Logik

WiSe 2018/19

Thomas Schneider

Teil 1: Aussagenlogik

Homepage der Vorlesung: http://tinyurl.com/ws1819-logik

Aussagenlogik: Überblick

- Aussagenlogik behandelt die logische Verknüpfung von Aussagen mittels Junktoren wie und, oder, nicht, gdw.
- Jeder Aussage ist ein Wahrheitswert (wahr/falsch) zugeordnet.
- Man interessiert sich insbesondere für den Wahrheitswert zusammengesetzter Aussagen, z. B.:

  „*A* oder *B*“ wahr gdw. *A* wahr oder *B* wahr

  A oder *B* könnten z. B. stehen für „Die Erde ist ein Planet“ oder „Bremen liegt am Ganges“. Davon wird abstrahiert.
- Die Ausdrucksstärke von Aussagenlogik ist sehr begrenzt.
- Es ergeben sich jedoch interessante algorithmische Probleme (z. B. das Erfüllbarkeitsproblem).

Übersicht Teil 1

1.1 Grundlagen

1.2 Normalformen und funktionale Vollständigkeit

1.3 Erfüllbarkeit, Gültigkeit, Folgerbarkeit, Horn-Formeln

1.4 Resolution

1.5 Kompaktheit

Syntax

Wir fixieren eine abzählbar unendliche Menge VAR $= \{x_1, x_2, x_3, \dots\}$ von *Aussagenvariablen*.

Intuitiv kann jedes x_i Wahrheitswert *wahr* oder *falsch* annehmen und repräsentiert eine Aussage wie „Bremen liegt am Ganges“.

Definition 1.1 (Aussagenlogik, Syntax)

Die Menge AL der *aussagenlogischen Formeln* ist induktiv definiert durch

- $0, 1 \in$ AL
- VAR $\subseteq$ AL
- Wenn $\varphi, \psi \in$ AL, dann auch $\neg\varphi$, $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$ in AL

Beispiele:

$\neg x_1, \quad \neg\neg x_3, \quad (x_1 \wedge \neg x_4), \quad ((x_1 \wedge x_3) \wedge 1), \quad (\neg(x_1 \vee x_2) \wedge \neg(\neg x_1 \vee \neg x_2))$

Sprechweisen und Konventionen

- $\neg\varphi$ sprechen wir „nicht φ“ (Negation),

  $(\varphi \vee \psi)$ sprechen wir „φ oder ψ“ (Disjunktion),

  $(\varphi \wedge \psi)$ sprechen wir „φ und ψ“ (Konjunktion)

- 1 steht für „wahr“, 0 für „falsch“, 0,1 sind die *Booleschen Konstanten*

- Die *atomaren* Formeln sind $\{0, 1\} \cup$ VAR

- Alle anderen Formeln sind *zusammengesetzt*

- Statt $x_1, x_2, \ldots$ verwenden wir manchmal auch andere Symbole für Variablen, insbesondere x, y, z

Sprechweisen und Konventionen

- Klammern werden weggelassen, wenn das Resultat eindeutig ist, wobei $\neg$ stärker bindet als $\wedge$ und $\vee$

  Also steht z. B. $\neg x \wedge y$ für $(\neg x \wedge y)$, nicht für $\neg(x \wedge y)$

  $x \wedge y \vee x' \wedge y'$ ist nicht eindeutig, darum nicht erlaubt

- Iterierte Konjunktionen und Disjunktionen sind implizit linksgeklammert

  Also z.B. $x \wedge y \wedge z$ für $((x \wedge y) \wedge z)$

Semantik

Definition 1.2 (Aussagenlogik, Semantik)

Eine *Belegung* ist eine Abbildung $V : \text{VAR} \to \{0, 1\}$. Sie definiert einen Wahrheitswert $V(\varphi)$ für jede Formel φ:

- $V(0) = 0$ und $V(1) = 1$
- $V(\neg\varphi) = 1 - V(\varphi)$
- $V(\varphi \wedge \psi) = \begin{cases} 1 & \text{falls } V(\varphi) = 1 \text{ und } V(\psi) = 1 \\ 0 & \text{sonst} \end{cases}$
- $V(\varphi \vee \psi) = \begin{cases} 1 & \text{falls } V(\varphi) = 1 \text{ oder } V(\psi) = 1 \\ 0 & \text{sonst} \end{cases}$

Wenn $V(\varphi) = 1$, dann sagen wir, dass φ von V *erfüllt* wird.

Wir schreiben dann auch $V \models \varphi$ und nennen V ein *Modell* von φ.

Semantik

Beispiel:

Belegung V mit $V(x_1) = 0$ und $V(x_i) = 1$ für alle $i > 1$

Dann z. B.

$$V(\neg x_1) = 1$$
$$V(\neg x_1 \wedge x_2) = 1$$
$$V(\neg(\neg x_1 \wedge x_2)) = 0$$
$$V(\neg(\neg x_1 \wedge x_2) \vee x_3) = 1$$

V ist also ein Modell von $\neg(\neg x_1 \wedge x_2) \vee x_3$.

Semantik

Die Semantik der Junktoren als **Wahrheitstafeln:**

$V(\varphi)$	$V(\neg\varphi)$
0	1
1	0

$V(\varphi)$	$V(\psi)$	$V(\varphi \wedge \psi)$
0	0	0
0	1	0
1	0	0
1	1	1

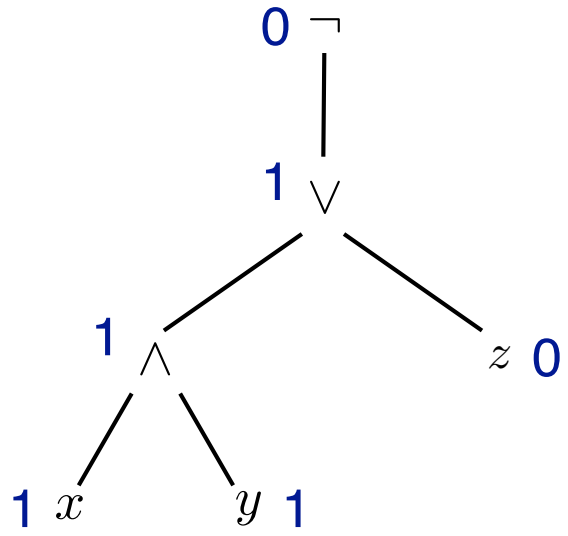
$V(\varphi)$	$V(\psi)$	$V(\varphi \vee \psi)$
0	0	0
0	1	1
1	0	1
1	1	1

Manuelle Auswertung bequem über Baumdarstellung von Formeln:

Beispiel $\neg((x \wedge y) \vee z)$,

$V(x) = V(y) = 1, V(z) = 0$:

(alle anderen Variablen 0)



Iterierte Konjunktion / Disjunktion

Bemerkung zur Notation:

- Wir schreiben

  $\bigwedge_{i=1..n} \varphi_i$ für $\varphi_1 \wedge \cdots \wedge \varphi_n$ (iterierte Konjunktion)

  $\bigvee_{i=1..n} \varphi_i$ für $\varphi_1 \vee \cdots \vee \varphi_n$ (iterierte Disjunktion)

- Wenn $n = 0$, dann

  $\bigwedge_{i=1..n} \varphi_i := 1$ (leere Konjunktion)

  $\bigvee_{i=1..n} \varphi_i := 0$ (leere Disjunktion)

Implikation

Weitere interessante Junktoren sind als Abkürzung definierbar, z. B.:

Implikation $\varphi \to \psi$ steht für $\neg\varphi \vee \psi$

Biimplikation $\varphi \leftrightarrow \psi$ steht für $(\varphi \to \psi) \wedge (\psi \to \varphi)$

$V(\varphi)$	$V(\psi)$	$V(\varphi \to \psi)$
0	0	1
0	1	1
1	0	0
1	1	1

$V(\varphi)$	$V(\psi)$	$V(\varphi \leftrightarrow \psi)$
0	0	1
0	1	0
1	0	0
1	1	1

Wir nehmen an, dass $\neg, \wedge, \vee$ stärker binden als $\to$ und $\leftrightarrow$,

$x \wedge y \to z$ steht also für $(x \wedge y) \to z$

Koinzidenzlemma

Oft ist es unpraktisch, alle (unendlich viele) Variablen belegen zu müssen.

Für den Wahrheitswert einer Formel φ ist nur die Belegung derjenigen Variablen von Bedeutung, die in φ vorkommen. Wir bezeichen diese mit $\text{Var}(\varphi)$.

Lemma 1.3 (Koinzidenzlemma)

Sei φ eine Formel und V_1, V_2 Belegungen mit $V_1(x) = V_2(x)$ für alle $x \in \text{Var}(\varphi)$. Dann ist $V_1(\varphi) = V_2(\varphi)$.

Beweis per Induktion über die Struktur von φ.

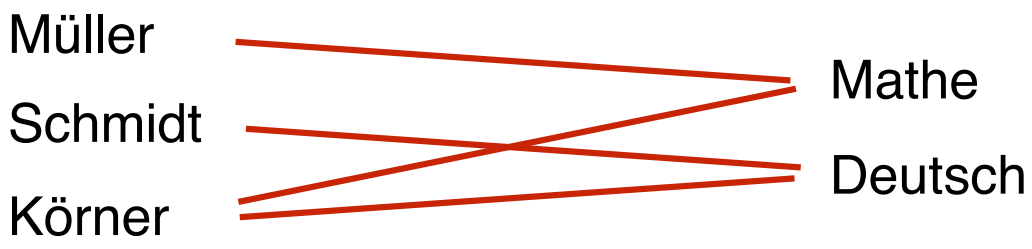
Wenn wir mit einer Formel φ arbeiten, so erlaubt uns das Koinzidenzlemma, in Belegungen nur die Variablen $\text{Var}(\varphi)$ (also endlich viele) zu betrachten.

Eine *Belegung für* φ ist eine Belegung, die nur die Variablen in $\text{Var}(\varphi)$ belegt.

Beispiel Repräsentation

Modellierung eines Zeitplanungs-Problems (Scheduling) in Aussagenlogik

An einer Schule gibt es drei Lehrerinnen mit folgenden Fächerkombinationen:



Es soll folgender Lehrplan erfüllt werden:

	Klasse a)	Klasse b)
Stunde I	Mathe	Deutsch
Stunde II	Deutsch	Deutsch
Stunde III	Mathe	Mathe

Dabei soll jede Lehrerin mindestens 2 Stunden unterrichten.

T1.1

Auswertung

Definition 1.4 (Auswertungsproblem)

Das *Auswertungsproblem der Aussagenlogik* ist:

Gegeben: Aussagenlogische Formel φ, Belegung V für φ

Frage: Gilt $V(\varphi) = 1$?

Theorem 1.5 (Komplexität Auswertungsproblem)

Das Auswertungsproblem der Aussagenlogik ist in Linearzeit lösbar.

Idee Algorithmus für *Polyzeit:*

- Verwende rekursiven Algorithmus, der den Wahrheitswert aller *Teilformeln* von φ bestimmt
- Der Wahrheitswert von atomaren Formeln ist durch V gegeben, zusammengesetzte Teilformeln per Rekursion + Verknüpfungstafel

Thm. 1.5: S. R. Buss: The Boolean Formula Value Problem Is in ALOGTIME. STOC 1987: 123-131. doi: 10.1145/28395.28409

Auswertung

Formale Definition der Teilformeln:

Definition 1.6 (Teilformeln)

Sei φ eine Formel. Die Menge $\mathsf{TF}(\varphi)$ der *Teilformeln* von φ ist induktiv definiert wie folgt:

- $\mathsf{TF}(\varphi) = \{\varphi\}$, wenn $\varphi \in \{0, 1\} \cup \mathsf{Var}$
- $\mathsf{TF}(\neg\varphi) = \{\neg\varphi\} \cup \mathsf{TF}(\varphi)$
- $\mathsf{TF}(\varphi \wedge \psi) = \{\varphi \wedge \psi\} \cup \mathsf{TF}(\varphi) \cup \mathsf{TF}(\psi)$
- $\mathsf{TF}(\varphi \vee \psi) = \{\varphi \vee \psi\} \cup \mathsf{TF}(\varphi) \cup \mathsf{TF}(\psi)$

Also z. B.:

$$\mathsf{TF}(\neg((x \wedge y) \vee z)) = \{x,\ y,\ z,\ x \wedge y,\ (x \wedge y) \vee z,\ \neg((x \wedge y) \vee z)\}$$

Es ist nun einfach, die Details des Algorithmus auszuarbeiten (Übung!)

Äquivalenz

Definition 1.7 (Äquivalenz)

Zwei Formeln φ und ψ sind *äquivalent*, wenn für alle Belegungen V gilt, dass $V(\varphi) = V(\psi)$. Wir schreiben dann $\varphi \equiv \psi$.

Z. B. gilt $x \wedge y \equiv \neg(\neg x \vee \neg y)$

Einfacher Beweis mittels Wahrheitstafeln für *Formeln* φ:

$V(x)$	$V(y)$	$V(x \wedge y)$
0	0	0
0	1	0
1	0	0
1	1	1

$V(x)$	$V(y)$	$V(\neg(\neg x \vee \neg y))$
0	0	0
0	1	0
1	0	0
1	1	1

Beachte: links stehen die Variablen aus $\mathrm{Var}(\varphi)$, es gibt also $2^{|\mathrm{Var}(\varphi)|}$ Zeilen

Äquivalenz: Ersetzungslemma

Äquivalente Formeln sind austauschbar:

Lemma 1.8 (Ersetzungslemma)

Seien φ and ψ äquivalente Formeln, ϑ eine Formel mit $\varphi \in \mathrm{TF}(\vartheta)$ und ϑ' eine Formel, die sich aus ϑ ergibt, indem ein beliebiges Vorkommen von φ durch ψ ersetzt wird. Dann gilt $\vartheta \equiv \vartheta'$.

T1.2

Beweis per Induktion über die Struktur von ϑ.

Allgemeines Vorgehen beim Induktionsbeweis:

Induktionsanfang:

Zeige die Aussage für alle atomaren Formeln $\vartheta \in \{0, 1\} \cup \mathrm{VAR}$.

Induktionsschritt: Zeige:

Wenn die Aussage für ϑ_1 und ϑ_2 gilt, (Induktionsvoraussetzung, IV)
dann auch für $\neg\vartheta_1$, $\vartheta_1 \wedge \vartheta_2$ und $\vartheta_1 \vee \vartheta_2$. (Induktionsbehauptung)

T1.3

Nützliche Äquivalenzen

Im Folgenden wollen wir **einige nützliche Äquivalenzen** etablieren

Genauer gesagt handelt es sich um Äquivalenzschemata, z. B.:

Für alle Formeln φ gilt: $\varphi \equiv \neg\neg\varphi$ Eliminieren doppelter Negation

Beweis per Wahrheitstafel

Nützliche Äquivalenzen

Folgende Äquivalenzen gelten für alle aussagenlogischen Formeln φ, ψ, ϑ:

- $\neg(\varphi \wedge \psi) \equiv \neg\varphi \vee \neg\psi$
  $\neg(\varphi \vee \psi) \equiv \neg\varphi \wedge \neg\psi$

  De Morgansche Gesetze

- $\varphi \wedge \varphi \equiv \varphi$
  $\varphi \vee \varphi \equiv \varphi$

  Idempotenz von Konjunktion und Disjunktion

- $\varphi \wedge \psi \equiv \psi \wedge \varphi$
  $\varphi \vee \psi \equiv \psi \vee \varphi$

  Kommutativität von Konjunktion und Disjunktion

- $\varphi \wedge (\psi \wedge \vartheta) \equiv (\varphi \wedge \psi) \wedge \vartheta$
  $\varphi \vee (\psi \vee \vartheta) \equiv (\varphi \vee \psi) \vee \vartheta$

  Assoziativität von Konjunktion und Disjunktion

Nützliche Äquivalenzen

Mehr nützliche Äquivalenzen:

- $\varphi \wedge (\psi \vee \vartheta) \equiv (\varphi \wedge \psi) \vee (\varphi \wedge \vartheta)$
  $\varphi \vee (\psi \wedge \vartheta) \equiv (\varphi \vee \psi) \wedge (\varphi \vee \vartheta)$
  Distributivgesetze
- $\varphi \wedge (\varphi \vee \psi) \equiv \varphi \equiv \varphi \vee (\varphi \wedge \psi)$
  Absorption
- $\varphi \wedge 1 \equiv \varphi$
  $\varphi \vee 0 \equiv \varphi$
  Neutrales Element für Konjunktion und Disjunktion
- $\varphi \wedge \neg\varphi \equiv 0$
  $\varphi \vee \neg\varphi \equiv 1$
  Kontradiktion und Tautologie

Auch für die (Bi)implikation gibt es interessante Äquivalenzen, z. B.:

- $\varphi \to \psi \equiv \neg\psi \to \neg\varphi$
  Kontraposition

Nachweis weiterer Äquivalenzen

Mittels dieser Äquivalenzen und dem Ersetzungslemma (EL) kann man durch Umformung neue Äquivalenzen nachweisen.

Zum Beispiel $\neg x \wedge \neg y \equiv \neg(x \vee (\neg x \wedge y))$

$\neg(x \vee (\neg x \wedge y))$	$\equiv$	$\neg x \wedge \neg(\neg x \wedge y)$	De Morgan
	$\equiv$	$\neg x \wedge (\neg\neg x \vee \neg y)$	De Morgan + EL
	$\equiv$	$\neg x \wedge (x \vee \neg y)$	doppelte Negation + EL
	$\equiv$	$(\neg x \wedge x) \vee (\neg x \wedge \neg y)$	Distributivgesetz
	$\equiv$	$0 \vee (\neg x \wedge \neg y)$	Kontradiktion + EL
	$\equiv$	$(\neg x \wedge \neg y) \vee 0$	Kommutativgesetz
	$\equiv$	$\neg x \wedge \neg y$	Neutrales Element Disjunktion

Aussagenlogik

1.1 Grundlagen

1.2 Normalformen und funktionale Vollständigkeit

1.3 Erfüllbarkeit, Gültigkeit, Folgerbarkeit, Horn-Formeln

1.4 Resolution

1.5 Kompaktheit

Boolesche Funktionen

Definition 1.9 (Boolesche Funktion)

Eine n-*stellige Boolesche Funktion* ist eine Funktion $f : \{0,1\}^n \to \{0,1\}$.

Für $n \geq 0$ bezeichne

- $\mathcal{B}^n$ die Menge aller n-stelligen Booleschen Funktionen
- $\mathcal{B}$ die Menge $\bigcup_{n \geq 0} \mathcal{B}^n$ aller Booleschen Funktionen

Zum Beispiel:

$\mathcal{B}^0$ besteht aus den beiden konstanten Funktionen 0 und 1.

$\mathcal{B}^1$ besteht aus vier Funktionen f_{00}, f_{10}, f_{01}, f_{11}:

Eingabe	f_{00}	f_{01}	f_{10}	f_{11}
0	0	0	1	1
1	0	1	0	1

Allgemein: $\mathcal{B}^n$ besteht aus 2^{2^n} Funktionen

Aussagenlog. Formeln versus Boolesche Funktionen

Jede aussagenlogische Formel φ mit $|\mathrm{Var}(\varphi)| = n$ berechnet n-stellige Boolesche Funktion f_φ:

- O. B. d. A. sei $\mathrm{Var}(\varphi) = \{x_1, \ldots, x_n\}$
- Belegung V für φ entspricht Eingabe für f_φ:

  i-ter Eingabewert ist $V(x_i)$
- Wert von f_φ bei Eingabe/Belegung V ist $V(\varphi)$

T1.4

Genau diese Funktion stellen wir in der Wahrheitstafel dar!

Umgekehrt findet sich zu jeder Booleschen Funktion auch eine Formel:

Theorem 1.10 (funktionale Vollständigkeit)

Zu jeder Booleschen Funktion $f \in \mathcal{B}$ gibt es eine Formel φ mit $f_\varphi = f$.

T1.5

Normalformen

Der Beweis des Satzes hat als weitere interessante Konsequenz:

Jede Formel ist äquivalent zu einer Formel der Form

$$(\ell_{1,1} \wedge \cdots \wedge \ell_{1,m_1}) \vee \cdots \vee (\ell_{n,1} \wedge \cdots \wedge \ell_{n,m_n})$$

wobei die $\ell_{i,j}$ jeweils die Form x oder $\neg x$ haben.

Dies ist die sogenannte *disjunktive Normalform*.
Dual dazu gibt es auch die wichtige *konjunktive Normalform*.

Normalformen

Definition 1.11 (KNF, DNF)

Ein *Literal* ist eine Formel der Form

- x (*positives Literal*) oder
- $\neg x$ (*negatives Literal*)

Eine Formel φ ist in *konjunktiver Normalform (KNF)*, wenn sie eine Konjunktion von Disjunktionen von Literalen ist:

$$\varphi = \bigwedge_{i=1..n} \bigvee_{j=1..m_i} \ell_{i,j}$$

Eine Formel φ ist in *disjunktiver Normalform (DNF)*, wenn sie eine Disjunktion von Konjunktionen von Literalen ist:

$$\varphi = \bigvee_{i=1..n} \bigwedge_{j=1..m_i} \ell_{i,j}$$

Normalformen

Theorem 1.12 (KNF/DNF-Umwandlung)

Jede Formel lässt sich effektiv in eine äquivalente Formel in KNF bzw. DNF wandeln.

T1.6

Beispiel:

$\varphi = (y \vee \neg(x \vee y)) \wedge \neg z$

x	y	z	$V(\varphi)$
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

DNF:

$$(\neg x \wedge \neg y \wedge \neg z) \vee (\neg x \wedge y \wedge \neg z) \vee (x \wedge y \wedge \neg z)$$

KNF:

$$\neg\Big((\neg x \wedge \neg y \wedge z) \vee (\neg x \wedge y \wedge z) \vee (x \wedge \neg y \wedge \neg z) \vee (x \wedge \neg y \wedge z) \vee (x \wedge y \wedge z) \Big)$$

$$\equiv$$

$$(x \vee y \vee \neg z) \wedge (x \vee \neg y \vee \neg z) \wedge (\neg x \vee y \vee z) \wedge (\neg x \vee y \vee \neg z) \wedge (\neg x \vee \neg y \vee \neg z)$$

Normalformen

Beachte:

- Sowohl DNF als auch KNF werden im Worst Case exponentiell groß

  (2^n viele Disjunkte / Konjunkte, wobei $n = |\mathrm{Var}(\varphi)|$)

- Das lässt sich auch nicht durch eine bessere Konstruktion verhindern

  Man kann z.B. zeigen, dass für die n-äre Paritätsfunktion gilt:

  - sie kann mit einer Formel polynomieller Größe dargestellt werden
  - jede DNF hat mindestens 2^n Disjunkte
  - jede KNF hat mindestens 2^n Konjunkte

  (n-äre Paritätsfunktion: $p_n(t) = 1$ gdw. t ungeradzahlig oft 1 enthält)

- Es gibt sogar Familien von Booleschen Funktionen $f_0, f_1, f_2, \ldots$
  mit $f_n \in \mathcal{B}^n$ für alle $n \geq 0$,
  die sich *gar nicht* mit Formeln polynomieller Größe darstellen lassen.

T1.x

Funktionale Vollständigkeit von Junktorenmengen

Wir haben gesehen:

Mittels der Junktoren $\neg, \wedge, \vee$ kann man für jede Boolesche Funktion f eine „äquivalente“ Formel φ konstruieren

Aus den De Morganschen Gesetzen folgt

$$\varphi \wedge \psi \equiv \neg(\neg\varphi \vee \neg\psi)$$
$$\varphi \vee \psi \equiv \neg(\neg\varphi \wedge \neg\psi)$$

also gilt dasselbe für die Junktormengen $\neg, \wedge$ und $\neg, \vee$

Allgemein stellt sich die Frage:
Welche Junktorenmengen sind in diesem Sinne vollständig?

Funktionale Vollständigkeit von Junktorenmengen

Die Konstanten 0,1 und die Junktoren $\neg, \wedge, \vee$ können als Boolesche Funktionen aus $\mathcal{B}^0$, $\mathcal{B}^1$, bzw. $\mathcal{B}^2$ aufgefasst werden.

Umgekehrt liefert jede Boolesche Funktion $f \in \mathcal{B}^n$ einen n-ären Junktor: Zeile $t = (w_1, \ldots, w_n) \in \{0,1\}^n$ in Wahrheitstafel hat Wert $f(t)$.

T1.7

Wir werden im Folgenden nicht streng zwischen Junktoren und Booleschen Funktionen unterscheiden.

Weitere interessante Junktoren neben $0, 1, \neg, \wedge, \vee, \rightarrow, \leftrightarrow$ z.B.:

Exklusives Oder

$V(\varphi)$	$V(\psi)$	$V(\varphi \oplus \psi)$
0	0	0
0	1	1
1	0	1
1	1	0

Nand

$V(\varphi)$	$V(\psi)$	$V(\varphi \mid \psi)$
0	0	1
0	1	1
1	0	1
1	1	0

Funktionale Vollständigkeit von Junktorenmengen

Definition 1.13 (funktionale Vollständigkeit)

Eine Menge $\Omega \subseteq \mathcal{B}$ von Booleschen Funktionen ist *funktional vollständig* wenn es für jede Boolesche Funktion $f \in \mathcal{B}^n$, $n \geq 1$ eine Formel φ mit Junktoren aus Ω gibt, so dass $f_\varphi = f$.

Wir wissen bereits, dass folgende Mengen funktional vollständig sind:

$$\{\neg, \wedge, \vee\} \qquad \{\neg, \wedge\} \qquad \{\neg, \vee\}$$

Weitere funktional vollständige Mengen:

- $\{\neg, \rightarrow\}$
  Da $\{\neg, \vee\}$ funktional vollständig und $\varphi \vee \psi \equiv \neg\varphi \rightarrow \psi$
- $\{\wedge, \oplus, 1\}$
  Da $\{\neg, \wedge\}$ funktional vollständig und $\neg\varphi \equiv 1 \oplus \varphi$

Funktionale Vollständigkeit von Junktorenmengen

Weitere funktional vollständige Menge:

- $\{|\}$

  Da $\{\neg, \wedge\}$ funktional vollständig, $\neg\varphi \equiv \varphi \mid \varphi$ und $\varphi \wedge \psi \equiv (\varphi \mid \psi) \mid (\varphi \mid \psi)$

T1.8

Nicht funktional vollständig z. B. $\{\wedge, \vee, \rightarrow\}$:

- Jede mit $\wedge, \vee, \rightarrow$ gebildete Formel φ erfüllt $f_\varphi(1, \ldots, 1) = 1$

  Beweis per Induktion über die Struktur von φ:

  - wenn $\varphi = x$, dann $V_1(\varphi) = 1$
  - wenn $\varphi = \psi \wedge \vartheta$ und $V_1(\psi) = V_1(\vartheta) = 1$, dann $V_1(\varphi) = 1$
  - analog für $\varphi = \psi \vee \vartheta$ und $\varphi = \psi \rightarrow \vartheta$

- Es gibt also keine zu $\neg x$ äquivalente Formel

Aussagenlogik

1.1 Grundlagen

1.2 Normalformen und funktionale Vollständigkeit

1.3 Erfüllbarkeit, Gültigkeit, Folgerbarkeit, Horn-Formeln

1.4 Resolution

1.5 Kompaktheit

Erfüllbarkeit, Gültigkeit

Definition 1.14 (Erfüllbarkeit, Gültigkeit)

Eine Formel heißt

- *erfüllbar*, wenn sie ein Modell hat (sonst *unerfüllbar*)
- *gültig* oder *Tautologie*, wenn jede Belegung ein Modell ist

Beispiele für unerfüllbare Formeln:

0 $x \wedge \neg x$ $x \wedge \neg y \wedge (x \rightarrow y)$ $(x \vee y) \wedge (\neg x \vee \neg y) \wedge (\neg x \vee y) \wedge (x \vee \neg y)$

Beispiele für gültige Formeln:

1 $x \vee \neg x$ $\neg(x \wedge y) \leftrightarrow \neg x \vee \neg y$

$(x \wedge y) \vee (\neg x \wedge \neg y) \vee (\neg x \wedge y) \vee (x \wedge \neg y)$

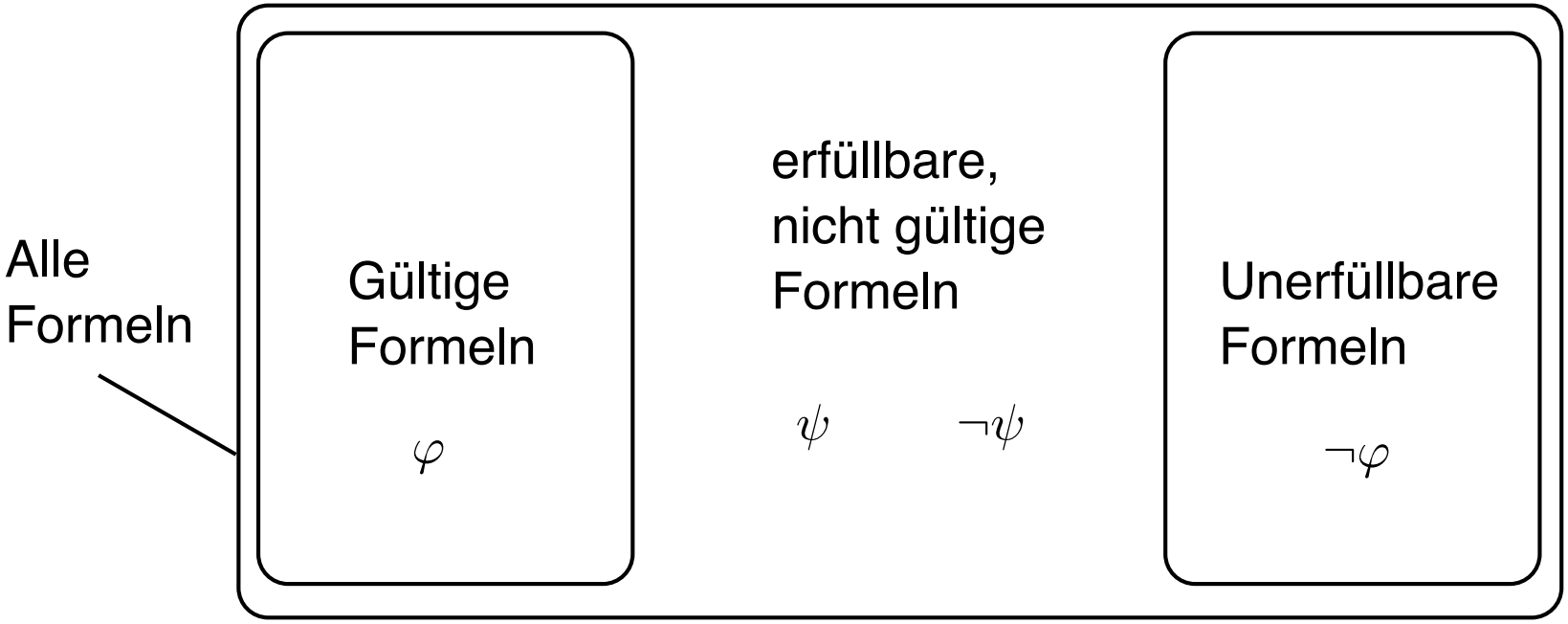
Dualität

Folgt direkt aus Definition Erfüllbarkeit/Tautologie + Semantik Negation:

Lemma 1.15 (Dualität Erfüllbarkeit, Gültigkeit)

Eine Formel φ ist

- gültig gdw. $\neg\varphi$ unerfüllbar ist.
- erfüllbar gdw. $\neg\varphi$ nicht gültig ist.



Erfüllbarkeits-, Gültigkeitsproblem

Definition 1.16 (Erfüllbarkeitsproblem, Gültigkeitsproblem)

Das *Erfüllbarkeitsproblem der Aussagenlogik* ist:

Gegeben: Aussagenlogische Formel φ

Frage: Ist φ erfüllbar?

Das *Gültigkeitsproblem der Aussagenlogik* ist:

Gegeben: Aussagenlogische Formel φ

Frage: Ist φ eine Tautologie?

Offensichtlicher, naiver Algorithmus für das **Gültigkeitsproblem:**

Zähle alle 2^n Belegungen für φ auf (wobei $n = |\text{Var}(\varphi)|$).

Für jede Belegung V prüfe in Linearzeit, ob $V \models \varphi$

Erfüllbarkeitsproblem auf (Komplement des) Gültigkeitsproblem(s) in Polyzeit reduzierbar mit Lemma 1.15 (und umgekehrt).

Komplexitätsresultate

Theorem 1.17 (Komplexität)

Das Erfüllbarkeitsproblem der Aussagenlogik ist NP-vollständig.
Dies gilt auch für Formeln in KNF, sogar bei max. 3 Literalen pro Konjunkt.

Das Gültigkeitsproblem der Aussagenlogik ist coNP-vollständig.
Dies gilt auch für Formeln in DNF, sogar bei max. 3 Literalen pro Disjunkt.

T1.9

Für Formeln in KNF ist Gültigkeit leicht (in Linearzeit) zu entscheiden, ebenso Erfüllbarkeit für Formeln in DNF (Beweis als Übung):

Lemma 1.18 (Einfache Fälle)

Eine DNF-Formel ist erfüllbar gdw. es ein Disjunkt gibt, das keine Literale der Form $x, \neg x$ enthält.

Eine KNF-Formel ist gültig gdw. jedes Konjunkt zwei Literale der Form $x, \neg x$ enthält.

Folgerbarkeit

Definition 1.19 (Folgerbarkeit, Konsequenz)

Eine Formel ψ ist *folgerbar* aus einer Formel φ,
wenn für alle Belegungen V mit $V \models \varphi$ auch gilt, dass $V \models \psi$.

Wir nennen ψ dann auch eine *Konsequenz* von φ und schreiben $\varphi \models \psi$.

Für eine (potentiell unendliche) Formel*menge* Γ schreiben wir $\Gamma \models \psi$,
wenn für alle Beleg. V mit $V \models \Gamma$ (d. h. $V \models \varphi$ für alle $\varphi \in \Gamma$) gilt: $V \models \psi$.

Beispiele:

$$x \wedge y \models x$$

$$x \models x \vee y$$

$$\varphi \wedge (\varphi \to \psi) \models \psi \qquad \text{(Modus Ponens)}$$

$$\{\varphi, \varphi \to \psi\} \models \psi \qquad \text{(Modus Ponens)}$$

Offensichtlich: $\varphi \equiv \psi$ gdw. $\varphi \models \psi$ und $\psi \models \varphi$

Wenn $|\Gamma| < \infty$, dann gilt: $\Gamma \models \psi$ gdw. $\bigwedge_{\varphi \in \Gamma} \varphi \models \psi$

Folgerbarkeit vs. Gültigkeit

Theorem 1.20 (Folgerbarkeit und Gültigkeit)

Für alle Formeln φ, ψ gilt:

1. $\varphi \models \psi$ gdw. $\varphi \to \psi$ gültig ist (aka *Deduktionstheorem*)
2. φ ist gültig gdw. $1 \models \varphi$.

Analog zum Erfüllbarkeits-/Gültigkeitsproblem kann man ein *Folgerbarkeitsproblem* definieren.

Das Lemma liefert auch wechselseitige Polyzeit-Reduktionen zwischen Gültigkeitsproblem und Folgerbarkeitsproblem.

Das Folgerbarkeitsproblem hat also dieselbe Komplexität wie das Gültigkeitsproblem (coNP-vollständig).

Horn-Formeln

Eine wichtige Klasse von Formeln mit besseren Berechnungseigenschaften sind die Horn-Formeln (nach Alfred Horn).

Definition 1.21 (Horn-Formel)

Eine *aussagenlogische Horn-Formel* ist eine KNF-Formel $\varphi = \bigwedge_i \bigvee_j \ell_{i,j}$, wobei jede Disjunktion $\bigvee_j \ell_{i,j}$ höchstens ein positives Literal enthält.

Beispiel: $(\neg x \vee \neg y \vee z) \wedge (\neg y \vee \neg z) \wedge x$

Vier mögliche Formen von Konjunkten *(Horn-Klauseln)*:

Negative Literale + 1 positives Literal	Nur negative Literale
Nur ein positives Literal	(Gar keine Literale $\equiv 0$, daher uninteressant)

Horn-Formeln

Anschaulicher:

$$
\begin{array}{rcll}
x & & & \textit{Fakt} \\
\neg x_1 \vee \cdots \vee \neg x_k \vee x & \equiv & x_1 \wedge \cdots \wedge x_k \rightarrow x & \textit{Regel} \\
\neg x_1 \vee \cdots \vee \neg x_k & \equiv & x_1 \wedge \cdots \wedge x_k \rightarrow 0 & \textit{Constraint}
\end{array}
$$

Beispiel Horn-Formel: Konjunktion von

$$
\begin{array}{rcl}
\text{Regen} & & \\
\text{Regen} & \rightarrow & \text{Niederschlag} \\
\text{Regen} & \rightarrow & \text{Temp}{\geq}0 \\
\text{Temp}{\geq}0 \wedge \text{Temp}{<}0 & \rightarrow & 0
\end{array}
\qquad
\begin{array}{rcl}
\text{Schnee} & & \\
\text{Schnee} & \rightarrow & \text{Niederschlag} \\
\text{Schnee} & \rightarrow & \text{Temp}{<}0
\end{array}
$$

Hierbei sind „Regen“, „Schnee“, „Temp$>$0“, . . . Aussagenvariablen

Erfüllbarkeitsproblem für Horn-Formeln

Theorem 1.22 (Effiziente Erfüllbarkeit)

Das Erfüllbarkeitsproblem für Horn-Formeln kann in Linearzeit gelöst werden.

Polyzeit-Algorithmus für Eingabe φ:

$V := \{x \in \text{VAR} \mid x \text{ ist Konjunkt von } \varphi\}$
while es gibt Konjunkt $x_1 \wedge \cdots \wedge x_k \to x$ mit $\{x_1, \ldots, x_k\} \subseteq V$ und $x \notin V$ **do**
    $V := V \cup \{x\}$
done
if es gibt ein Konjunkt $x_1 \wedge \cdots \wedge x_k \to 0$ mit $\{x_1, \ldots, x_k\} \subseteq V$ **then**
    return „unerfüllbar“
else
    return „erfüllbar“

Zu Thm. 1.22: W. F. Dowling, J. H. Gallier: Linear-time algorithms for testing the satisfiability of propositional Horn formulae. J. of Log Prog., 1(3): 267–284 (1984). doi:10.1016/0743-1066(84)90014-1

Beispiel T1.10

Erfüllbarkeitsproblem für Horn-Formeln

Polyzeit-Algorithmus für Eingabe φ:

$V := \{x \in \text{VAR} \mid x \text{ ist Konjunkt von } \varphi\}$
while es gibt Konjunkt $x_1 \wedge \cdots \wedge x_k \to x$ mit $\{x_1, \ldots, x_k\} \subseteq V$ und $x \notin V$ **do**
    $V := V \cup \{x\}$
done
if es gibt ein Konjunkt $x_1 \wedge \cdots \wedge x_k \to 0$ mit $\{x_1, \ldots, x_k\} \subseteq V$ **then**
    return „unerfüllbar“
else
    return „erfüllbar“

Wir unterscheiden im Folgenden nicht zwischen einer Belegung V (Abbildung VAR $\to \{0, 1\}$) und der Menge $\{x \mid V(x) = 1\}$

Lemma 1.23

Der Algorithmus ist korrekt und läuft in polynomieller Zeit.

T1.11

Minimale Modelle

Für erfüllbare Horn-Formel φ ist die im Algorithmus berechnete Belegung V ein *minimales* Modell in folgendem Sinne:

1. V ist Modell von φ
2. Wenn $\widehat{V}$ Modell von φ, dann $V \subseteq \widehat{V}$

Wir erhalten also als Korollar:

Korollar 1.24

Jede erfüllbare Horn-Formel hat ein minimales Modell.

Minimale Modelle haben zahlreiche interessante Eigenschaften.

Wir können sie z. B. für Beweise der **Nichtausdrückbarkeit** verwenden.

Ausdrucksstärke

Ausdrucksstärke von Horn-Formeln:

Welche AL-Formeln kann man als Horn-Formel ausdrücken, welche nicht?

Ausdrückbar z. B.:

$$x \to y \wedge z \quad \equiv \quad (x \to y) \wedge (x \to z)$$

$$x \vee y \to z \quad \equiv \quad (x \to z) \wedge (y \to z)$$

Nicht ausdrückbar z. B. $x \vee y$

(Beispiel: wir können ausdrücken, dass Temp$<0 \wedge$ Temp$\geq 0 \to 0$,
nicht aber, dass Temp$<0 \vee$ Temp≥ 0)

Lemma 1.25 (Nicht-Horn-Ausdrückbarkeit)

Keine Horn-Formel ist äquivalent zu $x \vee y$.

T1.12

Intuitiv: Horn-Formeln sind der disjunktionsfreie Teil von Aussagenlogik.

Zusammenfassung Schlussfolgerungsprobleme

	Auswertungs-problem	Erfüllbarkeits-problem	Gültigkeits-problem	Folgerbarkeits-problem
Horn-Formeln	in Linearzeit	in Polyzeit	in Linearzeit	in Polyzeit
Aussagenlogik	in Linearzeit	NP-vollständig	coNP-vollständig	coNP-vollständig
Prädikatenlogik 1. Stufe				
Prädikatenlogik 2. Stufe				

Aussagenlogik

1.1 Grundlagen

1.2 Normalformen und funktionale Vollständigkeit

1.3 Erfüllbarkeit, Gültigkeit, Folgerbarkeit, Horn-Formeln

1.4 Resolution

1.5 Kompaktheit

Resolution

Ein *Kalkül* besteht aus einer Sammlung rein syntaktischer *Umformungs-regeln,* mit denen man Formeln in andere Formeln transformieren kann.

Es gibt viele Kalküle für verschiedenste Logiken, z. B.

Sequenzenkalkül, Tableau-Kalkül, Hilbertsches Axiomensystem etc.

Zwei Arten von Kalkülen:

- Ausgehend von Axiomen und Schlussfolgerungsregeln, erzeuge genau die gültigen Formeln
- Ausgehend von einer gegebenen Formel, erzeuge durch Regelanwendung die Konstante 0 gdw. die Formel unerfüllbar ist

Wir betrachten einen wichtigen und eleganten Kalkül für Unerfüllbarkeit in Aussagenlogik: *Resolution*

Klauseln und Klauselmengen

Resolution arbeitet mit Formeln in KNF, jedoch in leicht anderer Darstellung

Definition 1.26 (Klausel, Klauselmenge)

Eine *Klausel* ist eine endliche Menge von Literalen. Die leere Klausel bezeichnen wir mit $\square$.

Einer KNF-Formel $\varphi = \bigwedge_{i=1..n} \bigvee_{j=1..m_i} \ell_{ij}$ wird *Klauselmenge* $M(\varphi)$ wie folgt zugeordnet:

- i-te Disjunktion $\bigvee_{j=1..m_i} \ell_{ij}$ erzeugt Klausel $C_i = \{\ell_{i1}, \ldots, \ell_{im_i}\}$
- $M(\varphi) = \{C_1, \ldots, C_n\}$.

Beispiel: die Formeln

$$(x_1 \vee \neg x_2) \wedge x_3\ , \quad (x_1 \vee x_1 \vee \neg x_2) \wedge (x_3 \vee x_3), \quad x_3 \wedge (x_1 \vee \neg x_2) \wedge (\neg x_2 \vee x_1)$$

haben alle die Klauselmenge $M = \{\{x_1, \neg x_2\}, \{x_3\}\}$.

Klauseln und Klauselmengen

Umgekehrt entspricht eine Klausel C der Formel $\bigvee_{\ell \in C} \ell$ und eine endliche Klauselmenge M entspricht der Formel $\bigwedge_{C \in M} \bigvee_{\ell \in C} \ell$.

Dies gibt uns auch eine Semantik für Klauseln und Klauselmengen.

Wir können also Begriffe wie Erfüllbarkeit und Äquivalenz für Klauseln und Klauselmengen verwenden.

Beachte:

- $\square$ entspricht der „leeren Disjunktion“ und ist unerfüllbar
- jede Klauselmenge, die $\square$ enthält, ist unerfüllbar
- die leere Klauselmenge entspricht der „leeren Konjunktion“ und ist erfüllbar

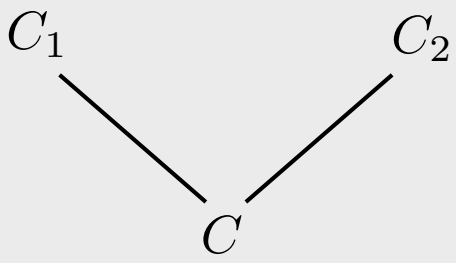
Resolvente

Zum Negieren von Literalen definiere $\overline{x} := \neg x$ und $\overline{\neg x} := x$

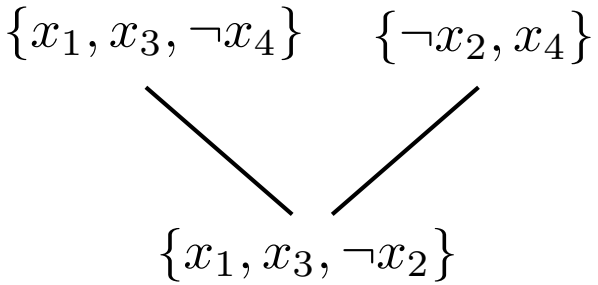
Definition 1.27 (Resolvente)

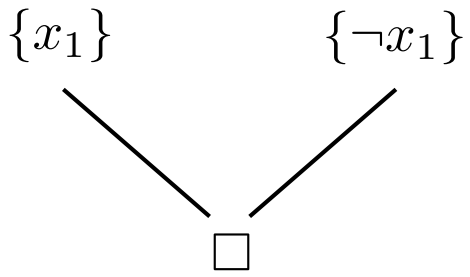
Seien C_1, C_2 Klauseln. Klausel C ist *Resolvente* von C_1 und C_2 gdw. es Literal ℓ gibt mit $\ell \in C_1$, $\overline{\ell} \in C_2$ und $C = (C_1 \setminus \{\ell\}) \cup (C_2 \setminus \{\overline{\ell}\})$.

Wir schreiben dann:



Beispiele:





Resolventenbildung

Lemma 1.28 (Resolutionslemma)

Sei M eine Klauselmenge, $C_1, C_2 \in M$ und C Resolvente von C_1 und C_2. Dann $M \equiv M \cup \{C\}$.

T1.13

Folgende Notation beschreibt das wiederholte Bilden von Resolventen.

Definition 1.29 (Res)

Für jede Klauselmenge M sei

- $\mathrm{Res}(M) := M \cup \{C \mid C \text{ Resolvente zweier Klauseln aus } M\}$
- $\mathrm{Res}^0(M) := M, \qquad \mathrm{Res}^{i+1}(M) := \mathrm{Res}(\mathrm{Res}^i(M))$
- $\mathrm{Res}^*(M) := \bigcup_{i \geq 0} \mathrm{Res}^i(M)$

Beispiel: $\varphi = x_1 \wedge (\neg x_1 \vee x_2) \wedge (\neg x_2 \vee x_3) \wedge \neg x_3$

T1.14

Analyse des Resolutionskalküls

Im Allgemeinen:

- Ein Kalkül *terminiert,*
  wenn sich für jede Eingabe nur endlich viele Formeln erzeugen lassen.
- Ein Kalkül heißt *korrekt,*
  wenn sich nur gewünschte Formeln erzeugen lassen.
- Ein Kalkül heißt *vollständig,*
  wenn sich jede gewünschte Formel erzeugen lässt.

Im Fall von Resolution:

□ gewünscht gdw. Eingabeformel unerfüllbar

Analyse des Resolutionskalküls

Terminierung:

Lemma 1.30 (Terminierung)

Für jede endliche Klauselmenge M gibt es ein $i \geq 1$ mit $\mathsf{Res}^*(M) = \mathsf{Res}^i(M)$.

Beweis:

M hat nur endlich viele Literale.

Darüber lassen sich nur endlich viele Klauseln bilden.

Also ist $\mathsf{Res}^*(M)$ endlich.

Daraus folgt: $\mathsf{Res}^*(M) = \mathsf{Res}^i(M)$ für ein $i \geq 1$.

Korrektheit und Vollständigkeit:

Theorem 1.31 (Resolutionssatz, Robinson 1965)

Eine endliche Klauselmenge M ist unerfüllbar gdw. $\square \in \mathsf{Res}^*(M)$.

T1.15

Der Satz kann auch für unendliche Klauselmengen bewiesen werden.

Resolution als Algorithmus

Wegen Lemma 1.30 (Terminierung) stabilisiert sich die Folge

$$M = \mathsf{Res}^0(M) \subseteq \mathsf{Res}^1(M) \subseteq \cdots$$

nach höchstens 2^n Schritten.

Dies liefert den folgenden Algorithmus für Erfüllbarkeit in der Aussagenlogik:

$R_0 := M$
$i := 0$
repeat
    $i := i + 1$
    $R_i := \mathsf{Res}(R_{i-1})$
    if $\square \in R_i$ **then return** „unerfüllbar“
until $R_i = R_{i-1}$
return „erfüllbar“

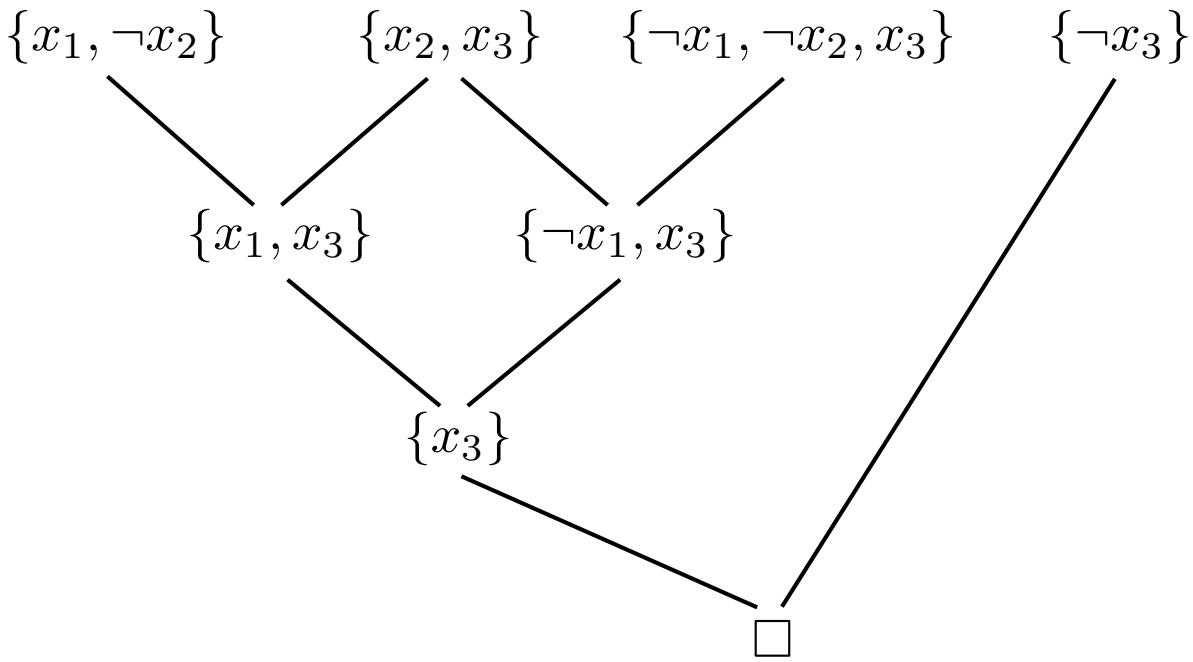
Resolutionsbeweise

Resolutionsbeweis:

Darstellung der Ableitung von $\square$ mittels Resolventen als Graph

Beispiel:

$$\varphi = (x_1 \vee \neg x_2) \wedge (x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3) \wedge \neg x_3$$



Resolutionsbeweise

Definition 1.32 (Resolutionsbeweis)

Sei M eine endliche Klauselmenge. Ein *Resolutionsbeweis für* M ist eine Folge $C_1, \ldots, C_m$ von Klauseln, für die gilt:

- Für alle $i \leq m$ ist entweder $C_i \in M$
  oder C_i ist Resolvente zweier C_j, C_k mit $j, k < i$.
- $C_m = \square$.

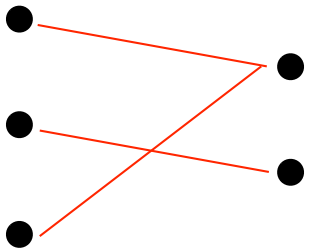
Beachte:

$\text{Res}^*(M)$ entspricht nicht *einem* Resolutionsbeweis, sondern enthält *alle* Resolutionsbeweise für die Unerfüllbarkeit von M
(und auch Klauseln, die in keinem Resolutionsbeweis vorkommen)

Exkurs: Beweislänge

Es gibt Klauselmengen, für die jeder Resolutions**beweis** exponentiell lang ist.

Schubfachprinzip: wenn man $n+1$ Objekte auf n Schubladen verteilt,
enthält mindestens eine Schublade 2 Objekte



Als aussagenlogische Formel:

$$(x_{11} \vee x_{12}) \wedge (x_{21} \vee x_{22}) \wedge (x_{31} \vee x_{32}) \rightarrow (x_{11} \wedge x_{21}) \vee (x_{11} \wedge x_{31}) \vee (x_{21} \wedge x_{31}) \vee (x_{12} \wedge x_{22}) \vee (x_{12} \wedge x_{32}) \vee (x_{22} \wedge x_{32})$$

Da das Schubfachprinzip gültig ist, ist die Negation dieser Formel unerfüllbar.

Exkurs: Beweislänge

Für $n+1$ Objekte und n Schubfächer, negiert, in KNF gewandelt:

$$\varphi_n = \bigwedge_{i=1..n+1} \bigvee_{j=1..n} x_{ij} \;\wedge\; \bigwedge_{1\leq i<i'\leq n+1} \bigwedge_{j=1..n} (\neg x_{ij} \vee \neg x_{i'j})$$

Gibt Folge von Formeln $\varphi_1, \varphi_2, \varphi_3, \ldots$. Ohne Beweis:

Theorem 1.33 (Haken 1985)

Es gibt Konstanten $k_1, k_2 > 1$ so dass für alle $n \geq k_1$:

- φ_n hat $\mathcal{O}(n^3)$ Klauseln mit je höchstens n Variablen
- jeder Resolutionsbeweis für φ_n hat Länge $\geq (k_2)^n$

Andere Kalküle haben aber u. U. kurze Beweise für diese Formelklasse.

Einheitsresolution

Definition 1.34 (Hornklausel)

Eine Klausel ist eine *Hornklausel*, wenn sie höchstens ein positives Literal enthält.

Beachte: $\square$, $\{x\}$, $\{\neg x\}$ sind also (spezielle) Horn-Klauseln

Definition 1.35 (Einheitsresolvente)

Seien C_1, C_2, C Klauseln. C ist *Einheitsresolvente* von C_1 und C_2, wenn C Resolvente von C_1 und C_2 ist und C_1 die Form $\{x\}$ hat.

Wir setzen

$$\mathrm{ERes}(M) := M \cup \{C \mid C \text{ Einheitsresolvente zweier Klauseln aus } M\}$$

und definieren $\mathrm{ERes}^i(M)$ und $\mathrm{ERes}^*(M)$ analog zu $\mathrm{Res}^i(M)$ und $\mathrm{Res}^*(M)$.

Beispiel: $\{\{\neg x_1, \neg x_2, \neg x_3, x_4\}, \{x_1\}, \{x_2\}, \{x_3\}, \{\neg x_3, \neg x_4\}\}$

T1.16

Einheitsresolution

Auf Hornklauseln ist Einheitsresolution ausreichend:

Theorem 1.36 (Resolutionssatz für Einheitsresolution)

Eine endliche Menge M von Hornklauseln ist unerfüllbar gdw. $\square \in \mathsf{ERes}^*(M)$

T1.17

Der Beweis zeigt auch, dass es für jede unerfüllbare Horn-Formel φ einen Resolutionsbeweis gibt, der höchstens $m \cdot (v+1)$ Schritte hat, wobei $m = \max\{|C| \mid C \in M(\varphi)\}$ und $v = |\mathsf{Var}(\varphi)|$:

- die Anzahl Variablen in V^* ist begrenzt durch v
- für jede Variable in V^* und für $\square$ jeweils Beweis der Länge m

Da $\mathsf{ERes}^*(M)$ *alle* Resolutionsbeweise für M enthält, ist der naive Einheits-Resolutionsalgorithmus dennoch kein Polyzeit-Verfahren

Man kann ihn aber durch eine weitere Einschränkung (Variablenordnung) zu einem Polyzeit-Algorithmus machen.

SAT-Solver

Erfüllbarkeit in Aussagenlogik nennt man auch das *SAT-Problem.*

Obwohl SAT NP-vollständig ist, gibt es heute sehr effiziente *SAT-Solver*, die auch sehr große Formeln (Tausende von Variablen) lösen können.

Dies ist deshalb von großer Bedeutung, weil sich viele NP-vollständige Probleme in sehr natürlicher Weise als KNF kodieren lassen

Moderne SAT-Solver basieren auf der sogenannten DPLL-Methode (nach Davis-Putnam-Logemann-Loveland)

Wirklich effizient werden SAT-Solver aber erst durch zahlreiche raffinierte (und teils mathematisch recht anspruchsvolle) Optimierungen

(Lingeling, Minisat, Glucose, zchaff, precosat, Sat4J – siehe SAT competitions)

DPLL

Einfacher Backtracking-Algorithmus für SAT (Eingabe Klauselmenge M):

- Wähle Literal ℓ, weise Wahrheitswert 1 zu
- Vereinfache M zu M^+ (s. Beweis Resolutionssatz)
- Prüfe M^+ auf Erfüllbarkeit (rekursiver Aufruf)
  wenn ja, gib „erfüllbar“ aus
  sonst wiederhole mit Wahrheitswert 0 für ℓ

DPLL benutzt Optimierungen, die den Suchraum wirksam beschränken, indem sie nichtdeterministische Entscheidungen frühzeitig vermeiden

- *Unit Propagation* (Einheitsresolution)
- *Pure Literal Elimination*
  (Löschen von Literalen, die nur positiv oder nur negativ in M auftreten)

DPLL – Hauptideen

Genauere Beschreibung der wesentlichen DPLL-Optimierungen

Unit Propagation (Einheitsresolution)

- Belege so früh wie möglich Einheitsklauseln $\{\ell\}$ entsprechend

$\rightsquigarrow$ Lösche alle Klauseln, die ℓ enthalten *(Unit Subsumption)*

$\rightsquigarrow$ Lösche $\neg\ell$ aus allen übrigen Klauseln (der eigentl. Resolutionsschritt!)

Pure Literal Elimination

- Literal ℓ ist *pur* in M, wenn M nur ℓ und nicht $\overline{\ell}$ enthält
- Pure Literale tragen nichts zur Unerfüllbarkeit von M bei
  (Setzen von $V(\ell) = 1$ macht alle Klauseln mit ℓ wahr)

$\rightsquigarrow$ Lösche alle Klauseln, die ℓ enthalten

Optimierungen werden *am Anfang* jedes Unteraufrufs angewendet

Der DPLL-Algorithmus

Function `DPLL(`M`)`**:**

input : Klauselmenge M

output: Wahrheitswert (`true` $\hat{=}$ „erfüllbar“, `false` $\hat{=}$ „unerfüllbar“)

while M *enthält Einheitsklausel* $\{\ell\}$ **do**

- Lösche alle Klauseln aus M, die ℓ enthalten `//Unit Subs.`
- Lösche $\neg\ell$ aus allen übrigen Klauseln `//Unit Res.`

if $\square \in M$ **then return** `false`

while M *enthält pures Literal* ℓ **do**

- Lösche alle Klauseln aus M, die ℓ enthalten `//Pure Lit Elim.`

if $M = \emptyset$ **then return** `true`

Wähle nichtdeterministisch ℓ in M `//nichtdet. Verzweigung`

if `DPLL(`$M \cup \{\{\ell\}\}$`)` **then return** `true`

else if `DPLL(`$M \cup \{\{\neg\ell\}\}$`)` **then return** `true`

else return `false`

Beispiel: $M = \{\{x_1, \neg x_2, x_3\},\ \{\neg x_1, x_2, \neg x_3\},\ \{\neg x_1, x_3, \neg x_4\},\ \{\neg x_1, x_3\}\}$

T1.18

Hilbert-Kalkül

Wir betrachten noch kurz ein weiteres Beispiel für einen Kalkül.

Der Hilbert-Kalkül verwendet Formeln über der Junktormenge $\{\rightarrow, \neg\}$ und basiert auf den folgenden Axiomenschemata:

1. $\varphi \rightarrow (\psi \rightarrow \varphi)$
2. $(\varphi \rightarrow (\psi \rightarrow \vartheta)) \rightarrow ((\varphi \rightarrow \psi) \rightarrow (\varphi \rightarrow \vartheta))$
3. $(\varphi \rightarrow \psi) \rightarrow (\neg\psi \rightarrow \neg\varphi)$
4. $\varphi \rightarrow (\neg\varphi \rightarrow \psi)$
5. $(\neg\varphi \rightarrow \varphi) \rightarrow \varphi$

Aus diesen Axiomenschemata kann man mittels einer einzigen Schlussfolgerungsregel, dem *Modus Ponens*, alle gültigen Formeln herleiten

Hilbert-Kalkül

Definition 1.37 (Herleitbarkeit im Hilbert-Kalkül)

Die Menge der *herleitbaren Formeln* ist die kleinste Menge, so dass:

- jede Instanz der Axiomenschemata 1–5 ist herleitbar
  (Instanz: Teilformeln φ, ψ, ϑ beliebig ersetzen)
- wenn φ herleitbar und $\varphi \to \psi$ herleitbar, dann ψ herleitbar
  (Modus Ponens)

Beispiel: die Formel $x \to x$ ist herleitbar **T1.19**

Ohne Beweis:

Theorem 1.38 (Korrektheit & Vollständigkeit Hilbert-Kalkül)

Eine Formel φ ist gültig gdw. sie im Hilbert-Kalkül herleitbar ist.

Resolutionskalkül vs. Hilbert-Kalkül

	Resolutionskalkül	Hilbert-Kalkül
Ziel	zeigt Unerfüllbarkeit gegebener Formel	erzeugt alle gültigen Formeln
Formeln	in KNF	über Junktormenge $\{\rightarrow, \neg\}$
Arbeits-weise	Herleitung der leeren Klausel mittels Resolventenbildung	Herleitung neuer Formeln aus Axiomen mittels Modus Ponens
Vollständ.-beweis	recht einfach	recht aufwändig
Anwendung	automatisches Entscheiden von Erfüllbarkeit	Modellierung mathematischen Schließens

Aussagenlogik

1.1 Grundlagen

1.2 Normalformen und funktionale Vollständigkeit

1.3 Erfüllbarkeit, Gültigkeit, Folgerbarkeit, Horn-Formeln

1.4 Resolution

1.5 Kompaktheit

Kompaktheit

Manchmal ist es nützlich, mit *unendlichen statt mit endlichen Mengen* aussagenlogischer Formeln zu arbeiten.

Definition 1.39 (Semantik Formelmengen)

Sei Γ eine (endliche oder unendliche) Formelmenge.

- Belegung V *erfüllt* Γ ($V \models \Gamma$), wenn $V \models \varphi$ für alle $\varphi \in \Gamma$.
- Γ ist *erfüllbar*, wenn es Belegung $V \models \Gamma$ gibt.
- Formel ψ *folgt aus* Γ ($\Gamma \models \psi$), wenn für alle $V \models \Gamma$ auch $V \models \psi$ gilt.

Ein zentrales Resultat zum Verständnis unendlicher Formelmengen ist der Kompaktheitssatz:

Theorem 1.40 (Kompaktheitssatz)

Für alle (potentiell unendlichen) Mengen $\Gamma \subseteq$ AL gilt:

Γ ist erfüllbar gdw. jede endliche Teilmenge von Γ erfüllbar ist.

Wir betrachten zunächst eine Beispielanwendung.

Kompaktheit – Beispielanwendung

Definition 1.41 (4-Färbbarkeit)

Ein (ungerichteter) *Graph* $G = (V, E)$ besteht aus

- einer Menge $V \subseteq \{v_1, v_2, \dots\}$ von *Knoten* und
- einer Menge E von *Kanten*, also Teilmengen $\{v, v'\} \subseteq V$ mit $v \neq v'$.

G heißt *4-färbbar*, wenn es eine Abbildung $f : V \to \{c_1, c_2, c_3, c_4\}$ gibt, so dass $f(v) \neq f(v')$ für alle $\{v, v'\} \in E$. So ein f heißt *4-Färbung*.

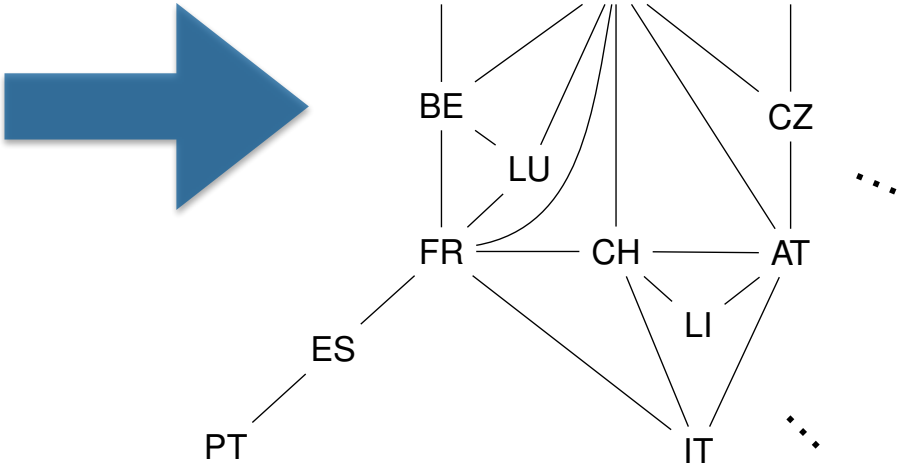
Der bekannte 4-Farben-Satz für endliche Graphen:

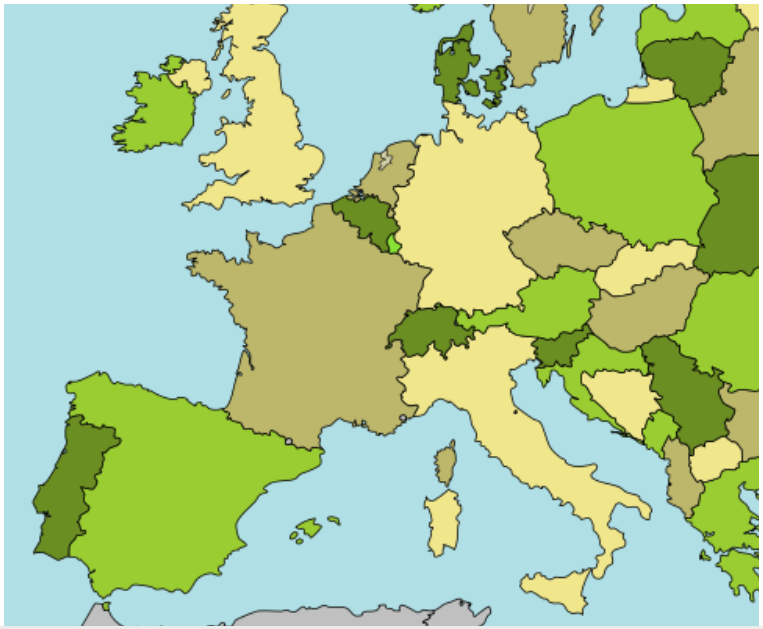
Theorem 1.42 (4-Farben-Satz, endliche Graphen)

Jeder endliche planare Graph ist 4-färbbar.

(planar = kann ohne sich überkreuzende Kanten gezeichnet werden)

Bedeutung des 4-Farben-Satzes





Thm. 1.42

Kompaktheit – Beispielanwendung

Mittels des Kompaktheitssatzes kann man den 4-Farben-Satz von endlichen auf unendliche Graphen übertragen:

Theorem 1.43 (4-Farben-Satz, beliebige Graphen)

Jeder (möglicherweise unendliche) planare Graph ist 4-färbbar.

Beweis.

Sei $G = (V, E)$ ein (möglicherweise unendlicher) planarer Graph.

Definiere Formelmenge

$$\begin{aligned}\Gamma = &\{x_{v1} \vee x_{v2} \vee x_{v3} \vee x_{v4} \mid v \in V\} \\ &\cup \{\neg(x_{vi} \wedge x_{vj}) \mid v \in V,\ 1 \leq i < j \leq 4\} \\ &\cup \{\neg(x_{vi} \wedge x_{wi}) \mid \{v, w\} \in E,\ 1 \leq i \leq 4\}\end{aligned}$$

Behauptung: Jede endliche Teilmenge $\Delta \subseteq \Gamma$ ist erfüllbar. T1.20

Mit Kompaktheitssatz folgt: Γ ist erfüllbar.

Jede erfüllende Belegung liefert eine 4-Färbung von G. ❏

(Der Satz wurde ursprünglich direkt für beliebige Graphen bewiesen.)

Kompaktheit

Wir beweisen nun den Kompaktheitssatz.

Zur Erinnerung:

- Belegung V *erfüllt* Γ ($V \models \Gamma$), wenn $V \models \varphi$ für alle $\varphi \in \Gamma$.
- Γ ist *erfüllbar*, wenn es Belegung $V \models \Gamma$ gibt.
- Formel ψ *folgt aus* Γ ($\Gamma \models \psi$), wenn für alle $V \models \Gamma$ auch $V \models \psi$ gilt.

Theorem 1.40 (Kompaktheitssatz)

Für alle (potentiell unendlichen) Mengen $\Gamma \subseteq$ AL gilt:

Γ ist erfüllbar gdw. jede endliche Teilmenge von Γ erfüllbar ist.

T1.21

Äquivalent (und manchmal natürlicher) ist die folgende Variante:

Theorem 1.44 (Kompaktheitssatz Variante 2)

Für alle (potentiell unendlichen) Mengen $\Gamma \subseteq$ AL und Formeln $\varphi \in$ AL gilt:

$\Gamma \models \varphi$ gdw. endliches $\Delta \subseteq \Gamma$ existiert mit $\Delta \models \varphi$.