SAT-Solving und Anwendungen
Normalformen

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

Universität Tübingen

29. April 2009

Normalformen

Warum Normalformen:

- 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

Wende folgende Regeln so lange wie möglich an:

- **[NNF1]** $\neg(P \vee Q)$ wird zu $\neg P \wedge \neg Q$ *(deMorgan)*
- **[NNF2]** $\neg(P \wedge Q)$ wird zu $\neg P \vee \neg Q$ *(deMorgan)*
- **[NNF3]** $\neg\neg P$ wird zu P *(Doppelnegation)*
- **[NNF4]** $\neg\top$ wird zu $\bot$ und $\neg\bot$ wird zu $\top$

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)

$$P' = (A \vee x) \wedge (x \Leftrightarrow (B \wedge C))$$
$$= (A \vee x) \wedge (\neg x \vee (B \wedge C)) \wedge (x \vee \neg(B \wedge C))$$
$$= (A \vee x) \wedge (\neg x \vee B) \wedge (\neg x \vee C) \wedge (x \vee \neg B) \wedge (x \vee \neg C)$$

- 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
```