3. Übungsblatt zu Logik, Semantik und Verifikation SS 2002

Prof. Dr. Gert Smolka, Dipl.-Inform. Tim Priesnitz
www.ps.uni-sb.de/courses/prog-lsv02/

Abgabe: 29. April in der Vorlesungspause

Aufgabe 3.1: (2) Geben Sie möglichst einfache Formeln A und B an, sodass

$$\emptyset \neq \mathcal{M}[\![A \Leftrightarrow B]\!] \neq \Sigma.$$

Aufgabe 3.2: Primbäume (4) Wir nehmen an, dass es unendlich viele Variablen gibt.

(a) Sei A eine gültige Formel. Wieviele zu A äquivalente Primbäume und wieviele zu A äquivalente Entscheidungsbäume gibt es? Geben Sie einen davon an.

(b) Sei A eine unerfüllbare Formel. Wieviele zu A äquivalente Primbäume und wieviele zu A äquivalente Entscheidungsbäume gibt es? Geben Sie einen davon an.

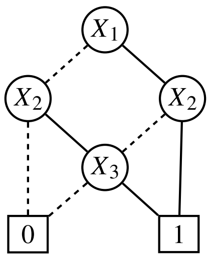
Aufgabe 3.3: Reduzierte OBDDs (6) Zeichnen Sie reduzierte OBDDs für die folgenden Formeln

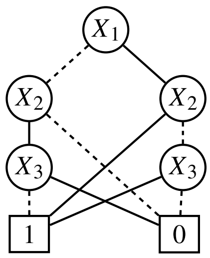
$$X_1 \wedge X_2 \qquad X_1 \vee X_2 \qquad X_1 \Rightarrow X_2$$

$$X_1 \Leftrightarrow X_2 \qquad X_2 \Rightarrow X_1 \qquad \neg(X_1 \wedge X_2)$$

Die Variablenordnung ist $X_1 < X_2$.

Aufgabe 3.4: OBDDs und Formeln (8) Gegeben sind die folgenden OBDDs:





Geben Sie für jeden OBDD an:

(a) Einen äquivalenten Primbaum für die Variablenordnung $X_3 < X_2 < X_1$.

(b) Die Menge der signifikanten Variablen.

(c) Eine äquivalente Formel in disjunktiver Normalform.

(d) Sind die gegebenen OBDDs reduziert?

Aufgabe 3.5: Reduktion von Entscheidungsbäumen (10) Betrachten Sie die Funktion

$$reduce \in DT \to PT$$
$$reduce(t) = T(\mathcal{T}[\![t]\!])$$

Geben Sie Gleichungen an, mit denen *reduce* für alle Argumente strukturell rekursiv berechnet werden kann. Verwenden Sie dabei die Funktion *red* aus dem Skript. Beweisen Sie die Gültigkeit ihrer Gleichungen.

Aufgabe 3.6: Primbaumübersetzer (20) Sie sollen die Primbaumoperationen *var*, *neg* und *and* in Standard ML realisieren. Variablen und Entscheidungsbäume realisieren wir wie folgt:

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

Dabei steht `F` für 0 und `T` für 1.

(a) Schreiben Sie Prozeduren

```
val var : int -> dt
val neg : dt -> dt
val And : dt * dt -> dt
```

die die entsprechenden Primbaumoperationen ralisieren.

(b) Realisieren Sie die Funktionen

$$or \in PT \times PT \to PT$$
$$or(t_1, t_2) = T(\mathcal{T}[\![t_1]\!] \cup \mathcal{T}[\![t_2]\!])$$

$$impl \in PT \times PT \to PT$$
$$impl(t_1, t_2) = T(-\mathcal{T}[\![t_1]\!] \cup \mathcal{T}[\![t_2]\!])$$

durch Prozeduren. Hinweis: Benutzen Sie die Prozeduren `neg` und `And`.

(c) Seien die Formeln

$$R_1 \stackrel{\text{def}}{=} \neg B \Rightarrow F$$
$$R_2 \stackrel{\text{def}}{=} F \wedge B \Rightarrow \neg E$$
$$R_3 \stackrel{\text{def}}{=} E \vee \neg B \Rightarrow \neg F$$
$$D \stackrel{\text{def}}{=} R_1 \wedge R_2 \wedge R_3$$

aus der Denksportaufgabe mit den Diätregeln gegeben. Weiter seien die Deklarationen

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

gegeben. Schreiben Sie Deklarationen, die die Bezeichner `r1`, `r2`, `r3` und `d` an die Primbäume für die Formeln R_1, R_2, R_3 und D binden.

(d) Schreiben Sie eine Prozedur

```
reduce: dt -> dt
```

die die Funktion *reduce* aus der vorhergehenden Aufgabe berechnet.

(e) Sei eine Klasse von Formeln wie folgt realisiert:

```
datatype for =
    FF                          (*  0            *)
  | TT                          (*  1            *)
  | V    of int                 (*  Variable     *)
  | NEG  of for                 (*  Negation     *)
  | AND  of for * for           (*  Konjunktion  *)
  | OR   of for * for           (*  Disjunktion  *)
  | IMPL of for * for           (*  Implikation  *)
  | EQUI of for * for           (*  Äquivalenz   *)
```

Schreiben Sie eine Prozedur

```
val com : for -> dt
```

die Formeln in Primbäume übersetzt.

(f) Schreiben Sie eine Prozedur

```
val equiv : for * for -> bool
```

die testet, ob zwei Formeln äquivalent sind.