RHEINISCH-
WESTFÄLISCHE
TECHNISCHE
HOCHSCHULE
AACHEN

LEHR- UND FORSCHUNGSGEBIET INFORMATIK II
RWTH Aachen · D-52056 Aachen · GERMANY
http://www-i2.informatik.rwth-aachen.de/lufgi2.html

Prof. Dr. Jürgen Giesl
Darius Dlugosz

Automatisierte Programmverifikation

Übungsblatt 6

Alle Aufgaben werden korrigiert. Die Abgabe ist bis Dienstag, 9. Januar 2001, in der Übung, Vorlesung oder im Raum 4211 möglich. Die Aufgaben werden in der Übung am 9. Januar 2001 besprochen.

Aufgabe 1: (0.5 + 0.5 + 1 Punkt)

Das Programm P bestehe aus den folgenden Datenstrukturen und Algorithmen:

$$
\begin{array}{lcl}
\textbf{structure}\ \ \mathsf{bool} & & \\
\quad \mathsf{true} & : & \mathsf{bool} \\
\quad \mathsf{false} & : & \mathsf{bool} \\
 & & \\
\textbf{structure}\ \ \mathsf{number} & & \\
\quad \mathcal{O} & : & \mathsf{number} \\
\quad \mathsf{succ} & : & \mathsf{number} \to \mathsf{number} \\
 & & \\
\textbf{function}\ \ \mathsf{le} : \mathsf{number} \times \mathsf{number} \to \mathsf{bool} & & \\
\quad \mathsf{le}(\mathcal{O}, y) & \equiv & \mathsf{true} \\
\quad \mathsf{le}(\mathsf{succ}(x), \mathcal{O}) & \equiv & \mathsf{false} \\
\quad \mathsf{le}(\mathsf{succ}(x), \mathsf{succ}(y)) & \equiv & \mathsf{le}(x, y) \\
 & & \\
\textbf{function}\ \ \mathsf{plus} : \mathsf{number} \times \mathsf{number} \to \mathsf{number} & & \\
\quad \mathsf{plus}(\mathcal{O}, y) & \equiv & y \\
\quad \mathsf{plus}(\mathsf{succ}(x), y) & \equiv & \mathsf{succ}(\mathsf{plus}(x, y)) \\
 & & \\
\textbf{function}\ \ \mathsf{double} : \mathsf{number} \to \mathsf{number} & & \\
\quad \mathsf{double}(\mathcal{O}) & \equiv & \mathcal{O} \\
\quad \mathsf{double}(\mathsf{succ}(x)) & \equiv & \mathsf{succ}(\mathsf{succ}(\mathsf{double}(x))) \\
 & & \\
\textbf{function}\ \ \mathsf{half} : \mathsf{number} \to \mathsf{number} & & \\
\quad \mathsf{half}(\mathcal{O}) & \equiv & \mathcal{O} \\
\quad \mathsf{half}(\mathsf{succ}(\mathcal{O})) & \equiv & \mathcal{O} \\
\quad \mathsf{half}(\mathsf{succ}(\mathsf{succ}(x))) & \equiv & \mathsf{succ}(\mathsf{half}(x))
\end{array}
$$

Sei φ die folgende Aussage:

$$\forall x : \mathsf{number}\ \ \mathsf{le}(\mathsf{double}(\mathsf{half}(x)), x) \ \equiv \ \mathsf{true}$$

(a) Versuchen Sie, mit Hilfe der strukturellen Induktion zu zeigen, dass φ eine wahre Aussage über P ist. Verwenden Sie dabei symbolische Auswertung unter Hypothesen, um die Induktionsformeln zu beweisen.

(b) Seien $q, p \in \mathcal{T}(\Sigma^c)$ und sei $\succ$ eine Relation mit $q \succ p$ gdw. $q = \mathsf{succ}(\mathsf{succ}(p))$. Zeigen Sie, dass $\succ$ fundiert ist.

(c) Zeigen Sie mit Hilfe der noetherschen Induktion: $\varphi \in Th_P$. Verwenden Sie dabei die Relation $\succ$ aus Teil (b) und die symbolische Auswertung unter Hypothesen, um die Induktionsformeln zu beweisen.

Aufgabe 2: (0.5 Punkte)

Das noethersche Induktionsprinzip besagt, dass aus der Gültigkeit der entsprechenden Induktionsformeln die Gültigkeit des zu beweisenden Satzes folgt. Beweisen Sie die Umkehrung des noetherschen Induktionsprinzips, d.h., dass bei jedem gültigen Satz auch die entsprechenden Induktionsformeln gültig sind.

Aufgabe 3: (0.5 + 1 Punkt)

Das Programm P bestehe aus der Datenstruktur number und den folgenden Datenstrukturen und Algorithmen:

$$
\begin{array}{lcl}
\textbf{structure}\ \mathsf{list} & & \\
\quad \mathsf{empty} & : & \mathsf{list} \\
\quad \mathsf{add} & : & \mathsf{number} \times \mathsf{list} \to \mathsf{list} \\
 & & \\
\textbf{function}\ \mathsf{append} : \mathsf{list} \times \mathsf{list} \to \mathsf{list} & & \\
\quad \mathsf{append}(\mathsf{empty}, l) & \equiv & l \\
\quad \mathsf{append}(\mathsf{add}(x, k), l) & \equiv & \mathsf{add}(x, \mathsf{append}(k, l)) \\
 & & \\
\textbf{function}\ \mathsf{reverse} : \mathsf{list} \times \mathsf{list} \to \mathsf{list} & & \\
\quad \mathsf{reverse}(\mathsf{empty}, l) & \equiv & l \\
\quad \mathsf{reverse}(\mathsf{add}(x, k), l) & \equiv & \mathsf{reverse}(k, \mathsf{add}(x, l))
\end{array}
$$

(a) Bestimmen Sie das strukturelle Induktionsaxiom der Datenstruktur list zum Beweis einer Formel $\forall\, l : \mathsf{list}\ \ \psi$.

(b) Zeigen Sie mit Hilfe der in (a) erzeugten Induktionsformeln, dass

$$\forall\, l_1, l_2, l_3 : \mathsf{list}\ \ \mathsf{reverse}(l_1, \mathsf{reverse}(l_2, l_3)) \equiv \mathsf{reverse}(\mathsf{append}(l_2, l_1), l_3)$$

eine wahre Aussage über P ist. Verwenden Sie dabei den Algorithmus SA_P.

Aufgabe 4: (1 Punkt)

Geben Sie eine Formel $\varphi \in Th_P$ an, die nicht aus den Axiomen AX_P eines Programms P folgt, aber durch Induktion bewiesen werden kann, wobei bei der Anwendung der Hypothesen die Regel

- $\psi_1|_\pi = \sigma(\xi)$ und $\psi_2 = \psi_1[TRUE]_\pi$

aus Definition 5.2.1 (Hypothesenanwendung) nicht anwendbar ist. Dabei darf Induktion nur einmal durchgeführt werden.

Aufgabe 5: (2 Punkte)

Die in der Vorlesung eingeführte Programmiersprache erlaubt keine verschränkte Rekursion (siehe Definition 3.1.5 (4) - Erweiterung von Programmen um Algorithmen). Wir erweitern jetzt diese Programmiersprache soweit, dass verschränkte Rekursion erlaubt wird.

Also ist es zum Beispiel möglich, ein Programm P um folgende Algorithmen zu erweitern:

$$\begin{array}{lcl}
\multicolumn{3}{l}{\textbf{function}\ \ \mathsf{even} : \mathsf{number} \to \mathsf{bool}} \\
\ \mathsf{even}(\mathcal{O}) & \equiv & \mathsf{true} \\
\ \mathsf{even}(\mathsf{succ}(x)) & \equiv & \mathsf{odd}(x) \\
\\
\multicolumn{3}{l}{\textbf{function}\ \ \mathsf{odd} : \mathsf{number} \to \mathsf{bool}} \\
\ \mathsf{odd}(\mathcal{O}) & \equiv & \mathsf{false} \\
\ \mathsf{odd}(\mathsf{succ}(x)) & \equiv & \mathsf{even}(x)
\end{array}$$

Skizzieren Sie eine Methode, mit der verschränkt rekursive Algorithmen in äquivalente Algorithmen ohne verschränkte Rekursion überführt werden können. Dabei ist ein Algorithmus f eines Programms P mit der zugehörigen Signatur Σ genau dann zu einem Algorithmus f' eines Programms P' äquivalent, wenn für alle $f(t^*) \in \mathcal{T}(\Sigma)$ gilt: $eval_P(f(t^*)) = eval_{P'}(f'(t^*))$.

Frohe Weihnachten
und ein
erfolgreiches und glückliches neues Jahr!