Automatisches Beweisen SS10
Konvertierung von aussagenlogischen Formeln in Normalformen

Prof. Dr. Wolfgang Küchlin
Wolfgang Braun

Universität Tübingen

22. Mai 2010

Einführung

- Wenn man eine Formel in konjunktive Normalform konvertieren möchte ist es möglich, dass die Formel exponentiell größer wird.
- Um dieses Problem besser angehen zu können wird hier ein alternatives Verfahren dargestellt, um die Formel leichter zu konvertieren.
- Die Methode besteht darin, dass man die Formel in eine Graphenstruktur überführt. Genauer gesagt wird die Formel in den sogenannten **h**orizontal-**p**ath-Graph und **v**ertical-**p**ath-Graph überführt.
  Siehe Peter B. Andrews: „An Introduction to Mathematical Logic and Type Theory“, Seite 52f.
- Die konjunktive- und disjunktive Normalform kann aus dieser Struktur direkt ausgelesen werden.

Wiederholung

Indem man folgende Berechnungen auf einer Formel durchführt, kann man sie in Negationsnormalform (NNF) umformen:

- $\neg(F \wedge G) \Rightarrow \neg F \vee \neg G$
- $\neg(F \vee G) \Rightarrow \neg F \wedge \neg G$

vp-Form

- entwickelt von Peter B. Andrews
- 2-dimensionale Aufschreibung von Formeln in NNF
  - Disjunktion horizontal
    $A \vee B$ wird dargestellt als $\begin{bmatrix} A & B \end{bmatrix}$
  - Konjuktion vertikal
    $A \wedge B$ wird dargestellt als $\begin{bmatrix} A \\ B \end{bmatrix}$

Beispiel

$F = (a \wedge \neg b) \vee (c \wedge (d \vee \neg f))$

vp-Form von F: $\left[\begin{bmatrix} a \\ \neg b \end{bmatrix} \quad \begin{bmatrix} & c \\ [d & \neg f] \end{bmatrix}\right]$

vp-Form (2)

Auslesen einer CNF/DNF aus der vp-Form

Idee: Lese horizontal/vertikal die Klauseln/Minterme ab.

Definition

Sei F eine Formel in NNF. Wir definieren $\mathcal{VP}(F)$ als die Menge der vertikalen Pfade durch F und $\mathcal{HP}(F)$ als die Menge der horizontalen Pfade durch F induktiv über den Formelaufbau:

- $\mathcal{VP}(F) = \{\langle F \rangle\}$, wenn F ein Literal ist.
- $\mathcal{VP}(F) = \mathcal{VP}(G) \cup \mathcal{VP}(H)\}$, wenn $F = G \vee H$.
- $\mathcal{VP}(F) = \{PQ \mid P \in \mathcal{VP}(G) \text{ und } Q \in \mathcal{VP}(H)\}$, wenn $F = G \wedge H$.

- $\mathcal{HP}(F) = \{\langle F \rangle\}$, wenn F ein Literal ist.
- $\mathcal{HP}(F) = \{PQ \mid P \in \mathcal{HP}(G) \text{ und } Q \in \mathcal{HP}(H)\}$, wenn $F = G \vee H$.
- $\mathcal{HP}(F) = \mathcal{HP}(G) \cup \mathcal{HP}(H)\}$, wenn $F = G \wedge H$.

Dabei bezeichnet PQ die Konkatenation der Pfade P und Q.

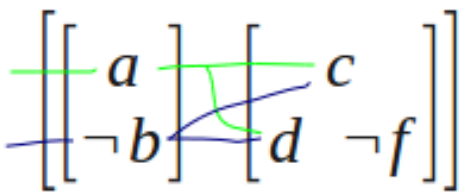
vp-Form (3)

Auslesen einer CNF/DNF aus der vp-Form (2)

- $\mathcal{VP}(F)$ liefert eine DNF von F: jeder Pfad liefert einen Term. Interpretiere hierzu $\mathcal{VP}(F) = \{\langle x_{1_1}, \ldots, x_{1_k}\rangle, \ldots, \langle x_{n_1}, \ldots, x_{n_l}\rangle\}$ als die Formel $(x_{1_1} \wedge \ldots \wedge x_{1_k}) \vee \ldots \vee (x_{n_1} \wedge \ldots \wedge x_{n_l})$.
- $\mathcal{HP}(F)$ liefert eine CNF von F: jeder Pfad liefert eine Klausel. Interpretiere hierzu $\mathcal{HP}(F) = \{\langle x_{1_1}, \ldots, x_{1_k}\rangle, \ldots, \langle x_{n_1}, \ldots, x_{n_l}\rangle\}$ als die Formel $(x_{1_1} \vee \ldots \vee x_{1_k}) \wedge \ldots \wedge (x_{n_1} \vee \ldots \vee x_{n_l})$.
- $\mathcal{VP}(F)$ bzw. $\mathcal{HP}(F)$ lassen sich rein mechanisch aus der vp-Form ablesen. Siehe Beispiel auf der nächsten Folie.

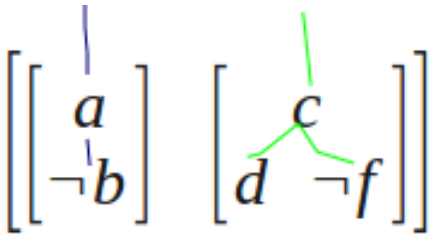
Beispiel

$F = (a \wedge \neg b) \vee (c \wedge (d \vee \neg f))$ mit vp-Form $\left[\begin{bmatrix} a \\ \neg b \end{bmatrix} \quad \begin{bmatrix} c \\ [d \quad \neg f] \end{bmatrix}\right]$



$\mathcal{HP}(F) = \{\langle a, c\rangle, \langle a, d, \neg f\rangle, \langle \neg b, c\rangle, \langle \neg b, d, \neg f\rangle\}$

$F = (a \vee c) \wedge (a \vee d \vee \neg f) \wedge (\neg b \vee c) \wedge (\neg b \vee d \vee \neg f)$



$\mathcal{VP}(F) = \{\langle a, \neg b\rangle, \langle c, d\rangle, \langle c, \neg f\rangle\}$

$F = (a \wedge \neg b) \vee (c \wedge d) \vee (c \wedge \neg f)$

weiteres Beispiel

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

- vp-Form von F: $\left[[x_1] \quad \left[\left[\begin{matrix} & x_2 \\ x_1 & & x_3 \end{matrix} \right] \quad \begin{bmatrix} x_2 \\ \neg x_1 \end{bmatrix} \right] \right]$
- CNF (HP):
  $(x_1 \vee x_2 \vee x_2) \wedge (x_1 \vee x_2 \vee \neg x_1) \wedge (x_1 \vee x_1 \vee x_3 \vee x_2) \wedge (x_1 \vee x_1 \vee x_3 \vee \neg x_1)$
  $\equiv (x_1 \vee x_2) \wedge (x_1 \vee x_3 \vee x_2)$
- DNF (VP): $x_1 \vee (x_2 \wedge x_1) \vee (x_2 \wedge x_3) \vee (x_2 \wedge \neg x_1)$

vp-Form im Computer

Die vp-Form lässt sich durch eine Pointerstruktur, genauer gesagt durch Graphen, im Computer realisieren, wobei für jede der beiden Lesarten (horizontal, vertikal) ein anderer (hp-, vp-) Graph entsteht.

Allgemeines

- Die Konstruktion des Graphens wird rekursiv über den Formelaufbau definiert. Die Formel muss in Negationsnormalform gegeben sein. **Für jedes Auftreten eines Literals wird ein neuer Knoten erstellt**.
- Jeder Knoten besteht aus einem Wert des entsprechenden Literals und aus seinen Kindern.
- Mit der folgenden Konstruktion kann man den hp- und vp-Graphen gleichzeitig aufbauen.
- Man kann jeden Knoten sowohl für den hp- als auch für den vp-Graphen verwenden. Man speichert sich für jeden Knoten sowohl die vp-Kinder als auch die hp-Kinder.

Konstruktion

horizontal-path-Graph

- Der hp-Graph von einem Literal l, ist ein einfacher Knoten n, wobei $Lit(n) = l$ gilt.
- Der hp-Graph von $\phi_1 \wedge \phi_2$ ist die Vereinigung von ϕ_1 und ϕ_2.
- Der hp-Graph von $\phi_1 \vee \phi_2$ ist die Vereinigung von ϕ_1 und ϕ_2 und die Blätter von ϕ_1 werden mit den Wurzelknoten von ϕ_2 mit einer Kante verbunden.

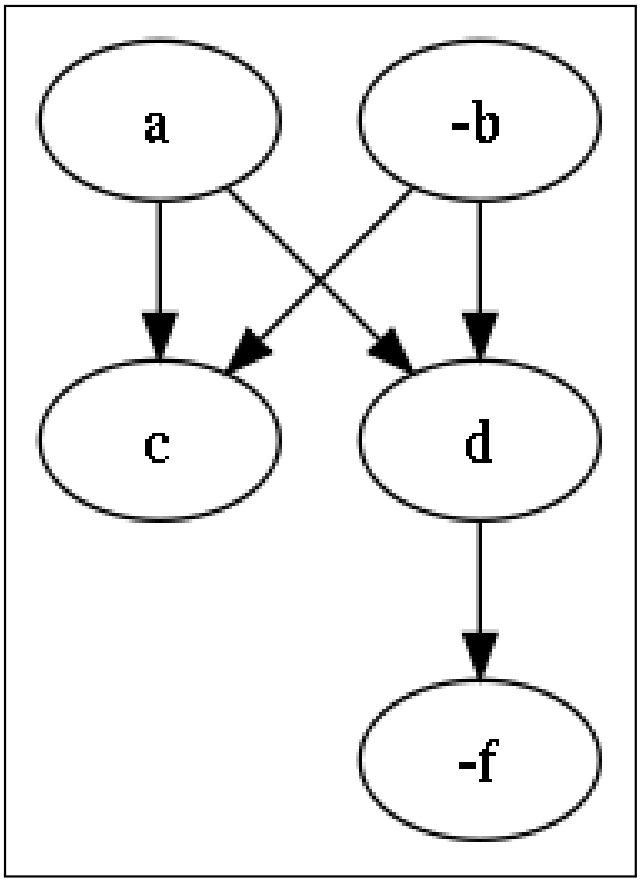
vertical-path-Graph

- Der vp-Graph von einem Literal l, ist ein einfacher Knoten n, wobei $Lit(n) = l$ gilt.
- Der vp-Graph von $\phi_1 \vee \phi_2$ ist die Vereinigung von ϕ_1 und ϕ_2.
- Der vp-Graph von $\phi_1 \wedge \phi_2$ ist die Vereinigung von ϕ_1 und ϕ_2 und die Blätter von ϕ_1 werden mit den Wurzelknoten von ϕ_2 mit einer Kante verbunden.

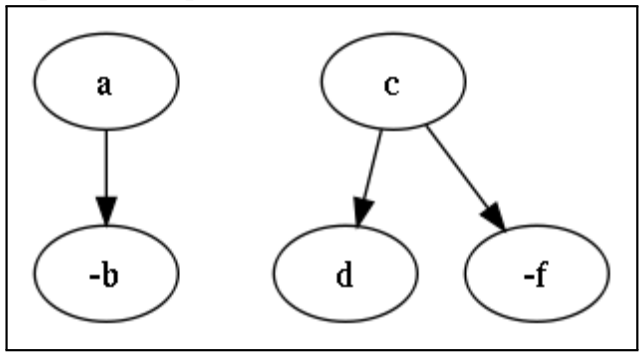
Beispiel

Gegeben sei die Formel $(a \wedge \neg b) \vee (c \wedge (d \vee \neg f))$. Mit der vorgestellten Konstruktion bekommt man den hp- und vp-Graphen.

hp-Graph



vp-Graph



Auslesen der Normalformen

Allgemein

- Wenn man den hp-Graphen benutzt bekommt man die konjunktive Normalform (KNF).
- Wenn man den vp-Graphen benutzt bekommt man die disjunktive Normalform (DNF).
- Man berechnet die Klauseln in der Normalform, indem man alle Pfade von jedem Wurzelknoten zu jedem Blattknoten aufsammelt. Jeder Pfad ist eine Klausel.

Beispiel (Am obigen Beispiel)

- Wenn man alle Pfade im hp-Graph aufsammelt, bekommt man die konjunktive Normalform $(a \vee c) \wedge (\neg b \vee c) \wedge (a \vee d \vee \neg f) \wedge (\neg b \vee d \vee \neg f)$
- Wenn man alle Pfade im vp-Graph aufsammelt, bekommt man die disjunktive Normalform $(a \wedge \neg b) \vee (c \wedge d) \vee (c \wedge \neg f)$

Weiterführendes

Anwendungen

- Es gibt auch Verfahren zur Bestimmung von Erfüllbarkeit von Formeln auf Negationsnormalform, die auf der Graphenstruktur arbeiten.
- Mit Hilfe dieses Verfahrens zur Konvertierung in konjunktive Normalform, ist es möglich Klauseln effektiv filtern zu lassen, ohne die gesammte Formel berechnen zu müssen.

Anmerkungen zur Konstruktion

- Mit Hilfe von Parsergeneratoren, z.B. yacc/bison für C(++) oder ANT für Java, kann man Formelparser für Formeln in Negationsnormalform leicht programmieren.
- Man gibt eine Grammatik an, welche Formeln in Negationsnormalform erkennen kann. Mit wenig Aufwand kann man dann ein Programm schreiben, welches Formeln in die Graphenstruktur überführen.

Beispiel (yacc/bison)

Die Grammatik sieht folgendermaßen aus:

```
formula: literal { graph->createNode($1); }
| formula AND formula { graph->andOperation(); }
| formula OR formula { graph->orOperation(); }
| LEFT_BRACE formula RIGHT_BRACE { }
```

Es ist nur noch nötig einen Lexer zu schreiben und die Graphenstruktur zu implementieren. Man gibt noch an, dass AND eine höhrere Präzendenz als OR hat, sodass “und” stärker bindet als “oder” wie es in der Aussagenlogik ist. Mit Hilfe des Parsergenerators wird selbstständig eine Funktion erzeugt, welche eine Formel effektiv in die Graphenstruktur parsen kann.