Logik, Semantik und Verifikation SS 2002: Musterlösung zum 3. Übungsblatt

Prof. Dr. Gert Smolka, Dipl.-Inform. Tim Priesnitz

Aufgabe 3.1: (2) Sei $A = X$ und $B = 0$, wobei $X, X_0 \in \mathit{Var}$. Nun gilt:

$$\begin{aligned}
& \mathcal{M}(A \Leftrightarrow B) \\
= \; & \mathcal{M}(A \Rightarrow B \wedge B \Rightarrow A) \\
= \; & \mathcal{M}(X \Rightarrow 0 \wedge 0 \Rightarrow X) \\
= \; & ((\Sigma - \mathcal{M}(X)) \cup \emptyset) \cap ((\Sigma - \emptyset) \cup \mathcal{M}(X)) \\
= \; & \{\sigma \in \Sigma \mid \sigma X = 0\}
\end{aligned}$$

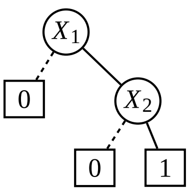
Aufgabe 3.2: Primbäume (4)

(a) Nach Proposition 4.4.9 aus dem Skript gilt: A ist gültig $\Leftrightarrow$ $A \mathrel{\models\!\!\!|} 1$. Es gibt somit genau einen zu A äquivalenten Primbaum (nämlich 1) und unendlich viele Entscheidungsbäume (beispielsweise 1 oder $1X_01$).

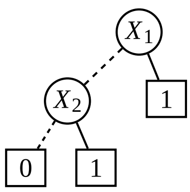
(b) Nach Proposition 4.4.9 aus dem Skript gilt: A ist unerfüllbar $\Leftrightarrow$ $A \mathrel{\models\!\!\!|} 0$. Es gibt somit genau einen zu A äquivalenten Primbaum (nämlich 0) und unendlich viele Entscheidungsbäume (beispielsweise 0 oder $0X_0(1X_00)$; beachte, dass der zweite Entscheidungsbaum nicht geordnet ist).

Aufgabe 3.3: Reduzierte OBDDs (6) Wir zeichnen aus Gründen der Übersicht alle OBDDs mit mehr als eine 0 und 1.

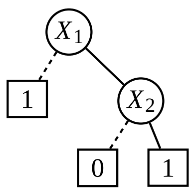
$X_1 \wedge X_2$



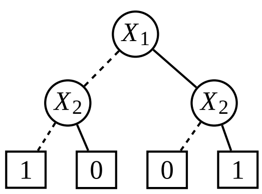
$X_1 \vee X_2$



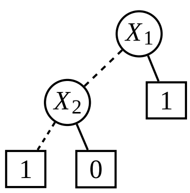
$X_1 \Rightarrow X_2$



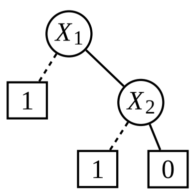
$X_1 \Leftrightarrow X_2$



$X_2 \Rightarrow X_1$

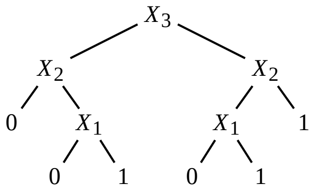


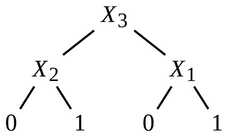
$\neg(X_1 \wedge X_2)$



Aufgabe 3.4: OBDDs und Formeln (8)

(a)





(b) Für beide OBDDs ist die Menge der signifikanten Variablen $\{X_1, X_2, X_3\}$.

(c)

$$\{\{X_1, X_2\}, \{X_1, \neg X_2, X_3\}, \{\neg X_1, X_2, X_3\}\}$$

und

$$\{\{X_1, X_2\}, \{X_1, \neg X_2, X_3\}, \{\neg X_1, X_2, \neg X_3\}\}$$

(d) Ja, denn alle Teilbäume mit gleicher Wurzel haben unterschiedliche Kinder.

Aufgabe 3.5: Reduktion von Entscheidungsbäumen (10)

$$\begin{aligned}
&\mathit{reduce}(0) = 0\\
&\mathit{reduce}(1) = 1\\
&\mathit{reduce}(t_0 x t_1) = \mathit{red}(\mathit{reduce}(t_0), x, \mathit{reduce}(t_1))
\end{aligned}$$

Zu zeigen ist:

(a) $0 = T(\mathcal{T}(0))$ Folgt nach Def. von T und $\mathcal{T}$.

(b) $1 = T(\mathcal{T}(1))$ Folgt nach Def. von T und $\mathcal{T}$.

(c) $T(\mathcal{T}(t_0 X t_1)) = \mathit{red}(\mathit{reduce}(t_0), X, \mathit{reduce}(t_1))$ Wir zeigen

$$\mathcal{T}(T(\mathcal{T}(t_0 X t_1))) = \mathcal{T}(\mathit{red}(\mathit{reduce}(t_0), X, \mathit{reduce}(t_1)))$$

Die linke Seite ergibt:

$$\begin{aligned}
& \mathcal{T}(T(\mathcal{T}(t_0 X t_1))) & \\
= \; & \mathcal{T}(t_0 X t_1) & \text{nach Lemma 4.8.9}\\
= \; & (D_{X,0} \cap \mathcal{T}(t_0)) \cup (D_{X,1} \cap \mathcal{T}(t_1)) & \text{nach Def. von } \mathcal{T}
\end{aligned}$$

Die rechte Seite ergibt:

$$\begin{aligned}
& \mathcal{T}(\mathit{red}(\mathit{reduce}(t_0), X, \mathit{reduce}(t_1))) & \\
= \; & \mathcal{T}(T(\mathcal{T}(\mathit{reduce}(t_0) X \mathit{reduce}(t_1)))) & \text{nach Def. von } \mathit{red}\\
= \; & \mathcal{T}(\mathit{reduce}(t_0) X \mathit{reduce}(t_1)) & \text{nach Lemma 4.8.9}\\
= \; & \mathcal{T}(T(\mathcal{T}(t_0)) X T(\mathcal{T}(t_1))) & \text{nach Def. von } \mathit{reduce}\\
= \; & (D_{X,0} \cap \mathcal{T}(T(\mathcal{T}(t_0)))) \cup (D_{X,1} \cap \mathcal{T}(T(\mathcal{T}(t_1)))) & \text{nach Def. von } \mathcal{T}\\
= \; & (D_{X,0} \cap \mathcal{T}(t_0)) \cup (D_{X,1} \cap \mathcal{T}(t_1)) & \text{nach Lemma 4.8.9}
\end{aligned}$$

Aufgabe 3.6: Primbaumübersetzer (20)

(a)

```
type     var = int
datatype dt  = F | T | D of dt * var * dt

fun var x = D(F,x,T)

fun neg F = T
  | neg T = F
  | neg (D(t0,x,t1)) = D(neg t0, x, neg t1)

fun red (e as (t0,_,t1)) = if t0=t1 then t0 else D e

fun And (F,_) = F
  | And (_,F) = F
  | And (T,t) = t
  | And (t,T) = t
  | And (t0 as D(t00,x,t01), t1 as D(t10,y,t11)) =
    if t0=t1 then t0
    else if x<y then red(And(t00,t1), x, And(t01,t1))
        else if x=y then red(And(t00,t10), x, And(t01,t11))
             else red(And(t0,t10), y, And(t0,t11))
```

(b)

```
fun or   (t1,t2) = neg(And(neg t1, neg t2))
fun impl (t1,t2) = neg(And(t1, neg t2))
```

(c)

```
(* Dietregeln *)

val b = var 1
val e = var 2
val f = var 3

val r1 = impl(neg b, f)
val r2 = impl(And(f,b), neg e)
val r3 = impl(or(e, neg b), neg f)
val d  = And(r1, And(r2,r3))


(* Äquivalenz -(x*y + -x*z) = x*-y + -x*-z *)

val x = var 1
val y = var 2
val z = var 3

val left  = neg(or(And(x,y), And(neg x,z)))
val right = or(And(x,neg y), And(neg x,neg z))

(* -(x*y) = -x + -y *)

val deMorgan = equi(neg(And(x,y)), or(neg x, neg y))

(* -d*d00 + d*d01 + -d*d10 + d*d11 = -d*d00*d10 + d*d01*d11 *)

val d   = var 1
val d00 = var 2
val d01 = var 3
val d10 = var 4
val d11 = var 5

val left = And(or(And(neg d, d00), And(d, d01)),
               or(And(neg d, d10), And(d, d11)))

val right = or(And(neg d, And(d00, d10)),
               And(    d, And(d01, d11)))
```

(d)

```
fun reduce F = F
  | reduce T = T
  | reduce (D(t0,x,t1)) = red(reduce t0, x, reduce t1)
```

(e)

```
datatype for =
    FF
  | TT
  | V    of int
  | NEG  of for
  | AND  of for * for
  | OR   of for * for
  | IMPL of for * for
  | EQUI of for * for

fun com FF = F
  | com TT = T
  | com (V x) = var x
  | com (NEG a) = neg (com a)
  | com (AND(a,b)) = And(com a, com b)
  | com (OR(a,b)) = or(com a, com b)
  | com (IMPL(a,b)) = impl(com a, com b)
  | com (EQUI(a,b)) = equi(com a, com b)

fun equi (t1,t2) = And(impl (t1,t2), impl (t2,t1))

val t = com (AND(IMPL(V 1,V 2), IMPL(V 2,V 1)))

val b = equi (FF, AND(V 1, NEG(V 1)))
```

(f)

```
fun equi (a,b) = com a = com b
```