10. 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: 17. Juni in der Vorlesungspause (HS 1 Mathe, Geb. 27!)

Aufgabe 10.1: Bindungsstruktur und Umbenennung (12) Seien:

- $A_1 \stackrel{\text{def}}{=} \exists X\ (X \leq Y \wedge \exists Y\ (Z \leq Y) \wedge Y \leq X)$
- $A_2 \stackrel{\text{def}}{=} \exists X\ \exists Y\ (X \leq Z \wedge \exists X\ (Y \leq X) \wedge \exists Y\ (X \leq Y) \wedge Y \leq Z)$
- $A_3 \stackrel{\text{def}}{=} \exists X\ (X \leq Z \wedge \exists Z\ (Z \leq Y)) \wedge \exists Z\ (X \leq Z) \wedge \exists Y\ (Z \leq X)$

(a) Machen Sie die Bindungsstruktur von A_1, A_2, A_3 durch Annotationen (Überstreichen und Indizieren von definierenden Auftreten, Indizieren von benutzenden Auftreten) explizit.

(b) Geben Sie die freien Variablen von A_1, A_2, A_3 an.

(c) Geben Sie für $i \in \{1, 2, 3\}$ Formeln A'_i mit $A'_i \sim A_i$ an, sodass in A'_i keine Variable sowohl gebunden als auch frei vorkommt ($A \sim B$ gilt genau dann, wenn A und B bis auf konsistente Umbenennung von gebundenen Variablen gleich sind; siehe Vorlesungsnotizen).

(d) Geben Sie für $i \in \{1, 2, 3\}$ Formeln A'_i mit $A'_i \sim A_i[X/Y][Z/X]$ an.

Aufgabe 10.2: Hoare-Regeln (12)

(a) Geben Sie eine Hoare-Regel für Until-Schleifen an (siehe voheriges Übungsblatt).

(b) Geben Sie eine Hoare-Regel für For-Schleifen an (siehe voheriges Übungsblatt).

(c) Viele Anfänger glauben, dass die Hoare-Regel für Zuweisungen so aussehen sollte:

$$\frac{}{\{B\}\ X := a\ \{B[a/X]\}}$$

Zeigen Sie mit einem Gegenbeispiel, dass die obige Regel nicht korrekt ist (also unter Umständen ungültige Hoare-Tripel liefert).

Aufgabe 10.3: Verifikation (10) Sei $I \in Assn$. Betrachten Sie die folgende Korrektheitsaussage:

```
{ N ≥ 1 }
P:=0; C:=1;
{ I }
while C≤N do
   P:=P+M; C:=C+1;
{ P = M * N }
```

(a) Geben Sie die Verifikationsbedingungen an.

(b) Geben Sie eine Invariante I an, für die alle Verifikationsbedingungen gelten.

Aufgabe 10.4: Verifikation (10) Sei $I \in Assn$. Betrachten Sie die folgende Korrektheitsaussage:

$\{ X \geq 0 \}$
```
Y:=0; Z:=0;
```
$\{ I \}$
```
while Z≤X-1 do
   Y:=Y+Z+Z+1; Z:=Z+1;
```
$\{ Y = X * X \}$

(a) Geben Sie die Verifikationsbedingungen an.

(b) Geben Sie eine Invariante I an, für die alle Verifikationsbedingungen gelten.

Aufgabe 10.5: Berechnen von Verifikationsbedingungen (6) Im folgenden wollen wir den Generator für Verifikationsbedingungen implementieren. Dabei sind die folgenden Datentypen vorgeben:

```
type con = int
type loc = string
datatype aexp = Con  of con
              | Loc  of loc
              | Add of aexp * aexp
              | Mul  of aexp * aexp
datatype bexp = True
              | False
              | LE  of aexp * aexp
              | Not of bexp
              | And of bexp * bexp
type     assn = bexp
datatype com  = Skip
              | Assign of loc * aexp
              | Seq    of com * com
              | If     of bexp * com * com
              | While  of assn * bexp * com
```

(a) Unsere Zusicherungen sind sehr primitiv. Programmieren Sie Prozeduren

```
val dis  : bexp -> bexp -> bexp
val imp  : bexp -> bexp -> bexp
val cond : bexp -> bexp -> bexp -> bexp
```

die Disjunktion, Implikation und Konditional mit Hilfe von Negation und Konjunktion repräsentieren. Dabei ist das Konditional $\text{cond}(b, b_1, b_2)$ definiert als $(b \wedge b_1) \vee (\neg b \wedge b_2)$.

(b) Programmieren Sie Substitution für arithmetische und Boolesche Ausdrücke:

```
val substa : aexp -> aexp -> loc -> aexp
val substb : bexp -> aexp -> loc -> bexp
```

Beispielsweise soll `substa` $a\ a'\ x$ den ausdruck $a[a'/x]$ liefern.

(c) Programmieren Sie eine Prozedur

```
val vc : assn -> com -> assn -> assn list
```

die zu einer Vorbedingung, einem Kommando, und einer Nachbedingung die Liste der Verifikationsbedingungen berechnet.