values v	::=	x	variable
	|	$()$	unit
	|	n	integer
	|	$\texttt{Left}\ v \mid \texttt{Right}\ v$	sum constructors
	|	(v_1, v_2)	pair
	|	$\texttt{fun}\ x \mapsto c$	function
	|	$\texttt{handler}\ (\texttt{ret}\ x \mapsto c_r; h)$	handler
computations c	::=	$\texttt{ret}\ v$	returned value
	|	$v_1\ v_2$	application
	|	$\texttt{match}\ v\ \texttt{with}\ \texttt{Left}\ x \mapsto c_1 \mid \texttt{Right}\ x \mapsto c_2$	sum match
	|	$\texttt{match}\ v\ \texttt{with}\ (x, y) \mapsto c$	product match
	|	$op(v; y.c)$	operation call
	|	$\texttt{let rec}\ f\ x = c_1\ \texttt{in}\ c_2$	recursive function
	|	$\texttt{do}\ x \leftarrow c_1\ \texttt{in}\ c_2$	sequencing
	|	$\texttt{with}\ v\ \texttt{handle}\ c$	handling
operation clauses h	::=	$\emptyset \mid h \cup \{op(x; k) \mapsto c_{op}\}$	

Figure 1: *EEFF* Term Syntax

(value) type A, B	::=	$\texttt{unit}$	unit type
	|	$\texttt{int}$	int type
	|	$\texttt{empty}$	empty type
	|	$A + B$	sum type
	|	$A \times B$	product type
	|	$A \to \underline{C}$	function type
	|	$\underline{C} \Rightarrow \underline{D}$	handler type
computation type $\underline{C}, \underline{D}$	::=	$A ! \Sigma / \mathcal{E}$	
signature Σ	::=	$\emptyset \mid \Sigma \cup \{op : A \to B\}$	
value context Γ	::=	$\varepsilon \mid \Gamma, x : A$	
template context $\mathtt{Z}$	::=	$\varepsilon \mid \mathtt{Z}, z : A \to *$	
template T	::=	$z\ v$	applied variable
	|	$\texttt{match}\ v\ \texttt{with}\ \texttt{Left}\ x \mapsto T_1 \mid \texttt{Right}\ x \mapsto T_2$	sum match
	|	$\texttt{match}\ v\ \texttt{with}\ (x, y) \mapsto T$	product match
	|	$\texttt{let}\ x = v\ \texttt{in}\ T$	value bind
	|	$op(v;\ y.T)$	operation call
(effect) theory $\mathcal{E}$	::=	$\emptyset \mid \mathcal{E} \cup \{\Gamma; \mathtt{Z} \vdash T_1 \sim T_2\}$	

Figure 2: *EEFF* Type Syntax