G54FOP: Lecture 16

Denotational Semantics and Domain Theory II

Henrik Nilsson

University of Nottingham, UK

This Lecture

- Denotational semantics for small imperative language.
- Introduction to semantics of loops and recursion.

Imperative Language (1)

Syntax of expressions:

e	$\longrightarrow$		*expressions:*
		x	*variable*
	$\mid$	n	*constant natural number,* $\mathbb{N}$
	$\mid$	**`true`**	*constant true*
	$\mid$	**`false`**	*constant false*
	$\mid$	**`not`** e	*logical negation*
	$\mid$	e **`&&`** e	*logical conjunction*

$\ldots$

Imperative Language (2)

$$
\begin{array}{lllr}
e & \rightarrow & & \textit{expressions:} \\
 & \ldots & & \\
 & \mid & e \; \mathtt{+} \; e & \textit{addition} \\
 & \mid & e \; \mathtt{-} \; e & \textit{subtraction} \\
 & \mid & e \; \mathtt{=} \; e & \textit{numeric equality test} \\
 & \mid & e \; \mathtt{<} \; e & \textit{numeric less than test}
\end{array}
$$

Imperative Language (3)

Syntax of commands:

$$\begin{array}{rcll}
c & \longrightarrow & & \textit{commands:} \\
 & | & \mathtt{skip} & \textit{no operation} \\
 & | & x \; \mathtt{:=} \; e & \textit{assignment} \\
 & | & c \; \mathtt{;} \; c & \textit{sequence} \\
 & | & \mathtt{if} \; e \; \mathtt{then} \; c \; \mathtt{else} \; c & \textit{conditional} \\
 & | & \mathtt{while} \; e \; \mathtt{do} \; c & \textit{iteration}
\end{array}$$

Semantics of Expressions (1)

We take the ***semantic domain*** to be $\mathbb{N}$ for simplicity.

We need a way to give meaning to ***variables***. A ***store*** maps a variable name to its value:

$$\begin{array}{rcl} \Sigma & = & x \longrightarrow \mathbb{N} \\ \sigma & : & \Sigma \end{array}$$

Semantics of Expressions (2)

We then need two ***semantic functions***, one for expressions (have no side effects in this language), one for commands.

Starting with the one for expressions:

$$\mathrm{E}[\![\cdot]\!] \ : \ e \rightarrow (\Sigma \rightarrow \mathbb{N})$$

(Note: $e \rightarrow (\Sigma \rightarrow \mathbb{N}) = e \rightarrow \Sigma \rightarrow \mathbb{N}$ etc.)

(Definition on whiteboard)

Semantics of Commands

A command is executed for its ***effects***: given a state, executing a command results in a new state. A command is a ***state transformer***.

In our case, the state comprises only the store:

$$\Sigma \quad = \quad x \longrightarrow \mathbb{N}$$

Thus, type of state transformer: $\Sigma \rightarrow \Sigma$.

Semantic function for commands:

$$\mathrm{C}[\![\cdot]\!] \quad : \quad c \longrightarrow (\Sigma \longrightarrow \Sigma) \qquad \text{[Not correct yet!]}$$

(Definition on whiteboard)