Zadanie 1

Do języka zdefiniowanego na wykładzie dodaj do wyrażeń nową konstrukcję :

$e ::= m \mid v \mid (e + e') \mid (e - e') \mid (e * e') \mid$ if e then $e1$ else $e2$

Zadanie 2

Do języka zdefiniowanego na wykładzie dodaj do wyrażeń nową konstrukcję :

$e ::= m \mid v \mid (e + e') \mid (e - e') \mid (e * e') \mid$ after c resultis e

Rozważ dwa przypadki

1. bez efektów ubocznych

2. z efektami ubocznymi

Zadanie 3

Do języka zdefiniowanego na wykładzie dodaj do wyrażeń nową konstrukcję :

$e ::= m \mid v \mid (e + e') \mid (e - e') \mid (e * e') \mid \textbf{let } x = e \textbf{ in } e'$

Rozwiązanie:

Zadanie 3

Do języka zdefiniowanego na wykładzie dodaj do wyrażeń nową konstrukcję :

$$e ::= m \mid v \mid (e + e') \mid (e - e') \mid (e * e') \mid \textbf{let } x = e \textbf{ in } e'$$

Rozwiązanie:

$$\frac{\langle e_1, s\rangle \rightsquigarrow \langle e_1', s\rangle}{\langle \textbf{let } x = e_1 \textbf{ in } e_2, s\rangle \rightsquigarrow \langle \textbf{let } x = e_1' \textbf{ in } e_2, s\rangle}$$

$$\frac{\langle e, s[x \mapsto n]\rangle \rightsquigarrow \langle e', s[x \mapsto n]\rangle}{\langle \textbf{let } x = n \textbf{ in } e, s\rangle \rightsquigarrow \langle \textbf{let } x = n \textbf{ in } e', s\rangle}$$

$$\langle \textbf{let } x = n \textbf{ in } n', s\rangle \rightsquigarrow \langle n', s\rangle$$

Zadanie 4

W języku zdefiniowanym na wykładzie zastąp instrukcję while instrukcją repeat , tj. zdefiniuj:

$c ::=$ skip $\mid v := e \mid c; c' \mid$ if b then c else $c' \mid$ repeat c until b

Zadanie 5

W języku zdefiniowanym na wykładzie zastąp instrukcję while instrukcją for , tj. zdefiniuj:

$c ::= \underline{\text{skip}} \mid v := e \mid c; c' \mid \underline{\text{if}}\ b\ \underline{\text{then}}\ c\ \underline{\text{else}}\ c' \mid \underline{\text{for}}\ v := e1\ \underline{\text{to}}\ e2\ \underline{\text{do}}\ c$

Rozważ dwa przypadki

1. wartość wyrażenia $e2$ oblicza się dokładnie raz przed pierwszym wykonaniem pętli

2. wartość wyrażenia $e2$ oblicza się przed każdym wykonaniem pętli

Zadanie 6

Do języka zdefiniowanego na wykładzie dodaj nową instrukcję or , tj.

$c ::= \underline{\text{skip}} \mid v := e \mid c; c' \mid \underline{\text{if}}\ \ b\ \underline{\text{then}}\ \ c\ \underline{\text{else}}\ \ c' \mid \underline{\text{while}}\ b\ \underline{\text{do}}\ c \mid c1\ \underline{\text{or}}\ c2$

Zadanie 7

Do języka zdefiniowanego na wykładzie dodaj do wyrażeń nową konstrukcję :

$e ::= m \mid v \mid (e + e') \mid (e - e') \mid (e * e') \mid$ after c resultis e

Rozważ dwa przypadki

1. bez efektów ubocznych

2. z efektami ubocznymi

Semantyka denotacyjna – rozwiązanie
Ad. 1. *funkcje semantyczne*

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

$$\mathcal{B} : Bexp \to [\Sigma_\perp \to B_\perp]$$

$$\mathcal{C} : Com \to [\Sigma_\perp \to \Sigma_\perp]$$

Równania opisujące wszystkie wyrażenia z wyjątkiem $\underline{\text{after}}\ c\ \underline{\text{resultis}}\ e$ pozostają bez zmian.

$$\mathcal{A}[[\ \underline{\text{after}}\ c\ \underline{\text{resultis}}\ e]] = \lambda\sigma \in \Sigma.\mathcal{A}[[e]](\mathcal{C}[[c]]\sigma)$$

Ad. 2. Teraz mamy:

$$\mathcal{A} : Aexp \to [\Sigma_\perp \to N_\perp \times \Sigma_\perp]$$

$$\mathcal{A}[[n]] = \lambda\sigma \in \Sigma.(n, \sigma)$$

$$\mathcal{A}[[v]] = \lambda\sigma \in \Sigma.(\sigma(v), \sigma)$$

$$\mathcal{A}[[e_0 + e_1]] = \lambda\sigma \in \Sigma.\textbf{if } (\mathcal{A}[[e_0]]\sigma)_1 = \perp_N \textbf{ then } (\perp_N, \perp_\Sigma) \textbf{ else}$$
$$\textbf{let } (n1, \sigma') = \mathcal{A}[[e_0]]\sigma \textbf{ in } (\textbf{if } (\mathcal{A}[[e_1]]\sigma')_1 = \perp_N \textbf{ then } (\perp_N, \perp_\Sigma)$$
$$\textbf{else let } (n2, \sigma'') = \mathcal{A}[[e_1]]\sigma' \textbf{ in } (n1 + n2, \sigma''))$$

$$\mathcal{A}[[\ \underline{\text{after}}\ c\ \underline{\text{resultis}}\ e]] = \lambda\sigma \in \Sigma.\mathcal{A}[[e]](\mathcal{C}[[c]]\sigma)$$

Zadanie 8

Do języka zdefiniowanego na wykładzie dodaj instrukcję:

$$c ::= \underline{\text{skip}} \mid v := e \mid c; c' \mid \underline{\text{if}}\ \ b\ \underline{\text{then}}\ \ c\ \underline{\text{else}}\ \ c' \mid \underline{\text{while}}\ b\ \underline{\text{do}}\ c \mid\ \ \underline{\text{var}}\ v = e\ \textbf{in}\ c$$

Zadanie 8

Do języka zdefiniowanego na wykładzie dodaj instrukcję:

$$c ::= \underline{\text{skip}} \mid v := e \mid c; c' \mid \underline{\text{if}}\ \ b\ \underline{\text{then}}\ \ c\ \underline{\text{else}}\ \ c' \mid \underline{\text{while}}\ b\ \underline{\text{do}}\ c \mid\ \ \underline{\text{var}}\ v = e\ \textbf{in}\ c$$

Rozwiązanie:

$$\mathcal{C}[[\ \underline{\text{var}}\ v = e\ \textbf{in}\ c]] = \lambda\sigma \in \Sigma.(\mathcal{C}[[c]]\sigma[\mathcal{A}[[e]]\sigma/v])[\sigma(v)/v]$$

Zadanie 9

Udowodnić za pomocą reguł logiki Hoare'a **częściową poprawność** programu w języku PJP.

$\{(Y > 0) \wedge (X > 0)\}$

$a := X;$

$b := Y;$

$z := 1;$

while $\neg(a = 0)$ **do**

(

if $\neg(a$ **mod** $2 = 0)$ **then**

$z := z * b$

else skip;

$b := b * b;$

$a := a$ **div** 2

)

$\{z = Y^X\}$

$\{(Y > 0) \wedge (X > 0)\}$
$\{ \qquad \}$
$a := X;\ b := Y;\ z := 1;$
$\{ \qquad \}$
while $\neg(a = 0)$ **do** $\{ \qquad \}$
$(\{ \qquad \}$
if $\neg(a$ **mod** $2 = 0)$ **then** $z := z * b$
else skip;
$\{ \qquad \}$
$b := b * b;$
$\{ \qquad \}$
$a :=$ a **div** 2
$\{ \qquad \})$
$\{ \qquad \}$
$\{z = Y^X\}$

$\{(Y > 0) \wedge (X > 0)\}$
$\{$ $\}$
$a := X;\ b := Y;\ z := 1;$
$\{(Y > 0) \wedge (X > 0) \wedge (z * b^a = Y^X)\}$
while $\neg(a = 0)$ **do** $\{(Y > 0) \wedge (X > 0) \wedge (z * b^a = Y^X) \wedge a \neq 0\}$
$(\{$
$\}$
if $\neg(a$ **mod** $2 = 0)$ **then** $z := z * b$
else skip;
$\{$ $\}$
$b := b * b;$
$\{$ $\}$
$a := a$ **div** 2
$\{(Y > 0) \wedge (X > 0) \wedge (z * b^a = Y^X)\})$
$\{$ $\}$
$\{z = Y^X\}$

$\{(Y > 0) \wedge (X > 0)\}$
$\{$ $\}$
$a := X;\ b := Y;\ z := 1;$
$\{(Y > 0) \wedge (X > 0) \wedge (z * b^a = Y^X)\}$
while $\neg(a = 0)$ **do** $\{(Y > 0) \wedge (X > 0) \wedge (z * b^a = Y^X) \wedge a \neq 0\}$
  $(\{$
    $\}$
    if $\neg(a$ **mod** $2 = 0)$ **then** $z := z * b$
                else skip;
    $\{$ $\}$
    $b := b * b;$
    $\{(Y > 0) \wedge (X > 0) \wedge (z * b^a$ **div** $^2 = Y^X)\}$
    $a := a$ **div** 2
    $\{(Y > 0) \wedge (X > 0) \wedge (z * b^a = Y^X)\})$
$\{$ $\}$
$\{z = Y^X\}$

$\{(Y > 0) \wedge (X > 0)\}$

$\{$ $\}$

$a := X;\ b := Y;\ z := 1;$

$\{(Y > 0) \wedge (X > 0) \wedge (z * b^a = Y^X)\}$

while $\neg(a = 0)$ **do** $\{(Y > 0) \wedge (X > 0) \wedge (z * b^a = Y^X) \wedge a \neq 0\}$

$(\{$

$\}$

if $\neg(a$ **mod** $2 = 0)$ **then** $z := z * b$

else skip;

$\{(Y > 0) \wedge (X > 0) \wedge (z * (b * b)^a$ **div** $^2 = Y^X)\}$

$b := b * b;$

$\{(Y > 0) \wedge (X > 0) \wedge (z * b^a$ **div** $^2 = Y^X)\}$

$a := a$ **div** 2

$\{(Y > 0) \wedge (X > 0) \wedge (z * b^a = Y^X)\})$

$\{$ $\}$

$\{z = Y^X\}$

$\{(Y > 0) \wedge (X > 0)\}$

$\{$ $\}$

$a := X;\ b := Y;\ z := 1;$

$\{(Y > 0) \wedge (X > 0) \wedge (z * b^a = Y^X)\}$

while $\neg(a = 0)$ **do** $\{(Y > 0) \wedge (X > 0) \wedge (z * b^a = Y^X) \wedge a \neq 0\}$

$(\{(Y > 0) \wedge (X > 0) \wedge ((z * (b^2)^a \ \mathbf{div}\ 2 = Y^X) \wedge (a \ \mathbf{mod}\ 2 = 0) \vee$

$(z * b * (b^2)^a \ \mathbf{div}\ 2 = Y^X) \wedge (a \ \mathbf{mod}\ 2 \neq 0))\}$

if $\neg(a \ \mathbf{mod}\ 2 = 0)$ **then** $z := z * b$

else skip;

$\{(Y > 0) \wedge (X > 0) \wedge (z * (b * b)^a \ \mathbf{div}\ 2 = Y^X)\}$

$b := b * b;$

$\{(Y > 0) \wedge (X > 0) \wedge (z * b^a \ \mathbf{div}\ 2 = Y^X)\}$

$a := a \ \mathbf{div}\ 2$

$\{(Y > 0) \wedge (X > 0) \wedge (z * b^a = Y^X)\})$

$\{$ $\}$

$\{z = Y^X\}$

$\{(Y > 0) \wedge (X > 0)\}$

$\{$ $\}$

$a := X;\ b := Y;\ z := 1;$

$\{(Y > 0) \wedge (X > 0) \wedge (z * b^a = Y^X)\}$

while $\neg(a = 0)$ **do** $\{(Y > 0) \wedge (X > 0) \wedge (z * b^a = Y^X) \wedge a \neq 0\}$

$(\{(Y > 0) \wedge (X > 0) \wedge ((z * (b^2)^a \ \textbf{div}\ 2 = Y^X) \wedge (a \ \textbf{mod}\ 2 = 0) \vee$

$(z * b * (b^2)^a \ \textbf{div}\ 2 = Y^X) \wedge (a \ \textbf{mod}\ 2 \neq 0))\}$

if $\neg(a \ \textbf{mod}\ 2 = 0)$ **then** $z := z * b$

else skip;

$\{(Y > 0) \wedge (X > 0) \wedge (z * (b * b)^a \ \textbf{div}\ 2 = Y^X)\}$

$b := b * b;$

$\{(Y > 0) \wedge (X > 0) \wedge (z * b^a \ \textbf{div}\ 2 = Y^X)\}$

$a := a \ \textbf{div}\ 2$

$\{(Y > 0) \wedge (X > 0) \wedge (z * b^a = Y^X)\})$

$\{(Y > 0) \wedge (X > 0) \wedge (z * b^a = Y^X) \wedge (a = 0)\}$

$\{z = Y^X\}$

$\{(Y > 0) \wedge (X > 0)\}$
$\{(Y > 0) \wedge (X > 0) \wedge (Y^X = Y^X)\}$
$a := X;\ b := Y;\ z := 1;$
$\{(Y > 0) \wedge (X > 0) \wedge (z * b^a = Y^X)\}$
while $\neg(a = 0)$ **do** $\{(Y > 0) \wedge (X > 0) \wedge (z * b^a = Y^X) \wedge a \neq 0\}$
$(\{(Y > 0) \wedge (X > 0) \wedge ((z * (b^2)^a \ \mathbf{div}\ 2 = Y^X) \wedge (a \ \mathbf{mod}\ 2 = 0) \vee$
$(z * b * (b^2)^a \ \mathbf{div}\ 2 = Y^X) \wedge (a \ \mathbf{mod}\ 2 \neq 0))\}$
if $\neg(a \ \mathbf{mod}\ 2 = 0)$ **then** $z := z * b$
else skip;
$\{(Y > 0) \wedge (X > 0) \wedge (z * (b * b)^a \ \mathbf{div}\ 2 = Y^X)\}$
$b := b * b;$
$\{(Y > 0) \wedge (X > 0) \wedge (z * b^a \ \mathbf{div}\ 2 = Y^X)\}$
$a := a \ \mathbf{div}\ 2$
$\{(Y > 0) \wedge (X > 0) \wedge (z * b^a = Y^X)\})$
$\{(Y > 0) \wedge (X > 0) \wedge (z * b^a = Y^X) \wedge (a = 0)\}$
$\{z = Y^X\}$

Zadanie 10

W języku zdefiniowanym na wykładzie zastąp instrukcję while instrukcją repeat , tj. skonstruuj regułę Hoare'a:

$c ::= \text{skip} \mid v := e \mid c; c' \mid \text{if } b \text{ then } c \text{ else } c' \mid \text{repeat } c \text{ until } b$

Zadanie 10

W języku zdefiniowanym na wykładzie zastąp instrukcję while instrukcją repeat , tj. skonstruuj regułę Hoare'a:

$c ::= \underline{\text{skip}} \mid v := e \mid c; c' \mid \underline{\text{if}}\ \ b\ \underline{\text{then}}\ \ c\ \underline{\text{else}}\ \ c' \mid\ \underline{\text{repeat}}\ c\ \underline{\text{until}}\ b$

$$\frac{\{\alpha\}c\{\beta\},\ \beta \wedge \neg b \Rightarrow \alpha}{\{\alpha\}\ \underline{\text{repeat}}\ c\ \underline{\text{until}}\ b\ \{\beta \wedge b\}}$$

Zadanie 11

Do języka programowania zdefiniowanego na wykładzie dodaj do deklaracji konstrukcję eqv , która oznacza zadeklarowanie zmiennej v, która ma mieć tę samą lokację co zmienna v'.

$d ::=$ var $v \mid$ eqv $(v, v') \mid d; d'$

Rozwiązanie – semantyka denotacyjna

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 $s \in S$

$$new(s) = (l_i, s[0/l_i]) \quad \exists j.s(l_j) = \underline{\text{wolna}} \wedge i = min\{j \mid s(l_j) = \underline{\text{wolna}}\ \}$$

Funkcja semantyczna

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

Równanie semantyczne

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

Zadanie 12

Do języka programowania zdefiniowanego na wykładzie dodaj do deklaracji konstrukcję exp

$d ::=$ var $v \mid$ exp $v = e \mid d; d'$

Zadanie 12

Do języka programowania zdefiniowanego na wykładzie dodaj do deklaracji konstrukcję exp

$d ::=$ var $v \mid$ exp $v = e \mid d; d'$

Nowe dziedziny semantyczne

- $Exp = [S \to N_\perp]$
- $Dv = Loc + Exp + \{$ puste $\}$

Zadanie 12

Do języka programowania zdefiniowanego na wykładzie dodaj do deklaracji konstrukcję exp

$d ::=$ var $v \mid$ exp $v = e \mid d; d'$

Nowe dziedziny semantyczne

- $Exp = [S \to N_\perp]$
- $Dv = Loc + Exp + \{$ puste $\}$

Równania semantyczne

$\mathcal{D}[[$ exp $v = e]] = \lambda\rho \in Env.\lambda s \in S.(\rho[\mathcal{A}[[e]]\rho/v], s)$

Zadanie 12

Do języka programowania zdefiniowanego na wykładzie dodaj do deklaracji konstrukcję exp

$d ::=$ var $v \mid$ exp $v = e \mid d; d'$

Nowe dziedziny semantyczne

- $Exp = [S \to N_\perp]$
- $Dv = Loc + Exp + \{$ puste $\}$

Równania semantyczne

$\mathcal{D}[[$ exp $v = e]] = \lambda\rho \in Env.\lambda s \in S.(\rho[\mathcal{A}[[e]]\rho/v], s)$

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

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

Zadanie 13

Do języka programowania zdefiniowanego na wykładzie dodaj procedury bezparametrowe takie, że zmienne globalne są traktowane statycznie, a procedury dynamicznie.

Rozwiązanie

Dziedziny semantyczne jak poprzednio oraz

- $Env_P = [PIde \to Proc + \{ \underline{\text{puste}} \}]$
- $Env_V = [VIde \to Loc + \{ \underline{\text{puste}} \}]$
- $Proc = [Env_P \to [S \to S_\perp]]$

$$\mathcal{D}_P[[\underline{\text{proc}}\ P; c]] = \lambda\rho_P \in Env_P.\lambda\rho_V \in Env_V.\lambda s \in S.\rho_P[f/P]$$

gdzie $f \in Proc$ jest zdefiniowana następująco:

$$f = \lambda\rho'_P \in Env_P.\lambda s' \in S.\mathcal{C}[[c]]\rho'_P\rho_V s'$$

Wywołanie

$\mathcal{C}$[[call P]] $= \lambda\rho_P \in Env_P.\lambda\rho_V \in Env_V.\lambda s \in S.$**if** $\rho_P P \notin Proc$ **then** $\perp_S$ **else** $(\rho_P P)\rho_P s.$

Zadanie 14

Zdefiniować przekazywanie parametrów przez wartość-wynik (value-result).

Rozważmy przykład:

```
 proc silnia(n,s);
      s:=1;
      while n>0  do (s:=s*n;n:=n-1)
```

i wywołanie

```
 call silnia(x,x)
```

Idea koncepcji wartość-wynik

```
n:=x;
s:=x;
      s:=1;
     while n>0  do (s:=s*n;n:=n-1)
x:=n;
x:=s;
```

Rozwiązanie

$Proc = [Loc \to [Loc \to [S \to S_\perp]]]$

$\mathcal{D}[[\ \underline{\text{proc}}\ P(v1, v2); c]] = \lambda\rho \in Env.\lambda s \in S.(\rho[f/P], s)$

gdzie $f \in Proc$ jest zdefiniowana następująco:

$f = \lambda l1 \in Loc.\lambda l2 \in Loc.\lambda s1 \in S.$(**let** $(l', s') = new(s1)$ **in**
(**let** $(l'', s'') = new(s')$ **in**
(**let** $s''' = \mathcal{C}[[c]]\rho[l'/v1, l''/v2]s''[s1(l1)/l', s1(l2)/l'']$))) **in**
$s'''[s'''(l')/l1][s'''(l'')/l2]$

$\mathcal{C}[[\ \underline{\text{call}}\ P(v1, v2)]] = \lambda\rho \in Env.\lambda s \in S.$**if** $\rho P \notin Proc$ **then** $\perp_S$
else if $\rho v1 \notin Loc$ **then** $\perp_S$ **else if** $\rho v2 \notin Loc$ **then** $\perp_S$
else $(\rho P)(\rho v1)(\rho v2)s.$

Zadanie 15

Język programowania rozszerzyć o funkcje. Zakładamy statyczne traktowanie zmiennych globalnych.

Składnia

- $e ::= n \mid v \mid F(e) \mid (e + e')$
- $d ::= \ \underline{\text{var}}\ v \mid \quad \underline{\text{fun}}\ f(v); e \mid d; d'$

Dziedziny semantyczne

- $VEnv = [Ide \to Loc + \{\ \underline{\text{puste}}\ \}]$
- $FEnv = [Ide \to Fun]$
- $Fun = [N \to [S \to N_\perp]]$

Funkcje semantyczne

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

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

Równania semantyczne

$$\mathcal{A}[[F(e)]] = \lambda\rho_v \in VEnv.\lambda\rho_f \in FEnv.\lambda s \in S.\textbf{if } \rho_f F \notin Fun \textbf{ then } \perp_N$$
$$\textbf{else if } \mathcal{A}[[e]]\rho_v\rho_f s \notin Sv \textbf{ then } \perp_N \textbf{ else } (\rho_f F)(\mathcal{A}[[e]]\rho_v\rho_f s)s.$$

$\mathcal{D}[[$ $\underline{\text{fun}}$ $F(v); e]] = \lambda\rho_v \in VEnv.\lambda s \in S.(\rho_v, \rho_f[f/F], s)$

gdzie $f \in Fun$ jest zdefiniowana następująco:

$f = \lambda w \in Sv.\lambda s' \in S.$ **let** $(l'', s'') = new(s')$ **in**

$\mathcal{A}[[e]](\rho_v[l''/v])\rho_f(s''[w/l''])$