Semantyka i weryfikacja programów 2022/23.
Egzamin 6/02/2023, zadanie 2 (semantyka denotacyjna)

Napisz semantykę denotacyjną dla języka o następującej gramatyce:

$$
\begin{array}{lcl}
\mathsf{Num} \ni n & ::= & \ldots -1 \mid 0 \mid 1 \mid \ldots \\
\mathsf{Var} \ni x & ::= & x_1 \mid x_2 \mid \ldots \\
\mathsf{VarP} \ni p & ::= & p_1 \mid p_2 \mid \ldots \\
\mathsf{Expr} \ni e & ::= & n \mid x \mid e_1 + e_2 \mid e_1 - e_2 \\
\mathsf{Block} \ni b & ::= & \texttt{begin}\ I\ \texttt{end} \\
\mathsf{Instr} \ni I & ::= & x := e \mid \texttt{var}\ x = e \mid I_1; I_2 \mid \texttt{skip} \mid \texttt{if}\ e = 0\ \texttt{then}\ I_1\ \texttt{else}\ I_2 \mid \\
 & & b \mid \texttt{while}\ e = 0\ \texttt{do}\ b \mid \texttt{call}\ p(e) \mid \texttt{proc}\ p(x)\ b
\end{array}
$$

W języku tym zakładamy, że wszystkie instrukcje wykonują się w jakimś bloku określonym za pomocą kategorii syntaktycznej Block. Deklaracje zmiennych indywiduowych (`var` x = e) są rozumiane statycznie w ramach bloków z tej kategorii. Przy czym deklaracja takiej zmiennej widoczna jest do końca najbardziej wewnętrznego bloku, w którym się znajduje, ale poza zakresem widoczności później wykonanych deklaracji zmiennej o tej samej nazwie w tym samym bloku. W związku z tym, że ciało pętli jest blokiem, oznacza to, iż wartość zmiennej zadeklarowanej w ciele pętli nie może bezpośrednio na tej zmiennej przetrwać między różnymi obrotami pętli.

W języku tym niestandardowo traktowane są deklaracje procedur (`proc` $p(x)$ b). Inaczej niż zwykle wykonywane są one dynamicznie. Instrukcja deklaracji procedury `proc` $p(x)$ b ma po dojściu sterowania do niej spowodować wykonanie bloku b, przy czym identyfikator x w kodzie b ma odpowiadać zmiennej, jaka jest widoczna bezpośrednio przed deklaracją. Oprócz wykonania b taka deklaracja wprowadza pod identyfikatorem p procedurę o kodzie b ze statyczną widocznością identyfikatorów i o parametrze formalnym x z przekazywaniem parametru przez wartość. Tak wprowadzona procedura jest widoczna od miejsca deklaracji do końca działania programu lub do następnego wykonania deklaracji procedury o tym samym identyfikatorze, w tym jest ona widoczna rekurencyjnie w kodzie tej procedury również w pierwszym wykonaniu bloku b.

Znaczenie pozostałych konstrukcji jest standardowe. Należy założyć, że identyfikatory zmiennych i procedur pochodzą z rozłącznych zbiorów (tzn. $\mathsf{Var} \cap \mathsf{VarP} = \emptyset$). Trzeba podać typy funkcji semantycznych dla wyrażeń, bloków i instrukcji, a także definicje dla funkcji semantycznych dla bloków i instrukcji. Należy też zdefiniować wszystkie dziedziny semantyczne, z których korzysta się przy podawaniu semantyki. Jeśli trzeba, można wykorzystać funkcję alloc odpowiedniego typu (ten typ należy podać), która określać będzie dotychczas niewykorzystywaną lokację.

Przykład:

```
01: begin
02:  var x = 0;
03:  var y = 0;
04:  while x = 0 do begin
05:    var z = 3;
06:    proc p(x) begin
07:      z := z + y + x;
08:      if z - 3 = 0
09:        then call p(1)
10:        else y := z + 1
11:    end;
12:    if y - 10 = 0 then
13:      x := 1
14:    else
15:      y := y + 1
16:  end;
17:  call p(3)
18: end
```

W wyniku działania tego programu wartością zmiennej `y` tuż przed wyjściem z bloku w wierszu `18`. będzie 23. Wykonanie kodu wygląda tak: najpierw następuje inicjalizacja `x` oraz `y` na 0. Następnie w wierszu `04`. wchodzimy do pętli `while` z warunkiem wejścia `x = 0`, który jest spełniony. Uruchamiamy zatem ciało pętli, co prowadzi co zadeklarowania zmiennej `z` i nadania jej wartości 3. Następnie wchodzimy do deklaracji procedury `p`, która w swoim bloku najpierw określi wartość zmiennej `z` jako sumę równą 3. Wykonanie następnej instrukcji `if` spowoduje wybranie gałęzi `then` i rekurencyjne wywołanie procedury `p` zadeklarowanej w wierszu `06`. Tym razem zmiena `z` zostanie określona jako suma równa 4 i w następnej instrukcji `if` zostanie wybrana gałąź `else`, gdzie wartość zmiennej `y` zostanie określona jako suma równa 5. W dalszym ciągu nastąpi wyjście z wywołania `p`. Następująca po deklaracji `p` instrukcja warunkowa w wierszu `12`. sprawdzi, że `y` równe 5 nie spełnia jej warunku, więc wykona przypisanie z wiersza `15`., co nada zmiennej `y` wartość 6. Następnie przystąpimy do kolejnego sprawdzenia warunku wejściowego pętli. Zmienna `x` nie zmieniła wartości, więc warunek pozostanie prawdziwy i znowu wejdziemy do wnętrza pętli, ponownie zadeklarujemy `z` i `p`. Wykonanie w ramach deklaracji ciała `p` spowoduje określenie wartości zmiennej `z` jako 3+6+0=9 i zmiennej `y` jako 10. Tym razem warunek instrukcji warunkowej w wierszu `12`. będzie prawdziwy, więc wykona się przypisanie z wiersza `13`. To doprowadzi do wyjścia z pętli, po którym nastąpi wywołanie w wierszu `17`. procedury `p` z parametrem aktualnym 3. Wartość zmiennej `z` w treści procedury `p` zostanie określona jako suma 9+10+3=22, warunek `if` nie będzie spełniony i nastąpi przypisanie na zmienną `y` wartości 23, wyjście z procedury `p` i wyjście z bloku w wierszu `18`.

Możliwe rozwiązanie:

Dziedziny Podstawowe dziedziny semantyczne mają tutaj następującą postać:

- $\mathsf{Val} = \mathbb{Z}_\perp$,
- $\mathsf{SVal} = \mathsf{Val} \cup \mathsf{Proc}$,
- $\mathsf{State} = (\mathsf{Loc}_\perp \uplus \mathsf{VarP}) \to \mathsf{SVal}$,
- $\mathsf{Proc} = \mathsf{Val} \to \mathsf{State} \to \mathsf{State}$,

Określimy środowisko:

$$\mathsf{Env} = \quad \mathsf{Var} \to \mathsf{Loc}_\perp$$

Funkcja alloc ma następujący typ: $\mathrm{alloc} : \mathsf{State} \to \mathsf{State} \times \mathsf{Loc}$.

Typy funkcji semantycznych

- $\mathcal{E} : \mathsf{Expr} \to \mathsf{Env} \to \mathsf{State} \to \mathsf{Val}$,
- $\mathcal{I} : \mathsf{Instr} \to \mathsf{Env} \to \mathsf{State} \to (\mathsf{Env} \times \mathsf{State})$.
- $\mathcal{B} : \mathsf{Block} \to \mathsf{Env} \to \mathsf{State} \to (\mathsf{Env} \times \mathsf{State})$.

Denotacja dla programów Zakładamy, że dla programu, który składa się z instrukcji I, denotacją jest $\mathcal{I}[\![I]\!]\,\rho\,s_0$, gdzie $\rho = \lambda x \in \mathsf{Var}.\perp$, $s_0 = \lambda l \in \mathsf{Loc}.\perp$.

Denotacja bloków

- $\mathcal{B}[\![\texttt{begin}\ I\ \texttt{end}]\!] = \lambda\rho \in \mathsf{Env}.\lambda s \in \mathsf{State}.(\rho, s')$
  gdzie $(\rho', s') = \mathcal{I}[\![I]\!]\,\rho\,s$

Denotacja instrukcji

- $\mathcal{I}[\![x := e]\!] = \lambda\rho \in \mathsf{Env}.\lambda s \in \mathsf{State}.\mathit{ifte}(\rho\,x \in \mathsf{Loc}, (\rho, s[\rho\,x \mapsto \mathcal{E}[\![e]\!]\,\rho\,s]), (\rho, \perp))$
- $\mathcal{I}[\![\texttt{var}\ x = e]\!] = \lambda\rho \in \mathsf{Env}.\lambda s \in \mathsf{State}.(\rho[x \mapsto l], s'[l \mapsto \mathcal{E}[\![e]\!]\,\rho\,s])$, gdzie $(s', l) = \mathrm{alloc}(s)$
- $\mathcal{I}[\![I_1;\ I_2]\!] = \lambda\rho \in \mathsf{Env}.\lambda s \in \mathsf{State}.\mathcal{I}[\![I_2]\!]\,\rho'\,s'$, gdzie $(\rho', s') = \mathcal{I}[\![I_1]\!]\,\rho\,s$
- $\mathcal{I}[\![\texttt{skip}]\!] = \lambda\rho \in \mathsf{Env}.\lambda s \in \mathsf{State}.(\rho, s)$
- $\mathcal{I}[\![\texttt{if}\ e = 0\ \texttt{then}\ I_1\ \texttt{else}\ I_2]\!] = \lambda\rho \in \mathsf{Env}.\lambda s \in \mathsf{State}.\mathit{ifte}(n =_{\mathsf{Val}} 0, \mathcal{I}[\![I_1]\!], \mathcal{I}[\![I_2]\!])\,\rho\,s$,
  gdzie $n = \mathcal{E}[\![e]\!]\,\rho\,s$
- $\mathcal{I}[\![b]\!] = \lambda\rho \in \mathsf{Env}.\lambda s \in \mathsf{State}.\mathcal{B}[\![b]\!]\,\rho\,s$,
- $\mathcal{I}[\![\texttt{while}\ e = 0\ \texttt{do}\ b]\!] = \lambda\rho \in \mathsf{Env}.\lambda s \in \mathsf{State}.(\mathsf{fix}\,F)(\rho, s)$,
  gdzie $F = \lambda\varphi\colon (\mathsf{Env} \times \mathsf{State}) \to \mathsf{Env} \times \mathsf{State}.\ \lambda(\rho', s') \in \mathsf{Env} \times \mathsf{State}.$
  $\mathit{ifte}(\mathcal{E}[\![e]\!]\,\rho'\,s' =_{\mathsf{Val}} 0, \quad \varphi(\mathcal{B}[\![b]\!]\,\rho'\,s'), \quad (\rho', s'))$
- $\mathcal{I}[\![\texttt{call}\ p(e)]\!] = \lambda\rho \in \mathsf{Env}.\lambda s \in \mathsf{State}.\mathit{ifte}(s\,p \in \mathsf{Proc}, (\rho, s\,p\,n\,s), (\rho, \perp))$,
  gdzie $n = \mathcal{E}[\![e]\!]\rho\,s$,
- $\mathcal{I}[\![\texttt{proc}\ p(x)\ b]\!] = \lambda\rho \in \mathsf{Env}.\lambda s \in \mathsf{State}.\mathcal{B}[\![b]\!]\,\rho\,s'$,
  gdzie $s' = s[p \mapsto P]$
  $P = \lambda n \in \mathsf{Val}.\lambda s'' \in \mathsf{State}.\mathcal{B}[\![b]\!]\,\rho'\,\hat{s}'$,
  przy czym $(\hat{s}, l) = \mathrm{alloc}(s''), \rho' = \rho[x \mapsto l], \hat{s}' = \hat{s}[l \mapsto n]$