Ludwig-Maximilians-Universität München
Institut für Informatik
Prof. Dr. Peer Kröger
Janina Bleicher, Florian Richter

München, 27.10.2016

Einführung in die Programmierung

WS 2016/17

Übungsblatt 2: Boolesche Algebra, Vollständige Induktion

Besprechung: 07./09./11.11.2016

Ende der Abgabefrist: Freitag, 4.11.2016 14:00 Uhr.

Aufgabe 2-1 *Boolsche Operatoren* **2+2+2 Punkte**

Aus der Vorlesung kennen Sie die boolschen Operatoren $\wedge : \mathbb{B}^2 \to \mathbb{B}$, $\vee : \mathbb{B}^2 \to \mathbb{B}$ sowie $\neg : \mathbb{B} \to \mathbb{B}$. Mittels funktionaler Komposition lassen sich weitere Operatoren definieren, zum Beispiel gilt:

$$\Longrightarrow : \mathbb{B}^2 \to \mathbb{B}, (x, y) \mapsto \neg x \vee y$$

Außerdem lassen sich dadurch Operatoren höherer Stelligkeit definieren, z.B. Negation der ersten und zweiten Komponente:

$$\neg_1 : \mathbb{B}^2 \to \mathbb{B}, (x, y) \mapsto \neg x$$

$$\neg_2 : \mathbb{B}^2 \to \mathbb{B}, (x, y) \mapsto \neg y$$

Sei nun $\mathfrak{B}_2 = \{f \in \mathbb{B}^2 \to \mathbb{B}\}$ die Menge aller 2-stelligen totalen boolschen Funktionen.

(a) Benutzen Sie die Junktorenmenge $\{\neg_1, \wedge\} \subset \mathfrak{B}_2$ und konstruieren Sie daraus $\vee \in \mathfrak{B}_2$ durch Komposition.
Hinweis: DeMorgansche Regeln helfen.

(b) $\uparrow$ (NAND) ist wie folgt definiert:

$$x \uparrow y = \neg(x \wedge y)$$

Zeigen Sie, dass sich aus $\{\uparrow\}$ die Junktorenmenge $\{\neg_1, \wedge, \vee\}$ konstruieren lässt.
Hinweis: Zeigen Sie die Herleitungen in der angegebenen Reihenfolge $\neg_1, \wedge, \vee$ und nutzen Sie Ergebnisse aus der vorherigen Aufgabe.

(c) Geben Sie die Mächtigkeit von $\mathfrak{B}_2$ an. Wieviele 3-stellige boolsche Funktionen $|\mathfrak{B}_3|$ gibt es?
Hinweis: Wieviele Wahrheitstabellen kann es für zwei Inputs x, y geben?

Aufgabe 2-2 *Vollständige Induktion* **2+2 Punkte**

(a) Zeigen Sie mittels vollst. Induktion: $1 + 2 + 4 + 8 + \ldots + 2^n = 2^{n+1} - 1$

(b) Sei a eine Folge, die folgendermaßen rekursiv definiert ist:

$$a_1 = 2$$

$$a_{n+1} = 2 - \frac{1}{a_n}$$

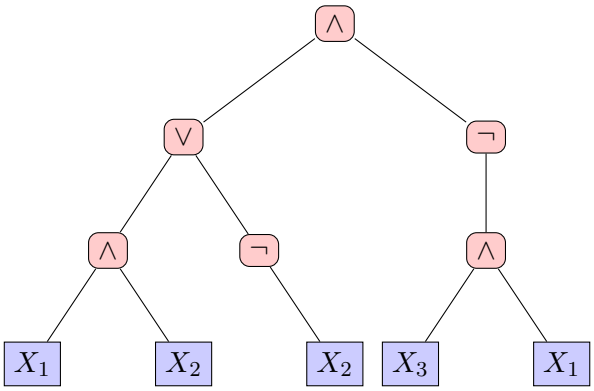
Zeigen Sie mittels vollst. Induktion: $a_n = \frac{n+1}{n}$.

Aufgabe 2-3 *Bäume* **1+2+2 Punkte**

Sei $B = \{\neg, \wedge, \vee\} \subset \mathfrak{B}$. Außerdem sei $\Sigma = \{X_1, \dots X_n\}$ eine endliche Menge von logischen Literalen. Diese können einen beliebigen Wahrheitswert aus $\{\text{True}, \text{False}\}$ annehmen. Wir definieren darüber die Menge der Wahrheitsbäume T_B induktiv:

$$\begin{aligned} \Sigma &\subseteq T_B \\ \neg(t) &\in T_B \text{ für } t \in T_B \\ \wedge(t_1, t_2) &\in T_B \text{ für } t_1, t_2 \in T_B : \\ \vee(t_1, t_2) &\in T_B \text{ für } t_1, t_2 \in T_B : \end{aligned}$$

Damit sind Blattknoten stets Literale und innere Knoten logische Junktoren. Als Beispiel soll folgende Darstellung dienen:



Der gezeigte Baum kann sequentiell dargestellt werden als

$$t' = \wedge(\vee(\wedge(X_1, X_2), \neg(X_2)), \neg(\wedge(X_3, X_1)))$$

Wir können nun eine rekursive Funktion definieren, die die induktive Konstruktion eines Baumes nutzt, um seinen Wahrheitswert zu bestimmen:

$$\begin{aligned} \text{evaluate} &: T_B \to \mathbb{B} \\ \text{evaluate}(X \in \Sigma) &= X \\ \text{evaluate}(\neg(t)) &= \neg\text{evaluate}(t) \\ \text{evaluate}(\wedge(t_1, t_2)) &= \text{evaluate}(t_1) \wedge \text{evaluate}(t_2) \\ \text{evaluate}(\vee(t_1, t_2)) &= \text{evaluate}(t_1) \vee \text{evaluate}(t_2) \end{aligned}$$

(a) Ist die Funktion evaluate(t) injektiv/surjektiv/bijektiv?

(b) Definieren Sie eine rekursive Funktion, die die Höhe eines Baumes zurückgibt:

$$\text{height} : T_B \to \mathbb{N}$$

Zur Vereinheitlichung: height$(X \in \Sigma) = 0$

(c) Eine Formel ist in Negationsnormalform (NNF), wenn Negationen nur noch direkt vor den atomaren Aussagen auftauchen. In unserem Fall bedeutet dies, dass Negationen nur unmittelbar vor den Blattknoten auftreten dürfen. Der oben gezeigte Baum t' erfüllt dies nicht. Es gilt aber

$$\text{NNF}(t') = \wedge(\vee(\wedge(X_1, X_2), \neg(X_2)), \vee(\neg(X_3), \neg(X_1))).$$

Definieren Sie nun eine rekursive Funktion NNF : $T_B \to T_B$, die einen Baum so transformiert, dass das Resultat in NNF vorliegt.

Aufgabe 2-4 *Java/Benutzerinteraktion* **0+2+3+0 Punkte**

In Aufgabe 0-3 wurde ein Programm kompiliert, das eine feste Ausgabe besaß. Die meisten Programme nutzen aber unterschiedliche Eingaben, um darauf zu reagieren und spezifische Ausgaben zu erzeugen. Dazu benötigen wir eine Benutzereingabe. Es gibt einige Möglichkeiten, dies in Java zu realisieren. Vorerst nutzen wir die Scannerklasse, die einige Eingabemethoden zur Verfügung stellt.

(a) Erstellen Sie wie in Aufgabe 0-3 eine .java-Datei mit dem Namen `Eingabe.java`. Geben Sie dieser Datei vorerst das Hello-World-Programm als Inhalt. Kompilieren Sie das Programm. Folgende Fehlermeldung wird erscheinen:

```
Eingabe.java:3: error: class HelloWorld is public,
should be declared in a file named HelloWorld.java
public class HelloWorld {
       ^
1 error
```

Java-Dateien enthalten immer genau eine öffentliche Klasse (public class). Ändern Sie daher den Namen der Klasse von `HelloWorld` zu `Eingabe`. Ändern Sie außerdem die Ausgabe von `"Hello World"` zu `"Wie alt bist du: "`.

(b) Nach der Ausgabe soll eine ganze Zahl eingelesen werden. Dazu importieren Sie mittels `import java.util.Scanner;` am Dateianfang eine Klasse, die Ihnen dabei hilft, Texteingaben zu verarbeiten. Danach nutzen Sie `Scanner sc = new Scanner(System.in);`, um dem Programm zu erklären, dass es einen Scanner vorbereiten soll, der Eingaben von der Eingabekonsole (`System.in`) erwartet. Dieser heißt `sc`. Anschließend soll der Scanner eine ganze Zahl des Benutzers einlesen: `int alter = sc.nextInt();`. Geben Sie auf geeignete Weise die Eingabe in der Konsole aus, z.B.: `"Dein Alter ist 13."`. Das Grundgerüst für diese Aufgabe liegt als `Eingabe.java` zum Download bereit.

(c) Nun reagieren Sie altersgerecht auf die Eingabe. Personen mit einem Alter über oder gleich 16 haben Zugriff auf Bier. Allen anderen wird dies verweigert. Dazu brauchen wir ein Konstrukt zur Abfrage von Bedingungen. Dabei gibt es im Programm eine Verzweigung, deren erster Block nur durchlaufen wird, wenn die Bedingung erfüllt ist. Ansonsten wird entweder das Konstrukt ganz übersprungen, oder der zweite Block durchgearbeitet. Im folgenden sehen Sie die Umsetzung in Java:

```
if(Bedingung) {
        //Wenn Bedingung erfuellt ist, dann tue alles in diesem Block.
}
else {
        //Wenn Bedingung nicht erfuellt ist, dann tue dies hier.
}
```

Formulieren Sie eine geeignete Bedingung zur Altersabfrage und geben Sie in beiden Anweisungsblöcken eine entsprechende Ausgabe. Beispielausgaben können wie folgt aussehen:

```
Wie alt bist du: 13
Kein Bier fuer dich.

Wie alt bist du: 28
Hier ist dein Bier.
```

(d) Für die Motivierten: Fragen Sie bei positiver Altersabfrage nach, wieviel Biere n gewünscht sind, und geben sie dann n-mal das Wort Bier aus. Falls n negativ ist, verweigern Sie die Bierausgabe. Dazu benötigen Sie zusätzlich zum `if`-Konstrukt noch eine `for`-Schleife.