Semantyka i weryfikacja programów 2023/24.
Egzamin poprawkowy 24/02/2024, zadanie 2 (semantyka denotacyjna)

Poniżej zaproponowana jest składnia pewnego języka programowania.

$$
\begin{aligned}
\textbf{Var} \ni x &::= x \mid y \mid z \mid \ldots \\
\textbf{Label} \ni j &::= i \mid j \mid k \mid \ldots \\
\textbf{Num} \ni n &::= \ldots \mid -1 \mid 0 \mid 1 \mid \ldots \\
\textbf{Expr} \ni e &::= n \mid x \mid e_1 + e_2 \mid e_1 - e_2 \mid e_1 \cdot e_2 \\
\textbf{Decl} \ni d &::= \textbf{var}\ x = e \mid \textbf{label}\ j \mid d_1; d_2 \\
\textbf{Inst} \ni I &::= \textbf{skip} \mid x := e \mid I_1; I_2 \mid \textbf{if}\ e >= 0\ \textbf{then}\ \{I_1\}\ \textbf{else}\ \{I_2\} \mid \\
&\qquad \textbf{begin}\ d\ \textbf{in}\ I\ \textbf{end} \mid \ j_1 := j_2 \mid \textbf{set}\ j \mid \textbf{goto}\ j
\end{aligned}
$$

Wyrażenia i instrukcje oprócz wymienionych poniżej mają standardowe znaczenie. Wyrażenia arytmetyczne wyliczają się do wartości liczbowych **Num** bez efektów ubocznych. Można założyć, że dana jest semantyka dla wyrażeń w stylu denotacyjnym, należy jedynie **jawnie zadeklarować** (sensowny) typ funkcji semantycznej. Język zawiera bloki **begin** d **in** I **end**, które pozwalają na deklarowanie widocznych lokalnie w danym bloku zmiennych.

Deklaracja zmiennej indywiduowej **var** $x = e$ w bloku wprowadza zmienną lokalną x o wartości początkowej równej aktualnej wartości wyrażenia e. Zmienna ta jest widoczna w sposób statyczny w ciele tego bloku. Mają tu zastosowanie standardowe zasady przesłaniania zmiennych.

Deklaracja **label** j wprowadza *zmienną skokową* o nazwie j. Zasady statycznej widoczności i przesłaniania tych zmiennych są identyczne jak w przypadku zmiennych indywiduowych. Wartością zmiennej skokowej j jest miejsce skoku, początkowa wartość takiej zmiennej powoduje skok na koniec bloku, w którym ta zmienna została zadeklarowana. Późniejsze przypisania $j_1 := j_2$ pozwalają zmieniać aktualne wartości zmiennych skokowych. Podobnie, instrukcja **set** j powoduje ustawienie aktualnej wartości zmiennej skokowej j na miejsce w kodzie następujące tuż za tą instrukcją. Wreszcie, instrukcja **goto** j powoduje natychmiastowy skok do miejsca w kodzie wskazywanego przez aktualną wartość zmiennej skokowej j.

Program, w którym używane są zmienne niewidoczne w danym kontekście, może mieć nieokreślone działanie.

Zadanie

Zadanie polega na napisaniu semantyki denotacyjnej w stylu kontynuacyjnym dla kategorii syntaktycznych **Inst** oraz **Decl** powyższego języka. W tym celu należy jednoznacznie zdefiniować typy funkcji semantycznych dla **Expr**, **Decl** oraz **Inst** wraz ze **wszystkimi** używanymi typami pomocniczymi. Można założyć, że dany jest typ **Ans** oznaczający finalne wyniki działania programu. Można też założyć, że dany jest nieskończony zbiór **Loc** wraz z odpowiednimi funkcjami newloc zwracającymi nową, nieużywaną lokację. Typy wykorzystywanych funkcji newloc (jednej lub więcej) należy **jawnie** zdefiniować. Należy też podać równania semantyczne dla **wszystkich** postaci deklaracji **Decl** oraz **wszystkich** postaci instrukcji **Inst** w tym języku.

verte!

Przykład

Rozważmy następującą instrukcję tego języka:

```
 0: begin
 1:   label i;
 2:   var x = 0
 3: in
 4:   begin
 5:     label k
 6:   in
 7:     x := x - 2;
 8:     set k;
 9:     i := k
10:   end;
11:   x := x + 1;
12:   if x >= 0 then { set i } else { goto i };
13:   x := x - 1;
14:   if x + 1 >= 0 then { goto i } else { skip }
15: end
```

Wyliczenie tej instrukcji w kolejnych linijkach spowoduje:

1: Wprowadzenie zmiennej skokowej i, o wartości równej skokowi do linii 15.

2: Wprowadzenie zmiennej indywiduowej x o wartości 0.

5: Wprowadzenie zmiennej skokowej k, o wartości równej skokowi do linii 10.

7: Zmniejszenie wartości x do -2.

8: Ustawienie wartości zmiennej skokowej k na skok do linii 9.

9: Nadanie zmiennej skokowej i wartości zmiennej skokowej k, czyli skoku do linii 9.

11: Zwiększenie wartości x do -1.

12: Sprawdzenie warunku `x >= 0` i wykonanie negatywnej gałęzi, czyli instrukcji skoku do wartości zmiennej skokowej i.

9: Wykonanie ponownie przypisania zmiennej skokowej i wartości zmiennej skokowej k, czyli skoku do linii 9.

11: Zwiększenie wartości x do 0.

15: Sprawdzenie warunku `x >= 0` i wykonanie pozytywnej gałęzi, czyli instrukcji `set i`, która nada zmiennej skokowej i wartość skoku bezpośrednio za tę instrukcję.

13: Zmniejszenie wartości x do -1.

14: Sprawdzenie warunku `x + 1 >= 0` i wykonanie pozytywnej gałęzi, czyli instrukcji skoku do wartości zmiennej skokowej i.

13: Zmniejszenie wartości x do -2.

14: Sprawdzenie warunku `x + 1 >= 0` i wykonanie negatywnej gałęzi, czyli instrukcji `skip`.

Rozwiązanie

Rozważmy następujące typy i funkcje pomocnicze:

$$\begin{aligned}
\mathbf{Jtore} &= \mathbf{Loc} \rightharpoonup_{\text{fin}} \mathbf{Cont}_S \\
\mathbf{Vtore} &= \mathbf{Loc} \rightharpoonup_{\text{fin}} \mathbf{Num} \\
\mathbf{JEnv} &= \mathbf{Label} \rightharpoonup \mathbf{Loc} \\
\mathbf{VEnv} &= \mathbf{Var} \rightharpoonup \mathbf{Loc} \\
\mathbf{Cont}_S &= \mathbf{Jtore} \rightharpoonup \mathbf{Vtore} \rightharpoonup \mathbf{Ans} \\
\mathbf{Cont}_D &= \mathbf{JEnv} \rightharpoonup \mathbf{VEnv} \rightharpoonup \mathbf{Jtore} \rightharpoonup \mathbf{Vtore} \rightharpoonup \mathbf{Ans} \\
&= \mathbf{JEnv} \rightharpoonup \mathbf{VEnv} \rightharpoonup \mathbf{Cont}_S \\
\text{newloc}_J &: \mathbf{Jtore} \to \mathbf{Loc} \\
\text{newloc}_V &: \mathbf{Vtore} \to \mathbf{Loc}
\end{aligned}$$

Używać będziemy funkcji semantycznych:

$$\begin{aligned}
\mathcal{E} &: \mathbf{Expr} \to \mathbf{VEnv} \rightharpoonup \mathbf{Vtore} \rightharpoonup \mathbf{Num} \\
\mathcal{D} &: \mathbf{Decl} \to \mathbf{Cont}_S \rightharpoonup \mathbf{Cont}_D \rightharpoonup \mathbf{Cont}_D \\
\mathcal{I} &: \mathbf{Inst} \to \mathbf{JEnv} \rightharpoonup \mathbf{VEnv} \rightharpoonup \mathbf{Cont}_S \rightharpoonup \mathbf{Cont}_S
\end{aligned}$$

Drugi argument funkcji semantycznej $\mathcal{D}$ to kontynuacja skacząca na koniec aktualnie rozważanego bloku.

Funkcja semantyczna $\mathcal{E}$ jest dana. Funkcję $\mathcal{D}$ zdefiniujemy następująco:

$$\begin{aligned}
\mathcal{D}[\![\mathbf{var}\ x = e]\!]\ \kappa_S\ \kappa_D = \lambda\rho_J.\ \lambda\rho_V.\ \lambda s_J.\ \lambda s_V.\ \kappa_D\ \rho_J\ \rho'_V\ s_J\ s'_V \qquad \text{where} \\
l = \text{newloc}_V(s_V) \\
q = \mathcal{E}[\![e]\!]\ \rho_V\ s_V \\
\rho'_V = \rho_V\big[x \mapsto l\big] \\
s'_V = s_V\big[l \mapsto q\big]
\end{aligned}$$

$$\begin{aligned}
\mathcal{D}[\![\mathbf{label}\ j]\!]\ \kappa_S\ \kappa_D = \lambda\rho_J.\ \lambda\rho_V.\ \lambda s_J.\ \lambda s_V.\ \kappa_D\ \rho'_J\ \rho_V\ s'_J\ s_V \qquad \text{where} \\
l = \text{newloc}_J(s_J) \\
\rho'_J = \rho_J\big[j \mapsto l\big] \\
s'_J = s_J\big[l \mapsto \kappa_S\big]
\end{aligned}$$

$$\mathcal{D}[\![d_1; d_2]\!]\ \kappa_S\ \kappa_D = \mathcal{D}[\![d_1]\!]\ \kappa_S\ (\mathcal{D}[\![d_2]\!]\ \kappa_S\ \kappa_D)$$

Podobnie, funkcja semantyczna $\mathcal{I}$ jest zdefiniowana następująco:

$$\mathcal{I}[\![\textbf{skip}]\!]\ \rho_J\ \rho_V\ \kappa_S = \kappa_S$$

$$\mathcal{I}[\![x := e]\!]\ \rho_J\ \rho_V\ \kappa_S = \lambda s_J.\ \lambda s_V.\ \kappa_S\ s_J\ s'_V \qquad \text{where}$$
$$l = \rho_V(x)$$
$$q = \mathcal{E}[\![e]\!]\ \rho_V\ s_V$$
$$s'_V = s_V\big[l \mapsto q\big]$$

$$\mathcal{I}[\![I_1; I_2]\!]\ \rho_J\ \rho_V\ \kappa_S = \mathcal{I}[\![I_1]\!]\ \rho_J\ \rho_V\ (\mathcal{I}[\![I_2]\!]\ \rho_J\ \rho_V\ \kappa_S)$$

$$\mathcal{I}[\![\textbf{if } e >= 0 \textbf{ then } \{I_1\} \textbf{ else } \{I_2\}]\!]\ \rho_J\ \rho_V\ \kappa_S = \lambda s_J.\ \lambda s_V.\ \text{if } \mathcal{E}[\![e]\!]\ \rho_V\ s_V \geq 0 \text{ then}$$
$$\mathcal{I}[\![I_1]\!]\ \rho_J\ \rho_V\ \kappa_S\ s_J\ s_V$$
$$\text{else}$$
$$\mathcal{I}[\![I_2]\!]\ \rho_J\ \rho_V\ \kappa_S\ s_J\ s_V$$

$$\mathcal{I}[\![\textbf{begin } d \textbf{ in } I \textbf{ end}]\!]\ \rho_J\ \rho_V\ \kappa_S = \mathcal{D}[\![d]\!]\ \kappa_S\ \big(\lambda\rho'_J.\ \lambda\rho'_V.\ \mathcal{I}[\![I]\!]\ \rho'_J\ \rho'_V\ \kappa_S\big)\ \rho_J\ \rho_V$$

$$\mathcal{I}[\![j_1 := j_2]\!]\ \rho_J\ \rho_V\ \kappa_S = \lambda s_J.\ \kappa_S\ s_J\big[\rho_J(j_1) \mapsto s_J(\rho_J(j_2))\big]$$

$$\mathcal{I}[\![\textbf{set } j]\!]\ \rho_J\ \rho_V\ \kappa_S = \lambda s_J.\ \kappa_S\ s_J\big[\rho_J(j) \mapsto \kappa_S\big]$$

$$\mathcal{I}[\![\textbf{goto } j]\!]\ \rho_J\ \rho_V\ \kappa_S = \lambda s_J.\ s_J\big(\rho_J(j)\big)\ s_J$$