Algorithmen für OBDD's

1. Reduziere
2. Boole'sche Operationen

1. Reduziere

siehe auch M.Huth und M.Ryan: Logic in Computer Science - Modelling and Reasoning about Systems, Cambridge Univ.Press, 2000 (Chapter 6, p.334-335)

(Input: OBDD B, Output: Reduzierte Form von B)

Idee: Bottom-up (beginne mit terminalen Knoten)

n Knoten in $B \mapsto id(n)$ so dass TeilOBDDs mit Wurzel n, m dieselbe Funktion darstellen gdw. $id(n) = id(m)$

1. Reduziere

Letztes Niveau: $id(0) := \#0, id(1) := \#1$

Niveau i: Annahme: $id(n)$ definiert für alle Knoten n auf Niveau $> i$.

$$\begin{tikzcd} \text{Niveau } i & & n \arrow[dl, "0"'] \arrow[dr, "1"] & \\ \text{Niveau } i+1 & lo(n) & & hi(n) \end{tikzcd}$$

Fall 1: $id(lo(n)) = id(hi(n))$.

Dann $id(n) := id(lo(n))$ [n redundanter Test $\mapsto$ Regel 2]

Fall 2: $\exists m : n, m$ mit derselben Variable P_i markiert

und $id(lo(n)) = id(lo(m))$, $id(hi(n)) = id(hi(m))$.

Dann: $id(n) := id(m)$ [n, m Wurzel isomorpher OBDDs $\mapsto$ Rege

Fall 3: Sonst: $n :=$ nächste nicht benutzte Markierung

2. Boole'sche Operationen

siehe auch M.Huth und M.Ryan: Logic in Computer Science -
Modelling and Reasoning about Systems,
Cambridge Univ.Press, 2000 (Chapter 6, p.336-337)

Satz 1.9 *(Shannon Expansion)*

Sei F eine Boole'sche Formel und sei P eine Variable.

(SE) $$F \equiv (\neg P \ \wedge \ F[0/P]) \ \vee \ (P \ \wedge \ F[1/P])$$

2. Boole'sche Operationen

$F \mapsto B_F \qquad\qquad G \mapsto B_G$

$$F \, \mathsf{op} \, G \equiv (\neg P \;\wedge\; (F[0/P] \mathsf{op} G[0/P])\,) \vee (P \;\wedge\; (F[1/P] \mathsf{op} G[1/P])\,)$$

2. Boole'sche Operationen

$\mathrm{apply}(\mathrm{op}, \mathrm{B_F}, \mathrm{B_G})$ Konstruktion von $B_{F\mathrm{op}G}$

Seien r_F die Wurzel von B_F und r_G die Wurzel von B_G.

Fall 1. r_F, r_G Terminalknoten, mit l_F bzw. l_G markiert

Dann: $B_{F\mathrm{op}G} = B_0$ falls $l_F \mathrm{B_{op}} l_G = 0$

$B_{F\mathrm{op}G} = B_1$ falls $l_F \mathrm{B_{op}} l_G = 1$

Fall 2. r_F und r_G mit P_i markiert

Dann: $B_{F\mathrm{op}G} =$

$$\begin{array}{ccc} & P_i & \\ \swarrow^{0} & & \searrow^{1} \\ \mathrm{apply}(\mathrm{op}, \mathrm{lo}(\mathrm{r_F}), \mathrm{lo}(\mathrm{r_G})) & & \mathrm{apply}(\mathrm{op}, \mathrm{hi}(\mathrm{r_F}), \mathrm{hi}(\mathrm{r_G})) \end{array}$$

2. Boole'sche Operationen

Fall 3. r_F mit P_i markiert; r_G Terminalknoten, oder mit P_j markiert, wobei $j > i$ (d.h. kein P_i Knoten in G)

Dann: $B_{F\mathrm{op}G} =$

$$\begin{tikzcd}[column sep=small] & P_i \arrow[dl, "0"'] \arrow[dr, "1"] & \\ \mathrm{apply}(\mathrm{op}, \mathrm{lo}(\mathrm{r_F}), \mathrm{B_G}) & & \mathrm{apply}(\mathrm{op}, \mathrm{hi}(\mathrm{r_F}), \mathrm{B_G}) \end{tikzcd}$$

Fall 4. r_G mit P_i markiert; r_F Terminalknoten, oder mit P_j markiert, wobei $j > i$ (d.h. kein P_i Knoten in F.)

Dann: $B_{F\mathrm{op}G} =$

$$\begin{tikzcd}[column sep=small] & P_i \arrow[dl, "0"'] \arrow[dr, "1"] & \\ \mathrm{apply}(\mathrm{op}, \mathrm{B_F}, \mathrm{lo}(\mathrm{r_G})) & & \mathrm{apply}(\mathrm{op}, \mathrm{B_F}, \mathrm{hi}(\mathrm{r_G})) \end{tikzcd}$$

Teil 1: Aussagenlogik

1.1 Syntax

1.2 Semantik

1.3 Modelle, Gültigkeit, Erfüllbarkeit

1.4 Strukturelle Induktion, Substitutionslemma

1.5 Boole'sche Funktionen

1.6 Binäre Entscheidungsbäume

1.7 BDDs, OBDDs

1.8 Aussagenformeln: Normalformen

1.9 DPLL

1.8 Normalformen

Wir definieren Konjunktionen von Formeln:

$$\bigwedge_{i=1}^{0} F_i = \top$$

$$\bigwedge_{i=1}^{1} F_i = F_1$$

$$\bigwedge_{i=1}^{n+1} F_i = \bigwedge_{i=1}^{n} F_i \wedge F_{n+1}$$

und Disjunktionen von Formeln:

$$\bigvee_{i=1}^{0} F_i = \bot$$

$$\bigvee_{i=1}^{1} F_i = F_1$$

$$\bigvee_{i=1}^{n+1} F_i = \bigvee_{i=1}^{n} F_i \vee F_{n+1}$$

Literale und Klauseln

Ein Literal ist entweder eine Aussagenvariable P, oder die Negation einer Aussagenvariablen $\neg P$.

Eine Klausel ist eine Disjunktion von Literalen.

(Leere Disjunktionen sind auch Klausel)

CNF und DNF

Eine Formel F ist in konjunktiver Normalform (CNF), falls F eine Konjunktion von Disjunktionen von Literalen ist (d.h. eine Konjunktion von Klauseln).

Eine Formel F ist in disjunktiver Normalform (DNF), falls F eine Disjunktion von Konjunktionen von Literalen ist.

Warnung: verschiedene Definitionen in der Literatur:

sind komplementäre Literale erlaubt?
sind wiederholte Literalen erlaubt?
sind leere Disjunktionen/Konjunktionen erlaubt?

CNF und DNF

Gültigkeitstests für CNF-Formeln oder Unerfüllbarkeittests für DNF-Formeln sind leicht:

Eine CNF Formel F ist gültig, genau dann, wenn jede Disjunktion in F zwei komplementären Literalen enthält.

Eine DNF Formel F ist unerfüllbar, genau dann, wenn jede Konjunktion in F zwei komplementären Literalen enthält.

Unerfüllbarkeittests für CNF-Formeln und Gültigkeitstests für DNF-Formeln sind schwer.

co-NP vollständig

Konversion zu CNF/DNF

Satz 1.10 *Jede Formel F ist mit einer Formel in CNF äquivalent.*

Jede Formel F ist mit einer Formel in DNF äquivalent.

Beweis (CNF)

Wende die folgenden Regeln, so lange wie möglich an:

1. Äquivalenzeliminierung

$$(F \leftrightarrow G) \Rightarrow_K (F \rightarrow G) \wedge (G \rightarrow F)$$

Konversion zu CNF/DNF

2. Implikationseliminierung

$$(F \rightarrow G) \Rightarrow_K (\neg F) \vee G$$

3. Negation-nach-innen

$$(\neg(F \vee G)) \Rightarrow_K (\neg F) \wedge (\neg G)$$

4. Eliminierung doppelter Negationen

$$\neg\neg F \Rightarrow_K F$$

Konversion zu CNF/DNF

5. Disjunktionen-nach-innen

$$(F \wedge G) \vee H \Rightarrow_K (F \vee H) \wedge (G \vee H)$$

6. $\bot$- und $\top$-Eliminierung

$$\begin{aligned}
(F \wedge \top) &\Rightarrow_K F \\
(F \wedge \bot) &\Rightarrow_K \bot \\
(F \vee \top) &\Rightarrow_K \top \\
(F \vee \bot) &\Rightarrow_K F \\
\neg \bot &\Rightarrow_K \top \\
\neg \top &\Rightarrow_K \bot
\end{aligned}$$

Konversion zu CNF/DNF

Terminierung: Beweis einfach für meisten Schritte
(Schritt 3 und 5 etwas komplizierter)

Korrektheit: Output: CNF-Formel, äquivalent zu der ursprünglichen Formel F

Komplexität: die Konversion zu CNF kann eine Formel erzeugen, die exponentiell in der Länge der ursprünglichen Formel ist.

Konversion zu DNF ähnlich

(aber "Konjunktionen-nach-innen" in Schritt 5).

Erfüllbarkeit erhaltende Transformationen

nicht immer praktisch:

> “Finde eine Formel G in CNF so dass $\models F \leftrightarrow G$”

einfacheres Problem:

> “Finde eine Formel G in CNF so dass $F \models \perp$ gdw. $G \models \perp$”

effiziente Transformation existiert.

Erfüllbarkeiterhaltende Transformationen

Idee: Eine Formel $F[F']$ ist erfüllbar gdw. $F[P] \wedge (F' \leftrightarrow P)$ erfüllbar (wobei P neue Aussagenvariable)

Rekursive Anwendung dieser Regel, für alle Teilformeln der Formel F (lineare Anzahl neuer Aussagenvariablen und Definitionen)

Konversion to CNF: keine exponentielle Explosion (für jede Formel $F' \leftrightarrow P$ maximal eine Anwendung des Distributivitäts-gesetzes)

Weitere Optimierungen

F enthält kein $\longrightarrow$ oder $\leftrightarrow$

Positive Polarität einer Teilformel F' in F:

- F' erscheint in F unter einer geraden Anzahl von Negationen

Negative Polarität einer Teilformel F' in F:

- F' erscheint in F unter einer ungeraden Anzahl von Negationen

Weitere Optimierungen

Satz 1.11 *Sei $F[F']$ eine Formel, die kein $\rightarrow$ oder $\leftrightarrow$ enthält.*

Falls F' positive Polarität in F hat,

so ist $F[F']$ erfüllbar gdw. $F[P] \wedge (P \rightarrow F')$ erfüllbar

Falls F' negative Polarität in F hat,

so ist $F[F']$ erfüllbar gdw. $F[P] \wedge (F' \rightarrow P)$ erfüllbar

Beweis: Übung