# Solution for Homework №2

Homework
PSV 2020/21

**Author:** *Jakub Bednarz*

**Semantic types** Let us start with the trivial types:

$$\begin{aligned}|\mathbf{Num}| &= \mathbb{Z}\\ |\mathbf{Var}| &= \mathbf{Var} \sqcup \{\mathtt{var}\}\end{aligned}$$

We introduce $\{\mathtt{var}\}$ into the semantic domain of variables (as a "formal" variable) in order to simplify certain operations. Now, we need to define environment and memory types:

$$\begin{aligned}\mathbf{Store} &= \mathbf{Loc} \rightharpoonup |\mathbf{Num}|\\ \mathbf{Setter} &= |\mathbf{Num}| \rightharpoonup (\mathbf{Store} \rightharpoonup \mathbf{Store})\\ \mathbf{Env} &= |\mathbf{Var}| \rightharpoonup \mathbf{Loc} \times \mathbf{Setter}\end{aligned}$$

where **Setter** is the type of the anonymous procedure in $\mathtt{var}\ x_1\ \mathtt{set\ to}\ x_2\ \mathtt{by}\ S$. The other types are standard:

$$\begin{aligned}|\mathbf{Expr}| &= \mathbf{Env} \times \mathbf{Store} \rightharpoonup |\mathbf{Num}|\\ |\mathbf{Decl}| &= \mathbf{Env} \times \mathbf{Store} \rightharpoonup \mathbf{Env} \times \mathbf{Store}\\ |\mathbf{Stmt}| &= \mathbf{Env} \times \mathbf{Store} \rightharpoonup \mathbf{Store}\end{aligned}$$

**Evaluation functions** As for the evaluations, we have $\mathcal{N}[\![\cdot]\!]$, $\mathcal{V}[\![\cdot]\!]$, $\mathcal{E}[\![\cdot]\!]$, $\mathcal{D}[\![\cdot]\!]$ and $\mathcal{S}[\![\cdot]\!]$ for **Num**, **Var**, **Expr**, **Decl** and **Stmt** respectively. Let us now define them:

1. $\mathcal{N}[\![\cdot]\!] : \mathbf{Num} \to |\mathbf{Num}|$:

   $$\mathcal{N}[\![n]\!] = n$$

2. $\mathcal{V}[\![\cdot]\!] : \mathbf{Var} \to |\mathbf{Var}|$:

   $$\mathcal{V}[\![x]\!] = x$$

   Given the nature of this evaluation, we will use it implicitly in the other definitions.

3. $\mathcal{E}[\![\cdot]\!] : \mathbf{Expr} \to |\mathbf{Expr}|$:

   $$\begin{aligned}\mathcal{E}[\![n]\!](\rho,\mu) &= \mathcal{N}[\![n]\!]\\ \mathcal{E}[\![x]\!](\rho,\mu) &= \mathtt{let}\ (\ell_x, \mathsf{set}_x) = \rho(\mathcal{V}[\![x]\!])\\ &\quad\ \mathtt{in}\ \mu(\ell_x)\\ \mathcal{E}[\![e_1 \star e_2]\!](\rho,\mu) &= \mathcal{E}[\![e_1]\!](\rho,\mu) \star \mathcal{E}[\![e_2]\!](\rho,\mu), \qquad \star \in \{+,-,*\}\end{aligned}$$

4. $\mathcal{D}[\![\cdot]\!] : \textbf{Decl} \to |\textbf{Decl}|$:

$$\mathcal{D}[\![\texttt{var } x_1 \texttt{ set to } x_2 \texttt{ by } S]\!](\rho,\mu) = \mathsf{newvar}(\mathcal{V}[\![x_1]\!], \mathsf{setter}(\mathcal{V}[\![x_1]\!], \mathcal{V}[\![x_2]\!], S, \rho), 0)(\rho,\mu)$$
$$\mathcal{D}[\![d_1;\ d_2]\!] = \mathcal{D}[\![d_2]\!] \circ \mathcal{D}[\![d_1]\!]$$
$$\mathcal{D}[\![\epsilon]\!] = \mathsf{id}_{\textbf{Env}\times\textbf{Store}}$$

Aside from $\mathsf{setter}$, whose implementation is more involved, we have introduced an auxiliary function:

$$\begin{aligned}\mathsf{newvar}(x, \mathsf{set}_x, \mathsf{init}_x)(\rho,\mu) = \texttt{let } & \ell_x = \mathsf{alloc}(\mu) \\ \texttt{in } & (\rho[x \leftarrow (\ell_x, \mathsf{set}_x)], \mu[\ell_x \leftarrow \mathsf{init}_x])\end{aligned}$$

which introduces (or overwrites) a variable $x : |\textbf{Var}|$ with a given setter $\mathsf{set}_x : \textbf{Setter}$ and an initial value $\mathsf{init}_x : |\textbf{Num}|$.

5. $\mathcal{S}[\![\cdot]\!] : \textbf{Stmt} \to |\textbf{Stmt}|$:

$$\mathcal{S}[\![x := e]\!] = \mathsf{proxyset}(\mathcal{V}[\![x]\!], e)$$
$$\mathcal{S}[\![\texttt{var } := e]\!] = \mathsf{proxyset}(\texttt{var}, e)$$
$$\mathcal{S}[\![S_1;\ S_2]\!](\rho,\mu) = \mathcal{S}[\![S_2]\!](\rho, \mathcal{S}[\![S_1]\!](\rho,\mu))$$
$$\mathcal{S}[\![\texttt{if } e = 0 \texttt{ then } S_1 \texttt{ else } S_2]\!] = \langle \textit{default} \rangle$$
$$\mathcal{S}[\![\texttt{while } e \neq 0 \texttt{ do } S]\!] = \langle \textit{default} \rangle$$
$$\mathcal{S}[\![\texttt{begin } d;\ S \texttt{ end}]\!] = \mathcal{S}[\![S]\!] \circ \mathcal{D}[\![d]\!]$$

where:

$$\begin{aligned}\mathsf{proxyset}(x, e)(\rho,\mu) = \texttt{let } & (\ell_x, \mathsf{set}_x) = \rho(x) \\ \texttt{in } & \mathsf{set}_x(\mathcal{E}[\![e]\!](\rho,\mu))(\mu)\end{aligned}$$

is a “variable” (including `var`) assignment using setter.

**Setter** All that is left is implementing $\mathsf{setter}$; this is the place where adding `var` to $|\textbf{Var}|$ will pay off. Given the recursive nature of the procedure, we will want to implement first a version of the procedure which “takes itself”. Mathematically, we have:

$$\mathsf{setter}_0(x_1, x_2, S, \rho)(\mathsf{recur})(\mu)(n) = \mathcal{S}[\![S]\!](\rho_3, \mu_3)$$
$$\mathsf{newvar}(x_1, \mathsf{recur}, 0)(\rho_2, \mu_2) = (\rho_3, \mu_3)$$
$$\mathsf{newvar}(x_2, \mathsf{truesetter}(\rho_1, \mathcal{V}[\![x_2]\!]), n) = (\rho_2, \mu_2)$$
$$\mathsf{newvar}(\texttt{var}, \mathsf{truesetter}(\rho, \mathcal{V}[\![x_1]\!]), 0)(\rho,\mu) = (\rho_2, \mu_2)$$

In words, $\mathsf{setter}_0(x_1, x_2, S, \rho) : \textbf{Setter} \rightharpoonup \textbf{Setter}$ is the “recurrent” version, in which we:

- set `var` pseudovariable to set $x_1$ directly, and assigns 0 to it (though it's inaccessible);
- set $x_2$ to direct write, and assigns $n$ to it (this is the formal parameter of the anonymous procedure);
- set $x_1$ to recur, and also assigns it the value 0;
- execute $S$ in this new state/environment, creating new state.

With that, we can write:

$$\mathsf{setter}(x_1, x_2, S, \rho) = \mathsf{fix}(\mathsf{setter}_0(x_1, x_2, S, \rho))$$