SAT-Solving und Anwendungen

Einführung / Aussagenlogik

Prof. Dr. Wolfgang Küchlin
Dipl. Inf. Christoph Zengler

Universität Tübingen

12. April 2011

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 $\mathbb{B} = \{\mathbf{T}, \mathbf{F}\}$ 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): $\neg, \wedge, \vee, \oplus, \rightarrow, \leftrightarrow$
- 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?
- Menge der Booleschen Wahrheitswerte: $\mathbb{B} = \{\mathbf{T}, \mathbf{F}\}$
- Dazu: Belegung der Aussagenvariablen mit **T** (wahr) oder **F** (falsch)
  - Variablenbelegung einer Variable x ist Funktion $\nu_0 : \mathcal{P} \to \mathbb{B}$
  - Notation $x \mapsto \mathbf{T}$ oder $x \mapsto \mathbf{F}$
- Funktion ν_0 kann rekursiv erweitert werden auf Formeln P ($\nu : \mathcal{F} \to \mathbb{B}$):
  - $\nu(x) = \nu_0(x)$ für $x \in \mathcal{P}$
  - $\nu(\top) = \mathbf{T}$
  - $\nu(\bot) = \mathbf{F}$
  - $\nu(\neg P) = \texttt{if } \nu(P) = \mathbf{T} \texttt{ then } \mathbf{F} \texttt{ else } \mathbf{T}$
  - $\nu(P \vee Q) = \texttt{if } \nu(P) = \mathbf{F} \texttt{ then } \nu(Q) \texttt{ else } \mathbf{T}$
  - $\nu(P \wedge Q) = \nu(\neg(\neg P \vee \neg Q))$
  - $\nu(P \to Q) = \nu(\neg F \vee G)$
  - $\nu(P \leftrightarrow Q) = \nu((F \to G) \wedge (G \to F))$
  - $\nu(P \oplus Q) = \nu(\neg(F \leftrightarrow G))$
- $\nu(P)$ heißt *Interpretation von P*

Semantik - Beispiel

- $\nu_0 = \{x \mapsto \mathbf{F}, y \mapsto \mathbf{T}\}$

Beispiel (Evaluation von Formeln)

$$
\begin{aligned}
& \nu(x \oplus y) \\
= \; & \nu(\neg(x \leftrightarrow y)) \\
= \; & \nu(\neg((x \rightarrow y) \wedge (y \rightarrow x))) \\
= \; & \nu(\neg((\neg x \vee y) \wedge (\neg y \vee x))) \\
= \; & \texttt{if } \nu((\neg x \vee y) \wedge (\neg y \vee x)) = \mathbf{F} \texttt{ then } \mathbf{T} \texttt{ else } \mathbf{F} \\
= \; & \ldots \\
= \; & \texttt{if } \mathbf{F} = \mathbf{F} \texttt{ then } \mathbf{T} \texttt{ else } \mathbf{F} \\
= \; & \mathbf{T}
\end{aligned}
$$

Semantik

Definition (Erfüllbarkeit)

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

- Notation: $\nu_0 \models P$

Definition (Tautologie)

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

- Notation: $\models P$

Definition (Kontradiktion)

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

- Notation: $\not\models P$

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

- Distributivgesetze

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

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

- Absorptionsgesetze

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

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

- DeMorgansche Gesetze

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

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

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

Basen von Operatoren

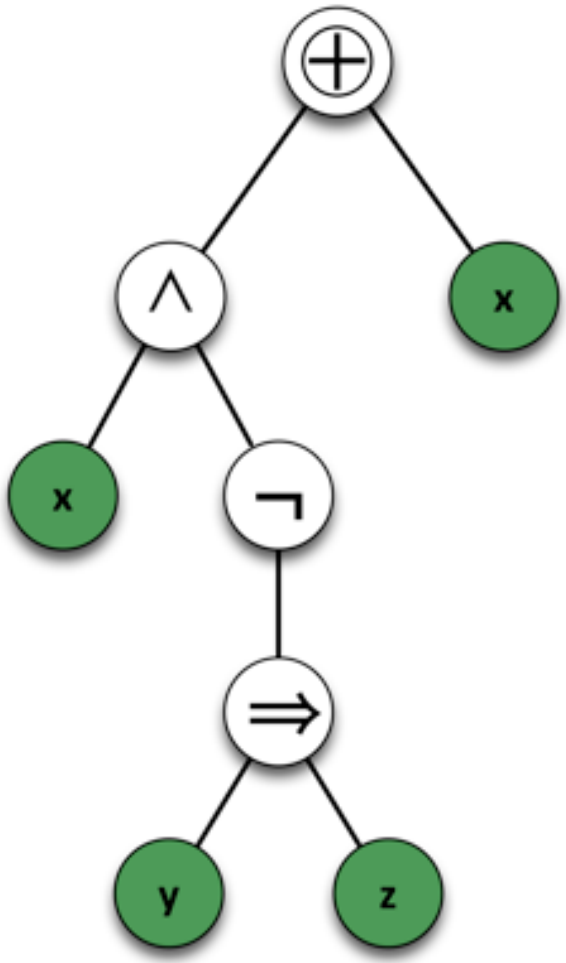
- Operatoren lassen sich durch andere ausdrücken
- Frage: Welche sind ausreichend?
- Was wir bereits wissen: $\rightarrow, \leftrightarrow$ und $\oplus$ lassen sich durch $\neg, \wedge, \vee$ ausdrücken:
  - $P \rightarrow Q = \neg P \vee Q$
  - $P \leftrightarrow Q = P \rightarrow Q \wedge Q \rightarrow P = (\neg P \vee Q) \wedge (\neg Q \vee P)$
  - $P \oplus Q = \neg(P \leftrightarrow Q) = \neg((\neg P \vee Q) \wedge (\neg Q \vee P))$
- $\vee$ 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

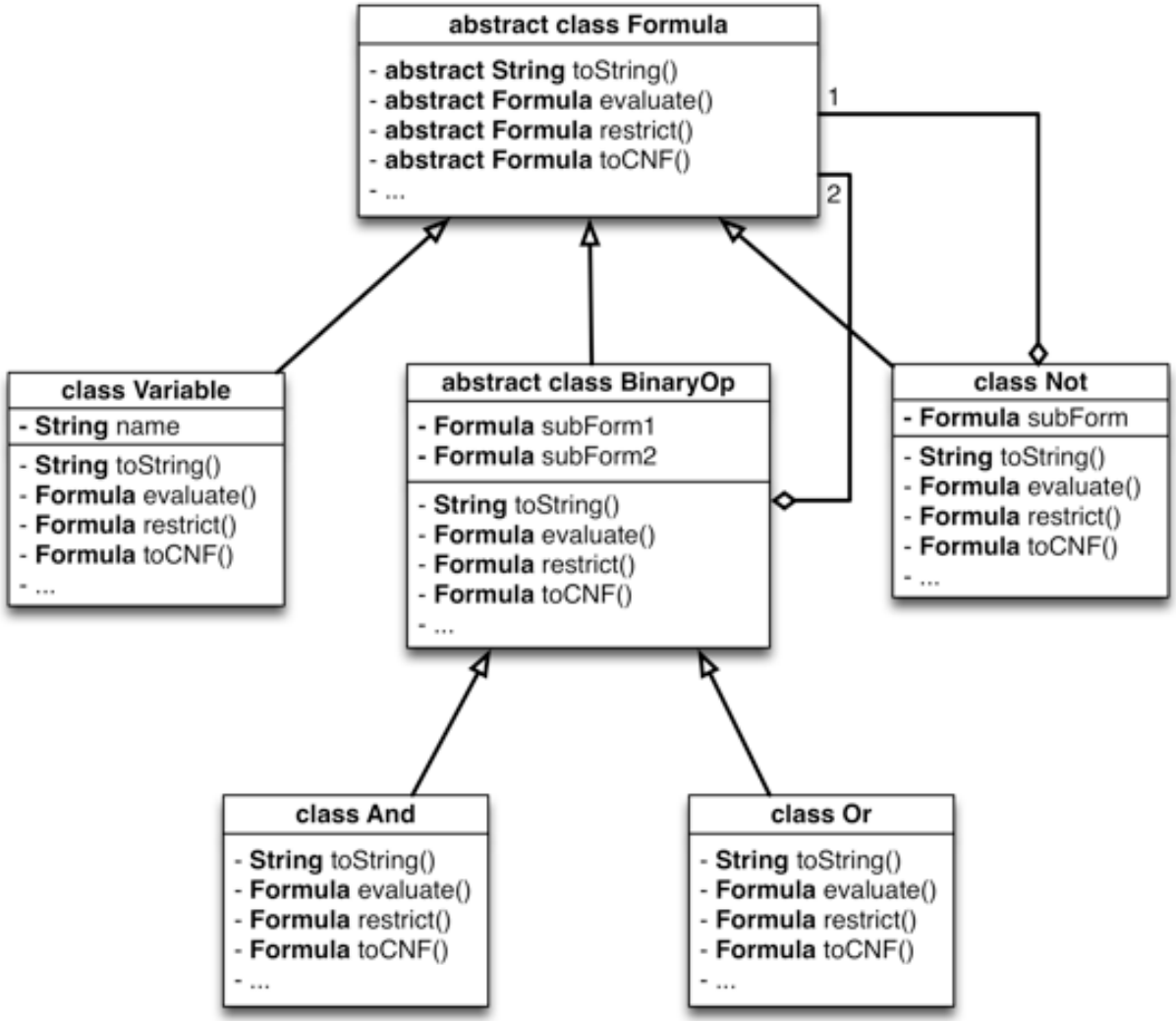


$$P = ((x \wedge \neg(y \rightarrow z)) \oplus x)$$

- 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. $\{x \mapsto \mathbf{T}, y \mapsto \mathbf{F}\}$)
- $(x \vee y) \wedge (\neg x \vee \neg y) \wedge y \wedge x$ 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
- $P_3 = (c_E \rightarrow \neg c_F \wedge \neg c_G) \wedge (c_F \rightarrow \neg c_E \wedge \neg c_G) \wedge (c_G \rightarrow \neg c_E \wedge \neg c_F)$

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:

- $P_7 = b_F \rightarrow c_E$ Wenn Bert Französisch, dann Christine Englisch
- $P_8 = c_E \rightarrow \neg a_G$ 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 $\{b_E \mapsto \mathbf{T}, a_F \mapsto \mathbf{T}, c_G \mapsto \mathbf{T}\}$, alle anderen Variablen $\mathbf{F}$)

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

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

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

Disjunktive Normalform (DNF)

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

- Disjunktion von **Mintermen**

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

Konjunktive Normalform (CNF)

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

- Konjunktion von **Klauseln**

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

CNF Transformation Beispiel

Beispiel (CNF Transformation)

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

Schritt 1: NNF

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

Schritt 2: CNF

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

CNF Darstellung

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

Vereinfachte Mengenschreibweise

Idee: Operatoren weglassen (da klar)

- 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:

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

Tseitin Verfahren zur CNF Berechnung

Idee: Führe neue Variable für Konjunktion ein

Beispiel (Tseitin Verfahren)

$P = A \vee (B \wedge C)$

- Setze $x \leftrightarrow (B \wedge C)$, x muss neue Variable sein (d.h. $x \notin var(P)$)
- Dann: $P' = (A \vee x) \wedge (x \leftrightarrow (B \wedge C))$
- P' ist dann erfüllbarkeitsäquivalent zu P, nicht äquivalent

Einschub - Erfüllbarkeitsäquivalenz

Zwei Formeln P und Q sind erfüllbarkeitsäquivalent, wenn gilt: P ist erfüllbar genau dann wenn Q erfüllbar ist

- Erfüllbarkeitsäquivalenz ist schwächer als Äquivalenz
- Äquivalenz schließt Erfüllbarkeitsäquivalenz ein

Tseitinverfahren (ctd.)

Beispiel (Tseitin Verfahren - Fortsetzung)

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

- Die letzte Zeile ist erfüllbarkeitsäquivalent zu $P'' = (A \vee x) \wedge (\neg x \vee B) \wedge (\neg x \vee C)$

Das Verfahren:

- Ersetze P solange durch P'', bis CNF erreicht

Ergebnis:

- Keine Verdoppelung von A, kein exponentiells Wachstum
- ABER: Einführung von neuen Variablen führt dazu, dass es bei P'' mehr erfüllende Belegungen geben kann als bei original P (Problem bei Model Counting)

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
```