Praktikum Beweiser

Praktikum SS 2004

Reinhold Letz
Institut für Informatik, TU München

Vorlesung: Donnerstags 14:15 - 15:45 Uhr, Multimediaraum MI 00.13.009A
Übung: Donnerstags 13:15 - 14:00 Uhr (ab 29.4.) ebenfalls in MI 00.13.009A
Klausur: Donnerstag, 22.07.04, 14:15 - 15:45 Uhr

Vorlesungsseite im Internet:

http://www4.in.tum.de/~letz/praktikum-beweiser-ss04.html

Einsatzbedingungen automatischer Beweiser

Logik als qualitatives „Rechnen“:

- Allgemeine Aussagen über Objekte verfügbar
- Abspeichern aller möglicherweise benötigten Kombinationen auf Grund des Umfangs nicht möglich

$\Rightarrow$ Berechnung des Ergebnisses zur Laufzeit (nach Bedarf) notwendig.

Beispiele:

- Deduktive Datenbanken
- Automatisches Beweisen mathematischer Sätze

Typische Probleme

- Ist eine Wissenbasis (ein Axiomensystem) konsistent (widerspruchsfrei)?
- Hat eine Aussagenmenge ein Modell?
- Ist eine Aussage allgemeingültig?
- Wichtige Anwendung:
  Wie lässt sich die Sicherheit („Safety“) von Systemen gewährleisten?
  $\Rightarrow$ Formale Verifikation von Software und Hardware

Informelle Argumentationen und Gültige Aussagen

Sokrates ist ein Mensch.
Alle Menschen sind sterblich.

Sokrates ist sterblich.

Sokrates ist sterblich.
Sokrates ist ein Mensch.

Alle Menschen sind sterblich.

Wenn jeder Arme einen reichen Vater hat, dann
gibt es einen Reichen mit einem reichen Großvater.

P und Q seien 2-stellige transitive Relationen.
$P \cup Q$ sei total und Q sei symmetrisch.

P ist total oder Q ist total.

Rolle der Logik

Ziel: Formalisierung von Argumentation

Mittel:

- Formale Sprache
- Interpretation
- Folgerungsbegriff
- Berechnungsverfahren
- Formalisierung und Rückübersetzung

Aussagenlogik

- Syntax und Semantik der Aussagenlogik, Gesetze, Normalformen
- Folgerungsbegriff
- Beweisverfahren
  - semantisch orientierte Verfahren
    - Wahrheitstafelmethode
    - Semantische Bäume (Davis/Putnam-Verfahren)
  - Binäre Entscheidungsdiagramme (BDDs)
- Logikkalküle (beweistheoretische Verfahren)
  - Frege/Hilbert-Kalküle
  - Sequenzen-Kalkül
  - Tableaukalkül
  - Resolutionskalkül
- Komplexitätsaspekte

Prädikatenlogik

- Einführung, Syntax und Semantik der Prädikatenlogik
- Gesetze, Normalformen
- Folgerungsbegriff
- Beweisverfahren, vertieft auf Prädikatenlogik
  - Tableauverfahren
  - Resolution
- Implementierungsaspekte
- Beweiser (SETHEO, DCTP, E)
- Problemsammlung (TPTP)

Syntax der Aussagenlogik

Alphabet einer Aussagenlogik:

- Eine unendliche Menge $\{p_1, p_2, \ldots\}$ von aussagenlogischen Variablen.
- Eine Menge von Konnektiven wie $\neg$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$, $\top$ und $\bot$ mit Stelligkeiten wie folgt: 0-stellig: $\top$ und $\bot$, 1-stellig: $\neg$, 2-stellig: $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$.

Eine atomare aussagenlogische Formel ist eine aussagenlogische Variable.

Die Menge der aussagenlogischen Formeln P für ein aussagenlogisches Alphabet A ist die kleinste Menge, für die gilt:

1. P enthält alle aussagenlogischen Variablen von A und alle 0-stelligen Konnektive.
2. Wenn $F \in P$, dann auch $\neg F$.
3. Für alle zweistelligen Konnektive $\circ$ und $F, G \in P$ gilt $(F \circ G) \in P$.

Semantik der Aussagenlogik

Interpretation/Belegung: Eine Interpretation oder Belegung für eine Aussagenlogik P mit Variablenmenge V ist eine Abbildung $\iota : \mathcal{V} \longrightarrow \{\mathbf{w}, \mathbf{f}\}$, wobei $\{\mathbf{w}, \mathbf{f}\}$ Wahrheitswertmenge heisst.

Fortsetzung einer Interpretation ι auf P wie folgt:

1. $\iota(\top) = \mathbf{w}$, $\iota(\bot) = \mathbf{f}$.

2. $\iota(\neg F) = \mathbf{w}$ falls $\iota(F) = \mathbf{f}$, ansonsten $\mathbf{f}$

3. $\iota((F \wedge G)) = \mathbf{w}$ gdw $\iota(F) = \iota(G) = \mathbf{w}$, ansonsten $\mathbf{f}$

4. $\iota((F \vee G)) = \iota(\neg(\neg F \wedge \neg G))$

5. $\iota((F \rightarrow G)) = \iota((\neg F \vee G))$

6. $\iota((F \leftrightarrow G)) = \iota(((F \rightarrow G) \wedge (G \rightarrow F)))$

Modellbegriff, Allgemeingültigkeit, Erfüllbarkeit

Eine Interpretation/Belegung ι erfüllt eine Formel F, wenn $\iota(F) = \mathbf{w}$, eine erfüllende Belegung heisst auch Modell der Formel.

Eine aussagenlogische Formel F ist

- erfüllbar, wenn sie ein Modell hat
- unerfüllbar, wenn sie kein Modell hat.
- allgemeingültig oder eine Tautologie, wenn alle Interpretation Modelle sind

Wichtiger Satz: F ist allgemeingültig $\Leftrightarrow \neg F$ unerfüllbar ist.

Folgerungsbegriff

Beim logischen Argumentieren schliessen wir aus einer Reihe von Annahmen $A_1, A_2, \ldots, A_n$ auf eine Formel F, in Zeichen:

$$\frac{A_1 \; A_2 \; \cdots \; A_n}{F}$$

Andere Aussagen (Theoreme) folgern wir aus der leeren Annahmenmenge, zum Beispiel :

$$\frac{}{(F \vee \neg F)}$$

Die nachfolgende Definitionen von $\ldots \models \ldots$ (und die im Abschnitt Kalküle folgende Definition von $\ldots \vdash \ldots$) stellen ebenfalls einen Bezug zwischen Annahmen (linke Seite) und Folgerungen (rechte Seite) dar.

Semantischer Folgerungsbegriff

- Eine Formel(menge) Δ folgt (semantisch) aus einer Formel(menge) Γ, in Zeichen

$$\Gamma \models \Delta$$

  gdw jede Interpretation, die (alle Formeln in) Γ erfüllt, auch (alle Formeln in) Δ erfüllt.

- (Die leere Formelmenge wird damit von jeder Interpretation erfüllt, man kann sie alo durch die die Formel $\top$ ersetzen).

- Falls F aus der leeren Annahmenmenge folgt (abgekürzt: $\models F$), dann ist F allgemeingültig.

Eigenschaften des Semantischen Folgerungsbegriffs

Seien Γ, Δ Formelmengen und

$$Th(\Delta) := \{F \mid \Delta \models F\}$$

d.h. die Menge der (semantischen) Folgerungen aus Δ.

Dann gelten folgende Eigenschaften für den Folgerungsbegriff:

- $\Delta \subseteq \Gamma \Rightarrow Th(\Delta) \subseteq Th(\Gamma)$ (Monotonie)
- $Th(Th(\Delta)) = Th(\Delta)$ (Idempotenz)
- $\Delta \subseteq Th(\Delta)$

Anwendung der Begriffe/Formalisierung

Natürlichsprachliche Argumentation:

> Eine notwendige Bedingung einer Reisekostenerstattung ist die fristgerechte Antragsstellung. Die Frist ist abgelaufen. Es ist daher keine Reisekostenerstattung möglich.

Formalisierung der Argumentation in Aussagenlogik:

$p :=$ Reisekostenerstattung

$q :=$ fristgerechte Antragstellung

$$\frac{\begin{array}{c} p \rightarrow q \\ \neg q \end{array}}{\neg p}$$

Umgesetzt mittels des semantischen Folgerungsbegriffs: $\{p \rightarrow q, \neg q\} \models \neg p$

Deduktionstheorem der Aussagenlogik

Zusammenhang zwischen dem (semantischen) Folgerungsbegriff und der (materialen) Implikation (bzw. der (mat.) Äquivalenz):

Für die semantische Folgerung einer Formel F aus einer Formel G gilt in der Aussagenlogik:

$$G \models F \qquad \Leftrightarrow \qquad \models G \rightarrow F$$

Die Formeln F und G heissen semantisch äquivalent (in Zeichen $F \equiv G$), gdw

$$F \models G \;\text{ und }\; G \models F$$

bzw.(mit Deduktionstheorem)

$$\models F \leftrightarrow G$$

Zentrale Probleme der Aussagenlogik

Beweis der Erfüllbarkeit einer Formel

Gegeben eine aussagenlogische Formel F:

- Zeige, dass F erfüllbar ist.
- Gebe ein Modell ι für F an. (Belegung ι als Zeuge/Beweisobjekt)
- Probleme sind NP-vollständig.

Beweis der Allgemeingültigkeit/Unerfüllbarkeit einer Formel

Gegeben eine aussagenlogische Formel F:

- Zeige, dass F allgemeingültig ist.
- Zeige, dass $\neg F$ unerfüllbar ist.
- Probleme sind coNP-vollständig.
- Gibt es immer „kurze“ Beweise für Allgemeingültigkeit/Unerfüllbarkeit?
  Wenn nein, dann gilt: NP $\neq$ coNP, und damit P $\neq$ NP.

Wahrheitstafelmethode

Aufgabe: Entscheidung der Erfüllbarkeit einer Formel

- Am Beispiel (es_regnet $\rightarrow$ strasse_nass) $\wedge \neg$ strasse_nass
- Zur Abkürzung: r $\hat{=}$ es_regnet, n $\hat{=}$ strasse_nass
- Erstellen einer Tafel mit allen Belegungen
- „Auswerten“ der Formel für jede Belegung

r	n	(	r	$\rightarrow$	n	)	$\wedge$	$\neg$	n
w	**w**		**w**	**w**	**w**		**f**	**f**	**w**
w	**f**		**w**	**f**	**f**		**f**	**w**	**f**
f	**w**		**f**	**w**	**w**		**f**	**f**	**w**
f	**f**		**f**	**w**	**f**		**w**	**w**	**f**

Effizienz des Verfahrens

- Jeder einzelne Auswertungsschritt ist linear (Zeit) in der Grösse von F
- Die Anzahl der Zeilen ist exponentiell in der Anzahl der Variablen von F
- Platzbedarf: naiv exponentiell, linear bei Wiederverwendung von Zeilen

Normalformen

- „Standarddarstellung“ für logische Formeln
- Berechnung von zentraler Bedeutung für das automatische Beweisen
- erlaubt die Erkennung und Beseitigung von Redundanzen
- Manche Inferenzregeln verlangen Eingaben in einer Normalform
- Nachteile:
  - Die ursprüngliche Gestalt der Formel und die so kodierte Information können bei der Normalformerzeugung verloren gehen
  - Die Grösse der Formel kann u.U. stark anwachsen

Literale und Klauseln, Klammerregeln

Literal:	eine Aussagenvariable oder eine negierte Aussagenvariable.
Komplement:	eines Literals ($\sim L$) bezeichne die jeweils andere Form.
Klausel:	Disjunktion von n Literalen ($n = 0$ zugelassen)
Leere Klausel:	(in Zeichen: $\square$) 0 Literale, enspricht $\bot$
Einerklausel:	1 Literal
Vollkonjunktion:	Konjunktion aus Literalen, wobei jede Variable (einer endlichen Menge $\mathcal{V}$) genau einmal vorkommt.

Klammerregeln:

- Mit der Vereinbarung einer abnehmenden Bindungsstärke von $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$ können in vielen Fällen Klammern weggelassen werden. Beispiel: $((p \wedge q) \vee r)$ kann auch als $(p \wedge q \vee r)$ notiert werden.
- Durch die Assoziativität von $\wedge$ und $\vee$ können entsprechend alle inneren Klammern der Klauseln weggelassen werden.
- Ebenso können die äussersten Klammern weggelassen werden.

Definitionen (leere Ausdrücke)

Bei einigen der folgenden Beweisverfahren erhält man durch Streichen von Literalen und Klauseln eventuell leere Klauseln und leere Konjunktionen. Die folgenden Definitionen befassen sich mit diesen Fällen.

- Die leere Konjunktion hat den Wahrheitswert **w**, da bei einer Konjunktion $C = c_1 \wedge \ldots \wedge c_n$ gilt:

  $$C = \mathbf{w} \text{ gdw für alle } c_i (1 \leq i \leq n) : c_i = \mathbf{w}$$

  Für $n = 0$ ist dies trivialerweise erfüllt.

- Die leere Disjunktion hat den Wahrheitswert **f**, da bei einer Disjunktion $D = d_1 \vee \ldots \vee d_n$ gilt:

  $$D = \mathbf{w} \text{ gdw es gibt ein } d_i (1 \leq i \leq n) : d_i = \mathbf{w}$$

  Für $n = 0$ kann dies trivialerweise nicht eintreffen.

- Als Folge davon hat die leere Klausel ($\square$) den Wahrheitswert **f**, und ebenso jede Klauselmenge, die die leere Klausel enthält.

Normalformen

Eine Formel F ist in Negationsnormalform (NNF), wenn das Negationszeichen nur als Teil eines Literals in F vorkommt

Beispiel: $\neg(p \vee q)$ ist nicht in NNF, $(\neg p \wedge \neg q)$ ist in NNF.

Konjunktive Normalform (KNF): Formel ist eine Konjunktion von Disjunktionen von Literalen.

Formal: seien $L_{i,j}$ die Literale von F, dann hat F die Gestalt $\bigwedge_{i=1}^{n}(\bigvee_{j=1}^{m_i} L_{i,j})$

Beispiel: $p \wedge r$, $p \vee r$, $((p \vee q) \wedge (p \vee \neg r))$, nicht in KNF: $((p \wedge q) \vee (p \wedge \neg r))$, $((p \vee q) \wedge (p \vee (q \wedge r)))$

Disjunktive Normalform (DNF): Formel ist eine Disjunktion von Konjunktionen von Literalen

Formal: seien $L_{i,j}$ die Literale von F, dann hat F die Gestalt $\bigvee_{i=1}^{n}(\bigwedge_{j=1}^{m_i} L_{i,j})$

Beispiel: $p \wedge q$, $p \vee q$, $((p \wedge q) \vee (p \wedge \neg q))$. Nicht in DNF: $((p \wedge q) \vee (p \wedge (q \vee r)))$

Normalformen (2)

Kanonische Disjunktive Normalform (KDNF): Formel ist eine Disjunktion von Vollkonjunktionen von Literalen

Formal: $\bigvee_{i=1}^{n}(\bigwedge_{j=1}^{m} L_{i,j})$

Beispiel: $((p \wedge q) \vee (p \wedge \neg q) \vee (\neg p \wedge q))$ ist in KDNF, wenn die Menge der Aussagenvariablen $\{p, q\}$ ist.

Formeln in KNF oder DNF kann man auch als Mengen von Mengen von Literalen notieren.

Beispiel: $((p \wedge q) \vee (p \wedge \neg q) \vee (\neg p \wedge q))$

wird zu $\{\{p, q\}, \{p, \neg q\}, \{\neg p, q\}\}$

Erzeugungsschema DNF / KNF

Mittels der Wahrheitstafelmethode

- Für die Formel wird die Wahrheitswertetafel aufgestellt
- Für die Erzeugung der DNF
  - Nimmt man die Zeilen, die zu **w** auswerten und
  - erzeugt für diese Zeilen je eine Konjunktion
  - mit **w** belegte Variable werden positiv übernommen
  - mit **f** belegte Variable werden negiert übernommen
- Für die Erzeugung der KNF
  - Nimmt man die Zeilen, die zu **f** auswerten und
  - erzeugt für diese Zeilen je eine Disjunktion
  - mit **w** belegte Variable werden negiert übernommen
  - mit **f** belegte Variable werden positiv übernommen

Erzeugungschema (Beispiel)

Gegeben sei die Formel F mit der Wahrheitstafel

p	q	r	F	p	q	r	F
w	**w**	**w**	**f**	**w**	**w**	**f**	**f**
w	**f**	**w**	**w**	**w**	**f**	**f**	**w**
f	**w**	**w**	**f**	**f**	**w**	**f**	**f**
f	**f**	**w**	**f**	**f**	**f**	**f**	**w**

ergibt als KDNF-Formel: $(p \wedge \neg q \wedge r) \vee (p \wedge \neg q \wedge \neg r) \vee (\neg p \wedge \neg q \wedge \neg r)$

bzw. vereinfacht: $(p \wedge \neg q) \vee (\neg p \wedge \neg q \wedge \neg r)$

oder als KNF-Formel:
$(\neg p \vee \neg q \vee \neg r) \wedge (\neg p \vee \neg q \vee r) \wedge (p \vee \neg q \vee \neg r) \wedge (p \vee \neg q \vee r) \wedge (p \vee q \vee \neg r)$

bzw. vereinfacht: $\neg q \wedge (p \vee \neg r)$

Erzeugung KNF (II)

Das Verfahren mit den Wahrheitswerten bietet sich nur für Formeln mit wenigen Variablen an. Allgemeiner verwendbar ist folgende Methode der systematischen Umformung (hier für KNF):

1. Man beseitige alle Operatoren außer $\vee$, $\wedge$ und $\neg$ durch äquivalente Ausdrücke mit diesen drei Operatoren
2. Man ersetze jedes Vorkommen einer Teilformel der Form
   $\neg\neg F$ durch F
   $\neg(F \vee G)$ durch $(\neg F \wedge \neg G)$
   $\neg(F \wedge G)$ durch $(\neg F \vee \neg G)$
   bis keine solche Teilformel mehr vorkommt
3. Man ersetze jedes Vorkommen einer Teilformel der Form
   $(F \vee (G \wedge H))$ durch $((F \vee G) \wedge (F \vee H))$
   $((F \wedge G) \vee H)$ durch $((F \vee H) \wedge (G \vee H))$
   bis keine solche Teilformel mehr vorkommt
4. Die resultierende Formel ist in KNF.

Erzeugung KNF II (Beispiel)

Gegeben sei folgende Formel F:

$$(p \vee \neg q) \rightarrow r$$

Wir formen F um:

$(p \vee \neg q) \rightarrow r$	
$\neg(p \vee \neg q) \vee r$	Beseitigung von $\rightarrow$
$(\neg p \wedge \neg\neg q) \vee r$	deMorgan
$(\neg p \wedge q) \vee r$	DoppelNegation
$(\neg p \vee r) \wedge (q \vee r)$	Distributivität

und wir haben die zu F äquivalente KNF

Komplexität der NF-Transformation (Beispiel)

Gegeben sei $F = (G \leftrightarrow (H \leftrightarrow I))$ $\equiv$

$G \leftrightarrow ((H \rightarrow I) \wedge (I \rightarrow H))$ $\equiv$

$G \leftrightarrow ((\neg H \vee I) \wedge (\neg I \vee H))$ $\equiv$

$((G \rightarrow ((\neg H \vee I) \wedge (\neg I \vee H))) \wedge [((\neg H \vee I) \wedge (\neg I \vee H)) \rightarrow G])$ $\equiv$

$((\neg G \vee ((\neg H \vee I) \wedge (\neg I \vee H))) \wedge [((H \wedge \neg I) \vee (I \wedge \neg H)) \vee G])$ $\equiv$

$((\neg G \vee \neg H \vee I) \wedge (\neg G \vee \neg I \vee H)) \wedge [((H \wedge \neg I) \vee I) \wedge ((H \wedge \neg I) \vee \neg H) \vee G])$ $\equiv$

$((\neg G \vee \neg H \vee I) \wedge (\neg G \vee \neg I \vee H)) \wedge [((H \vee I) \wedge (\neg I \vee I)) \wedge ((H \vee \neg H) \wedge (\neg I \vee \neg H)) \vee G])$

$\equiv$

$$((\neg G \vee \neg H \vee I) \wedge (\neg G \vee \neg I \vee H) \wedge (H \vee I \vee G) \wedge (\neg I \vee \neg H \vee G))$$

Für das Formelschema $(F_1 \leftrightarrow (F_2 \leftrightarrow (\ldots F_n)))$ entsteht damit nach Reduktion der Tautologien eine Formel mit 2^{n-1} Elementen der Länge n.

(Die Vermutung, dass es sich dabei um die kürzeste äquivalente KNF handelt, bleibt zu beweisen.)

Komplexität der NF-Transformation

Satz: Im allgemeinen gibt es zu einer beliebigen Formel F nicht immer eine logisch äquivalente Formel in Klauselnormalform polynomieller Grösse.

Durch die Einführung von Abkürzungen kann jedoch eine Formel (die neue Variablen enthält) mit linearer (Aussagenlogik) bzw. quadratischer (Prädikatenlogik) Grösse gefunden werden.

Satz: Die Transformation erhält Erfüllbarkeit und Unerfüllbarkeit.

Einführung von Abkürzungen

- Ist die Formel nicht atomar, führe eine Abkürzung (= neue Variable) für die Formel ein und hänge die Definition der Abkürzung an eine Liste an. Die entstandene Hauptformel ist jetzt in KNF (ohne die Liste).

- Ist die Liste leer, hat das Verfahren terminiert, falls nicht, betrachte die erste Definition der Liste. Treten innerhalb der Definition komplexe Teilformeln auf, ersetze jede dieser (maximal 2) auftretendenen komplexen Teilformeln durch neue Variablen und füge die entsprechenden Definitionen an die Liste an.

- Bilde die KNF-Form der ersten Definition der Liste

- Konjugiere diese KNF-Formel mit der Hauptformel zur neuen Hauptformel und lösche die Definition aus der Liste.

- Fahre mit der Bearbeitung der Liste fort.

Komplexität der NF-Trafo (Beispiel)

Gegeben sei die nichtatomare Formel $F := (G \leftrightarrow (H \leftrightarrow I))$
Wir definieren eine neue Variable R_1 und fügen die Definition

$$R_1 \leftrightarrow \boxed{(G \leftrightarrow (H \leftrightarrow I))}$$

in eine Liste ein. Die Hauptformel ist jetzt R_1. Wir nehmen die erste Definition der Liste und ersetzen nichtatomare Teilformeln (auf der rechten Seite, d.h. in der Box) durch neue Definitionen. Die erste Definition der Liste lautet jetzt

$$R_1 \leftrightarrow \boxed{(G \leftrightarrow R_2)},$$

die zweite Definition lautet

$$R_2 \leftrightarrow \boxed{(H \leftrightarrow I)}.$$

Entsprechend der Liste der KNF Umformungen ersetzen wir die (erste) Definition durch ihre KNF Form:

$$((\neg R_1 \vee \neg G \vee R_2) \wedge (\neg R_1 \vee \neg R_2 \vee G) \wedge \\ (G \vee R_2 \vee R_1) \wedge (\neg R_2 \vee \neg G \vee R_1)),$$

konjugieren diese Formel mit der Hauptformel und löschen die Definition aus der Liste.

Im nächsten Element der Liste kommen keine komplexen Teilformeln mehr vor. Wir erzeugen die KNF Form dieser Definition:

$$((\neg R_2 \vee \neg H \vee I) \wedge (\neg R_2 \vee \neg I \vee H) \wedge \\ (H \vee I \vee R_2) \wedge (\neg I \vee \neg H \vee R_2))$$

Wir konjugieren diese Formel mit der Hauptformel und löschen sie aus der Liste.

Das Ergebnis unserer Umformung (die Hauptformel) lautet:

$$
\begin{array}{ll}
(R_1 \wedge (\neg R_1 \vee \neg G \vee R_2) \wedge (\neg R_1 \vee \neg R_2 \vee G) & \wedge \\
(G \vee R_2 \vee R_1) \wedge (\neg R_2 \vee \neg G \vee R_1) & \wedge \\
(\neg R_2 \vee \neg H \vee I) \wedge (\neg R_2 \vee \neg I \vee H) & \wedge \\
(H \vee I \vee R_2) \wedge (\neg I \vee \neg H \vee R_2)) &
\end{array}
$$

Die Liste ist leer, das Verfahren hat terminiert.

Liste der KNF Umformungen

Bei der Einführung von Abkürzungen nehmen wir auf folgende Umformungen Bezug:

$R_i \leftrightarrow (F_1 \leftrightarrow F_2)$	$((\neg R_i \vee \neg F_1 \vee F_2) \wedge$ $(\neg R_i \vee F_1 \vee \neg F_2) \wedge$ $(R_i \vee F_1 \vee F_2) \wedge$ $(R_i \vee \neg F_1 \vee \neg F_2))$
$R_i \leftrightarrow (F_1 \rightarrow F_2)$	$((\neg R_i \vee \neg F_1 \vee F_2) \wedge$ $(R_i \vee F_1) \wedge (R_i \vee \neg F_2))$
$R_i \leftrightarrow (F_1 \wedge F_2)$	$((R_i \vee \neg F_1 \vee \neg F_2) \wedge$ $(\neg R_i \vee F_1) \wedge (\neg R_i \vee F_2))$
$R_i \leftrightarrow (F_1 \vee F_2)$	$((\neg R_i \vee F_1 \vee F_2) \wedge$ $(R_i \vee \neg F_1) \wedge (R_i \vee \neg F_2))$
$R_i \leftrightarrow (\neg F_1)$	$((\neg R_i \vee \neg F_1) \wedge (R_i \vee F_1))$

NF Komplexität (Abschätzung)

- Eine komplexe Formel wird durch ein Zeichen ersetzt
- Jedes Bikonditional wird durch eine KNF Formel aus 12 Zeichen ersetzt
- Jedes Konditional, Disjunktion oder Konjunktion wird durch eine KNF Formel aus 7 Zeichen ersetzt
- Jede Negation wird durch eine KNF Formel aus 4 Zeichen ersetzt

Als Beispiel betrachten wir das Formelschema

$(F_1 \leftrightarrow (F_2 \leftrightarrow (\ldots F_n)))$

n	Zeichen ohne Abk. $2^{(n-1)} \cdot n$	Zeichen mit Abk. $1 + (n-1) \cdot 12$
2	$2^1 \cdot 2 = 4$	$1 + 12 = 13$
3	$2^2 \cdot 3 = 12$	$1 + 24 = 25$
4	$2^3 \cdot 4 = 32$	$1 + 36 = 37$
5	$2^4 \cdot 5 = 80$	$1 + 48 = 49$
$\ldots$	$\ldots$	$\ldots$
10	$2^9 \cdot 10 = 5120$	$1 + 108 = 109$

Kleinste Normalform

Notiert man in einer Interpretation jede Variable als Binärzahl (z.B. '0' falls $\iota(p) = f$ und '1' falls $\iota(p) = w$) und legt man auf den Variablen eine Ordnung fest, so kann man jede Interpretation als Binärzahl formulieren.

Jetzt lässt sich die Menge aller Interpretationen partionieren: In einer Menge liegen alle Interpretionen mit einer geraden Anzahl von '1', in der anderen Menge mit ungerader Anzahl von '1'. Beide Mengen haben den Umfang von 2^{n-1} Elementen.

Da sich in beiden Mengen jedes Element mindestens in zwei Werten der Variablen von jedem anderen Element in der Menge unterscheidet, können in beiden Mengen keine Elemente zusammengefasst werden (wie z.B. $(p \wedge q) \vee (p \wedge \neg q)$ zu p zusammengefasst werden können). Die disjunktive Normalform muss daher 2^{n-1} Elemente enthalten (analog konjunktive Normalform).

Frege/Hilbert-Kalküle

Logikkalkül = Nichtdeterministischer Algorithmus für Gültigkeit/Unerfüllbarkeit

Axiomatische Kalküle: Axiome und Inferenzregeln

Beispiel: Kalkül von Lukasiewicz für $\{\neg, \rightarrow\}$:

Axiomenschemata:

$$(A_1) \qquad (\Phi \rightarrow \Psi) \rightarrow ((\Psi \rightarrow \mathrm{X}) \rightarrow (\Phi \rightarrow \mathrm{X}))$$

$$(A_2) \qquad (\neg\Phi \rightarrow \Phi) \rightarrow \Phi$$

$$(A_3) \qquad \Phi \rightarrow (\neg\Phi \rightarrow \Psi)$$

Inferenzregel (Modus Ponens):

$$\frac{\Phi \qquad \Phi \rightarrow \Psi}{\Psi}$$

Beispielherleitung im Hilbert-Kalkül

Herleitung der Formel $p \to p$ im Kalkül von Lukasiewicz:

(1)	$(p \to (\neg p \to p)) \to (((\neg p \to p) \to p) \to (p \to p))$	(A_1)
(2)	$p \to (\neg p \to p)$	(A_3)
(3)	$((\neg p \to p) \to p) \to (p \to p)$	(MP 2,1)
(4)	$(\neg p \to p) \to p$	(A_2)
(5)	$p \to p$	(MP 4,3)

Sequenzenkalkül (Gentzen)

„Natürliche“ Kalküle (Gentzen, 1934):

- Natürliche Deduktion
- Sequenzenkalkül

Natürlichkeit:

- Keine künstlichen Axiome
- Natürlichkeit der Regeln

Sequenz:

$$\underbrace{F_1, \ldots, F_m}_{\text{Antezedens}} \vdash \underbrace{G_1, \ldots, G_n}_{\text{Sukzedens}}$$

$\hat{=}$ unter den Annahmen $F_1, \ldots, F_m$ gilt eine der Formeln G_i

$\hat{=}$

$$(F_1 \wedge \cdots \wedge F_m) \rightarrow (G_1 \vee \cdots \vee G_n)$$

Sequenzenkalkül (2)

Regelname	Antezedens	Sukzedens
[Axiom]	$\dfrac{}{D \vdash D}$	
[¬-Einführung]	$\dfrac{\Gamma \vdash \Theta, A}{\neg A, \Gamma \vdash \Theta}$	$\dfrac{A, \Gamma \vdash \Theta}{\Gamma \vdash \Theta, \neg A}$
[∨-Einführung]	$\dfrac{A, \Gamma \vdash \Theta \qquad B, \Gamma \vdash \Theta}{A \vee B, \Gamma \vdash \Theta}$	$\dfrac{\Gamma \vdash \Theta, A}{\Gamma \vdash \Theta, A \vee B} \quad \dfrac{\Gamma \vdash \Theta, B}{\Gamma \vdash \Theta, A \vee B}$
[∧-Einführung]	$\dfrac{A, \Gamma \vdash \Theta}{A \wedge B, \Gamma \vdash \Theta} \quad \dfrac{B, \Gamma \vdash \Theta}{A \wedge B, \Gamma \vdash \Theta}$	$\dfrac{\Gamma \vdash \Theta, A \qquad \Gamma \vdash \Theta, B}{\Gamma \vdash \Theta, A \wedge B}$
[→-Einführung]	$\dfrac{\Gamma \vdash \Theta, A \qquad B, \Delta \vdash \Lambda}{A \to B, \Gamma, \Delta \vdash \Theta, \Lambda}$	$\dfrac{A, \Gamma \vdash \Theta, B}{\Gamma \vdash \Theta, A \to B}$

Strukturelle Regeln des Sequenzenkalküls

Regelname	Antezedenz	Sukzedens
[Verdünnung]	$\dfrac{\Gamma \vdash \Theta}{D, \Gamma \vdash \Theta}$	$\dfrac{\Gamma \vdash \Theta}{\Gamma \vdash \Theta, D}$
[Zusammenziehung]	$\dfrac{D, D, \Gamma \vdash \Theta}{D, \Gamma \vdash \Theta}$	$\dfrac{\Gamma \vdash \Theta, D, D}{\Gamma \vdash \Theta, D}$
[Vertauschung]	$\dfrac{\Gamma, D, E, \Delta \vdash \Theta}{\Gamma, E, D, \Delta \vdash \Theta}$	$\dfrac{\Gamma \vdash \Theta, E, D, \Lambda}{\Gamma \vdash \Theta, D, E, \Lambda}$

[Schnitt] $$\frac{\Gamma \vdash \Theta, D \qquad D, \Delta \vdash \Lambda}{\Gamma, \Delta \vdash \Theta, \Lambda}$$

Beispiele für Sequenzenbeweise

$$\dfrac{\dfrac{\dfrac{A \vdash A}{\vdash A, \neg A}}{\neg\neg A \vdash A}}{\vdash \neg\neg A \to A}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{A \vdash A}{\vdash A, \neg A}}{\vdash \neg A, A} \qquad \dfrac{}{B \vdash B}}{A \to B \vdash \neg A, B}}{A \to B \vdash \neg A, \neg A \vee B}}{A \to B \vdash \neg A \vee B, \neg A}}{A \to B \vdash \neg A \vee B, \neg A \vee B}}{A \to B \vdash \neg A \vee B}}{\vdash (A \to B) \to (\neg A \vee B)}$$

Der Tableaukalkül

- Der Tableaukalkül arbeitet auf Beweisbäumen
- Er erzeugt Widerspruchsbeweise:
  Um die Gültigkeit einer Formel F zu beweisen, wird die Unerfüllbarkeit von $\neg F$ gezeigt
- Verschiedene Varianten: Nicht-Normalform- und Normalform-Tableau
- Der Tableaukalkül ist analytisch, er geht von dem zu Beweisenden aus und dekomponiert

  (Gegensatz: Synthetisch, erzeugt das zu Beweisende)

Die Dekompositionsregeln des Tableaukalküls

Einteilung der Formeln in einen

- konjunktiven Typ α und einen
- disjunktiven Typ β

Konjunktiv: α		Disjunktiv: β	
α	α_1, α_2	β	β_1, β_2
$F \wedge G$	F, G	$F \vee G$	F, G
$\neg(F \vee G)$	$\neg F, \neg G$	$\neg(F \wedge G)$	$\neg F, \neg G$
$\neg(F \rightarrow G)$	$F, \neg G$	$F \rightarrow G$	$\neg F, G$
$\neg\neg F$	F		

Dekompositionsregeln:

$$\frac{\alpha}{\alpha_i} \qquad \frac{\beta}{\beta_1 \mid \beta_2}$$

Tableaukalkül (formal)

Gegeben: Eine Formel F.

Tableau:

- Ein Baum mit einem Ast, auf dessen Knoten sich nur F befindet, ist ein Tableau für F.
- Ein Baum, der durch Anwendung einer der Dekompositionsregeln auf ein Tableau für F entsteht, ist ein Tableau für F.

Abschluß:

- Ein Tableauast ist (atomar) geschlossen, wenn sich auf ihm eine (atomare) Formel F und $\neg F$ befinden.
- Ein Tableau ist (atomar) geschlossen, wenn alle seine Äste (atomar) geschlossen sind.

Ein geschlossenes Tableau für $\neg F$ ist ein Tableaubeweis für F.

Beispieltableau

(1) $\neg((p \to q) \to (\neg p \vee q))$

$\mid \alpha(1)$

(2) $p \to q$

$\mid \alpha(1)$

(3) $\neg(\neg p \vee q)$

$\mid \alpha(3)$

(4) $\neg\neg p$

$\mid \alpha(3)$

(5) $\neg q$

$\mid \beta(2)$

(6) $\neg p \quad \mid \quad q \quad (7)$

$\mid \alpha(4) \qquad *$

(8) p

$*$

Eigenschaften des Tableaukalküls

Vollständigkeit und Korrektheit:
$\neg F$ unerfüllbar (bzw. F allgemeingültig) gdw es ein (atomar) geschlossenes Tableau für $\neg F$ gibt.

$\alpha\beta$-Unterformeleigenschaft: Die Formel jedes Nichtwurzelknotens ist α- oder β-Unterformel einer Vorgängerformel auf dem Ast.

Satz: Die $\alpha\beta$-Unterformelrelation ist wohlfundiert, d.h., es gibt keine unendlichen Dekompositionsketten.

Striktheit:

- Jede Formel vom Typ β darf auf jedem Ast nur einmal angewendet werden.
- Jede Formel vom Typ α darf für jedes α_i auf jedem Ast nur einmal angewendet werden.

Eigenschaften des Tableaukalküls (2)

Regularität: Keine Formel darf mehr als einmal auf einem Ast vorkommen.

Aus Regularität folgt Striktheit.

Satz: Für jede aussagenlogische Formel gibt es nur endlich große und endlich viele reguläre Tableaux.

Beweiskonfluenz:
Ein Kalkül ist beweiskonfluent, wenn sich zu jedem Zeitpunkt ein Beweis durch weitere Anwendung der Kalkülregeln finden läßt.

Satz: Der (reguläre) Tableaukalkül ist beweiskonfluent.

Korrektheit des Tableaukalküls

Zu zeigen: Ist ein Tableau für F geschlossen, so ist F unerfüllbar.

Bzw.: Wenn eine aussagenlogische Formel F erfüllbar ist, dann hat jedes Tableau für F (mindestens) einen offenen Ast.

Hilfssatz: S sei beliebige erfüllbare Menge aussagenlogischer Formeln:

- Falls $\alpha \in S$, dann sind alle $S \cup \{\alpha_i\}$ erfüllbar.
- Falls $\beta \in S$, dann ist ein $S \cup \{\beta_i\}$ erfüllbar.

Korrektheitsbeweis:
Tableauast erfüllbar gdw Menge der Formeln auf Ast erfüllbar (offenbar ist jeder erfüllbare Ast offen).
Induktion (über Inferenzschritte): Ind.-Anfang: Ast erfüllbar nach Annahme.
Ind.-Schritt (n,$n+1$): Tableau T_n hat erfüllbaren Ast B (Ind.-Annahme). Entweder T_{n+1} enthält B, dann erfüllbar, oder T_{n+1} enthält Äste $B \cup \{F_i\}$. Nach Hilfssatz muß einer dieser erfüllbar sein. $q.e.d$

Systematisches Tableau

Systematische Tableau(folge) für Formel F: Benutzbarkeitsmarkierung an speziellen Knoten im Tableau. Wurzelknoten von $\mathcal{T}_0$ benutzbar, falls F kein Literal.

Wiederhole:

- Falls kein benutzbarer Knoten in T_n, stop.
- Ansonsten wähle linkesten obersten benutzbaren Knoten K_n in T_n markiert mit Formel F_n.
  - Expandiere alle offenen Äste durch K_n vollständig nach dem Typ von F_n (d.h. $\begin{matrix}\alpha_1\\ \alpha_2\end{matrix}$ bzw. $\beta_1 \mid \beta_2$).
  - Markiere alle neuen Knoten, die nicht mit Literalen markiert sind, als benutzbar.
  - Danach markiere K_n als unbenutzbar sowie alle Knoten auf (atomar) geschlossenen Ästen.

Hintikkamengen

Eine Menge aussagenlogischer Formeln S ist abwärts gesättigt falls gilt:

- Wenn Formel vom Typ $\alpha \in S$, dann für alle α_i: $\alpha_i \in S$.
- Wenn Formel vom Typ $\beta \in S$, dann gibt es ein β_i: $\beta_i \in S$.

(Atomare) Hintikka-Menge S:

- S abwärts gesättigt und
- S enthält keine (atomare) Formel und ihre Negation.

Hintikkas Hilfssatz: Jede (atomare) Hintikkamenge ist erfüllbar.

Tableau: Vollständigkeit

Systematisches Tableau für F:
letztes Tableau T^* aller Tableaux in der systematischen Tableaufolge für F (existiert wegen Wohlfundiertheit der $\alpha\beta$-Unterformelrelation).

Zu zeigen: Ist F unerfüllbar, so ist das systematische Tableau für F geschlossen.

Hilfssatz: Falls B (atomar) offener Ast eines systematischen aussagenlogischen Tableaux, dann ist die Menge der Formeln auf B eine (atomare) Hintikkamenge.

Beweis der Vollständigkeit des Tableaukalküls:
Sei F unerfüllbare aussagenlogische Formel und T^* systematisches aussagenlogisches Tableau für F.
Indirekt: Angenommen, B sei atomar offener Ast in T^*. Nach Hilfssatz würde atomare Hintikkamenge für Formeln auf B existieren und nach Hintikkas Hilfssatz ein Modell für F. *q.e.d.*

Klauseltableaux

In der Praxis erheblicher Effizienzgewinn, wenn Wurzelformel des Tableau in konjunktiver Normalform oder eine Menge von Klauseln

Wir sprechen dann von Klauseltableaux

Klauseltableau: Gegeben ein Klauselmenge S

- Wurzelknoten mit S markiert
- Ist T ein Klauseltableau für S, B ein Zweig in T, $c = \{l_1 \vee \cdots \vee l_n\}$ eine Klausel aus S, dann erhält man wieder ein Klauseltableau für S, wenn man an B n neue Zweige, die jeweils $l_1, \ldots, l_n$ enthalten, anhängt.

Der Abschluß ist definiert wie bei allgemeinen Tableaus.

Vorsicht: Man geht davon aus, dass die Klauselmenge bereits die negierte Eingabeformel darstellt! Es wird also keine Negation mehr angewendet

Konnektionstableaux

Konnektion in einer Klauselmenge S: 2-elementige Menge komplementärer Literalvorkommnisse

Idee: Benutzung von Konnektionen zur Kontrolle des Tableauaufbaus

Knotenfamilie: alle Knoten mit demselben unmittelbaren Vorgänger; die Disjunktion ihrer Literale heisst Tableauklausel

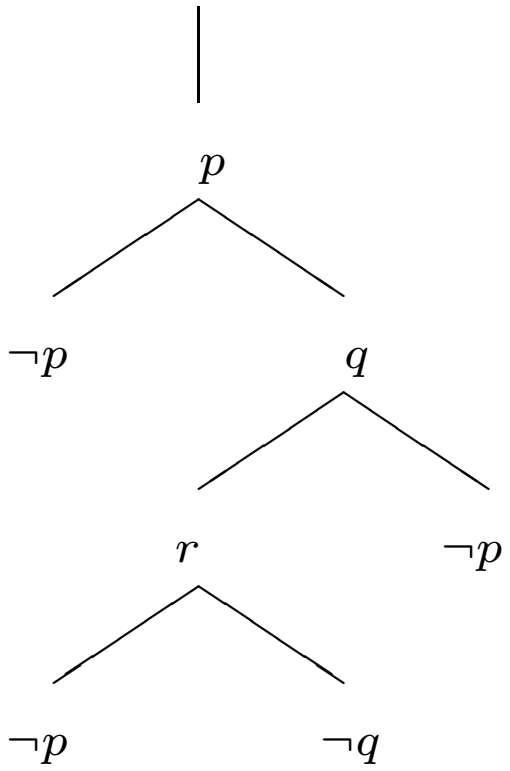
Startklausel: Tableauklausel der obersten Knotenfamilie

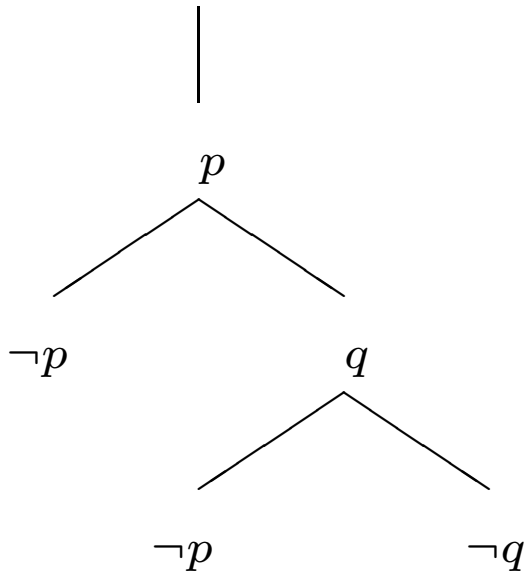
Klauseltableau schwach konnektiert: In jeder Knotenfamilie außer der obersten gibt es einen Knoten mit einem komplementären Vorgänger.

Klauseltableau konnektiert: In jeder Knotenfamilie außer der obersten gibt es einen Knoten mit einem komplementären unmittelbaren Vorgänger.

Konnektionstableaux (2)

Beispiel für eine schwach konnektiertes und ein konnektiertes Klauseltableau (sog. Konnektionstableau):





Essentialität und Relevanz

Gegeben sei eine Klauselmenge S und eine Klausel $c \in S$:

- c essentiell in S, falls S unerfüllbar und $S \setminus \{c\}$ erfüllbar.
- S minimal unerfüllbar, falls S unerfüllbar und jede echte Teilmenge von S erfüllbar.
- c relevant, falls c essentiell in einer Teilmenge von S.

Beispiel: $S = \{p, q, \neg p \vee q, r \vee \neg p, \neg p \vee \neg q\}$

	p	q	$\neg p \vee q$	$r \vee \neg p$	$\neg p \vee \neg q$
Relevant in S	j	j	j	n	j
Essentiell in S	j	n	n	n	j

Das Konnektionslemma

Verstärkung einer Klauselmenge: Gegeben

- eine Menge S von Klauseln und eine Literalmenge $P = \{L_1, \ldots, L_n\}$,
- eine Teilmenge S' von S, in der keine Literale aus P (als Disjunktionsglieder) vorkommen.

Die Klauselmenge $S' \cup P$ heißt Verstärkung von S durch P, geschrieben: $P \rhd S$.

Konnektions- oder Partnerlemma:
Gegeben Klausel c, die relevant in S und L beliebiges Literal (als Disjunktionsglied) in c. Dann gibt es eine Klausel $c' \in S$ derart, das gilt:

1. c' enthält das zu L komplementäre Literal $\sim L$.
2. c' ist relevant in $\{L\} \rhd S$.

Beweis des Konnektionslemmas

Sei S' bel. minimal unerfüllbare Teilmenge von S mit $c \in S'$.

Es gibt eine Interpretation $\iota : \iota(c) = \mathbf{f}$ und $\iota(S' \setminus \{c\}) = \mathbf{w}$.

Definiere ι' so, daß $\iota'(L) = \mathbf{w}$ und sonst $\iota' = \iota$.

Da $\iota'(c) = \mathbf{w}$, gibt es eine Klausel $c' \in S' : \iota'(c') = \mathbf{f}$.

Wir zeigen: c' erfüllt die Bedingungen 1 und 2.

1. c' enthält $\sim L$, da sonst $\iota(c') = \mathbf{f}$.

2. Da c' essentiell in S' ist, gibt es eine Interpretation ι'' mit $\iota''(c') = \mathbf{f}$ und $\iota''(S' \setminus \{c'\}) = \mathbf{w}$.

   Da $\iota''(L) = \mathbf{w}$, ist c' essentiell in $S' \cup \{L\}$ und damit auch in ihrer Teilmenge $\{L\} \rhd S'$.

   Da $\{L\} \rhd S' \subseteq \{L\} \rhd S$, ist c' relevant in $\{L\} \rhd S$. *q.e.d.*

Vollständigkeit des Konnektionstableaukalkülks

Satz: Sei S eine endliche unerfüllbare Klauselmenge und c eine beliebige relevante Klausel in S. Dann gibt es ein geschlossenes reguläres Konnektionstableau für S mit Startklausel c.

Indeterministische Prozedur zur Widerlegungskonstruktion:

Wähle c als Startklausel. Solange Tableau nicht geschlossen, wiederhole:

1. Wähle offenen Ast B mit Literalmenge $P = \{L_1, \ldots, L_m, L\}$.

2. Wähle Klausel $c' \in S$, die relevant in $P \rhd S$ und $\sim L$ enthält.

3. Erweitere den Ast B mit der Klausel c'.

Vollständigkeit des Konnektionstableaukalkülks (2)

Beweis:

Zunächst ist klar, daß die Prozedur lediglich reguläre Konnektionstableaux erzeugen kann.

Dann zeigen wir, daß für jeden offenen Ast eine entsprechende Klausel c' zur Extension existiert, mittel Induktion über die Länge des Astes.

Ind.-Anfang: folgt aus Konnektionslemma.

Ind.-Schritt $(n,n+1)$: Sei B offener Ast mit Tableauklausel c und Literalmenge $P = \{L_1, \ldots, L_n, L\}$. Nach Ind.-Annahme ist c relevant in $\{L_1, \ldots, L_n\} \rhd S$. Sei S' bel. minimal unerfüllbare Teilmenge von $\{L_1, \ldots, L_n\} \rhd S$ mit $c \in S'$. Aufgrund des Konnektionslemmas gibt es eine Klausel $c' \in S'$, die $\sim L$ enthält und relevant in $\{L\} \rhd S'$ ist.

Schließlich muß die Prozedur auf jeden Fall terminieren wegen der Regularität und weil S endlich ist. *q.e.d.*

Konnektiertheit und Beweiskonfluenz

Satz: Der schwach konnektierte Klauseltableaukalkül und der Konnektionstableaukalkül sind nicht beweiskonfluent.

Beweis: Man betrachte die unerfüllbare Klauselmenge $S = \{p, q, \neg q\}$ und das Tableau mit Startklausel p. *q.e.d.*

Konsequenzen:

- Es gibt kein zum systematischen Verfahren analoges Tableau„sättigungs“verfahren.
- Stattdessen müssen alle schwach konnektierten Klauseltableaux bzw. Konnektionstableaux aufgezählt werden, d.h. bei der Suche ist Rücksetzen erforderlich.
- Für erfüllbare Eingabemengen werden keine Modelle berechnet.

Resolution

Resolution basiert auf der Mengendarstellung von Klauselformeln

- Kalkül zur Herleitung von Klauseln (Resolventen) aus Paaren von Klauseln (Elternklauseln)
- Seien c_1, c_2 Literalmengen und a eine aussagenlogische Variable, die sog. Resolventenvariable.

Resolutionsregel:
$$\frac{c_1 \cup \{a\} \qquad \{\neg a\} \cup c_2}{(c_1 \setminus \{a\}) \cup (c_2 \setminus \{\neg a\})}$$

Beispiel:
$$\frac{a \vee b \vee c \qquad a \vee \neg b \vee \neg d}{a \vee c \vee \neg d} \qquad \frac{\{a, b, c\} \qquad \{a, \neg b, \neg d\}}{\{a, c, \neg d\}}$$

Man beachte das durch die Mengendarstellung bedingte „Merging“.

Korrektheit der Resolution: Jede Resolvente wird von der Menge ihrer Elternklauseln impliziert.

Resolution (2)

Resolutionsherleitung einer Klausel c_n aus einer Menge von Klauseln S:
Folge $D = c_1, \ldots, c_n$ von Klauseln derart, dass für jede Klausel c_k in D gilt:

- $c_k \in S$ oder
- es gibt Klauseln c_i, c_j in D mit $i, j < k$ und c_k ist Resolvente von c_i, c_j.

Resolutionswiderlegung von S:
Resolutionsherleitung der leeren Klausel $\emptyset$ (oft auch geschrieben $\square$) aus S

Beispiel: $S = \{\{\mathsf{r},\mathsf{n}\}, \{\neg\mathsf{r},\mathsf{n}\}, \{\neg\mathsf{n}\}\}$.

Eine Resolutionswiderlegung von S: $\{\mathsf{r},\mathsf{n}\}, \{\neg\mathsf{r},\mathsf{n}\}, \{\neg\mathsf{n}\}, \{\mathsf{n}\}, \emptyset$

(Widerlegungs-)Vollständigkeit der Resolution: Zu jeder unerfüllbaren Klauselmenge S gibt es eine Resolutionswiderlegung.

Man beachte, dass nicht jede Klausel c, die von einer Klauselmenge impliziert wird, auch mittels Resolution hergeleitet werden kann.

Resolution (3)

Resolution als Entscheidungsverfahren

Gegeben eine aussagenlogische Formel F und die Frage, ob F erfüllbar ist:

- Transformiere F in eine (erfüllbarkeits-)äquivalente Klauselmenge S.
- Bilde die Menge S^* aller mittels Resolution aus S herzuleitenden Klauseln, die sog. Sättigungsmenge von S.
- $\emptyset \notin S^*$ gdw S erfüllbar ist.
- Beachte: Im Erfüllbarkeitsfall liefert das Verfahren aber kein Modell.

Effizienz des Verfahrens: S^* ist im worst-case exponentiell in S.

Vollständigkeitserhaltende Optimierungen

- Subsumptionslöschung: Lösche jede subsumierte Klausel, d.h. jede Klausel, die eine echte Obermenge einer hergeleiteten Klausel ist.
- Tautologielöschung: Lösche jede tautologische Klausel, d.h. jede Klausel, die eine aussagenlogische Variable und ihre Negation enthält.

Semantische Bäume

- Optimierung der Wahrheitstafelmethode
- Schrittweise Partitionierung der Belegungsmenge (Äste)
- Typischerweise für Klauselformeln

Semantischer Baum für Klauselmenge S:

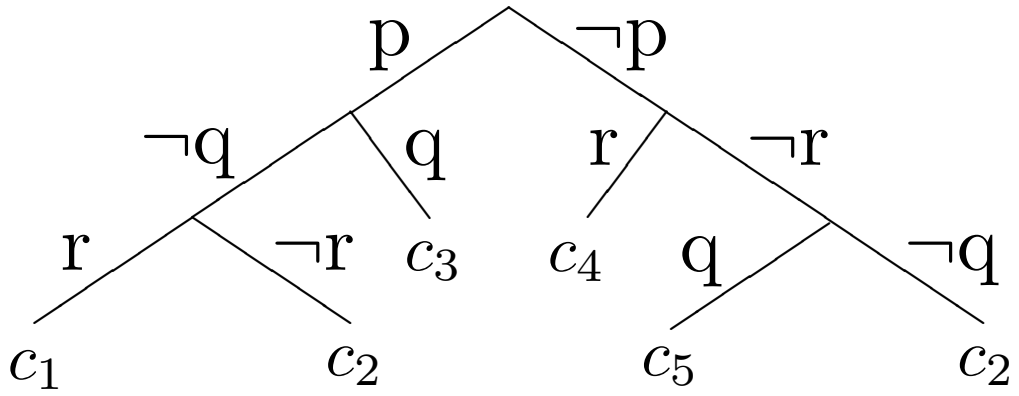
Binärbaum, dessen Kanten mit Literalen und dessen Blattknoten (partiell) mit Klauseln aus S markiert sind wie folgt:

- Jedes Paar von Kanten, die aus einem gemeinsamen Knoten entspringen, ist mit einer Variablen p, die in S vorkommt, bzw. mit $\neg p$ markiert, wobei p nicht bereits auf dem Ast vorkommt.
- jeder Blattknoten kann mit einer Klausel $c \in S$ markiert sein, falls die Komplemente aller Literale in c auf dem Ast vorkommen.

Eine Semantischer Baum für S heisst geschlossen, falls jeder Blattknoten mit einer Klausel (aus S) markiert ist.

Semantische Bäume (2)

Beispiel: Geschlossener semantischer Baum für Klauselmenge $\{c_1, c_2, c_3, c_4, c_5\}$



c_1: q $\vee$ $\neg$r
c_2: q $\vee$ r
c_3: $\neg$p $\vee$ $\neg$q
c_4: p $\vee$ $\neg$ r
c_5: p $\vee$ $\neg$q $\vee$ r

Satz: Eine Klauselmenge ist unerfüllbar gdw es einen geschlossenen semantischen Baum für sie gibt.

Beweis: Einfach über die Partitionierung der Belegungsmenge und die Semantik von Klauselmengen

Semantische Bäume (3)

Gesättigter semantischer Baum für S:

kein Erweiterungsschritt mehr möglich, d.h. jeder Ast ist entweder durch eine Klausel aus S geschlossen oder er hat maximale Länge und kann nicht durch eine Klausel aus S geschlossen werden

Semantische Bäume und Modellfindung:

Wenn eine Formelmenge S erfüllbar ist, dann gilt für jeden gesättigten semantische Baum T für S folgendes: Die Literalmengen M auf den offenen Ästen von T entsprechen genau den Modellen von S; genauer: für alle M: $l \in M$ gdw $\iota(l) = \mathbf{w}$.

Bezug zur Wahrheitstafelmethode:

- Äste entsprechen Modellen bzw. Modellmengen
- Vorteil gegenüber Wahrheitstafelmethode: Geschlossene Äste müssen nicht maximale Länge haben

Semantische Bäume (4)

Durch Astliterale M gegebene partielle Belegung Teilbelegung (partielle Belegung) ι wie folgt definiert:

$$\iota(p) = \begin{cases} \mathbf{w} & \text{falls } p \in M \\ \mathbf{f} & \text{falls } \neg p \in M \\ \text{undefiniert} & \text{sonst} \end{cases}$$

Partielle Evaluierung von S und Anwendung der Absorptionsregeln für $\bot$ und $\top$

Gegeben durch Astliterale M definierte Teilbelegung $\iota : V \to \{\mathbf{w}, \mathbf{f}\}$:

- Ersetzung aller Variablen $p \in V$ durch $\top$ bzw. $\bot$, wenn $\iota(p) = \mathbf{w}$ bzw. $\mathbf{f}$
- Löschung aller Klauseln, die $\top$ oder $\neg\bot$ enthalten
- Löschung aller Literale $\bot$ und $\neg\top$ aus den Klauseln
- Resultierende Klauselmenge bezeichnen wir mit $M(S)$
- Ast M kann abgeschlossen werden, wenn $M(S)$ die leere Klausel enthält

Implementierung Semantischer Bäume

Möglichkeit der effizienten Implementation

- Baumabarbeitung mittels Tiefenverfahren und Backtracking
- Deaktivierung/Reaktivierung von wahren Klauseln, d.h. $M(c) = \top$
- Dekrementierung/Inkrementierung von Zählern $\ell(c)$ an Klauseln c mit $M(c) \neq \top$: $\ell(c) = |M(c)|$

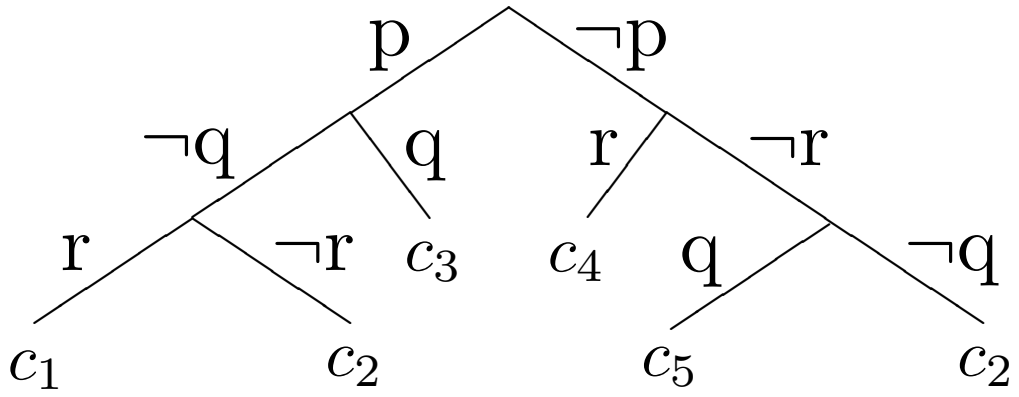
Effizienz des Verfahrens (im worst case):

- Linearer Platzbedarf
- Jeder Expansionschritt/Kontraktionsschritt hat linearen Zeitbedarf
- Anzahl der Schritte: exponentiell

Man beachte:

- Variablenfolge auf Ast bestimmt die Grösse des semantischen Baums
- Bei verschiedenen geschlossenen semantischen Bäumen für dieselbe Klauselmenge u.U. exponentielle Grössenunterschiede

Abarbeitung eines Semantischen Baums



c_1: q $\vee$ $\neg$r
c_2: q $\vee$ r
c_3: $\neg$p $\vee$ $\neg$q
c_4: p $\vee$ $\neg$ r
c_5: p $\vee$ $\neg$q $\vee$ r

Astliterale	$\{q,\neg r\}$	$\{q,r\}$	$\{\neg p,\neg q\}$	$\{p,\neg r\}$	$\{p,\neg q,r\}$	Aktion
	2	2	2	2	3	Start
p^l	2	2	1	$\top$	$\top$	$\Downarrow l$
$p^l,\neg q^l$	1	1	$\top$	$\top$	$\top$	$\Downarrow l$
$p^l,\neg q^l,r^l$	0	$\top$	$\top$	$\top$	$\top$	$\Downarrow l$
$p^l,\neg q^l$	1	1	$\top$	$\top$	$\top$	$\Uparrow l$
$p^l,\neg q^l,\neg r^r$	$\top$	0	$\top$	$\top$	$\top$	$\Downarrow r$
$p^l,\neg q^l$	1	1	$\top$	$\top$	$\top$	$\Uparrow r$
p^l	2	2	1	$\top$	$\top$	$\Uparrow l$
p^l,q^r	$\top$	$\top$	0	$\top$	$\top$	$\Downarrow r$
p^l	2	2	1	$\top$	$\top$	$\Uparrow r$
$\neg p^r$	2	2	$\top$	1	2	$\Downarrow r$
$\neg p^r,r^l$	1	$\top$	$\top$	0	$\top$	$\Downarrow l$
$\neg p^r$	2	2	$\top$	1	2	$\Uparrow l$
$\neg p^r,\neg r^r$	$\top$	1	$\top$	$\top$	1	$\Downarrow r$
$\neg p^r,\neg r^r,q^l$	$\top$	$\top$	$\top$	$\top$	0	$\Downarrow l$
$\neg p^r,\neg r^r$	$\top$	1	$\top$	$\top$	1	$\Uparrow l$
$\neg p^r,\neg r^r,\neg q^r$	$\top$	0	$\top$	$\top$	$\top$	$\Downarrow r$
	2	2	2	2	3	$\Uparrow r\ \Uparrow r\ \Uparrow r$

Steuerung Semantischer Baum-Verfahren

Lineare Strategien:

- Unit propagation: Verzweige nach Atom p, falls eine Unitklausel p oder ¬p existiert, und schliesse den ¬p- bzw. den p-Ast.
- Purity propagation: Verzweige nach Atom p, das nur in einer Polarität, p bzw. ¬p, vorkommt, und schliesse den ¬p- bzw. den p-Ast.

Zwei Formen des (don't care) Indeterminismus: Heuristisch gesteuert

- Variablenauswahl: welche Variable als nächstes?
- Zeichenauswahl: in welchen Ast zuerst, p oder ¬p?
- Nach Häufigkeit des Vorkommens
- In kurzen Klauseln
- Nach verschiedenen (z.t. magischen) Kriterien

Implementation Semantischer Baum-Verfahren

Datenstrukturen

1. Atomliste: Liste von Atomen (Variablen)

2. Atom (Variable):

Nil / + / −	Pos-Klauseln(Atom)	Neg-Klauseln(Atom)

3. Klausel:

Nil / Atom	Aktuelle Länge	Pos-Atome(Klausel)	Neg-Atome(Klausel)

4. Literal := Atom | ¬ Atom

5. Interpretation: Feld von Atomen

Implementation Semantischer Baum-Verfahren (2)

```
Prozedur: verzweige:
if      unevaluiertes-atom-in(Atomliste):
        Atom := wähle-Atom(Atomliste); Literal := wähle-Literal(Atom);
        widerlege(Literal); widerlege(∼Literal)
else    print(Erfüllbar); break

Prozedur: widerlege(Literal):
Atom := atom(Literal);
if      Literal = Atom:
        Klauseln := Neg-Klauseln(Atom); True-Klauseln := Pos-Klauseln(Atom)
else    Klauseln := Pos-Klauseln(Atom); True-Klauseln := Neg-Klauseln(Atom);
dekrementiere-zähler(Klauseln);
if      not klausel-der-Länge-0-in(Klauseln):
        markiere-als-evaluiert(Atom,Literal); deaktiviere(True-Klauseln,Atom);
        verzweige;
        markiere-als-unevaluiert(Atom); reaktiviere(True-Klauseln,Atom);
inkrementiere-zähler(Klauseln)
```

Implementation Semantischer Baum-Verfahren (3)

Weitere Datenstrukturen zur Effizienzsteigerung:

- Liste der Klauseln mit aktueller Länge 1
- Aktuelle Häufigkeit der Literale (Problem: effiziente dynamische Aktualisierung)

Neuer Ansatz zur deutlichen Effizienzsteigerung (System CHAFF)

- Statt Klauselllängen-Dekrementierung Arbeiten mit 2 Wächterliteralen/Klausel
- Reduktion der Listen Pos-Klauseln(Atom), Neg-Klauseln(Atom) auf Klauseln, in denen $\sim$Atom Wächterliteral (Reduktion von $O(\Sigma_{c \in S}|c|)$ auf $O(|S|)$)
- Neues Vorgehen bei partieller Evaluierung bzgl. eines Literals l:
  - Betrachte nur Klauseln, in denen $\sim l$ Wächterliteral
  - Wenn c nicht wahr und ein Wächterliteral falsch, bestimme anderes unevaluiertes Literal als Wächter, falls möglich
  - Andernfalls hat c eine aktuelle Länge ≤ 1
- Nachteile: Purity nicht mehr feststellbar, Literalgewichte nicht aktuell (damit Einschränkung der Möglichkeiten heuristischer Suchraumreduktion)

Semantische Bäume und Tableaux

Resultate zu minimalen Beweislängen:

- Der Kalkül der semantischen Bäume kann den Klauseltableaukalkül linear simulieren.
- Der Klauseltableaukalkül kann den Kalkül der semantischen Bäume nicht polynomiell simulieren.

Schwäche des Tableaukalküls:

- Keine disjunkte Partitionierung der Interpretationsmengen, betrachte z.B. die Klauselmenge $S = \{a \vee b, a \vee \neg b, \neg a \vee b\}$
- Kann behoben werden durch: Faktorisierung, Umklappen (β-Regel mit Komplementen), Hochklappen, etc.

Semantische Bäume und Resolution

Resultate zu minimalen Beweislängen:

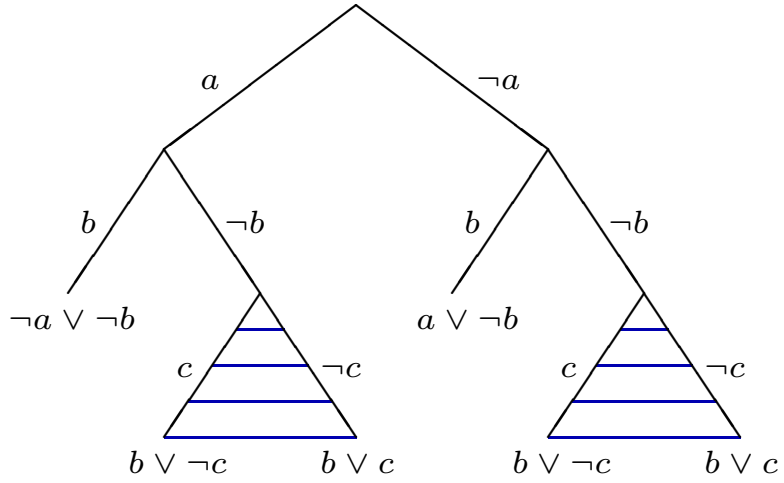
- Die Resolution kann den Kalkül der semantischen Bäume quadratisch simulieren.
- Der Kalkül der semantischen Bäume kann die Resolution nicht polynomiell simulieren.

Aber: Im average case ist der Kalkül der semantischen Bäume der Resolution überlegen

- Linearer Platzbedarf
- Effizientere Kalkülschritte
- Bessere Steuerung der Beweisfindung durch „lineare“ Strategien und heuristisch gesteuerte Variablen- und Zeichenauswahl
- Behebung der Beweislängenproblematik durch Lemmas (sog. Learning)

Das Duplikationsproblem in Semantischen Bäumen

Das Problem des Mehrfachvorkommens gleicher Unterbäume:

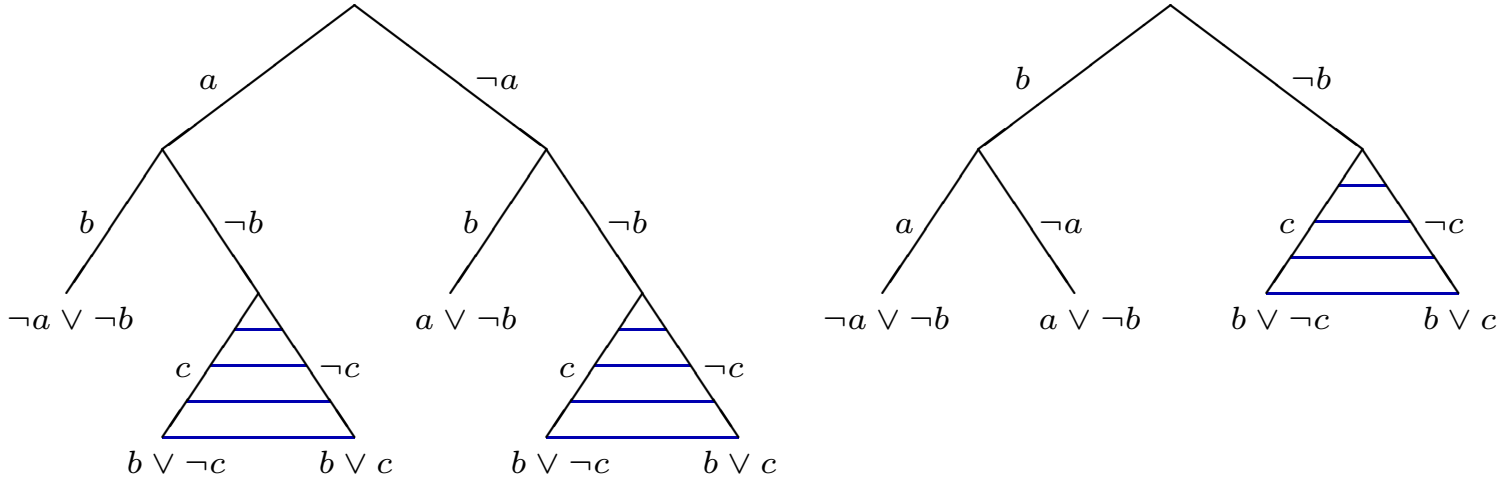


Lösungsansätze:

- Änderung der Variablenordnung
- Wiederverwendung von Unterbäumen (Lemmas)

Das Duplikationsproblem in Semantischen Bäumen (2)

Änderung der Variablenselektion:

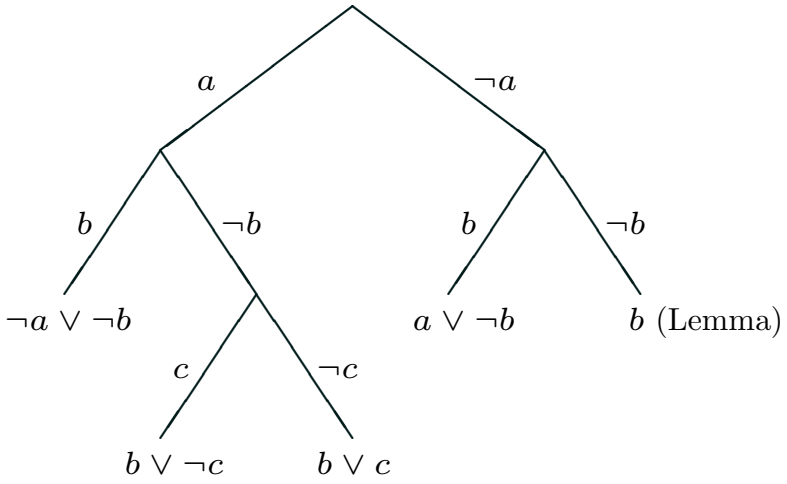


- Aber wie findet man die beste Variablenselektion?
  Dieses Problem ist NP-hart.
- Ausserdem gilt: Doppelvorkommnisse lassen sich i.A. nicht vermeiden, denn minimale semantische Bäume entsprechen Resolutionsbäumen

Semantische Bäume mit Lemmas

Erweiterung des Kalküls der semantischen Bäume:

- Versuch der Simulation der vollen Resolution
- Durch Extraktion von Resolventen (Lemmas) aus gelösten Unterbäumen



$\neg a \vee \neg b$

$b \vee \neg c$

$b \vee c$

$a \vee \neg b$

b (Lemma)

$\neg a$ (Lemma)