Tema 7. Más sobre semántica denotacional

En este capítulo extendemos el lenguaje **WHILE** con *bloques* y *procedimientos* y daremos la semántica denotacional para esta extensión.

Entornos y memorias

En primer lugar, expandimos el lenguaje con bloques que tienen declaraciones de variables y procedimientos (igual que como vimos en el capítulo 3). A diferencia del capítulo 3 solo daremos el caso de *ámbito estáticos*.

Memoria y direcciones

Definición: (Direcciones)
Decimos que el dominio $\mathbf{Loc}$ representa las posibles localizaciones de las variables.
Definición: (Entorno de variables)
Definimos la siguiente familia de funciones como *entorno de variables* $\mathbf{Env}_V = \mathbf{Var} \to \mathbf{Loc}$ que asignan a cada variable, su dirección.
Definición: (Memoria)
Definimos la siguiente familia de funciones como *memoria* $\mathbf{Store} = \mathbf{Loc} \cup \{\mathrm{next}\} \to \mathbb{Z}$ que asignan a cada dirección, un valor.

De esta forma el estado como lo definíamos anteriormente será la composición de funciones del entorno de variables y la memoria. Esto motiva la siguiente definición.

Definición: (*Lookup*)
Definimos la función *lookup* con tipo $\mathbf{Env}_V \to \mathbf{Store} \to \mathbf{Store}$ a

$$\mathrm{lookup}\ env_V\ sto = sto \circ env_V.$$

Es decir, que dado un entorno de variables y una memoria, obtenemos el estado equivalente.

Con estas nuevas definiciones el tipo de la función semántica denotacional pasará a ser

$$\mathcal{S}'_{\mathrm{ds}} : \mathbf{Stm} \to \mathbf{Env}_V \to (\mathbf{Store} \hookrightarrow \mathbf{Store})$$

y su definición será

$$
\begin{aligned}
\mathcal{S}'_{\mathrm{ds}}[\![x := a]\!]\ env_V\ sto &= sto\,[l \mapsto \mathcal{A}[\![a]\!]\,(\mathrm{lookup}\ env_V\ sto)] \\
&\quad \text{donde } l = env_V\ x \\
\mathcal{S}'_{\mathrm{ds}}[\![\texttt{skip}]\!]\ env_V &= \mathrm{id} \\
\mathcal{S}'_{\mathrm{ds}}[\![S_1\ ;\ S_2\ ]\!]\ env_V &= (\mathcal{S}'_{\mathrm{ds}}[\![S_2]\!]\ env_V) \circ (\mathcal{S}'_{\mathrm{ds}}[\![S_1]\!]\ env_V) \\
\mathcal{S}'_{\mathrm{ds}}[\![\texttt{if}\ b\ \texttt{then}\ S_1\ \texttt{else}\ S_2]\!]\ env_V &= \mathrm{cond}\,(\mathcal{B}[\![b]\!] \circ (\mathrm{lookup}\ env_V)\,, \mathcal{S}'_{\mathrm{ds}}[\![S_1]\!]\ env_V, \mathcal{S}'_{\mathrm{ds}}[\![S_2]\!]\ env_V) \\
\mathcal{S}'_{\mathrm{ds}}[\![\texttt{while}\ b\ \texttt{do}\ S]\!]\ env_V &= \mathrm{FIX}\ F \\
&\quad \text{donde } F\ g = \mathrm{cond}\,(\mathcal{B}[\![b]\!] \circ (\mathrm{lookup}\ env_V)\,, g \circ (\mathcal{S}'_{\mathrm{ds}}[\![S]\!]\ env_V)\,, \mathrm{id})
\end{aligned}
$$

Declaraciones

Una vez actualizada la definición de las instrucciones básicas de **WHILE**, debemos dar una semántica para las nuevas construcciones, es decir, la declaración de variables y procedimientos así como los bloques y las llamadas. Empecemos por la declaración de variables. En primer lugar el tipo de esta función será

$$
\mathcal{D}^V_{\mathrm{ds}} : \mathrm{Dec}_V \to (\mathbf{Env}_V \times \mathbf{Store}) \to (\mathbf{Env}_V \times \mathbf{Store})\,.
$$

Es decir, dada una declaración de una variable, actualiza el estado. Su definición es, por tanto,

$$
\begin{aligned}
&\mathcal{D}^V_{\mathrm{ds}}[\![\varepsilon]\!] = \mathrm{id} \\
&\mathcal{D}^V_{\mathrm{ds}}[\![\texttt{var}\ x := a;\ D_V]\!]\,(env_V, sto) = \\
&\qquad\qquad \mathcal{D}^V_{\mathrm{ds}}[\![D_V]\!]\,(env_V\,[x \mapsto l]\,, sto\,[l \mapsto v]\,[\mathrm{next} \mapsto \mathrm{new}\ l])\,, \\
&\qquad \text{donde } l = sto\ \mathrm{next} \text{ y } v = \mathcal{A}[\![a]\!]\,(\mathrm{lookup}\ env_V\ sto)\,.
\end{aligned}
$$

Con lo que al declarar cada variable le asignamos la dirección l (que es la siguiente disponible), le damos el valor correspondiente a esa dirección y actualizamos el siguiente hueco.

Por tanto, la semántica de un bloque sin procedimientos será

$$
\begin{aligned}
\mathcal{S}'_{\mathrm{ds}}[\![\texttt{begin}\ D_V\ \varepsilon\ \texttt{end}]\!]\ env_V\ sto &= \mathcal{S}_{\mathrm{d}}[\![S]\!]\ env'_V\ sto' \\
&\quad \text{donde } \mathcal{D}^V_{\mathrm{ds}}[\![D_V]\!]\,(env_V, sto) = (env'_V, sto')\,.
\end{aligned}
$$

Veamos, entonces, la declaración de procedimientos. Los entornos de procedimientos serán funciones de tipo

$$
\mathbf{Env}_P = \mathbf{Pname} \to (\mathbf{Store} \hookrightarrow \mathbf{Store})
$$

Es decir, vamos a asignar a cada procedimiento su *significado* directamente (es conveniente recordar que en el caso operacional, asignábamos a cada procedimiento la sentencia que ejecutaba, pero no su significado). Con esto, la función semántica tendrá tipo

$$
\mathcal{D}^P_{\mathrm{ds}} : \mathbf{Dec}_p \to \mathbf{Env}_V \to \mathbf{Env}_P \to \mathbf{Env}_P
$$

y su definición recursiva (para procedimientos no recursivos),

$$\mathcal{D}^P_{\text{ds}}[\![\varepsilon]\!] = \text{id}$$
$$\mathcal{D}^P_{\text{ds}}[\![\texttt{proc}\ p\ \texttt{is}\ S;\ D_P]\!]\ env_V\ env_P = \mathcal{D}^P_{\text{ds}}[\![D_P]\!]\ env_V\ (env_P\ [p \mapsto g])\,,$$
$$\text{donde } g = \mathcal{S}_{\text{ds}}[\![S]\!]\ env_V\ env_P.$$

Es decir, asignamos al procedimiento p el valor de su sentencia S.

La función semántica $\mathcal{S}_{\text{ds}}$

Por último, veamos como queda al final la función semántica. En primer lugar, su tipo tendrá que tener en cuenta los entornos de variable y de variables. Por tanto,

$$\mathcal{S}_{\text{ds}} : \mathbf{Stm} \to \mathbf{Env}_V \to \mathbf{Env}_P \to (\mathbf{Store} \hookrightarrow \mathbf{Store})$$

Por otro lado, la definición de esta nueva función semántica será igual a la dada por $\mathcal{S}'_{\text{ds}}$ únicamente añadiendo como parámetro el entorno de procedimientos env_P. Sin embargo, los bloques y las llamadas sí debemos especificarlas:

$$\mathcal{S}_{\text{ds}}[\![\texttt{begin}\ D_V\ D_P\ S\ \texttt{end}]\!]\ env_V\ env_P\ sto = \mathcal{S}_{\text{ds}}[\![S]\!]\ env'_V\ env'_P\ sto'$$
$$\text{donde } \mathcal{D}^V_{\text{ds}}[\![D_V]\!]\ (env_V, sto) = (env'_V, sto')$$
$$\text{y } \mathcal{D}^P_{\text{ds}}[\![D_P]\!]\ env'_V\ env_P = env'_P$$
$$\mathcal{S}_{\text{ds}}[\![\texttt{call}\ p]\!]\ env_V\ env_P = env_P\ p$$

Procedimientos recursivos

Lo último que nos queda por ver es el significado de los procedimientos con llamadas recursivas en sus instrucciones. A diferencia de la forma que utilizábamos para tratar con este caso en la semántica operacional en la que actualizábamos el entorno cada vez que se realizaba una llamada recursiva, en este caso, la recursión será manejada *de una sola vez* cuando se realice la declaración del procedimiento. Esto guarda una clara relación con la iteración de los bucles por lo que, de la misma manera, utilizaremos el *punto fijo* de un funcional. En definitiva,

$$\mathcal{D}^P_{\text{ds}}[\![\varepsilon]\!] = \text{id}$$
$$\mathcal{D}^P_{\text{ds}}[\![\texttt{proc}\ p\ \texttt{is}\ S;\ D_P]\!]env_V, env_P = \mathcal{D}^P_{\text{ds}}[\![D_P]\!]\ env_V\ (env_P\ [p \mapsto \text{FIX}\ F])\,,$$
$$\text{donde } F\ g = \mathcal{S}_{\text{ds}}[\![S]\!]\ env_V\ env_P\ [p \mapsto g]\,.$$