SEMANTYKA I WERYFIKACJA - ĆW. 9

Kontynuujemy semantykę denotacyjną.

1. Oblicz semantykę denotacyjną programu `while true do skip`. Przypomnienie funkcji, której punkt stały liczymy:

$$\Phi(F) = \lambda s.if(\mathcal{B}[\![b]\!]s, \lambda s'.F(\mathcal{E}[\![I]\!]s'), s)$$

   Rozwiązanie: Funkcja pusta jest punktem stałym.

2. **Zadanie:** Obliczyć denotację programu

```
x := 1;
while x < 3 do
  x := x+1
```

   Rozwiązanie: Trzeba po kolei wyliczyć wszystkie semantyki podprogramów. W przypadku pętli, zacząć od $\emptyset$ i wyliczać kolejne Φ^n aż się ustabilizuje. (W sumie nigdy się nie ustabilizuje! Ale najmniejszy punkt stały istnieje.)

3. Dodajmy bloki i deklaracje zmiennych:

$$I ::= \cdots \mid \texttt{begin}\ d;\ I\ \texttt{end}$$
$$d ::= \texttt{var}\ x = e \mid d_1; d_2$$

   Rozwiązanie: Rozbijamy stan wyróżniając środowisko, czyli wprowadzamy dziedziny pomocnicze:

$$l \in \mathbf{Loc} = \{0, 1, 2, 3, \ldots\}$$
$$s \in \mathbf{Store} = \mathbf{Loc} \to \mathbf{Num}$$
$$\rho \in \mathbf{Env} = \mathbf{Var} \to \mathbf{Loc}$$

   Funkcje semantyczne mają typy:

$$\mathcal{I}[\![]\!] : \mathbf{Instr} \to \mathbf{Env} \to \mathbf{Store} \to \mathbf{Store}$$
$$\mathcal{D}[\![]\!] : \mathbf{Decl} \to \mathbf{Env} \to \mathbf{Store} \to (\mathbf{Env} \times \mathbf{Store})$$

   i są zdefiniowane tak:

$$\mathcal{D}[\![\texttt{var}\ x = e]\!] = \lambda\rho.\lambda s.(\rho[x \mapsto l], s[l \mapsto \mathcal{E}[\![e]\!]\rho s]) \quad \text{gdzie } l = \texttt{newloc}(s)$$
$$\mathcal{D}[\![d_1; d_2]\!] = \lambda\rho.\lambda s.\mathcal{D}[\![d_2]\!]\rho' s' \quad \text{gdzie } (\rho', s') = \mathcal{D}[\![d_1]\!]\rho s$$
$$\mathcal{I}[\![\texttt{begin}\ d;\ I\ \texttt{end}]\!] = \lambda\rho.\lambda s.\mathcal{I}[\![I]\!]\rho' s' \quad \text{gdzie } (\rho', s') = \mathcal{D}[\![d]\!]\rho s$$

Uwaga: trzeba też przerobić wszystkie definicje innych instrukcji, dopisując w nich środowiska! To jest znany problem braku modularności semantyki denotacyjnej.

4. Teraz dodajmy procedury, z widocznością statyczną i przekazywaniem przez zmienną:

$$I ::= \cdots \mid \texttt{begin}\ d;\ I\ \texttt{end} \mid \texttt{call}\ x_1(x_2)$$
$$d ::= \texttt{var}\ x = e \mid \texttt{proc}\ x_1(x_2)\ I \mid d_1; d_2$$

Rozwiązanie: Zróbmy rozwiązanie z dwoma środowiskami, procedur i zmiennych, czyli dodatkowo dziedziny:

$$\pi \in \mathbf{PEnv} = \mathbf{Var} \to \mathbf{Proc}$$
$$P \in \mathbf{Proc} = \mathbf{Loc} \to \mathbf{Store} \to \mathbf{Store}$$

Uwaga: w semantyce operacyjnej procedura przechowywała treść funkcji. Zgodnie z filozofią denotacyjną trzymamy tylko jej semantykę, czyli funkcję ze stanów w stany (ale bez środowisk, bo środowisko jest ustalone), która dodatkowo dostaje lokację parametru aktualnego.

Funkcje semantyczne mają typy:

$$\mathcal{I}[\![]\!] : \mathbf{Instr} \to \mathbf{PEnv} \to \mathbf{Env} \to \mathbf{Store} \to \mathbf{Store}$$
$$\mathcal{D}[\![]\!] : \mathbf{Decl} \to \mathbf{PEnv} \to \mathbf{Env} \to \mathbf{Store} \to (\mathbf{PEnv} \times \mathbf{Env} \times \mathbf{Store})$$

a nowe równania wyglądają tak:

$$\mathcal{D}[\![\texttt{proc}\ x_1(x_2)\ I]\!] = \lambda\pi.\lambda\rho.\lambda s.\langle \pi[x_1 \mapsto P], \rho, s\rangle$$
$$\text{gdzie } P = \lambda l.\lambda s'.\mathcal{I}[\![I]\!]\pi\rho[x_2 \mapsto l]s'$$
$$\mathcal{I}[\![\texttt{call}\ x_1(x_2)]\!] = \lambda\pi.\lambda\rho.\lambda s.\pi x_1(\rho x_2)s$$

Drugie równanie można η-zredukować i napisać krócej:

$$\mathcal{I}[\![\texttt{call}\ x_1(x_2)]\!] = \lambda\pi.\lambda\rho.\pi x_1(\rho x_2)$$

Uwaga: tu nie ma rekurencji!

5. Teraz przeróbmy to na przekazywanie przez wartość, nadal z widocznością statyczną. (Rozszerzamy składnię o dowolne wyrażenia jako parametry aktualne.)

Rozwiązanie: Semantyka procedury nie przyjmuje teraz lokacji tylko liczbę:

$$\pi \in \mathbf{PEnv} = \mathbf{Var} \to \mathbf{Proc}$$
$$P \in \mathbf{Proc} = \mathbf{Num} \to \mathbf{Store} \to \mathbf{Store}$$

Funkcje semantyczne mają typy jak poprzednio, a równania semantyczne zmieniają się tak:

$$\begin{aligned}\mathcal{D}[\![\texttt{proc } x_1(x_2)\ I]\!] &= \lambda\pi.\lambda\rho.\lambda s.\langle \pi[x_1 \mapsto P], \rho, s\rangle \\ &\quad \text{gdzie } P = \lambda n.\lambda s'.\mathcal{I}[\![I]\!]\pi\rho[x_2 \mapsto l]s'[l \mapsto n] \\ &\quad\quad \text{gdzie } l = newloc(s') \\ \mathcal{I}[\![\texttt{call } x(e)]\!] &= \lambda\pi.\lambda\rho.\lambda s.\pi x_1(\mathcal{E}[\![e]\!]\rho s)s\end{aligned}$$

Tego drugiego równania już się nie da η-zredukować.

6. Dorobić rekurencję.

   Rozwiązanie: Przy deklaracji procedury trzeba zmienić definicję P na rekurencyjną, stałopunktową:

$$\begin{aligned}\mathcal{D}[\![\texttt{proc } x_1(x_2)\ I]\!] &= \lambda\pi.\lambda\rho.\lambda s.\langle \pi[x_1 \mapsto P], \rho, s\rangle \\ &\quad \text{gdzie } P = \lambda n.\lambda s'.\mathcal{I}[\![I]\!]\pi[x_1 \to P]\rho[x_2 \mapsto l]s'[l \mapsto n] \\ &\quad\quad \text{gdzie } l = newloc(s')\end{aligned}$$

   Formalnie, definiujemy funkcjonał wyższego rzędu na typie **Proc**:

$$\Phi(F) = \lambda n.\lambda s'.\mathcal{I}[\![I]\!]\pi[x_1 \to F]\rho[x_2 \mapsto l]s'[l \mapsto n]$$

   i bierzemy jego najmniejszy punkt stały (kluczowe jest m.in. to, że $\mathcal{I}[\![I]\!]$ jest monotoniczne ze względu na środowisko procedur).

   Pytanie: A dlaczego nie działa trik z semantyki operacyjnej?

   Odpowiedź: Bo semantyka (element **Proc**) już nie zależy od środowiska, więc nie można jej poprawionego środowiska przekazać.

7. Jeśli starczy czasu:

   - przerobić na dynamiczne wiązanie (zmienić typ **Proc**, żeby przyjmowała także **PEnv** i **Env**)