

# Logik in der Künstlichen Intelligenz (Logikbasierte Systeme der Wissensverarbeitung)

## Aussagenlogik

Prof. Dr. M. Schmidt-Schauß

SoSe 2022

Stand der Folien: 26. April 2022

## Überblick

### Inhalt des Kapitels:

- Aussagenlogik allgemein
- Darstellung von Wissen in der Aussagenlogik
- Entscheidungs- und Deduktionsverfahren

KI-Ansatz der Verwendung einer Logik

basiert auf der **Wissenrepräsentationshypothese** (Brian Smith)

„Die Verarbeitung von Wissen lässt sich *trennen* in: *Repräsentation von Wissen*, wobei dieses Wissen eine Entsprechung in der realen Welt hat; und in einen *Inferenzmechanismus*, der Schlüsse daraus zieht.“

## Wissenrepräsentations- und inferenzsysteme (1)

### Allgemein:

- Programme bauen auf modellierten Fakten, Wissen, Beziehungen auf
- führen Operationen darauf durch, um neue Schlüsse zu ziehen

## Wissenrepräsentations- und inferenzsysteme (2)

### Komponenten und Eigenschaften:

- **Formale Sprache** zur Festlegung der Syntax (von Wissen und Regeln)
- Festlegung der Semantik (Bedeutung)
- **Inferenzprozedur** (operationale Semantik), z.B. Syntaktische Manipulation von Formeln
- **Korrektheit**: Inferenzprozedur erhält die Semantik
- **Vollständigkeit**: Inferenzprozedur findet alle notwendigen Folgerungen
- **Nachweis der Korrektheit / Vollständigkeit**:  
Untersuche Implementierung bzgl. der Semantik

Implementierung: **Parser** und **Implementierung der Inferenzprozedur**

## Aussagenlogik: Anwendbarkeit

- Man hat nur **endliche viele** elementare Aussagen (Variablenamen) die wahr oder falsch sein können.
- Diese Aussagen haben keine Parameter / innere Struktur
- Logische Zusammenhänge sind bekannt zwischen den elementaren Aussagen.
- Alle Zusammenhänge sind im Prinzip mit UND, ODER, NICHT, IMPLIZIERT formulierbar
- Schlüsse und Herleitungen auf den elementaren Aussagen...



### Beispiel

“Der Himmel ist blau”  
 “ich muss mehr lernen”  
 “Es ist kalt”

**Aber: der Text oder die “innere Bedeutung” kann nicht beim Schließen verwendet werden.**

## Syntax



Sei  $X$  ein Nichtterminal für aussagenlogische Variablen

### Syntax aussagenlogischer Formeln

$$A ::= X \mid 0 \mid 1 \\ \mid (A \wedge A) \mid (A \vee A) \mid (\neg A) \\ \mid (A \Rightarrow A) \mid (A \Leftrightarrow A)$$

Dabei entspricht

- 0 = falsche Aussage
- 1 = wahre Aussage

Eigentlich: 0 und 1 nicht nötig, man könnte auch  $A \wedge \neg A$  und  $A \vee \neg A$  stattdessen verwenden



## Syntaktische Konventionen



Variablenamen sollten aussagekräftig gewählt werden.

- „Wenn es heute nicht regnet, werde ich Fahrrad fahren“
- $\neg \text{heuteRegnetEs} \Rightarrow \text{fahrradfahren}$

Klammerung usw:

- Wir lassen Klammern oft weg (Ersatz: Klammerregeln)
- Da  $\vee, \wedge$  assoziativ: links- / rechts-Klammerung egal
- Priorität:  $\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow$ .
- Auch andere Operatoren werden manchmal verwendet:  $\Leftarrow, \text{NAND}, \text{NOR}, \text{XOR}$

## Sprechweisen



- $\wedge, \vee, \neg, \Rightarrow, \Leftrightarrow$  nennt man **Junktoren**
- Bezeichnung der einzelnen Junktoren:
  - $A \wedge B$ : **Konjunktion** (Verundung)
  - $A \vee B$ : **Disjunktion** (Veroderung)
  - $A \Rightarrow B$ : **Implikation** (Folgerung), (aus  $A$  folgt  $B$ )
  - $A \Leftrightarrow B$ : **Äquivalenz** (Bimplikation)
  - $\neg A$ : **negierte Formel** (Negation).

# Sprechweisen (2)

**Atom** = Aussagenlogische Variable, 0, oder 1

**Literal** = Atom oder  $\neg A$ , wobei  $A$  ein Atom ist

Beispiele:

- $X$  ist ein Atom
- $X, \neg X$  und 0 sind Literale
- $\neg(\neg X)$  ist weder Atom noch Literal (sondern Formel)

# Semantik der Junktoren

Definiere  $f_{op}$  für jeden Junktor

- $f_{\neg} : \{0, 1\} \rightarrow \{0, 1\}$  mit  $f_{\neg}(0) = 1$  und  $f_{\neg}(1) = 0$
- Für  $op \in \{\vee, \wedge, \Rightarrow, \Leftrightarrow, \Leftarrow, NOR, NAND, XOR\}$ :  
 $f_{op} : \{0, 1\}^2 \rightarrow \{0, 1\}$  mit

$a$	$b$	$f_{\wedge}(a, b)$	$f_{\vee}(a, b)$	$f_{\Rightarrow}(a, b)$	$f_{\Leftarrow}(a, b)$	$f_{NOR}(a, b)$	$f_{NAND}(a, b)$	$f_{\Leftrightarrow}(a, b)$	$f_{XOR}(a, b)$
1	1	1	1	1	1	0	0	1	0
1	0	0	1	0	1	0	1	0	1
0	1	0	1	1	0	0	1	0	1
0	0	0	0	1	1	1	1	1	0

# Interpretation

**Definition:**

Eine **Interpretation** (Variablenbelegung)  $I$  ist eine Funktion:

$$I : \{\text{aussagenlogische Variablen}\} \rightarrow \{0, 1\}.$$

Fortsetzung von  $I$  auf Aussagen:

- $I(0) := 0, I(1) := 1$
- $I(\neg A) := f_{\neg}(I(A))$
- $I(A \text{ op } B) := f_{op}(I(A), I(B))$ , wobei  $op \in \{\wedge, \vee, \Rightarrow, \Leftrightarrow, \Leftarrow, \dots\}$

# Beispiel

Sei  $I$  Interpretation mit:  $I(X) = 0, I(Y) = 1, I(Z) = 1$

$$\begin{aligned}
 & I(\neg(X \Rightarrow Z) \Rightarrow (Y \vee \neg Z)) \\
 &= f_{\Rightarrow}(I(\neg(X \Rightarrow Z)), I(Y \vee \neg Z)) \\
 &= f_{\Rightarrow}(f_{\neg}(I(X \Rightarrow Z)), f_{\vee}(I(Y), I(\neg Z))) \\
 &= f_{\Rightarrow}(f_{\neg}(f_{\Rightarrow}(I(X), I(Z))), f_{\vee}(I(Y), f_{\neg}(I(Z)))) \\
 &= f_{\Rightarrow}(f_{\neg}(f_{\Rightarrow}(0, 1)), f_{\vee}(1, f_{\neg}(1))) \\
 &= f_{\Rightarrow}(f_{\neg}(1), f_{\vee}(1, 0)) \\
 &= f_{\Rightarrow}(0, 1) \\
 &= 1
 \end{aligned}$$

# Modell

## Definition

Eine Interpretation  $I$  ist genau dann ein **Modell** für die Aussage  $F$ , wenn  $I(F) = 1$  gilt.

Wir schreiben in diesem Fall:  $I \models F$

Alternative Sprechweisen:

- $F$  gilt in  $I$
- $I$  macht  $F$  wahr



# Eigenschaften von Formeln



Sei  $A$  ein Aussage.

- $A$  ist eine **Tautologie** (Satz, allgemeingültig), gdw. für alle Interpretationen  $I$  gilt:  $I \models A$ .
- $A$  ist ein **Widerspruch** (widersprüchlich, unerfüllbar), gdw. für alle Interpretationen  $I$  gilt:  $I(A) = 0$ .
- $A$  ist **erfüllbar** (konsistent), gdw. eine Interpretation  $I$  existiert mit:  $I(A) = 1$  bzw.  $I \models A$
- $A$  ist **falsifizierbar**, gdw. eine Interpretation  $I$  existiert mit  $I(A) = 0$ .



# Eigenschaften von Formeln



erfüllbar

falsifizierbar

# Beispiele (1)



$X \vee \neg X$

- ist eine Tautologie, denn für jede Interpretation  $I$  gilt:  $I(X \vee \neg X) = f_{\vee}(f_{\neg}(I(X)), I(X)) = 1$ , da  $f_{\neg}(I(X))$  oder  $I(X)$  gleich zu 1 sein muss.
- ist erfüllbar (jede Interpretation ist ein Modell)
- ist nicht falsifizierbar
- ist kein Widerspruch

## Beispiele (2)

$$X \wedge \neg X$$

- ist ein Widerspruch, denn für jede Interpretation  $I$  gilt:  
 $I(X \wedge \neg X) = f_{\wedge}(f_{\wedge}(I(X)), I(\neg X)) = 0$ , da  $f_{\neg}(I(X))$  oder  $I(X)$  gleich zu 0 sein muss.
- ist falsifizierbar (für jede Interpretation  $I$  gilt  $I(X \wedge \neg X) = 0$ )
- ist nicht erfüllbar
- ist keine Tautologie

## Beispiele (3)

$$(X \Rightarrow Y) \Rightarrow ((Y \Rightarrow Z) \Rightarrow (X \Rightarrow Z))$$

- ist eine Tautologie.
- ist erfüllbar
- ist nicht falsifizierbar
- ist kein Widerspruch

## Beispiele (4)

$$X \vee Y$$

- ist erfüllbar. Z.B. ist  $I$  mit  $I(X) = 1, I(Y) = 1$  ein Modell für  
 $X \vee Y: I(X \vee Y) = f_{\vee}(I(X), I(Y)) = f_{\vee}(1, 1) = 1$ .
- ist falsifizierbar, denn mit  $I(X) = 0, I(Y) = 0$  gilt  
 $I(X \vee Y) = 0$
- ist keine Tautologie
- ist kein Widerspruch

## Beispiele (5)

„Abendrot Schlechtwetterbot“

- Formel dazu: [Abendrot](#)  $\Rightarrow$  [Schlechtes Wetter](#)
- Erfüllbar und Falsifizierbar
- Weder Tautologie noch Widerspruch

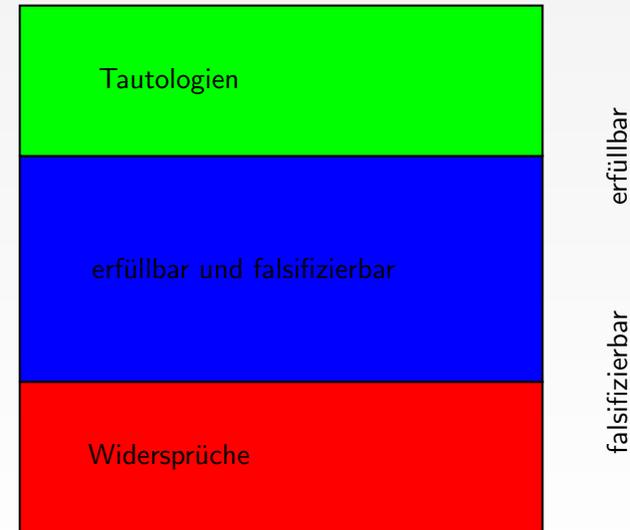
## Beispiele (6)

Wenn der Hahn kräht auf dem Mist, ändert sich das Wetter oder es bleibt wie es ist.

- Formel dazu:  
 $\text{Hahn\_kraeht\_auf\_Mist} \Rightarrow (\text{Wetteraenderung} \vee \neg \text{Wetteraenderung})$
- ist eine Tautologie
- ist erfüllbar
- ist weder widersprüchlich noch falsifizierbar



## Eigenschaften von Formeln



## Komplexitäten

### Theorem: Aussagenlogik

- Es ist entscheidbar, ob eine Aussage eine Tautologie (Widerspruch, erfüllbar, falsifizierbar) ist.  
 Einfachstes Verfahren: **Wahrheitstabelle**
- Die Frage „Ist  $A$  erfüllbar?“ ist  $\mathcal{NP}$ -vollständig.
- Die Frage „Ist  $A$  falsifizierbar?“ ist ebenso  $\mathcal{NP}$ -vollständig.
- Die Frage „Ist  $A$  Tautologie?“ ist  $\text{CoNP}$ -vollständig.
- Die Frage „Ist  $A$  ein Widerspruch?“ ist  $\text{CoNP}$ -vollständig.

## Exkurs: Komplexitätsklassen (zur Erinnerung)

Problemlasse = Wortproblem über einer formalen Sprache

$$\text{SAT} := \{F \mid F \text{ ist erfüllbare aussagenlogische Formel}\}.$$

Wortproblem: Liegt eine gegebene Formel in der Sprache SAT ?

Antwort: Ja ( oder Nein )

## Exkurs: Komplexitätsklassen (2)

$\mathcal{NP}$  = Alle Sprachen, deren Wortproblem auf einer nichtdeterministischen Turingmaschine in **polynomieller Zeit** lösbar ist.

$Co\mathcal{NP} = \{L \mid L^C \in \mathcal{NP}\}$   
wobei  $L^C$  das Komplement der Sprache  $L$  ist.

Offensichtlich:  $L \in \mathcal{NP} \iff L^C \in Co\mathcal{NP}$

Beispiel:  $UNSAT := \{F \mid F \text{ ist Widerspruch}\}$

$UNSAT \in Co\mathcal{NP}$ , denn:

$UNSAT^C = (\text{Alle Formeln} \setminus UNSAT) = SAT$ , und  $SAT \in \mathcal{NP}$ .



## Exkurs: Komplexitätsklassen (3)

$L \in \mathcal{NP}$  ist  **$\mathcal{NP}$ -vollständig** gdw.:  
für jede Sprache  $L' \in \mathcal{NP}$  gibt es eine Polynomialzeit-Kodierung  $R: L' \rightarrow L$  mit  $x \in L'$  gdw.  $R(x) \in L$

$L \in Co\mathcal{NP}$  ist  **$Co\mathcal{NP}$ -vollständig** gdw.:  
für jede Sprache  $L' \in Co\mathcal{NP}$  gibt es eine Polynomialzeit-Kodierung  $R: L' \rightarrow L$  mit  $x \in L'$  gdw.  $R(x) \in L$



## Exkurs: Komplexitätsklassen (4)

**Theorem**

$L$  ist  $Co\mathcal{NP}$ -vollständig gdw.:  $L^C$  ist  $\mathcal{NP}$ -vollständig

## Komplexitäten (nochmal)

**Theorem**

- Es ist entscheidbar, ob eine Aussage eine Tautologie (Widerspruch, erfüllbar, falsifizierbar) ist.
- Die Frage „Ist  $A$  erfüllbar?“ ist  $\mathcal{NP}$ -vollständig.
- Die Frage „Ist  $A$  falsifizierbar?“ ist ebenso  $\mathcal{NP}$ -vollständig.
- Die Frage „Ist  $A$  Tautologie?“ ist  $Co\mathcal{NP}$ -vollständig.
- Die Frage „Ist  $A$  ein Widerspruch?“ ist  $Co\mathcal{NP}$ -vollständig.



## Fazit

- Sequentielle Algorithmen zur Lösung beider Problemklassen: nur Exponential-Zeit Algorithmen sind bekannt (solange  $\mathcal{P} = \mathcal{NP}$  unbekannt)
- $\mathcal{NP} \stackrel{?}{=} \text{CoNP}$  ist unbekannt, allgemeine Vermutung: Nein
- Probleme aus  $\mathcal{NP}$  kann man mit Glück (Raten) schnell lösen (Lösungen sind polynomiell verifizierbar)
- Probleme aus  $\text{CoNP}$  scheinen schwieriger als  $\mathcal{NP}$ -Probleme. z.B. Tautologie: man muss alle Interpretation durchprobieren



### Beispiel Suche

## Folgerungsbegriffe



### Unterscheide zwischen

- **Semantische Folgerung**  
definiert mittels der Semantik
- **Syntaktische Folgerung** (Herleitung, Ableitung)  
Verfahren mit einer prozeduralen Vorschrift, oft: nicht-deterministischer Kalkül.



## Semantische Folgerung



- $\mathcal{F}$  eine Menge von (aussagenlogischen) Formeln
- $G$  eine weitere Formel.

$G$  **folgt semantisch** aus  $\mathcal{F}$   
gdw.

Für alle Interpretationen  $I$ :  
wenn für alle  $F \in \mathcal{F}$  gilt  $I(F) = 1$ , dann auch  $I(G) = 1$ .

**Schreibweise:**  $\mathcal{F} \models G$

## Einfache Resultate



Betrachte  $\{F_1, \dots, F_n\} \models G$

- 1 Wenn ein  $F_i$  **widersprüchlich**, dann kann man alles folgern:  
Es gilt dann für **jede Formel**  $G$ :  $\{F_1, \dots, F_n\} \models G$ .
- 2 Wenn ein  $F_i$  **eine Tautologie** ist, dann kann man  $F_i$  **weglassen**:  
 $\{F_1, \dots, F_n\} \models G$  ist dasselbe wie  
 $\{F_1, \dots, F_{i-1}, F_{i+1}, \dots, F_n\} \models G$

## Deduktionstheorem der Aussagenlogik

### Deduktionstheorem

$\{F_1, \dots, F_n\} \models G$  gdw.  $F_1 \wedge \dots \wedge F_n \Rightarrow G$  ist Tautologie.

Aussagen  $F, G$  nennt man **äquivalent** ( $F \sim G$ ),  
gdw.  $F \iff G$  eine Tautologie ist.

Beobachtungen:

- $F \sim G$  gdw. für alle  $I$ :  $I \models F$  gdw.  $I \models G$
- $X \wedge Y$  **nicht äquivalent** zu  $X' \wedge Y'$   
(Variablenbelegung spielen eine Rolle!)

## Syntaktische Folgerung

- $\mathcal{A}$ : (nichtdeterministischer) Algorithmus (Kalkül), der aus einer Menge von Formeln  $\mathcal{H}$  eine neue Formel  $H$  berechnet
- $H$  **folgt syntaktisch** aus  $\mathcal{H}$
- Schreibweisen  $\mathcal{H} \vdash_{\mathcal{A}} H$  oder auch  $\mathcal{H} \rightarrow_{\mathcal{A}} H$

### Definition

- Der Algorithmus  $\mathcal{A}$  ist **korrekt** (sound), gdw. aus  $\mathcal{H} \vdash_{\mathcal{A}} H$  stets  $\mathcal{H} \models H$  folgt.
- Der Algorithmus  $\mathcal{A}$  ist **vollständig** (complete), gdw.  $\mathcal{H} \models H$  impliziert, dass  $\mathcal{H} \vdash_{\mathcal{A}} H$

## Korrekt & Vollständiger Algorithmus (naiv)

Gegeben:  $\{F_1, \dots, F_n\}$  und Tautologiechecker

Verfahren:

- Zähle alle Formeln  $F$  auf
- Prüfe ob  $F_1 \wedge \dots \wedge F_n \implies F$  Tautologie
- Wenn ja:  $\{F_1, \dots, F_n\} \models F$  (wegen Deduktionstheorem)

Problem:

Immer unendlich viele  $F$ , da alle Tautologien  
aus  $\{F_1, \dots, F_n\}$  folgen.

Es gilt:  $\{F_1, \dots, F_n\} \models G$  für alle Tautologien  $G$

## Methoden zur Tautologie-Erkennung

(auch Erfüllbarkeit)

- Wahrheitstabelle
- BDDs (binary decision diagrams): Aussagen als boolesche Funktionen, kompakte Darstellung
- Genetische Algorithmen (Erfüllbarkeit)
- Suchverfahren mit zufälliger Suche
- Davis-Putnam-Verfahren (später): Fallunterscheidung mit Simplifikationen
- Tableauekalkül (später): Syntaktische Analyse der Formeln

## Substitution

Wir schreiben  $A[B/x]$  für die **Ersetzung** aller Vorkommen von Variable  $x$  in Formel  $A$  durch Formel  $B$

Beispiel:

$$(x \vee y) \implies (\neg(x \wedge z))[(z \implies (w \wedge u))/x]$$

$$= ((z \implies (w \wedge u)) \vee y) \implies (\neg((z \implies (w \wedge u)) \wedge z))$$

Verallgemeinerung:  $A[B_1/x_1, \dots, B_n/x_n]$  ist die **parallele Ersetzung** aller  $x_i$  durch  $B_i$

Beispiel:

$$(x \vee y) \implies (\neg(x \wedge z))[(z \implies (w \wedge u))/x, (a \wedge b)/z]$$

$$= ((z \implies (w \wedge u)) \vee y) \implies (\neg((z \implies (w \wedge u)) \wedge (a \wedge b)))$$

**Nicht:**

$$(x \vee y) \implies (\neg(x \wedge z))[(z \implies (w \wedge u))/x, (a \wedge b)/z] =$$

$$((z \implies (w \wedge u)) \vee y) \implies (\neg(((a \wedge b) \implies (w \wedge u)) \wedge (a \wedge b)))$$



## Substitution erhält Äquivalenz

### Satz

Gilt  $A_1 \sim A_2$  und  $B_1, \dots, B_n$  weitere Aussagen, dann gilt auch  $A_1[B_1/x_1, \dots, B_n/x_n] \sim A_2[B_1/x_1, \dots, B_n/x_n]$ .

Beweis: Seien  $\{x_1, \dots, x_m\}$  alle Variablen, die in  $A_i, B_i$  vorkommen

Wahrheitstafel für  $A_1, A_2$ :

$x_1$	...	$x_n$	$x_{n+1}$	...	$x_m$	$A_1$	$A_2$
...	...	...	...	...	...	...	...
$c_{j,1}$	...	$c_{j,n}$	$c_{j,n+1}$	...	$c_{j,m}$	$d_j$	$d'_j$
$b_{i,1}$	...	$b_{i,n}$	$c_{i,n+1}$	...	$c_{i,m}$	$d_i$	$d'_i$
...	...	...	...	...	...	...	...

Da  $A_1 \sim A_2$ , muss gelten  $d_j = d'_j$  für alle  $j$

Wahrheitstafel für  $A_1[B_1/x_1, \dots, B_n/x_n]$  und  $A_2[B_1/x_1, \dots, B_n/x_n]$ :

$x_1$	...	$x_n$	$x_{n+1}$	...	$x_m$	$B_1$	...	$B_n$	$A_1[B_1/x_1, \dots, B_n/x_n]$	$A_2[B_1/x_1, \dots, B_n/x_n]$
...	...	...	...	...	...	...	...	...	...	...
$c_{i,1}$	...	$c_{i,n}$	$c_{i,n+1}$	...	$c_{i,m}$	$b_{i,1}$	...	$b_{i,n}$	$d_i$	$d'_i$
...	...	...	...	...	...	...	...	...	...	...

Zeige  $d_i = d'_i$ :

$$d_i = A_1[b_{i,1}/x_1, \dots, b_{i,n}/x_n, c_{i,n+1}/x_{n+1}, \dots, c_{i,m}/x_m]$$

$$d'_i = A_2[b_{i,1}/x_1, \dots, b_{i,n}/x_n, c_{i,n+1}/x_{n+1}, \dots, c_{i,m}/x_m]$$



## Äquivalenz ist kompatibel mit Kontexten

Notation:  $F[A]$  Formel mit  $A$  an einer Stelle ( $F$  ist ein Kontext)

### Satz

Sind  $A, B$  äquivalente Aussagen, und ist  $F[A]$  eine weitere Aussage, dann sind  $F[A]$  und  $F[B]$  ebenfalls äquivalent.

(Formaler: für alle Kontexte  $F$  gilt:

$$\text{Wenn } A \sim B, \text{ dann auch } F[A] \sim F[B])$$

## Kommutativität, Assoziativität, Idempotenz

$\wedge$  und  $\vee$  sind kommutativ, assoziativ, und idempotent, d.h.:

Für alle Formeln  $\mathcal{F}, \mathcal{G}, \mathcal{H}$ :

$$\mathcal{F} \wedge \mathcal{G} \iff \mathcal{G} \wedge \mathcal{F}$$

$$\mathcal{F} \wedge \mathcal{F} \iff \mathcal{F}$$

$$\mathcal{F} \wedge (\mathcal{G} \wedge \mathcal{H}) \iff (\mathcal{F} \wedge \mathcal{G}) \wedge \mathcal{H}$$

$$\mathcal{F} \vee \mathcal{G} \iff \mathcal{G} \vee \mathcal{F}$$

$$\mathcal{F} \vee (\mathcal{G} \vee \mathcal{H}) \iff (\mathcal{F} \vee \mathcal{G}) \vee \mathcal{H}$$

$$\mathcal{F} \vee \mathcal{F} \iff \mathcal{F}$$

## Weitere Rechenregeln

$\neg(\neg A)$	$\iff$	$A$	
$(A \implies B)$	$\iff$	$(\neg A \vee B)$	
$(A \iff B)$	$\iff$	$((A \implies B) \wedge (B \implies A))$	
$\neg(A \wedge B)$	$\iff$	$\neg A \vee \neg B$	(DeMorgansche Gesetze)
$\neg(A \vee B)$	$\iff$	$\neg A \wedge \neg B$	
$A \wedge (B \vee C)$	$\iff$	$(A \wedge B) \vee (A \wedge C)$	Distributivität
$A \vee (B \wedge C)$	$\iff$	$(A \vee B) \wedge (A \vee C)$	Distributivität
$(A \implies B)$	$\iff$	$(\neg B \implies \neg A)$	Kontraposition
$A \vee (A \wedge B)$	$\iff$	$A$	Absorption
$A \wedge (A \vee B)$	$\iff$	$A$	Absorption

$A, B, C$  beliebige Aussagen!



## Normalformen

### Disjunktive Normalform (DNF)

- Disjunktion von Konjunktionen von Literalen
- $(L_{1,1} \wedge \dots \wedge L_{1,n_1}) \vee \dots \vee (L_{m,1} \wedge \dots \wedge L_{m,n_m})$   
wobei  $L_{i,j}$  Literale sind.

### Konjunktive Normalform (CNF)

- Konjunktion von Disjunktionen von Literalen
- $(L_{1,1} \vee \dots \vee L_{1,n_1}) \wedge \dots \wedge (L_{m,1} \vee \dots \vee L_{m,n_m})$   
wobei  $L_{i,j}$  Literale sind.



## Konjunktive Normalform

$$\underbrace{(L_{1,1} \vee \dots \vee L_{1,n_1})}_{\text{Klausel}} \wedge \dots \wedge \underbrace{(L_{m,1} \vee \dots \vee L_{m,n_m})}_{\text{Klausel}}$$

Daher auch: **Klauselnormalform**

**Mengenschreibweise:**

$$\{ \underbrace{\{L_{1,1}, \dots, L_{1,n_1}\}}_{\text{Klausel}}, \dots, \underbrace{\{L_{m,1}, \dots, L_{m,n_m}\}}_{\text{Klausel}} \}$$

(gerechtfertigt, da  $\vee, \wedge$  kommutativ, assoziativ, idempotent)

Konjunktive Normalform = Klauselnormalform = Klauselmenge



## Spezielle Klauseln

- **Leere Klausel**  $\emptyset := 0 = \text{Widerspruch}$   
Beachte: enthält die CNF eine leere Klausel, so ist die Formel auch widersprüchlich:  
 $\{C_1, \dots, C_n, \emptyset\} = C_1 \wedge \dots \wedge C_n \wedge 0 \sim 0$
- **Einklausel:**  $\{p\}$  mit  $p$  Literal (auch **Unit-Klausel**)  
Beachte:  $I(\{C_1, \dots, C_n, \{p\}\}) = 1$  impliziert  $I(p) = 1$   
D.h. jedes Modell muss auch  $p$  wahr machen

## CNF / DNF Berechnung

Es gilt:

**Satz**

Zu jeder Aussage kann man eine äquivalente CNF finden und ebenso eine äquivalente DNF.

Allerdings ist die CNF / DNF nicht eindeutig im obigen Satz.  
Verfahren zur Berechnung: später



## CNF/DNF: Tautologie und Widerspruch



Für die CNF gilt:

- Eine Klausel  $C$  ist eine Tautologie gdw.  $C = C' \cup \{l, \neg l\}$  (d.h. Literal  $l$  kommt positiv und negativ vor)
- Eine Klauselmeng (CNF) ist eine Tautologie gdw. alle Klauseln Tautologien sind

**Dual dazu:**

Für die DNF gilt: (Monom =  $\{l_1, \dots, l_n\}$  mit  $(l_1 \wedge \dots \wedge l_n)$ )

- Ein Monom  $D$  ist widersprüchlich gdw.  $D = D' \cup \{l, \neg l\}$  (d.h. Literal  $l$  kommt positiv und negativ vor)
- Eine DNF ist widersprüchlich gdw. alle Monome widersprüchlich sind



## CNF/DNF: Tautologie und Widerspruch (2)

**Satz**

- Eine Aussage in **CNF** kann man in Zeit  $O(n * \log(n))$  auf **Tautologieeigenschaft** testen.
- Eine Aussage in **DNF** kann man in Zeit  $O(n * \log(n))$  auf **Unerfüllbarkeit** testen.

Dabei ist  $n$  die syntaktische Größe der CNF/DNF

Beweis: Verwende die Eigenschaften, und sortiere die Klauseln bzw. Monome

## CNF/DNF: Tautologie und Widerspruch (3)

**Beachte:** Die Umkehrungen gelten **nicht**

- Test einer CNF auf Un-/Erfüllbarkeit
  - Test einer DNF auf Allgemeingültig- / Falsifizierbarkeit
- wohl nur exponentiell sequentiell durchführbar

(unter der Annahme:  $\mathcal{P} \neq \mathcal{NP}$ )

Begründung: später

# CNF/DNF: Tautologie und Widerspruch (4)

## Nochmal

Eine Aussage in CNF kann man in Zeit  $O(n * \log(n))$  auf Tautologieeigenschaft testen.

## Impliziert:

Transformation einer Formel  $F$  in äquivalente CNF geht **nicht in Polynomialzeit** (Annahme dabei  $P \neq CoNP$ )

## Begründung:

- anderenfalls könnte man Tautologieeigenschaft in Polynomialzeit entscheiden
- aber Tautologieeigenschaft ist  $CoNP$ -vollständig
- DNF: Analog



# Dualitätsprinzip

Methoden, Kalküle, Algorithmen haben stets eine **duale Variante**

- $\wedge \leftrightarrow \vee$
- $0 \leftrightarrow 1$
- $CNF \leftrightarrow DNF$
- Tautologie  $\leftrightarrow$  Widerspruch
- Test auf Allgemeingültigkeit  $\leftrightarrow$  Test auf Unerfüllbarkeit.

Daher:

- Beweissysteme können Allgemeingültigkeit oder Unerfüllbarkeit testen
- Geschmacksfrage, keine prinzipielle Frage

## Satz

Sei  $F$  eine Formel.

Dann ist  $F$  allgemeingültig gdw.  $\neg F$  ein Widerspruch ist



# Berechnung der Klauselnormalform

## Algorithmus Transformation in CNF

**Eingabe:** Formel  $F$

**Algorithmus:**

Führe nacheinander die folgenden Schritte durch:

- 1 Elimination von  $\leftrightarrow$  und  $\Rightarrow$ :

$$\begin{aligned} F \leftrightarrow G &\rightarrow (F \Rightarrow G) \wedge (G \Rightarrow F) \\ F \Rightarrow G &\rightarrow \neg F \vee G \end{aligned}$$

- 2 Negation ganz nach innen schieben:

$$\begin{aligned} \neg \neg F &\rightarrow F \\ \neg(F \wedge G) &\rightarrow \neg F \vee \neg G \\ \neg(F \vee G) &\rightarrow \neg F \wedge \neg G \end{aligned}$$

- 3 Distributivität (und Assoziativität, Kommutativität) iterativ anwenden, um  $\wedge$  nach außen zu schieben ("Ausmultiplikation").

$$F \vee (G \wedge H) \rightarrow (F \vee G) \wedge (F \vee H)$$

Ausgabe ist CNF

# Beispiel

$$(X \Rightarrow Y) \iff (Y \Rightarrow (X \wedge Z))$$

Schritt 1: Äquivalenz und Implikation entfernen

$$\begin{aligned} &(X \Rightarrow Y) \iff (Y \Rightarrow (X \wedge Z)) \\ \rightarrow &((X \Rightarrow Y) \Rightarrow (Y \Rightarrow (X \wedge Z))) \wedge ((Y \Rightarrow (X \wedge Z)) \Rightarrow (X \Rightarrow Y)) \\ \rightarrow &((\neg X \vee Y) \Rightarrow (Y \Rightarrow (X \wedge Z))) \wedge ((Y \Rightarrow (X \wedge Z)) \Rightarrow (X \Rightarrow Y)) \\ \rightarrow &((\neg X \vee Y) \Rightarrow (\neg Y \vee (X \wedge Z))) \wedge ((Y \Rightarrow (X \wedge Z)) \Rightarrow (X \Rightarrow Y)) \\ \rightarrow &((\neg X \vee Y) \Rightarrow (\neg Y \vee (X \wedge Z))) \wedge ((\neg Y \vee (X \wedge Z)) \Rightarrow (X \Rightarrow Y)) \\ \rightarrow &((\neg X \vee Y) \Rightarrow (\neg Y \vee (X \wedge Z))) \wedge ((\neg Y \vee (X \wedge Z)) \Rightarrow (\neg X \vee Y)) \\ \rightarrow &(\neg(\neg X \vee Y) \vee (\neg Y \vee (X \wedge Z))) \wedge ((\neg Y \vee (X \wedge Z)) \Rightarrow (\neg X \vee Y)) \\ \rightarrow &(\neg(\neg X \vee Y) \vee (\neg Y \vee (X \wedge Z))) \wedge (\neg(\neg Y \vee (X \wedge Z)) \vee (\neg X \vee Y)) \end{aligned}$$



## Beispiel (2)

### Schritt 2: Negationen nach innen schieben

$$\begin{aligned}
 & (\neg(\neg X \vee Y) \vee (\neg Y \vee (X \wedge Z))) \wedge (\neg(\neg Y \vee (X \wedge Z)) \vee (\neg X \vee Y)) \\
 \rightarrow & ((\neg\neg X \wedge \neg Y) \vee (\neg Y \vee (X \wedge Z))) \wedge (\neg(\neg Y \vee (X \wedge Z)) \vee (\neg X \vee Y)) \\
 \rightarrow & ((X \wedge \neg Y) \vee (\neg Y \vee (X \wedge Z))) \wedge (\neg(\neg Y \vee (X \wedge Z)) \vee (\neg X \vee Y)) \\
 \rightarrow & ((X \wedge \neg Y) \vee (\neg Y \vee (X \wedge Z))) \wedge ((\neg\neg Y \wedge \neg(X \wedge Z)) \vee (\neg X \vee Y)) \\
 \rightarrow & ((X \wedge \neg Y) \vee (\neg Y \vee (X \wedge Z))) \wedge ((Y \wedge \neg(X \wedge Z)) \vee (\neg X \vee Y)) \\
 \rightarrow & ((X \wedge \neg Y) \vee (\neg Y \vee (X \wedge Z))) \wedge ((Y \wedge (\neg X \vee \neg Z)) \vee (\neg X \vee Y))
 \end{aligned}$$

## Beispiel (4)

$$(\neg Y \vee X) \wedge (\neg Y \vee Z \vee X) \wedge (\neg Y \vee Z) \wedge (\neg X \vee Y) \wedge (\neg X \vee Y \vee \neg Z)$$

Klauselmenge dazu:

$$\{\{\neg Y, X\}, \{\neg Y, Z, X\}, \{\neg Y, Z\}, \{\neg X, Y\}, \{\neg X, Y, \neg Z\}\}$$

## Beispiel (3)

### Schritt 3: Ausmultiplizieren

$$\begin{aligned}
 & ((X \wedge \neg Y) \vee (\neg Y \vee (X \wedge Z))) \wedge ((Y \wedge (\neg X \vee \neg Z)) \vee (\neg X \vee Y)) \\
 \rightarrow & ((X \wedge \neg Y) \vee ((\neg Y \vee X) \wedge (\neg Y \vee Z))) \wedge ((Y \wedge (\neg X \vee \neg Z)) \vee (\neg X \vee Y)) \\
 \rightarrow & (((X \wedge \neg Y) \vee (\neg Y \vee X)) \wedge ((X \wedge \neg Y) \vee (\neg Y \vee Z))) \wedge ((Y \wedge (\neg X \vee \neg Z)) \vee (\neg X \vee Y)) \\
 \rightarrow & (((\neg Y \vee X) \vee (X \wedge \neg Y)) \wedge ((X \wedge \neg Y) \vee (\neg Y \vee Z))) \wedge ((Y \wedge (\neg X \vee \neg Z)) \vee (\neg X \vee Y)) \\
 \rightarrow & (((\neg Y \vee X \vee X) \wedge (\neg Y \vee X \vee \neg Y)) \wedge ((X \wedge \neg Y) \vee (\neg Y \vee Z))) \wedge ((Y \wedge (\neg X \vee \neg Z)) \vee (\neg X \vee Y)) \\
 \rightarrow & (((\neg Y \vee X \vee X) \wedge (\neg Y \vee X \vee \neg Y)) \wedge ((\neg Y \vee Z) \vee (X \wedge \neg Y))) \wedge ((Y \wedge (\neg X \vee \neg Z)) \vee (\neg X \vee Y)) \\
 \rightarrow & (((\neg Y \vee X \vee X) \wedge (\neg Y \vee X \vee \neg Y)) \wedge ((\neg Y \vee Z \vee X) \wedge (\neg Y \vee Z \vee \neg Y))) \wedge ((Y \wedge (\neg X \vee \neg Z)) \vee (\neg X \vee Y)) \\
 \rightarrow & (((\neg Y \vee X \vee X) \wedge (\neg Y \vee X \vee \neg Y)) \wedge ((\neg Y \vee Z \vee X) \wedge (\neg Y \vee Z \vee \neg Y))) \wedge ((\neg X \vee Y \vee Y) \wedge (\neg X \vee Y \vee \neg X \vee \neg Z)) \\
 = & (\neg Y \vee X \vee X) \wedge (\neg Y \vee X \vee \neg Y) \wedge (\neg Y \vee Z \vee X) \wedge (\neg Y \vee Z \vee \neg Y) \wedge (\neg X \vee Y \vee Y) \wedge (\neg X \vee Y \vee \neg X \vee \neg Z) \\
 = & (\neg Y \vee X) \wedge (\neg Y \vee X) \wedge (\neg Y \vee Z \vee X) \wedge (\neg Y \vee Z) \wedge (\neg X \vee Y) \wedge (\neg X \vee Y \vee \neg Z) \\
 = & (\neg Y \vee X) \wedge (\neg Y \vee Z \vee X) \wedge (\neg Y \vee Z) \wedge (\neg X \vee Y) \wedge (\neg X \vee Y \vee \neg Z)
 \end{aligned}$$

## Transformation in CNF

- Die erhaltene CNF ist äquivalent zur ursprünglichen Formel
- Laufzeit: Im Worst-Case exponentiell
- Grund: CNF kann exponentiell groß werden:
- die Elimination von  $\Leftrightarrow$  verdoppelt die Größe:  
 $(A_1 \Leftrightarrow A_2) \rightarrow (A_1 \Rightarrow A_2) \wedge (A_2 \Rightarrow A_1)$
- Ausmultiplikation mittels Distributivgesetz:  
 $A_1 \vee (B_1 \wedge B_2) \rightarrow (A_1 \vee B_1) \wedge (A_1 \vee B_2)$   
verdoppelt  $A_1$

Beachte: Unter Erhaltung der Äquivalenz geht es nicht besser (da sonst Tautologiecheck polynomiell möglich)

# Schnelle CNF-Berechnung



## Algorithmus zur schnellen CNF-Berechnung (Tseitin-Transformation)

- in polynomieller Laufzeit
- Erhält die **Tautologieeigenschaft nicht**
- Aber: Erhält die **Erfüllbarkeit**:
- $F$  erfüllbar gdw. schnelle CNF zu  $F$  erfüllbar

# Schnelle CNF-Berechnung (2)



## Idee dabei:

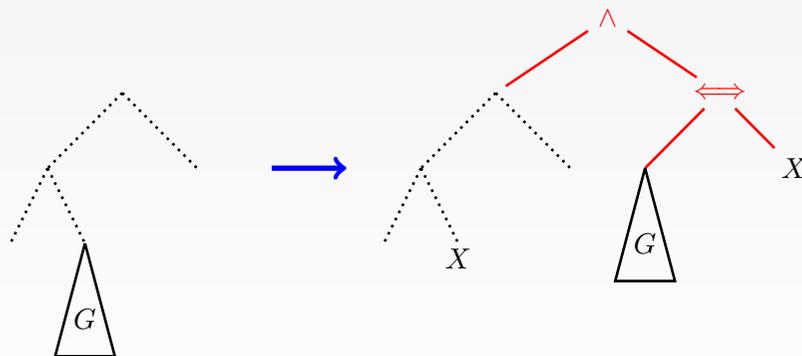
- Exponentielles Anwachsen bei der CNF-Berechnung: ist exponentiell in der **Tiefe** der Formel (Tiefe: Formel als Baum hingemalt)
- Daher: Vorverarbeitung: Klopfe die Formel flach

Einführen von Abkürzungen:

$$F[G] \rightarrow F[X] \wedge (X \iff G), \text{ wobei } X \text{ neue Variable}$$



# Schnelle CNF-Berechnung (3)



# Erhaltung der Erfüllbarkeit



## Satz

$F[G]$  ist erfüllbar gdw.  $(G \iff X) \wedge F[X]$  erfüllbar ist. Hierbei muß  $X$  eine Variable sein, die nicht in  $F[G]$  vorkommt.

Wesentlicher Schritt im Beweis:

- Interpretation  $I$ , die bezeugt, dass  $F[G]$  erfüllbar ist, d.h.  $I(F[G]) = 1$
- Erstelle  $I'$  mit

$$I'(Y) = \begin{cases} I(Y), & \text{wenn } Y \neq X \\ I(G) & \text{wenn } Y = X \end{cases}$$



## Eigenschaften

- Am Ende: Alle Subformeln  $H_i$  haben Tiefe  $\leq 3$   
 $\implies$  Transformation in konstanter Zeit
- Anzahl der eingef. Abkürzungen: max. linear in der Größe der Formel
- $H_i[G_1, \dots, G_m] \rightarrow (G_1 \Leftrightarrow X_1) \wedge \dots \wedge (G_m \Leftrightarrow X_m) \wedge H_i[X_1, \dots, X_m]$ 
  - Wenn  $H_i[G_1, \dots, G_m]$  Tiefe  $n \geq 3$  hat
  - $H_i[X_1, \dots, X_m]$  hat Tiefe 3
  - $G_j$  hat Tiefe höchstens  $n - 3$
  - die neuen Formeln  $(G_j \Leftrightarrow X_j)$  haben Tiefe höchstens  $n - 2$
  - D.h. 1 Formel der Tiefe  $n$  wird im worst case ersetzt durch  $m$  Formeln der Tiefe  $n - 2$  und eine Formel der Tiefe 3

Bei effizienter Implementierung gilt daher:

### Satz

Zu jeder aussagenlogischen Formel kann unter Erhaltung der Erfüllbarkeit eine CNF in Zeit  $O(n)$  berechnet werden, wobei  $n$  die Größe der aussagenlogischen Formel ist.

## Resolutionsverfahren für die Aussagenlogik

Testet Formel auf **Widersprüchlichkeit**

**Tautologiecheck** für Formel  $F$ : Starte mit  $\neg F$

Denn es gilt:

### Satz

Eine Formel  $A_1 \wedge \dots \wedge A_n \Rightarrow F$  ist allgemeingültig gdw.  $A_1 \wedge \dots \wedge A_n \wedge \neg F$  widersprüchlich ist.

Semantisch:

### Satz

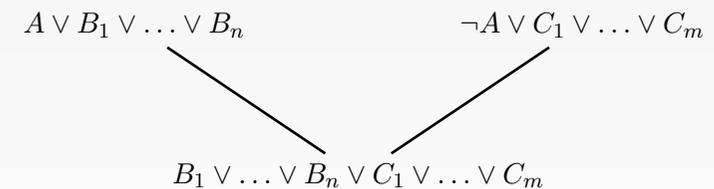
$\{A_1, \dots, A_n\} \models F$  gdw. es keine Interpretation  $I$  gibt, so dass  $I \models \{A_1, \dots, A_n, \neg F\}$

## Resolutionsregel

$$\begin{array}{r}
 A \vee B_1 \vee \dots \vee B_n \quad \text{Elternklausel 1} \\
 \neg A \vee C_1 \vee \dots \vee C_m \quad \text{Elternklausel 2} \\
 \hline
 B_1 \vee \dots \vee B_n \vee C_1 \vee \dots \vee C_m \quad \text{Resolvente}
 \end{array}$$

Doppelvorkommen von Literalen werden (implizit) eliminiert

## Resolutionsregel: Graphische Darstellung



# Resolutionsverfahren

- 1 Starte mit Klauselmenge  $\mathcal{C}$
- 2 Wähle Elternklauseln  $E_1, E_2$  aus  $\mathcal{C}$
- 3 **Füge** Resolvente  $R$  zur Klauselmenge  $\mathcal{C}$  **hinzu**:

$$\mathcal{C} := \mathcal{C} \cup \{R\}$$

Iteriere die obigen Schritte solange bis:

- Keine neue Resolvente mehr herleitbar, oder
- $\mathcal{C}$  enthält die leere Klausel  $\emptyset$ .

# Beispiel

Zeige

$$(A \wedge (A \Rightarrow B) \wedge (B \Rightarrow C)) \Rightarrow C \text{ allgemeingültig.}$$

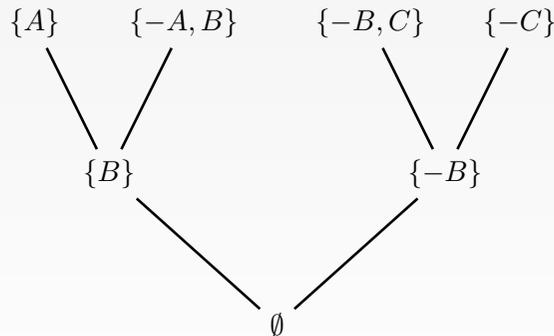
Starte mit negierter Formel (Test auf Widersprüchlichkeit)

$$\neg((A \wedge (A \Rightarrow B) \wedge (B \Rightarrow C)) \Rightarrow C)$$

CNF-Berechnung ergibt die Klauselmenge:

$$\{\{A\}, \{-A, B\}, \{-B, C\}, \{-C\}\}$$

# Beispiel (2)



# Korrektheit

## Satz

Wenn  $\mathcal{C} \rightarrow \mathcal{C}'$  mit Resolution, dann ist  $\mathcal{C}$  äquivalent zu  $\mathcal{C}'$ .

Beweis: Betrachte

- $\mathcal{C} = \{D_1, \dots, D_n, E_1, E_2\}$  und  $\mathcal{C}' = \{D_1, \dots, D_n, E_1, E_2, R\}$
- $E_1 = \{A, B_1, \dots, B_m\}$  und  $E_2 = \{\neg A, C_1, \dots, C_{m'}\}$
- $R = \{B_1, \dots, B_m, C_1, \dots, C_{m'}\}$

Zeige: Für jede Interpretation  $I$  gilt:  $I(\mathcal{C}) = I(\mathcal{C}')$ .

Fallunterscheidung:

$I(\mathcal{C}') = 1$ : Dann gilt für alle  $i$ :  $I(D_i) = 1$ ,  $I(E_i) = 1$  und  $I(R) = 1$  und daher auch  $I(\mathcal{C}) = 1$

$I(\mathcal{C}') = 0$ : Wenn es eine Klausel  $D_i$  oder  $E_i$  gibt mit  $I(D_i) = 0$ , dann auch  $I(\mathcal{C}) = 0$ .

Sonst: nur  $I(R) = 0$ , d.h.  $I(B_i) = I(C_j) = 0$  für alle  $i, j$

Betrachte  $I(A)$ :

Wenn  $I(A) = 0$  kann  $I(E_1) = 1$  nicht gelten

Wenn  $I(A) = 1$  kann  $I(E_2) = 1$  nicht gelten

Widerspruch.

## Vollständigkeit (1)

### Satz

Die Resolution auf einer aussagenlogischen Klauselmengen terminiert, wenn man einen Resolutionsschritt nur ausführen darf, wenn sich die Klauselmengen vergrößert.

Beweis: Es gibt nur endlich viele verschiedene Klauseln, da Resolution keine neuen Variablen einführt.

## Vollständigkeit (2)

### Theorem

Für eine unerfüllbare (aussagenlogische) Klauselmengen findet Resolution nach endlich vielen Schritten die leere Klausel.

Beweis: Siehe Skript / Bücher...

D.h.: Für jede Klauselmengen  $\mathcal{C}$  kann Resolution entscheiden:  
 $\mathcal{C}$  widersprüchlich?

**ja** (Klausel  $\emptyset$  wird hergeleitet) oder **nein** (Terminierung ohne  $\emptyset$ ).

## Tautologie-Test

Sei  $F$  beliebige Formel

- 1 Starte mit  $\neg F$
- 2 Transformiere  $\neg F$  in CNF  $\mathcal{C}$   
(auch schnelle CNF erhält Erfüllbarkeit & Widersprüchlichkeit)
- 3 Verwende Resolution beginnend mit  $\mathcal{C}$
- 4 wenn  $\mathcal{C}$  widersprüchlich, dann  $\neg F$  widersprüchlich, und daher  $F$  Tautologie
- 5 wenn  $\mathcal{C}$  erfüllbar, dann  $\neg F$  erfüllbar, und daher  $F$  keine Tautologie

## Herleitung

Zeige  $\{F_1, \dots, F_n\} \models G$ :

- 1 Zeige dass  $(F_1 \wedge \dots \wedge F_n) \implies G$  allgemeingültig  
(Deduktionstheorem)
- 2  $\neg((F_1 \wedge \dots \wedge F_n) \implies G)$  ist äquivalent zu  
 $(F_1 \wedge \dots \wedge F_n \wedge \neg G)$
- 3 Verwende CNF-Algorithmus und anschließend Resolution

Durch Aufzählen kann man alle semantischen Folgerungen auch syntaktisch damit durchführen. Daher: Verfahren ist vollständig.

# Komplexität

Es gibt Formeln, so dass **exponentiell viele** Resolutionen nötig sind, um die leere Klausel herzuleiten (Armin Haken 1985)

Die sogenannten **Taubenschlagformeln**

$TSF^{n+1}$ :  $n + 1$  Tauben sitzen in  $n$  Löchern, wobei in jedem Loch höchstens eine Taube sitzt

$$TSF^{n+1} = \left( \bigwedge_{T=1}^{n+1} \bigvee_{L=1}^n S_L^T \right) \wedge \left( \bigwedge_{T=1}^n \bigwedge_{X=T+1}^{n+1} \bigwedge_{L=1}^n (S_L^T \Rightarrow \neg S_L^X) \right)$$

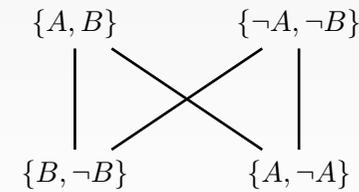
- $S_L^T$  = Taube  $T$  sitzt in Loch  $L$
- $\bigwedge_{T=1}^{n+1} \bigvee_{L=1}^n S_L^T$  = jede Taube  $T$  sitzt in einem der  $n$  Löcher
- $\bigwedge_{T=1}^n \bigwedge_{X=T+1}^{n+1} \bigwedge_{L=1}^n (S_L^T \Rightarrow \neg S_L^X)$  = Jedes Loch einzeln besetzt

# Resolutionsverfahren

**Ein Nachteil:**

Modelle bei erfüllbaren Formeln nicht immer ablesbar

Beispiel:



## Beispiel: Warum will man Modelle finden?

### Die Frage nach dem Pfefferdieb

3 Verdächtige: Hutmacher, Schnapphase, Hasel-Maus

- Genau einer von ihnen ist der Dieb.  
 $(H \vee S \vee M) \wedge (H \Rightarrow \neg(S \vee M)) \wedge (S \Rightarrow \neg(H \vee M)) \wedge (M \Rightarrow \neg(H \vee S))$
- Unschuldige sagen immer die Wahrheit
- Schnapphase: „Der Hutmacher ist unschuldig.“  
 $\neg S \Rightarrow \neg H$
- Hutmacher: „Die Hasel-Maus ist unschuldig“  
 $\neg H \Rightarrow \neg M$

Wer ist der Dieb?

Kodierung:  $H, S, M$ , für „ist schuldig“, Modell liefert Lösung

## Beispiel (Forts.)

Wissen:

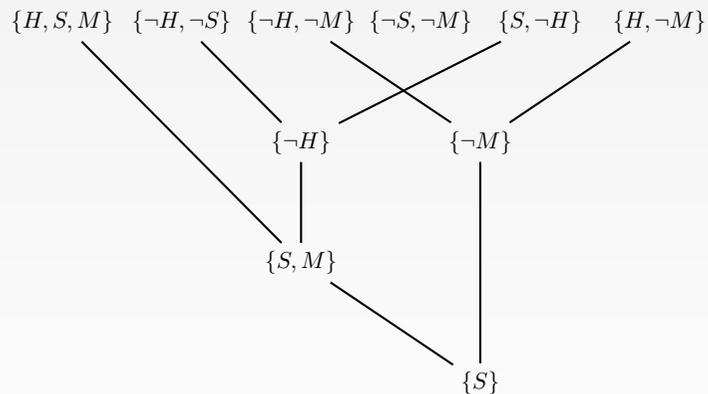
- $H \vee S \vee M$
- $H \Rightarrow \neg(S \vee M)$
- $S \Rightarrow \neg(H \vee M)$
- $M \Rightarrow \neg(H \vee S)$
- $\neg S \Rightarrow \neg H$
- $\neg H \Rightarrow \neg M$

Klauselmengen dazu:

- $\{H, S, M\}$
- $\{\neg H, \neg S\}$
- $\{\neg H, \neg M\}$
- $\{\neg S, \neg M\}$
- $\{S, \neg H\}$
- $\{H, \neg M\}$



## Resolution dazu



- Klauselmengemenge nur erfüllbar, wenn die 1-Klauseln wahr sind.
- Mögliches Modell daher  $S, \neg M, \neg H$
- Prüfung ergibt: Ja ist Modell
- Also ist der Schnapphase schuldig



## Optimierungen des Resolutionsverfahrens

- Bestimmte Klauseln brauchen nicht betrachtet zu werden.;
- diese können **gelöscht**, bzw. brauchen erst gar nicht erzeugt zu werden
- Optimierung, da weniger Klauseln betrachtet werden müssen



## Optimierungen des Resolutionsverfahrens (2)

**Tautologische Klauseln** = Klauseln die  $A$  und  $\neg A$  enthalten

### Satz

Sei  $K$  tautologische Klausel,  $\mathcal{C}$  Klauselmengemenge:  
 $\mathcal{C}$  ist äquivalent zu  $\mathcal{C} \cup \{K\}$

Beweis: Klar, da jede Interpretation  $K$  wahr macht

Im Resolutionsverfahren: **Lösche** tautologische Klauseln

## Optimierungen des Resolutionsverfahrens (3)

Ein Literal  $L_i$  ist **isoliertes Literal** in Klauselmengemenge  $\mathcal{C}$  gdw.  $\bar{L}_i$  nicht in  $\mathcal{C}$  vorkommt, wobei  $\bar{L}_i := \begin{cases} \neg X_i, & \text{falls } L_i = X_i \\ X_i, & \text{falls } L_i = \neg X_i \end{cases}$

### Satz

Sei  $\mathcal{C}$  Klauselmengemenge,  $\mathcal{C}'$  Klauselmengemenge in der alle Klauseln gelöscht sind, die isolierte Literale enthalten.

$\mathcal{C}$  ist **erfüllbarkeits**äquivalent zu  $\mathcal{C}'$ , d.h.  $\mathcal{C}$  ist erfüllbar gdw.  $\mathcal{C}'$  ist erfüllbar.

„ $\Rightarrow$ “: Jede Interpretation die  $\mathcal{C}$  erfüllt, erfüllt auch  $\mathcal{C}'$  (weniger Klauseln).

„ $\Leftarrow$ “: Wenn  $I$  Klauselmengemenge  $\mathcal{C}'$  erfüllt, dann erfüllt  $I'$  Klauselmengemenge  $\mathcal{C}$ , wobei

$$I'(X) = \begin{cases} 1, & \text{für isolierte Literale } L_i = X \\ 0, & \text{für isolierte Literale } L_i = \neg X \\ I(X), & \text{sonst} \end{cases}$$

Im Resolutionsverfahren: **Lösche** Klauseln mit isolierten Literalen

## Beachte ...

Löschen von isolierten Literalen ist **keine** Äquivalenzumformung

Beispiel:  $\{\{A, \neg A\}, \{B\}\}$  ist nicht äquivalent zu  $\{\{A, \neg A\}\}$

- $\{\{A, \neg A\}\}$  ist Tautologie
- $\{\{A, \neg A\}, \{B\}\}$  ist falsifizierbar (mit  $I(B) = 0$ )

Da Resolution nach (Un-)erfüllbarkeit sucht, reicht aber Erfüllbarkeitsäquivalenz aus.

## Optimierungen des Resolutionsverfahrens (4)

Eine Klausel  $K_1$  **subsumiert** Klausel  $K_2$  gdw.  $K_1 \subseteq K_2$ .  
Sprechweise auch: „ $K_2$  wird subsumiert von  $K_1$ “

### Satz

Die Klauselmenge  $\mathcal{C} \cup \{K_1\} \cup \{K_2\}$  ist äquivalent zu  $\mathcal{C} \cup \{K_1\}$ , wenn  $K_1$  die Klausel  $K_2$  subsumiert.

Sei  $I$  Interpretation. Zwei Fälle:

- $I(\mathcal{C} \cup \{K_1\} \cup \{K_2\}) = 1$ . Dann macht  $I$  alle Klauseln wahr, also gilt  $I(\mathcal{C} \cup \{K_1\}) = 1$
- $I(\mathcal{C} \cup \{K_1\} \cup \{K_2\}) = 0$ . Wenn  $I(\mathcal{C} \cup \{K_1\}) = 0$ , dann klar. Sonst nehme an  $I(K_2) = 0$ , aber  $I(\mathcal{C} \cup \{K_1\}) = 1$ . Da  $K_2 \supseteq K_1$  muss aber gelten  $I(K_1) = 0$ . Widerspruch, daher  $I(\mathcal{C} \cup \{K_1\}) = 0$

Im Resolutionsverfahren: **Lösche** subsumierte Klauseln

## Optimierungen des Resolutionsverfahrens (5)

### Zusammenfassend

- Tautologische Klauseln
- Klauseln die isolierte Literale enthalten
- Subsumierte Klauseln

können im Resolutionsverfahren gelöscht werden, da das Finden der leeren Klausel (ja/nein) dadurch nicht verändert wird. (Erfüllbarkeit, Widersprüchlichkeit bleibt gleich)

## Das DPLL-Verfahren

### Namensgebung

- Martin Davis und Hilary Putnam, 1960: Resolution-basiertes Verfahren
- Martin **D**avis, Hilary **P**utnam, George **L**ogemann und Donald W. **L**oveland, 1962: Verbesserung des Verfahrens (insbes. Platz)
- Oft DPLL-Verfahren, aber auch DP-Verfahren, Davis-Putnam-Prozedur

### Grobe Eigenschaften

- Grundlage vieler moderner SAT-Solver
- Kombination aus: Resolution mit Subsumtion und Fallunterscheidung
- Verwendet Backtracking
- Kann Modelle für erfüllbare Klauseln einfach erzeugen

# DPLL-Algorithmus (1)

## Eingabe:

- Klauselmenge  $\mathcal{C}$
- Annahme: Tautologische Klauseln bereits entfernt

## Ausgabe:

- true, wenn  $\mathcal{C}$  unerfüllbar
- false, wenn  $\mathcal{C}$  erfüllbar
- Erweiterung: auch ein Modell wenn  $\mathcal{C}$  erfüllbar

DPLL beantwortet daher die Frage:  
Ist  $\mathcal{C}$  widersprüchlich?  
Antwortet Ja bzw. Nein.



# DPLL-Algorithmus (2)



## Algorithmus DPLL-Prozedur

### Funktion DPLL( $\mathcal{C}$ ):

```

if  $\emptyset \in \mathcal{C}$  then return true; // unerfüllbar
if  $\mathcal{C} = \emptyset$  then return false; // erfüllbar
if  $\mathcal{C}$  enthält 1-Klausel  $\{L\}$  then
    Sei  $\mathcal{C}'$  die Klauselmenge, die aus  $\mathcal{C}$  entsteht, indem
        (1) alle Klauseln, die  $L$  enthalten, entfernt werden und
        (2) in den verbleibenden Klauseln alle Literale  $\bar{L}$  entfernt werden
            (wobei  $\bar{L} = \neg X$ , wenn  $L = X$  und  $\bar{L} = X$ , wenn  $L = \neg X$ )
    DPLL( $\mathcal{C}'$ ); // rekursiver Aufruf
if  $\mathcal{C}$  enthält isoliertes Literal  $L$  then
    Sei  $\mathcal{C}'$  die Klauselmenge, die aus  $\mathcal{C}$  entsteht, indem alle Klauseln,
    die  $L$  enthalten, entfernt werden.
    DPLL( $\mathcal{C}'$ ); // rekursiver Aufruf
// Nur wenn keiner der obigen Fälle zutrifft:
Wähle Atom  $A$ , das in  $\mathcal{C}$  vorkommt;
return DPLL( $\mathcal{C} \cup \{\{A\}\}$ )  $\wedge$  DPLL( $\mathcal{C} \cup \{\{\neg A\}\}$ ) // Fallunterscheidung
    
```



# DPLL-Prozedur: Erläuterungen

## **if** $\mathcal{C}$ enthält 1-Klausel $\{L\}$ **then**

Sei  $\mathcal{C}'$  die Klauselmenge, die aus  $\mathcal{C}$  entsteht, indem

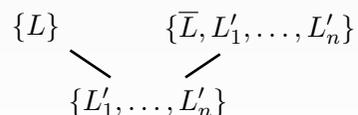
- (1) alle Klauseln, die  $L$  enthalten, entfernt werden und
- (2) in den verbleibenden Klauseln alle Literale  $\bar{L}$  entfernt werden  
(wobei  $\bar{L} = \neg X$ , wenn  $L = X$  und  $\bar{L} = X$ , wenn  $L = \neg X$ )

- (1) entspricht der Subsumtion:

$$\{A\}, \dots, \{A, B, C\}, \dots$$

entferne  $\{A, B, C\}$ , da  $\{A\}$  die Klausel  $\{A, B, C\}$  subsumiert.

- (2) entspricht der Resolution mit anschließender Subsumtion



- Entfernen von  $\{A\}$ : „Setze  $I(A) = 1$ “, sonst sowieso nicht erfüllbar

# DPLL-Prozedur: Erläuterungen (2)



## **if** $\mathcal{C}$ enthält isoliertes Literal $L$ **then**

Sei  $\mathcal{C}'$  die Klauselmenge, die aus  $\mathcal{C}$  entsteht, indem alle Klauseln, die  $L$  enthalten, entfernt werden.

DPLL( $\mathcal{C}'$ ); // rekursiver Aufruf

Entspricht der Löschrregel für isolierte Literale

// Nur wenn keiner der obigen Fälle zutrifft:

Wähle Atom  $A$ , das in  $\mathcal{C}$  vorkommt;

**return** DPLL( $\mathcal{C} \cup \{\{A\}\}$ )  $\wedge$  DPLL( $\mathcal{C} \cup \{\{\neg A\}\}$ ) // Fallunterscheidung

Fallunterscheidung, ob  $A$  wahr oder falsch ist (bei erfüllbar)

bei Unerfüllbarkeitsfrage:  $\mathcal{C}$  muss für  $A$  wahr und  $\neg A$  jeweils unerfüllbar sein.

# DPLL in Haskell

```

dpll :: [[Int]] -> Bool
dpll [] = False
dpll cnf
  | [] 'elem' cnf = True
  | otherwise =
    case getUnit cnf of
      Just [x] ->
        dpll [delete (negate x) clause | clause <- cnf, not (x 'elem' clause)]
      Nothing ->
        if not (null isolated) then
          dpll [clause | clause <- cnf, let (isolit:_)=isolated,
                                           not (isolit 'elem' clause)]
        else (dpll ([lit]:cnf)) && (dpll ([negate lit]:cnf))
    where
      literals = (nub . sort . concat) cnf
      isolated = [lit | lit <- literals, not ((negate lit) 'elem' literals)]
      ((lit:clause):_) = cnf
      getUnit []       = Nothing
      getUnit ([x]:_)  = Just [x]
      getUnit (_:xxs)  = getUnit xxs
    
```



# Fallunterscheidung

- Verschiedene Heuristiken **welches Literal** gewählt wird.
- Gute Heuristik:
  - Wähle Literal so, dass es in möglichst kurzen Klauseln vorkommt
- Erhöht Wahrscheinlichkeit, dass große Anteile der Klauselmenge in wenigen Schritten gelöscht werden.



# Beispiel: Pfefferdieb

## Wissen:

- $H \vee S \vee M$
- $H \Rightarrow \neg(S \vee M)$
- $S \Rightarrow \neg(H \vee M)$
- $M \Rightarrow \neg(H \vee S)$
- $\neg S \Rightarrow \neg H$
- $\neg H \Rightarrow \neg M$

## Klauselmenge dazu:

- $\{H, S, M\}$
- $\{\neg H, \neg S\}$
- $\{\neg H, \neg M\}$
- $\{\neg S, \neg M\}$
- $\{S, \neg H\}$
- $\{H, \neg M\}$

# Beispiel (2)

DPLL( $\{\{H, S, M\}, \{\neg H, \neg S\}, \{\neg H, \neg M\}, \{\neg S, \neg M\}, \{S, \neg H\}, \{H, \neg M\}\}$ )

- Keine 1-Klauseln
- Keine isolierten Literale
- Daher Fallunterscheidung

- (1) DPLL( $\{\{S\}, \{H, S, M\}, \{\neg H, \neg S\}, \{\neg H, \neg M\}, \{\neg S, \neg M\}, \{S, \neg H\}, \{H, \neg M\}\}$ )
- (2) DPLL( $\{\{\neg S\}, \{H, S, M\}, \{\neg H, \neg S\}, \{\neg H, \neg M\}, \{\neg S, \neg M\}, \{S, \neg H\}, \{H, \neg M\}\}$ )

## Beispiel (3)

Fall (1):

DPLL( $\{\{S\}, \{H, S, M\}, \{\neg H, \neg S\}, \{\neg H, \neg M\}, \{\neg S, \neg M\}, \{S, \neg H\}, \{H, \neg M\}\}$ )

- 1-Klausel  $\{S\}$
- entferne alle Klauseln die  $\{S\}$  enthalten:  
 $\{\{S\}, \{H, S, M\}, \{\neg H, \neg S\}, \{\neg H, \neg M\}, \{\neg S, \neg M\}, \{S, \neg H\}, \{H, \neg M\}\}$

ergibt  $\{\{\neg H, \neg S\}, \{\neg H, \neg M\}, \{\neg S, \neg M\}, \{H, \neg M\}\}$

entferne  $\neg S$  aus allen Klauseln:  $\{\{\neg H, \neg S\}, \{\neg H, \neg M\}, \{\neg S, \neg M\}, \{H, \neg M\}\}$

ergibt  $\{\{\neg H\}, \{\neg H, \neg M\}, \{\neg M\}, \{H, \neg M\}\}$

Rekursiver Aufruf: DPLL( $\{\{\neg H\}, \{\neg H, \neg M\}, \{\neg M\}, \{H, \neg M\}\}$ )

DPLL( $\{\{\neg H\}, \{\neg H, \neg M\}, \{\neg M\}, \{H, \neg M\}\}$ )

- 1-Klausel  $\{\neg H\}$
- Entfernen der Klauseln mit  $\neg H$  und Löschen von  $H$  ergibt  $\{\{\neg M\}\}$   
 Rekursiver Aufruf: DPLL( $\{\{\neg M\}\}$ )

DPLL( $\{\{\neg M\}\}$ ) ergibt rekursiver Aufruf: DPLL( $\{\}$ )

DPLL( $\{\}$ ) ergibt `false` → erfüllbar, Modell?



## DPLL: Modell generieren

- 1 Isolierte Literale werden als wahr angenommen.
- 2 Literale in 1-Klauseln werden ebenfalls als wahr angenommen.
- 3 Dadurch nicht belegte Variablen können für das vollständige Modell beliebig belegt werden



## Komplexität

Im Worst Case:

- Exponentielle Laufzeit
- Exponentiell in: **Anzahl der verschiedenen Variablen**

Besser geht es nach aktuellem Stand des Wissens nicht, da SAT  $\mathcal{NP}$ -vollständig.

und: DPLL löst auch  $\text{co}\mathcal{NP}$ -vollständige Probleme

## DPLL zum Problemlösen

- Idee: Kodiere Problem als Aussagenlogische Formel
- Modell entspricht Lösung des Problems
- Erst Umformung in CNF (geht mit schneller CNF), dann DPLL
- Oft: Kodierung direkt als CNF
- Wir betrachten einige Beispielanwendungen

## Logelei aus der Zeit

Abianer sagen die Wahrheit, Bebianer Lügen.

- 1 Knasi: Knasi ist Abianer.  
 $knasi \iff knisi$
- 2 Knesi: Wenn Knösi Bebianer, dann ist auch Knusi ein Abianer.  
 $knesi \iff (\neg knoesi \implies knusi)$
- 3 Knisi: Wenn Knusi Abianer, dann ist Knesi Bebianer.  
 $knisi \iff (knusi \implies \neg knesi)$
- 4 Knosi: Knesi und Knüsi sind beide Abianer.  
 $knosi \iff (knesi \wedge knuesi)$
- 5 Knusi: Wenn Knüsi Abianer ist, dann ist auch Knisi Abianer.  
 $knusi \iff (knuesi \implies knisi)$
- 6 Knösi: Entweder ist Knasi oder Knisi Abianer.  
 $knoesi \iff (knasi \text{ XOR } knisi)$
- 7 Knüsi: Knosi ist Abianer.  
 $knuesi \iff knosi$



## DPLL-Algorithmus liefert:

```
(knasi <=> knisi)
/\
(knesi <=> (-knoesi => knusi))
/\
(knisi <=> (knusi => -knesi))
/\
(knosi <=> (knesi /\ knuesi))
/\
(knusi <=> (knuesi => knisi))
/\
(knoesi <=> (-(knasi <=> knisi)))
/\
(knuesi <=> knosi)
```

Das Ergebnis des DP-Algorithmus ist:  
Fuer die berechnete Klauselmeng existiert ein Modell:  
[-knuesi,-knosi,-knoesi,-knasi,knesi, knusi,-knisi]

Knesi und Knusi sind Abianer,  
Knüsi, Knosi, Knösi, Knisi sind Bebianer



## Diebstahl von Salz

Verdächtige: Lakai mit Froschgesicht, Lakai mit Fischgesicht, Herzbube.

- Frosch: der Fisch wars  
 $froschSagtWahrheit \implies fischIstDieb$
- Fisch: ich wars nicht  
 $fischSagtWahrheit \implies \neg fischIstDieb$
- Herzbube: ich wars  
 $bubeSagtWahrheit \implies bubeIstDieb$
- Genau einer ist der Dieb  
 $(fischIstDieb \wedge \neg froschIstDieb \wedge \neg bubeIstDieb) \vee (\neg fischIstDieb \wedge froschIstDieb \wedge \neg bubeIstDieb) \vee (\neg fischIstDieb \wedge \neg froschIstDieb \wedge bubeIstDieb)$
- höchstens einer hat gelogen  
 $(\neg froschSagtWahrheit \implies froschSagtWahrheit \wedge bubeSagtWahrheit) \wedge (\neg fischSagtWahrheit \implies froschSagtWahrheit \wedge bubeSagtWahrheit) \wedge (\neg bubeSagtWahrheit \implies froschSagtWahrheit \wedge fischSagtWahrheit)$

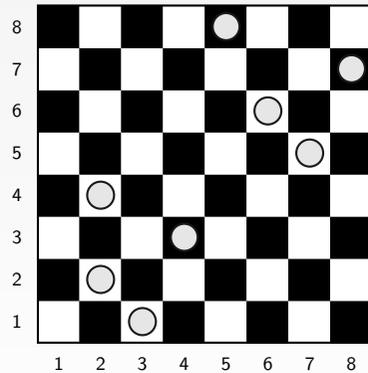
## DPLL-Algorithmus liefert ...

```
(froschSagtWahrheit => fischIstDieb)
/\
(fischSagtWahrheit => -fischIstDieb)
/\
(bubeSagtWahrheit => bubeIstDieb)
/\
( (fischIstDieb /\ -froschIstDieb /\ -bubeIstDieb)
  \/\ (-fischIstDieb /\ froschIstDieb /\ -bubeIstDieb)
  \/\ (-fischIstDieb /\ -froschIstDieb /\ bubeIstDieb) )
/\
( (-froschSagtWahrheit => froschSagtWahrheit /\ bubeSagtWahrheit)
  /\ (-fischSagtWahrheit => froschSagtWahrheit /\ bubeSagtWahrheit)
  /\ (-bubeSagtWahrheit => froschSagtWahrheit /\ fischSagtWahrheit) )
```

Das Ergebnis des DP-Algorithmus ist:  
Fuer die berechnete Klauselmeng existiert ein Modell:  
[-froschIstDieb,bubeIstDieb,bubeSagtWahrheit,froschSagtWahrheit,-froschSagtWahrheit,-fischIstDieb]

$\implies$  Herzbube ist der Dieb, und Lakai mit Froschgesicht hat gelogen

## N-Damen als SAT-Problem



## N-Damen als SAT-Problem (2)

Direkte Kodierung als CNF, Aussagenlogische Variablen sind Zahlen (negative Zahlen = negierte Literale)

```
nDamen n =
  (proZeileEineDame n)
  ++ (proSpalteEineDame n)
  ++ (bedrohendePaare n)
```

koordinateZuZahl (x,y) n = (x-1)\*n+y

```
proZeileEineDame n =
  [[koordinateZuZahl (i,j) n | j <- [1..n]] | i <- [1..n]]
```

```
proSpalteEineDame n =
  [[koordinateZuZahl (i,j) n | i <- [1..n]] | j <- [1..n]]
```



## N-Damen als SAT-Problem (2)

```
bedrohendePaare n =
  [[negate (koordinateZuZahl (x1,y1) n),
    negate (koordinateZuZahl (x2,y2) n)]
   | x1 <- [1..n],
     y1 <- [1..n],
     x2 <- [1..n],
     y2 <- [1..n],
     (x1,y1) < (x2,y2),
     bedroht (x1,y1,x2,y2)]
```

```
bedroht (a,x,b,y)
  | a == b = True
  | x == y = True
  | abs (a-b) == abs (y-x) = True
  | otherwise = False
```

## N-Damen als SAT-Problem (3)

```
*Main> nDamen 4
[[1,2,3,4], [5,6,7,8], [9,10,11,12], [13,14,15,16],
 [1,5,9,13], [2,6,10,14], [3,7,11,15], [4,8,12,16],
 [-1,-2], [-1,-3], [-1,-4], [-1,-5], [-1,-6], [-1,-9], [-1,-11], [-1,-13], [-1,-16],
 [-2,-3], [-2,-4], [-2,-5], [-2,-6], [-2,-7], [-2,-10], [-2,-12], [-2,-14], [-3,-4],
 [-3,-6], [-3,-7], [-3,-8], [-3,-9], [-3,-11], [-3,-15], [-4,-7], [-4,-8], [-4,-10],
 [-4,-12], [-4,-13], [-4,-16], [-5,-6], [-5,-7], [-5,-8], [-5,-9], [-5,-10], [-5,-13],
 [-5,-15], [-6,-7], [-6,-8], [-6,-9], [-6,-10], [-6,-11], [-6,-14], [-6,-16], [-7,-8],
 [-7,-10], [-7,-11], [-7,-12], [-7,-13], [-7,-15], [-8,-11], [-8,-12], [-8,-14],
 [-8,-16], [-9,-10], [-9,-11], [-9,-12], [-9,-13], [-9,-14], [-10,-11], [-10,-12],
 [-10,-13], [-10,-14], [-10,-15], [-11,-12], [-11,-14], [-11,-15], [-11,-16],
 [-12,-15], [-12,-16], [-13,-14], [-13,-15], [-13,-16], [-14,-15], [-14,-16], [-15,-16]]
```

DPLL liefert Modell: [-13,-16,15,9,-11,-14,-12,-10,8,-7,-6,-5,-4,-3,2,-1]

Tatsächlich gibt es zwei Modelle:

```
*DPexamples> davisPutnamAlle (generate_nqueens 4)
[[-13,-16,15,9,-11,-14,-12,-10,8,-7,-6,-5,-4,-3,2,-1],
 [-15,-13,-9,-8,-6,-4,-2,-1,5,12,14,3,-16,-10,-7,-11]]
```

## Nützliche Generatoren: Mindestens eine wahr



Sei  $S = \{F_1, \dots, F_n\}$  eine Menge von Formeln.

$$at\_least\_one(S) = (F_1 \vee \dots \vee F_n)$$

Z.B.  $at\_least\_one(\{X_1, X_2, X_3\}) = (X_1 \vee X_2 \vee X_3)$

## Nützliche Generatoren: Genau eine wahr



Sei  $S = \{F_1, \dots, F_n\}$  eine Menge von Formeln.

$$exactly\_one(S) = at\_least\_one(S) \wedge at\_most\_one(S)$$

Z.B.

$$at\_least\_one(\{X_1, X_2, X_3\}) = (X_1 \vee X_2 \vee X_3)$$

$$at\_most\_one(\{X_1, X_2, X_3\}) = (\neg X_1 \vee \neg X_2) \wedge (\neg X_1 \vee \neg X_3) \wedge (\neg X_2 \vee \neg X_3)$$

$$exactly\_one(\{X_1, X_2, X_3\}) = (X_1 \vee X_2 \vee X_3) \wedge (\neg X_1 \vee \neg X_2) \wedge (\neg X_1 \vee \neg X_3) \wedge (\neg X_2 \vee \neg X_3)$$

## Nützliche Generatoren: Höchstens eine wahr



Sei  $S = \{F_1, \dots, F_n\}$  eine Menge von Formeln.

$$at\_most\_one(S) = \bigwedge \{(\neg F_i \vee \neg F_j) \mid i \in \{1, \dots, n\}, j \in \{1, \dots, n\}, i \neq j\}$$



Z.B.

$$at\_most\_one(\{X_1, X_2, X_3\}) = (\neg X_1 \vee \neg X_2) \wedge (\neg X_1 \vee \neg X_3) \wedge (\neg X_2 \vee \neg X_3) \wedge (\neg X_2 \vee \neg X_3) \wedge (\neg X_3 \vee \neg X_1) \wedge (\neg X_3 \vee \neg X_2)$$

Beachte: Man kann noch optimieren (Symmetrien entfernen):

$$at\_most\_one(S) = \bigwedge \{(\neg F_i \vee \neg F_j) \mid i \in \{1, \dots, n\}, j \in \{1, \dots, n\}, i < j\}$$

Z.B.

$$at\_most\_one(\{X_1, X_2, X_3\}) = (\neg X_1 \vee \neg X_2) \wedge (\neg X_1 \vee \neg X_3) \wedge (\neg X_2 \vee \neg X_3)$$

## Beispiel: Klausuren verteilen



- $s$  Studenten schreiben Klausuren
- $k$  Klausurtypen
- Paare von benachbart sitzenden Studenten gegeben
- Problem: Welcher Student bekommt welchen Klausurtyp
- So dass: Keine benachbarten Studenten bekommen gleiche Klausur

Kodierte in CNF, so dass Modell eine Zuordnung: Student  $\leftrightarrow$  Klausurtyp liefert

## Beispiel: Klausuren verteilen

$s_i^j =$  Student  $i$  schreibt Klausurtyp  $j$

$klausur\_formel(s, k, benachbart) =$

Jeder Student erhält genau eine Klausur:

$$\bigwedge_{i=1}^s \underbrace{exactly\_one(\{s_i^j \mid j \in \{1..k\}\})}_{\text{Student } i \text{ genau eine Klausur}}$$

alle Studenten

Benachbarte Studenten, nicht die gleiche Klausur:

$$\bigwedge_{(a,b) \in benachbart} \bigwedge_{j=1}^k (\neg s_a^j \vee \neg s_b^j)$$

## Verallgemeinerung: $K$ aus $N$

- $at\_most(K, S)$ : höchstens  $K$  viele Formeln aus  $S$  erfüllt
- $at\_least(K, S)$ : mindestens  $K$  viele Formeln aus  $S$  erfüllt
- $exactly(K, S)$ : genau  $K$  viele Formeln aus  $S$  erfüllt

Vorarbeit: Alle Teilmengen der Mächtigkeit  $K$

$$all\_subsets(S, K) = \{S' \subseteq S \mid |S'| = K\}$$

Z.B. Rekursive Berechnung:

$$\begin{aligned} all\_subsets(S, 0) &= \{\{\}\} \\ all\_subsets(S, K) &= \{S\}, \text{ wenn } |S| = K \\ all\_subsets(\{s\} \cup S, K) &= all\_subsets(S, K) \\ &\quad \cup \{\{s\} \cup S' \mid S' \in all\_subsets(S, K-1)\} \end{aligned}$$

## Höchstens $K$ wahr

$$at\_most(K, S) = \bigwedge \left\{ \underbrace{\neg(\bigwedge S')}_{\text{nicht alle wahr}} \mid \underbrace{S'}_{K+1 \text{ Formeln}} \in all\_subsets(S, K+1) \right\}$$

$$= \bigwedge \left\{ \underbrace{\neg(F_1 \wedge \dots \wedge F_{K+1})}_{\text{nicht alle wahr}} \mid \{F_1, \dots, F_{K+1}\} \in all\_subsets(S, K+1) \right\}$$

$$= \bigwedge \left\{ \underbrace{(\neg F_1 \vee \dots \vee \neg F_{K+1})}_{\text{mind. 1 falsch}} \mid \{F_1, \dots, F_{K+1}\} \in all\_subsets(S, K+1) \right\}$$

In CNF (wenn  $F_i$  Literale sind)

## Mindestens $K$ wahr

$$at\_least(K, S) = \bigvee \left\{ \underbrace{\bigwedge S'}_{K \text{ wahr}} \mid S' \in all\_subsets(S, K) \right\}$$

Z.B.

$$at\_least(3, \{X_1, X_2, X_3, X_4\}) = (X_1 \wedge X_2 \wedge X_3) \vee (X_1 \wedge X_2 \wedge X_4) \vee (X_1 \wedge X_3 \wedge X_4) \vee (X_2 \wedge X_3 \wedge X_4)$$

Nachteil: Nicht in CNF (wenn  $S$  nur Literale enthält)

Mindestens  $K$  wahr (2)

$$\begin{aligned}
 &at\_least(K, S) \\
 &= at\_least\_one(S) \\
 &\quad \wedge \bigwedge \{f \Rightarrow \bigvee S' \mid f \in S, S' \in all\_subsets(S \setminus \{f\}, |S| - (K - 1))\} \\
 &= at\_least\_one(S) \\
 &\quad \wedge \bigwedge \{\neg f \vee \bigvee S' \mid f \in S, S' \in all\_subsets(S \setminus \{f\}, |S| - (K - 1))\}
 \end{aligned}$$



Idee dabei: mind. 1 Formel aus  $S$  wahr,  
und wenn eine wahr, dann noch  $K - 1$  weitere wahr...

Z.B.

$$\begin{aligned}
 &at\_least(3, \{X_1, X_2, X_3, X_4\}) = \\
 &(X_1 \vee X_2 \vee X_3 \vee X_4) \\
 &\wedge (X_1 \Rightarrow (X_2 \vee X_3)) \wedge (X_1 \Rightarrow (X_2 \vee X_4)) \wedge (X_1 \Rightarrow (X_3 \vee X_4)) \\
 &\wedge (X_2 \Rightarrow (X_1 \vee X_3)) \wedge (X_2 \Rightarrow (X_1 \vee X_4)) \wedge (X_2 \Rightarrow (X_3 \vee X_4)) \\
 &\wedge (X_3 \Rightarrow (X_1 \vee X_2)) \wedge (X_3 \Rightarrow (X_1 \vee X_4)) \wedge (X_3 \Rightarrow (X_2 \vee X_4)) \\
 &\wedge (X_4 \Rightarrow (X_1 \vee X_2)) \wedge (X_4 \Rightarrow (X_1 \vee X_3)) \wedge (X_4 \Rightarrow (X_2 \vee X_3))
 \end{aligned}$$

Genau  $K$  wahr

$$exactly(K, S) = at\_least(K, S) \wedge at\_most(K, S)$$

## Beispiel: Logelei aus der Zeit



Tom, ein Biologiestudent, sitzt verzweifelt in der Klausur, denn er hat vergessen, das Kapitel über die Wolfswürmer zu lernen. Das Einzige, was er weiß, ist, dass es bei den Multiple-Choice-Aufgaben immer **genau 3 korrekte** Antworten gibt. Und dies sind die angebotenen Antworten:

- Wolfswürmer werden oft von Igelwürmern gefressen.
- Wolfswürmer meiden die Gesellschaft von Eselswürmern.
- Wolfswürmer ernähren sich von Lammwürmern.
- Wolfswürmer leben in der banesischen Tundra.
- Wolfswürmer gehören zur Gattung der Hundswürmer.
- Wolfswürmer sind grau gestreift.
- Genau eine der beiden Aussagen b) und e) ist richtig.
- Genau eine der beiden Aussagen a) und d) ist richtig.
- Genau eine der beiden Aussagen c) und h) ist richtig.
- Genau eine der beiden Aussagen f) und i) ist richtig.
- Genau eine der beiden Aussagen c) und d) ist richtig.
- Genau eine der beiden Aussagen d) und h) ist richtig.

Können Sie Tom helfen, die richtigen Aussagen herauszufinden?

## Beispiel: Logelei aus der Zeit

A, B, ..., L = entsprechende Aussage ist wahr

- Genau eine der beiden Aussagen b) und e) ist richtig.  
 $G \iff B \text{ XOR } E$
  - Genau eine der beiden Aussagen a) und d) ist richtig.  
 $H \iff A \text{ XOR } D$
  - Genau eine der beiden Aussagen c) und h) ist richtig.  
 $I \iff C \text{ XOR } H$
  - Genau eine der beiden Aussagen f) und i) ist richtig.  
 $J \iff F \text{ XOR } I$
  - Genau eine der beiden Aussagen c) und d) ist richtig.  
 $K \iff C \text{ XOR } D$
  - Genau eine der beiden Aussagen d) und h) ist richtig.  
 $L \iff D \text{ XOR } H$
- genau 3 korrekte Antworten  
 $exactly(3, \{A, B, C, D, E, F, G, H, I, J, K, L\})$

# Lösen mit DPLL

Direkte Kodierung mit 1,...,12 statt A,...,B ergibt:

```
*DPexamples> davisPutnam cwuermer
[-11,-10,-9,-7,-6,-5,-2,3,-12,-1,8,4]
```

D.h. 3,4,8 sind wahr = C,D,H sind wahr

- c) Wolfswürmer ernähren sich von Lammwürmern.
- d) Wolfswürmer leben in der banesischen Tundra.
- h) Genau eine der beiden Aussagen a) und d) ist richtig.

# Sudoku-Lösen mit DPLL

**Sudoku:**

7	9	6	4	8	1	5	3	2
1	2	3	6	5	9	8	4	7
5	8	4	3	2	7	6	9	1
4	6	9	8	1	2	7	4	3
8	3	5	7	9	4	1	2	6
2	1	7	5	6	3	9	8	4
9	4	1	2	7	5	3	6	8
3	7	8	9	4	6	2	1	5
6	5	2	1	3	8	4	7	9

# Kodierung von Sudokus als Klauselmengen

- Direkt Ganzzahlen als Variablennamen
- Negative Zahlen = negierte Variable
- Dreistellige Zahlen  $XYV \in \{111, 112, \dots, 999\}$
- $X$  = Zeile
- $Y$  = Spalte
- $V$  = Zahl die in der Zelle stehen kann in  $\{1, \dots, 9\}$
- D.h. pro Zelle 9 Variablen, von den genau eine Wahr sein muss
- Insgesamt 729 Variablen

# Kodierung von Sudokus als Klauselmengen

**Eingabe:** Sudoku (teilweise gefüllt)

**Erzeuge Klauselmengen:**

- Klauselmengen =
- Startbelegung
  - ∪ FelderEindeutigBelegt
  - ∪ ZeilenBedingung
  - ∪ SpaltenBedingung
  - ∪ QuadratBedingung

Startbelegung:

- Gegebene Zahlen werden auf wahr gesetzt:  
Startbelegung: Wenn in Feld  $(x, y)$  die Zahl  $z$  steht:  
Füge 1-Klausel  $\{xyz\}$  hinzu



# Kodierung von Sudokus als Klauselmenge

$$\text{FelderEindeutigBelegt} = \bigwedge_{X=1}^9 \bigwedge_{Y=1}^9 \text{exactly\_one}(\{XY1, \dots, XY9\})$$

$$\text{ZeilenBedingung} = \bigwedge_{X=1}^9 \bigwedge_{V=1}^9 \text{at\_most\_one}(\{X1V, \dots, X9V\})$$

$$\text{SpaltenBedingung} = \bigwedge_{Y=1}^9 \bigwedge_{V=1}^9 \text{at\_most\_one}(\{1YV, \dots, 9YV\})$$



# Mögliche Anwendung: Produktkonfigurator / Auto-Konfigurator

“Die Produktkonfiguration beschreibt das Zusammensetzen eines Produktes aus vorgegebenen Produktkomponenten (sogenannte **Selektion und Kombination**) und die Selektion inhaltlicher Ausprägungen der Komponenteneigenschaften (sogenannte **Parametrisierung**) unter Einhaltung der **Konfigurationsregeln**. Die **Konfigurationsmöglichkeiten** ergeben sich aus den Selektions-, Kombinations- und Parametrisierungsmöglichkeiten eines Produktes eingeschränkt durch die **Konfigurationsregeln**”



# Kodierung von Sudokus als Klauselmenge (2)

QuadratBedingung =

$$\bigwedge_{V=1}^9 \left( \begin{array}{l} \text{at\_most\_one}(\{11V, 12V, 13V, 21V, 22V, 23V, 31V, 32V, 33V\}) \wedge \\ \text{at\_most\_one}(\{14V, 15V, 16V, 24V, 25V, 26V, 34V, 35V, 36V\}) \wedge \\ \text{at\_most\_one}(\{17V, 18V, 19V, 27V, 28V, 29V, 37V, 38V, 39V\}) \wedge \\ \text{at\_most\_one}(\{41V, 42V, 43V, 51V, 52V, 53V, 61V, 62V, 63V\}) \wedge \\ \text{at\_most\_one}(\{44V, 45V, 46V, 54V, 55V, 56V, 64V, 65V, 66V\}) \wedge \\ \text{at\_most\_one}(\{47V, 48V, 49V, 57V, 58V, 59V, 67V, 68V, 69V\}) \wedge \\ \text{at\_most\_one}(\{71V, 72V, 73V, 81V, 82V, 83V, 91V, 92V, 93V\}) \wedge \\ \text{at\_most\_one}(\{74V, 75V, 76V, 84V, 85V, 86V, 94V, 95V, 96V\}) \wedge \\ \text{at\_most\_one}(\{77V, 78V, 79V, 87V, 88V, 89V, 97V, 98V, 99V\}) \end{array} \right)$$



# Mögliche Anwendung: Produktkonfigurator / Auto-Konfigurator

Aussagenlogik und DPLL-Beweiser mit Modellausgabe kann hilfreich sein:

Es gibt nur **endlich** viele Möglichkeiten:

- Farben
- Ja/nein Entscheidungen: Klimaanlage j/n: Ja /nein
- Liste von Varianten (zB Motoren)
- Reeller Wert (zB PS-Zahl): kann diskretisiert werden.

Schwieriger:

- Preis Obergrenze: Durch Addieren der Komponentenpreise

Mögliche Funktionen:

- Überprüfung der Konfigurationsregeln
- Ausgabe der möglichen Varianten.
- Testen des Systems, z.B bei Erweiterung / Anpassungen