

Automatisches Beweisen

– Kap. 4.1: Aussagenlogik –

April 2011

Prof. Dr. Wolfgang Küchlin

Dipl.-Inform., Dr. sc. techn. (ETH)

Arbeitsbereich Symbolisches Rechnen

Wilhelm-Schickard-Institut für Informatik

Fakultät für Informations- und Kognitionswissenschaften

Universität Tübingen

Steinbeis Transferzentrum

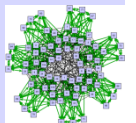
Objekt- und Internet-Technologien (OIT)

Wolfgang.Kuechlin@uni-tuebingen.de

<http://www-sr.informatik.uni-tuebingen.de>

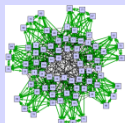


SR



Logischer Formalismus

- **Syntax:** Wie werden Formeln gebildet?
 - Typisch: nach bestimmten Regeln gebildete Zeichenreihen
- **Semantik:** Was ist die Bedeutung einer Formel?
 - Wie kann der Wahrheitswert einer Formel bestimmt werden?
- **Kalkül:** Nach welchen Regeln können aus wahren Formeln weitere wahre Formeln abgeleitet werden?
 - Inferenzregeln / Deduktion



Aussagenlogik: Syntax

➤ Bestandteile von Formeln:

- **Aussagenvariablen:** (x, y, z, \dots) : Platzhalter für beliebige (atomare) Aussagen, wie z.B. „5 ist eine Primzahl“, „ $2 > 3$ “
- **Konstanten:** \perp [false] und \top [true] (oder auch f, t)
- **Junktoren:** mindestens \neg [„nicht“], \wedge [„und“], \vee [„oder“]
Nach Bedarf weitere, z.B.: \rightarrow , \leftrightarrow , \oplus
Zur Bildung komplexer Formeln, die zusammengesetzte Aussagen repräsentieren.
- **Hilfssymbole:** (Klammern)

➤ Formal: Formeln = (bestimmte) Zeichenreihen



Aussagenlogik: Syntax (2)

- Aussagenlogische Formeln (induktiv definiert):
 - Alle Aussagenvariablen sind Formeln
 - \perp und \top sind Formeln
 - Falls F und G Formeln, so auch $(\neg F)$, $(F \wedge G)$, $(F \vee G)$, $(F \rightarrow G)$, $(F \leftrightarrow G)$, $(F \oplus G)$
 - Nichts sonst ist eine Formel
- Beispiele:
 - $((\neg x) \vee (y \wedge z)) \vee \perp$
 - $((x \wedge \neg(y \rightarrow z)) \oplus x)$
- Präzedenz: (abnehmende Bindungsstärke)
 - $\neg, \wedge, \vee, \rightarrow, \leftrightarrow, \oplus$



Aussagenlogik: Begriffe und Symbole

- $\mathcal{B} = \{0, 1\}$: Menge der (booleschen) **Wahrheitswerte**
(0: falsch, 1: wahr)
- $\mathcal{V} = \{x_0, \dots, x_n, \dots, x, y, z, \dots\}$: Menge der
Aussagenvariablen
- $\mathcal{L} = \Phi_0 \cup \neg \Phi_0 = \mathcal{V} \cup \{\neg x \mid x \in \mathcal{V}\}$: Menge der **Literale**
- **Var(F)**: Menge der in F vorkommenden Variablen
 - Z.B:
$$\text{Var}(((x \wedge \neg(y \rightarrow z)) \oplus x)) = \{x, y, z\}$$



Semantik von Formeln (allgemein)

- Wie wird die Bedeutung einer Formel bestimmt?
- Bilde die Menge der neuen (unbekannten) Formeln \mathcal{F} ab in eine Menge bekannter (berechenbarer) Formeln \mathcal{B}
- Semantische Abbildung zur Interpretation (Bewertung / Valuation v , Bedeutungsfunktion β , meaning function μ)

$$\beta : \mathcal{F} \rightarrow \mathcal{B}$$

- Problem: endliche Definition dieser Abbildung
- Lösung: induktive Definition über Formelaufbau in \mathcal{F}
 - definiere Abbildung der Elementarsymbole
 - definiere Abbildung der Operatoren
 - Induktion

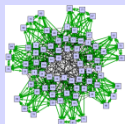
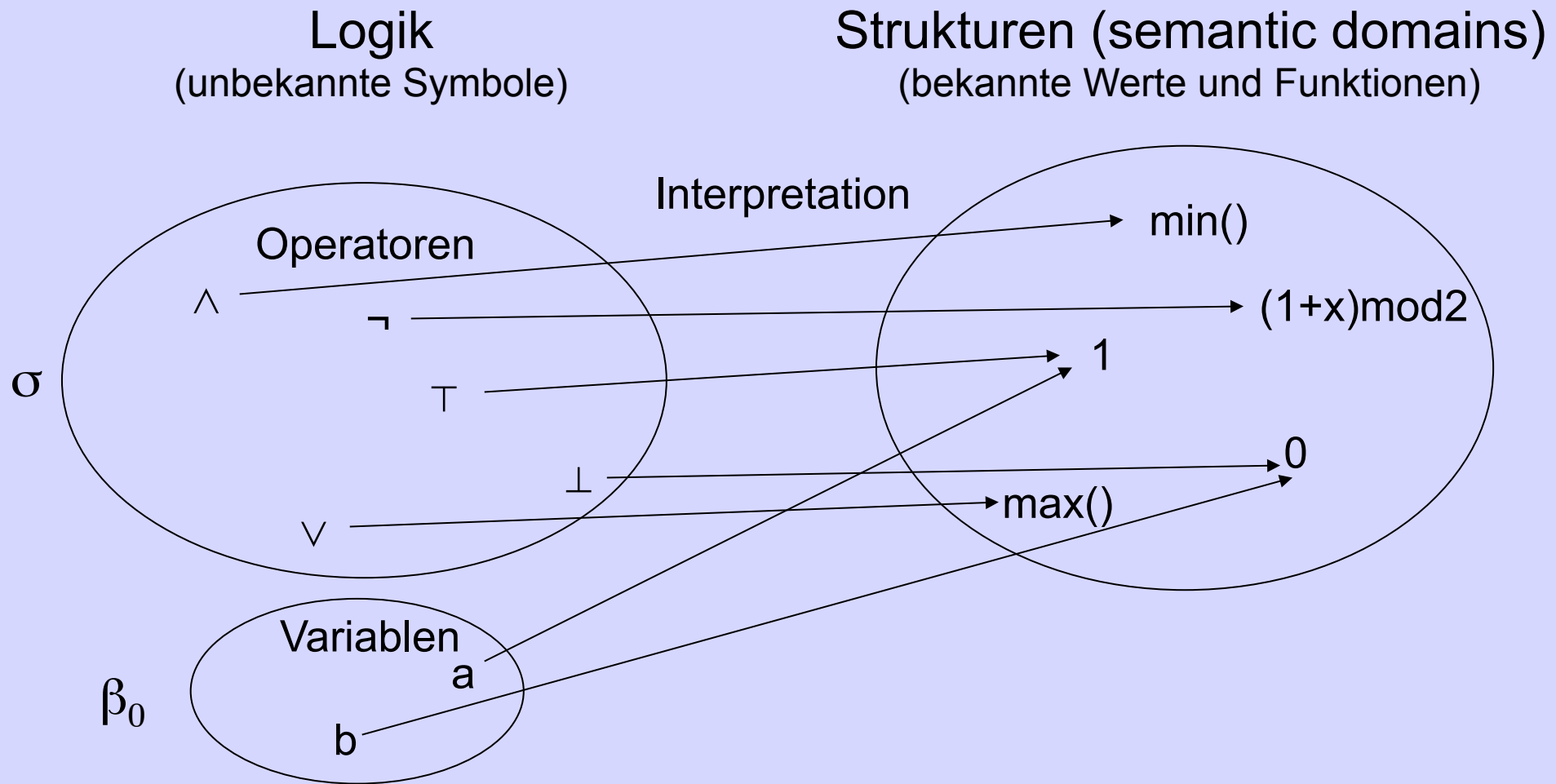


Aussagenlogik: Semantik

- Annahme (möglicherweise wechselnd) über Wahrheitswerte der atomaren Aussagen (Aussagenvariablen)
 - **Variablenbelegung:** $\beta_0 : \text{Var}(F) \rightarrow \mathcal{B}$
 - Z.B: $\{x \mapsto 0, y \mapsto 1, z \mapsto 1\}$
- Abbildung der Operatoren auf bekannte Funktionen
 - $\top \mapsto 1; \perp \mapsto 0; \neg(.) \mapsto 1-(.);$
 - $\vee(.,.) \mapsto \max(.,.); \wedge \mapsto \min(.,.);$
 - Abbildungsfunktion σ : $\sigma(\vee) = \max; \sigma(\wedge) = \min; \dots$
- Berechnung des Wahrheitswertes einer Formel über **Interpretation** β
 - $\beta(\top) = 1; \beta(\perp) = 0;$
 - $\beta(\text{op}(F, G)) = \sigma[\text{op}](\beta(F), \beta(G))$



Semantik der Aussagenlogik



Anwendungsbeispiel: *Mercedes-Dokumentation*

Example: E-Class

- approx. 1.500 codes (options, countries, ..)
- approx. 3.000 rules in product overview (PÜ)
- rule based BOM with approx. 35.000 pos.-variants..



$$B(P09) = 297 + 540 + 543;$$

$$B(610) = (512 / 527 / 528) + 608 + -978;$$

$$Z(P09) = (M271 + -M013 / M272) + 830 / \\ Z04 + -(M273 + Z27)$$

$$Z(682) = 623 / 830 / 513L$$

$$B(450) = L + 965 + 670 + 837 + -P34 + ((M271 / M651) + (953 / 955) + (100A / 200A) \\ + (334 / 335) + (301 / 336 / 337) / \\ / M642 + (2XXL / 557L / 571L) + (953 / 955) + (100A / 200A) + (334 / 335) + (301 / 337));$$



Anwendungsbeispiel: *Mercedes-Dokumentation*

Beispiel: E-Klasse



- ca. 1.500 Codes (Ausstattungen, Länder, ..)
- ca. 3.000 Regeln Produktübersicht (PÜ)
- regelbasierte Stückliste mit ca. 35.000 PosV.

$$B(P09) = 297 + 540 + 543;$$

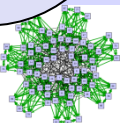
$$B(610) = (512 / 527 / 528) + 608 + -978;$$

$$Z(P09) = (M271 + -M013 / M272) + 830 / \\ Z04 + -(M273 + Z27)$$

$$Z(682) = 623 / 830 / 513L$$

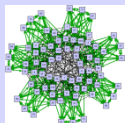
$$B(450) = L + 965 + 670 + 837 + -P34 + ((M271 / M65 \\ + (334 / 335) + (301 / 336 / 337) / \\ / M642 + (2XXL / 557L / 571L) + (953 / 955) + (100A / 2$$

P09: Sonnenschutzpaket
297: Rollos Fond-Türen links+rechts
540: Rollo elektr. Heckfenster
543: Sonnenbl. li+re m. Make-up-S
610: Nachtsichtsystem
512: Comand APS mit DVD-Wechsel.
527: Comand DVD APS mit NAVI
528: Comand DVD+ ohne NAVI
608: Autom. Fernlichtschaltung
978: Spezial-Fahrgestell
682: Feuerlöscher
830: Zusatzteile China
623: Golf-Staaten-Ausführung
513L: Belgien
M271: R4-Ottomotor
M272: V6-Ottomotor
M273: V8-Ottomotor
M013: Motor leistungsreduziert
Z04: B4 Panzerung
Z27: Bodenpanzerung
450: TAXI-International



Aussagenlogik: Beispiel 1 zur Semantik

- Variablenbelegung β_0 gegeben.
- Interpretation $\beta: \{\text{Formeln}\} \rightarrow \mathcal{B}$ rekursiv definiert:
 - $\beta(x) = \beta_0(x)$ für Aussagenvariable x
 - $\beta(f) = 0$
 - $\beta(t) = 1$
 - $\beta(\neg F) = 1 - \beta(F)$
 - $\beta(F \vee G) = \max(\beta(F), \beta(G))$
 - $\beta(F \wedge G) = \min(\beta(F), \beta(G))$
 - $\beta(F \rightarrow G) = \beta(\neg F \vee G)$
 - $\beta(F \leftrightarrow G) = \beta((F \rightarrow G) \wedge (G \rightarrow F))$
 - $\beta(F \oplus G) = \beta(\neg(F \leftrightarrow G))$

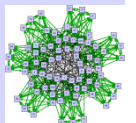


Aussagenlogik: Beispiel 2 zur Semantik

- Oft wird eine Abbildung in die natürliche Sprache benutzt:
 $\vee \mapsto \text{“oder”}$; $\wedge \mapsto \text{“und”}$;
 - Probleme: vieldeutig (“oder”), nicht maschinell berechenbar
- Alternative Semantik über Schaltfunktionen (geg. als Tabellen)

a, b	NUL	NOR		NOTa		NOTb	XOR	NAND	AND	EQV	b	IMP	a		OR	ONE
0 0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
0 1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1 0	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1 1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

$\vee \mapsto \text{OR}$; $\wedge \mapsto \text{AND}$; $\rightarrow \mapsto \text{IMP}$; $\leftrightarrow \mapsto \text{EQV}$; $\oplus \mapsto \text{XOR}$;



Aussagenlogik: Semantik (3)

- Eine Formel F heißt **erfüllbar**, wenn es eine Variablenbelegung $\beta_0 : \text{Var}(F) \rightarrow \mathcal{B}$ gibt, so dass $\beta(F) = 1$. β wird dann auch **Modell** von F genannt (schreibe $\beta \models F$ oder auch $\models_{\beta} F$).
 - Im Fall der Aussagenlogik ist die Interpretation der Junktoren fest. Dann heißt auch schon β_0 Modell von F .
- Eine Formel F heißt (**allgemein-**)**gültig** (schreibe $\models F$), falls für alle $\beta_0 : \text{Var}(F) \rightarrow \mathcal{B}$ gilt, dass $\beta(F) = 1$.
- Berechnung der Erfüllbarkeit z.B. unter Zuhilfenahme von Wahrheitstabellen



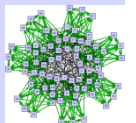
Wahrheitstabelle

$$F = ((x \wedge \neg(y \rightarrow z)) \oplus x)$$

x	y	z	$y \rightarrow z$	$\neg(y \rightarrow z)$	$x \wedge \neg(y \rightarrow z)$	$(x \wedge \neg(y \rightarrow z)) \oplus x$
0	0	0	1	0	0	0
0	0	1	1	0	0	0
0	1	0	0	1	0	0
0	1	1	1	0	0	0
1	0	0	1	0	0	1
1	0	1	1	0	0	1
1	1	0	0	1	1	0
1	1	1	1	0	0	1

Also ist F erfüllbar, aber nicht allgemeingültig.

Problem: n Variablen führen zu 2^n Zeilen in der Tabelle, daher nicht für große Formeln geeignet.



semantischer Folgerungsbegriff

- Eine Formel F **folgt (semantisch)** für eine Interpretation β **aus einer Formelmenge \mathcal{M} ,**
i.Z. $\mathcal{M} \models_{\beta} F$, falls $\beta(\mathcal{M})=1$ auch $\beta(F)=1$ impliziert.
- Gilt $\mathcal{M} \models_{\beta} F$ für alle Interpretationen β , so **folgt F (semantisch) aus \mathcal{M} ,** i.Z. $\mathcal{M} \models F$.
 - Jede Belegung, die \mathcal{M} wahr macht, macht auch F wahr.
- Schreibweisen:
 - $G \models F$ für $\{G\} \models F$
 - $\models F$ für $\{ \} \models F$
 - Jede Belegung macht F wahr. Die Menge der Vorbedingungen ist leer.



Äquivalenzumformungen

Für $\models (F \leftrightarrow G)$ schreiben wir auch $F \equiv G$.

- Achtung: \equiv ist ein Meta-Symbol, kein Operator der A-Logik
- \equiv bezeichnet eine Äquivalenzrelation (\rightarrow nachprüfen!)

➤ Distributivgesetze:

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

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

➤ Absorptionsgesetze:

$$F \vee (F \wedge G) \equiv F \wedge (F \vee G) \equiv F$$

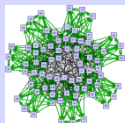
➤ DeMorgan'sche Gesetze:

$$\neg(F \wedge G) \equiv \neg F \vee \neg G$$

$$\neg(F \vee G) \equiv \neg F \wedge \neg G$$

➤ Kommutativität und Assoziativität von \wedge , \vee .

➤ .. und weitere



Boole'sche Algebra

$[\mathcal{B}; \wedge, \vee, ', 0, 1]$ ist eine **Boole'sche Algebra**, wenn

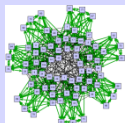
1. \wedge und \vee sind assoziativ und kommutativ
2. es gelten die Distributivgesetze
3. $x \wedge 0 = 0, x \wedge 1 = x \quad \forall x$
4. $x \vee 0 = x, x \vee 1 = 1 \quad \forall x$
5. Zu jedem x existiert genau ein x' mit $x \wedge x' = 0$ und $x \vee x' = 1$.

Aus diesen Axiomen folgen (durch *equational reasoning*) weitere Gleichungen, z.B. die Absorptionsgesetze und die De Morgan'schen Regeln.



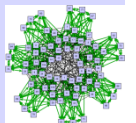
Zur Geschichte (Quelle: Wikipedia, Nov. 2008)

Die boolesche Algebra ist nach George Boole benannt, da sie auf dessen Logikkalkül von 1847 zurückgeht, in dem er erstmals algebraische Methoden in der Klassenlogik und Aussagenlogik anwandte. Ihre heutige Form verdankt sie der Weiterentwicklung durch Mathematiker wie John Venn, W. Stanley Jevons, Charles Pierce, Ernst Schröder und Giuseppe Peano. In Booles originaler Algebra entspricht die Multiplikation dem UND, die Addition dagegen weder dem exklusiven ENTWEDER-ODER noch dem inklusiven ODER ("mindestens eines von beiden ist wahr"). Die genannten Boole-Nachfolger gingen dagegen vom inklusiven ODER aus: Schröder entwickelte 1877 das erste formale Axiomensystem einer booleschen Algebra in additiver Schreibweise, Peano brachte es 1888 in die heutige Form und führte dabei die Symbole \cup und \cap ein. Das aussagenlogische ODER-Zeichen \vee stammt aus den Principia Mathematica von Russell / Whitehead 1910; Arend Heyting führte 1930 die Symbole \wedge und \neg ein. Den Namen „boolesche Algebra“ bzw. „boolean algebra“ prägte Henry Maurice Scheffer erst 1913. Das exklusive ENTWEDER-ODER, das Booles originaler Algebra näher kommt, legte erst Ivan Ivanovich Žegalkin 1927 dem booleschen Ring zugrunde, dem Marshall Harvey Stone 1936 den Namen gab. 1940 benutzte Claude Shannon boolesche Algebren erstmals zur Beschreibung elektrischer Schaltungen.



Beweise in der Aussagenlogik

- Mit der Äquivalenzrelation \equiv bilden die Formeln der Aussagenlogik eine Boolesche Algebra (\rightarrow nachprüfen!)
 - Also folgen die üblichen Gesetze (de Morgan etc) durch equational reasoning.
- Also stehen jetzt 2 Beweisverfahren für $F \equiv G$ zur Verfügung:
 - per Definition von \equiv beweise $\models (F \leftrightarrow G)$ über Belegungen
 - Führe Gleichheitsbeweise (equational reasoning) in der Booleschen Algebra (Umformungen von F zu F' und/oder G zu G' die beide identisch gleich sind, also $F' = G'$).
- Problem: keines der Verfahren ist effizient!



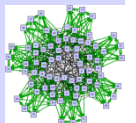
Deduktionstheorem

Sei \mathcal{M} eine Menge von Formeln und seien F und G Formeln. Dann gilt:

$$\mathcal{M} \cup \{ F \} \models G \text{ impliziert } \mathcal{M} \models (F \rightarrow G)$$

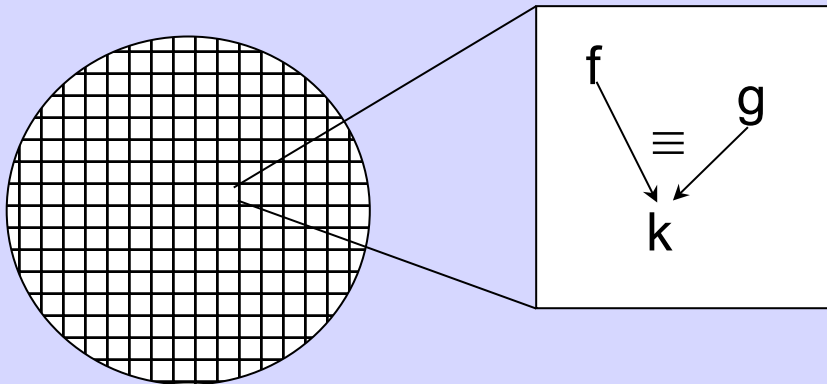
Mithilfe des Deduktionstheorems lässt sich das Folgerungsproblem letztlich auf das Tautologieproblem zurückführen.

$$\text{Bsp: } F \models G \text{ impliziert } \models (F \rightarrow G)$$



Normalformen

- Allgemeine Bedeutung von Normalformen:
 - Einheitliche Darstellung (bestimmtes Aussehen)
 - Einfachere Datenstrukturen + Algorithmen
 - Einfachere Beweise (da einfacherer Formelaufbau)
- Unterscheidung zur kanonischen Form:
 - kan. Form ist eindeutiger Repräsentant pro Äquivalenzklasse



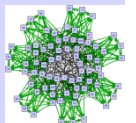
Normalformen: vereinfachte Formel

- F heißt **vereinfacht**, wenn $F = \neg f$ oder $F = f$ oder wenn f in F nicht vorkommt (bzw. wenn $F = \top$ oder $F = \perp$ oder wenn \top, \perp nicht in F vorkommen).
- **Beispiel:** $\neg (x \wedge f) \rightarrow \neg y$ ist nicht vereinfacht
- **Algorithmus:** Wende folgende (Reduktions-)Regeln so lange wie möglich an:
 - $F \vee f \Rightarrow F$
 - $f \vee F \Rightarrow F$
 - $F \wedge f \Rightarrow f$
 - $f \wedge F \Rightarrow f$
 - $F \vee t \Rightarrow t$
 - $t \vee F \Rightarrow t$
 - $F \wedge t \Rightarrow F$
 - $t \wedge F \Rightarrow F$



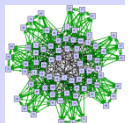
Normalformen: Negationsnormalform (NNF)

- F ist in NNF: Negationen kommen nur direkt vor Variablen oder vor f vor
- **Beispiel:** $\neg(x \vee y)$ ist nicht in NNF, $\neg x \wedge \neg y$ ist in NNF
- **Algorithmus:**
 - Ausgehend von vereinfachter Form
 - Wende folgende Regeln so lange wie möglich an:
 - $\neg(F \wedge G) \Rightarrow \neg F \vee \neg G$
 - $\neg(F \vee G) \Rightarrow \neg F \wedge \neg G$
 - $\neg\neg F \Rightarrow F$



Normalformen: Konjunktive Normalform (CNF)

- F ist in **CNF**: F ist Konjunktion von Klauseln
- **Klausel** (*clause*): Disjunktion von Literalen
 - **Horn-Klausel**: hat höchstens ein positives Literal.
- Beispiel: $(x \vee y \vee \neg z)$ ist eine Klausel, $(\neg z \vee y)$ ist Horn-K.
- Beispiel: $(x \vee y \vee \neg z) \wedge (\neg z \vee y) \wedge x$ ist in CNF
- Auch CNF ist immer noch keine kanonische Form
 - $(x \vee \neg y) \wedge (x \vee y) \equiv x$. Beide Seiten sind in verschiedener CNF.
- Algorithmus:
 - Ausgehend von NNF
 - Wende folgende Regeln so lange wie möglich an
 - $F \vee (G \wedge H) \Rightarrow (F \vee G) \wedge (F \vee H)$
 - $(G \wedge H) \vee F \Rightarrow (G \vee F) \wedge (H \vee F)$



Normalformen: CNF - Darstellung

➤ Operatoren in CNF durch Form bestimmt:

\vee immer in Klauseln, \wedge immer zwischen Klauseln

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

➤ Vereinfachung: Mengenschreibweise

$$F = \{\{x, \neg y\}, \{x, \neg z\}, \{z, \neg y\}, \{z, \neg x\}\}$$

- Operatoren weggelassen
- Formel = Menge von Klauseln
- Klausel = Menge von Literalen
- CNF zählt Menge von Constraints auf, die **simultan** erfüllt sein müssen, damit die beschriebene Funktion =1 wird.
- d.h. CNF zählt Bedingungen auf, die simultan erfüllt sein müssen, damit Funktion **nicht** =0 wird.



Normalformen: DNF - Darstellung

➤ Operatoren in DNF durch Form bestimmt:

\wedge immer in Mintermen, \vee immer zwischen Mintermen

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

➤ Anmerkungen:

- DNF zählt die 1-Stellen der Funktion unmittelbar auf
- übliche Denkweise des Menschen
- möglicherweise schlecht, wenn es viele 1-Stellen gibt



CNF direkt aus Funktionstabelle

➤ Direkte Konstruktion der CNF (für Menschen)

- In welchen Fällen wird die Funktion =0?
- Keiner dieser Fälle darf eintreten, also Fälle simultan ausschließen!

➤ Beispiel

- $F = x \vee (y \wedge z)$
- Nullstellen: $(\neg x \wedge \neg y \wedge \neg z) \vee (\neg x \wedge \neg y \wedge z) \vee (\neg x \wedge y \wedge \neg z)$
- Nullstellen simultan ausschließen:
$$\neg(\neg x \wedge \neg y \wedge \neg z) \wedge \neg(\neg x \wedge \neg y \wedge z) \wedge \neg(\neg x \wedge y \wedge \neg z) =$$
$$= (x \vee y \vee z) \wedge (x \vee y \vee \neg z) \wedge (x \vee \neg y \vee z) = [\text{Resolution, subsumption}]$$
$$(x \vee y) \wedge (x \vee z)$$



Beispiele zu DNF und CNF aus Funktionstabellen

$$F = ((x \wedge \neg(y \rightarrow z)) \oplus x)$$

x	y	z	$y \rightarrow z$	$\neg(y \rightarrow z)$	$x \wedge \neg(y \rightarrow z)$	$(x \wedge \neg(y \rightarrow z)) \oplus x$
0	0	0	1	0	0	0
0	0	1	1	0	0	0
0	1	0	0	1	0	0
0	1	1	1	0	0	0
1	0	0	1	0	0	1
1	0	1	1	0	0	1
1	1	0	0	1	1	0
1	1	1	1	0	0	1

CNF: keine der Nullstellen darf vorkommen

DNF: jede der Eins-Stellen darf vorkommen



Dualität von Formeln

F heißt **dual** zu G, falls F aus G durch Vertauschen der Junktoren \wedge und \vee und der Symbole f und t entsteht.

Schreibweise: F^δ für die zu F duale Formel

Satz: Falls $F \equiv G$, so auch $F^\delta \equiv G^\delta$



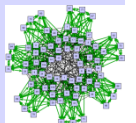
CNF Konversion aus DNF

➤ Konversion DNF \leftrightarrow CNF über Dualität

- $\text{CNF}(F) = \text{dual}(\text{DNF}(\text{dual}(F)))$
- $\text{DNF}(F) = \text{dual}(\text{CNF}(\text{dual}(F)))$

➤ Begründung für DNF \rightarrow CNF

- $F \equiv \neg\neg F \equiv \neg \text{DNF}(\neg F) \equiv \text{CNF}(\neg\neg F) \equiv \text{CNF}(F)$
- $\text{dual}(F)$ hat gleiche Form wie $\neg F$, aber mit negierten Variablen
- $\text{DNF}(\text{dual}(F))$ hat also gleiche Form wie $\text{DNF}(\neg F)$ aber neg. V.
- $\text{dual}(\text{DNF}(\text{dual}(F)))$ hat Form wie $\neg \text{DNF}(\neg F)$, aber doppelt neg. V.
- $\text{dual}(\text{DNF}(\text{dual}(F))) = \neg \text{DNF}(\neg F) = \text{CNF}(F)$
- \rightarrow dies entspricht der direkten Konstruktion:
Menge der simultanen Bedingungen, dass **nicht** $F=0$.



CNF Konversion direkt aus Formel

➤ Bilde CNF von unten nach oben aus dem Parse-Tree

- Definitionen:
 - Klauseln $X = \{x_1, \dots, x_m\}$, $Y = \{y_1, \dots, y_n\}$
 - $\{X, Y\}$ repräsentiert $X \wedge Y$ mit Klauseln X und Y
 - $X \cup Y := \{x_1, \dots, x_m, y_1, \dots, y_n\}$
- $\text{CNF}(x) := \{\{x\}\}$
- $\text{CNF}(\wedge, \{X_1, \dots, X_m\}, \{Y_1, \dots, Y_n\}) = \{X_1, \dots, X_m, Y_1, \dots, Y_n\}$
- $\text{CNF}(\vee, \{X_1, \dots, X_m\}, \{Y_1, \dots, Y_n\}) =$
 $\{X_1 \cup Y_1, \dots, X_m \cup Y_1, X_1 \cup Y_2, \dots, X_m \cup Y_2, \dots, X_1 \cup Y_n, \dots, X_m \cup Y_n\}$

➤ Vorteil:

- Beginn mit kleinen Formeln, sukzessive Vereinfachung
- $\{\{x\}, \{x, y\}\} = \{\{x\}\}$, sowie $\{\{x\}, \neg x\} = T$

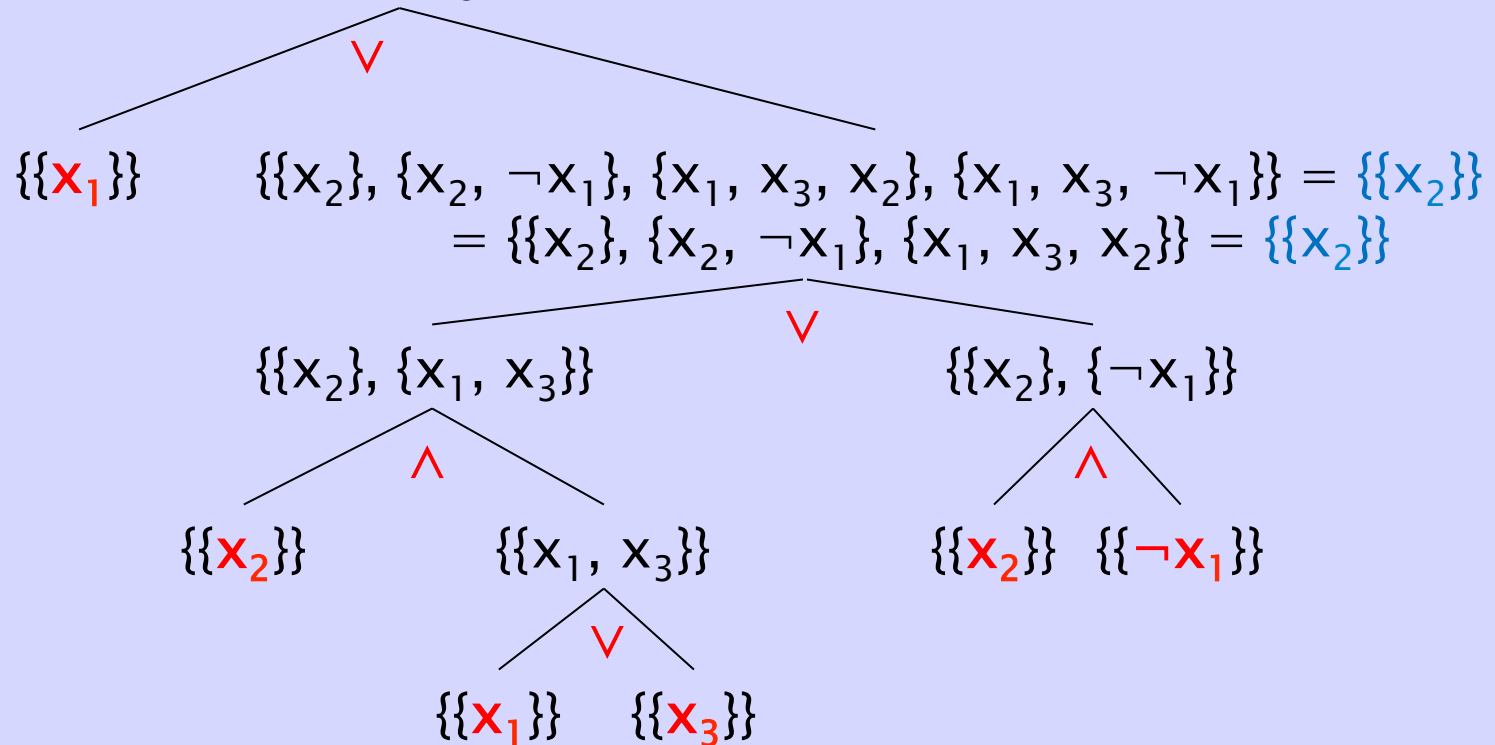


Beispiel zur bottom-up CNF Konversion

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

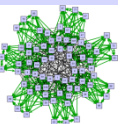
$$\{\{x_1, x_2\}, \{x_1, x_2, \neg x_1\}, \{x_1, x_3, x_2\}\} =$$

$$\{\{x_1, x_2\}, \{x_1, x_3, x_2\}\} = \{\{x_1, x_2\}\}$$



Normalformen: CNF-Transformation - Komplexität

- Problem: Anwendung des Distributivgesetzes kann die Größe der Formel (annähernd) verdoppeln
- Im schlimmsten Fall: $\text{CNF}(F)$ exponentiell größer als F
- Beispiel: $(x_{11} \wedge x_{12}) \vee \dots \vee (x_{n1} \wedge x_{n2})$.
 - CNF enthält 2^n Klauseln mit je n Variablen (\rightarrow Induktion)
- Lässt sich dieses Wachstum vermeiden?



CNF: Tseitin-Transformation

- Einführung von (neuen) Hilfsvariablen, um Teilformeln abzukürzen, d.h. äquivalent zu ersetzen.
- Neue zusätzliche Constraints erzwingen Äquivalenz der Hilfsvariable zur Teilformel
- Das geht ganz allgemein, wir demonstrieren und benutzen das für CNF
 - Abkürzung von $(G \wedge H)$ durch x innerhalb von $F \vee (G \wedge H)$.
 - Fordere $x \equiv (G \wedge H)$; dadurch wird $F \vee (G \wedge H)$ äquivalent zur Klausel $F \vee x$.
 - Aus $x \leftrightarrow (G \wedge H)$ erhalte $x \rightarrow (G \wedge H)$ und $(G \wedge H) \rightarrow x$, also die Klauseln $(\neg x \vee G)$, $(\neg x \vee H)$, $(\neg G \vee \neg H \vee x)$ zusätzlich zu $(F \vee x)$.
 - Dies ist die „volle“ Tseitin-Transformation



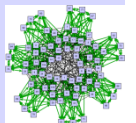
CNF: Tseitin-Transformation

- Problem: Verdoppelung von G und H durch die volle Tseitin-Transformierte $(F \vee x) \wedge (x \rightarrow (G \wedge H)) \wedge ((G \wedge H) \rightarrow x)$
- Lösung: Erhalte „normale“ Tseitin-CNF-Transformation durch Verzicht auf Constraint $((G \wedge H) \rightarrow x)$.
 - falls $\beta(G \wedge H) = 1$ und $\beta(F) = 1$, dann sind jetzt zwei verschiedene Erweiterungen β^x möglich mit (alternativ) $\beta^x(x) = 1$ und $\beta^x(x) = 0$.
 - die Belegung der Tseitin-Variablen interessiert nicht, die bisherige „Kernbelegung“ β gewinnt man durch Ignorieren von x.
 - Beim Zählen der Anzahl erfüllender Belegungen $\#F$ einer Formel F verfälscht die Tseitin-Transformation das Ergebnis.
- Sei F eine Formel, F^* sei die Tseitin-Transformierte
 - F und F^* sind i.A. nicht äquivalent, aber erfüllbarkeits-äquivalent



CNF: Tseitin-Transformation

- $F \models G$: Formel F ist erfüllbarkeits-äquivalent zu G
- $F \models G$ iff $[F \text{ ist erfüllbar gdw. } G \text{ ist erfüllbar}]$.
 - Beispiel: $x \models y$, aber $x \neq y$
 - Zu beweisen: Für jede erfüllende Belegung β_F von F existiert eine (i.A. andere) erfüllende Belegung β_G von G (und für jedes erfüllende β_G von G existiert ein erfüllendes β_F von F).
 - Zum Vergleich: $F \equiv G$ gdw. Jede erfüllende Belegung β von F ist auch eine erfüllende Belegung von G (und umgekehrt).
 - Beispiel CNF-Konversion: $F \vee (G \wedge H) \neq (F \vee x) \wedge (x \rightarrow (G \wedge H))$
 - Falls $\exists \beta$ mit $\models_{\beta} F$ und $\not\models_{\beta} (G \wedge H)$ und $\beta(x)=1$, dann gilt $\models_{\beta} F \vee (G \wedge H)$, aber $\not\models_{\beta} (F \vee x) \wedge (x \rightarrow (G \wedge H))$
 - Für β^x mit $\models_{\beta^x} (F \vee x) \wedge (x \rightarrow (G \wedge H))$ gilt auch $\models_{\beta^x} F \vee (G \wedge H)$
 - Aus β^x mit $\models_{\beta^x} (F \vee x) \wedge (x \rightarrow (G \wedge H))$ gewinnen wir durch Nichtbeachtung der Tseitin-Variable x (die sonst nicht vorkommt) ein β mit $\models_{\beta} F \vee (G \wedge H)$.



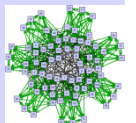
CNF: Tseitin-Transformation

➤ Begründung des Verfahrens von Tseitin:

- $F \vee (G \wedge H) \cong (F \vee x) \wedge (G \vee \neg x) \wedge (H \vee \neg x)$
- $(G \wedge H) \vee F \cong (F \vee x) \wedge (G \vee \neg x) \wedge (H \vee \neg x)$

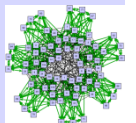
➤ Beweis der Erfüllbarkeits-Äquivalenz

- Bemerkung: $(F \vee x) \wedge (G \vee \neg x) \wedge (H \vee \neg x) \equiv (F \vee x) \wedge (x \rightarrow G) \wedge (x \rightarrow H)$
- “ \rightarrow ”: Falls $\beta(F \vee (G \wedge H))=1$, dann $\beta(F)=1$ oder $\beta(G \wedge H)=1$
 - Falls $\beta(F)=1$, dann existiert auch Erweiterung von β zu β^x mit $\beta^x(x)=0$, also $\beta^x(F)=1$ und $\beta^x(x \rightarrow G)=1$ und $\beta^x(x \rightarrow H)=1$, egal wie G, H von β bewertet sind (Bem.: Falls zufällig $\beta(G \wedge H)=1$ funktioniert auch $\beta^x(x)=1$.)
 - Falls $\beta(F)=0$, dann folgt $\beta(G \wedge H)=1$. Also existiert Erweiterung β^x von β mit $\beta^x(x)=1$, also $\beta^x(F \vee x)=1$ und $\beta^x(x \rightarrow G)=1$ und $\beta^x(x \rightarrow H)=1$.
- „ \leftarrow ”: $\beta^x((F \vee x) \wedge (x \rightarrow G) \wedge (x \rightarrow H))=1$.
 - Falls $\beta^x(x)=1$, dann auch $\beta^x(G)=1$ und $\beta^x(H)=1$, also $\beta^x(F \vee (G \wedge H))=1$.
 - Falls $\beta^x(x)=0$, dann auch $\beta^x(F)=1$, also $\beta^x(F \vee (G \wedge H))=1$.



CNF: Tseitin-Transformation

- Nach einer Tseitin-Transformation kann sich die Anzahl erfüllender Belegungen erhöhen (und zwar um den Faktor 2 für jede neue Tseitin-Hilfsvariable).
- Erinnerung Tseitin: $F \vee (G \wedge H) \cong (F \vee x) \wedge (x \rightarrow G) \wedge (x \rightarrow H)$
 - Falls $\beta(F)=1$ und $\beta(G \wedge H)=1$, dann $\beta^x((F \vee x) \wedge (x \rightarrow G) \wedge (x \rightarrow H))=1$ sowohl für β^x mit $\beta^x(x)=0$ als auch für β^x mit $\beta^x(x)=1$
 - in diesem Fall führt β zu 2 erfüllenden Belegungen, sonst nur zu einer.
 - Umgekehrt: Falls es im transformierten System zwei erfüllende Belegungen $\beta^{x=1}$ und $\beta^{x=0}$ gibt, die sich nur auf einer Tseitin-Variable x unterscheiden, dann gibt es im Ursprungssystem dazu nur eine erfüllende Belegung
 - denn auf jeder Ursprungsvariablen v gilt $\beta^{x=1}(v) = \beta^{x=0}(v) = \beta(v)$.
 - Für die gemeinsame Restriktion β gilt in diesem Fall $\beta(F)=1$ und $\beta(G \wedge H)=1$, denn aus $\beta^{x=0}(F \vee x)$ folgt $\beta(F)=1$ und aus $\beta^{x=1}((x \rightarrow G) \wedge (x \rightarrow H))=1$ folgt $\beta(G \wedge H)=1$.



CNF: Tseitin-Transformation

➤ Zusammenfassung

➤ CNF-Transformations-Verfahren von Tseitin:

- $F \vee (G \wedge H) \cong (F \vee x) \wedge (G \vee \neg x) \wedge (H \vee \neg x)$
- $(G \wedge H) \vee F \cong (F \vee x) \wedge (G \vee \neg x) \wedge (H \vee \neg x)$

➤ Ergebnis:

- keine Verdoppelung von F, kein exponentielles Wachstum bei CNF-Transformation
- Erfüllbarkeits-Äquivalenz
- Anzahl erfüllender Belegungen kann sich mit jeder Tseitin-Variable gegenüber dem Ausgangssystem verdoppeln.

