0.1 OCaml expressions (without objects)

$$\textsc{Tuple}\ \frac{\forall i.\ \Gamma, \Phi \vdash \langle e_i : \tau'_i \rangle \qquad \Gamma, \Phi \vdash \tau \prec \tau_0\ *\ ..\ *\ \tau_n \qquad \forall i. \Gamma, \Phi \vdash \tau_i \equiv \tau'_i}{\Gamma, \Phi \vdash \langle (e_0,\ ..\ ,\ e_n) : \tau \rangle}$$

$$\textsc{Match}\ \frac{\begin{array}{c}\Gamma, \Phi \vdash \langle e : \tau_{scrut} \rangle \qquad \forall i.\ \Gamma, \Phi, \Sigma, \Sigma \vdash \langle p_i : \tau_i \rangle \Rightarrow (\overline{v_i : \tau_i}), (\overline{\tau_{\exists_i}}), \Phi_i \\ \forall i.\ let\ \Gamma_i = \Gamma \oplus_{\mathcal{V}} (\overline{v_i : \tau_i}) \oplus (\overline{\tau_{\exists_i}}) \\ \forall i.\ \Gamma_i, \Phi_i \vdash \langle e_i : \tau_{e_i} \rangle \qquad \forall i.\ \Gamma, \Phi \vdash \tau_{e_i}\ wf \qquad \forall i.\ \Gamma_i, \Phi_i \vdash \tau \equiv \tau_{e_i}\end{array}}{\Gamma, \Phi \vdash \langle \mathtt{match}\ e\ \mathtt{with}\ \overline{\mid p \to e} : \tau \rangle}$$

$$\textsc{Record}\ \frac{\begin{array}{c} let\ \tau_{rec} = \Gamma.Types(\tau) \\ \forall i. let\ \tau_{l_i} = find\_label(\Gamma, \Phi, l_i, \tau_{rec}) \qquad \forall i.\ \Gamma, \Phi \vdash \langle e : \tau_i \rangle \\ \Gamma, \Phi, \Sigma \vdash \tau_0 \leq \tau_{l_0} \Rightarrow \theta_0 \qquad \forall i_{\geq 1}.\ \Gamma, \Phi, \theta_{i-1} \vdash \tau_i \leq \tau_{l_i} \Rightarrow \theta_i \\ let\ \tau_{inst} = \theta_n(\tau_{rec}) \qquad \Gamma, \Phi \vdash \tau_{inst} \equiv \tau \end{array}}{\Gamma, \Phi \vdash \langle \{l_0 = e_0;\ ..\ ; l_n = e_n\} : \tau \rangle}$$

$$\textsc{Field}\ \frac{\begin{array}{c} \Gamma, \Phi \vdash \langle e : \tau_e \rangle \\ let\ \tau_{rec} = \Gamma.Types(\tau_e) \qquad let\ \tau_l = find\_label(\Gamma, \Phi, l, \tau_{rec}) \\ \Gamma, \Phi, \Sigma \vdash \tau \leq \tau_l \Rightarrow \theta \qquad let\ \tau_{inst} = \theta(\tau_{rec}) \qquad \Gamma, \Phi \vdash \tau_{inst} \equiv \tau_e \end{array}}{\Gamma, \Phi \vdash \langle e.l : \tau \rangle}$$

$$\textsc{Set-Field}\ \frac{\begin{array}{c}\Gamma, \Phi \vdash \langle e_1 : \tau_1 \rangle \qquad \Gamma, \Phi \vdash \langle e_2 : \tau_2 \rangle \\ \mathit{let}\ \tau_{rec} = \Gamma.Types(\tau_1) \qquad \mathit{let}\ \tau_l = \mathit{find\_label}(\Gamma, \Phi, l, \tau_{rec}) \\ \mathit{label\_kind}(\Gamma, \Phi, l, \tau_{rec})\ \mathit{is\ Mutable} \qquad \Gamma, \Phi, \Sigma \vdash \tau_2 \leq \tau_l \Rightarrow \theta \\ \mathit{let}\ \tau_{inst} = \theta(\tau_{rec}) \qquad \Gamma, \Phi \vdash \tau_{inst} \equiv \tau_1 \qquad \Gamma, \Phi \vdash \tau \equiv \texttt{unit}\end{array}}{\Gamma, \Phi \vdash \langle e_1.l \leftarrow e_2 : \tau \rangle}$$

$$\textsc{Array}\ \frac{\begin{array}{c}\forall i.\ \Gamma, \Phi \vdash \langle e_i : \tau_i \rangle \\ \forall i_{\geq 1}.\ \Gamma, \Phi \vdash \tau_{i-1} \equiv \tau_i \qquad \Gamma, \Phi \vdash \tau \prec \tau_{arg}\ \texttt{array} \qquad \Gamma, \Phi \vdash \tau_0 \equiv \tau_{arg}\end{array}}{\Gamma, \Phi \vdash \langle [|e_0;\ ..\ ; e_n|] : \tau \rangle}$$

$$\textsc{Sequence}\ \frac{\begin{array}{c}\Gamma, \Phi \vdash \langle e_1 : \tau_1 \rangle \\ \Gamma, \Phi \vdash \langle e_2 : \tau_2 \rangle \qquad \Gamma, \Phi \vdash \tau_1 \equiv \texttt{unit} \qquad \Gamma, \Phi \vdash \tau \equiv \tau_2\end{array}}{\Gamma, \Phi \vdash \langle e_1;\ e_2 : \tau \rangle}$$

$$\textsc{While}\ \frac{\begin{array}{c}\Gamma, \Phi \vdash \langle e_1 : \tau_1 \rangle \qquad \Gamma, \Phi \vdash \langle e_2 : \tau_2 \rangle \\ \Gamma, \Phi \vdash \tau_1 \equiv \texttt{bool} \qquad \Gamma, \Phi \vdash \tau_2 \equiv \texttt{unit} \qquad \Gamma, \Phi \vdash \tau \equiv \texttt{unit}\end{array}}{\Gamma, \Phi \vdash \langle \texttt{while}\ e_1\ \texttt{do}\ e_2\ \texttt{done} : \tau \rangle}$$

$$\textsc{For}\ \frac{\begin{array}{c}\Gamma, \Phi \vdash \langle e_1 : \tau_1 \rangle \\ \Gamma, \Phi \vdash \langle e_2 : \tau_2 \rangle \qquad \Gamma, \Phi \vdash \tau_1 \equiv \texttt{int} \qquad \Gamma, \Phi \vdash \tau_2 \equiv \texttt{int} \\ \Gamma \oplus_{\mathcal{V}} (x, \tau_1), C.\Phi \vdash \langle e_3 : \tau_3 \rangle \qquad \Gamma, \Phi \vdash \tau_3 \equiv \texttt{unit} \qquad \Gamma, \Phi \vdash \tau \equiv \texttt{unit}\end{array}}{\Gamma, \Phi \vdash \langle \texttt{for}\ x = e_1\ \texttt{to}\ e_2\ \texttt{do}\ e_3\ \texttt{done} : \tau \rangle}$$

$$\textsc{Raise}\ \frac{\Gamma,\Phi \vdash \langle e : \tau_e\rangle \qquad \Gamma,\Phi \vdash (\tau_e \equiv \texttt{exn}) \qquad \Gamma,\Phi \vdash (\tau \equiv \texttt{unit})}{\Gamma,\Phi \vdash \langle \textbf{assert}\ e : \tau\rangle}$$

$$\textsc{Assert}\ \frac{\Gamma,\Phi \vdash \langle e : \tau_e\rangle \qquad \Gamma,\Phi \vdash (\tau_e \equiv \texttt{bool}) \qquad \Gamma,\Phi \vdash (\tau \equiv \texttt{unit})}{\Gamma,\Phi \vdash \langle \textbf{assert}\ e : \tau\rangle}$$

$$\textsc{Assert-False}\ \frac{\Gamma,\Phi \vdash \langle false : \tau_e\rangle \qquad \Gamma,\Phi \vdash (\tau_e \equiv \texttt{bool})}{\Gamma,\Phi \vdash \langle \textbf{assert}\ false : \tau\rangle}$$

$$\textsc{Lazy}\ \frac{\Gamma,\Phi \vdash \langle e : \tau_e\rangle \qquad \Gamma,\Phi \vdash \tau \prec \tau_{arg}\ \texttt{Lazy.t} \qquad \Gamma,\Phi \vdash \tau_e \equiv \tau_{arg}}{\Gamma,\Phi \vdash \langle \textbf{lazy}\ e : \tau\rangle}$$

$$\textsc{Variant-Const}\ \frac{\Gamma,\Phi \vdash \tau \prec [(\rho) \,..\, T \,..\, > \,..\, T \,..\,]}{\Gamma,\Phi \vdash \langle `T : \tau\rangle}$$

$$\textsc{Variant}\ \frac{\begin{array}{c}\Gamma,\Phi \vdash \tau \prec [(\rho) ..\ T\ of\ \tau_{arg} \,..\, > \,..\, T \,..\,] \\ \Gamma,\Phi \vdash \langle e : \tau_e\rangle \qquad \Gamma,\Phi \vdash \tau_{arg} \equiv \tau_e\end{array}}{\Gamma,\Phi \vdash \langle `T\ e : \tau\rangle}$$

$$\textsc{Constraint}\ \frac{\begin{array}{c}\Gamma,\Phi,\Sigma \vdash \langle t : \tau_{cstr}\rangle \Rightarrow \mathcal{V} \\ \Gamma,\Phi \vdash \langle e : \tau_e\rangle \qquad \Gamma,\Phi,\Sigma \vdash \tau_e \leq \tau_{cstr} \Rightarrow \theta \qquad \Gamma,\Phi \vdash \tau \equiv \tau_e\end{array}}{\Gamma,\Phi \vdash \langle (e : t) : \tau\rangle}$$

$$\textsc{Newtype}\ \frac{\begin{array}{c} let\ \Gamma' = \Gamma.Types \oplus (\texttt{t} : \mathcal{R}) \qquad \Gamma' \vdash \langle e : \tau_e\rangle \qquad let\ \alpha\ fresh(\Gamma) \\ let\ \theta = [\texttt{t} \to \alpha] \qquad let\ \tau_t = \theta(\tau_e) \qquad \Gamma,\Phi \vdash \tau \equiv \tau_t \qquad \Gamma,\Phi \vdash \tau\ wf\end{array}}{\Gamma,\Phi \vdash \langle \textbf{fun}\ (\textbf{type}\ \texttt{t}) \to e : \tau\rangle}$$

$$\textsc{Let-Module}\ \frac{\begin{array}{c}\Gamma,\Phi \vdash \langle M : \mathcal{M}\rangle \qquad \Gamma.Modules \oplus (\texttt{I}, \mathcal{M}), \Phi \vdash \langle e : \tau_e\rangle \\ \Gamma,\Phi \vdash \tau_e\ wf \qquad \Gamma,\Phi \vdash \tau \equiv \tau_e\end{array}}{\Gamma,\Phi \vdash \langle \textbf{let module}\ \texttt{I} = M\ \textbf{in}\ e : \tau\rangle}$$

0.2 OCaml patterns

$$\textsc{Pat-Record}\ \frac{\begin{array}{c} let\ \tau_{rec} = find\_record(\Gamma, \Phi, \tau) \\ \forall i.\ let\ \tau_{l_i} = find\_label(\Gamma, \Phi, l_i, \tau_{rec}) \\ \Gamma, \Phi, \mathcal{V}, \mathcal{T}\ \vdash \langle p_0 : \tau_0 \rangle \Rightarrow \mathcal{V}_0, \mathcal{T}_0, \Phi_0 \\ \forall i_{\geq 1}.\ \Gamma, \Phi_{i-1}, \mathcal{V}_{i-1}, \mathcal{T}_{i-1}\ \vdash \langle p_i : \tau_i \rangle \Rightarrow \mathcal{V}_i, \mathcal{T}_i, \Phi_i \\ \Gamma, \Phi_0 \vdash \tau_0 \leq \tau_{l_0} \Rightarrow \theta_0 \qquad \forall i_{\geq 1}.\ \Gamma, \Phi_i, \theta_{i-1} \vdash \tau_i \leq \tau_{l_i} \Rightarrow \theta_i \\ let\ \tau_{inst} = \theta_n(\tau_{rec}) \qquad \Gamma, \Phi_n \vdash \tau_{inst} \equiv \tau \end{array}}{\Gamma, \Phi, \mathcal{V}, \mathcal{T}\ \vdash \langle \{ l_0 = p_0;\ ..\ ; l_n = p_n \} : \tau \rangle \Rightarrow \mathcal{V}_n, \mathcal{T}_n, \Phi_n}$$

$$\textsc{Pat-Array}\ \frac{\begin{array}{c} \Gamma, \Phi, \mathcal{V}, \mathcal{T}\ \vdash \langle p_0 : \tau_0 \rangle \Rightarrow \mathcal{V}_0, \mathcal{T}_0, \Phi_0 \\ \forall i_{\geq 1}.\ \Gamma, \Phi_{i-1}, \mathcal{V}_{i-1}, \mathcal{T}_{i-1} \vdash \langle p_i : \tau_i \rangle \Rightarrow \mathcal{V}_i, \mathcal{T}_i, \Phi_i \\ \forall i_{\geq 1}.\ \Gamma, \Phi_i \vdash \tau_{i-1} \equiv \tau_i \qquad \Gamma, \Phi \vdash \tau \prec \tau'\ \texttt{array} \qquad \Gamma, \Phi_0 \vdash \tau_0 \equiv \tau \end{array}}{\Gamma, \Phi, \mathcal{V}, \mathcal{T} \vdash \langle [| p_0;\ ..\ ; p_n |] : \tau \rangle \Rightarrow \mathcal{V}_n, \mathcal{T}_n, \Phi_n}$$

$$\textsc{Pat-Lazy}\ \frac{\begin{array}{c} \Gamma, \Phi, \mathcal{V}, \mathcal{T} \vdash p : \tau_p \Rightarrow \mathcal{V}', \mathcal{T}', \Phi' \\ \Gamma, \Phi \vdash \tau \prec \tau'\ \texttt{Lazy.t} \qquad \Gamma, \Phi' \vdash \tau' \equiv \tau_p \end{array}}{\Gamma, \Phi, \mathcal{V} \vdash \langle \texttt{lazy}\ p : \tau \rangle \Rightarrow \mathcal{V}', \mathcal{T}', \Phi'}$$

$$\textsc{Pat-Variant-Const}\ \frac{\Gamma, \Phi_p \vdash \tau \prec [(\rho)\ T > ..\ ]}{\Gamma, \Phi, \mathcal{V}, \mathcal{T} \vdash \langle `T : \tau \rangle \Rightarrow \mathcal{V}, \mathcal{T}, \Phi}$$

$$\textsc{Pat-Variant}\ \frac{\begin{array}{c} \Gamma, \Phi \vdash \tau \prec [(\rho)\ T\ of\ \tau_{arg} > ..\ ] \\ \Gamma, \Phi, \mathcal{V}, \mathcal{T} \vdash \langle p : \tau_p \rangle \Rightarrow \mathcal{V}', \mathcal{T}', \Phi' \qquad \Gamma, \Phi' \vdash \tau_{arg} \equiv \tau_p \end{array}}{\Gamma, \Phi, \mathcal{V}, \mathcal{T} \vdash \langle `T\ p : \tau \rangle \Rightarrow \mathcal{V}', \mathcal{T}', \Phi'}$$

0.3 OCaml type annotations

These annotations can appear as constraints on expressions, as types of labels from record declaration or arguments of constructor declarations.