Prof. Dr. A. Poetzsch-Heffter
Peter Zeller, M. Sc.

TU Kaiserslautern
Fachbereich Informatik
AG Softwaretechnik

Übungsblatt 2: Logik (SS 2017)

Abgabe: Freitag, 28. April, 15:30

Bitte geben Sie **zu dritt** ab (Ausnahmen können Sie mit Ihrem Tutor besprechen).

Die Abgabekästen stehen neben Raum 34-401.7 bei der AG Softwaretechnik. Sie können die Abgabe natürlich auch direkt in der Übung bei Ihrem Tutor abgeben. Die Abgabe kann handschriftlich oder als Ausdruck abgegeben werden, in deutscher oder englischer Sprache.

Wenden Sie sich bei Fragen zum Aufgabenblatt an Ihren Tutor oder besuchen Sie die Fragestunde (siehe Homepage).

Aufgabe 1 Strukturelle Induktion

Die *Tiefe* $t(A)$ einer aussagenlogischen Formel A ist wie folgt definiert:

- Ist A eine atomare Formel, so ist $t(A) = 0$.
- Ist $A \equiv (B * C)$ für einen binären Operator $* \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$, so gilt

$$t(A) = \max\{t(B), t(C)\} + 1.$$

- Ist $A \equiv \neg(B)$, so definieren wir $t(A) = t(B) + 1$.

Außerdem sei $|A|$ die Länge der Formel A, d.h. die Anzahl der Zeichen in A: $|p_i| = 1$, $\left|(\neg A)\right| = 3 + |A|$ und $|(A * B)| = 3 + |A| + |B|$ für $* \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$.

Beweisen Sie mit struktureller Induktion *über den Aufbau* der aussagenlogischen Formeln, dass in jeder vollständig geklammerten aussagenlogischen Formel A ...

a) $|A| \leq 5 \cdot op(A) + 1$

   Hier ist $op(A)$ wie auf dem letzten Blatt definiert als die Anzahl der Operatoren $(\neg, \vee, \wedge, \rightarrow, \leftrightarrow)$ in A.

b) $|A| \leq 4 \cdot 2^{t(A)} - 3$

Aufgabe 2 Semantik von Formeln

a) Sei ψ eine Belegung mit $\psi(p) = 1$ und $\psi(q) = \psi(r) = 0$. Berechnen Sie

$$\mathcal{B}_\psi\left(\neg(p \wedge q) \rightarrow r\right)$$

   schrittweise anhand der Definition der Semantik für die Aussagenlogik.

b) Beweisen oder widerlegen Sie die folgen Aussagen. Verwenden Sie dazu die Semantik der Aussagenlogik und Wahrheitstafeln.

   1. $q \rightarrow \left(r \rightarrow (p \vee q)\right)$ ist eine Tautologie.
   2. $q \rightarrow p \models p \rightarrow q$.
   3. $\neg p \vee \neg q \models\!\!\!\dashv \neg(p \wedge q)$.
   4. $(p \rightarrow q) \wedge (p \rightarrow \neg q)$ ist widerspruchsvoll.
   5. $\left\{\neg p \vee \neg q, p \vee \neg r, \neg q \vee \neg r\right\}$ ist erfüllbar.
   6. $(p \wedge r) \in Folg\left(\{p, \neg p\}\right)$

Aufgabe 3 Programmverifikation

Gegeben ist die folgende Java Code, welcher das Minimum von 3 Zahlen berechnen soll.

```
1  /*@
2    @ ensures \result <= x
3    @      && \result <= y
4    @      && \result <= z;
5    @*/
6  public static int min3(int x, int y, int z) {
7    if (x < y && x < z) {
8      return x;
9    } else if (y < x && y < z) {
10     return y;
11   } else {
12     return z;
13   }
14 }
```

Wir wollen nun versuchen zu zeigen, dass in Zeile 12 ein korrektes Ergebnis zurückgegeben wird. Dazu können wir ein Tool wie OpenJML verwenden (das Beispiel kann online unter http://www.rise4fun.com/OpenJMLESC/Anr ausprobiert werden). Solche Tools verwenden Logik um die Korrektheit eines Programms zu überprüfen.

Bei dieser Übung sollen Sie sich überlegen, welche logischen Schlüsse ein Verifikationstool wie OpenJML ziehen muss, um die Korrektheit des Programms zu beweisen. Um das Problem auf die Aussagenlogik zu reduzieren, verwenden wir die folgenden Variablen:

$p_{x<x}, p_{x<y}, p_{x<z},\; p_{y<x}, p_{y<y}, p_{y<z},\; p_{z<x}, p_{z<y}, p_{z<z}$

$p_{x\leq x}, p_{x\leq y}, p_{x\leq z},\; p_{y\leq x}, p_{y\leq y}, p_{y\leq z},\; p_{z\leq x}, p_{z\leq y}, p_{z\leq z}$

Unser mathematisches Wissen über Zahlen können wir dann auch in Formeln der Aussagenlogik ausdrücken:

Zum Beispiel: $p_{x\leq y} \to \neg p_{y<x}$ (Totalität) und $p_{x<y} \land p_{y<z} \to p_{x<z}$ (Transitivität).

Sei M die Menge aller Formeln, welche mathematische Eigenschaften über die Variablen beschreiben (Totalität, Transitivität, …).

a) In Zeile 12 wissen wir, dass die vorherigen Bedingungen nicht erfüllt waren. Wir wissen also $\Sigma = \left\{\neg(p_{x<y} \land p_{x<z}), \neg(p_{y<x} \land p_{y<z})\right\}$

   Zeigen Sie, dass daraus nicht folgt, dass das Ergebnis korrekt ist:

$$M \cup \Sigma \nvDash p_{z\leq x} \land p_{z\leq y} \land p_{z\leq z}$$

   Geben Sie dazu ein Beispiel mit konkreten Zahlen an, in dem die Prozedur eine falsche Rückgabe liefert.

b) Wir korrigieren nun das Programm und ändern Zeile 9 zu:

```
} else if (y < z) {
```

   Zeigen Sie, dass das Ergebnis in Zeile 12 dann korrekt ist. Welche mathematischen Eigenschaften (Formeln aus M) können benutzt werden um die Korrektheit zu zeigen?