21 czerwca 2004

Zadanie 1
Dany jest język programowania:

Składnia

- $e ::= n \mid v \mid (e + e') \mid (e - e')$
- $b ::= t \mid b \vee b' \mid e = e' \mid e <= e' \mid \neg b$[1]
- $c ::= \underline{\text{skip}} \mid v := e \mid c; c' \mid \underline{\text{do}}\ e\ \underline{\text{times}}\ c$

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

Zakładając, że semantyka operacyjna wyrażeń arytmetycznych i boolowskich oraz instrukcji pustej, przypisania i złożenia instrukcji jest standardowa i dane są dla nich reguły (patrz wykład), zdefiniować za pomocą semantyki operacyjnej konstrukcję $\underline{\text{do}}\ e\ \underline{\text{times}}\ c$. Znaczenie tej konstrukcji ma być następujące: instrukcja c wykonuje się tyle razy, ile wskazuje wartość wyrażenia e. Podać reguły przy założeniu, że

1. wartość wyrażenia e wyliczana jest tylko raz przed pierwszym wykonaniem pętli;
2. wartość wyrażenia e wyliczana jest przed każdym wykonaniem pętli.

   W obu przypadkach uwzględnić możliwość powstania błędów (np. $e < 0$).

[1] Mając zdefiniowane $=, <= i \neg$ możemy uznać, że zdefiniowane są wszystkie relacje pochodne, tzn. $<, >, >=, <>$

Zadanie 2
Język programowania zdefiniowany jest następująco:

Składnia

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

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

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

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

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

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

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

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

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

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

$\qquad\qquad \textbf{if } \mathcal{A}[\![e]\!]\rho s = \perp_N \textbf{ then } \perp_S \textbf{ else } s[\mathcal{A}[\![e]\!]\rho s/\rho v]$

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

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

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

(a)

```
begin
    var x;  var y;
     x:=1; y:=2;
    begin
        var z;  eqv (y,x)
         y:=8; z:=x+y
        {z=       , x=       , y=       }
    end ;{ x=       , y=       }
end ;
```

(b)

```
begin
    var x;  var y;  var z;
     y:=2; z:=3;
    begin
        eqv (y,x);
         y:=x+z;
        {y=       , x=       }
         begin
           var x;
           x:=3; z:=x+y;
          {z=       , y=       , x=       }
         end {y=       , x=       }
    end ;{y=       , x=       }
end ;
```

Zadanie 3
Udowodnij poprawność programu względem danej specyfikacji wstawiając stosowne asercje w nawiasy klamrowe.

$\{X > 1 \wedge Y \geq 0\}$

```
{                                                    }
x:=X;
y:=Y;
z:=1;
{                                                     }
while y<>0 do
{                                                     }
    begin
    {                                                 }
    while 2|y do
    {                                                 }
       begin
       {                                              }
       x:=x*x;
       y:=y/2;
       {                                                }
       end;
    {                                                   }
    {                                                   }
    y:=y-1;
    z:=z*x
{                                                       }
    end;
```

$\{z = X^Y\}$