SEMANTYKA I WERYFIKACJA - ĆW. 11

Kontynuujemy kontynuacje.

1. Zróbmy semantykę funkcji (jeden parametr, przekazywanie przez zmienną). Tym razem niech identyfikatory funkcji i zmiennych będą rozróżniane syntaktycznie (dziedziny syntaktyczne **Var** i **FName**).

$$
\begin{aligned}
e &::= \cdots \mid f(x) \\
I &::= \cdots \mid \texttt{begin}\ d;\ I\ \texttt{end} \mid \texttt{return}\ e \\
d &::= \texttt{var}\ x = e \mid \texttt{fun}\ f(x)\ I \mid d_1; d_2
\end{aligned}
$$

Uwaga, wyrażenia mogą teraz mieć efekty uboczne, więc kontynuacje wyrażeń muszą mieć bogatszą semantykę. Wprowadzamy też kontynuacje deklaracji.

$$
\begin{aligned}
\mathbf{Cont}_E &= \mathbf{Num} \to \mathbf{Store} \to \mathbf{Ans} \\
\mathbf{Cont}_B &= \mathbf{Bool} \to \mathbf{Store} \to \mathbf{Ans} \\
\mathbf{Cont}_D &= \mathbf{Env} \to \mathbf{Store} \to \mathbf{Ans}
\end{aligned}
$$

Poza tym środowisko instrukcji musi jeszcze dodatkowo przechowywać kontynuację wyrażenia do wyskakiwania przez `return`:

$$
\begin{aligned}
\mathcal{I}[\![]\!] &: \mathbf{Exp} \to \mathbf{Env} \to \mathbf{Cont}_E \to \mathbf{Cont} \to \mathbf{Store} \to \mathbf{Ans} \\
\mathcal{E}[\![]\!] &: \mathbf{Exp} \to \mathbf{Env} \to \mathbf{Cont}_E \to \mathbf{Store} \to \mathbf{Ans} \\
\mathcal{B}[\![]\!] &: \mathbf{BExp} \to \mathbf{Env} \to \mathbf{Cont}_B \to \mathbf{Store} \to \mathbf{Ans} \\
\mathcal{D}[\![]\!] &: \mathbf{Decl} \to \mathbf{Env} \to \mathbf{Cont}_D \to \mathbf{Store} \to \mathbf{Ans}
\end{aligned}
$$

Procedury możemy pamiętać w środowisku, przy czym semantyka procedury jest kontynuacyjna:

$$
\begin{aligned}
\mathbf{Env} &= (\mathbf{Var} \to \mathbf{Loc}) \times (\mathbf{FName} \to \mathbf{Fun}) \\
\mathbf{Fun} &= \mathbf{Loc} \to \mathbf{Cont}_E \to \mathbf{Store} \to \mathbf{Ans}
\end{aligned}
$$

Równania dla deklaracji:

$$
\begin{aligned}
\mathcal{D}[\![\texttt{var}\ x = e]\!] &= \lambda\rho.\lambda\kappa.\mathcal{E}[\![e]\!]\rho(\lambda n.\lambda s'.\kappa(\rho[x \mapsto l])(s'[l \mapsto n])) \\
&\qquad \text{gdzie } l = \texttt{new}(s') \\
\mathcal{D}[\![\texttt{fun}\ f(x)\ I]\!] &= \lambda\rho\lambda\kappa.\kappa(\rho[f \mapsto P]) \\
&\qquad \text{gdzie } P = \lambda l.\lambda\beta.\mathcal{I}[\![I]\!](\rho[x \mapsto l])\beta(\beta(0)) \\
\mathcal{D}[\![d_1; d_2]\!] &= \lambda\rho\lambda\kappa.\mathcal{D}[\![d_1]\!]\rho(\lambda\rho'.\mathcal{D}[\![d_2]\!]\rho'\kappa)
\end{aligned}
$$

Zwróćmy uwagę, jak różnie są traktowane deklaracje zmiennych (kontrola jest przekazywana wyrażeniu e) i funkcji (kontrola wraca do kontynuacji, bez wykonywanie instrukcji I). Poza tym, w funkcji P są przechowywane

dwie kontynuacje dla I: jedna do wyskakiwania, jedna do zwykłego zakończenia, przy czym ustaliliśmy, że domyślnie zwracaną wartością funkcji jest 0.

Równania dla instrukcji (uwaga, dwie kontynuacje do wyboru!):

$$\mathcal{I}[\![\texttt{begin } d; I \texttt{ end}]\!] = \lambda\rho.\lambda\beta.\lambda\kappa.\lambda s.\mathcal{D}[\![d]\!]\rho(\lambda\rho'.\mathcal{I}[\![I]\!]\rho'\beta\kappa)s$$
$$\mathcal{I}[\![\texttt{return } e]\!] = \lambda\rho.\lambda\beta.\lambda\kappa.\mathcal{E}[\![e]\!]\rho\beta$$

W drugim równaniu kontynuacja κ jest ignorowana.

Wreszcie równanie dla wywołania funkcji:

$$\mathcal{E}[\![f(x)]\!] = \lambda\rho.(\rho(f))(\rho(x))$$

Cała reszta argumentów się η-redukuje.

2. Wyjątki:

$$I ::= \cdots \mid \texttt{try } I_1 \texttt{ catch } exn\ I_2 \mid \texttt{throw } exn$$

Rozwiązanie: do środowiska dorzucamy mapę z nazw wyjątków w kontynuacje:

$$\mathbf{Env} = (\mathbf{Var} \to \mathbf{Loc}) \times (\mathbf{FName} \to \mathbf{Fun}) \times (\mathbf{Exn} \to \mathbf{Cont})$$

Poza tym dziedziny i typy funkcji semantycznych się nie zmieniają. Nowe równania:

$$\mathcal{I}[\![\texttt{try } I_1 \texttt{ catch } exn\ I_2]\!] = \lambda\rho.\lambda\beta.\lambda\kappa.\mathcal{I}[\![I_1]\!](\rho[exn \mapsto \mathcal{I}[\![I_2]\!]\rho\beta\kappa])\beta\kappa$$
$$\mathcal{I}[\![\texttt{throw } exn]\!] = \lambda\rho.\lambda\beta.\lambda\kappa.\rho(x)$$