4. Übungsblatt zu Einführung in die Computationale Logik, SS 2003

Prof. Dr. Gert Smolka, Marco Kuhlmann, MSc
`http://www.ps.uni-sb.de/courses/cl-ss03/`

Lesen Sie im Skript: Text über Induktionsbeweise und Kapitel 4 bis einschließlich Abschnit 4.11

Aufgabe 4.1 Seien X, Y, Z Variablen mit $X < Y < Z$ und sei A die Formel $(X \Leftrightarrow Y) \Leftrightarrow Z$.

(a) Geben Sie den Primbaum für A an.

(b) Geben Sie den Primbaum für $\neg A$ an.

(c) Geben Sie den Primbaum für $\widehat{A}$ an.

(d) Geben Sie den Primbaum für $A \wedge A$ an.

(e) Geben Sie den Primbaum für $A \vee \neg A$ an.

(f) Geben Sie den Primbaum für $A \Leftrightarrow \widehat{A}$ an.

Aufgabe 4.2 Sei $Var = \{X\}$ und $A \in For$ mit $A \mathrel{|\!\!\models\!\!|} 0$.

(a) Geben Sie drei verschiedene Entscheidungsbäume an, die zu A äquivalent sind.

(b) Wieviele Entscheidungsbäume gibt es, die zu A äquivalent sind?

(c) Geben Sie alle geordneten Entscheidungsbäume an, die zu A äquivalent sind.

(d) Geben Sie drei reduzierte Entscheidungsbäume an, die zu A äquivalent sind.

Aufgabe 4.3 Seien X, Y Variablen mit $X < Y$. Geben Sie die Primbäume für die folgenden Formeln an:

$$\begin{array}{ccccc} X & X \wedge Y & X \Rightarrow Y & X - Y & X \mathbin{\bar{\vee}} Y \\ \neg X & X \vee Y & X \Leftrightarrow Y & X \mathbin{\bar{\wedge}} Y & X \mathrel{\bar{\Leftrightarrow}} Y \end{array}$$

Aufgabe 4.4 Was ist an dem folgenden Induktionsbeweis verkehrt? Geben Sie zunächst die Induktionsannahme an.

Behauptung: $\forall n \in \mathbb{N}\colon 2^n = 1$.

Beweis: Durch Induktion über $n \in \mathbb{N}$. Wir unterscheiden zwei Fälle.

Sei $n = 0$. Dann $2^n = 2^0 = 1$.

Sei $n > 0$. Aus der Induktionsannahme folgt $2^{n-1} = 1$ und $2^{n-2} = 1$. Also

$$2^n = \frac{2^{n-1} \cdot 2^{n-1}}{2^{n-2}} = \frac{1 \cdot 1}{1} = 1.$$

Aufgabe 4.5 Wir stellen Entscheidungsbäume in SML gemäß der folgenden Typdeklarationen dar:

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

Dabei steht `F` für 0 und `T` für 1.

(a) Schreiben Sie eine Prozedur `neg:dt→dt`, die zu einem Entscheidungsbaum A einen zu $\neg A$ äquivalenten Entscheidungsbaum liefert.

(b) Schreiben Sie eine Prozedur `dual:dt→dt`, die zu einem Entscheidungsbaum A einen zu $\widehat{A}$ äquivalenten Entscheidungsbaum liefert.

(c) Schreiben Sie eine Prozedur `reduce:dt→dt`, die zu einem Entscheidungsbaum A einen zu A äquivalenten reduzierten Entscheidungsbaum liefert.

Die Prozeduren sollen zu reduzierten Bäumen reduzierte Bäume und zu geordneten Bäumen geordnete Bäume liefern.

Aufgabe 4.6 Wir stellen Entscheidungsbäume in SML gemäß der folgenden Typdeklarationen dar:

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

Dabei steht `F` für 0 und `T` für 1. Die Variablen seien so wie die Werte von `int` geordnet.

(a) Schreiben Sie eine Prozedur `var:var→dt`, die zu einer Variable X den zur Formel X äquivalenten Primbaum liefert.

(b) Schreiben Sie eine Prozedur `neg:dt→dt`, die zu einem Primbaum A den zu $\neg A$ äquivalenten Primbaum liefert.

(c) Schreiben Sie eine Prozedur `And:dt*dt→dt`, die zu zwei Primbäumen A, B den zu $A \wedge B$ äquivalenten Primbaum liefert.

(d) Schreiben Sie eine Prozedur `or:dt*dt→dt`, die zu zwei Primbäumen A, B den zu $A \vee B$ äquivalenten Primbaum liefert. Benutzen Sie dafür die Prozeduren `neg` und `And`.

(e) Schreiben Sie eine Prozedur `simplify:dt→dt`, die zu einem Entscheidungsbaum A den zu A äquivalenten Primbaum liefert. Benutzen Sie dafür die bereits geschriebenen Prozeduren.

(f) Wir stellen Formeln gemäß der folgenden Typdeklaration dar:

```
datatype for =
    FF                         (*  0            *)
  | TT                         (*  1            *)
  | V    of var                (*  Variable     *)
  | NEG  of for                (*  Negation     *)
  | AND  of for * for          (*  Konjunktion  *)
  | OR   of for * for          (*  Disjunktion  *)
  | EQUI of for * for          (*  Äquivalenz   *)
  | NAND of for * for          (*  Nand         *)
  | NOR  of for * for          (*  Nor          *)
  | XOR  of for * for          (*  Xor          *)
```

Schreiben Sie eine Prozedur `com:for→dt`, die zu einer Formel A den zu A äquivalenten Primbaum liefert.

(g) Schreiben Sie eine Prozedur `equiv:for*for→bool`, die testet, ob zwei Formeln äquivalent sind.

Aufgabe 4.7 Sei X_0 eine Variable und sei $For' = \{ A \in For \mid VV(A) \subseteq \{X_0\} \}$. Geben Sie eine möglichst kleine kanonische Formelmenge an, in die sich For' übersetzen lässt.

Aufgabe 4.8 Wir definieren die Formelmenge $For_1 \subseteq For$ rekursiv wie folgt:

(a) Wenn X eine Variable ist, dann sind die Formeln X und $\neg X$ in For_1.

(b) Wenn A und B in For_1 sind, dann sind die Formeln $A \wedge B$ und $A \vee B$ in For_1.

Geben Sie Vereinfachungsregeln an, mithilfe derer sich alle Formeln aus *For* in äquivalente Formeln aus For_1 übersetzen lassen. Nehmen Sie dabei an, dass X_0 eine Variable ist. (Hinweis: Es mag sich lohnen, einen Blick in Abschnitt 4.8 des Skriptes zu werfen.)