Vorlesung
– Automatisches Beweisen –

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

Vorlesung Sommersemester 2010:

Aussagenlogik

Prof. Dr. W. Küchlin

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 ausgerechnet 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,*...): Platzhalter für beliebige (atomare) Aussagen, wie z.B. „5 ist eine Primzahl“, „2 > 3“
  - Konstanten: ⊥ [false] und ⊤ [true] (oder auch f, t)
  - Junktoren: mindestens ¬ [„nicht“], ∧ [„und“], ∨ [„oder“]

    Nach Bedarf weitere, z.B.: →, ↔, ⊕

    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
  - f und t 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:
  - $(((\neg x) \vee (y \wedge z)) \vee f)$
  - $((x \wedge \neg(y \rightarrow z)) \oplus x)$
- 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)
- $\mathcal{V} = \{x_0, ..., x_n, ..., x, y, z, ...\}$: 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:

$$\mathrm{Var}(((x \wedge \neg(y \rightarrow z)) \oplus x)) = \{x, y, z\}$$

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 (Interpretation ν, 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

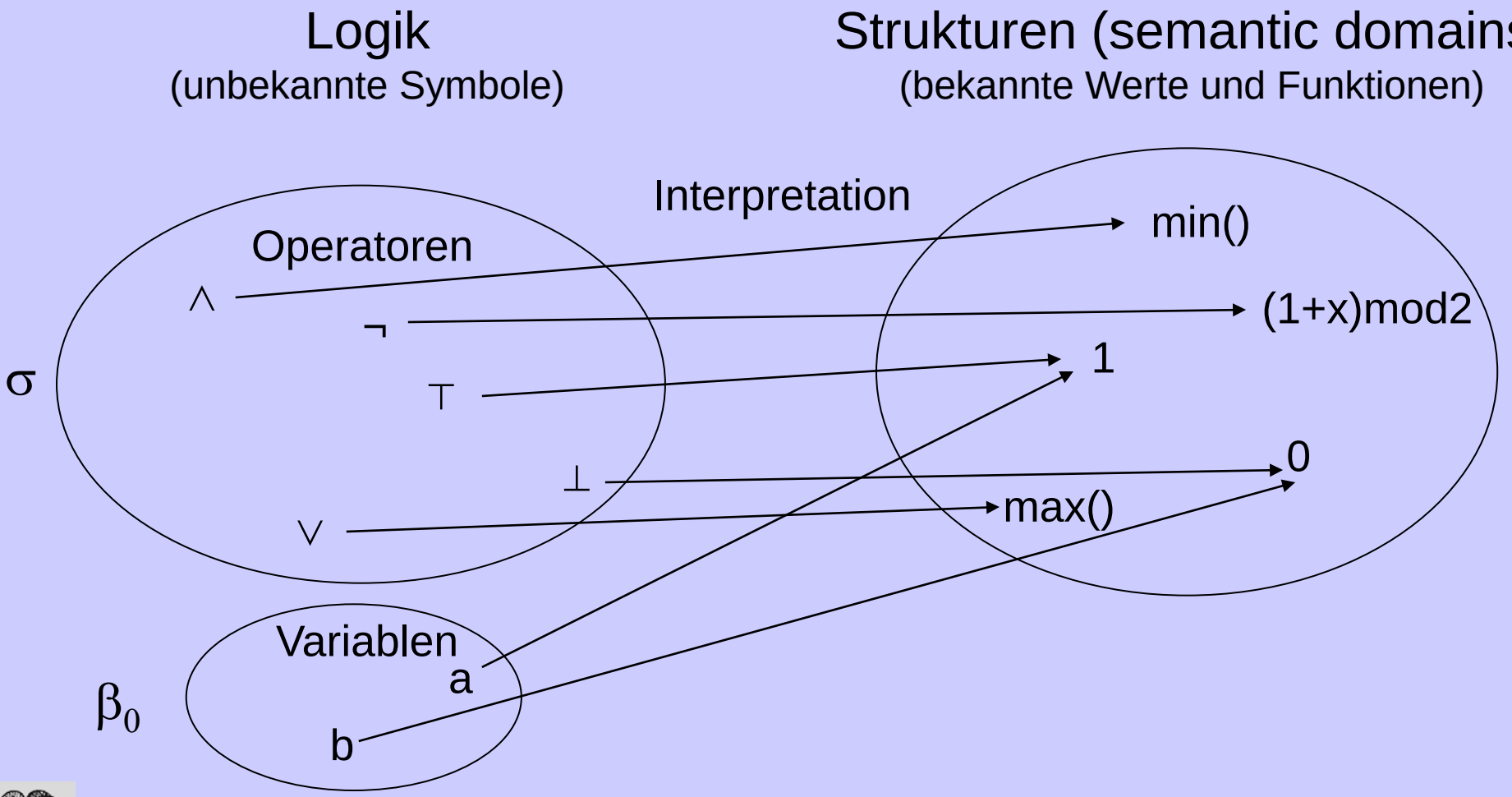
- Annahme (möglicherweise wechselnd) über Wahrheitswerte der atomaren Aussagen (Aussagenvariablen)
  - Variablenbelegung: $\beta_0 : \mathrm{Var(F)} \rightarrow \mathcal{B}$
  - Z.B: $\{x \mapsto 0, y \mapsto 1, z \mapsto 1\}$
- Abbildung der Operatoren auf bekannte Funktionen

  t ↦ 1; f ↦ 0; ¬(.) ↦ 1-(.);

  ∨(.,.) ↦ max(., .); ∧ ↦ min(., .);

  Abbildungsfunktion σ: σ(∨)= max; σ(∧) = min; …
- Berechnung des Wahrheitswertes einer Formel über Interpretation β
  - β(t) = 1; β(f) = 0;
  - β(op(F,G)) = σ[op](β(F), β(G))

Semantik der Aussagenlogik



Aussagenlogik: Beispiel 1 zur Semantik

- Variablenbelegung β_0 gegeben.
- Interpretation β: {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: $\vee \mapsto$ “oder”; $\wedge \mapsto$ “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

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

Aussagenlogik: Semantik (3)

- Eine Formel F heißt erfüllbar, wenn es eine Variablenbelegung $\beta_0 : \mathrm{Var}(\mathrm{F}) \to \mathcal{B}$ gibt, so dass $\beta(\mathrm{F})=1$. β wird dann auch Modell von F genannt (i.Z. $\beta \models \mathrm{F}$ oder auch $\models_\beta \mathrm{F}$).
  - 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 (i.Z. $\models \mathrm{F}$), falls für alle $\beta_0 : \mathrm{Var}(\mathrm{F}) \to \mathcal{B}$ gilt, dass $\beta(\mathrm{F})=1$.
- 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 $\{\ \} \models F$

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 (→ 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:

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

- Kommutativität und Assoziativität von $\wedge$, $\vee$.

- .. und weitere

Boolesche 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. $x\wedge 0 = 0$, $x\wedge 1 = x$ $\forall x$
4. $x\vee 0 = x$, $x\vee 1 = 1$ $\forall x$
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 ∪ und ∩ ein. Das aussagenlogische ODER-Zeichen ∨ stammt aus den Principia Mathematica von Russell / Whitehead 1910; Arend Heyting führte 1930 die Symbole ∧ und ¬ 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 ≡ bilden die Formeln der Aussagenlogik eine Boolesche Algebra (→ 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 ≡ 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

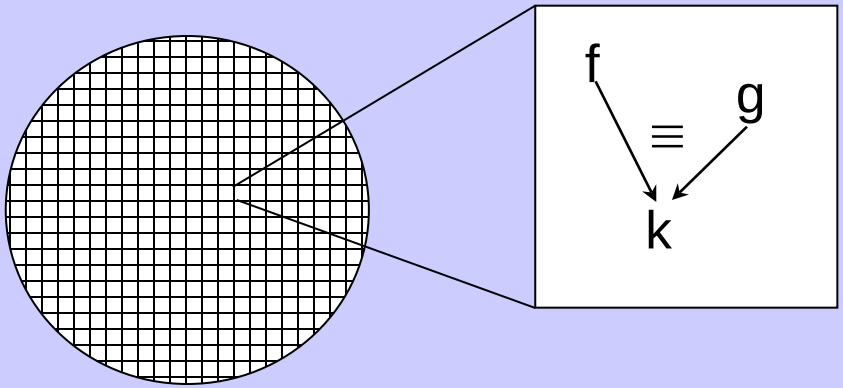
Sei $\mathcal{M}$ eine Menge von Formeln und seien F und G Formeln. Dann gilt:

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

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

Normalformen

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



Normalformen: vereinfachte Formel

- F heißt vereinfacht, wenn F = ¬f oder F = f oder wenn f in F nicht vorkommt (bzw. wenn F=⊤ oder F=⊥ oder ⊤, ⊥ nicht in F vorkommen).
- **Beispiel:** ¬ (x∧f)→¬y ist nicht vereinfacht
- **Algorithmus:** Wende folgende (Reduktions-)Regeln so lange wie möglich an:
  - F ∨ f ⇛ F F ∨ t ⇛ t
  - f ∨ F ⇛ F t ∨ F ⇛ t
  - F ∧ f ⇛ f F ∧ t ⇛ F
  - f ∧ F ⇛ f t ∧ F ⇛ 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:
    - $\neg(F \wedge G) \Rrightarrow \neg F \vee \neg G$
    - $\neg(F \vee G) \Rrightarrow \neg F \wedge \neg G$
    - $\neg\neg F \Rrightarrow F$

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
    - $F \vee (G \wedge H) \Rrightarrow (F \vee G) \wedge (F \vee H)$
    - $(G \wedge H) \vee F \Rrightarrow (G \vee F) \wedge (H \vee F)$

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:

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

$$= (x \vee y \vee z) \wedge (x \vee y \vee \neg z) \wedge (x \vee \neg y \vee z) = \text{[Resolution,subsumption]}$$

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

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 ←→ CNF über Dualität**
  - CNF(F) = dual(DNF(dual(F))
  - DNF(F) = dual(CNF(dual(F))
- **Begründung für DNF → CNF**
  - F ≡ ¬¬F ≡ ¬DNF(¬F) ≡ CNF(¬¬F) ≡ CNF(F)
  - dual(F) hat gleiche Form wie ¬F, aber mit negierten Variablen
  - DNF(dual(F)) hat also gleiche Form wie DNF(¬F) aber neg. V.
  - dual(DNF(dual(F)) hat Form wie ¬DNF(¬F), aber doppelt neg. V.
  - dual(DNF(dual(F)) = ¬DNF(¬F) = CNF(F)
  - → dies entspricht der direkten Konstruktion:
    Menge der simultanen Bedingungen, dass **nicht** F=0.

CNF Konversion direkt aus Formel

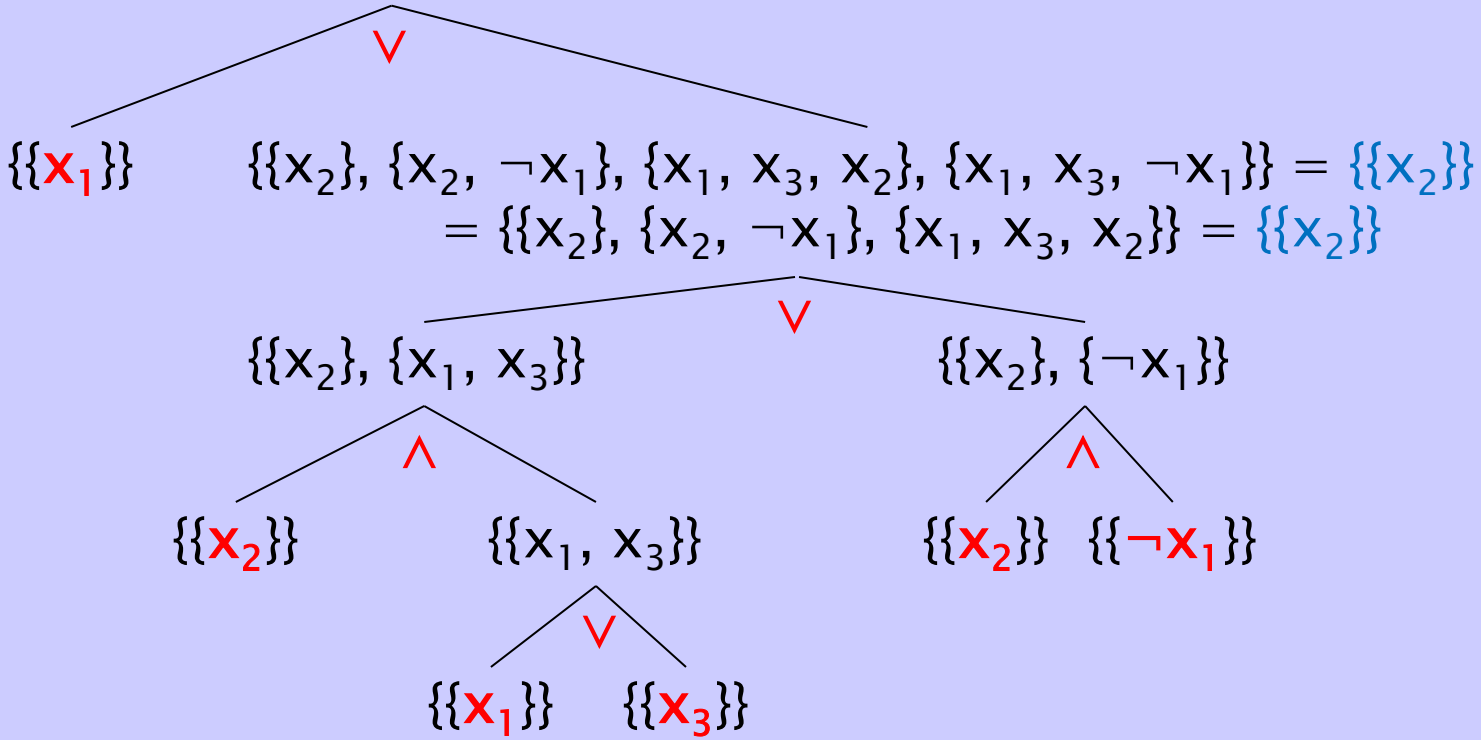
- **Bilde CNF von unten nach oben aus dem Parse-Tree**
  - Definitionen:
    - Klauseln $X = \{x_1,\ldots,x_m\}$, $Y = \{y_1,\ldots,y_n\}$
    - $\{X, Y\}$ repräsentiert $X \wedge Y$ mit Klauseln X und Y
    - $X \cup Y := \{x_1,\ldots,x_m,y_1,\ldots,y_n\}$
  - $CNF(x) := \{\{x\}\}$
  - $CNF(\wedge,\{X_1,\ldots,X_m\},\{Y_1,\ldots,Y_n\}) = \{X_1,\ldots,X_m,Y_1,\ldots,Y_n\}$
  - $CNF(\vee,\{X_1,\ldots,X_m\},\{Y_1,\ldots,Y_n\}) =$

    $\{X_1 \cup Y_1,\ldots,X_m \cup Y_1,\ X_1 \cup Y_2,\ldots,X_m \cup Y_2,\ldots,\ X_1 \cup Y_n,\ldots,X_m \cup Y_n\}$
- **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: CNF(F) exponentiell größer als F
- Beispiel: $(x_{11} \wedge x_{12}) \vee \ldots \vee (x_{n1} \wedge x_{n2})$.
  - CNF enthält 2^n Klauseln mit je n Variablen (→ Induktion)

- Lässt sich dieses Wachstum vermeiden?

CNF: Tseitin-Transformation

- Einführung von (neuen) Hilfsvariablen, um Teilformeln abzukürzen, d.h. zu ersetzen.
- Neue zusätzliche Constraints erzwingen Äquivalenz der Hilfsvariable zur Teilformel
- Das geht ganz allgemein, wir benutzen das für CNF
  - Abkürzung von $(G \wedge H)$ innerhalb $F \vee (G \wedge H)$.
  - Setze $x \equiv (G \wedge H)$; dadurch wird $F \vee (G \wedge H)$ zur Klausel $F \vee x$.
  - Aus $x \equiv (G \wedge H)$ erhalte $x \rightarrow (G \wedge H)$ und $(G \wedge H) \rightarrow x$, also die Klauseln $(\neg x \vee G)$, $(\neg x \vee H)$, $(\neg G \vee \neg H \vee x)$, sowie $(F \vee x)$
- Sei F eine Formel, F* sei die Tseitin-Transformierte
  - F und F* sind i.A. nicht äquivalent, aber erfüllbarkeits-äquivalent

CNF: Tseitin-Transformation

- $F \cong G$: Formel F ist erfüllbarkeits-äquivalent zu G
- $F \cong G$ iff [F ist erfüllbar gdw. G ist erfüllbar].
  - Beispiel: $x \cong y$, aber $x \not\equiv y$
  - Beispiel CNF-Konversion: $F \vee (G \wedge H) \not\equiv (F \vee x) \wedge (x \leftrightarrow (G \wedge H))$
    - Falls $\exists \beta$ mit $\models_\beta F$ und $\models_\beta (G \wedge H)$, dann erweitere β um $x = 0$ zu β^x
      Dann gilt $\models_{\beta^x} F \vee (G \wedge H)$, aber $\not\models_{\beta^x} (F \vee x) \wedge (x \Leftrightarrow (G \wedge H))$
  - Beispiel CNF-Konversion: $F \vee (G \wedge H) \cong (F \vee x) \wedge (x \leftrightarrow (G \wedge H))$
    - Falls $\exists \beta$ mit $\models_\beta F \vee (G \wedge H)$, dann erweitere β so, dass $\models_{\beta^x} (x \leftrightarrow (G \wedge H))$, und somit $\models_{\beta^x} (F \vee x) \wedge (x \leftrightarrow (G \wedge H))$
    - Bemerkung: diese Erweiterung ist eindeutig, da $\beta^x(x) = \beta(G \wedge H)$.
    - Falls $\exists \beta^x$ mit $\models_{\beta^x} (F \vee x) \wedge (x \leftrightarrow (G \wedge H))$, dann gibt es 2 Fälle
      - Falls $\beta^x(x) = 0$, dann ist $\beta^x(F) = 1$ und somit $\models_\beta F \vee (G \wedge H)$, da $x \notin$ vars(F)
      - Falls $\beta^x(x) = 1$, dann ist $\beta^x(G \wedge H) = 1$ und somit gleichermaßen $\models_\beta F \vee (G \wedge H)$.
    - Bemerkung: aus β^x mit $\models_{\beta^x} (F \vee x) \wedge (x \leftrightarrow (G \wedge H))$ gewinnen wir durch Nichtbeachtung der Tseitin-Variable x also ein β mit $\models_\beta F \vee (G \wedge H)$

CNF: Tseitin-Transformation

- „Volle“ Tseitin-CNF-Transformation
  - Ersetze $F \vee (G \wedge H)$ durch $(F \vee x) \wedge (x \rightarrow (G \wedge H)) \wedge ((G \wedge H) \rightarrow x)$, bzw. durch die Klauseln $(F \vee x)$, $(\neg x \vee G)$, $(\neg x \vee H)$, $(\neg G \vee \neg H \vee x)$,
- Problem: Verdoppelung von G und H
- Lösung: „Normale“ Tseitin-CNF-Transformation durch Verzicht auf Constraint $(\neg G \vee \neg H \vee x)$.
  - 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 aber nicht weiter, die „Kernbelegung“β gewinnt man wie bisher durch Ignorieren von x.
  - Beim Zählen der Anzahl erfüllender Belegungen #F einer Formel F verfälscht die Tseitin-Transformation das Ergebnis.

CNF: Tseitin-Transformation

- Zusammenfassung
- CNF-Transformations-Verfahren von Tseitin:
  - $F \vee (G \wedge H) \mathrel{\widehat{=}} (F \vee x) \wedge (G \vee \neg x) \wedge (H \vee \neg x)$
  - $(G \wedge H) \vee F \mathrel{\widehat{=}} (F \vee x) \wedge (G \vee \neg x) \wedge (H \vee \neg x)$
- Ergebnis:
  - keine Verdoppelung von F, kein exponentielles Wachstum bei CNF-Transformation
  - Erfüllbarkeits-Äquivalenz

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: $(F \vee x) \wedge (G \vee \neg x) \wedge (H \vee \neg x) \equiv (F \vee x) \wedge (x \rightarrow G) \wedge (x \rightarrow H)$
  - "→": 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
    - 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 $\beta^x(x \rightarrow H)=1$.
  - „←": $\beta^x ((F \vee x) \wedge (x \rightarrow G) \wedge (x \rightarrow H))=1$.
    - 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 $\beta(G \wedge H)=1$.