22 grudnia 2003

Zadanie 1
Dany jest język programowania:

Składnia

- $e ::= n \mid v \mid (e + e') \mid (e - e')$
- $b ::= t \mid b \vee b' \mid e = e' \mid e <= e' \mid \neg b$
- $c ::= \underline{\text{skip}} \mid v := e \mid c; c' \mid \underline{\text{if}}\ b\ \underline{\text{then}}\ c1\ \underline{\text{else}}\ c2 \mid \underline{\text{undo}}$

  gdzie $n \in \{\ldots, -1, 0, 1, \ldots\}$ – zbiór liczb całkowitych, a $t \in \{tt, ff\}$ – zbór stałych logicznych.

Instrukcja undo ma następujące znaczenie:

1. Jeśli jest to pierwsza instrukcja programu, to błąd.
2. Jeśli poprzednią wykonaną instrukcją było c, to przywracany jest stan sprzed wykonania c.

Podać postać konfiguracji dla instrukcji[1] i reguły dla instrukcji undo w przypadku

1. gdy $\underline{\text{undo}}\ ; \underline{\text{undo}} \equiv \underline{\text{skip}};$
2. gdy $c1; c2; \underline{\text{undo}}\ ; \underline{\text{undo}}$ oznacza stan sprzed wykonania $c1$.

Zakładamy, że dana jest semantyka operacyjna wyrażeń arytmetycznych i boolowskich oraz zdefiniowane są reguły dla instrukcji pustej, przypisania, złożenia instrukcji oraz $\underline{\text{if}}\ b\ \underline{\text{then}}\ c1\ \underline{\text{else}}\ c2$.

[1] Zauważmy, że postać konfiguracji podana na wykładzie nie zadziała w przypadku undo .

Zadanie 2
Język programowania zdefiniowany jest następująco:

Składnia

- $e ::= n \mid v \mid (e + e')$
- $d ::= \ \underline{\text{var}}\ v \mid d; d'$
- $c ::= \underline{\text{skip}} \mid v := e \mid c; c' \mid \ \underline{\text{begin}}\ d; c\ \underline{\text{end}}\ \mid \underline{\text{if}}\ v\ \underline{\text{then}}\ e1, e2$

Dziedziny semantyczne

- $Loc = \{l_0, l_1, \ldots\}$ – zbiór lokacji (nieskończony).
- $Sv = N$
- $S = [Loc \to Sv + \{\ \underline{\text{wolna}}\ \}]$
- $Dv = Loc + \{\ \underline{\text{puste}}\ \}$
- $Env = [Ide \to Dv]$
- $new : S \to Loc \times S$ dla $s \in S$

$$new(s) = \begin{cases} (l_i, s[0/l_i]) & \exists j.s(l_j) = \ \underline{\text{wolna}} \wedge \\ & i = min\{j \mid s(l_j) = \ \underline{\text{wolna}}\ \} \\ (l_0, s[0/l_0]) & \text{w p.p.} \end{cases}$$

Funkcje semantyczne

$\mathcal{A} : Aexp \to [Env \to [S \to N_\perp]]$

$\mathcal{D} : Decl \to [Env \to [S \to Env \times S]]$

$\mathcal{C} : Com \to [Env \to [S \to S_\perp]]$

Równania semantyczne

$\mathcal{A}[\![n]\!] = \lambda\rho \in Env.\lambda s \in S.n$

$\mathcal{A}[\![v]\!] = \lambda\rho \in Env.\lambda s \in S.\textbf{if } \rho v \in Loc \textbf{ then } s(\rho v) \textbf{ else } \perp_N$

$\mathcal{A}[\![e + e']\!] = \lambda\rho \in Env.\lambda s \in S.\textbf{if } \mathcal{A}[\![e]\!]\rho s = \perp_N \textbf{ then } \perp_N \textbf{ else}$
$\textbf{if } \mathcal{A}[\![e']\!]\rho s = \perp_N \textbf{ then } \perp_N \textbf{ else } (\mathcal{A}[\![e]\!]\rho s + \mathcal{A}[\![e']\!]\rho s)$

$\mathcal{D}[\![\ \underline{\text{var}}\ v]\!] = \lambda\rho \in Env.\lambda s \in S.(\rho[(new(s))_1/v], (new(s))_2)$

$\mathcal{D}[\![d; d']\!] = \lambda\rho \in Env.\lambda s \in S.\mathcal{D}[\![d']\!](\mathcal{D}[\![d]\!]\rho s)_1(\mathcal{D}[\![d]\!]\rho s)_2$

$\mathcal{C}[\![\underline{\text{skip}}]\!] = \lambda\rho \in Env.\lambda s \in S.s$

$\mathcal{C}[\![v := e]\!] = \lambda\rho \in Env.\lambda s \in S.\textbf{if } \rho v \notin Loc \textbf{ then } \perp_S \textbf{ else}$
$\textbf{if } \mathcal{A}[\![e]\!]\rho s = \perp_N \textbf{ then } \perp_S \textbf{ else } s[\mathcal{A}[\![e]\!]\rho s/\rho v]$

$\mathcal{C}[\![\ \underline{\text{begin}}\ d; c\ \underline{\text{end}}\ ]\!] = \lambda\rho \in Env.\lambda s \in S.\mathcal{C}[\![c]\!](\mathcal{D}[\![d]\!]\rho s)_1(\mathcal{D}[\![d]\!]\rho s)_2$

$\mathcal{C}[\![c; c']\!] = \lambda\rho \in Env.\lambda s \in S.\textbf{if } \mathcal{C}[\![c]\!]\rho s = \perp_S \textbf{ then } \perp_S \textbf{ else } \mathcal{C}[\![c']\!]\rho(\mathcal{C}[\![c]\!]\rho s)$

$\mathcal{C}[\![\underline{\text{if}}\ v\ \underline{\text{then}}\ e1, e2]\!] = \lambda\rho \in Env.\lambda s \in S.\textbf{if } \rho v \notin Loc \textbf{ then } \perp_S \textbf{ else if } \mathcal{A}[\![v]\!]\rho s = \perp_N \textbf{ then } \perp_S$
$\textbf{else if } \mathcal{A}[\![v]\!]\rho s = \mathcal{A}[\![0]\!]\rho s \textbf{ then if } \mathcal{A}[\![e1]\!]\rho s = \perp_N \textbf{ then } \perp_S \textbf{ else } s[\mathcal{A}[\![e1]\!]\rho s/\rho v]$
$\textbf{else if } \mathcal{A}[\![e2]\!]\rho s = \perp_N \textbf{ then } \perp_S \textbf{ else } s[\mathcal{A}[\![e2]\!]\rho s/\rho v]$

Podaj wartości zmiennych x, y i z we wskazanych momentach programu:

(a)

```
begin
    var x;  var y;
     x:=1; y:=2;
    begin
       var z;  var x;
        if z  then x+y, 1;
        if x  then x+y, z;
       {z=       , x=       , y=       }
    end ;
   if y  then y, x;
{ x=       , y=       }
end ;
```

(b)

```
begin
    var x;  var y;  var z;
     y:=2;
     if x  then y, z;
    {z=       , y=       , x=       }
        begin
          var y;  var z;
          if y  then y+1, x ;
          if y  then y+1, x;
          z:=y+x
         {z=       , y=       , x=       }
        end ;
if z  then x+y, z+1;
{z=       , y=       , x=       }
end ;
```

Zadanie 3

Udowodnij poprawność programu względem danej specyfikacji wstawiając stosowne asercje w nawiasy klamrowe.

$\{Y > 0 \wedge X \geq 0\}$

```
{                                                                    }
x:=X;
y:=Y;
z:=0;
{                                                                    }
while x<>0 do
{                                                                    }
    begin
    {                                                                }

    if not( 2|x) then z:=z+y;
    {                                                                }
       y:=y*2;
       x:=x div 2;
{                                                                    }
    end;
```

$\{z = X * Y\}$