Semantyka i weryfikacja programów 2023/24.
Egzamin 24/02/2024, zadanie 1 (semantyka operacyjna)

Napisz semantykę operacyjną dla języka ze *statycznymi zmiennymi lokalnymi* o następującej gramatyce:

$$
\begin{array}{lcl}
\mathsf{Num} \ni n & ::= & \ldots -\mathtt{1} \mid \mathtt{0} \mid \mathtt{1} \mid \ldots \\
\mathsf{Var} \ni x & ::= & x_1 \mid x_2 \mid \ldots \\
\mathsf{VarP} \ni p & ::= & p_1 \mid p_2 \mid \ldots \\
\mathsf{Expr} \ni e & ::= & n \mid x \mid e_1 + e_2 \mid e_1 - e_2 \\
\mathsf{Dec} \ni d & ::= & \mathtt{var}\ x = e \mid \mathtt{proc}\ p(\mathtt{var}\ x = e)\ I \mid d_1; d_2 \\
\mathsf{Instr} \ni I & ::= & x := e \mid I_1;\ I_2 \mid \mathtt{skip} \mid \mathtt{if}\ e\ \mathtt{<>}\ 0\ \mathtt{then}\ I_1\ \mathtt{else}\ I_2 \mid \\
 & & \mathtt{begin}\ d; I\ \mathtt{end} \mid \mathtt{while}\ e\ \mathtt{<>}\ 0\ \mathtt{do}\ I \mid \mathtt{call}\ p(e) \mid \mathtt{call}\ p() \\
\mathsf{Prog} \ni P & ::= & \mathtt{prog}\ I
\end{array}
$$

W języku tym procedury są rekurencyjne, wiązanie identyfiktatorów wolnych w ciele procedury jest statyczne.

Niestandardowo działa tutaj przekazywanie parametru do procedury. Deklaracja postaci

$$\mathtt{proc}\ p(\mathtt{var}\ x = e)\ I$$

oznacza, że kod procedury p będzie wykonywany z parametrem formalnym x, który jako *statyczna zmienna lokalna* w momencie deklaracji zostanie zainicjowany wartością wyrażenia e, a następnie będzie funkcjonował w programie jak zmienna wspólna dla wszystkich wywołań tej procedury, z bieżącą wartością zachowywaną pomiędzy kolejnymi jej wywołaniami. Zmienna ta może być wykorzystywana jak „zwykłe" zmienne (wprowadzane deklaracjami zmiennych), ale jedynie w wywołaniach oraz w kodzie procedury p (nie jest widoczna poza ciałem tej procedury). W szczególności każde wywołanie powyższej procedury postaci

$$\mathtt{call}\ p()$$

oznacza, że zmienna x na wejściu do kodu procedury będzie miała swą dotychczasową wartość bieżącą (taką, jak w momencie, gdy sterowanie opuszczało kod ostatniego czynnego wywołania p lub, przy pierwszym wywołaniu tej procedury, wartość nadaną jej przy inicjalizacji w deklaracji procedury). Natomiast wywołanie postaci

$$\mathtt{call}\ p(e)$$

spowoduje, że obliczona w momencie wywołania wartość wyrażenia e zostanie tuż przed wykonaniem ciała procedury przypisana na widoczną w ciele statyczną zmienną lokalną x.

Należy założyć, że identyfikatory zmiennych i procedur pochodzą z rozłącznych zbiorów (tzn. $\mathsf{Var} \cap \mathsf{VarP} = \emptyset$). Również należy przyjąć, że różne deklaracje procedur wprowadzają różne statyczne zmienne lokalne, nawet jeśli ich nazwy w tych różnych deklaracjach się pokrywają.

W rozwiązaniu należy zdefiniować zbiór konfiguracji oraz podać reguły przejścia dla instrukcji oraz deklaracji.

Można przyjąć, że dana jest pomocnicza funkcja semantyczna dla wyrażeń:

- $\mathcal{E} : \mathsf{Expr} \to \mathsf{Env} \to \mathsf{Store} \to \mathsf{Num}$

gdzie, jak zwykle, Num to zbiór liczb całkowitych. Można też przyjąć, że dana jest funkcja $\mathrm{alloc} : \mathsf{Store} \to \mathsf{Store} \times \mathsf{Loc}$ alokująca nieużywaną lokację.

verté

Przykład 1

```
01: prog
02: begin
03:   var x = 2;
04:   var y = 3;
05:   proc p(var x = 4)
06:     begin
07        var z = 1;
08:       x := x + y;
09:       y := x + z;
10:     end
11:   proc q(var y = 5)
12:     y := y + x;
13:   call p();
14:   call p(x);
15:   call q(x);
16:   call q();
17:   call q(y);
18:   call p(y);
19: end
```

W wyniku działania tego programu tuż przed wyjściem z bloku `begin...end` w wierszu `19` wartość zmiennej x będzie 2, zaś wartość zmiennej y będzie 23. Statyczna zmienna lokalna x z procedury p będzie miała wartość 22, zaś statyczna zmienna lokalna y z procedury q będzie miała wartość 13. Zobaczmy, jak do tego dojdzie. Po wywołaniu procedury p w wierszu `13` zmienna globalna x nie zmieni wartości, zaś zmienna y przybierze wartość 8. Natomiast statyczna zmienna lokalna x z procedury p przybierze wartość 7. Następnie po wywołaniu procedury p w wierszu `14` zmienna globalna x nie zmieni wartości, zaś zmienna y przybierze wartość 11. Natomiast statyczna zmienna lokalna x z procedury p przybierze wartość 10. Dalej wywołanie procedury q w wierszu `16` nie spowoduje zmiany zmiennych globalnych. Jednak zmieni się wartość statycznej zmiennej lokalnej y z procedury q z 4 na 6. Wywołanie procedury q w wierszu `17` nie spowoduje zmiany zmiennych globalnych. Jednak ponownie zmieni się wartość statycznej zmiennej lokalnej y tym razem na 13. Wreszcie wywołanie procedury p w wierszu `18` spowoduje, że zmienna y przybierze wartość 23, zaś statyczna zmienna lokalna x z procedury p przybierze wartość 22.

Rozwiązanie:

Semantyka naturalna (duże kroki):
Konfiguracje końcowe: $\mathsf{T} = \mathsf{Store}$.
Konfiguracje: $\Gamma = (\mathsf{Instr} \times \mathsf{Env} \times \mathsf{Store}) \cup \mathsf{T}$.
Konfiguracje deklaracji: $(\mathsf{Dec} \times \mathsf{Env} \times \mathsf{Store}) \cup (\mathsf{Env} \times \mathsf{Store})$

- Środowiska są postaci $\rho \in \mathsf{Env}_n = \mathsf{EnvV} \times \mathsf{EnvP}_n$, gdzie
  - $\mathsf{EnvV} = \mathsf{Var} \rightharpoonup \mathsf{Loc}$
  - $\mathsf{EnvP}_n = \mathsf{VarP} \rightharpoonup \mathsf{Proc}_{n-1}$
- Dla $\rho = \langle \rho_v, \rho_p \rangle \in \mathsf{Env}$ przyjmujemy notacje:
  - $\rho[x \mapsto l] = \langle \rho_v[x \mapsto l], \rho_p \rangle$ oraz $\rho(x) = \rho_v(x)$, gdy $x \in \mathsf{Var}, l \in \mathsf{Loc}$,
  - $\rho[p \mapsto P] = \langle \rho_v, \rho_p[p \mapsto P] \rangle$ oraz $\rho(p) = \rho_p(p)$, gdy $p \in \mathsf{VarP}, P \in \mathsf{Proc}$.

  Zakładamy tutaj, że $\mathsf{Loc} \cap \mathbb{Z} = \emptyset$.
- $\mathsf{Store} = \mathsf{Loc} \rightharpoonup \mathbb{Z}$
- $\mathsf{Proc}_n = \mathsf{Var} \times \mathsf{Loc} \times \mathsf{Env}_{n-1} \times \mathsf{Instr}$
- $\mathsf{EnvP}_0 = \{\emptyset\}$
- $\mathsf{Env} = \bigcup_{i \in \mathbb{N}} \mathsf{Env}_i$
- $\mathsf{Proc} = \bigcup_{i \in \mathbb{N} - \{0\}} \mathsf{Proc}_i$

Niestandardowa dziedzina dla procedur pozwala na przechowywanie informacji o umiejscowieniu w składzie wartości lokalnej zmiennej statycznej związanej z procedurą.
Relacja "dojścia" ("dużych kroków") dla programów:

$$\rightsquigarrow_P \subseteq \mathsf{Prog} \times \mathsf{T}$$

Relacja "dojścia" ("dużych kroków") dla instrukcji:

$$\rightsquigarrow \subseteq (\mathsf{Instr} \times \mathsf{Env} \times \mathsf{Store}) \times \mathsf{T}$$

Relacja "dojścia" ("dużych kroków") dla deklaracji:

$$\rightsquigarrow_d \subseteq (\mathsf{Dec} \times \mathsf{Env} \times \mathsf{Store}) \times (\mathsf{Env} \times \mathsf{Store})$$

Reguły:

Reguły dla programu:

$$\frac{I, \emptyset, \emptyset \rightsquigarrow s}{\texttt{prog}\ I, \rightsquigarrow_P s}$$

Standardowe reguły dla zwykłych obliczeń:

$$\frac{}{x{:=}e, \rho, s \rightsquigarrow s[\rho(x) \mapsto \mathcal{E}[\![E]\!]\,\rho\,s]} \qquad \frac{}{\texttt{skip}, \rho, s \rightsquigarrow s}$$

$$\frac{I_1, \rho, s \rightsquigarrow s' \quad I_2, \rho, s' \rightsquigarrow s''}{I_1; I_2, \rho, s \rightsquigarrow s''}$$

$$\frac{d, \rho, s \rightsquigarrow_d \rho', s' \quad I, \rho', s' \rightsquigarrow s''}{\texttt{begin}\ d; I\ \texttt{end}, \rho, s \rightsquigarrow s''}$$

$$\frac{\begin{array}{c}\rho(p) = \langle x, l, \rho', I \rangle \\ I, \rho'[x \mapsto l][p \mapsto \rho(p)], s \rightsquigarrow s'\end{array}}{\texttt{call}\ p(), \rho, s \rightsquigarrow s'}$$

$$\frac{\begin{array}{c}\mathcal{E}[\![e]\!]\, \rho\, s = v \quad \rho(p) = \langle x, l, \rho', I \rangle \\ I, \rho'[x \mapsto l][p \mapsto \rho(p)], s'[l \mapsto v] \rightsquigarrow s'\end{array}}{\texttt{call}\ p(e), \rho, s \rightsquigarrow s'}$$

Reguły dla while:

$$\frac{\mathcal{E}[\![e]\!]\, \rho\, s = 0}{\texttt{while}\ e\ \texttt{<>0}\ \texttt{do}\ I, \rho, s \rightsquigarrow s}$$

$$\frac{\begin{array}{c}\mathcal{E}[\![e]\!]\, \rho\, s \neq 0 \quad I, \rho, s \rightsquigarrow s' \\ \texttt{while}\ e\ \texttt{<>0}\ \texttt{do}\ I, \rho, s' \rightsquigarrow s''\end{array}}{\texttt{while}\ e\ \texttt{<>0}\ \texttt{do}\ I, \rho, s \rightsquigarrow s''}$$

Reguły dla if:

$$\frac{I_1, \rho, s \rightsquigarrow s' \quad \mathcal{E}[\![e]\!]\, \rho\, s \neq 0}{\texttt{if}\ e\ \texttt{<>0}\ \texttt{then}\ I_1\ \texttt{else}\ I_2, \rho, s \rightsquigarrow s'}$$

$$\frac{I_2, \rho, s \rightsquigarrow s' \quad \mathcal{E}[\![e]\!]\, \rho\, s = 0}{\texttt{if}\ e\ \texttt{<>0}\ \texttt{then}\ I_1\ \texttt{else}\ I_2, \rho, s \rightsquigarrow s'}$$

Reguły dla deklaracji:

$$\frac{\mathrm{alloc}(s) = \langle s', l \rangle \quad \mathcal{E}[\![e]\!]\, \rho\, s = v}{\texttt{var}\ x = e, \rho, s \rightsquigarrow_d \rho[x \mapsto l], s'[l \mapsto v]}$$

$$\frac{\mathrm{alloc}(s) = \langle s', l \rangle \quad \mathcal{E}[\![e]\!]\, \rho\, s = v}{\texttt{proc}\ p(\texttt{var}\ x = e)\ I, \rho, s \rightsquigarrow_d \rho[p \mapsto \langle x, l, \rho, I \rangle], s'[l \mapsto v]}$$

$$\frac{d_1, \rho, s \rightsquigarrow_d \rho', s' \quad d_2, \rho', s' \rightsquigarrow_d \rho'', s''}{d_1\,;d_2, \rho, s \rightsquigarrow_d \rho'', s''}$$