Semantyka i weryfikacja programów
zadanie 2 (semantyka denotacyjna)
Egzamin poprawkowy 22/02/2022

Napisz semantykę denotacyjną w stylu kontynuacyjnym dla deklaracji, instrukcji i wyrażeń języka o gramatyce:

$$
\begin{aligned}
\mathit{Num} \ni n &::= \texttt{0} \mid \texttt{1} \mid -\texttt{1} \mid \texttt{2} \mid -\texttt{2} \mid \cdots \\
\mathit{Bool} \ni t &::= \texttt{true} \mid \texttt{false} \\
\mathit{Var} \ni x &::= \texttt{x} \mid \texttt{y} \mid \cdots \\
\mathit{FName} \ni p &::= \texttt{f} \mid \texttt{g} \mid \cdots \\
\mathit{Expr} \ni e &::= n \mid x \mid f() \mid e_1 + e_2 \mid e_1 * e_2 \mid e_1 - e_2 \\
\mathit{BExpr} \ni b &::= \texttt{true} \mid \texttt{false} \mid e_1 < e_2 \mid e_1 = e_2 \mid b_1 \wedge b_2 \mid \texttt{not}\ b \\
\mathit{Decl} \ni d &::= \texttt{var}\ x = e \mid \texttt{fun}\ f\ \texttt{do}\ (I)\ \texttt{upon return do}\ (J) \mid d_1; d_2 \\
\mathit{Instr} \ni I &::= x := e \mid I_1; I_2 \mid \texttt{if}\ b\ \texttt{then}\ I_1\ \texttt{else}\ I_2 \mid \texttt{while}\ b\ \texttt{do}\ I \mid \texttt{begin}\ d; I\ \texttt{end} \mid \texttt{return}\ e
\end{aligned}
$$

Jest to język z rekurencyjnymi funkcjami bezparametrowymi oraz z mechanizmem wyłapywania momentów wyjścia z funkcji. Zmienne oraz funkcje deklarowane są w blokach `begin` $d; I$ `end`; widoczność zmiennych oraz funkcji jest statyczna. Wyliczanie wyrażeń odbywa się od lewej do prawej (co ma znaczenie, gdyż funkcje mogą mieć efekty uboczne).

Wywołanie $f()$ funkcji wprowadzonej deklaracją `fun` f `do` (I) `upon return do` (J) realizowane jest przez wykonanie instrukcji I (ciała funkcji).

- Instrukcja `return` e w czasie obliczania ciała I funkcji f kończy to obliczenie, a w bieżącym stanie obliczana jest wartość q wyrażenia e, po czym wykonywana jest instrukcja J. Jeżeli instrukcja J nie zakończyła się wykonaniem kolejnego `return` e' to obliczenie wywołania $f()$ kończy się z wynikiem q. Jeżeli w trakcie wykonywania instrukcji J natrafimy na instrukcję `return` e' to obliczenie instrukcji J jest przerywane i wyliczana jest wartość q' wyrażenia e'. W tym przypadku obliczenie wywołania $f()$ kończy się z wynikiem q'. Wszystkie poczynione w trakcie powyższych obliczeń zmiany stanu są zachowywane.
- Jeśli wykonanie ciała I zakończyło się bez wykonania instrukcji `return` to wynikiem wywołania $f()$ jest wartość 0 i instrukcja J **nie jest** uruchamiana.

Rekurencja sprowadza się do tego, że funkcja f powinna być widoczna w swoim ciele I. Nie dopuszczamy natomiast rekurencyjnych wywołań f z wnętrza instrukcji J.

Semantyka pozostałych konstrukcji jest standardowa.

Na przykład, w instrukcji:

```
begin
  var x = 10;
  fun f do (x := x + 1) upon return do (x := x + 100);
  fun g do (return f()) upon return do (x := f() + x);
  begin
    fun g do (return x) upon return do (return g()+1);

    x := g() + x
  end
end
```

przed wyjściem z zewnętrznego bloku zmienna x przyjmuje wartość 13, gdyż obliczenie `x := g() + x` spowoduje po kolei:

1. wywołanie (drugiej, z wewnętrznego bloku) funkcji g, której ciało to `return x`, co powoduje zapamiętanie wartości $q = 10$,
2. $>$ wykonanie instrukcji `return g()+1`,
3. $>>$ wywołanie (pierwszej, z zewnętrznego bloku!) funkcji g, której ciało to `return f()`,

4. >>> wywołanie funkcji f, której ciało to `x := x + 1`, więc po jego wykonaniu zmienna x przyjmie wartość 11, zaś wartością wywołania tej funkcji jest 0,
5. >> wyjście z wykonania ciała (pierwszej, z zewnętrznego bloku) funkcji g z podaną przez `return` wartością 0, co spowoduje wykonanie instrukcji `x := f() + x`,
6. >>> wywołanie funkcji f, której ciało to `x := x + 1`, więc po jego wykonaniu zmienna x przyjmie wartość 12, zaś wartością wywołania tej funkcji jest 0,
7. >> przypisanie zmiennej x wartości $0 + 12 = 12$ (wartość wywołania $f()$ plus aktualna wartość x),
8. > wartość wywołania $g()$ (pierwszej z funkcji g, z zewnętrznego bloku) to 0, zaś wartość wyrażenia `g()+1` to 1,
9. tę ostatnią wartość przyjmuje wywołanie funkcji $g()$ (drugiej z funkcji g, z wewnętrznego bloku), więc wartość wyrażenia `g()+x` w instrukcji przypisania wynosi 13, bo zmienna x ma obecnie wartość 12,
10. i taka wartość jest ostatecznie przypisywana zmiennej x.

W rozwiązaniu należy podać:

- wszystkie potrzebne dziedziny pomocnicze i typy wszystkich czterech funkcji semantycznych,
- równania semantyczne dla wyrażeń postaci `x` oraz `f()`,
- równania semantyczne dla instrukcji postaci `return` e,
- równania semantyczne dla deklaracji postaci `var` $x = e$ oraz `fun` f `do` (I) `upon return do` (J).

Pozostałe równania można pominąć, jeżeli są standardowe, zupełnie analogiczne do standardowych lub bardzo podobne do już napisanych (np. wyrażenia logiczne) – zaznaczając to w rozwiązaniu i opisując ewentualne potrzebne ich modyfikacje.

Można założyć, że programy nie korzystają z niezadeklarowanych funkcji. Można też założyć, że instrukcja `return` e może wystąpić tylko w jakiejś funkcji: jej ciele I lub odpowiedniej instrukcji J. Można też przyjąć istnienie funkcji matematycznej *newloc* (obliczającej nową, nie używaną do tej pory lokację) o wybranym przez siebie typie (należy go explicite podać).