# SEMANTYKA I WERYFIKACJA - ĆW. 10

Zaczynamy kontynuacje. Zaczniemy od języka TINY z blokami, ale bez deklaracji procedur (czyli jedno środowisko **Env**).

1. Niech **Ans** będzie dziedziną "ostatecznych wyników działania programu". W dużym stopniu wszystko jedno co to jest; można przyjąć **Ans** = **Store**.

   Dziedziny kontynuacji:

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

   **Uwaga:** na slajdach z wykładu kontynuacje wyrażeń biorą dodatkowo stan. Tutaj to pomijamy; można sobie samemu porównać.

   Normalny typ funkcji semantycznej dla instrukcji jest taki:

   $$\mathcal{I}[\![]\!] : \mathbf{Instr} \to \mathbf{Env} \to \mathbf{Store} \to \mathbf{Store}$$

   Idea: przekazujemy element **Store** "dalej", do kontekstu. Potem ten kontekst oblicza z tego element **Ans**. Ale równie dobrze możemy uznać, że z kontekstu dostajemy funkcję **Store** $\to$ **Ans** i sami ją sobie uruchomimy. Możemy ją też np. zignorować, co potem pozwoli na większą elastyczność. Tak więc typy funkcji semantycznych:

   $$\mathcal{I}[\![]\!] : \mathbf{Instr} \to \mathbf{Env} \to \mathbf{Cont}_I \to \mathbf{Store} \to \mathbf{Ans}$$

   (opowiedzieć o włożeniu $X \to C^{C^X}$). Kolejność argumentów jest taka, żeby można było ostatnią frakcję **Store** $\to$ **Ans** traktować jako element $\mathbf{Cont}_I$. W pozostałych funkcjach, kontynuacje muszą brać wyniki ewaluacji:

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

   Funkcje semantyczne, na początek dla prostych instrukcji:

   $$\begin{aligned}\mathcal{I}[\![\texttt{skip}]\!] &= \lambda\rho.\lambda\kappa\lambda s.\kappa(s)\\ \mathcal{I}[\![x := e]\!] &= \lambda\rho\lambda\kappa\lambda s.\kappa(s[x \mapsto \mathcal{E}[\![e]\!]\rho s])\end{aligned}$$

   to przy semantyce bezpośredniej wyrażeń. Przy semantyce kontynuacyjnej zupełnie inaczej:

   $$\mathcal{I}[\![x := e]\!] = \lambda\rho\lambda\kappa\lambda s.\mathcal{E}[\![e]\!]\rho(\lambda n.\kappa(s[x \mapsto n]))s$$

Dalej, chyba najbardziej pouczająca reguła:

$$\mathcal{I}[\![I_1; I_2]\!] = \lambda\rho\lambda\kappa.\mathcal{I}[\![I_1]\!]\rho(\mathcal{I}[\![I_2]\!]\rho\kappa)$$

Teraz proste równania dla semantyki kontynuacyjnej wyrażeń:

$$\begin{aligned}
\mathcal{E}[\![n]\!] &= \lambda\rho\lambda\kappa\lambda s.\kappa(n) \\
\mathcal{E}[\![x]\!] &= \lambda\rho\lambda\kappa\lambda s.\kappa(s(\rho(x))) \\
\mathcal{E}[\![e_1 + e_2]\!] &= \lambda\rho\lambda\kappa\lambda s.\mathcal{E}[\![e_1]\!]\rho(\lambda n.\mathcal{E}[\![e_2]\!]\rho(\lambda m.\kappa(n+m))s)s
\end{aligned}$$

itd. `While` można napisać naiwnie, równaniem stałopunktowym. Zadanie: napisać reguły dla deklaracji.

2. Pierwsze ciekawe zastosowanie kontynuacji, czyli nagłe zatrzymanie programu:

$$I ::= \cdots \mid \texttt{abort}$$

*Rozwiązanie:* Wszystkie dziedziny semantyczne pozostają bez zmian. Typ funkcji semantycznej dla instrukcji to więc:

$$\mathcal{I}[\![]\!] : \mathbf{Instr} \to \mathbf{Env} \to \mathbf{Cont}_I \to \mathbf{Store} \to \mathbf{Ans}$$

przy czym trzeba tu zdecydować czym właściwie jest **Ans**. Na przykład przyjmijmy wygodnie, że **Ans** = **Store**. Wtedy odpowiednie równanie to:

$$\mathcal{I}[\![\texttt{abort}]\!]\rho\kappa s = s$$

Tutaj po raz pierwszy instrukcja, otrzymawszy kontynuację, zignorowała ją.

3. Wyskakiwanie z pętli, czyli:

$$I ::= \cdots \mid \texttt{break}$$

*Rozwiązanie:* dodatkowy element środowiska to jedna kontynuacja, która pamięta co się miało dziać po zakończeniu pętli. Tak więc:

$$\mathcal{I}[\![]\!] : \mathbf{Instr} \to \mathbf{Env} \to \mathbf{Cont}_I \to \mathbf{Cont}_I \to \mathbf{Store} \to \mathbf{Ans}$$

i równania (przy semantyce bezpośredniej wyrażeń):

$$\begin{aligned}
\mathcal{I}[\![\texttt{break}]\!]\rho\beta\kappa &= \beta \\
\mathcal{I}[\![\texttt{skip}]\!]\rho\beta\kappa &= \kappa \\
\mathcal{I}[\![\texttt{while } b \texttt{ do } I]\!]\rho\beta\kappa s &= ifte(\mathcal{B}[\![b]\!]\rho s, \\
&\qquad \mathcal{I}[\![I]\!]\rho\kappa(\mathcal{I}[\![\texttt{while } b \texttt{ do } I]\!]\rho\beta\kappa)s, \\
&\qquad \kappa s)
\end{aligned}$$

4. Z kolei:

$$I ::= \cdots \mid \texttt{continue}$$

Dziedzina jest ta sama (choć przy użyciu `break` i `continue` na raz trzeba by pamiętać dwie kontynuacje), ale w drugiej linii ostatniego równania, zamiast pierwszego $\kappa$ powinno być

$$\mathcal{I}[\![\texttt{while}\ b\ \texttt{do}\ I]\!]\rho\beta\kappa$$