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

12. Übung zu „Automatisierte Programmverifikation“, SS 03
Abgabe: Mi, 30.07.03, in der Frontalübung

Aufgabe 1 (8 Punkte)

Uns stehen die Datenstruktur number und die folgenden Datenstrukturen und Algorithmen zur Verfügung:

structure list
nil : list
cons : number × list → list

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 times : number × number → number
$\mathsf{times}(\mathcal{O}, y) \equiv \mathcal{O}$
$\mathsf{times}(\mathsf{succ}(x), y) \equiv \mathsf{plus}(y, \mathsf{times}(x, y))$

function pred : number → number
$\mathsf{pred}(\mathcal{O}) \equiv \mathcal{O}$
$\mathsf{pred}(\mathsf{succ}(x)) \equiv x$

function minus : number × number → number
$\mathsf{minus}(x, \mathcal{O}) \equiv x$
$\mathsf{minus}(x, \mathsf{succ}(y)) \equiv \mathsf{minus}(\mathsf{pred}(x), y)$

function ge : number × number → bool
$\mathsf{ge}(x, \mathcal{O}) \equiv \mathsf{true}$
$\mathsf{ge}(\mathcal{O}, \mathsf{succ}(y)) \equiv \mathsf{false}$
$\mathsf{ge}(\mathsf{succ}(x), \mathsf{succ}(y)) \equiv \mathsf{ge}(x, y)$

function if : bool × number × number → number
$\mathsf{if}(\mathsf{true}, x, y) \equiv x$
$\mathsf{if}(\mathsf{false}, x, y) \equiv y$

function subtract : number × number → number
$\mathsf{subtract}(x, \mathcal{O}) \equiv x$
$\mathsf{subtract}(\mathcal{O}, \mathsf{succ}(y)) \equiv \mathcal{O}$
$\mathsf{subtract}(\mathsf{succ}(x), \mathsf{succ}(y)) \equiv \mathsf{subtract}(x, y)$

function quot : number × number → number
$\mathsf{quot}(x, \mathcal{O}) \equiv \mathcal{O}$
$\mathsf{quot}(\mathcal{O}, \mathsf{succ}(y)) \equiv \mathcal{O}$
$\mathsf{quot}(\mathsf{succ}(x), \mathsf{succ}(y)) \equiv \mathsf{succ}(\mathsf{quot}(\mathsf{subtract}(x, y), \mathsf{succ}(y)))$

function log : number → number
$\mathsf{log}(\mathcal{O}) \equiv \mathcal{O}$
$\mathsf{log}(\mathsf{succ}(\mathcal{O})) \equiv \mathcal{O}$
$\mathsf{log}(\mathsf{succ}(\mathsf{succ}(x))) \equiv \mathsf{succ}(\mathsf{log}(\mathsf{succ}(\mathsf{quot}(x, \mathsf{succ}(\mathsf{succ}(\mathcal{O}))))))$

function f : number × number → number
$\mathsf{f}(\mathcal{O}, y) \equiv \mathsf{succ}(\mathcal{O})$
$\mathsf{f}(\mathsf{succ}(x), y) \equiv \mathsf{f}(\mathsf{f}(x, y), y)$

function g : number × number → number
$\mathsf{g}(\mathcal{O}, y) \equiv \mathcal{O}$
$\mathsf{g}(\mathsf{succ}(x), y) \equiv \mathsf{g}(\mathsf{g}(x, y), y)$

function append : list × list → list
$\mathsf{append}(\mathsf{nil}, l) \equiv l$
$\mathsf{append}(\mathsf{cons}(n, k), l) \equiv \mathsf{cons}(n, \mathsf{append}(k, l))$

function reverse : list → list
$\mathsf{reverse}(\mathsf{nil}) \equiv \mathsf{nil}$
$\mathsf{reverse}(\mathsf{cons}(n, l)) \equiv \mathsf{append}(\mathsf{reverse}(l), \mathsf{cons}(n, \mathsf{nil}))$

function shuffle : list → list
$\mathsf{shuffle}(\mathsf{nil}) \equiv \mathsf{nil}$
$\mathsf{shuffle}(\mathsf{cons}(n, l)) \equiv \mathsf{cons}(n, \mathsf{shuffle}(\mathsf{reverse}(l))))$

function even : number → bool
$\mathsf{even}(\mathcal{O}) \equiv \mathsf{true}$
$\mathsf{even}(\mathsf{succ}(x)) \equiv \mathsf{odd}(x)$

function odd : number → bool
$\mathsf{odd}(\mathcal{O}) \equiv \mathsf{false}$
$\mathsf{odd}(\mathsf{succ}(x)) \equiv \mathsf{even}(x)$

function gcd : number × number → number
$\mathsf{gcd}(x, \mathcal{O}) \equiv x$
$\mathsf{gcd}(\mathcal{O}, \mathsf{succ}(y)) \equiv \mathsf{succ}(y)$
$\mathsf{gcd}(\mathsf{succ}(x), \mathsf{succ}(y)) \equiv \mathsf{if}(\mathsf{ge}(x, y),$
$\quad \mathsf{gcd}(\mathsf{minus}(x, y), \mathsf{succ}(y)),$
$\quad \mathsf{gcd}(\mathsf{succ}(x), \mathsf{minus}(y, x)))$

Im folgenden soll die Terminierung von Programmen mit Dependency Pairs gezeigt werden. Führen Sie dabei folgende Schritte aus:

1) Bestimmen Sie alle Dependency Pairs von P.
2) Geben Sie für jedes Paar $\langle s, t\rangle \in DP(P)$ die Menge $\mathcal{U}(E_P, t)$ der verwendbaren Gleichungen an.
3) Bestimmen Sie die Ungleichungen gemäß Satz 6.4.3, die erfüllt sein müssen, damit die Terminierung von P gezeigt werden kann.
4) Geben Sie das Argument-Filterungssystem $\mathcal{AF}$ an, das Sie verwenden, um die Terminierung zu zeigen.
5) Geben Sie eine lexikographische Pfadordnung, die die die Ungleichungen aus 3) bei dem in 4) gewählten Argument-Filterungssystem erfüllt.

   Sollten Sie keine lexikographische Pfadordnung finden, versuchen Sie, geeignete Relationen $\succ$ und $\succsim$ zu finden, die die Ungleichungen aus 3) erfüllen.

Versuchen Sie, wie oben beschrieben, die Terminierung von P zu zeigen, das aus folgenden Datenstrukturen und Algorithmen besteht:

a) number, plus, times, pred, minus, ge, if, gcd
b) number, subtract, quot, log
c) number, f
d) number, g
e) number, list, append, reverse, shuffle
f) number, even, odd

Aufgabe 2 (2 Punkte)

Sei P ein Programm und sei $\succsim$ eine Quasi-Ordnung. Beweisen Sie die folgende Aussage:

Wenn $l \succsim r$ für alle Gleichungen $l \equiv r \in E_P$ gilt, die in einer Auswertung $s \Rightarrow_P t$ verwendet werden, und $\succsim$ stabil, monoton und transitiv ist, dann gilt auch $s \succsim t$.

Aufgabe 3 (2 Punkte)

Betrachten wir alternative Terminierungsverfahren mit Dependency Pairs.

a) Beweisen oder widerlegen Sie die folgende Behauptung: Wenn für alle $\langle s, t\rangle \in DP(P)$ eine Reduktionsordnung $\succ$ mit $s \succ t$ existiert, dann terminiert das Programm P.
b) Wir ersetzen in allen Paaren $\langle s, t\rangle \in DP(P)$ die Tupelsymbole durch die zugehörigen Funktionssymbole (z.B. aus $\langle \mathsf{PLUS}(\mathsf{succ}(x), y), \mathsf{PLUS}(x, y)\rangle$ entsteht das Paar $\langle \mathsf{plus}(\mathsf{succ}(x), y), \mathsf{plus}(x, y)\rangle$).

   Welche Folgen ergeben sich daraus?
   Gilt der Satz 6.3.6 immer noch?