Technische Universität München **WS 2006/07**
Institut für Informatik
Prof. Manfred Broy
A. Gruler, B. Spanfelner **Musterlösung zu Blatt 12**

Übungen zu „Formale Methoden“

Aufgabe 1 Coffee Can Problem [LÖSUNG]

(a) Programm P:

s **var** Seq Beans;
$b_1,\ b_2$ **var** Beans;
a_w **var** Nat;

$a_w := anz_{white}$;
do $\big(\#(s) > 1\big)$ **then** $b_1,\ b_2,\ s := \mathsf{first}(s),\ \mathsf{snd}(s),\ \mathsf{rest}\big(\mathsf{rest}(s)\big)$;
if $(b_1 == b_2)$ **then** $s := \langle black \rangle \circ s$;
if $(b_1 == white)$ **then** $a_w := a_w - 2$ **fi**
[] $\neg(b_1 == b_2)$ **then** $s := \langle white \rangle \circ s$;
fi
od

Bemerkungen:
- Wir verwenden $\#(s)$ als Abkürzung für $\mathsf{length}(s)$ bzw. $\mathsf{snd}(s)$ als Abkürzung für $\mathsf{first}\big(\mathsf{rest}(s)\big)$.
- Die unterstrichenen Anweisungen sind nur für den späteren Korrektheitsnachweis und nicht für den eigentlich geforderten Algorithmus notwendig.

(b) Pro Schleifeniteration wird insgesamt 1 (schwarze oder weiße) Bohne aus dem Gefäß entnommen, die Anzahl der Bohnen im Gefäß (und somit die Länge der Sequenz) wird also pro Schleifeniteration um eins kleiner. Wir wählen den Terminierungsausdruck daher folgendermaßen: $E(s) = \#(s)$

Für die Terminierung ist mit Hilfe von E und einer Invariante I (siehe Teilaufgabe (d).) zu zeigen:

i) $Q \Rightarrow I$ Gilt, siehe Teilaufgabe (d).

ii) $\big(E(s) = 0) \wedge I\big) \Rightarrow \neg C_i$
$\big(\#(s) = 0) \ \wedge \ I\big) \ \Rightarrow \ \neg\big(\#(s) > 1\big)$ gilt.

iii) $\{(E(s) = n+1) \ \wedge \ C_i \ \wedge \ I\} \ \ S_i \ \ \{(E(s) = n) \ \wedge \ I\}$
Betrachte:
$\{(E(s) = \#(s)+1) \ \wedge \ (\#(s) > 1) \ \wedge \ I\}$
$b_1, b_2, s := \ldots;$ if ... fi
$\{(E(s) = \#(s)) \ \wedge \ I\}$

Zeige:

$$\Big((E(s) = \#(s)+1) \wedge (\#(s) > 1) \wedge I\Big) \ \Rightarrow \ \mathsf{wp}\Big(b_1, b_2, s := \ldots;\ \mathsf{if} \ldots \mathsf{fi},\ \ (E(s) = \#(s)) \wedge I\Big)$$

Gilt (Teilaufgabe (d) und $\#\big(\langle x\rangle \circ \mathsf{rest}(\mathsf{rest}(s))\big) \ = \ \#\big(\langle x\rangle \circ \mathsf{rest}(s)\big) + 1$).

$\overset{(i),(ii),(iii)}{\Longrightarrow}$ Terminierung von P.

(c) Nach Ende von P gilt die Aussage Q:

$$Q \ \equiv \ \Big(\big(\mathsf{odd}(anz_{white}) \ \wedge (\mathsf{length}(s) = 1)\big) \ \Rightarrow \ \big(\mathsf{first}(s) == white\big) \Big) \ \wedge \ \Big(\big(\mathsf{even}(anz_{white}) \ \wedge (\mathsf{length}(s) = 1)\big) \ \Rightarrow \ \big(\mathsf{first}(s) == black\big) \Big)$$

Ist also die Anzahl der weißen Bohnen anz_{white} zu Beginn des Algorithmus ungerade, so bleibt zum Schluss eine weiße Bohne übrig, anderfalls eine schwarze.

(d) **Invariante:** Die Anzahl der weißen Bohnen verändert sich während einer Schleifeniteration entweder gar nicht oder nimmt pro Iteration um 2 ab. D.h. je nach Grad des Anfangswertes ist die Anzahl der weißen Bohnen entweder *immer* gerade oder ungerade. Dies wird durch das folgende Prädikat I ausgedrückt:

$$I \ \equiv \ \Big(\mathsf{even}(anz_{white}) \ \Rightarrow \ \mathsf{even}(a_w) \Big) \ \wedge \ \Big(\mathsf{odd}(anz_{white}) \ \Rightarrow \ \mathsf{odd}(a_w) \Big)$$

Mit der Invariante I läßt sich die Korrektheit des Programms P zeigen:

i) I gilt vor Ausführung der **do**-Schleife, da:

$$\mathsf{wp}(a_w := anz_{white},\ I) \ \equiv \ true$$

ii) I bleibt während eines Schleifendurchlaufs gültig, es ist also zu zeigen:

$$\Big((\#(s) > 1) \ \wedge \ I\Big) \ \ \Rightarrow \ \ \mathsf{wp}\big(b_1, b_2, s := \ldots;\ \mathsf{if} \ldots \mathsf{fi},\ \ I\big)$$

Berechne dazu:

$$\begin{aligned} &\mathsf{wp}\big(\ b_1, b_2, s := \ldots;\ \mathsf{if} \ldots \mathsf{fi},\ \ I\big) && (*)\\ \equiv \ &\mathsf{wp}\Big(b_1, b_2, s := \ldots;\ ,\ \ \mathsf{wp}\big(\mathsf{if} \ldots \mathsf{fi},\ I\big)\Big)\\ \equiv \ &\mathsf{wp}\big(\mathsf{if} \ldots \mathsf{fi},\ I\big)\Big[\mathsf{first}(s) \ / \ b_1,\ \mathsf{snd}(s) \ / \ b_2,\ \mathsf{rest}\big(\mathsf{rest}(s)\big) \ / \ s\Big] \end{aligned}$$

$$
\begin{array}{ll}
\overset{(**)}{\equiv} & \Big(\big(\mathsf{first}(s) == \mathsf{snd}(s)\big) \;\Rightarrow\; \big((\mathsf{first}(s) == \langle white\rangle) \Rightarrow I\big[a_w - 2 \;/\; a_w\big]\big)\Big)\Big[black \circ s \;/\; s\Big] \;\wedge \\
& \Big(\neg\big(\mathsf{first}(s) == \mathsf{snd}(s)\big) \;\Rightarrow\; I\Big) \\
\equiv & \Big(\big(\mathsf{first}(\langle black\rangle \circ s) == \mathsf{snd}(\langle black\rangle \circ s)\big) \\
& \quad \Rightarrow\; \big((\mathsf{first}(\langle black\rangle \circ s) == \langle white\rangle) \Rightarrow I\big[a_w - 2 \;/\; a_w\big]\big)\Big) \;\wedge \\
& \Big(\neg\big(\mathsf{first}(s) == \mathsf{snd}(s)\big) \;\Rightarrow\; I\Big) \\
\equiv & \neg\big(\mathsf{first}(s) == \mathsf{snd}(s)\big) \;\Rightarrow\; I
\end{array}
$$

$$
\begin{array}{ll}
& \mathsf{wp}\big(\ \mathsf{if} \ldots \mathsf{fi},\ I\big) \qquad\qquad (**) \\
\equiv & \Big((b_1 == b_2) \;\vee\; \neg(b_1 == b_2)\Big) \;\wedge \\
& \Big(\big(b_1 == b_2\big) \;\Rightarrow\; \mathsf{wp}\big(s := \langle black\rangle \circ s;\ \mathsf{if}\ (b_1 == white)\ \mathsf{then}\ a_w := a_w - 2\ \mathsf{fi},\ \ I\big)\Big) \;\wedge \\
& \Big(\neg\big(b_1 == b_2\big) \;\Rightarrow\; \mathsf{wp}\big(s := \langle white\rangle \circ s,\ \ I\big)\Big) \\
\equiv & \mathit{true} \;\wedge \\
& \Big(\big(b_1 == b_2\big) \;\Rightarrow\; \mathsf{wp}\big(s := \langle black\rangle \circ s,\ \ \mathsf{wp}\big(\mathsf{if}\ (b_1 == white)\ \mathsf{then}\ a_w := a_w - 2\ \mathsf{fi},\ \ I\big)\big)\Big) \\
& \wedge\ \Big(\neg\big(b_1 == b_2\big) \;\Rightarrow\; I\big[\langle white\rangle \circ s \;/\; s\big]\Big) \\
\overset{(***)}{\equiv} & \Big(\big(b_1 == b_2\big) \;\Rightarrow\; \mathsf{wp}\big(s := \langle black\rangle \circ s,\ \ \big((b_1 == \langle white\rangle) \;\Rightarrow\; I\big[a_w - 2 \;/\; a_w\big]\big)\big)\Big) \;\wedge \\
& \Big(\neg\big(b_1 == b_2\big) \;\Rightarrow\; I\big[\langle white\rangle \circ s \;/\; s\big]\Big) \\
\equiv & \Big(\ \big(b_1 == b_2\big) \;\Rightarrow\; \big((b_1 == \langle white\rangle) \;\Rightarrow\; I\big[a_w - 2 \;/\; a_w\big]\big)\Big[\langle black\rangle \circ s \;/\; s\Big]\ \Big) \;\wedge \\
& \Big(\neg\big(b_1 == b_2\big) \;\Rightarrow\; I\big[\langle white\rangle \circ s \;/\; s\big]\Big)
\end{array}
$$

$$
\begin{array}{ll}
& \mathsf{wp}\big(\ \mathsf{if}\ (b_1 == white)\ \mathsf{then}\ a_w := a_w - 2\ \mathsf{fi},\ \ I\big) \qquad\qquad (***) \\
\equiv & (b_1 == \langle white\rangle) \;\wedge\; \Big(\big(b_1 == \langle white\rangle\big) \;\Rightarrow\; \mathsf{wp}\big(a_w := a_w - 2,\ \ I\big)\Big) \\
\equiv & (b_1 == \langle white\rangle) \;\Rightarrow\; \mathsf{wp}\big(a_w := a_w - 2,\ \ I\big) \\
\equiv & (b_1 == \langle white\rangle) \;\Rightarrow\; I\big[a_w - 2 \;/\; a_w\big]
\end{array}
$$

Es bleibt also insgesamt zu zeigen:

$$\Big(\big(\#(s) > 1\big) \;\wedge\; I\Big) \;\Rightarrow\; \Big(\neg\big(\mathsf{first}(s) == \mathsf{snd}(s)\big) \;\Rightarrow\; I\Big) \qquad \text{gilt.}$$

iii) I gilt nach Ausführung der Schleife und impliziert Q:

$$\Big(I \;\wedge\; \neg\big(\#(s) > 1\big)\Big) \;\Rightarrow\; Q \qquad \text{gilt.}$$

$\overset{(i),(ii),(iii)}{\Longrightarrow}$ Das Prädikat Q gilt somit nach der Schleife.

Aufgabe 2 Design by Contract [LÖSUNG]

Für die Lösung der Aufgabe werden hier nur folgende Methoden betrachtet:

- add(int index, Object element)
- add(Object o)
- clear()
- equals(Object o)
- get(int index)
- remove(int index)
- set(int index, Object element)

Des Weiteren gehen wir für die Aufgabe davon aus, dass die Liste nur Integerwerte speichern kann.

(a)

class AbstractList =
{ attributes $length$: **var** Int
$list$: **var** Seq Int
methods size = ()Nat:
Pre: true
Post: $li\acute{s}t = li\grave{s}t \wedge$ size = length
add = (index:Nat, element:Int):
Pre: $len\grave{g}th \geq 1 \wedge 0 < in\grave{d}ex \leq len\grave{g}th$
Post: $len\acute{g}th = len\grave{g}th + 1 \wedge$
$\exists$ j,k $\in$ Seq Int: $li\grave{s}t$ = j $\circ$ k $\wedge$ #j = $in\grave{d}ex$ -1 $\wedge$
$li\acute{s}t$ = j $\circ \langle$element$\rangle \circ$ k
Inv: $\forall a \in Int : a \in li\grave{s}t \Rightarrow a \in li\acute{s}t$ (Bsp. für eine Methodeninvariante)
add = (element:Int):
Pre: $len\grave{g}th \geq 0$
Post: $len\acute{g}th = len\grave{g}th + 1 \wedge li\acute{s}t = li\grave{s}t \circ \langle element \rangle$
clear =():
Pre: true
Post: $len\acute{g}th = 0 \wedge li\acute{s}t = \langle\rangle$
get: =(index:Nat):
Pre: 0 < index $\leq len\grave{g}th$
Post: $\exists i \in Nat$, $\exists k, l \in SeqInt$: (i = index $\wedge$ $li\grave{s}t$ = j $\circ$ k $\wedge$ #j=i-1) $\wedge$
get = first(k) $\wedge$ $li\acute{s}t = li\grave{s}t$
Inv: $\forall a \in Int : a \in li\grave{s}t \Rightarrow a \in li\acute{s}t$
equals = (o:AbstractList)Bool:
Pre: length $\geq 0 \wedge$ o.size()≥ 0
Post: equals = $len\grave{g}th == o.size() \wedge$
$\forall i \in Nat : i \leq len\grave{g}th \Rightarrow get(i) == o.get(i)$
Inv: $\forall a \in Int : a \in li\grave{s}t \Rightarrow a \in li\acute{s}t \wedge$
$\forall$ r $\in Nat$, r $\leq$ length: $li\grave{s}t$.get(r) = $li\acute{s}t$.get(r)

remove = (index: Nat):
Pre: $0 \leq \text{index} \leq \grave{length} \wedge \grave{length} > 0$
Post: ($\exists$ k,l,m $\in$ Seq Int: $\grave{list} = \text{k} \circ \text{l} \circ \text{m} \wedge$
$\#\text{k} = \text{index-1} \wedge \#\text{l} = 1 \wedge$
$\acute{list} = \text{k} \circ \text{m}) \wedge \acute{length} = \grave{length}$ - 1
set = (index: Nat, element:Int)
Pre: $0 \leq index \leq \grave{length}$
Post: $\exists$ k,l,m $\in$ Seq Int: $\grave{list} = \text{k} \circ \text{l} \circ \text{m} \wedge$
$\#\text{k} = \text{index-1} \wedge \#\text{l} = 1 \Rightarrow$
$\acute{list} = \text{k} \circ \text{l}' \circ \text{m} \wedge \text{l}' = \langle \text{element} \rangle$

Klasseninvariante: length = #list

(b) Die Vorbedingungen der Methoden dürfen abgeschwächt werden, die Nachbedingungen dürfen verstärkt werden.
D.h. sei P die Vorbedingung einer Methode M und Q die Nachbedingung. Um M mit einer Methode M´ überschreiben zu können so dass M´ immer für M eingesetzt werden kann muss für die Vorbedingung P´ von M´ gelten P $\Rightarrow$ P´ und für die Nachbedingung Q´ von M´ muss gelten Q´ $\Rightarrow$ Q.

Beispiele sind:

Abschwächung der Vorbedingung:
add = (index:Int, element:Int):
Pre: index $\leq$ length+1
Verstärkung der Nachbedingfung:
rnd =()Int:
Post: true
rnd =()Int:
Post: rnd=z $\wedge$ z<100