Term Syntax

values v	::=	x	variable
	$\mid$	$()$	unit constant
	$\mid$	$\texttt{true} \mid \texttt{false}$	boolean constants
	$\mid$	$\texttt{fun}\ x \mapsto c$	function
	$\mid$	$\texttt{handler}\ (\texttt{ret}\ x \mapsto c_r; h)$	handler
computations c	::=	$\texttt{if}\ v\ \texttt{then}\ c_1\ \texttt{else}\ c_2$	conditional
	$\mid$	$v_1\ v_2$	application
	$\mid$	$\texttt{ret}\ v$	returned value
	$\mid$	$op(v;\ y.c)$	operation call
	$\mid$	$\texttt{do}\ x \leftarrow c_1\ \texttt{in}\ c_2$	sequencing
	$\mid$	$\texttt{with}\ v\ \texttt{handle}\ c$	handling
operation clauses h	::=	$\emptyset \mid h \cup \{op(x;\ k) \mapsto c_{op}\}$	

Type Syntax

$$
\begin{array}{rcll}
\text{(value) type } A, B & ::= & \texttt{unit} & \text{unit type} \\
& \mid & \texttt{bool} & \text{boolean type} \\
& \mid & A \to \underline{C} & \text{function type} \\
& \mid & \underline{C} \Rightarrow \underline{D} & \text{handler type} \\
\text{computation type } \underline{C}, \underline{D} & ::= & A ! \Sigma / \mathcal{E} & \\
\text{signature } \Sigma & ::= & \emptyset \mid \Sigma \cup \{ op : A \to B \} &
\end{array}
$$

Type Syntax (additions)

$$
\begin{array}{rcl}
\text{value context } \Gamma & ::= & \varepsilon \mid \Gamma, x : A \\
\text{template context } \mathtt{Z} & ::= & \varepsilon \mid \mathtt{Z}, z : A \to * \\
\text{template } T & ::= & z\ v \\
 & \mid & \texttt{if } v \texttt{ then } T_1 \texttt{ else } T_2 \\
 & \mid & op(v;\ y.T) \\
\text{(effect) theory } \mathcal{E} & ::= & \emptyset \mid \mathcal{E} \cup \{\Gamma; \mathtt{Z} \vdash T_1 \sim T_2\}
\end{array}
$$

The *any type* $*$ used in template types can be instantiated to any computation type so that we can reuse templates.

Full equation

$$\Gamma; \mathrm{Z} = (x : \mathtt{string}, y : \mathtt{string}); (z : \mathtt{unit} \longrightarrow *)$$

$$\Gamma; \mathrm{Z} \vdash \mathit{print}(x; \_.\mathit{print}(y; \_.z\ (\,))) \sim \mathit{print}(x \hat{}\, y; \_.z\ (\,))$$