

Logik Teil 1: Aussagenlogik

Vorlesung im Wintersemester 2012/2013

Aussagenlogik

- 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 Ausdruckstärke von Aussagenlogik ist sehr **begrenzt**
- Es ergeben sich jedoch **interessante algorithmische Probleme** (z.B. das Erfüllbarkeitsproblem)

Übersicht Teil 1

- Kapitel 1.1: Grundlagen
- Kapitel 1.2: Normalformen und funktionale Vollständigkeit
- Kapitel 1.3: Erfüllbarkeit, Gültigkeit, Folgerbarkeit, Horn-Formeln
- Kapitel 1.4: Resolution
- Kapitel 1.5: Kompaktheit

Aussagenlogik

Kapitel 1.1: Grundlagen

Syntax

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

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

Definition Syntax Aussagenlogik

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

- $0, 1 \in \text{AL}$
- $\text{VAR} \subseteq \text{AL}$
- Wenn $\varphi, \psi \in \text{AL}$, 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 *Boolschen Konstanten*
- Die *atomaren* Formeln sind $\{0, 1\} \cup \text{Var}$
- Alle anderen Formeln sind *zusammengesetzt*
- Statt x_1, x_2, \dots 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 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 $((x \wedge y) \wedge z)$

Definition Semantik Aussagenlogik

Eine *Belegung* ist eine Abbildung $V : \text{Var} \rightarrow \{0, 1\}$. 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 .

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 Verknüpfungstafeln:

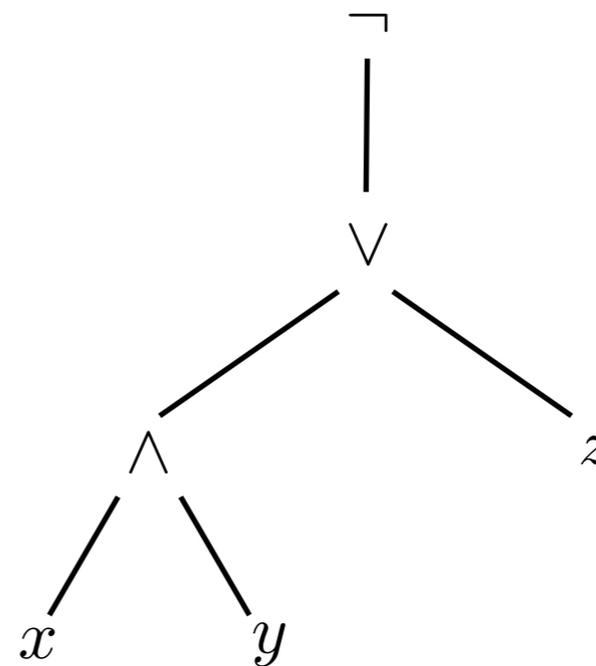
$V(\varphi)$	$V(\neg\varphi)$	$V(\varphi)$	$V(\psi)$	$V(\varphi \wedge \psi)$	$V(\varphi)$	$V(\psi)$	$V(\varphi \vee \psi)$
0	1	0	0	0	0	0	0
1	0	0	1	0	0	1	1
		1	0	0	1	0	1
		1	1	1	1	1	1

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)



Semantik

Die Semantik der Junktoren als Verknüpfungstafeln:

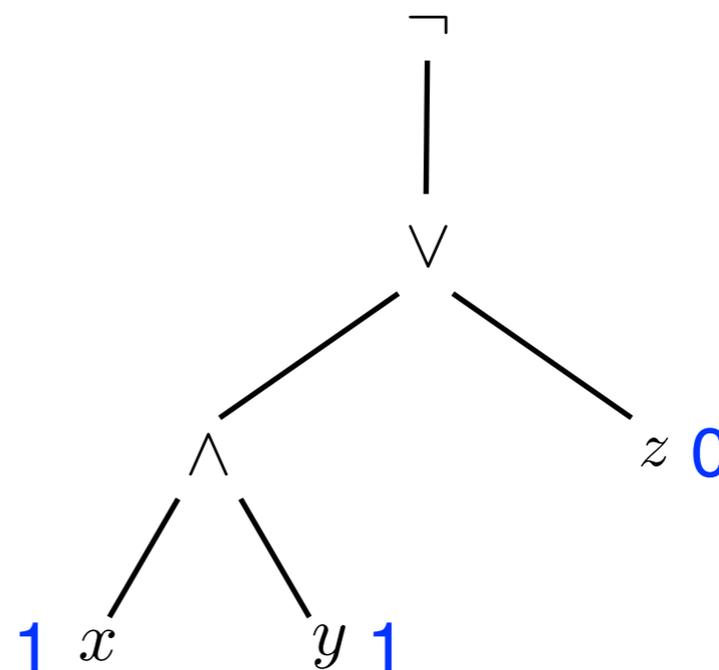
$V(\varphi)$	$V(\neg\varphi)$	
0	1	
1	0	
$V(\varphi)$	$V(\psi)$	$V(\varphi \wedge \psi)$
0	0	0
0	1	0
1	0	0
1	1	1
$V(\varphi)$	$V(\psi)$	$V(\varphi \vee \psi)$
0	0	0
0	1	1
1	0	1
1	1	1

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)



Semantik

Die Semantik der Junktoren als Verknüpfungstafeln:

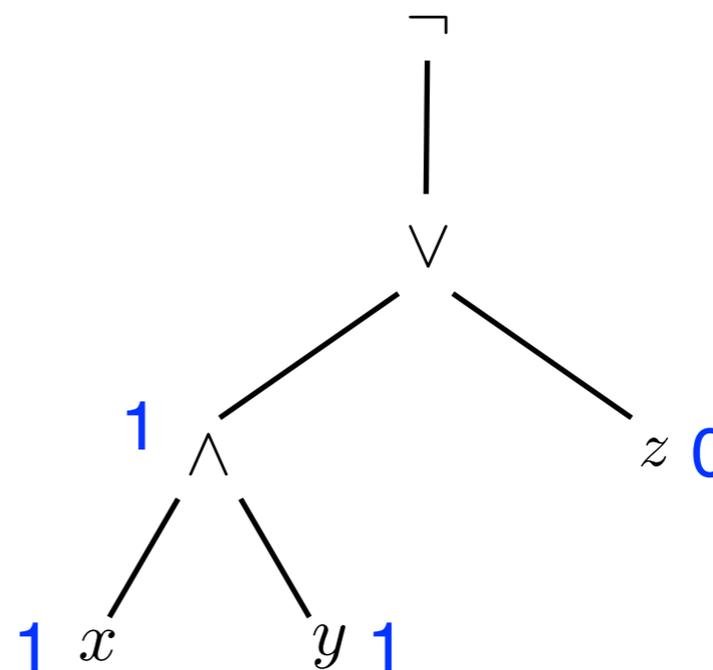
$V(\varphi)$	$V(\neg\varphi)$	$V(\varphi)$	$V(\psi)$	$V(\varphi \wedge \psi)$	$V(\varphi)$	$V(\psi)$	$V(\varphi \vee \psi)$
0	1	0	0	0	0	0	0
1	0	0	1	0	0	1	1
		1	0	0	1	0	1
		1	1	1	1	1	1

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)



Semantik

Die Semantik der Junktoren als Verknüpfungstafeln:

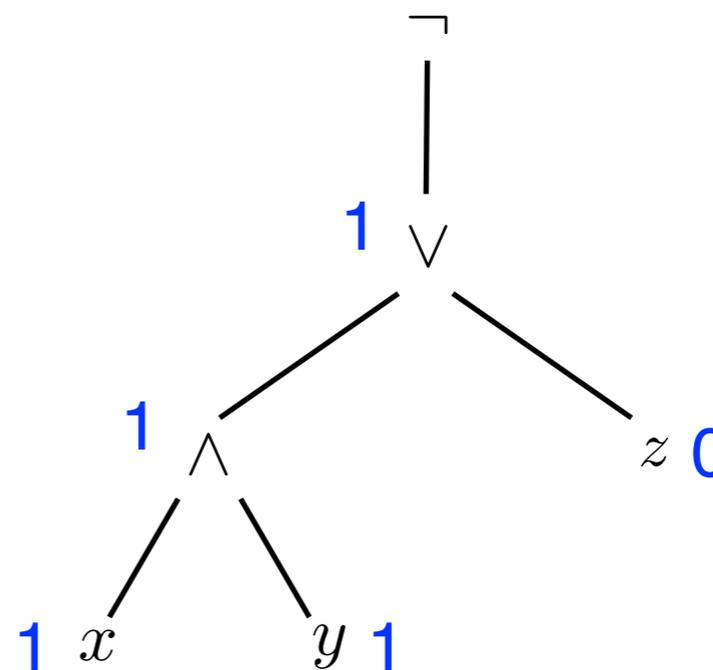
$V(\varphi)$	$V(\neg\varphi)$	$V(\varphi)$	$V(\psi)$	$V(\varphi \wedge \psi)$	$V(\varphi)$	$V(\psi)$	$V(\varphi \vee \psi)$
0	1	0	0	0	0	0	0
1	0	0	1	0	0	1	1
		1	0	0	1	0	1
		1	1	1	1	1	1

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)



Semantik

Die Semantik der Junktoren als Verknüpfungstafeln:

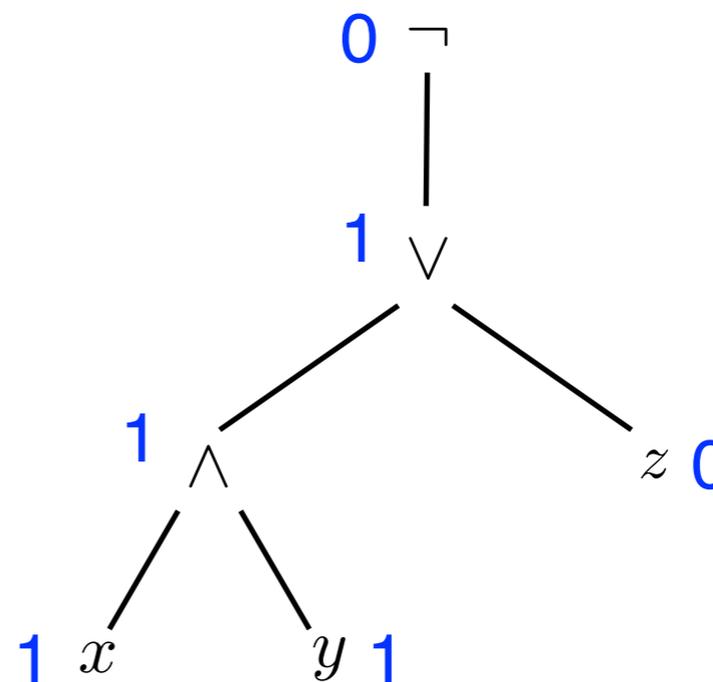
$V(\varphi)$	$V(\neg\varphi)$	$V(\varphi)$	$V(\psi)$	$V(\varphi \wedge \psi)$	$V(\varphi)$	$V(\psi)$	$V(\varphi \vee \psi)$
0	1	0	0	0	0	0	0
1	0	0	1	0	0	1	1
		1	0	0	1	0	1
		1	1	1	1	1	1

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

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

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

- Wenn $n = 0$, dann

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

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

Implikation

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

Implikation

$\varphi \rightarrow \psi$ steht für $\neg\varphi \vee \psi$

Biimplikation

$\varphi \leftrightarrow \psi$ steht für $(\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)$

$V(\varphi)$	$V(\psi)$	$V(\varphi \rightarrow \psi)$
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 \rightarrow und \leftrightarrow ,

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

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 bezeichnen diese mit $\text{Var}(\varphi)$.

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 (Übung).

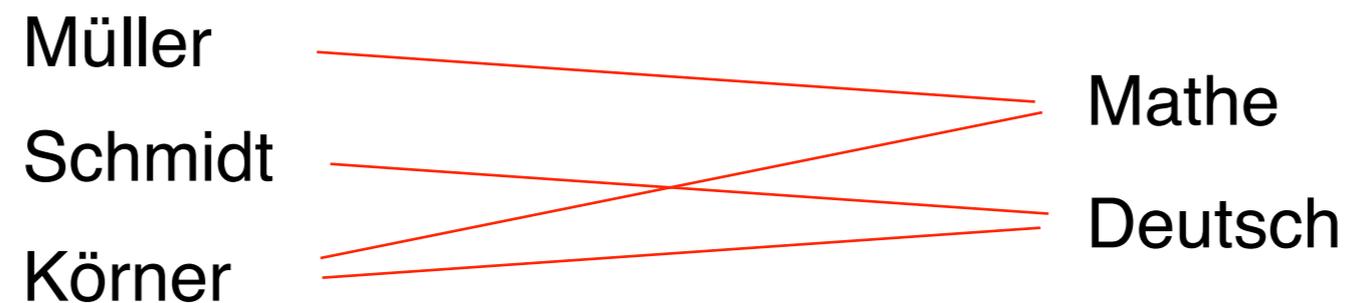
Wenn wir mit einer Formel φ arbeiten, so erlaubt uns das Koinzidenzlemma, in Belegungen nur die Variablen $\text{Var}(\varphi)$ (also endlich viele) zu betrachten.

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 Lehrer 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 jeder Lehrer mindestens 2 Stunden unterrichten



Auswertung

Definition Auswertungsproblem

Das *Auswertungsproblem der Aussagenlogik* ist:

Gegeben: Aussagenlogische Formel φ , Belegung V für φ

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

Theorem (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

Auswertung

Formale Definition der Teilformeln:

Definition Teilformeln

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

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

Also z.B.:

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

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

Äquivalenz

Definition Äquivalenz

Zwei Formeln φ and 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 :

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

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

Äquivalenz

Äquivalente Formeln sind austauschbar:

Ersetzungslemma

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

Beweis per Induktion über die Struktur von ϑ . 

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

Äquivalenz

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

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

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

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

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

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

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

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

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

de Morgansche Gesetze

Idempotenz von Konjunktion
und Disjunktion

Kommutativität von Konjunktion
und Disjunktion

Assoziativität von Konjunktion
und Disjunktion

Äquivalenz

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.:

- $\varphi \rightarrow \psi \equiv \neg\psi \rightarrow \neg\varphi$

Kontraposition

Äquivalenz

Mittels dieser Äquivalenzen und dem Ersetzungslemma 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

Äquivalenz

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

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

$$\begin{aligned} \neg(x \vee (\neg x \wedge y)) &\equiv \neg x \wedge \neg(\neg x \wedge y) && \text{de Morgan} \\ &\equiv \neg x \wedge (\neg\neg x \vee \neg y) && \text{de Morgan + EL} \end{aligned}$$

Äquivalenz

Mittels dieser Äquivalenzen und dem Ersetzungslemma 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

Äquivalenz

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

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

$$\begin{aligned}\neg(x \vee (\neg x \wedge y)) &\equiv \neg x \wedge \neg(\neg x \wedge y) && \text{de Morgan} \\ &\equiv \neg x \wedge (\neg\neg x \vee \neg y) && \text{de Morgan + EL} \\ &\equiv \neg x \wedge (x \vee \neg y) && \text{doppelte Negation + EL} \\ &\equiv (\neg x \wedge x) \vee (\neg x \wedge \neg y) && \text{Distributivgesetz}\end{aligned}$$

Äquivalenz

Mittels dieser Äquivalenzen und dem Ersetzungslemma 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

Äquivalenz

Mittels dieser Äquivalenzen und dem Ersetzungslemma 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

Äquivalenz

Mittels dieser Äquivalenzen und dem Ersetzungslemma 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

Kapitel 1.2: Normalformen und funktionale Vollständigkeit

Boolsche Funktionen

Definition Boolsche Funktion

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

Für $n \geq 0$ bezeichne

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

Boolsche Funktionen

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

- O.B.d.A sei $\text{Var} = \{x_1, \dots, x_n\}$
- Für jedes Tupel $t = (w_1, \dots, w_n) \in \{0, 1\}^n$ sei V_t die Belegung für φ , die definiert ist durch $V_t(x_i) = w_i$, für $1 \leq i \leq n$
- Definiere $f_\varphi(t) = V_t(\varphi)$ für alle $t \in \{0, 1\}^n$

Umgekehrt findet sich zu jeder Booleschen Funktion auch eine Formel:

Theorem (Funktionale Vollständigkeit)

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



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 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:

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

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

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

Normalformen

Theorem (KNF/DNF Umwandlung)

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



Normalformen

Theorem (KNF/DNF Umwandlung)

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

Beispiel:

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



Normalformen

Theorem (KNF/DNF Umwandlung)

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

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

Normalformen

Theorem (KNF/DNF Umwandlung)

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

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:

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

Normalformen

Theorem (KNF/DNF Umwandlung)

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

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:

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

\equiv

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

Normalformen

Beachte:

- Sowohl DNF als auch KNF werden im worst case exponentiell groß (2^n viele Disjunkte / Konjunkte, wobei $n = |\text{Var}(\varphi)|$)
- 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 der Größe $\mathcal{O}(n^3)$ 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)

Normalformen

Idee Paritätsfunktion der Größe $O(n)$: Divide and Conquer

(Parität von $x_0x_1x_2x_3$ ergibt sich aus der von x_0x_1 und x_2x_3 mittels Parität!)

- Für 2 Variablen: $\varphi_{01} = (\neg x_0 \wedge x_1) \vee (x_0 \wedge \neg x_1)$
- Für 4 Variablen: $\varphi_{0123} = (\neg \varphi_{01} \wedge \varphi_{23}) \vee (\varphi_{01} \wedge \neg \varphi_{23})$
- etc

wie φ_{01} , aber mit Variablen x_2, x_3

Für $n = 2^m$ Variablen ergibt sich Syntaxbaum mit

- Tiefe $3m$ und Verzweigung höchstens 2
- also mit Größe $\leq O(2^{3m}) = O(n^3)$

Kleine Modifikationen wenn Anzahl Variablen keine Zweierpotenz

Funktionale Vollständigkeit

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 Junktormengen sind in diesem Sinne vollständig?

Funktionale Vollständigkeit

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 $t = (w_1, \dots, w_n) \in \{0, 1\}^n$ in Wahrheitstafel hat Wert $f(t)$.

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

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

Funktionale Vollständigkeit

Definition 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:

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

$$\{\neg, \wedge\}$$

$$\{\neg, \vee\}$$

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

Weitere funktional vollständige Menge:

- $\{\mid\}$

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

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

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

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

Kapitel 1.3: Erfüllbarkeit, Gültigkeit, Folgerbarkeit, Horn-Formeln

Erfüllbarkeit, Gültigkeit

Definition 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:

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

Beispiele für gültige Formeln:

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

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

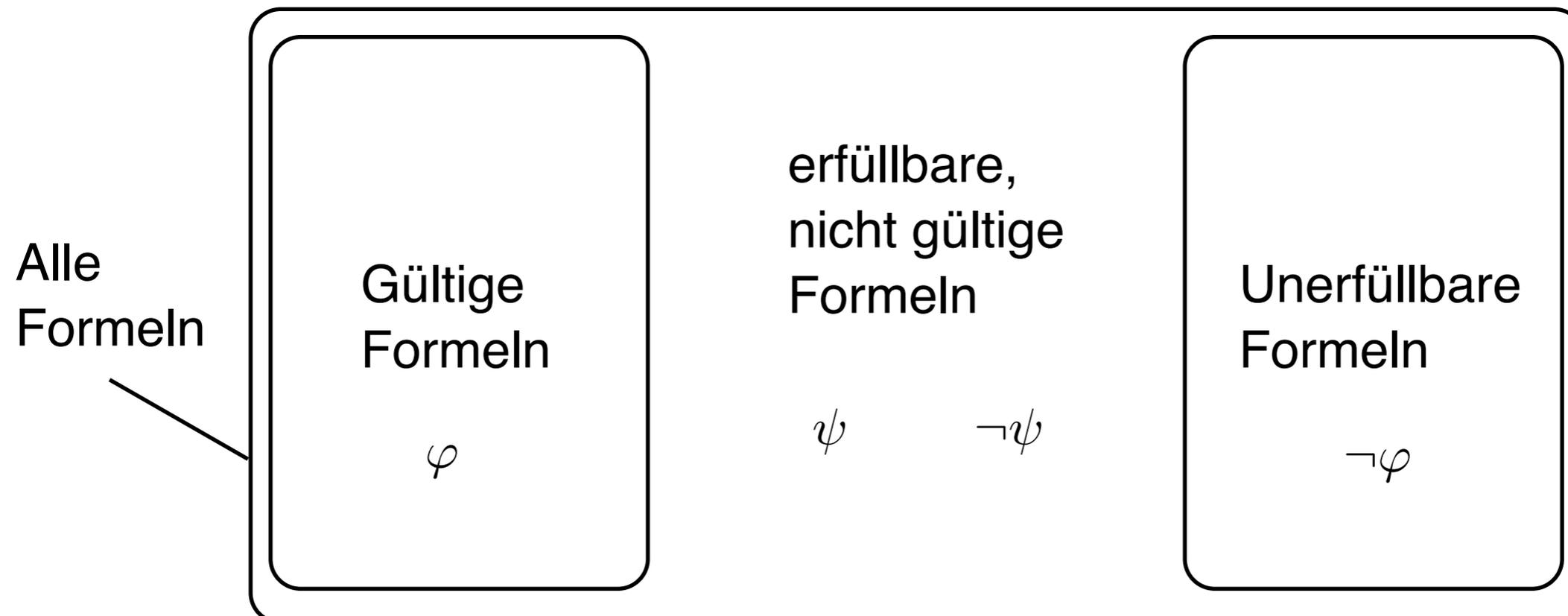
Erfüllbarkeit, Gültigkeit

Folgt direkt aus Definition Erfüllbarkeit/Tautologie + Semantik Negation:

Lemma (Dualität Erfüllbarkeit, Gültigkeit)

Eine Formel φ ist

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



Erfüllbarkeit, Gültigkeit

Definition 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 Gültigkeit:

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 Polynomialzeit reduzierbar mit vorigem Lemma (und umgekehrt).

Erfüllbarkeit, Gültigkeit

Theorem (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 co-NP-vollständig.

Dies gilt auch für Formeln in DNF, sogar bei max. 3 Literalen pro Disjunkt

Für Formeln in KNF ist Gültigkeit leicht (in Linearzeit) zu entscheiden, ebenso Erfüllbarkeit für Formeln in DNF.

Lemma (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 Folgerbarkeit, Konsequenz

Eine Formel φ ist *folgerbar* aus einer Formel ψ gdw. 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 $\varphi \models \psi$.

Beispiele:

$$x \wedge y \models x$$

$$x \models x \vee y$$

$$\varphi \wedge (\varphi \rightarrow \psi) \models \psi \quad (\text{Modus Ponens})$$

Offensichtlich:

$$\varphi \equiv \psi \text{ gdw. } \varphi \models \psi \text{ und } \psi \models \varphi$$

Für eine (potentiell unendliche) Formelmengemenge Γ ist $\Gamma \models \psi$ in der offensichtlichen Weise definiert.

$$\text{Beispiel: } \{\varphi, \varphi \rightarrow \psi\} \models \psi \quad (\text{Modus Ponens})$$

Folgerbarkeit

Theorem (Folgerbarkeit und Gültigkeit)

Für alle Formeln φ, ψ gilt:

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

Analog zu Erfüllbarkeits/Gültigkeitsproblem kann man ein *Folgerbarkeitsproblem* definieren

Das Lemma liefert auch wechselseitige Polynomialzeit-Reduktionen zwischen Gültigkeitsproblem und Folgerbarkeitsproblem

Das Folgerbarkeitsproblem hat also dieselbe Komplexität wie das Gültigkeitsproblem (co-NP-vollständig)

Horn Formeln

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

Definition Horn-Formel

Eine *aussagenlogische Horn-Formel* ist eine Formel $\varphi = \bigwedge_i \bigvee_j \ell_{i,j}$ in KNF, 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:

Negative Literale + 1 positives Literal

Nur ein positives Literal

Nur negative Literale

(Gar keine Literale
 $\equiv 0$, daher uninteressant)

Horn Formeln

Anschaulicher:

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

Beispiel Horn-Formel: Konjunktion von

$$\begin{array}{llll} \text{Regen} & & \text{Schnee} & \\ \text{Regen} \rightarrow \text{Niederschlag} & & \text{Schnee} \rightarrow \text{Niederschlag} & \\ \text{Regen} \rightarrow \text{Temp} \geq 0 & & \text{Schnee} \rightarrow \text{Temp} < 0 & \\ \text{Temp} > 0 \wedge \text{Temp} \leq 0 \rightarrow 0 & & & \end{array}$$

Hierbei sind “Regen”, “Schnee”, “Temp > 0”, etc Aussagenvariablen

Horn Formeln

Theorem (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 $x_1 \wedge \dots \wedge x_k \rightarrow x$ mit $\{x_1, \dots, x_k\} \subseteq V$ und $x \notin V$ do

$V := V \cup \{x\}$

done

if es gibt ein Konjunkt $x_1 \wedge \dots \wedge x_k \rightarrow 0$ mit $\{x_1, \dots, x_k\} \subseteq V$ then

 return “unerfüllbar”

else

 return “erfüllbar”

Beispiel ●

Horn Formeln

Polyzeit-Algorithmus für Eingabe φ :

$V := \{x \in \text{Var} \mid x \text{ ist Konjunkt von } \varphi\}$

while es gibt Konjunkt $x_1 \wedge \dots \wedge x_k \rightarrow x$ mit $\{x_1, \dots, x_k\} \subseteq V$ und $x \notin V$ do

$V := V \cup \{x\}$

done

if es gibt ein Konjunkt $x_1 \wedge \dots \wedge x_k \rightarrow 0$ mit $\{x_1, \dots, x_k\} \subseteq V$ then

return “unerfüllbar”

else

return “erfüllbar”

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

Lemma

Der Algorithmus ist korrekt und läuft in polynomieller Zeit.

Horn Formeln

Ausdrucksstärke von Horn-Formeln:

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

Ausdrückbar z.B.: $x \vee y \rightarrow z \equiv (x \rightarrow z) \wedge (y \rightarrow z)$

$x \rightarrow y \wedge z \equiv (x \rightarrow y) \wedge (x \rightarrow z)$

Nicht ausdrückbar z.B. $x \vee y$

(Beispiel: wir können ausdrücken, dass $\text{Temp} < 0 \wedge \text{Temp} \geq 0 \rightarrow 0$,
nicht aber, dass $\text{Temp} < 0 \vee \text{Temp} \geq 0$)

Lemma (Nicht-Horn-Ausdrückbarkeit)

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



Aussagenlogik

Kapitel 1.4: Resolution

Resolution

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

Es gibt viele verschiedene Arten von Kalkülen, z.B.

Sequenzkalkül, 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 ein wichtiges und elegantes Kalkül für Unerfüllbarkeit in Aussagenlogik: *Resolution*

Resolution

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

Definition Klausel, Klauselmenge

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

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

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

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 $M = \{\{x_1, \neg x_2\}, \{x_3\}\}$.

Resolution

Umgekehrt entspricht eine Klausel C der Formel $\bigvee_{\ell \in C} \ell$ und eine endliche Klauselmengemenge 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 Klauselmengemenge, die \square enthält, ist unerfüllbar
- die leere Klauselmengemenge entspricht der ‘leeren Konjunktion’ und ist erfüllbar

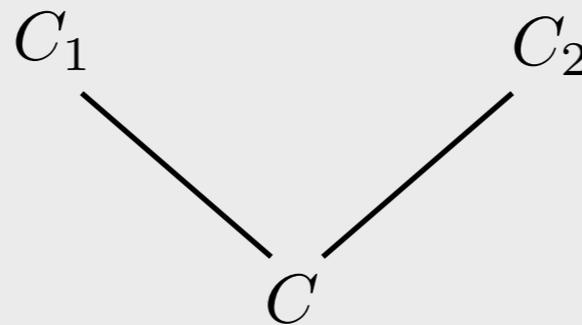
Resolution

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

Definition Resolvente

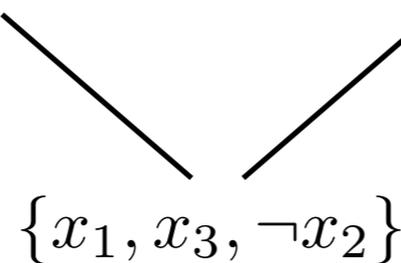
Seien C_1, C_2 Klauseln. Klausel C ist *Resolvente* von C_1 und C_2 gdw. es Literal l gibt mit $l \in C_1, \bar{l} \in C_2$ und $C = (C_1 \setminus \{l\}) \cup (C_2 \setminus \{\bar{l}\})$.

Wir schreiben dann

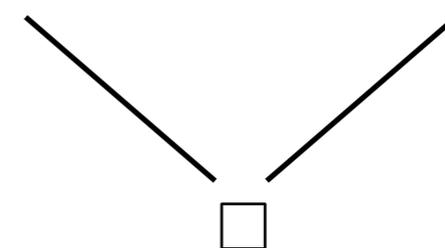


Beispiele:

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



$\{x_1\}$ $\{\neg x_1\}$



Resolution

Lemma (Resolutionslemma)

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

Folgende Notation beschreibt das wiederholte Bilden von Resolventen

Definition Res

Für jede Klauselmengende M sei

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

Beispiel:

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

Resolution

Im allgemeinen:

- Ein Kalkül heißt *korrekt*, wenn sich nur gewünschte Formeln erzeugen lassen
- Ein Kalkül heißt *vollständig*, wenn sich jede gewünschte Formel erzeugen läßt

In diesem Fall: \square gewünscht gdw. Eingabeformel unerfüllbar

Der folgende Satz etabliert Korrektheit und Vollständigkeit der Resolution

Theorem (Resolutionssatz, Robinson 1965)

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

Der Satz kann auch für unendliche Klauselmengen bewiesen werden.

Resolution

Alle generierten Klauseln enthalten nur Literale, die schon in M vorkommen

Da es nur 2^n Klauseln über diesen Literalen gibt (mit n Anzahl Literale in M), stabilisiert sich die Folge

$$M = \text{Res}^0(M) \subseteq \text{Res}^1(M) \subseteq \dots$$

nach höchstens 2^n Schritten.

Dies liefert den folgenden Algorithmus für Erfüllbarkeit in der Aussagenlogik:

$$R_0 := \emptyset \quad R_1 := M$$

$$i := 1$$

while $R_{i-1} \subsetneq R_i$ do

$$i := i + 1$$

$$R_i := \text{Res}(R_{i-1})$$

if $\square \in R_i$ then return 'unerfüllbar'

done

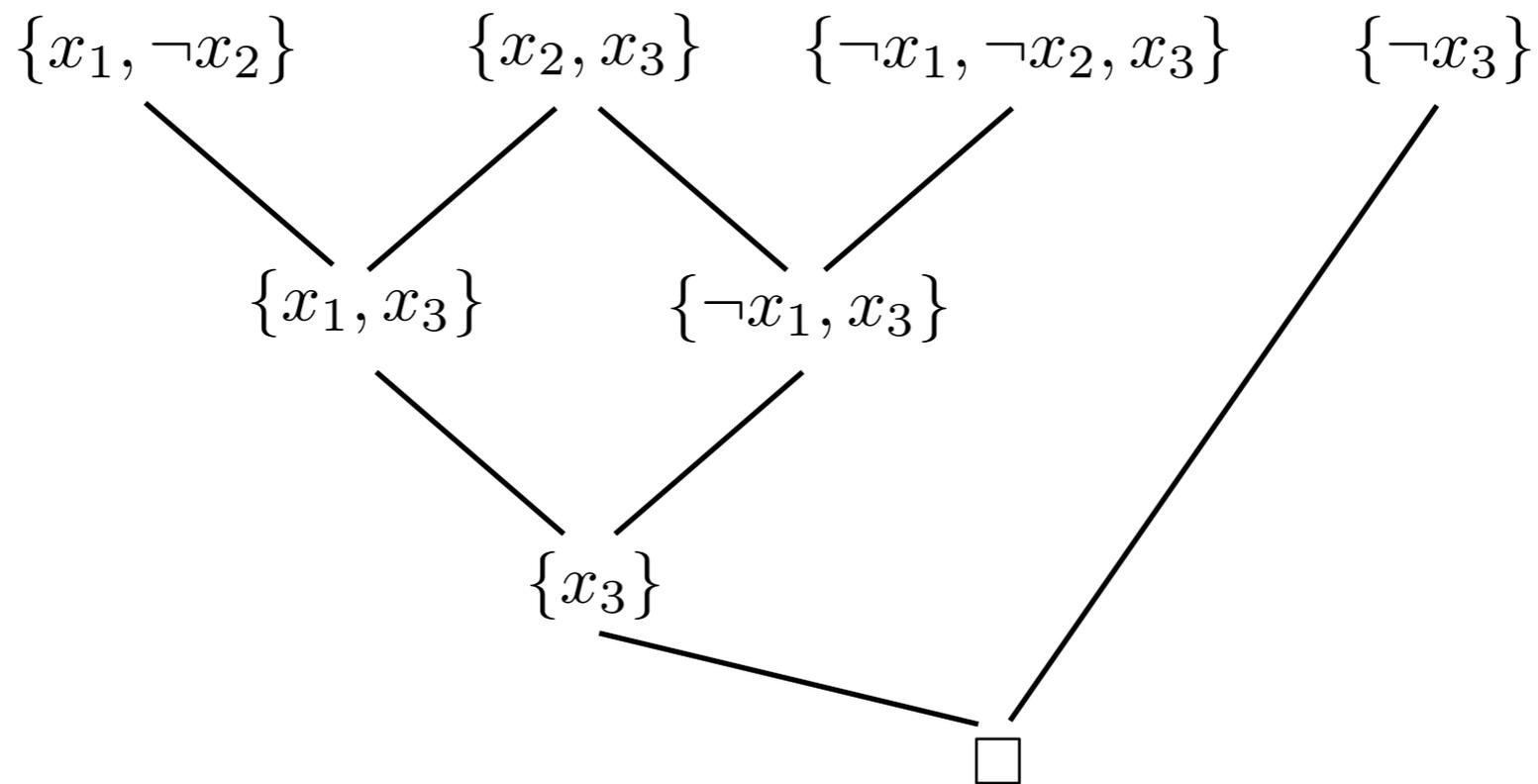
return 'erfüllbar'

Resolutionsbeweise

Resolutionsbeweis:

Darstellung der Ableitung von \square mittels Resolventen als Graph

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



Im allgemeinen können Resolutionsbeweise exponentiell lang werden
(nicht überraschend angesichts der NP-Vollständigkeit von Erfüllbarkeit)

Resolutionsbeweise

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)

Für die meisten Klauselmengen ist ein Resolutionsbeweis sehr viel übersichtlicher als die Menge $\text{Res}^*(M)$

Einheitsresolution

Definition Hornklausel

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

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

Definition 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 und $|C_1| = 1$ oder $|C_2| = 1$.

Wir setzen

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

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

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



Einheitsresolution

Auf Hornklauseln ist Einheitsresolution ausreichend:

Theorem (Resolutionssatz für Einheitsresolution)

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

Man kann zeigen: für jede unerfüllbare Menge von Hornklauseln gibt es einen Resolutionsbeweis von polynomieller Größe

Einheitsresolution

Auf Hornklauseln ist Einheitsresolution ausreichend:

Theorem (Resolutionssatz für Einheitsresolution)

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

Man kann zeigen: für jede unerfüllbare Menge von Hornklauseln gibt es einen Resolutionsbeweis von polynomieller Größe

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

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

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 sehr viele NP-vollständige Probleme in sehr natürlicher Weise als KNF kodieren lassen

Moderne SAT-Solver verwenden die sogenannte DPLL-Methode (nach Davis-Putnam-Logemann-Loveland), die man als spezielle Form von Resolution ansehen kann.

Wirklich effizient werden SAT-Solver aber erst durch zahlreiche raffinierte (und teils mathematisch recht anspruchsvolle) Optimierungen (Minisat, zchaff, precosat, siehe SAT competitions)

Hilbert-Kalkül

Wir betrachten noch kurz ein weiteres Beispiel für ein Kalkül

Das Hilbert-Kalkül verwendet Formeln über der Junktormenge $\{\rightarrow, \neg\}$, 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 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 $\varphi \rightarrow \psi$ herleitbar, dann ψ herleitbar
(Modus Ponens)

Beispiel: die Formel $x \rightarrow x$ ist herleitbar



Ohne Beweis:

Theorem (Korrektheit + Vollständigkeit Hilbert-Kalkül)

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

Resolutionskalkül vs Hilbert-Kalkül

Resolutionskalkül	Hilbert-Kalkül
zeigt Unerfüllbarkeit gegebener Formel	erzeugt alle gültigen Formeln
Formeln in KNF	Formeln über Junktormenge $\{\rightarrow, \neg\}$
Herleitung der leeren Klausel mittels Resolventenbildung	Herleitung neuer Formeln aus Axiomen mittels Modus Ponens
Vollständigkeitsbeweis relativ einfach	Vollständigkeitsbeweis relativ aufwändig
Anwendung: automatisches Entscheiden von Erfüllbarkeit	Anwendung: Modellierung mathematisches Schliessen

Kapitel 1.5: Kompaktheit

Kompaktheit

Verschiedene natürliche Probleme aus der Informatik können als *unendliche Mengen aussagenlogischer Formeln* repräsentiert werden

Ein zentrales Resultat zum Verständnis unendlicher Formelmengen ist der folgende Kompaktheitsatz

Theorem (Kompaktheitssatz)

Für alle $\Gamma \subseteq \text{AL}$ und $\varphi \in \text{AL}$ gilt:

1. Γ ist erfüllbar gdw. jede endliche Teilmenge von Γ erfüllbar ist
2. $\Gamma \models \varphi$ gdw. endliches $\Delta \subseteq \Gamma$ existiert mit $\Delta \models \varphi$

Es ist leicht zu sehen, dass Punkt 1+2 äquivalent sind.

Kompaktheit - Beispielanwendung

Definition 4-färbbar

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 $f : V \rightarrow \{c_1, c_2, c_3, c_4\}$ 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 (4-Farben-Satz, endliche Graphen) ohne Beweis

Jeder endliche planare Graph ist 4-färbbar.

(planar = kann ohne sich überkreuzende Kanten gezeichnet werden)

Kompaktheit - Beispielanwendung

Mittels des Kompaktheitssatzes kann man den Satz auf unendliche Graphen übertragen:

Theorem (4-Farben-Satz)

Jeder (endliche oder unendliche) planare Graph ist 4-färbbar.

(Der Satz wurde aber auch ursprünglich direkt für beliebige Graphen bewiesen)

