TECHNISCHE UNIVERSITÄT MÜNCHEN
FAKULTÄT FÜR INFORMATIK

Lehrstuhl für Sprachen und Beschreibungsstrukturen
Prof. Dr. Helmut Seidl
A. Berlea, F. Forster, T. Gawlitza

SoS 2007
Übungsblatt 2
Ausgegeben am: 27. April 2007
Abgabe bis: **11. Mai 2007**

Übungen zu Einführung in die Informatik II

Aufgabe 1 (H) Verifikation (3 Punkte)

Gegeben sei folgendes MiniJava-Programm:

```
int n, i, result;

n = read();
if (n < 0)
        n = -n;

result = 0;
i = 0;
while (i!=n) {
        result = result + i;
        i = i + 1;
        result = result - i;
}
```

a) Erstellen Sie den Kontrollfluss-Graphen!

b) Zeigen Sie, dass am Stop-Knoten die Zusicherung `result` $= -$`n` stets erfüllt ist.

Aufgabe 2 (H) Verifikation effizienter Berechnung von x^N (8 Punkte)

Eine effizienten Berechnung von x^N (für $N \geq 0$) basiert auf den folgenden Gleichheiten, wobei $\{b_m b_{m-1} \ldots b_1 b_0\}_2$ die Repräsentation der Zahl N zur Basis 2 ist:

$$x^N = x^{\{b_m b_{m-1} \ldots b_1 b_0\}_2} = x^{b_m \cdot 2^m + b_{m-1} \cdot 2^{m-1} + \ldots + b_1 \cdot 2^1 + b_0 \cdot 2^0} = x^{b_m \cdot 2^m} \cdot x^{b_{m-1} \cdot 2^{m-1}} \cdot \ldots \cdot x^{b_1 \cdot 2^1} \cdot x^{b_0 \cdot 2^0}$$

Beispielsweise ist $25 = \{11001\}_2$ und dementsprechend $x^{25} = x^{2^4} \cdot x^{2^3} \cdot x$. Wir berechnen dieses Produkt, indem wir die Faktoren von rechts nach links akkumulieren. Dabei benutzen wir:

$$x^{b_i \cdot 2^i} = \begin{cases} 1 & \text{falls } b_i = 0 \\ x & \text{falls } b_i = 1 \text{ und } i = 0 \\ \left(x^{2^{i-1}}\right)^2 & \text{falls } b_i = 1 \text{ und } i > 0 \end{cases}$$

Eine Implementierung dieser Idee in Mini-Java sieht wie folgt aus:

```
int x,N;
x = read();
N = read();

if (N<0) {
    write("N has to be positive!");
}
else{
    int n,prod,tower,k;
    n=N;
    prod=1;
    tower=x;
    k=0;
    while(n!=0){
        k=k+1;
        if (n%2==1) prod=prod*tower;
        n = n/2;
        tower = tower*tower;
    }
    write(prod);
}
```

a) Erstellen Sie den Kontrollfluss-Graphen!

b) Drücken Sie die Werte von `prod`, `tower` und `n` am Ende jedes Schleifendurchlaufs in Abhängigkeit von `x`, `N` und `k` aus!

c) Bestimmen Sie geeignete Zusicherungen an jedem Programmpunkt, so dass am Ende des Programms $\texttt{prod} = \texttt{x}^{\texttt{N}}$ sichergestellt wird! Überprüfen Sie die lokale Konsistenz der Zusicherungen!

Hinweise:

- Testen Sie die Richtigkeit Ihrer Zusicherungen mit Hilfe von `assert`-Anweisungen, bevor Sie mit dem Überprüfen der lokalen Konsistenz beginnen!
- Als Hilfestellung seien Ihnen folgende Rechenregeln gegeben. Für $a \in \mathbb{N}_0$, $b, c \in \mathbb{N}$ gilt:

$$a \textbf{ div } (b \cdot c) = (a \textbf{ div } b) \textbf{ div } c \qquad (1)$$
$$a \textbf{ mod } (b \cdot c) = (a \textbf{ mod } b) + b \cdot ((a \textbf{ div } b) \textbf{ mod } c) \qquad (2)$$