

# Logik

WiSe 2018/19  
Thomas Schneider

## Teil 1: Aussagenlogik

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

### Übersicht Teil 1

**NEXT** →

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

### 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**).

### 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 1.1 (Aussagenlogik, Syntax)

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:

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

## Sprechweisen und Konventionen

- $\neg\varphi$  sprechen wir „nicht  $\varphi$ “ (Negation),  
 $(\varphi \vee \psi)$  sprechen wir „ $\varphi$  oder  $\psi$ “ (Disjunktion),  
 $(\varphi \wedge \psi)$  sprechen wir „ $\varphi$  und  $\psi$ “ (Konjunktion)
- 1 steht für „wahr“, 0 für „falsch“, 0,1 sind die *Booleschen 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 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  $V : \text{VAR} \rightarrow \{0, 1\}$ . Sie definiert einen Wahrheitswert  $V(\varphi)$  für jede Formel  $\varphi$ :

- $V(0) = 0$  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  $\varphi$  von  $V$  *erfüllt* wird.

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

## 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  $\neg(\neg x_1 \wedge x_2) \vee x_3$ .

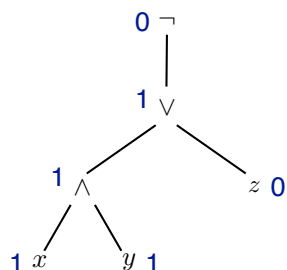
## Semantik

Die Semantik der Junktoren als **Wahrheitstafeln**:

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

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

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

- 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  $(\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)$

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

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

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

## Koinzidenzlemma

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

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

### Lemma 1.3 (Koinzidenzlemma)

Sei  $\varphi$  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  $\varphi$ .

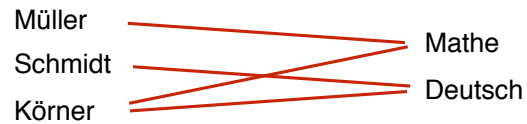
Wenn wir mit einer Formel  $\varphi$  arbeiten, so erlaubt uns das Koinzidenzlemma, in Belegungen nur die Variablen  $\text{Var}(\varphi)$  (also endlich viele) zu betrachten.

Eine *Belegung für  $\varphi$*  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  $\varphi$ , Belegung  $V$  für  $\varphi$

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  $\varphi$  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](https://doi.org/10.1145/28395.28409)

## Auswertung

Formale Definition der Teilformeln:

### Definition 1.6 (Teilformeln)

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

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

Also z. B.:

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

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

## Äquivalenz

### Definition 1.7 (Äquivalenz)

Zwei Formeln  $\varphi$  und  $\psi$  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*  $\varphi$ :

$V(x)$	$V(y)$	$V(x \wedge y)$	$V(x)$	$V(y)$	$V(\neg(\neg x \vee \neg y))$
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: Ersetzungslemma

Äquivalente Formeln sind austauschbar:

### Lemma 1.8 (Ersetzungslemma)

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

T1.2

Beweis per Induktion über die Struktur von  $\vartheta$ .

Allgemeines Vorgehen beim Induktionsbeweis:

Induktionsanfang:

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

Induktionsschritt: Zeige:

Wenn die Aussage für  $\vartheta_1$  und  $\vartheta_2$  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  $\varphi$  gilt:  $\varphi \equiv \neg\neg\varphi$       Eliminieren doppelter Negation

Beweis per Wahrheitstafel

## Nützliche Äquivalenzen

Folgende Äquivalenzen gelten für alle aussagenlogischen Formeln  $\varphi, \psi, \vartheta$ :

- $\neg(\varphi \wedge \psi) \equiv \neg\varphi \vee \neg\psi$       De Morgansche Gesetze
- $\neg(\varphi \vee \psi) \equiv \neg\varphi \wedge \neg\psi$
- $\varphi \wedge \varphi \equiv \varphi$       Idempotenz von Konjunktion
- $\varphi \vee \varphi \equiv \varphi$       und Disjunktion
- $\varphi \wedge \psi \equiv \psi \wedge \varphi$       Kommutativität von Konjunktion
- $\varphi \vee \psi \equiv \psi \vee \varphi$       und Disjunktion
- $\varphi \wedge (\psi \wedge \vartheta) \equiv (\varphi \wedge \psi) \wedge \vartheta$       Assoziativität von Konjunktion
- $\varphi \vee (\psi \vee \vartheta) \equiv (\varphi \vee \psi) \vee \vartheta$       und Disjunktion

## Nützliche Äquivalenzen

Mehr nützliche Äquivalenzen:

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

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

- $\varphi \rightarrow \psi \equiv \neg\psi \rightarrow \neg\varphi$       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  $f : \{0, 1\}^n \rightarrow \{0, 1\}$ .

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  $\varphi$  mit  $|\text{Var}(\varphi)| = n$  berechnet  $n$ -stellige Boolesche Funktion  $f_\varphi$ :

- O. B. d. A. sei  $\text{Var}(\varphi) = \{x_1, \dots, x_n\}$
- Belegung  $V$  für  $\varphi$  entspricht Eingabe für  $f_\varphi$ :  
 $i$ -ter Eingabewert ist  $V(x_i)$
- Wert von  $f_\varphi$  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  $\varphi$  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

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

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  $\varphi$  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  $\varphi$  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.

Beispiel:

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

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:

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

≡

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

T1.6

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

### 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 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  $f_0, f_1, f_2, \dots$  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  $\varphi$  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  $t = (w_1, \dots, w_n) \in \{0, 1\}^n$  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

	$V(\varphi)$	$V(\psi)$	$V(\varphi   \psi)$
	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  $\varphi$  mit Junktoren aus  $\Omega$  gibt, so dass  $f_\varphi = f$ .

Wir wissen bereits, dass folgende Mengen funktional vollständig sind:

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

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:

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

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

T1.8

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

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

Beweis per Induktion über die Struktur von  $\varphi$ :

- 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



- 1.1 Grundlagen
- 1.2 Normalformen und funktionale Vollständigkeit
- NEXT** → 1.3 Erfüllbarkeit, Gültigkeit, Folgerbarkeit, Horn-Formeln
- 1.4 Resolution
- 1.5 Kompaktheit

## 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:

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

Beispiele für gültige Formeln:

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

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

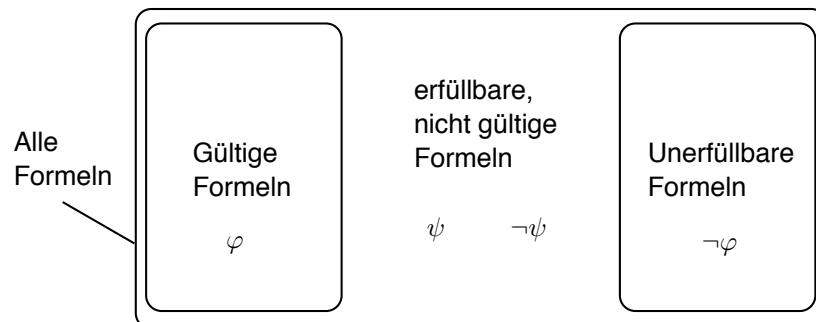
# Dualität

Folgt direkt aus Definition Erfüllbarkeit/Tautologie + Semantik Negation:

## Lemma 1.15 (Dualität Erfüllbarkeit, Gültigkeit)

Eine Formel  $\varphi$  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  $\varphi$

Frage: Ist  $\varphi$  erfüllbar?

Das *Gültigkeitsproblem* der Aussagenlogik ist:

Gegeben: Aussagenlogische Formel  $\varphi$

Frage: Ist  $\varphi$  eine Tautologie?

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

Zähle alle  $2^n$  Belegungen für  $\varphi$  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  $\psi$  ist *folgerbar* aus einer Formel  $\varphi$ ,  
wenn für alle Belegungen  $V$  mit  $V \models \varphi$  auch gilt, dass  $V \models \psi$ .

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

Für eine (potentiell unendliche) Formelmeng  $\Gamma$  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:

$$x \wedge y \models x$$

$$x \models x \vee y$$

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

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

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  $\varphi, \psi$  gilt:

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

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

Beispiel Horn-Formel: Konjunktion von

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

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  $\varphi$ :

```

V := {x ∈ VAR | x ist Konjunkt von φ}
while es gibt Konjunkt x1 ∧ ... ∧ xk → x mit {x1, ..., xk} ⊆ V und x ∉ V do
  V := V ∪ {x}
done
if es gibt ein Konjunkt x1 ∧ ... ∧ xk → 0 mit {x1, ..., xk} ⊆ 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  $\varphi$ :

```

V := {x ∈ VAR | x ist Konjunkt von φ}
while es gibt Konjunkt x1 ∧ ... ∧ xk → x mit {x1, ..., xk} ⊆ V und x ∉ V do
  V := V ∪ {x}
done
if es gibt ein Konjunkt x1 ∧ ... ∧ xk → 0 mit {x1, ..., xk} ⊆ 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 1.23

Der Algorithmus ist korrekt und läuft in polynomieller Zeit.

**T1.11**

## Minimale Modelle

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

1.  $V$  ist Modell von  $\varphi$
2. Wenn  $\hat{V}$  Modell von  $\varphi$ , dann  $V \subseteq \hat{V}$

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

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

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 1.25 (Nicht-Horn-Ausdrückbarkeit)

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

T1.12

Intuitiv: Horn-Formeln sind der disjunktionfreie 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
- NEXT** → 1.4 Resolution
- 1.5 Kompaktheit

## Resolution

Ein *Kalkül* besteht aus einer Sammlung rein *syntaktischer Umformungsregeln*, 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  $\varphi = \bigwedge_{i=1..n} \bigvee_{j=1..m_i} l_{ij}$  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  $M = \{\{x_1, \neg x_2\}, \{x_3\}\}$ .

## 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

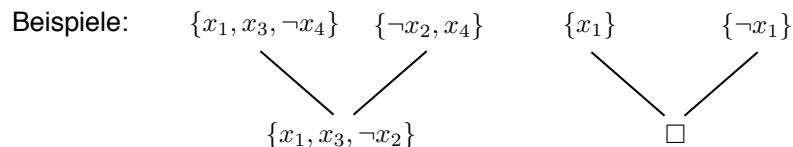
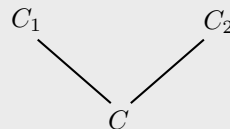
## Resolvente

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

### Definition 1.27 (Resolvente)

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

Wir schreiben dann:



## 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

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

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

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 Klauselmeng  $M$  gibt es ein  $i \geq 1$  mit  $\text{Res}^*(M) = \text{Res}^i(M)$ .

### Beweis:

$M$  hat nur endlich viele Literale.

Darüber lassen sich nur endlich viele Klauseln bilden.

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

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

### Korrektheit und Vollständigkeit:

#### Theorem 1.31 (Resolutionssatz, Robinson 1965)

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

Der Satz kann auch für **unendliche** Klauselmengen bewiesen werden.

T1.15

## Resolution als Algorithmus

Wegen Lemma 1.30 (Terminierung) 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:

```
R0 := M
i := 0
repeat
  i := i + 1
  Ri := Res(Ri-1)
  if □ ∈ Ri then return „unerfüllbar“
until Ri = Ri-1
return „erfüllbar“
```

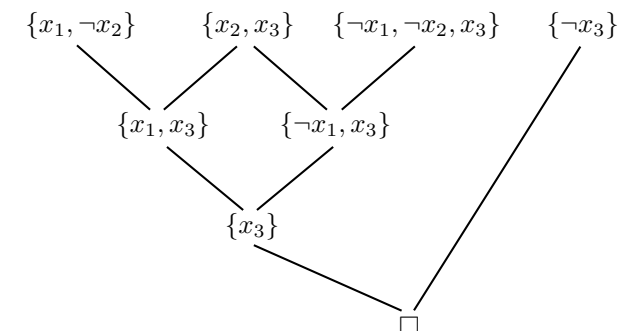
## Resolutionsbeweise

### Resolutionsbeweis:

Darstellung der Ableitung von □ 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 Klauselmeng. Ein *Resolutionsbeweis* für  $M$  ist eine Folge  $C_1, \dots, C_m$  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  $j, k < i$ .
- $C_m = \square$ .

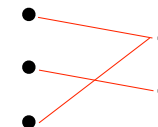
Beachte:

$\text{Res}^*(M)$  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 Resolutionsbeweis** 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  $\varphi_1, \varphi_2, \varphi_3, \dots$ . Ohne Beweis:

### Theorem 1.33 (Haken 1985)

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

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

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

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

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

T1.16

## Einheitsresolution

Auf Hornklauseln ist Einheitsresolution ausreichend:

### Theorem 1.36 (Resolutionssatz für Einheitsresolution)

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

T1.17

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

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

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

Man kann ihn aber durch eine weitere Einschränkung (Variablenordnung) 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  $\ell$ , 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  $\ell$

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  
↪ Lösche alle Klauseln, die  $\ell$  enthalten (*Unit Subsumption*)  
↪ Lösche  $\neg\ell$  aus allen übrigen Klauseln (der eigentl. Resolutionsschritt!)

*Pure Literal Elimination*

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

Optimierungen werden *am Anfang* jedes Unteraufrufs angewendet



## Der DPLL-Algorithmus

```

Function DPLL( $M$ ):
  input : Klauselmenge  $M$ 
  output: Wahrheitswert ( $\text{true} \hat{=}$  „erfüllbar“,  $\text{false} \hat{=}$  „unerfüllbar“)

  while  $M$  enthält Einheitsklausel  $\{\ell\}$  do
    Lösche alle Klauseln aus  $M$ , die  $\ell$  enthalten //Unit Subs.
    Lösche  $\neg\ell$  aus allen übrigen Klauseln //Unit Res.
  if  $\square \in M$  then return false
  while  $M$  enthält pures Literal  $\ell$  do
    Lösche alle Klauseln aus  $M$ , die  $\ell$  enthalten //Pure Lit Elim.
  if  $M = \emptyset$  then return true
  Wähle nichtdeterministisch  $\ell$  in  $M$  //nichtdet. Verzweigung
  if DPLL( $M \cup \{\{\ell\}\}$ ) then return true
  else if DPLL( $M \cup \{\{-\ell\}\}$ ) 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  $\varphi, \psi, \vartheta$  beliebig ersetzen)
- wenn  $\varphi$  herleitbar und  $\varphi \rightarrow \psi$  herleitbar, dann  $\psi$  herleitbar (*Modus Ponens*)

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

**T1.19**

Ohne Beweis:

### Theorem 1.38 (Korrektheit & Vollständigkeit Hilbert-Kalkül)

Eine Formel  $\varphi$  ist gültig gdw. sie im Hilbert-Kalkül herleitbar ist.

## Resolutionskalkül vs. Hilbert-Kalkül

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

- 1.1 Grundlagen
- 1.2 Normalformen und funktionale Vollständigkeit
- 1.3 Erfüllbarkeit, Gültigkeit, Folgerbarkeit, Horn-Formeln
- 1.4 Resolution
- NEXT** → 1.5 Kompaktheit

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

## Definition 1.39 (Semantik Formelmengen)

Sei  $\Gamma$  eine (endliche oder unendliche) Formelmenge.

- Belegung  $V$  *erfüllt*  $\Gamma$  ( $V \models \Gamma$ ), wenn  $V \models \varphi$  für alle  $\varphi \in \Gamma$ .
- $\Gamma$  ist *erfüllbar*, wenn es Belegung  $V \models \Gamma$  gibt.
- Formel  $\psi$  *folgt aus*  $\Gamma$  ( $\Gamma \models \psi$ ), 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  $\Gamma \subseteq \text{AL}$  gilt:

$\Gamma$  ist erfüllbar gdw. jede endliche Teilmenge von  $\Gamma$  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  $f : V \rightarrow \{c_1, c_2, c_3, c_4\}$  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 1.42 (4-Farben-Satz, endliche Graphen)

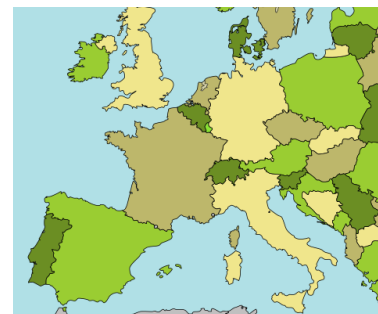
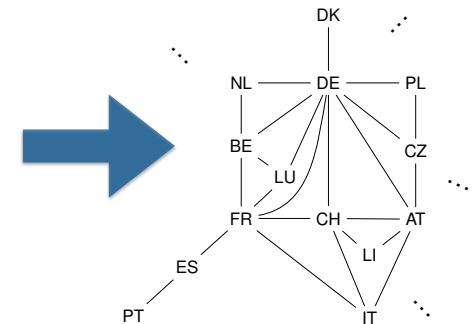
Jeder *endliche* planare Graph ist 4-färbbar.

(planar = kann ohne sich überkreuzende Kanten gezeichnet werden)

# Bedeutung des 4-Farben-Satzes



CC BY-SA 3.0 Wikimedia, Alexrk2



CC BY-SA 2.5 Wikimedia, User:michael

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

### Theorem 1.40 (Kompaktheitssatz)

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

$\Gamma$  ist erfüllbar gdw. jede endliche Teilmenge von  $\Gamma$  erfüllbar ist.

T1.21

Äquivalent (und manchmal natürlicher) ist die folgende Variante:

### Theorem 1.44 (Kompaktheitssatz Variante 2)

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

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