RHEINISCH-
WESTFÄLISCHE
TECHNISCHE
HOCHSCHULE
AACHEN

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

Prof. Dr. J. Giesl
D. Dlugosz

6. Übung zu „Automatisierte Programmverifikation“, SS 03
Abgabe: Mi, 18.06.03, in der Frontalübung

Aufgabe 1 (2 Punkte)

Die *symbolische Auswertung* sei wie folgt definiert:
Sei P ein Programm und seien ψ_1 und ψ_2 quantorfreie Formeln über der zu P gehörenden Signatur. Dann gilt $\psi_1 \rightarrow_P \psi_2$ („ψ_1 wird in einem Schritt symbolisch ausgewertet zu ψ_2“) gdw. $\psi_1 \rightarrow_{AX_P} \psi_2$.

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

structure bool

true	:	bool
false	:	bool

function le : number × number → bool

$\mathsf{le}(\mathcal{O}, y)$	$\equiv$	true
$\mathsf{le}(\mathsf{succ}(x), \mathcal{O})$	$\equiv$	false
$\mathsf{le}(\mathsf{succ}(x), \mathsf{succ}(y))$	$\equiv$	$\mathsf{le}(x, y)$

function plus : number × number → number

$\mathsf{plus}(\mathcal{O}, y)$	$\equiv$	y
$\mathsf{plus}(\mathsf{succ}(x), y)$	$\equiv$	$\mathsf{succ}(\mathsf{plus}(x, y))$

function double : number → number

$\mathsf{double}(\mathcal{O})$	$\equiv$	$\mathcal{O}$
$\mathsf{double}(\mathsf{succ}(x))$	$\equiv$	$\mathsf{succ}(\mathsf{succ}(\mathsf{double}(x)))$

function half : number → number

$\mathsf{half}(\mathcal{O})$	$\equiv$	$\mathcal{O}$
$\mathsf{half}(\mathsf{succ}(\mathcal{O}))$	$\equiv$	$\mathcal{O}$
$\mathsf{half}(\mathsf{succ}(\mathsf{succ}(x)))$	$\equiv$	$\mathsf{succ}(\mathsf{half}(x))$

Berechnen Sie durch Angabe der Auswertungsfolgen die Normalform bzgl. der symbolischen Auswertung von

a) $\neg\neg\mathsf{plus}(\mathsf{succ}(x), y) \equiv \mathsf{plus}(\mathsf{succ}(x), y)$,

b) $\mathsf{le}(\mathsf{double}(\mathsf{half}(\mathcal{O})), \mathcal{O}) \equiv \mathsf{true}$,

c) $\mathsf{le}(\mathsf{double}(\mathsf{half}(\mathsf{succ}(\mathcal{O}))), \mathsf{succ}(\mathcal{O})) \equiv \mathsf{true}$,

d) $\mathsf{le}(\mathsf{plus}(\mathsf{succ}(\mathcal{O}), \mathsf{half}(\mathsf{succ}(\mathsf{succ}(\mathcal{O})))), \mathsf{succ}(\mathsf{succ}(x))) \equiv \mathsf{true} \rightarrow \mathsf{plus}(\mathsf{succ}(\mathcal{O}), x) \equiv \mathsf{succ}(\mathcal{O})$,

e) $\neg\,\mathsf{double}(\mathsf{half}(\mathcal{O})) \equiv \mathcal{O} \vee \neg\,\mathsf{half}(\mathsf{succ}(\mathcal{O})) \equiv \mathsf{double}(\mathsf{half}(\mathsf{succ}(\mathcal{O}))$.

Aufgabe 2 (0.5 Punkte)

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

structure sexpr

nil	:	sexpr
atom	:	number → sexpr
cons	:	sexpr × sexpr → sexpr

structure tree

leaf	:	tree
node	:	tree × number × tree → tree

Geben Sie die Axiome AX_{sexpr} und AX_{tree} an.

Aufgabe 3 (2 Punkte)

Sei P ein terminierendes Programm. Zeigen Sie: $AX_P \subseteq Th_P$.

Aufgabe 4 (2 Punkte)

Sei P ein terminierendes Programm.

a) Skizzieren Sie ein Verfahren, mit dem „$\varphi \in Th_P$“ für alle $\varphi \in \mathcal{F}_g(\Sigma, \varnothing)$ bewiesen oder widerlegt werden kann.

b) Eine Formel $\varphi \in \mathcal{F}_g(\Sigma, \mathcal{V})$ heißt Existenzformel gdw. $\mathcal{V}(\varphi) = \varnothing$ oder $\varphi = \exists x_1 : s_1 \ldots \exists x_n : s_n\ \psi$, ψ ist quantorfrei und $\mathcal{V}(\psi) \subseteq \{x_1, \ldots, x_n\}$. Skizzieren Sie ein Verfahren, mit dem „$\varphi \in Th_P$“ für jede Existenzformel $\varphi \in Th_P$ bewiesen werden kann. Wie verhält sich Ihr Verfahren bei Eingabe einer Existenzformel $\varphi \notin Th_P$?

Aufgabe 5 (2 Punkte)

Die in der Vorlesung eingeführte Programmiersprache erlaubt verschränkt rekursive Algorithmen. Somit ist es zum Beispiel möglich, ein Programm P um folgende Algorithmen zu erweitern:

function even : number → bool

even($\mathcal{O}$)	≡	true
even(succ(x))	≡	odd(x)

function odd : number → bool

odd($\mathcal{O}$)	≡	false
odd(succ(x))	≡	even(x)

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: $eval_P(f(t^*)) = eval_{P'}(f'(t^*))$.