SAT-Solving und Anwendungen

Einführung / Aussagenlogik

Prof. Dr. Wolfgang Küchlin
Dr. Eray Gençay
Rouven Walter, M.Sc.

Universität Tübingen

19. Oktober 2017

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Logischer Formalismus

Syntax

Wie werden Formeln gebildet?

- klassisch-mathematisch: bestimmte ausgezeichnete Zeichenreihen
  - (typisch: induktive Definitionen)
- informatisch: Sprache einer Grammatik

Semantik

Was ist die Bedeutung einer Formel?

- Allgemein: Abbildung in einen (bekannten) Semantik-Bereich
- Hier: Semantik-Bereich $\mathbb{B} = \{1, 0\}$ der Boole'schen Wahrheitswerte

Kalkül

Wie kann die Gültigkeit einer Formel berechnet werden?

- Inferenzregeln, Algorithmen

Syntax

- Bestandteile von Formeln:
  - Aussagenvariablen aus einer unendlichen Menge $\mathcal{P}$ von Variablen: Platzhalter für beliebige (atomare) Aussagen, wie z.B. "5 ist eine Primzahl", "Eine Woche hat 7 Tage", "$4 = 2$" etc.
  - Konstanten $\top$, $\bot$: Zur Repräsentation der wahren bzw. falschen Aussage.
  - Operatoren ($\wedge$ und / $\vee$ oder / $\neg$ nicht / $\rightarrow$ Implikation / $\leftrightarrow$ Äquivalenz / $\oplus$ xor): zur Bildung komplexer Formeln (zus.-gesetzte Aussagen)
  - Hilfssymbole: Klammern
- Induktive Definition der Menge $\mathcal{F}$ aller gültigen Formeln:
  - Jede Aussagenvariable $x \in \mathcal{P}$ ist in $\mathcal{F}$
  - Die Konstanten $\top$ und $\bot$ sind in $\mathcal{F}$
  - Sind P und Q in $\mathcal{F}$, so sind auch $(\neg P), (P \wedge Q), (P \vee Q), (P \rightarrow Q), (P \leftrightarrow Q), (P \oplus Q)$ in $\mathcal{F}$
  - Nichts sonst ist eine gültige Formel

Syntax (ctd.)

- (EBNF)-Grammatik für Formeln: Übung.
- Zur Einsparung von Klammern:
  Priorität der Operatoren (abnehmend): $\neg, \wedge, \vee, \oplus, \rightarrow, \leftrightarrow$
  Bei gleichen Operatoren: Rechts-assoziative Klammerung
- Literal: positives oder negatives Vorkommen einer Aussagenvariable $\{x, \neg x\}$
- $var(P)$: Menge der Variablen, die in der Formel P vorkommen

Beispiel (Prioritäten)

$x \wedge y \vee \neg z \leftrightarrow x \oplus \neg z \wedge \neg w$ ist klammerfreie Schreibweise für:
$((x \wedge y) \vee (\neg z)) \leftrightarrow (x \oplus ((\neg z) \wedge (\neg w)))$

Syntax - Beispiele

Konvention:

- *Aussagenvariablen: Kleinbuchstaben* $x, y, z, ..., x_1, x_2, ...$
- *Meta-Symbole zur Bezeichnung beliebiger Formeln: Großbuchstaben* $P, Q, R, ..., P_1, P_2, ...$

Beispiel (Gültige Formeln)

- x
- $x \vee y$
- $(x \vee y) \wedge (y \oplus z) \rightarrow y$

Beispiel (Formel-Muster)

- $(x \vee y) \wedge (y \oplus z) \rightarrow P$
- $var((x \vee y) \wedge (y \oplus z) \rightarrow P) = \{x, y, z\} \cup var(P)$

Semantik

- Wie wird der Wahrheitsgehalt einer Formel bestimmt?
- 2-elementige Menge der Booleschen Wahrheitswerte:
  $\mathbb{B} = \{\mathbf{T}, \mathbf{F}\}$ oder $\{\mathit{true}, \mathit{false}\}$ etc. Wir verwenden: $\{1, 0\}$.
- Dazu: Belegung der Aussagenvariablen mit 1 (wahr) oder 0 (falsch)
  - Variablenbelegung einer Variable x ist Funktion $\nu_0 : \mathcal{P} \to \mathbb{B}$
  - Notation $x \mapsto 1$ oder $x \mapsto 0$
- Funktion ν_0 kann rekursiv erweitert werden auf Formeln P ($\nu : \mathcal{F} \to \mathbb{B}$):
  - $\nu(x) = \nu_0(x)$ für $x \in \mathcal{P}$
  - $\nu(\top) = 1$
  - $\nu(\bot) = 0$
  - $\nu(\neg P) = 1 - \nu(P)$
  - $\nu(P \vee Q) = \texttt{MAX}(\nu(P), \nu(Q))$
  - $\nu(P \wedge Q) = \texttt{MIN}(\nu(P), \nu(Q))$
  - $\nu(P \to Q) = \nu(\neg P \vee Q)$
  - $\nu(P \leftrightarrow Q) = \nu((P \to Q) \wedge (Q \to P))$
  - $\nu(P \oplus Q) = \nu(\neg(P \leftrightarrow Q))$
- Auf anderen $\mathbb{B}$ müssen `MAX` und `MIN` entsprechend definiert werden.
- $\nu(P)$ heißt Interpretation von P

Semantik - Beispiel

- $\nu_0 = \{x \mapsto 0, y \mapsto 1\}$

Beispiel (Evaluation von Formeln)

$$
\begin{aligned}
& \nu(x \oplus y) \\
= \; & \nu(\neg(x \leftrightarrow y)) \\
= \; & 1 - \nu(x \leftrightarrow y) \\
= \; & 1 - \nu((x \to y) \wedge (y \to x)) \\
= \; & 1 - \texttt{MIN}(\nu(x \to y), \nu(y \to x)) \\
= \; & 1 - \texttt{MIN}(\nu(\neg x \vee y), \nu(\neg y \vee x)) \\
= \; & 1 - \texttt{MIN}(\texttt{MAX}(\nu(\neg x), \nu(y)), \texttt{MAX}(\nu(\neg y), \nu(x))) \\
= \; & 1 - \texttt{MIN}(\texttt{MAX}(1 - \nu(x), 1), \texttt{MAX}(1 - \nu(y), 0)) \\
= \; & 1 - \texttt{MIN}(\texttt{MAX}(1 - 0, 1), \texttt{MAX}(1 - 1, 0)) \\
= \; & 1 - \texttt{MIN}(1, 0) \\
= \; & 1 - 0 \\
= \; & 1
\end{aligned}
$$

Semantik

Definition (Erfüllbarkeit)

Eine Formel P heißt **erfüllbar**, wenn eine Variablenbelegung $\nu_0 : var(P) \to \mathbb{B}$ existiert, so dass $\nu(P) = 1$ (ν_0 ist ein Modell von P)

- Notation: $\nu_0 \models P$ (oder auch $\models_{\nu_0} P$)

Definition (Tautologie)

Eine Formel P ist eine **Tautologie** (oder heißt **allgemeingültig**), falls für alle $\nu_0 : var(P) \to \mathbb{B}$ gilt, dass $\nu_0 \models P$

- Notation: $\models P$

Definition (Kontradiktion)

Eine Formel P ist eine **Kontradiktion** (oder heißt **unerfüllbar**), falls kein $\nu_0 : var(P) \to \mathbb{B}$ existiert, so dass $\nu_0 \models P$

Semantik

Definition (Semantische Folgerung)

Eine Formel Q **folgt semantisch** aus einer Formel P, wenn für jede Variablenbelegung $\nu_0 : var(P) \cup var(Q) \to \mathbb{B}$ gilt: Ist $\nu(P) = 1$, dann ist auch $\nu(Q) = 1$.

- Notation: $P \models Q$
  - Satz: Äquivalent dazu ist: $\models P \to Q$
- Notation: $P \not\models Q$ bedeutet, dass (mindestens) ein ν_0 existiert, für das $\nu(P) = 1$, aber $\nu(Q) = 0$. (Dieses ν_0 stellt dann ein Gegenbeispiel dar.)
- Oft folgt Q aus einer ganzen Reihe von Formeln (z.B. von Axiomen) $P_1, \ldots, P_n$.
- Notation: $P_1, \ldots, P_n \models Q$. Äquivalent dazu: $\models (P_1 \wedge \ldots \wedge P_n) \to Q$.

Definition (Semantische Äquivalenz)

Zwei Formeln P und Q sind **semantisch äquivalent**, falls $P \models Q$ und $Q \models P$ gilt.

- Notation: $P \equiv Q$
- Satz: Äquivalent dazu: $\models P \leftrightarrow Q$

Wahrheitstabellen

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

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

- P ist **erfüllbar** aber keine **Tautologie**
- Problem: n Variablen benötigen 2^n Tabellenzeilen, daher nicht für große Formeln geeignet

Äquivalenzumformungen

- Doppelte Negation

$$\neg\neg P \equiv P$$

- Distributivgesetze

$$P \vee (Q \wedge R) \equiv (P \vee Q) \wedge (P \vee R)$$

$$P \wedge (Q \vee R) \equiv (P \wedge Q) \vee (P \wedge R)$$

- Absorptionsgesetze

$$P \vee (P \wedge Q) \equiv P$$

$$P \wedge (P \vee Q) \equiv P$$

- DeMorgansche Gesetze

$$\neg(P \wedge Q) \equiv \neg P \vee \neg Q$$

$$\neg(P \vee Q) \equiv \neg P \wedge \neg Q$$

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

Basen von Operatoren

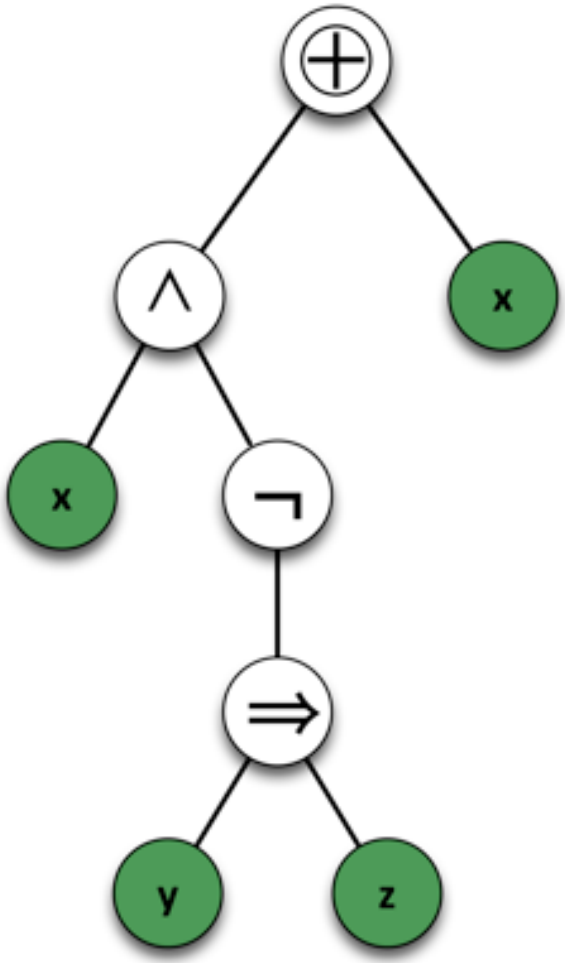
- Operatoren lassen sich durch andere ausdrücken
- Frage: Welche sind ausreichend?
- Was wir bereits wissen: $\rightarrow, \leftrightarrow$ und $\oplus$ lassen sich durch $\neg, \wedge, \vee$ ausdrücken:
  - $P \rightarrow Q \equiv \neg P \vee Q$
  - $P \leftrightarrow Q \equiv (P \rightarrow Q) \wedge (Q \rightarrow P) \equiv (\neg P \vee Q) \wedge (\neg Q \vee P)$
  - $P \oplus Q \equiv \neg(P \leftrightarrow Q) \equiv \neg((\neg P \vee Q) \wedge (\neg Q \vee P))$
- $\vee$ lässt sich mit der DeMorgan Regel eliminieren

$\neg, \wedge$ ist eine minimale Basis

- d.h. sämtliche aussagenlogische Formeln lassen sich unter Verwendung von $\neg$ und $\wedge$ darstellen
- weitere minimale Basen: $\{\neg, \vee\}, \{\wedge, \oplus\}, \{\vee, \leftrightarrow\}, \{\overline{\wedge}\}, \{\overline{\vee}\}$

Darstellung von Formeln

- Als Zeichenreihen (wie auf den letzten Folien)
- Als Bäume

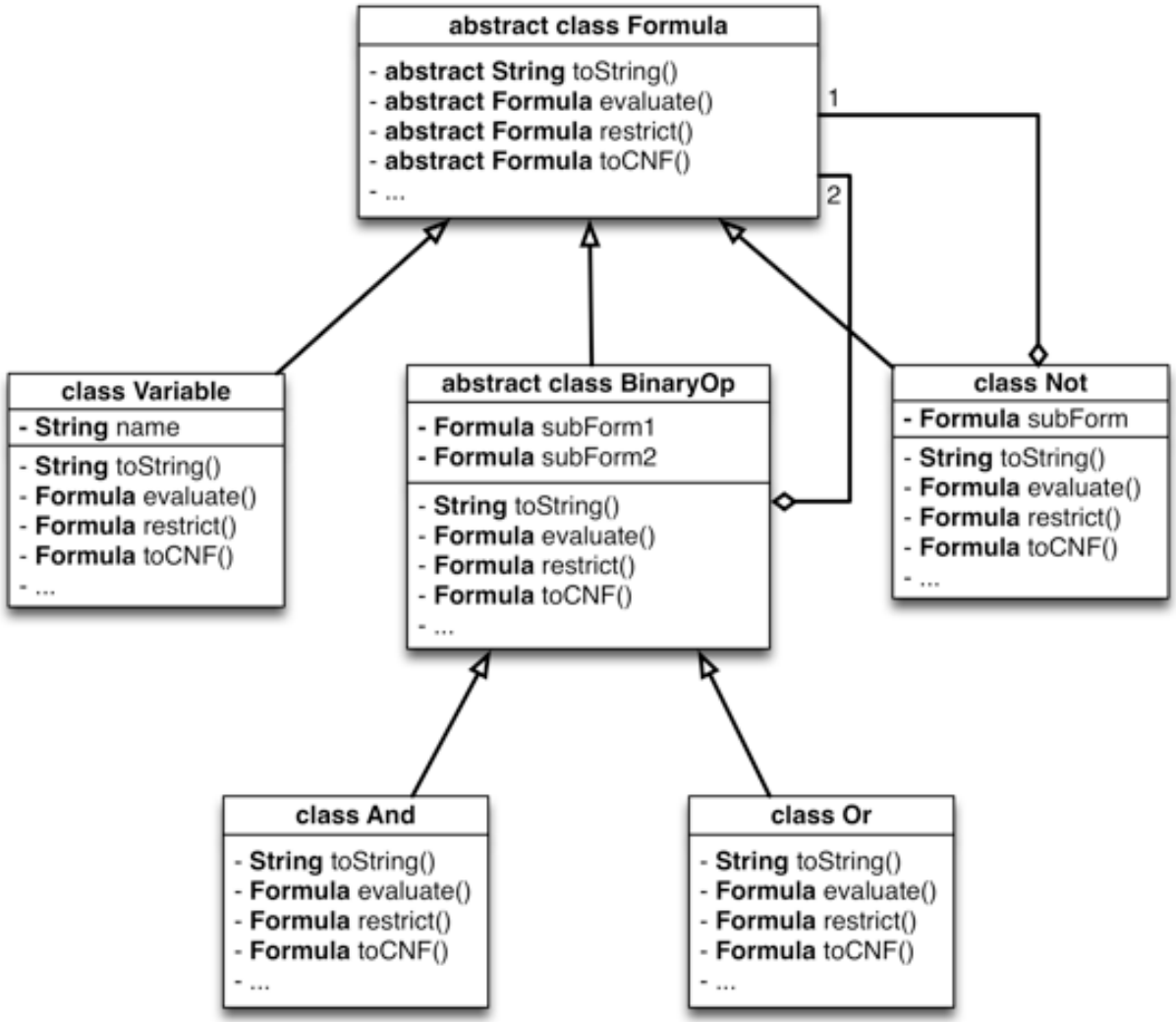


$$P = ((x \wedge \neg(y \rightarrow z)) \oplus x)$$

- Innere Knoten: Operatoren
- Blätter: Aussagenvariablen oder Konstanten

Formeln in Java

- Abstrakte Basisklasse: `Formula`
- Abstrakte Basisklasse für 2-stellige Operationen: `BinaryOp`
- Für jeden Knotentyp eine abgeleitete Klasse: `Variable`, `Not`, `And`,...



Das SAT-Problem

Fragestellung: Ist eine gegebene Formel P erfüllbar oder nicht **(SATisfiability)**.

Beispiel (SAT Probleme)

- $(x \vee y) \wedge (\neg x \vee \neg y)$ ist erfüllbar (z.B. $\{x \mapsto 1, y \mapsto 0\}$)
- $(x \vee y) \wedge (\neg x \vee \neg y) \wedge y \wedge x$ ist nicht erfüllbar
- $x \wedge y \vee \neg z \leftrightarrow x \oplus \neg z \wedge \neg y$ ist nicht mehr durch einfaches Hinschauen lösbar...

Vorschau (mehr dazu in zwei Wochen): Das SAT-Problem ist (mit großer Wahrscheinlichkeit) nicht in polynomialer Zeit entscheidbar.

Wer einen polynomiellen Algorithmus für SAT findet, bekommt **1 Mio. $**[1]

[1] `http://www.claymath.org/millennium/P_vs_NP/`

Ein kleines Anwendungsbeispiel - 1

3 Lehrer, 3 Fächer

- Albrecht (a) gibt Französisch (F) und Geschichte (G)
- Bert (b) gibt Englisch (E) und Französisch (F)
- Christine (c) gibt alle drei Fächer (E, F, G)

Wir wollen die Vorlieben der Lehrer beachten:

- Wenn Bert Französisch gibt, will Christine Englisch geben
- Wenn Christine Englisch gibt, will Albrecht kein Geschichte geben

Jeder Lehrer darf nur ein Fach geben und jedes Fach muss unterrichtet werden

Codierung

Variable a_F bedeutet, dass Lehrer A das Fach F gibt

- Anhand der Fächerkombinationen gibt es 7 Variablen:

$$a_F, a_G, b_E, b_F, c_E, c_F, c_G$$

Ein kleines Anwendungsbeispiel - 2

Codierung Teil 1: Jeder Lehrer darf nur ein Fach geben...

- $P_1 = (a_F \oplus a_G)$ Lehrer a gibt entweder F oder G
- $P_2 = (b_E \oplus b_F)$ Lehrer b gibt entweder E oder F
- $P_3 = (c_E \to \neg c_F \wedge \neg c_G) \wedge (c_F \to \neg c_E \wedge \neg c_G) \wedge (c_G \to \neg c_E \wedge \neg c_F)$

Codierung Teil 2: Jedes Fach muss unterrichtet werden...

- $P_4 = a_F \vee b_F \vee c_F$ Französisch (F) wird unterrichtet
- $P_5 = a_G \vee c_G$ Geschichte (G) wird unterrichtet
- $P_6 = b_E \vee c_E$ Englisch (E) wird unterrichtet

Codierung der Vorlieben:

- $P_7 = b_F \to c_E$ Wenn Bert Französisch, dann Christine Englisch
- $P_8 = c_E \to \neg a_G$ Wenn Christine Englisch, dann Albrecht kein Geschichte

Gesamte Formel: $P = P_1 \wedge P_2 \wedge P_3 \wedge P_4 \wedge P_5 \wedge P_6 \wedge P_7 \wedge P_8$

- Erfüllbar! (z.B mit $\{b_E \mapsto 1, a_F \mapsto 1, c_G \mapsto 1\}$, alle anderen Variablen 0)

Normalformen

Warum Normalformen:

- Einfachere Inferenzmechanismen möglich
- Einheitliche Darstellung von Formeln
- Einfachere Datenstrukturen zum Speichern von Formeln
- Einfachere Dateiformate zum Speichern von Formeln

Im folgenden:

- Negationsnormalform (NNF)
- Disjunktive Normalform (DNF)
- Konjunktive Normalform (CNF)

Vor allem CNF spielt wichtige Rolle im Bereich SAT-Solving. Alle *state-of-the-art* Solver benutzen intern CNF. Oft ist schon eine Menge von Constraints gegeben. Dann kann jeder Constraint separat in CNF umgeformt werden. Menschen benutzen of DNF um sich zu vergegenwärtigen, wann eine komplexe Bedingung zutrifft.

Negationsnormalform

Negation nur direkt vor Aussagenvariablen (d.h. nicht vor komplexen Formeln)

Beispiel (Negationsnormalform)

- $\neg(x \vee (y \wedge \neg z))$ nicht in NNF (wegen $\neg(x...)$)
- $\neg x \wedge (\neg y \vee z)$ ist in NNF

Algorithmus zur Umformung in NNF

$$\mathrm{nnf}(\varphi) := \begin{cases} \varphi & \text{falls } \varphi = P \text{ oder } \varphi = \neg P, \quad P \in \mathit{var}(\varphi) \\ \mathrm{nnf}(P) \odot \mathrm{nnf}(Q) & \text{falls } \varphi = P \odot Q, \quad \odot \in \{\wedge, \vee\} \\ \mathrm{nnf}(P) & \text{falls } \varphi = \neg\neg P \\ \mathrm{nnf}(\neg P) \wedge \mathrm{nnf}(\neg Q) & \text{falls } \varphi = \neg(P \vee Q) \\ \mathrm{nnf}(\neg P) \vee \mathrm{nnf}(\neg Q) & \text{falls } \varphi = \neg(P \wedge Q) \end{cases}$$

Disjunktive Normalform (DNF)

Formel ist eine Disjunktion (Oder) von Konjunktionen (Und)

- Disjunktion von **Mintermen** (ihr Wert ist das Minimum der Werte der Literale)

Beispiel (DNF)

- $(x_1 \wedge y_1 \wedge z_1) \vee (x_2 \wedge y_2) \vee z_2$

Algorithmus zur Umformung in DNF (naiv)

Wende folgende Regel so lange wie möglich auf NNF an:

- **[DNF1]** $P \wedge (Q \vee R)$ wird zu $(P \wedge Q) \vee (P \wedge R)$

Vorteile:

- Positive Aufzählung der Eins-Stellen („was geht“)
- Für Erfüllbarkeitstest (SAT) muss nur ein Minterm erfüllbar sein

Nachteil:

- Kann exponentielles Wachstum gegenüber der Originalformel haben (s.o.: Teilformel P verdoppelt sich).

Konjunktive Normalform (CNF)

Formel ist eine Konjunktion (Und) von Disjunktionen (Oder)

- Konjunktion von **Klauseln** / **Maxtermen**

Beispiel (CNF)

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

Algorithmus zur Umformung in CNF (naiv)

Wende folgende Regel so lange wie möglich auf NNF an:

- **[CNF1]** $P \vee (Q \wedge R)$ wird zu $(P \vee Q) \wedge (P \vee R)$

Vorteile:

- Negative Aufzählung der Nullstellen („was geht nicht“)
- Klauseln sind Randbedingungen (constraints) der Erfüllbarkeit

Nachteil:

- Kann exponentielles Wachstum gegenüber der Originalformel haben (s.o.: Teilformel P verdoppelt sich).

CNF Transformation Beispiel

Beispiel (CNF Transformation)

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

Schritt 1: NNF

$$\begin{aligned}
& (x \wedge z) \vee \neg(y \vee (x \vee z)) \text{ [NNF1]} \\
\equiv\ & (x \wedge z) \vee (\neg y \wedge \neg(x \vee z)) \text{ [NNF1]} \\
\equiv\ & (x \wedge z) \vee (\neg y \wedge (\neg x \wedge \neg z))
\end{aligned}$$

Schritt 2: CNF

$$\begin{aligned}
& (x \wedge z) \vee (\neg y \wedge (\neg x \wedge \neg z)) \text{ [CNF1]} \\
\equiv\ & ((x \wedge z) \vee \neg y) \wedge ((x \wedge z) \vee (\neg x \wedge \neg z)) \text{ [CNF1]} \\
\equiv\ & (x \vee \neg y) \wedge (z \vee \neg y) \wedge ((x \wedge z) \vee \neg x) \wedge ((x \wedge z) \vee \neg z) \text{ [CNF1]} \\
\equiv\ & (x \vee \neg y) \wedge (z \vee \neg y) \wedge (x \vee \neg x) \wedge (z \vee \neg x) \wedge (x \vee \neg z) \wedge (z \vee \neg z) \\
\equiv\ & (x \vee \neg y) \wedge (z \vee \neg y) \wedge (z \vee \neg x) \wedge (x \vee \neg z)
\end{aligned}$$

CNF Darstellung

- Operatoren in CNF durch Form bestimmt:
  - $\vee$ immer in Klauseln, $\wedge$ immer zwischen Klauseln

Vereinfachte Mengenschreibweise

Idee: Operatoren weglassen (da klar); außerdem gilt $a \vee a \equiv a$ und $C \wedge C \equiv C$

- Formel: Menge von n Klauseln $\{C_1, C_2, ..., C_n\}$
- Klausel: Menge von m Literalen $\{x_1, x_2, ..., x_m\}$

Beispiel (Mengenschreibweise)

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

Mengenschreibweise:

$$P' = \{\{x, \neg y\}, \{z, \neg y\}, \{z, \neg x\}, \{x, \neg z\}\}$$

Automatisches Beweisen mit Resolution

- Die übliche Beweis-Aufgabe ist, zu zeigen, dass $P \models Q$ bzw. $P_1, \ldots, P_n \models Q$.
- Resolution ist ein Kalkül zur mechanischen Herleitung einer **Resolvente** R aus zwei **Elternklauseln** M und D, sodass $M, D \models R$.

Definition (Resolution)

Seien $M = \{x_1, x_2, ..., x_m\}$ und $D = \{y_1, y_2, ..., y_n\}$ zwei Klauseln, die ein Paar von komplementären Literalen x_i und $y_k \equiv \neg x_i$ enthalten. Dann entsteht durch **Resolution** von M und D **über** x_i die **Resolvente** $R = (M \setminus \{x_i\}) \cup (D \setminus \{y_k\})$. Im Fall einer **Unit Klausel** $M = \{x\}$ haben wir eine **Unit Resolution**. Im Fall zweier Unit Eltern ist R die **Leere Klausel** $\{\}$, üblicherweise mit $\square$ bezeichnet.

Satz

Es gilt $M, D \models R$.

Beweis: Sei ν_0 gegeben sodass $\nu(M \wedge D) = 1$. Dann gibt es 2 Fälle:

- Falls $\nu(x_i) = 0$, dann ist $\nu(M \setminus \{x_i\}) = 1$.
- Falls $\nu(x_i) = 1$, dann ist $\nu(y_k) = \nu(\neg x_i) = 0$ und somit $\nu(D \setminus \{y_k\}) = 1$.

In beiden Fällen ist also $\nu(R) = 1$.

Erfüllbarkeitsäquivalenz

Definition (Erfüllbarkeitsäquivalenz)

Zwei Formeln P und Q sind **erfüllbarkeitsäquivalent** (e-äquiv.), falls gilt:
P ist erfüllbar genau dann, wenn Q erfüllbar ist

- P e-äquiv. Q gdw. sowohl P als auch Q sind erfüllbar *oder* sowohl P als auch Q sind nicht erfüllbar.
- Falls erfüllende Belegungen für P und für Q existieren, dürfen diese völlig unterschiedlich sein.
- Erfüllbarkeitsäquivalenz ist schwächer als Äquivalenz. Bei Äquivalenz müssen die erfüllenden Belegungen identisch sein.
- Semantische Äquivalenz schließt also Erfüllbarkeitsäquivalenz ein

Überprüfung der Erfüllbarkeitsäquivalenz

- Für konkrete Formeln P und Q wird einfach geprüft, ob SAT(P) = SAT(Q).
- Sind P und Q beliebige Formeln, so muss es zu jeder erfüllenden Belegung von P eine erfüllende Belegung von Q geben und umgekehrt. Dies kann nur für bestimmte Bauarten von P und Q gelingen.

Erfüllbarkeitsäquivalenz

Beispiel

- x und y sind e-äquiv., **nicht** semantisch äquivalent
- x und $\neg x$ sind e-äquiv., **nicht** semantisch äquivalent
- $\top$ und x sind e-äquiv., **nicht** semantisch äquivalent
- $\bot$ und x sind **nicht** e-äquiv., **nicht** semantisch äquivalent
- $x \to y$ und $x \leftarrow y$ sind e-äquiv., **nicht** semantisch äquivalent
- $x \to y$ und $\neg x \vee y$ sind e-äquiv. und semantisch äquivalent

Beispiel

- Allgemeine P und Q sind nicht e-äquiv., z.B. wenn $P = \top$ und $Q = \bot$.
- Im Allgemeinen sind P und $\neg P$ nicht e-äquiv., z.B. wenn $P = \top$ oder $P = \bot$.
- P und $P \wedge x$ sind e-äquiv. falls x sonst nicht in P vorkommt. Falls es eine erfüllende Belegung für P gibt, dann auch für $P \wedge x$ (erweitere um $x = 1$). Falls es eine erfüllende Belegung für $P \wedge x$ gibt, so erfüllt diese auch P.
- P und $P \vee x$ sind i.A. nicht e-äquiv.. Falls P nicht erfüllbar ist, so ist $P \vee x$ trotzdem erfüllbar.

Tseitin Verfahren zur CNF Berechnung

Problem: Verdopplung von A in **[CNF1]** $A \vee (B \wedge C)$ wird zu $(A \vee B) \wedge (A \vee C)$
Idee: Führe Hilfsvariable für Konjunktion ein

Tseitin Verfahren

- Bilde Negationsnormalform
- Für $P = A \vee (B \wedge C)$ setze $x \leftrightarrow (B \wedge C)$,
  x muss neue Variable sein, d.h. $x \notin \mathit{var}(P)$
- Setze: $P' := (A \vee x) \wedge (x \leftrightarrow (B \wedge C))$
- P ist erfüllbarkeitsäquivalent zu P' (siehe Beweis auf nächster Folie)
- Anzahl erfüllender Belegungen (sog. *model counting*) bleibt gleich.

Nun ist

$$\begin{aligned} P' &= (A \vee x) \wedge (x \leftrightarrow (B \wedge C)) \\ &\equiv (A \vee x) \wedge (\neg x \vee (B \wedge C)) \wedge (x \vee \neg(B \wedge C)) \\ &\equiv (A \vee x) \wedge (\neg x \vee B) \wedge (\neg x \vee C) \wedge (x \vee \neg B \vee \neg C) \end{aligned}$$

Resultat: Menge von Klauseln mit zusätzlichen Variablen

Tseitinverfahren (ctd.)

Erfüllbarkeitsäquivalenz der Tseitin-Transformation

$P = A \vee (B \wedge C)$ ist erfüllbarkeitsäquivalent zu
$P' = (A \vee x) \wedge (\neg x \vee B) \wedge (\neg x \vee C) \wedge (x \vee \neg B \vee \neg C)$.
Beweis:

$\Rightarrow$: Sei β erfüllende Belegung für P.

- Sei $\beta(B \wedge C) = 0$. Dann $\beta(B) = 0$ oder $\beta(C) = 0$ und $\beta(A) = 1$. Es ex. eindeutige Erweiterung $\beta' = \beta \cup \{x \mapsto 0\}$ mit $\beta'(P') = 1$.
  (Bem.: $\beta'(x) = 1$ funktioniert nicht)
- Sei $\beta(B \wedge C) = 1$. Dann $\beta(B) = \beta(C) = 1$. Dann ex. eindeutige Erweiterung β' mit $\beta'(x) = 1$ sodass $\beta'(P') = 1$.
  (Bem.: $\beta'(x) = 0$ funktioniert nicht)

Bem.: $P \not\models P'$, da die „falsche“ Erweiterung β' jeweils immer noch P erfüllt, nicht aber P'.

$\Leftarrow$: Sei β' erfüllende Belegung für P'.

- Sei $\beta'(x) = 0$. Dann auch $\beta'(A) = 1$ und somit $\beta'(P) = 1$.
- Sei $\beta'(x) = 1$. Dann $\beta'(B) = \beta'(C) = 1$ und somit auch $\beta'(P) = 1$.

Bem.: es folgt, dass $P' \models P$.

Tseitinverfahren (ctd.)

Lemma von Plaisted-Greenbaum

$P' = (A \vee x) \wedge (\neg x \vee B) \wedge (\neg x \vee C) \wedge (x \vee \neg B \vee \neg C)$
ist erfüllbarkeitsäquivalent zu
$P'' = (A \vee x) \wedge (\neg x \vee B) \wedge (\neg x \vee C)$
Beweis:

$\Rightarrow$: P' impliziert P''.

$\Leftarrow$: Sei β'' eine erfüllende Belegung für P''.

- Sei $\beta''(x) = 1$. Dann ist auch $\beta''(x \vee \neg B \vee \neg C) = 1$ und daher $\beta''(P') = 1$.
- Sei $\beta''(x) = 0$. Dann ist nicht notwendigerweise $\beta''(x \vee \neg B \vee \neg C) = 1$, sondern nur, wenn auch $\beta''(B) = 0$ oder $\beta''(C) = 0$. Falls aber $\beta''(B) = 1$ und $\beta''(C) = 1$, dann betrachte eine weitere Belegung β''', die mit β'' identisch ist bis auf die umgekehrte Belegung von x, also $\beta'''(x) = 1$. Da x nicht in B, C vorkommt, ist $\beta'''(B) = \beta''(B) = 1$ und $\beta'''(C) = \beta''(C) = 1$. Also ist insgesamt $\beta'''(P') = 1$.

Tseitinverfahren (ctd.)

Verfahren von Tseitin-Plaisted-Greenbaum

Ersetze $P = A \vee (B \wedge C)$
erfüllbarkeitsäquivalent zu
$P'' = (A \vee x) \wedge (\neg x \vee B) \wedge (\neg x \vee C)$
solange bis CNF erreicht.

Ergebnis:

- Benötigt weniger Klauseln als reines Tseitin Verfahren
- ABER: Falls es eine erfüllende Belegung von P gibt für die $\beta(A) = 1$ und $\beta(B) = 1$ und $\beta(C) = 1$, dann gibt es hierfür zwei erweiterte Belegungen, die P'' erfüllen, nämlich $\beta \cup \{x \mapsto 1\}$ und $\beta \cup \{x \mapsto 0\}$.
- Einführung von neuen Variablen führt dazu, dass es bei P'' mehr erfüllende Belegungen geben kann als bei original P (Problem bei Model Counting von P, d.h. beim Zählen der Anzahl erfüllender Belegungen von P)

Dimacs Format für CNF

DIMACS

- Center for Discrete Mathematics and Theoretical Computer Science
- Rutgers and Princeton Universities, AT&T, Alcatel-Bell Labs, NEC, ...
- sowie: HP Labs, IBM, Microsoft, Georgia Tech, Rensselaer, ...

Kompaktes Dateiformat zur Speicherung von CNF Formeln

- Standard für alle SAT Benchmarks

Dimacs Format

1. Optionale Kommentarzeilen: `c Kommentar`
2. Präambel: p cnf n m mit
   - n Anzahl der Variablen
   - m Anzahl der Klauseln
3. Klauseln:
   - Liste der Literale, durch Leerzeichen getrennt, 0 als Abschluss
   - Variablen repräsentiert durch Integers (> 0), '-' als Negation

Beispiel für Dimacs

Beispiel (Dimacs Format)

$$P = \{\{x_1, x_2, \neg x_3\}, \{x_3, \neg x_4\}, \{\neg x_1, \neg x_3, x_4\}, \{\neg x_2, x_3, x_5\}\}$$

```
c Dies ist eine Formel in CNF
c mit 5 Variablen und 4 Klauseln.
p cnf 5 4
1 2 -3 0
3 -4 0
-1 -3 4 0
-2 3 5 0
```