{ ist Primzahl}\}$$

oder induktiv, z.B.:

M_1 ist die kleinste Menge für die gilt:

1) $aa \in M_1$ und $bb \in M_1$ (Basisfälle)
2) wenn $w \in M_1$, dann gilt $awa \in M_1$ und $bwb \in M_1$ (abgeleitete Fälle).

Beispiele für Elemente in M_1: abbaabba, baab, nicht in M_1: babab

Induktive Definitionen erlauben Induktionsbeweise:
eine Eigenschaft E gilt für alle Elemente einer induktiv definierten Menge, wenn
1) E für die Basisfälle gilt (hier aa und bb), und
2) E für die abgeleiteten Fälle gilt, vorausgesetzt E gilt für die Elemente, aus denen sie abgeleitet werden (hier w).

Beispiel: zeige, dass jedes Element von M_1 eine gerade Anzahl von a's und b's besitzt.

1) aa besitzt 2 a's und 0 b's, bb besitzt 0 a's und 2 b's
2) Angenommen w besitzt m a's und n b's, wobei m und n gerade.
   awa besitzt m+2 a's und n b's, beide gerade.
   bwb besitzt m a's und n+2 b's, beide gerade. QED

Extensionalitätsprinzip: Mengen sind gleich, wenn sie dieselben Elemente haben:

$$M = N \text{ <-> für alle } x \text{ gilt: } (x \in M \text{ <=> } x \in N)$$

Vorsicht: es sind nicht beliebige Mengenkonstruktionen durch Aussagen möglich:

Russellsche Antinomie:

es gibt sicherlich Mengen, die sich nicht selbst als Element enthalten.
Gibt es auch die Menge all dieser Mengen:

$$\{ M \mid M \notin M \}$$

Angenommen, es gäbe diese Menge. Enthält sie sich selbst? 2 Möglichkeiten:
a) sie enthält sich nicht selbst: dann enthält sie sich nach Definition.
b) sie enthält sich selbst: dann enthält sie sich nicht nach Definition.

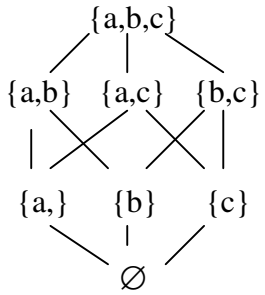
Sätze wie „x ist Primzahl“ nennt man auch Aussageformen.
Nicht jede beliebige Aussageform ist also geeignet für die Definition von Mengen verschiedene Festlegungen, die Antinomien ausschliessen, wurden entwickelt, damit die Mengenlehre als Grundlage der Mathematik gelten kann (-> Hauptstudium)

hier naiver Mengenbegriff ausreichend.

Grundbegriffe der Mengenlehre:

Teilmenge: $N \subseteq M$ gdw. für alle x: $x \in N$ impliziert $x \in M$.
echte Teilm.: $N \subset M$ gdw. $N \subseteq M$ und nicht $M \subseteq N$.
leere Menge: $\emptyset$
Potenzmenge einer Menge M: $P(M) = \{ X \mid X \subseteq M \}$

Beispiel: P({a,b,c})



Für alle Mengen M, N, P gilt:

$M \subseteq M$ (Reflexivität)
$N \subseteq M$ und $M \subseteq P$ impliziert $N \subseteq P$ (Transitivität)
$N \subseteq M$ und $M \subseteq N$ impliziert $M = N$ (Antisymmetrie)

Mengenalgebra:

$M \cap N = \{x \mid x \in M \text{ und } x \in N\}$	Schnitt von M und N
$M \cup N = \{x \mid x \in M \text{ oder } x \in N\}$	Vereinigung von M und N
$M \setminus N = \{x \mid x \in M \text{ und } x \notin N\}$	Differenz von M und N

falls $M \cap N = \emptyset$ nennt man M und N disjunkt.

Eigenschaften der Operationen:

Kommutativität: $M \cap N = N \cap M$
$M \cup N = N \cup M$

Assoziativität: $M \cap (N \cap P) = (M \cap N) \cap P$
$M \cup (N \cup P) = (M \cup N) \cup P$

Distributivität: $M \cap (N \cup P) = (M \cap N) \cup (M \cap P)$
$M \cup (N \cap P) = (M \cup N) \cap (M \cup P)$

Idempotenz: $M \cap M = M$
$M \cup M = M$

Veranschaulichung Venn-Diagramme (Folie Gerber)

Ist M Teilmenge einer (Grund)-Menge G, so heisst $C_G(M) = G \setminus M$ Komplement von M bezüglich G.
Häufig ist aus Kontext klar, welches G gemeint ist, und man spricht einfach vom Komplement von M und notiert es C(M)

de Morgansche Gesetze:

$C(M \cap N) = C(M) \cup C(N)$
$C(M \cup N) = C(M) \cap C(N)$

Ein Mengensystem ist eine Menge von Mengen (z.B. Potenzmenge)

Vereinigung eines Mengensystems M: $\cup M = \{ x \mid \text{es gibt } X \in M \text{ mit } x \in X\}$
Schnitt eines Mengensystems M: $\cap M = \{ x \mid \text{für alle } X \in M \text{ gilt } x \in X\}$

Mengensysteme oft durch Indexmengen charakterisiert:
gegeben Menge von Indizes I, $M = \{M_i \mid i \in I\}$, dann schreiben wir statt $\cup M$: $\cup_{i \in I} M_i$
(analog für $\cap$)

Verallgemeinerte Distributivität:

$N \cap \cup_{i \in I} M_i = \cup_{i \in I} (M_i \cap N)$
$N \cup \cap_{i \in I} M_i = \cap_{i \in I} (M_i \cup N)$

Verallgemeinerte de Morgansche Gesetze:

$C(\cup_{i \in I} M_i) = \cap_{i \in I} C(M_i)$
$C(\cap_{i \in I} M_i) = \cup_{i \in I} C(M_i)$

Seien M und N Mengen, das kartesische Produkt (auch: Kreuzprodukt) von M und N ist die Menge

$$M \times N = \{(a,b) \mid a \in M, b \in N\}$$

(a,b) bezeichnet man als geordnetes Paar

Seien $M_1, ..., M_n$ Mengen

$$M_1 \times ... \times M_n = \{(x_1, ..., x_n) \mid x_i \in M_i\}$$

ist die Menge der n-Tupel über Mengen $M_1, ..., M_n$.

3-Tupel heißen auch Tripel, 4-Tupel Quadrupel, 5-Tupel Quintupel
falls $M_1 = ... = M_n = M$, so heißt $M_1 \times ... \times M_n$ auch M^n.

3. Aussagenlogik

3.1 Einführung

Was ist Logik? Wissenschaftsgebiet, das Folgerungsbeziehungen untersucht
math. Logik? mit math. Mitteln, Hauptziel: Präzisierung des Folgerungsbegriffs

Folgern: einige Beispiele

I.
Wenn es regnet, ist es nass.
Es regnet _
Es ist nass.

II.
Es ist Sonntag oder es ist Montag.
Es ist nicht Sonntag. _
Es ist Montag.

III.
Wenn warm ist, ist es nicht kalt.
?
Wenn es kalt ist, ist es nicht warm

IV.
Wenn es Sommer ist, ist es warm.
Es ist warm. ?
Es ist Sommer.

Die Fälle I und II sind klar, III und IV weniger
(wir werden sehen, dass III gültig ist, IV nicht).
Grundfrage: Wann kann man Sätze folgern, wann nicht?
Beobachtung: Folgerungen hängen nicht von Inhalt der Sätze ab, nur von deren Form:

Wenn die Sonne scheint, ist es warm.
die Sonne scheint _
Es ist warm.

Wir können also vom Inhalt abstrahieren (und einfach A, B, C ... schreiben -> Variablen)
relevant ist nur, dass Aussagen wahr oder falsch (dargestellt als w/f, t/f oder1/0) sein können.

was interessiert uns an der Form? abhängig von der Logik
Aussagenlogik: nur einfache log. Verknüpfungen
Prädikatenlogik: auch Quantoren (für alle, es gibt), Prädikate, Funktionen.

3.2 Syntax und Semantik der Aussagenlogik

Syntax: Aufbau der zugrundeliegenden formalen Sprache
Semantik: Bedeutung der Ausdrücke der formalen Sprache

Relevante Verknüpfungen (Junktoren):

	und	oder	wenn ...dann	nicht	genau dann ...wenn
Symbole:	$\wedge$	$\vee$	->	$\neg$	<->

Aussagen werden durch Formeln repräsentiert.

Def.: Sei V eine Menge von aussagenlogischen Variablen. Die Menge der (aussagenlogischen) Formeln über V ist die kleinste Menge, für die gilt:
1. jedes Element von V ist eine Formel,

2. wenn P und Q Formeln sind, so auch ¬P, (P∧Q), (P∨Q), (P->Q), (P <->Q).

Die Elemente von V nennt man auch atomare Formeln.
Klammern kann man weglassen, wenn Formeln aufgrund von Bindungsregeln eindeutig sind:
Bindungsstärke (abnehmend): ¬, ∧, ∨, ->, <->

Wann sind Formeln wahr oder falsch? Kann man nur beantworten, wenn man
a) Bedeutung der Junktoren und
b) die Wahrheitswerte der Variablen kennt

a) Bedeutung der Junktoren entspricht intuitiver Bedeutung in Umgangssprache. Beschrieben durch Wahrheitstafeln:

P	Q	¬P	P ∧ Q	P ∨ Q	P -> Q	P <-> Q
1	1	0	1	1	1	1
1	0	0	0	1	0	0
0	1	1	0	1	1	0
0	0	1	0	0	1	1

b) man braucht eine Interpretation, d.h. eine Abbildung I: V -> {0,1), die jeder Variablen einen Wahrheitswert zuordnet, um Formel auszuwerten.

Beispiel: F = (A ∨ B) -> (C ∧ D)
I(A) = I(C) = 1, I(B) = I(D) = 0,
I wertet F zu 0 aus.

Anmerkung: jede Interpretation läßt sich als ein möglicher Zustand des Weltausschnitts auffassen, der durch V beschrieben werden kann.

Interpretationen, die eine Formel F (bzw. eine Menge von Formeln M) zu 1 auswerten, heißen Modelle von F (bzw. M).

Folgerbarkeit: Q folgt aus F (bzw. M) heißt: immer wenn F (bzw. M) wahr ist, muß auch Q wahr sein, also: jedes Modell von F (bzw. M) ist Modell von Q.

Wir schreiben F |= Q bzw. M |= Q

Folgerbarkeit lässt sich (in einfachen Fällen) anhand von Wahrheitstabellen überprüfen

Beispiel (siehe oben III):

P	Q	¬P	¬Q	P -> ¬Q	Q -> ¬P
1	1	0	0	0	0
1	0	0	1	1	1
0	1	1	0	1	1
0	0	1	1	1	1

Zeilen entsprechen Interpretationen. Interpretationen in Zeile 2,3,4 sind Modelle von P -> ¬Q, diese sind auch Modelle von Q -> ¬P, also P -> ¬Q |= Q -> ¬P

Beispiel 2 (siehe oben IV):

P	Q	P -> Q
1	1	1
1	0	0
0	1	1
0	0	1

Modelle von P -> Q und Q entsprechen Zeilen 1 und 3, Zeile 3 macht aber P falsch, also folgt P nicht aus P -> Q und Q.

2 Formeln heißen äquivalent (Symbol: ≡), wenn jede Interpretation sie gleich auswertet.
Formeln, die in allen Interpretationen wahr sind, heißen allgemeingültig (Tautologien): A∨¬A
Formeln, die in allen Interpretationen falsch sind, widersprüchlich (Kontradiktionen): A ∧ ¬A
Formeln, die mindestens 1 Modell besitzen, heißen erfüllbar.

Satz: Folgende Aussagen sind äquivalent:
1) P |= Q.
2) P -> Q ist Tautologie.
3) P ∧ ¬Q ist Kontradiktion.

Beweis: Wir zeigen 1) => 2), 2) => 3) und 3) => 1)

1) => 2):
Es gelte P |= Q. Betrachte eine Interpretation I. Es gibt 2 Fälle:

a) I ist Modell von P. Da P |= Q ist jedes Modell von P auch Modell von Q. Damit wird auch P -> Q in I zu 1 ausgewertet.
b) I ist nicht Modell von Q. Dann ist der Wahrheitswert von P in I 0 und damit der Wahrheitswert von P -> Q in I ist 1.

Da P -> Q in allen Interpretation zu 1 ausgewertet wird, ist P -> Q Tautologie.

2) => 3):
Wenn P -> Q Tautologie ist, so wird in jeder Interpretation P zu 0 ausgewertet oder Q zu 1. Damit gibt es keine Interpretation, die P zu 1 auswertet und Q zu 0. Also gibt es keine Interpretation, die P ∧ ¬Q zu 1 auswertet, und P ∧ ¬Q ist Kontradiktion.

3) => 1):
Sei P ∧ ¬Q Kontradiktion. Dann gibt es keine Interpretation, in der P zu 1 und Q zu 0 ausgewertet wird. Also wertet jedes Modell von P auch Q zu 1 aus. Damit gilt P |= Q.
QED.

Häufig verwendete Äquivalenzen:

¬¬P ≡ P
P -> Q ≡ ¬P ∨ Q
P <-> Q ≡ (P∧Q) ∨ (¬P∧¬Q)

$\neg(P \vee Q)$	$\equiv (\neg P \wedge \neg Q)$	de Morgansche Regeln
$\neg(P \wedge Q)$	$\equiv (\neg P \vee \neg Q)$	
$(P \wedge Q) \vee R$	$\equiv (P \vee R) \wedge (Q \vee R)$	Distributivität
$(P \vee Q) \wedge R$	$\equiv (P \wedge R) \vee (Q \wedge R)$	

(Teil)-Formeln können durch äquivalente Formeln ersetzt werden!
Die entstehende Formel ist äquivalent zur ursprünglichen.

3.3 Beispiel Geldautomat

Zur Illustration wollen wir Aussagenlogik benutzen, um das Verhalten eines Geldautomaten zu beschreiben. Wir wollen folgende Aussagen repräsentieren:

Intuitive Bedeutung:	verwendete Variable:
eingegebene Karte ist gültig	K
eingegebene PIN ok	P
Kontostand ok	S
Geldbetrag auszahlen	A
Rückgabe Karte	R

1) Die Karte wird einbehalten, genau dann wenn die eingegebene PIN falsch ist:
P <-> R

Es gibt 32 Interpretationen, 16 davon mit P=R (also Modelle von 1)

	K	P	S	A	R	
1.	0	0	0	0	0	+
2.	0	0	0	1	0	-
3.	0	0	1	0	0	+
4.	0	0	1	1	0	-
5.	0	1	0	0	1	+
6.	0	1	0	1	1	-
7.	0	1	1	0	1	+
8.	0	1	1	1	1	-
9.	1	0	0	0	0	+
10.	1	0	0	1	0	-
11.	1	0	1	0	0	+
12.	1	0	1	1	0	-
13,	1	1	0	0	1	+
14.	1	1	0	1	1	-
15.	1	1	1	0	1	-
16.	1	1	1	1	1	+

2) Wir legen fest, dass der gewünschte Betrag ausgezahlt wird, wenn die Karte gültig, die PIN korrekt, und der Kontostand ausreichend ist:
$K \wedge P \wedge S \rightarrow A$

Interpretation 15 kein Modell mehr

3) Es soll auch gelten, dass nur unter den obigen Bedingungen ausgezahlt wird:
$A \rightarrow K \wedge P \wedge S$

Jetzt sind nur noch die 8 mit + gekennzeichneten Interpretationen Modelle.

was können wir aus 1), 2), 3) und ¬K folgern? Z. B. ¬A
was können wir aus 1), 2), 3) und A folgern? Z. B. K, P, S, R
was können wir aus 1), 2), 3) und ¬A folgern? Z. B. ¬K v ¬P v ¬S

2.4 Inferenz

Eine Inferenzregel dient der Ableitung von Formeln aus bereits gegebenen Formeln.

Allgemeine Form:

$$\frac{F_1, ..., F_n}{F}$$

$F_1, ..., F_n$ nennt man Prämissen, F Konklusion der Inferenzregel.

Die Regel ist korrekt, falls $F_1, ..., F_n \models F$.

Beispiele korrekter Inferenzregeln. Seien P,Q,R beliebige Formeln, T beliebige Tautologie:

(1) Modus ponens: $\frac{P, P \rightarrow Q}{Q}$

(2) Modus tollens: $\frac{P \rightarrow Q, \neg Q}{\neg P}$

(3) Negations-Eliminierung: $\frac{\neg\neg P}{P}$

(4) Negations-Einführung: $\frac{P}{\neg\neg P}$

(5) Kontraposition: $\frac{P \rightarrow Q}{\neg Q \rightarrow \neg P}$

(6) Resolution: $\frac{P \vee Q, \neg P \vee R}{Q \vee R}$

(7) T-Einführung: $\frac{}{T}$

Sei M eine Menge von Formeln, R eine Menge von Inferenzregeln.
$I \in R$ ist anwendbar in M, wenn M die Prämissen von I enthält.
Ein formaler R-Beweis für F_n aus M ist eine Folge von Formeln $(F_1, F_2, ..., F_n)$, so dass für alle $i \in \{1,...,n\}$ gilt:

1) $F_i \in M$ (Prämisse), oder
2) es gibt $I \in R$ mit Konklusion F_i, I anwendbar in $\{F_1,...,F_{i-1}\}$.

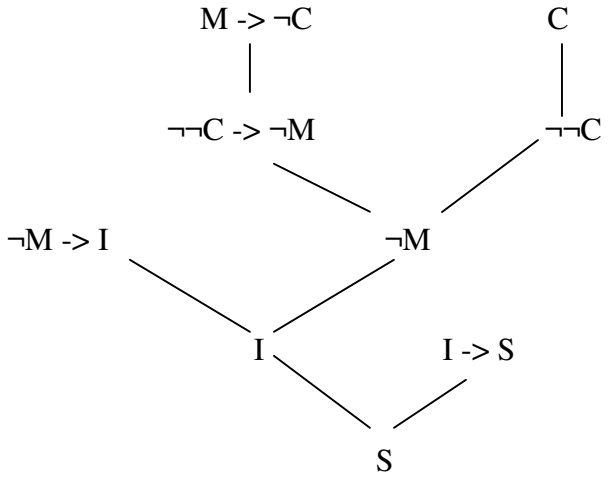
R heißt vollständig, falls für alle M, F gilt: wenn $M \models F$ dann gibt es einen formalen R-Beweis für F aus M. (z.B. ist Modus ponens mit passenden T-Einführungsregeln vollständig).

Beispiel: Information über Peter P.:
M = {**M**athematiker -> ¬Mag_**C**omputer, ¬**M**athematiker -> **I**nformatiker, **I**nformatiker -> **S**chlau, Mag_**C**omputer}

Ein formaler Beweis für **S**chlau (Verwendung von Abkürzungen, Regelnr. vor abgel. Formel).

(M -> ¬C, (5) ¬¬C -> ¬M, C, (4) ¬¬C, (1) ¬M, ¬M -> I, (1) I, I -> S, (1) S)

Anmerkung: Beweise können auch (anschaulicher) als Bäume repräsentiert werden.



Beweisen mit Resolution

Widerspruchsbeweise zeigen Folgerungsbeziehungen $M \models F$ durch Nachweis der Inkonsistenz von $M \cup \{\neg F\}$.

Beobachtung: Resolutionsregel lässt sich verallgemeinern zu

A1 v...v Aj v...v Ak, B1 v...v Bi v...v Bm
A1 v...v [Aj v]...v Ak v B1 v...v [Bi v]...v Bm, falls Aj = ¬Bi (geklammerte Teile entfallen)

Terminologie:
Ein Literal ist eine atomare Formel oder eine negierte atomare Formel.
Eine Formel ist in konjunktiver Normalform (KNF), wenn sie eine Konjunktion von Disjunktionen von Literalen ist.

Disjunktionen von Literalen heißen auch Klauseln.
Statt Konjunktionen von Klauseln verwendet man äquivalent Klauselmengen.
Die aus keinem Disjunktionsglied bestehende leere Klausel [] ist äquivalent zu 0.

Satz: Jede Formel kann in eine äquivalente Formel in KNF überführt werden.

Zur Überführung in KNF:	verwendete Äquivalenzen:
1. löse -> und <-> auf	P -> Q ≡ ¬P v Q
	P <-> Q ≡ (P∧Q) v (¬P∧¬Q)
2. bringe Negationszeichen vor Variablen	de Morgansche Regeln:
	¬(P v Q) ≡ (¬P ∧ ¬Q)
	¬(P ∧ Q) ≡ (¬P v ¬Q)
3. multipliziere aus	z.B. (a ∧ b) v c ==> (a v c) ∧ (b v c)

Resolutionsverfahren:

1. negiere zu überprüfende Formel F und füge sie zu Prämissen P hinzu,
2. überführe P ∪ {¬F} in KNF,
3. wende die Resolutionsregel an und versuche, die leere Klausel [] abzuleiten.

Satz: F ist folgerbar aus M gdw. auf diese Weise die leere Klausel hergeleitet werden kann.

Bsp:

A v B A -> C ¬C -> ¬B

ableitbar C?

A v B	A v B	(1)		
A -> C	¬A v C	(2)		
¬C -> ¬B	C v ¬B	(3)		
¬C	¬C	(4)		
	¬B	(5)	Resolvente aus 3,4	
	¬A	(6)	Resolvente aus 2,4	
	A	(7)	Resolvente aus 1,5	
	[]	(8)	Resolvente aus 6,7	C also ableitbar.