Vorlesung Logik, SS 2003

Gerhard Brewka

Gebraucht die Zeit, sie geht so schnell von hinnen,
Doch Ordnung lehrt Euch Zeit gewinnen.
Mein teurer Freund, ich rat' Euch drum
Zuerst Collegium logicum.
Da wird der Geist Euch wohl dressiert,
In spanische Stiefeln eingeschnürt,
Dass er bedächtiger so fortan
Hinschleiche die Gedankenbahn,
Und nicht etwa, die Kreuz und Quer',
Irrlichteliere hin und her

Mephisto, Faust

0. Einführung

Was ist Logik? Lehre von den formalen Beziehungen zwischen Sätzen (insbesondere Folgerungsbeziehungen)

als eigenständige Wissenschaft begründet von Aristoteles. Er unterscheidet:

Lehre vom Begriff (Klassifizierung von Begriffen, Definitionslehre)
Lehre vom Urteil (Struktur, Klassifikation von Aussagesätzen)
Lehre vom Schluss (Folgerungsbeziehungen aufgrund der Struktur der Sätze)

Beiträge von Stoikern (Theorien der Implikation, logische Antinomien)
mittelalterl. Scholastik (Arbeiten zu Sprachphilosophie und log. Semantik)
Leibniz Vorläufer der modernen Logik: log. Symbolsprache für alle Wissenschaften
Entwicklung der modernen mathematischen Logik ab Mitte 19. Jh. (Frege, Boole, Russell)

Charakteristika:
formale Kunstsprache, modelltheoretische Semantik, korrekte und vollständige Kalküle

Bemerkung: es gibt viele verschiedene math. Logiken (mehrwertige, modale, deontische, dynamische, lineare, nichtmonotone, fuzzy, temporale L. etc.)

wenn wir von "der" Logik sprechen, ist die klassische Aussagen- oder Prädikatenlogik gemeint.

Warum ist Logik relevant für Informatiker?

Beschreibungsmittel für formale Sachverhalte
Semantik von Programmiersprachen
Logisches Programmieren (PROLOG)
Programmverifikation
Automatisches Beweisen
Wissensrepräsentation
Schaltalgebra
usw.

Logik untersucht Folgerungen aufgrund der Struktur der Sätze:

Wenn es regnet, dann ist die Straße nass.
Es regnet.
Die Straße ist nass

Schluss unabhängig von Bedeutung von "es regnet", "die Straße ist nass":

Wenn es grubelt, dann brabelt es.
Es grubelt.
Es brabelt

Folgender Schluss ist nicht-logisch, da er Wissen über die Bedeutung der Sätze voraussetzt:

Peter ist mein Bruder
Peter ist mit mir verwandt

Ein logischer Schluss liegt erst vor, wenn das zusätzliche Wissen explizit gemacht wird:

Wenn Peter mein Bruder ist, ist er mit mir verwandt.
Peter ist mein Bruder.
Peter ist mit mir verwandt.

Logik abstrahiert von der Bedeutung der in Schlüssen verwendeten Sätze.
Relevant ist nur (im Falle der klassischen Logik)

1) der Aufbau komplexer Sätze (wenn ... dann, und, oder, ...)
2) dass Sätze entweder wahr oder falsch sind

Da die natürlicher Sprache vage, mehrdeutig und kontextabhängig ist, verwendet die Logik eine Kunstsprache:

Eine festgelegte Menge von Symbolen (etwa A, B, C, A1, A2, ...) repräsentiert elementare Aussagen, andere Symbole ($\wedge$, v, ->,$\neg$) Verknüpfungen von solchen Aussagen.

math. Exaktheit, auf Kosten von Ausdrucksfähigkeit, Flexibilität, Nuancenreichtum

Bsp.: A $\wedge$ B in der Aussagenlogik äquivalent zu B $\wedge$ A, aber:

Peter bekam Fieber und der Arzt verschrieb ein Medikament.
Der Arzt verschrieb ein Medikament und Peter bekam Fieber.

Grundlage dieser Vorlesung:

Uwe Schöning, Logik für Informatiker, 4. Auflage, Spektrum Akademischer Verlag, 1995, DM 29.90

Grobgliederung:

1. Aussagenlogik
   - 1.1 Syntax
   - 1.2 Semantik
   - 1.3 Äquivalenz und Normalformen
   - 1.4 Hornformeln
   - 1.5 Endlichkeitssatz
   - 1.6 Resolution
   - 1.7 Das Davis-Putnam- Verfahren
   - 1.8 Tableauverfahren
2. Prädikatenlogik
3. Logik-Programmierung

Zur Einstimmung:

Wenn ich kein Bier zum Essen trinke, dann esse ich immer Fisch.

Wenn ich Eis esse oder kein Bier trinke, dann esse ich nie Fisch.

Vereinfachung? Immer Bier und entweder kein Fisch oder kein Eis.

(kann man anhand des Durchspielens aller Möglichkeiten sehen).

1. Aussagenlogik

1.1 Syntax

Aussagenlogik untersucht Verknüpfungen wie "und", "oder", "nicht", "wenn ... dann" zwischen atomaren und komplexen Sätzen.

Sätze selbst sind entweder wahr oder falsch. Ansonsten ist ihre Bedeutung irrelevant.

Atomare Sätze werden durch beliebige Symbole repräsentiert: bei Schöning A_1, A_2, ...;
oft sind für den Benutzer Symbole bequem, die die intendierte Bedeutung wiedergeben (Regen, Nass, Verwandt), werden bei Schöning als Abkürzung für ein A_i eingeführt.

Nochmal: intendierte Bedeutung für die Logik irrelevant!!

Def. (Syntax der Aussagenlogik)
Sei A eine Menge von Symbolen (genannt aussagenlogische Variablen oder atomare Formeln).
Die Menge der (aussagenlogischen) Formeln über A ist die kleinste Menge, für die gilt:

1. Jede atomare Formel aus A ist eine Formel.
2. Wenn F und G Formeln sind, so sind $(F \wedge G)$ und $(F \vee G)$ Formeln.
3. Wenn F eine Formel ist, so ist $\neg F$ eine Formel.

$\neg F$ heißt Negation von F, $(F \wedge G)$ Konjunktion von F und G,
$(F \vee G)$ Disjunktion von F und G.
Eine Formel F, die als Teil einer Formel G vorkommt, heißt Teilformel von G.

Abkürzungen:

(F1 -> F2)	statt	$(\neg F1 \vee F2)$
(F1 <-> F2)	statt	$((F1 \wedge F2) \vee (\neg F1 \wedge \neg F2))$

Bemerkung: man könnte diese Symbole auch direkt in der Definition der Syntax der Aussagenlogik einführen. Die Einführung als Abkürzung vereinfacht aber Induktionsbeweise über den Aufbau von Formeln.
Zu beachten: Implikation gibt umgangssprachliche Verwendung von "wenn...dann" nur näherungsweise wieder.
1) Wenn Leipzig in Sachsen liegt, dann dauert die Logikvorlesung 90 Minuten.
2) Wenn Leipzig Hauptstadt von Bayern ist, dann ist Goethe mit Madonna verheiratet.

Beide Implikationen logisch gesehen wahr, weil dann-Teil von 1 wahr und wenn-Teil von 2 falsch. Beide Sätze würde man so nie äußern (Gricesche Konversationsmaximen, wenn ... dann drückt in der natürlichen Sprache häufig kausalen Zusammenhang aus, wird von Logik nicht erfasst)

1.2 Semantik der Aussagenlogik

Ziel: jede komplexe Formel in Abhängigkeit von den Wahrheitswerten der vorkommenden atomaren Formeln zu wahr oder falsch auswerten.

Als Wahrheitswerte verwenden wir 0 (falsch) und 1 (wahr).

Def.: (Belegung)
Eine (Wahrheits-) Belegung ist eine Funktion I: A -> {0,1}, wobei A die Menge der atomaren Formeln ist.

Bem.: Belegungen werden oft auch Interpretationen genannt.

Def.: (Wahrheitswert komplexer Formeln)
Sei I eine Belegung. Wir erweitern I zu einer Funktion I^: E -> {0,1}, wobei E die Menge aller Formeln ist, die aus atomaren Formeln in A aufgebaut sind:

1. Für jede atomare Formel Ai $\in$ A ist I^(Ai) = I(Ai)

2. I^((F $\wedge$ G)) = 1 falls I^(F) = 1 und I^(G) = 1
0 sonst

3. I^((F v G)) = 1 falls I^(F) = 1 oder I^(G) = 1
0 sonst

4. I^(¬F) = 1 falls I^(F) = 0
0 sonst

Da I^ Erweiterung von I ist, d.h. für atomare Formeln mit I übereinstimmt, schreiben wir vereinfachend I statt I^.

Beispiel:
Sei I eine Belegung, die A zu 1, B zu 0 und C zu 1 auswertet.
Wir bestimmen den Wahrheitswert der Formel ((A v B) $\wedge$ (¬C v ¬B)).

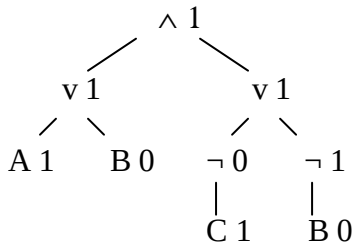
I(((A v B) $\wedge$ (¬C v ¬B))) = 1 gdw
I((A v B)) = 1 und I((¬C v ¬B)) = 1 gdw
[I(A) = 1 oder I(B) = 1] und [I(¬C) = 1 oder I(¬B) = 1] gdw
[I(A) = 1 oder I(B) = 1] und [I(C) = 0 oder I(B) = 0]

Da I(A) = 1 und I(B) = 0 ist diese Bedingung erfüllt, d.h. die Formel wird zu wahr ausgewertet.

Weniger umständlich: wir ersetzen die atomaren Formeln durch ihre Wahrheitswerte und propagieren:

((1 v 0) ∧ (¬1 v ¬0))
((1 v 0) ∧ (0 v 1))
(1 ∧ 1)
1

entspricht folgender Baumdarstellung:



Wirkung der Junktoren lässt sich durch Wahrheitstafeln darstellen:

F	¬F
0	1
1	0

F	G	F ∧ G
0	0	0
0	1	0
1	0	0
1	1	1

F	G	F v G
0	0	0
0	1	1
1	0	1
1	1	1

F	G	F -> G
0	0	1
0	1	1
1	0	0
1	1	1

F	G	F <-> G
0	0	1
0	1	0
1	0	0
1	1	1

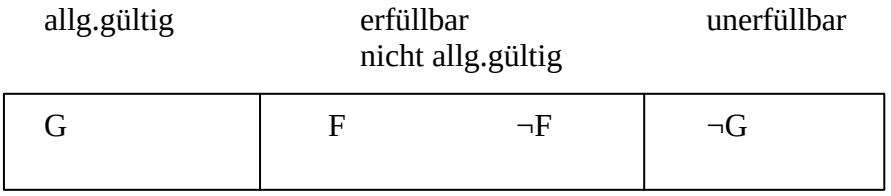
Def. (Modell, Gültigkeit, Erfüllbarkeit)
Sei F eine Formel. Eine Belegung, die F zu 1 auswertet, heißt Modell von F.
Falls I Modell von F schreiben wir: I |= F, falls nicht: I |=/= F.

Eine Formel F heißt erfüllbar, wenn F mindestens ein Modell besitzt.
Unerfüllbare Formeln heißen auch widersprüchlich (Kontradiktionen).

F heißt allgemeingültig (Tautologie), falls jede Belegung Modell für F ist.
Notation: |= F für F ist Tautologie, |=/= F für F ist nicht Tautologie.

Eine Belegung ist Modell einer Menge von Formeln M, wenn sie jedes Element von M zu 1 auswertet. M ist erfüllbar, wenn M mindestens ein Modell besitzt.

Satz: F ist Tautologie gdw. ¬F unerfüllbar.
Bew.: F Tautologie gdw jede Belegung wertet F zu 1 aus gdw keine Belegung wertet ¬F zu 1 aus gdw ¬F hat kein Modell.



Def.: (logische Folgerung)
Seien F1, ..., Fk, G Formeln. G folgt logisch aus F1, ..., Fk gdw jede zu F1, ..., Fk, und G passende Belegung, die Modell von {F1, ..., Fk} ist, auch Modell von G ist.

Satz: Folgende Aussagen sind äquivalent:

1. G folgt logisch aus F1, ..., Fk .
2. ((...(F1 ∧ F2) ∧... ∧ Fk) -> G) ist Tautologie.
3. ((...(F1 ∧ F2) ∧... ∧ Fk) ∧ ¬G) ist unerfüllbar.

Bew.: Übung

Beispiel: B folgt logisch aus B v ¬C, C

Wahrheitswert einer Formel F hängt nur von in F vorkommenden atomaren Formeln ab.
Wenn F n atomare Formeln enthält, so gibt es 2^n zu testende Belegungen. Diese können in Wahrheitstafeln systematisch beschrieben werden.

A	B	(¬A v B)	(¬B v A)	(¬A v B) ∧ (¬B v A)
0	0	1	1	1
0	1	1	0	0
1	0	0	1	0
1	1	1	1	1

Rechte Spalte: Wahrheitsverlauf der jeweiligen Formel
Wahrheitstafelmethode im Allgemeinen sehr aufwendig: exponentiell viele Zeilen.

Wieviele Formeln mit n atomaren Formeln und verschiedenen Wahrheitsverläufen gibt es?
Tafel hat 2^n Zeilen, an jeder Stelle im Wahrheitsverlauf 1 oder 0, also: 2^{2^n}

1.3 Äquivalenz und Normalformen

Def.: (Äquivalenz)
Zwei Formeln F und G heißen äquivalent (Notation: $F \equiv G$), falls für alle Belegungen I gilt: $I(F) = I(G)$.

Satz: Seien F und G Formeln. $F \equiv G$ gdw. F <-> G ist Tautologie

Satz: (Ersetzbarkeit)
Seien F und G äquivalente Formeln. Sei H eine Formel, die F als Teilformel besitzt. H' entstehe aus H durch Ersetzen eines Vorkommens von F in H durch G. Dann gilt: $H \equiv H'$.

Beweis: Induktion über Formelaufbau von H.

Satz: Es gelten folgende Äquivalenzen:

$(F \wedge F)$	$\equiv F$	
$(F \vee F)$	$\equiv F$	(Idempotenz)
$(F \wedge G)$	$\equiv (G \wedge F)$	
$(F \vee G)$	$\equiv (G \vee F)$	(Kommutativität)
$((F \wedge G) \wedge H)$	$\equiv (F \wedge (G \wedge H))$	
$((F \vee G) \vee H)$	$\equiv (F \vee (G \vee H))$	(Assoziativität)
$(F \wedge (F \vee G))$	$\equiv F$	
$(F \vee (F \wedge G))$	$\equiv F$	(Absorption)
$(F \wedge (G \vee H))$	$\equiv ((F \wedge G) \vee (F \wedge H))$	
$(F \vee (G \wedge H))$	$\equiv ((F \vee G) \wedge (F \vee H))$	(Distributivität)
$\neg\neg F$	$\equiv F$	(doppelte Negation)
$\neg(F \wedge G)$	$\equiv (\neg F \vee \neg G)$	
$\neg(F \vee G)$	$\equiv (\neg F \wedge \neg G)$	(de Morgansche Regeln)
$(F \vee G)$	$\equiv F$, falls F Tautologie	
$(F \wedge G)$	$\equiv G$, falls F Tautologie	(Tautologieregeln)
$(F \vee G)$	$\equiv G$, falls F unerfüllbar	
$(F \wedge G)$	$\equiv F$, falls F unerfüllbar	(Unerfüllbarkeitsregeln)

Beweis: Wahrheitstafeln (Beispiel nach Wahl)

Bemerkung 1: wegen Assoziativität können Klammern wegfallen :
(A v B v C v D) anstelle von entsprechenden vollständig geklammerten Formeln.

Weitere Klammerersparnis durch Weglassen äußerer Klammern und Bindungskonventionen:
¬ stärker als ∧ stärker als v stärker als -> stärker als <->

Damit können wir schreiben A ∧ B v C -> D statt (((A ∧ B) v C) -> D)

Bemerkung 2: Mit Hilfe der de Morganschen Regeln läßt sich jede Formel in eine äquivalente Formel ohne v bzw. ohne ∧ umwandeln. {¬,∧} bzw. {¬,v} genügen also eigentlich als Basis zum Aufbau der Syntax der Aussagenlogik (ebenso: {¬, ->} und andere Mengen).

Beispiel zum Gebrauch der Äquivalenzen:

¬Bier -> Fisch	Bier v Fisch	(1)
Eis v ¬Bier -> ¬Fisch	¬(Eis v ¬Bier) v ¬Fisch	(2)

(Bier v Fisch) ∧ (¬(Eis v ¬Bier) v ¬Fisch)	de Morgan
≡ (Bier v Fisch) ∧ ((¬Eis ∧ ¬¬Bier) v ¬Fisch)	doppelte Negation
≡ (Bier v Fisch) ∧ ((¬Eis ∧ Bier) v ¬Fisch)	Kommutativität
≡ (Bier v Fisch) ∧ (¬Fisch v (¬Eis ∧ Bier))	Distributivität
≡ (Bier v Fisch) ∧ ((¬Fisch v ¬Eis) ∧ (¬Fisch v Bier))	Kommutativität
≡ (Bier v Fisch) ∧ ((Bier v ¬Fisch) ∧ (¬Fisch v ¬Eis))	Assoziativität
≡ ((Bier v Fisch) ∧ (Bier v ¬Fisch)) ∧ (¬Fisch v ¬Eis))	Distributivität
≡ (Bier ∧ (Fisch v ¬Fisch)) ∧ (¬Fisch v ¬Eis)	Kommutativität
≡ ((Fisch v ¬Fisch) ∧ Bier) ∧ (¬Fisch v ¬Eis)	Tautologieregel
≡ Bier ∧ (¬Fisch v ¬Eis)	

Def.: (Normalformen)
Ein Literal ist eine atomare Formel (positives Literal) oder die Negation einer atomaren Formel (negatives Literal).
Eine Formel F ist in konjunktiver Normalform (KNF), falls sie eine Konjunktion von Disjunktionen von Literalen ist.
Eine Formel F ist in disjunktiver Normalform (DNF), falls sie eine Disjunktion von Konjunktionen von Literalen ist.

Satz: Für jede Formel F gibt es eine äquivalente Formel in KNF und eine äquivalente Formel in DNF.

Beweis durch Induktion über Formelaufbau von F.

Bemerkung: es gibt unterschiedliche Formeln in DNF (KNF), die äquivalent sind:
Beispiel (DNF)
(A ∧ B ∧ C) v (A ∧ B ∧ ¬C) v F äquivalent zu
(A ∧ B) v F

Umformung in KNF:

Gegeben: Formel F (Abkürzungen -> und <-> seien bereits ersetzt)

1. Ersetze in F jedes Vorkommen einer Teilformel der Bauart

¬¬G	durch G
¬(G v H)	durch (¬G ∧ ¬H)
¬(G ∧ H)	durch (¬G v ¬H)

bis keine solche Teilformel mehr vorkommt.

2. Ersetze in F jedes Vorkommen einer Teilformel der Bauart

(F v (G ∧ H))	durch ((F v G) ∧ (F v H))
((F ∧ G) v H))	durch ((F v H) ∧ (G v H))

bis keine solche Teilformel mehr vorkommt (ausdistribuieren).

(Bei Umformung in DNF muss im letzten Schritt v und ∧ vertauscht werden)

Beispiel:

(A -> B) -> (B -> C)	Eliminieren ->
¬(A -> B) v (¬B v C)	Eliminieren ->
¬(¬A v B) v (¬B v C)	de Morgan, doppelte Negation
(A ∧ ¬B) v ¬B v C	schon in DNF, Distributivität
((A v ¬B) ∧ (¬B v ¬B)) v C	Distributivität
(A v ¬B v C) ∧ (¬B v ¬B v C)	KNF

Weitere Vereinfachung: wegen Idempotenz und da A ∧ (A v B) äquivalent zu A:
(¬B v C) (diese Schritte sind im obigen Algorithmus noch nicht enthalten)

Bemerkung: KNF und DNF können auch direkt aus Wahrheitstafel abgelesen werden.

DNF: Die Konjunktionen von Literalen entstehen aus Zeilen, für die der Wert von F 1 ist. Atome, deren Wert in der Zeile 1 ist, werden positive Literale, die anderen negative.

KNF: Die Disjunktionen von Literalen entstehen aus Zeilen, für die Wert von F 0 ist. Atome, deren Wert in der Zeile 0 ist, werden positive Literale, die anderen negative.

(dass das so korrekt ist sieht man wie folgt: Bilde DNF für ¬F, wende de Morgan an. Das liefert Formel ¬G, so dass G in KNF, also G KNF von F. In G wurden alle positiven Literale der DNF zu negativen und umgekehrt)

Beispiel:

ABC	(A -> B)	(B -> C)	(A -> B) -> (B -> C)
0 0 0	1	1	1
0 0 1	1	1	1
0 1 0	1	0	0
0 1 1	1	1	1
1 0 0	0	1	1
1 0 1	0	1	1
1 1 0	1	0	0
1 1 1	1	1	1

Ablesen DNF:

$(\neg A \wedge \neg B \wedge \neg C) \vee (\neg A \wedge \neg B \wedge C) \ldots$

Ablesen KNF:

$(A \vee \neg B \vee C) \wedge (\neg A \vee \neg B \vee C)$
äquivalent zu $(\neg B \vee C)$

1.4 Hornformeln

wichtiger, effizient zu behandelnder Spezialfall (benannt nach Alfred Horn)

Def.: (Hornformeln)
Eine Formel F ist eine Hornformel, falls F in KNF ist und jedes Konjunktionsglied (also jede Disjunktion) in F höchstens ein positives Literal enthält.

Bsp.: $(A \vee \neg B) \wedge (\neg C \vee \neg A \vee D) \wedge (\neg A \vee \neg B) \wedge D \wedge \neg E$

äquivalent: $(B \rightarrow A) \wedge (C \wedge A \rightarrow D) \wedge (A \wedge B \rightarrow 0) \wedge (1 \rightarrow D) \wedge (E \rightarrow 0)$
0 steht für unerfüllbare Formel, 1 für Tautologie

Effizienter Erfüllbarkeitstest für Hornformeln:

Eingabe: Hornformel F

1. Markiere jedes Vorkommen von A in F, falls F Teilformel der Form $(1 \rightarrow A)$ besitzt;
2. while es gibt in F Teilformel $G = (A1 \wedge \ldots \wedge An \rightarrow B)$ oder $(A1 \wedge \ldots \wedge An \rightarrow 0)$ mit
   $A1 \wedge \ldots \wedge An$ markiert, B unmarkiert do
   if G hat erste Form then markiere jedes Vorkommen von B
   else gib aus "unerfüllbar" und stoppe;
3. gib aus "erfüllbar" und stoppe.

Die erfüllende Belegung ergibt sich aus den markierten Atomen: I(Ai) = 1 gdw Ai markiert.

Satz: Obiger Markierungsalgorithmus ist korrekt und stoppt nach maximal n Markierungsschritten (n = Anzahl der Atome in F).

Beweisskizze:
Komplexitätsabschätzung offensichtlich.

Korrektheit: Schritte 1 und 2 markieren nur solche Atome, die in allen Modellen von F den Wert 1 haben müssen. Falls es dazu kommt, dass 0 markiert werden müßte (else-Fall), kann es kein Modell geben und die Ausgabe in 2. ist korrekt.
Ansonsten ist nach Verlassen von 2. die Belegung I mit I(Ai) = 1 gdw. Ai markiert ein Modell von F, denn für jede Implikation G in F gilt: entweder die Vorbedingung und die

Nachbedingung von G sind wahr in I oder die Vorbedingung von I ist falsch. In beiden Fällen ist die Implikation wahr.

Beispiel:

4-Beiner ∧ Miaut -> Katze	4-Beiner ∧ Bellt -> Hund
Katze -> Haustier	Hund -> Haustier
Katze -> Jagt-Mäuse	Hund -> Jagt-Katzen
1 -> 4-Beiner	1 -> Bellt

[Haustier ∧ Jagt-Katzen -> 0]

Haustier ∧ Jagt-Katzen ableitbar?
gdw WB ∧ ¬(Haustier ∧ Jagt-Katzen) unerfüllbar
gdw WB ∧ (Haustier ∧ Jagt-Katzen -> 0) unerfüllbar.

Bemerkung1 : der Algorithmus berechnet das kleinste Modell von F. Hierbei gilt:
M = M' gdw M(A) = M'(A) für alle atomaren Formeln A (wie üblich ist 0 < 1).
Für allgemeine Formeln kann es mehrere minimale Modelle geben (etwa A v B).

Bemerkung 2: Falls eine Hornformel keine Implikation der Form (A1 ∧ ... ∧ An -> 0) enthält, so ist sie immer erfüllbar. Ebenso wenn keine Teilformel der Form (1 -> A) vorkommt.

1.5 Endlichkeitssatz (Kompaktheitstheorem)

Satz: Eine Menge M von Formeln ist erfüllbar gdw. jede der endlichen Teilmengen von M erfüllbar ist.

Beweis: "=>" offensichtlich, da Modell für M auch Modell für jede Teilmenge von M.
"<=". Für jedes n = 1 sei Mn die Menge der Formeln in M, die nur Atome A1, ..., An enthalten. Es gibt k = 2 hoch 2 hoch n verschiedene, nicht äquivalente Formen in Mn.
Also: für jedes F in Mn gibt es i = k mit F ≡ Fi. Jedes Modell für {F1, ..., Fk} ist Modell für Mn. So ein Modell existiert nach Voraussetzung. Wir nennen es I_n. Es ist gleichzeitig Modell für M1, M2, ..., Mn-1 (denn diese Mengen sind Teilmengen von Mn).

Das gesuchte Modell I für M wird so konstruiert (wir schreiben (An, 1) in I, statt I(An) = 1). Ebenso für 0):

Stufe 0: I := {};
J := {1,2,3 ...}.

Stufe n > 0: if es gibt unendlich viele Indizes i in J mit $I_i(A_n) = 1$ then
begin I := I ∪ {(A_n,1)};
J := J - {i | $I_i(A_n)$? 1}
end
else begin I := I ∪ {(A_n,0)};
J := J - {i | $I_i(A_n)$? 0}
end.

In jeder Stufe n wird I um (An,0) oder (An,1) erweitert, aber nie um beides. I ist daher wohldefinierte Funktion mit Definitionsbereich {A1, A2, ...} und Wertebereich {0,1}.

Wir zeigen: I ist Modell für M. Sei F beliebige Formel in M. In F kommen endlich viele Atome vor, der höchste Index eines solchen Atoms sei k, d.h. F in Mk, Mk+1, ... und jede der Belegungen Ik, Ik+1, ... ist Modell von F.

Bei der Konstruktion von I wird J zwar "ausgedünnt", aber nie endlich. Damit verbleiben auch unendlich viele Indizes i > k in J. Für all diese i gilt: $I_i(A_1) = I(A_1)$, ..., $I_i(A_k) = I(A_k)$, und deshalb ist I Modell von F.

Bemerkung: Beweis nicht-konstruktiv: if-Bedingung nicht algorithmisch überprüfbar. Reine Gedankenkonstruktion, I kann nicht tatsächlich von Algorithmus berechnet werden.

Satz später in folgende Form verwendet: wenn unendliche Menge unerfüllbar ist, dann gibt es eine endliche Teilmenge, die bereits unerfüllbar ist.

1.6 Resolution

syntaktische Umformungsregel, die aus zwei Formeln eine neue erzeugt (die Resolvente). wird verwendet, um Unerfüllbarkeit einer Formel(menge) zu testen.

Anmerkung:
Mengen von syntaktischen, mechanisch ausführbaren Umformungsregeln heißen auch Kalküle. Kalküle erzeugen aus vorgegebenen Mengen von Formeln neue.
Kalküle heißen korrekt, wenn sie nur die gewünschten Formeln ableiten,
vollständig, wenn sie alle gewünschten Formeln ableiten

Resolution setzt voraus, dass Formel(menge) in KNF vorliegt (sonst Umformen).

Bequem ist Darstellung der KNF in Mengennotation: statt
$F = (L_{1,1} \vee ... \vee L_{1,n1}) \wedge \ldots \wedge (L_{k,1} \vee ... \vee L_{k,nk})$
schreiben wir
$F = \{\{L_{1,1}, ..., L_{1,n1}\}, ..., \{L_{k,1}, ..., L_{k,nk}\}\}$

Jedes Element von F heißt Klausel.

Def.:
Seien K1, K2 Klauseln. R heißt Resolvente von K1 und K2, falls es ein Literal L gibt mit L in K1 und –L in K2 und es gilt: $R = (K1 - L) \cup (K2 - -L)$.
Hierbei ist $-L = \neg L$ falls $L=Ai$, $-L = Ai$ falls $L = \neg Ai$.

Beispiel: {A,¬C} ist Resolvente aus {A, B} und {¬B, ¬C}.

Spezialfall: die leere Klausel {} entsteht durch Resolvieren von widersprüchlichen Einerklauseln, etwa {A},{¬A}. Sie entspricht falsch und wird auch als ⊥ oder [] repräsentiert.

Frage: Kann Resolution von Hornformeln nicht-Hornformel produzieren?

Resolutionslemma:
Sei F eine Formel in KNF, dargestellt als Klauselmenge, R Resolvente zweier Klauseln aus F. Dann sind F und $F \cup \{R\}$ äquivalent.

Bew.: Sei I eine Belegung. Zu zeigen: $I \models F \cup \{R\}$ gdw. $I \models F$.
Falls $I \models F \cup \{R\}$, dann gilt offensichtlich auch $I \models F$.
Es gelte umgekehrt $I \models F$. Dann gilt $I \models K$ für jede Klausel K in F. Ferner sei $R = (K1 - L) \cup (K2 - \neg L)$.
Fall 1: $I \models L$. Dann gilt $I \models K2 - \neg L$ und deshalb $I \models R$.
Fall 2: $I \not\models L$. Dann gilt $I \models K1 - L$ und deshalb $I \models R$.

Bemerkung: Resolvente nicht äquivalent zu beiden resolvierten Klauseln, schwächer. Deshalb werden resolvierte Klauseln nicht ersetzt.
$A \vee \neg B, B \vee C \Rightarrow A \vee C$ $\{\neg A,B,C\}$ ist Modell der Resolvente, aber nicht der resolvierten Klauseln.

Def.:
Sei F eine Klauselmenge. Wir definieren:

$$\begin{aligned} Res(F) &= F \cup \{R \mid R \text{ ist Resolvente zweier Klauseln in } F\} \\ Res^0(F) &= F \\ Res^{n+1}(F) &= Res(Res^n(F)) \\ Res^*(F) &= \cup_{n=0} Res^n(F) \end{aligned}$$

Frage: Bestimme Res1(F) für $F = \{\{A,\neg B,C\}, \{B,C\},\{\neg A,C\}, \{B,\neg C\},\{\neg C\}\}$

Res1: ... $\{A,C\}, \{\neg B,C\}, \{A,C,\neg C\}, \{A,B,\neg B\},\{A,\neg B\}, \{B\}, \{\neg A,B\},\{\neg A\}$

Bemerkung: Wieviele verschiedene Klauseln aus n Atomen kann es geben? 4^n (jedes Ai kann nicht , pos, neg, doppelt vorkommen). Daraus folgt für endliche Klauselmengen:
es gibt ein k, so dass $Res^k(F) = Res^{k+1}(F) = ... = Res^*(F)$.

Resolutionssatz (der Aussagenlogik)
Eine Klauselmenge F ist unerfüllbar gdw. $\perp$ in $Res^*(F)$.

Beweis: "<=" folgt aus Resolutionslemma: F äquivalent $Res^n(F)$ für alle n. Da $\perp$ in $Res^k(F)$ für ein k und damit $Res^k(F)$ unerfüllbar ist, so muss auch F unerfüllbar sein.
"=>" Sei F unerfüllbar. Falls F unendlich, so muss es wegen Kompaktheit endliche Teilmenge von F geben, die unerfüllbar ist. Wir können uns deshalb auf endliche Formelmengen beschränken.
Wir zeigen $\perp$ in $Res^*(F)$ durch Induktion über die Anzahl n der in F vorkommenden Atome:
Induktionsanfang: n = 0, dann muss $F = \{\perp\}$ sein und damit $\perp$ in $Res^*(F)$.
Induktionsschritt: Sei F eine Klauselmenge mit Atomen A1, ..., An+1. Wir definieren aus F zwei Klauselmengen F_0 und F_1 wie folgt:

F_0 entsteht aus F durch Streichen jedes Vorkommens von A_{n+1} in einer Klausel sowie durch Streichen der Klauseln, in denen $\neg A_{n+1}$ vorkommt. (Belegung von A_{n+1} mit 0 fixieren).
F_1 entsteht aus F durch Streichen jedes Vorkommens von $\neg A_{n+1}$ in einer Klausel sowie durch Streichen der Klauseln, in denen A_{n+1} vorkommt. (Belegung von A_{n+1} mit 1 fixieren).
F_0 und F_1 sind unerfüllbar. Wäre etwa F0 erfüllbar, so könnte man aus einem Modell M für F_0 ein Modell M' für F konstruieren, indem man A1, ..., An wie in M auswertet und An+1 den Wert 0 gibt. Analog kann gezeigt werden, dass F_1 unerfüllbar ist.
Damit ist die Induktionsvoraussetzung auf F_0 und F_1 anwendbar.
Es muss also einen Resolutionsbeweis aus F0 für $\perp$ geben, d.h. eine Folge von Klauseln K1, ..., Km, so dass Km = $\perp$ und jedes Ki entweder in F_0 ist oder Resolvente zweier Klauseln, die in der Folge früher auftreten. Benutzt man nun statt K1, ..., Km die ursprünglichen Klauseln, erhält man einen Resolutionsbeweis entweder wieder für $\perp$, oder für A_{n+1}. Im ersten Fall ist bewiesen, dass die leere Klause aus F ableitbar ist.

Ebenso gibt es einen Resolutionsbeweis für $\perp$ aus F_1, der durch Verwendung der ursprünglichen Klauseln zu einem Beweis für $\perp$ oder für $\neg A_{n+1}$ wird. Wieder ist im ersten Fall die Behauptung bewiesen. Ansonsten lässt sich aus An+1 und $\neg A_{n+1}$ in einem weiteren Resolutionsschritt die leere Klause ableiten, d.h. $\perp$ in Res*(F).

Aus Resolutionssatz lässt sich folgender Algorithmus ableiten:

```
Eingabe: Formel in KNF (repräsentiert als Klauselmenge)
repeat  G := F;
        F := Res(F)
until (⊥ in F) or (F = G);
if ⊥ in F then "unerfüllbar" else "erfüllbar";
```

Es muss natürlich nicht immer ganz Res* berechnet werden, es genügt, eine Herleitung der leeren Klausel zu finden:

Def.: Eine Deduktion (Herleitung, Beweis) der leeren Klausel aus einer Klauselmenge F ist eine Folge K1, ..., Km von Klauseln für die gilt:

1) Km = $\perp$
2) für alle i (1 = i = m): entweder Ki aus F oder Ki Resolvente von Kj, Kk mit j,k < i.

früheres Beispiel: Haustier $\wedge$ Jagt-Katzen ableitbar aus:

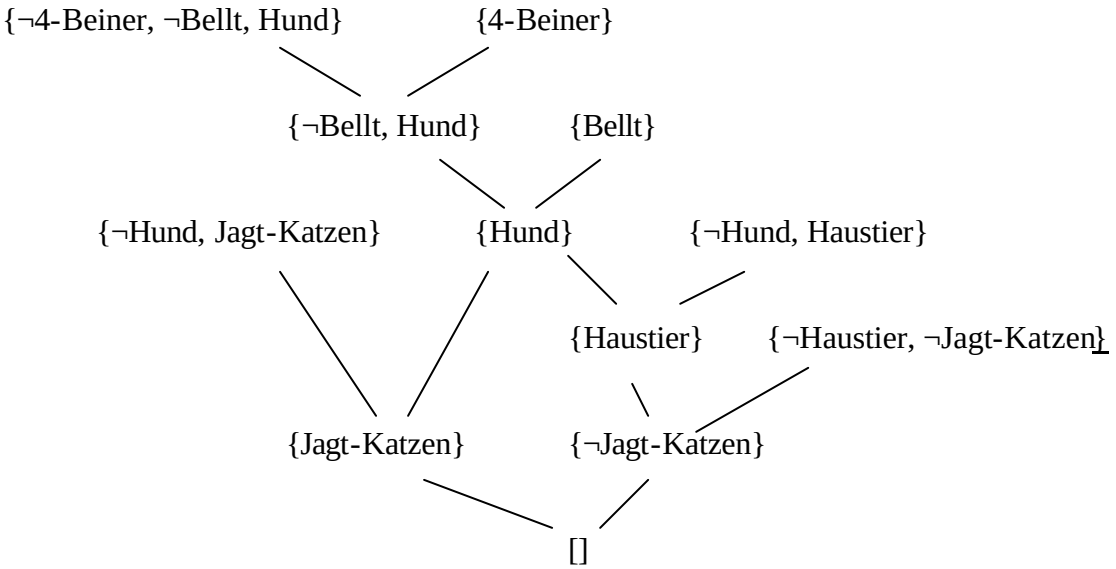
1)	4-Beiner $\wedge$ Miaut -> Katze	{¬4-Beiner, ¬Miaut, Katze}
2)	4-Beiner $\wedge$ Bellt -> Hund	{¬4-Beiner, ¬Bellt, Hund}
3)	Katze -> Haustier	{¬Katze, Haustier}
4)	Hund -> Haustier	{¬Hund, Haustier}
5)	Katze -> Jagt-Mäuse	{¬Katze, Jagt-Mäuse}
6)	Hund -> Jagt-Katzen	{¬Hund, Jagt-Katzen}
7)	4-Beiner	{4-Beiner}
8)	Bellt	{Bellt}

9) ¬Haustier v ¬Jagt-Katzen {¬Haustier, ¬Jagt-Katzen}

Deduktion:

10)	{¬Bellt, Hund}		2,7
11)	{Hund}		10,8
12)	{Haustier}		11,4
13)	{¬Jagt-Katzen}		12,9
14)	{Jagt-Katzen}	11,6	
15)	[]		14,13

Resolutionsgraph:



Weiteres Beispiel: wir zeigen mit Resolution, dass (¬B ∧ ¬C ∧ D) v (¬B ∧ ¬D) v (C ∧ D) v B Tautologie ist.

{B, C,¬D}, {B, D}, {¬C, ¬D},{¬B} unerfüllbar?
ja.

Frage: warum macht man Widerspruchsbeweise? In Katzenbeispiel wird doch Haustier und Jagt-Katzen abgeleitet?

Verfahren nur für Widerlegungsbeweise vollständig. Z. B folgt A v B aus A, aber A v B ist nicht per Resolution aus A ableitbar.

Def.: Einschränkung der Resolutionsverfahrens:
Unit-Resolution entsteht durch Beschränkung der Resolution auf Fälle, in denen mindestens eine resolvierte Klausel einelementig ist (Einer-Klausel).
Input-Resolution entsteht durch Beschränkung der Resolution auf Fälle, in denen mindestens eine resolvierte Klausel Eingabe-Klausel (aus F) ist.

Satz: Sowohl Unit- wie Input-Resolution sind vollständig für Hornklauseln, d.h. falls F Hornklauselmenge, so ist die leere Klausel ableitbar aus F gdw F unerfüllbar.

Katzen-Beispiel bereits Unit-Resolution.

Gegenbeispiel im allgemeinen Fall (Unit): {¬A,¬B}, {¬A, B}, {A, ¬B}, {A, B}. unerfüllbar, aber natürlich keine Unit-Resolution möglich.

Bei der linearen Resolution wird in jedem (bis auf den ersten) Resolutionsschritt die vorher erzeugte Resolvente verwendet (zusammen mit einer Klausel, die entweder in F ist oder vorher erzeugt wurde).

Satz: Lineare Resolution ist vollständig.

1.7 Das Davis-Putnam-Verfahren

(nicht: Putman!)

Erfüllbarkeitstest für Formel F = {C1, ..., Cn} in Klauselform

Def.: (reduzierte Klauselmenge)

Sei F eine Menge von Klauseln, L ein Literal. Die um L reduzierte Klauselmenge F_L entsteht aus F durch

a) Streichen aller Klauseln, die L enthalten
b) Streichen des Komplements von L aus den verbleibenden Klauseln.

Beispiel:
F = {{A, B}, {C, ¬A}, {¬B}}

Reduzieren um A ergibt:
F_A = {{C}, {¬B }}

Reduzieren um ¬B ergibt:
$F_{¬B}$ = {{A}, {C, ¬A}}

Anmerkung:
jedes Modell M von F, das L zu wahr auswertet, ist Modell von F_L.
Zu jedem Modell M' von F_L gibt es ein Modell M von F, das die Atome in F_L gleich auswertet wie M'. (M(L) = 1, M(A) = M'(A) für alle Atome, die in F_L vorkommen)
also: F_L erfüllbar genau dann wenn F Modell hat, das L zu wahr auswertet.

Beobachtungen:

1. Wenn [] in F, dann ist F unerfüllbar.
2. Wenn Einerklausel der Form {L} in F, so ist F erfüllbar genau dann wenn F_L erfüllbar.
3. Sei L Literal in F. F ist erfüllbar gdw. F_L erfüllbar oder F_{-L} erfüllbar.

Daraus lässt sich direkt der folgende rekursive Erfüllbarkeitstest ableiten:

```
boolean Satisfiable(S)          ;; liefert true falls Klauselmenge S erfüllbar, false sonst
begin
        if S = {} return true;
        if {} in S return false;
        if S enthält Einerklausel select Einerklausel {L} and return Satisfiable(S_L)
        else select Literal L and return (Satisfiable(S_L) or Satisfiable(S_-L));
end;
```

Beispiel 1:

erfüllbar {{A, B}, {C, ¬A}, {¬B}}		gdw.
erfüllbar {{A}, {C, ¬A}}		gdw.
erfüllbar {{C}}		gdw.
erfüllbar {}		true

Beispiel 2:

erfüllbar {{A, B}, {¬A, B}, {A, ¬B}}?		gdw. (wähle A)
erfüllbar {{B}}	oder erfüllbar {{B}, {¬B}}	gdw.
erfüllbar {}	oder erfüllbar {{}}	true

1.8 Tableauverfahren

Tableau für F: Baum, dessen Knoten mit Formeln markiert sind. Wurzel mit F markiert. Pfad von Wurzel zu Blatt repräsentiert Konjunktion der Formeln an Knoten, der gesamte Baum Disjunktion dieser Konjunktionen

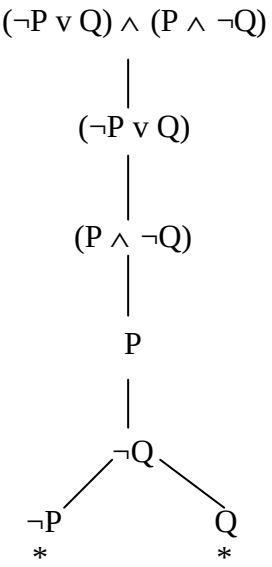
Grundidee: F ist unerfüllbar, wenn für jeden Pfad von der Wurzel zu einem Blatt gilt: die entsprechende Konjunktion ist unerfüllbar.

Erzeugungsregeln für Tableaus:

$\frac{\neg\neg H}{H}$	$\frac{G_1 \wedge G_2}{G_1 \\ G_2}$	$\frac{\neg(G_1 \wedge G_2)}{\neg G_1 \mid \neg G_2}$	$\frac{G_1 \vee G_2}{G_1 \mid G_2}$	$\frac{\neg(G_1 \vee G_2)}{\neg G_1 \\ \neg G_2}$

zu lesen:
wenn in Pfad $\neg\neg H$ vorkommt, erweitere ihn um H,
wenn in Pfad $G_1 \wedge G_2$ vorkommt, erweitere ihn um G_1 und um G_2,
wenn in Pfad $\neg(G_1 \wedge G_2)$ vorkommt, verzweige und erweitere um linken Nachfolger G_1 und rechten Nachfolger G_2,
etc.

Beispiel:



Def.: (Tableau für F)
Die Menge der Tableaus für eine Formel F ist induktiv wie folgt definiert:

1) der Baum, der aus einem mit F markierten Knoten besteht, ist ein Tableau für F.
2) wenn T ein Tableau für F ist und T' durch Anwendung einer Erzeugungsregel aus T entsteht, so ist T' ein Tableau für F.

Def.:
Ein Ast eines Tableaus ist ein Pfad von der Wurzel zu einem Blatt.
Ein Ast heißt abgeschlossen, wenn in ihm eine Formel und ihre Negation vorkommt.
Ein Tableau heißt abgeschlossen, wenn jeder Ast abgeschlossen ist.

Satz: F ist unerfüllbar gdw. es ein abgeschlossenes Tableau für F gibt.

Im Beispiel: linker Ast abgeschlossen, weil ¬P und P enthalten, rechter weil ¬Q und Q enthalten.

Vorteil: keine Umformung in KNF nötig.