1 Syntax

1.1 Language E of Expressions

Language	Types	Expressions	Comments
E (Ch.4)	`num` `str`	x $\texttt{num}[n]$ $\texttt{str}[s]$ $e_1 + e_2$ $e_1 * e_2$ $e_1 \hat{}\ e_2$ $\texttt{let}\ x\ \texttt{be}\ e_1\ \texttt{in}\ e_2$	
ED (Ch.8.1)		$\texttt{fun}\ f(x:\tau_1):\tau_2\ =\ e_1\ \texttt{in}\ e_2$ $e_1(e_2)$	Limited extension, superceded by next: First-order functions, with their names are from a different variable supply here.
EF (Ch.8.2)	$\tau_1 \to \tau_2$	$\lambda(x:\tau)e$ $e_1(e_2)$	Full functions as first-class citizens, with variable names.

1.2 Language T of Gödel total functions

Language	Types	Expressions	Comments
T (Ch.9)	`nat` $\tau_1 \to \tau_2$	$x, \texttt{z}, \texttt{s}(e)$ $\texttt{rec}\ x\ (\texttt{z} \hookrightarrow e_0, \texttt{s}(x)\ \texttt{with}\ y \hookrightarrow e_1)$ $\lambda(x:\tau)e$ $e_1(e_2)$	**T:** Total functions (limited recursion)
Pairs (Ch.10)	`unit` $\tau_1 \times \tau_2$	$()$ $< e_1, e_2 >$ $e.\texttt{l}$ $e.\texttt{r}$	Pairs, generalised in next extension
Products (Ch.10)	$< \tau_i >_{i \in I}$	$< e_i >_{i \in I}$ $e.i$	I a finite index set
Alternative (Ch.11)	`void` $\tau_1 + \tau_2$	`abort` $\texttt{l}.e$ $\texttt{r}.e$ $\texttt{case}\ e(\texttt{l}.x_1 \hookrightarrow e_1, \texttt{r}.x_2 \hookrightarrow e_2)$	choice between two things, generalised in next extension
Sum (Ch.11)	$< \tau_i >_{i \in I}$	$\texttt{i}.e$ $\texttt{case}\ e\ \ < \imath.x_i \hookrightarrow e_i >_{i \in I}$	Choice from finite index set I. Can express Booleans and Enums.
Infinite (Ch.14)	./.	$\texttt{map}_{t.\tau}(x.e')\ e$	The general type operation may use $+$, $\times$, `unit` and `void` from before. Restricted to *positive* operation.

1.3 Language family PCF of (general) recursive functions

Language	**Types**	**Expressions**	Comments
PCF (Ch.19)	$\mathtt{nat}$ $\tau_1 \to \tau_2$	x $\mathtt{z}$ $\mathtt{s}(e)$ $\mathtt{ifz}\ e\ (e_0,\ x.e_1)$ $\lambda(x:\tau)e$ $e_1(e_2)$ $\mathtt{fix}\ (x:\tau)\ \mathtt{is}\ e$	
FPC (Ch.20)	t $\mathtt{rec}t\ \mathtt{is}\ \tau$	$\mathtt{fold}_{t.\tau}(e)$ $\mathtt{unfold}(e)$	Full functions as first-class citizens, with variable names.

2 Typing

Language	**Rules**	Comments
E (Ch.4)	$\dfrac{}{\Gamma, x : \tau \vdash x : \tau}$ $\quad$ $\dfrac{}{\Gamma \vdash \mathtt{str}[s] : \mathtt{str}}$ $\quad$ $\dfrac{}{\Gamma \vdash \mathtt{num}[n] : \mathtt{num}}$	Typing axiom and atoms
	$\dfrac{\Gamma \vdash e_1 : \mathtt{num} \quad \Gamma \vdash e_2 : \mathtt{num}}{\Gamma \vdash \mathtt{plus}(e_1, e_2) : \mathtt{num}}$ $\quad$ $\dfrac{\Gamma \vdash e_1 : \mathtt{num} \quad \Gamma \vdash e_2 : \mathtt{num}}{\Gamma \vdash \mathtt{times}(e_1, e_2) : \mathtt{num}}$	`num` operations
	$\dfrac{\Gamma \vdash e_1 : \mathtt{str} \quad \Gamma \vdash e_2 : \mathtt{str}}{\Gamma \vdash \mathtt{cat}(e_1, e_2) : \mathtt{str}}$ $\quad$ $\dfrac{\Gamma \vdash e : \mathtt{str}}{\Gamma \vdash \mathtt{len}(e) : \mathtt{num}}$	conversions
	$\dfrac{\Gamma \vdash e_1 : \tau_1 \quad \Gamma, x : \tau_1 \vdash e_2 : \tau_2}{\Gamma \vdash \mathtt{let}\ x\ \mathtt{be}\ e_1\ \mathtt{in}\ e_2 : \tau_2}$	local binding
ED (Ch.8.1)	...	♠**JB:** TODO♠