Programowanie 2009 Programming 2009

Lista zadań nr 9 Problem set no. 9

Na zajęcia 5–6 maja 2009 Due May 6, 2009

Oto składnia abstrakcyjna języka „While" opisanego na wykładzie:

Here is the abstract synatax of the "While" language described during the lecture:

$$
\begin{array}{rcl}
\mathcal{N} & ::= & \text{integer literals} \\
\mathcal{X} & ::= & \text{identifiers} \\
\mathcal{A} & ::= & \mathcal{N} \mid \mathcal{X} \mid \mathcal{A}\ \mathcal{O}\ \mathcal{A} \\
\mathcal{O} & ::= & \texttt{+} \mid \texttt{-} \mid \texttt{*} \mid \texttt{div} \mid \texttt{mod} \\
\mathcal{B} & ::= & \texttt{true} \mid \texttt{false} \mid \mathcal{A}\ \mathcal{R}\ \mathcal{A} \mid \texttt{¬}\ \mathcal{B} \mid \mathcal{B}\ \mathcal{P}\ \mathcal{B} \\
\mathcal{R} & ::= & \texttt{<} \mid \texttt{≤} \mid \texttt{>} \mid \texttt{≥} \mid \texttt{=} \mid \texttt{≠} \\
\mathcal{P} & ::= & \texttt{∨} \mid \texttt{∧} \\
\mathcal{C} & ::= & \texttt{skip} \mid \mathcal{X}\ \texttt{:=}\ \mathcal{A} \mid \mathcal{C}\ \texttt{;}\ \mathcal{C} \mid \texttt{if}\ \mathcal{B}\ \texttt{then}\ \mathcal{C}\ \texttt{else}\ \mathcal{C} \mid \texttt{while}\ \mathcal{B}\ \texttt{do}\ \mathcal{C}
\end{array}
$$

Niech zbiór adresów $\mathbb{A} = \mathcal{X}$, przestrzeń stanów pamieci $\Pi = \mathbb{Z}^{\subseteq \mathcal{X}}$, a funkcja semantyczna dla wyrażeń arytmetycznych $[\![\cdot]\!] : \mathcal{A} \to \Pi \to \mathbb{Z}$, funkcja semantyczna dla wyrażeń logicznych $[\![\cdot]\!] : \mathcal{B} \to \Pi \to \{T, F\}$ i funkcja semantyczna dla instrukcji $[\![\cdot]\!] : \mathcal{C} \to \Pi \to \Pi$ będą zadane następująco:

Let the set of addresses $\mathbb{A} = \mathcal{X}$, the set of memory states $\Pi = \mathbb{Z}^{\subseteq \mathcal{X}}$, and the semantic function for arithmetic expressions $[\![\cdot]\!] : \mathcal{A} \to \Pi \to \mathbb{Z}$, the semantic function for Boolean expressions $[\![\cdot]\!] : \mathcal{B} \to \Pi \to \{T, F\}$ and the semantic function for instructions $[\![\cdot]\!] : \mathcal{C} \to \Pi \to \Pi$ be defined as follows:

$$
\begin{array}{rcl}
[\![n]\!]\pi & = & n, \quad \text{for } n \in \mathcal{N} \\
[\![X]\!]\pi & = & \pi(X), \quad \text{for } X \in \mathcal{X} \\
[\![a_1 \oplus a_2]\!]\pi & = & [\![a_1]\!]\pi \oplus [\![a_2]\!]\pi, \quad \text{for } (\oplus) \in \mathcal{O} \\
[\![\texttt{true}]\!]\pi & = & T \\
[\![\texttt{false}]\!]\pi & = & F \\
[\![a_1 \otimes a_2]\!]\pi & = & \begin{cases} T, & \text{if } [\![a_1]\!]\pi \otimes [\![a_2]\!]\pi, \\ F, & \text{otherwise} \end{cases} \quad \text{for } (\otimes) \in \mathcal{R} \\
[\![\texttt{¬}\ b]\!]\pi & = & \begin{cases} T, & \text{if } [\![b]\!]\pi = F, \\ F, & \text{otherwise} \end{cases} \\
[\![b_1\ \texttt{∨}\ b_2]\!]\pi & = & \begin{cases} T, & \text{if } [\![b_1]\!]\pi = T \text{ or } [\![b_2]\!]\pi = T, \\ F, & \text{otherwise} \end{cases} \\
[\![b_1\ \texttt{∧}\ b_2]\!]\pi & = & \begin{cases} T, & \text{if } [\![b_1]\!]\pi = T \text{ and } [\![b_2]\!]\pi = T, \\ F, & \text{otherwise} \end{cases} \\
[\![\texttt{skip}]\!]\pi & = & \pi \\
[\![X\ \texttt{:=}\ a]\!]\pi & = & \pi[X/[\![a]\!]\pi] \\
[\![c_1\ \texttt{;}\ c_2]\!] & = & [\![c_2]\!] \circ [\![c_1]\!] \\
[\![\texttt{if}\ b\ \texttt{then}\ c_1 \texttt{else}\ c_2]\!]\pi & = & \begin{cases} [\![c_1]\!]\pi, & \text{if } [\![b]\!]\pi = T, \\ [\![c_2]\!]\pi, & \text{otherwise} \end{cases} \\
[\![\texttt{while}\ b\ \texttt{do}\ c]\!]\pi & = & \begin{cases} [\![\texttt{while}\ b\ \texttt{do}\ c]\!]([\![c]\!]\pi), & \text{if } [\![b]\!]\pi = T, \\ \pi, & \text{otherwise} \end{cases}
\end{array}
$$

Oto reguły semantyki aksjomatycznej języka „While”:

Here are the rules of the axiomatic semantics of the “While” language:

$$\frac{}{\{\phi\}\ \texttt{skip}\ \{\phi\}} \qquad \frac{}{\{\phi[X/e]\}\ X\ \texttt{:=}\ e\ \{\phi\}}$$

$$\frac{\{\phi\}\ c_1\ \{\psi\} \quad \{\psi\}\ c_2\ \{\rho\}}{\{\phi\}\ c_1\,;c_2\ \{\rho\}} \qquad \frac{\{\phi \wedge b\}\ c_1\ \{\psi\} \quad \{\phi \wedge \neg b\}\ c_2\ \{\psi\}}{\{\phi\}\ \texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2\ \{\psi\}} \qquad \frac{\{\phi \wedge b\}\ c\ \{\phi\}}{\{\phi\}\ \texttt{while}\ b\ \texttt{do}\ c\ \{\phi \wedge \neg b\}}$$

$$\frac{\models \phi \Rightarrow \psi \quad \{\psi\}\ c\ \{\rho\}}{\{\phi\}\ c\ \{\rho\}} \qquad \frac{\{\phi\}\ c\ \{\psi\} \quad \models \psi \Rightarrow \rho}{\{\phi\}\ c\ \{\rho\}}$$

Reguła całkowitej poprawności dla instrukcji pętli:

The total completeness rule for the loop instruction:

$$\frac{[\phi \wedge b \wedge e = k]\ c\ [\phi \wedge e < k] \quad \models \phi \wedge b \Rightarrow e \geq 0}{[\phi]\ \texttt{while}\ b\ \texttt{do}\ c\ [\phi \wedge \neg b]}$$

Grupy zasadnicze

Basic groups

Zadanie 1 (1 pkt). Napisz kompletne drzewo dowodu dla poniższego sądu:

Problem 1 (1 p). Write a complete proof tree for the following judgement:

$\{\mathtt{x} \geq 0 \wedge \mathtt{y} > 0 \wedge \mathtt{x} = i \wedge \mathtt{y} = j\}$ `while x≠0 do z:=y mod x; y:=x; x:=z; done` $\{\mathtt{y} = \gcd(i, j)\}$

W poniższych zadaniach dopisz pomiędzy instrukcjami asercje tak, by otrzymać dowód, że programy spełniają podane specyfikacje. 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 $n \in \mathbb{N}$.

In the following problems insert assertions between instructions to obtain proofs that the programs satisfy given specifications. In these problems `odd` stands for oddity predicate, gcd — the greatest common divisor, and F_n — then n-th element of the Fibonacci sequence, *i.e.*, $F_0 = F_1 = 1$ and $F_{n+2} = F_{n+1} + F_n$ for $n \in \mathbb{N}$.

Zadanie/Problem 2 (1 p).

$\{\mathtt{x} = i \wedge \mathtt{y} = j\}$
```
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
```
$\{\mathtt{s} = i \cdot j\}$

Zadanie/Problem 3 (1 p).

$\{\mathtt{x} = i \wedge \mathtt{y} = j \wedge j \geq 0\}$
```
z := 1;
while y > 0 do
   if odd(y)
   then
      z := z*x;
   fi
   y := y div 2;
   x := x * x;
done
```
$\{\mathtt{z} = i^j\}$

Zadanie/Problem 4 (1 p).

$\{\mathtt{x} = i \wedge \mathtt{y} = j\}$
```
while x ≠ y do
   while x > y do
      x := x - y;
   done;
   while y > x do
      y = y - x;
   done;
done
```
$\{\mathtt{x} = \mathtt{y} \wedge \mathtt{x} = \gcd(i, j)\}$

Zadanie/Problem 5 (1 p).

$\{\mathtt{z} = i \wedge i \geq 0\}$
```
x := 1;
y := 1;
while z > 0 do
   y := x+y;
   x := y-x;
   z := z-1;
done
```
$\{\mathtt{x} = F_n\}$

Zadanie/Problem 6 (1 p).

$\{\mathtt{x} = i \wedge \mathtt{y} = j\}$
```
while y ≠ 0 do
   z := y;
   y := x mod y;
   x := z;
done
```
$\{\mathtt{x} = \gcd(i, j)\}$

Grupy rozszerzone

Zadanie 1 (1 pkt). Udowodnij twierdzenie: dla dowolnego programu C zachodzi $\{true\}C\{true\}$. Wyprowadź stąd wniosek: dla dowolnej asercji ϕ i dowolnego programu C zachodzi $\{\phi\}C\{true\}$. Udowodnij twierdzenie: dla dowolnego programu C zachodzi $\{false\}C\{false\}$. Wyprowadź stąd wniosek: dla dowolnej asercji ϕ i dowolnego programu C zachodzi $\{false\}C\{\phi\}$.

Zadanie 2 (1 pkt). Rozważmy następujący program:

Extended groups

Problem 1 (1 p). Prove the theorem: The assertion $\{true\}C\{true\}$ holds for any program C. Conclude that the assertion $\{\phi\}C\{true\}$ holds for any assertion ϕ and any program C. Prove the theorem: the assertion $\{false\}C\{false\}$ holds for any program C. Conclude that the assertion $\{false\}C\{\phi\}$ holds for any assertion ϕ and any program C.

Problem 2 (1 p). Consider the following program:

```
[ n = i ]
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
[ k = ? ]
```

Udowodnij, że powyższy program uruchomiony w dowolnym stanie pamięci zatrzyma się. Wpisz odpowiednie wyrażenie w miejsce znaku zapytania w warunku końcowym i udowodnij całkowitą poprawność powyższego programu względem podanych asercji.

Prove that the program presented above started in an arbitrary memory state eventually stops. Write an appropriate expression in place of the question mark in the postcondition and prove the total correctness of the program with respect to the given assertions.

Zadanie 3 (1 pkt). Rozważmy następujący algorytm znajdowania najogólniejszego unifikatora pary termów $t \stackrel{?}{=} s$:

Problem 3 (1 p). Consider the following algorithm for finding the most general unifier for a pair of terms $t \stackrel{?}{=} s$:

$R \leftarrow \{t \stackrel{?}{=} s\}$
$\theta \leftarrow []$
while $R \neq \emptyset$ **do**
  select $t \stackrel{?}{=} s$ **from** R
  case $t \stackrel{?}{=} s$ **of**
    $x \stackrel{?}{=} x$ where $x \in \mathcal{X}$:
      skip
    $x \stackrel{?}{=} t$ **or** $t \stackrel{?}{=} x$, where $x \in \mathcal{X}$ and $x \neq t$:
      if $x \notin \mathrm{FV}(t)$ **then**
        $R \leftarrow R[x/t]$
        $\theta \leftarrow \theta[x/t]$
      else return not unifiable
    $f(t_1, \ldots, t_n) \stackrel{?}{=} f(s_1, \ldots, s_n)$:
      $R \leftarrow R \cup \{t_1 \stackrel{?}{=} s_1, \ldots, t_n \stackrel{?}{=} s_n\}$
    $f(t_1, \ldots, t_n) \stackrel{?}{=} g(s_1, \ldots, s_m)$ where $f \neq g$:
      return not unifiable
return θ

Pokaż, że ten algorytm zawsze się zatrzymuje. Przyjmując następujący niezmiennik pętli:

Prove that this algorithm always eventually stops. Assuming the following loop invariant:

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

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

prove that when the algorithm stops θ is the most general unifier of t and s.

Zadanie 4 (1 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*).

Problem 4 (1 p). Here is an algorithm that tests satisfiability of 2-CNF formulas of the form $\bigwedge_{i=1}^{n}(k_i \Rightarrow l_i)$, where k_i and l_i are literals, *i.e.*, propositional variables, negated propositional variables or symbols **t**, **f** (see *the Whitebook*).

```
1   [ true ]
2   T := Succ(t)
3   F := Pred(f)
4   if T ∩ F ≠ ∅
5   then
6       Unsatisfiable := true
7   else
8       Unsatisfiable := false
9       while Unsatisfiable = false ∧ T ∪ F ≠ V do
10          x := choose(V \ (T ∪ F))
11          if Succ(x) ∩ Pred(¬x) ≠ ∅
12          then
13              x := ¬x
14          fi
15          if Succ(x) ∩ Pred(¬x) ≠ ∅
16          then
17              Unsatisfiable := true
18          else
19              T := T ∪ Succ(x)
20              F := F ∪ Pred(¬x)
21          fi
22      done
23      if Unsatisfiable = false
24      then
25          σ0(x) = { 1, if x ∈ T,   for x ∈ V
                    { 0, if x ∈ F,
26      fi
27  fi
28  [ (Unsatisfiable = true ∧ ∀σ.σ ⊭ φ) ∨ (Unsatisfiable = false ∧ σ0 ⊨ φ) ]
```

gdzie

where

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

a E^* oznacza zwrotne i przechodnie domknięcie relacji E. Wykaż poprawność tego algorytmu, tj. udowodnij podane asercje całkowitej poprawności.

and E^* stands for the symmetric and transitive closure of the relation E. Show the correctness of this algorithm, *i.e.*, prove the given total correctness assertions.

Zadanie 5 (1 pkt). Wynajdź reguły wnioskowania dla asercji częściowej poprawności dla następujących instrukcji:

Problem 5 (1 p). Find the inference rules for partial correctness assertions for the following instructions:

1. `abort`
2. `if` b `then` C
3. `repeat` C `until` b
4. `case` e `of` n_1`:` C_1`;` $\ldots$ `;` n_k`:` C_k `esac`
5. `for` x `from` e_1 `to` e_2 `do` C `done`
6. `for` x `from` e_1 `downto` e_2 `do` C `done`

Znaczenie instrukcji jest podobne jak w Pascalu.

The meaning of the instructions is like in Pascal.

Zadanie 6 (1 pkt). Napisz program C, który zamienia miejscami zawartość dwóch komórek pamięci `x` i `y` nie korzystając z żadnych innych komórek. Udowodnij, że

Problem 6 (1 p). Write a program C that swaps the contents of two memory cells `x` and `y` not using any other cells. Prove that:

$$\{\mathtt{x} = i \wedge \mathtt{y} = j\}\ C\ \{\mathtt{x} = j \wedge \mathtt{y} = i\}.$$

Co się stanie, gdy pominiemy (idealizujące) założenie, że komórki pamięci mogą przechowywać dowolnie wielkie liczby?

What happens if we drop an (idealistic) assumption that memory cells can store arbitrarily large numbers?

Grupa zaawansowana

Advanced group

Zadanie 1 (2 pkt). Niech

Problem 1 (2 p). Let

$$\eta, \pi \models \{\phi\}c\{\psi\} \quad \Longleftrightarrow \quad (\eta, \pi \models \phi) \wedge (\pi \in \mathrm{Dom}[\![c]\!]) \Rightarrow (\eta, [\![c]\!]\pi \models \psi)$$

Udowodnij adekwatność semantyki aksjomatycznej względem semantyki denotacyjnej, tj. pokaż, że jeśli $\vdash \{\phi\}c\{\psi\}$, to $\models \{\phi\}c\{\psi\}$ (każdy sąd dowodliwy jest prawdziwy).

Prove the soundness of the axiomatic semantics with respect to the denotational semantics, *i.e.*, show that if $\vdash \{\phi\}c\{\psi\}$, then $\models \{\phi\}c\{\psi\}$ (every provable judgement is true).

Zadanie 2 (2 pkt). Rozwiąż poprzednie zadanie dla asercji całkowitej poprawności, dla których:

Problem 2 (2 p). Solve the previous problem for total correctness assertions, for which:

$$\eta, \pi \models [\phi]c[\psi] \quad \Longleftrightarrow \quad (\eta, \pi \models \phi) \Rightarrow (\pi \in \mathrm{Dom}[\![c]\!]) \wedge (\eta, [\![c]\!]\pi \models \psi)$$