Metody programowania 2016

Lista zadań nr 0

Na zajęcia 22 lutego i 19 kwietnia 2017

Lista jest wręczana studentom bezpośrednio na zajęciach. Punkty (po 2 za każde zadanie) otrzymują osoby, które rozwiążą dane zadanie przy tablicy.

Przypomnij sobie z zajęć *Wstępu do Informatyki* asercje częściowej poprawności programów. Były one podobne do tych, którymi zajmiemy się w poniższych zadaniach.

W zadaniach 1–6 wynajdź niezmienniki pętli (zależności między wartościami zmiennych programu, które pozostają zachowane podczas iteracji) i uzasadnij, że po zakończeniu działania programów wartości zmiennych spełniają podane zależności. Podaj dla jakich wartości początkowych zmiennych podane programy się zatrzymują. W zadaniach tych *odd* oznacza predykat nieparzystości, *gcd* — największy wspólny podzielnik, zaś F_n — n-ty wyraz ciągu Fibonacciego, tzn. $F_0 = F_1 = 1$ oraz $F_{n+2} = F_{n+1} + F_n$ dla $n \in \mathbb{N}$.

Reguły wnioskowania dla asercji częściowej poprawności programów rozważanych w zadaniach z bieżącej listy

$$\frac{\vdash \phi \Rightarrow \psi \quad \{\psi\}\ P\ \{\rho\}}{\{\phi\}\ P\ \{\rho\}}\ \text{(Pre)}$$

$$\frac{\{\phi\}\ P\ \{\psi\} \quad \vdash \psi \Rightarrow \rho}{\{\phi\}\ P\ \{\rho\}}\ \text{(Post)}$$

$$\frac{}{\{\phi[x/e]\}\ x := e\ \{\phi\}}\ \text{(Asgn)}$$

$$\frac{\{\phi\}\ P_1\ \{\psi\} \quad \{\psi\}\ P_2\ \{\rho\}}{\{\phi\}\ P_1\,;\,P_2\ \{\rho\}}\ \text{(Seq)}$$

$$\frac{\{\phi \wedge b\}\ P\ \{\psi\} \quad \vdash \phi \wedge \neg b \Rightarrow \psi}{\{\phi\}\ \mathbf{if}\ b\ \mathbf{then}\ P\ \mathbf{fi}\ \{\psi\}}\ \text{(If)}$$

$$\frac{\{\phi \wedge b\}\ P_1\ \{\psi\} \quad \{\phi \wedge \neg b\}\ P_2\ \{\psi\}}{\{\phi\}\ \mathbf{if}\ b\ \mathbf{then}\ P_1\ \mathbf{else}\ P_2\ \mathbf{fi}\ \{\psi\}}\ \text{(IfElse)}$$

$$\frac{\{\phi \wedge b\}\ P\ \{\phi\}}{\{\phi\}\ \mathbf{while}\ b\ \mathbf{do}\ P\ \mathbf{done}\ \{\phi \wedge \neg b\}}\ \text{(While)}$$

Zadanie 1 (2 pkt).

```
{z = i · j − x · y}
while ¬odd(x) do
    y := 2 · y
    x := x div 2
done
{z = i · j − x · y ∧ odd(x)}
```

Zadanie 2 (2 pkt).

```
{x = i ∧ y = j}
s := 0
while x ≠ 0 do
    while ¬odd(x) do
        y := 2 · y
        x := x div 2
    done
    s := s + y
    x := x − 1
done
{s = i · j}
```

Zadanie 3 (2 pkt).

```
{x = i ∧ y = j ∧ j ≥ 0}
z := 1
while y > 0 do
    if odd(y)
    then
        z := z · x
    fi
    y := y div 2
    x := x · x
done
{z = i^j}
```

Zadanie 4 (2 pkt).

```
{x = i ∧ y = j}
while x ≠ y do
    while x > y do
        x := x − y
    done
    while y > x do
        y := y − x
    done
done
{x = y ∧ x = gcd(i, j)}
```

Zadanie 5 (2 pkt).

```
{z = i ∧ i ≥ 0}
x := 1
y := 1
while z > 0 do
    y := x + y
    x := y − x
    z := z − 1
done
{x = F_i}
```

Zadanie 6 (2 pkt).

```
{x = i ∧ y = j}
while y ≠ 0 do
    z := y
    y := x mod y
    x := z
done
{x = gcd(i, j)}
```

Zadanie 7 (2 pkt). Udowodnij, że dla dowolnych wartości początkowych zmiennych k, m i n poniższy program zatrzyma się. *Wskazówka:* zdefiniuj „miarę złożoności” konfiguracji pamięci $\mu(n,m) = \langle n, m \rangle$. Zauważ, że jeśli uporządkujesz zbiór wartości tej miary leksykograficznie, to wartość μ zmniejsza się podczas każdej iteracji pętli. Przypomnij z kursu logiki fakt, że porządek leksykograficzny na parach liczb naturalnych jest dobry. Wywnioskuj stąd, że program się zatrzymuje. (Uwaga: początkowe wartości zmiennych k, m i n niekoniecznie są nieujemne!) Tak przy okazji: jaką funkcję oblicza ten program?

```
m := 2 · n
k := 0
while m > 0 do
    m := m − 1
    k := k + 1
    if m = 0
    then
        n := n − 1
        m := 2 · n
    fi
done
```

Zadanie 8 (2 pkt). Pokaż, że przedstawiony poniżej, znany z kursu logiki, algorytm znajdowania najogólniejszego unifikatora pary termów $t \stackrel{?}{=} s$ zawsze się zatrzymuje. Następnie przyjmując niezmiennik pętli:

$$\{\rho \mid \rho \text{ unifikuje } t \stackrel{?}{=} s\} = \{\theta\rho \mid \rho \text{ unifikuje } R\}$$

udowodnij, że po zakończeniu pracy algorytmu θ jest najogólniejszym unifikatorem $t \stackrel{?}{=} s$.

Algorytm unifikacji

```
R := {t ≟ s}
θ := []
while R ≠ ∅ do
    select t ≟ s from R
    case t ≟ s of
        x ≟ x, where x ∈ 𝒳:
            skip
        x ≟ t or t ≟ x, where x ∈ 𝒳 and x ≠ t:
            if x ∉ FV(t)
            then
                R := R[x/t]
                θ := θ[x/t]
            else
                return "not unifiable"
            fi
        f(t_1, ..., t_n) ≟ f(s_1, ..., s_n):
            R := R ∪ {t_1 ≟ s_1, ..., t_n ≟ s_n}
        f(t_1, ..., t_n) ≟ g(s_1, ..., s_m), where f ≠ g:
            return "not unifiable"
    esac
done
return θ
```

Zadanie 9 (2 pkt). Oto algorytm badający spełnialność formuł 2-CNF postaci $\bigwedge_{i=1}^{n}(k_i \Rightarrow l_i)$, gdzie k_i oraz l_i są literałami, tj. zmiennymi zdaniowymi, zanegowanymi zmiennymi zdaniowymi lub symbolami **t**, **f** (por. *Whitebook*).

```
T := Succ(t)
F := Pred(f)
Satisfiable := T ∩ F = ∅
while Satisfiable ∧ T ∪ F ≠ V do
    x := choose(V \ (T ∪ F))
    if Succ(x) ∩ Pred(¬x) ≠ ∅
    then
        x := ¬x
    fi
    if Succ(x) ∩ Pred(¬x) ≠ ∅
    then
        Satisfiable := false
    else
        T := T ∪ Succ(x)
        F := F ∪ Pred(¬x)
    fi
done
for x ∈ V do
    σ_0(x) := { 1, if x ∈ T
              { 0, otherwise
done
```

gdzie operacja $\neg$ przyporządkowuje zmiennej zdaniowej zmienną zanegowaną a zmiennej zanegowanej — odpowiednią zmienną zdaniową, zaś

$$\begin{aligned} V &= \{p, \neg p \mid p \in \mathrm{Var}(\phi)\} \cup \{\mathbf{t}, \mathbf{f}\}, \\ E &= \{\langle k_i, l_i\rangle, \langle \neg l_i, \neg k_i\rangle\}_{i=1}^{n}, \\ \mathrm{Succ}(x) &= \{y \in V \mid \langle x, y\rangle \in E^*\}, \\ \mathrm{Pred}(x) &= \{y \in V \mid \langle y, x\rangle \in E^*\}, \end{aligned}$$

a E^* oznacza zwrotne i przechodnie domknięcie relacji E. Wykaż, że program zawsze się zatrzymuje. Wykaż poprawność tego algorytmu, tj. udowodnij, że po jego zatrzymaniu jest prawdziwa formuła

$$(\neg \mathit{Satisfiable} \wedge \forall\sigma(\sigma(\phi) = 0)) \vee (\mathit{Satisfiable} \wedge \sigma_0(\phi) = 1).$$