

Metody programowania 2015

Lista zadań nr 0

Na zajęcia 23–26 lutego 2015

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 zajmujemy się w poniższych zadaniach.

W zadaniach 1–6 znajdź 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 dzielnik, 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

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

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

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

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

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

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

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

Zadanie 1 (2 pkt).

```
1  {z = i · j - x · y}
2  while ¬odd(x) do
3      y := 2 · y
4      x := x div 2
5  done
6  {z = i · j - x · y ∧ odd(x)}
```

Zadanie 2 (2 pkt).

```
1  {x = i ∧ y = j}
2  s := 0
3  while x ≠ 0 do
4      while ¬odd(x) do
5          y := 2 · y
6          x := x div 2
7      done
8      s := s + y
9      x := x - 1
10 done
11 {s = i · j}
```

Zadanie 3 (2 pkt).

```
1  {x = i ∧ y = j ∧ j ≥ 0}
2  z := 1
3  while y > 0 do
4      if odd(y)
5      then
6          z := z · x
7      fi
8      y := y div 2
9      x := x · x
10 done
11 {z = ij}
```

Zadanie 4 (2 pkt).

```
1  {x = i ∧ y = j}
2  while x ≠ y do
3      while x > y do
4          x := x - y
5      done
6      while y > x do
7          y := y - x
8      done
9  done
10 {x = y ∧ x = gcd(i, j)}
```

Zadanie 5 (2 pkt).

```
1  {z = i ∧ i ≥ 0}
2  x := 1
3  y := 1
4  while z > 0 do
5      y := x + y
6      x := y - x
7      z := z - 1
8  done
9  {x = Fi}
```

Zadanie 6 (2 pkt).

```
1  {x = i ∧ y = j}
2  while y ≠ 0 do
3      z := y
4      y := x mod y
5      x := z
6  done
7  {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?

```

1  m := 2 · n
2  k := 0
3  while m > 0 do
4      m := m - 1
5      k := k + 1
6      if m = 0
7          then
8              n := n - 1
9              m := 2 · n
10         fi
11     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

```

3  R := {t \stackrel{?}{=} s}
4  \theta := []
5  while R \neq \emptyset do
6      select t \stackrel{?}{=} s from R
7      case t \stackrel{?}{=} s of
8          x \stackrel{?}{=} x, where x \in \mathcal{X}:
9              skip
10         x \stackrel{?}{=} t \text{ or } t \stackrel{?}{=} x, where x \in \mathcal{X} \text{ and } x \neq t:
11             if x \notin FV(t)
12                 then
13                     R := R[x/t]
14                     \theta := \theta[x/t]
15             else
16                 return “not unifiable”
17         fi
18         f(t_1, \dots, t_n) \stackrel{?}{=} f(s_1, \dots, s_n):
19             R := R \cup \{t_1 \stackrel{?}{=} s_1, \dots, t_n \stackrel{?}{=} s_n\}
20         f(t_1, \dots, t_n) \stackrel{?}{=} g(s_1, \dots, s_m), where f \neq g:
21             return “not unifiable”
22     esac
23 done
24 return \theta

```

Zadanie 9 (2 pkt). Oto algorytm badający spełnialność formuły 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 \mathbf{t} , \mathbf{f} (por. *Whitebook*).

```

1  T := Succ(\mathbf{t})
2  F := Pred(\mathbf{f})
3  Satisfiable := T \cap F = \emptyset
4  while Satisfiable \wedge T \cup F \neq V do
5      x := choose(V \setminus (T \cup F))
6      if Succ(x) \cap Pred(\neg x) \neq \emptyset
7          then
8              x := \neg x
9          fi
10     if Succ(x) \cap Pred(\neg x) \neq \emptyset
11         then
12             Satisfiable := false
13         else
14             T := T \cup Succ(x)
15             F := F \cup Pred(\neg x)
16         fi
17     done
18     for x \in V do
19         \sigma_0(x) := \begin{cases} 1, & \text{if } x \in T \\ 0, & \text{otherwise} \end{cases}
20     done

```

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

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

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

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