React-tRace Core

Toby Ueno

November 11, 2025

Syntax:

$$\begin{aligned}
v &::= x \mid \mathsf{true} \mid \mathsf{false} \mid n \mid s \mid \lambda x : \tau . e \mid \mathcal{C} \mid \mathsf{text}\ v \mid \mathsf{tag}(v, [\overline{v}], [\overline{v}]) \\
&\quad \mid \mathsf{attr}(v, v) \mid \mathsf{onClick}\ v \\
e &::= \mathsf{return}\ v \mid \mathsf{let}\ x = e\ \mathsf{in}\ e \mid v\ v \mid v \oplus v \mid \mathsf{if}\ v\ \mathsf{then}\ e\ \mathsf{else}\ e \\
\hat{e} &::= e \mid \mathsf{let}\ (x, x_{\mathsf{set}}) = \mathsf{useState}^{\ell}(v)\ \mathsf{in}\ \hat{e} \mid \mathsf{let}\ x = e\ \mathsf{in}\ \hat{e} \\
P &::= e \mid \mathsf{let}\ x = e\ \mathsf{in}\ P \mid \mathsf{let}\ \mathcal{C}(x : \tau) = \hat{e}\ \mathsf{in}\ P \\
\tau &::= \mathbf{1} \mid \mathsf{bool} \mid \mathsf{int} \mid \tau \xrightarrow{\epsilon} \tau \mid \mathsf{view} \mid \mathsf{component}\ \tau \mid \mathsf{setter}\ \tau \mid \mathsf{attr} \\
\epsilon &::= \cdot \mid \mathsf{Set}
\end{aligned}$$

Internal representations:

$$\begin{aligned}
t &::= \mathsf{lit}\ v \mid \mathsf{node}(id, v, [\overline{v}], [\overline{t}]) \mid p \\
\pi &::= \{\mathsf{spec} : \langle \mathcal{C}, v \rangle, \mathsf{st} : [\overline{\ell \mapsto v}], \mathsf{child} : t\} \\
m &::= [\overline{p \mapsto \pi}] \\
\mathcal{D} &::= [\overline{\mathcal{C} \mapsto x.\hat{e}}] \\
\mu &::= \mathsf{start}(P) \mid \mathsf{idle} \mid \mathsf{rerender}(m)
\end{aligned}$$

Value typing: $\Gamma \vdash_\Delta v : \tau$

$$\frac{x : \tau \in \Gamma}{\Gamma \vdash_\Delta x : \tau}\ \textsc{TVal-Var} \qquad \frac{}{\Gamma \vdash_\Delta () : \mathbf{1}}\ \textsc{TVal-Unit}$$

$$\frac{}{\Gamma \vdash_\Delta \mathsf{true} : \mathsf{bool}}\ \textsc{TVal-True} \qquad \frac{}{\Gamma \vdash_\Delta \mathsf{false} : \mathsf{bool}}\ \textsc{TVal-False}$$

$$\frac{}{\Gamma \vdash_\Delta n : \mathsf{int}}\ \textsc{TVal-Int} \qquad \frac{}{\Gamma \vdash_\Delta s : \mathsf{string}}\ \textsc{TVal-Str}$$

$$\frac{\Gamma, x : \tau_1 \vdash_\Delta e : \tau_2 \mid \epsilon}{\Gamma \vdash_\Delta \lambda x : \tau_1 . e : \tau_2}\ \textsc{TVal-Lam} \qquad \frac{\mathcal{C} : \mathsf{component}\ \tau \in \Delta}{\Gamma \vdash_\Delta \mathcal{C} : \mathsf{component}\ \tau}\ \textsc{TVal-Comp}$$

$$\frac{\Gamma \vdash_\Delta v : \tau \qquad \tau \in \{\mathsf{bool}, \mathsf{int}, \mathsf{string}\}}{\Gamma \vdash_\Delta \mathsf{text}\ v : \mathsf{view}}\ \textsc{TVal-Text}$$

$$\frac{\Gamma \vdash_\Delta v : \mathsf{string} \qquad (\forall v_a)\ \Gamma \vdash_\Delta v_a : \mathsf{attr} \qquad (\forall v_c)\ \Gamma \vdash_\Delta v_c : \mathsf{view}}{\Gamma \vdash_\Delta \mathsf{tag}(v, [\overline{v_a}], [\overline{v_c}]) : \mathsf{view}}\ \textsc{TVal-Tag}$$

$$\frac{\Gamma \vdash_\Delta v_k : \mathsf{string} \qquad \Gamma \vdash_\Delta v_v : \mathsf{string}}{\Gamma \vdash_\Delta \mathsf{attr}(v_k, v_v) : \mathsf{attr}}\ \textsc{TVal-Attr}$$

$$\frac{\Gamma \vdash_\Delta v : \mathbf{1} \rightarrow^{\epsilon} \mathbf{1}}{\Gamma \vdash_\Delta \mathsf{onClick}\ v : \mathsf{attr}}\ \textsc{TVal-OnClick}$$

Base expression typing: $\Gamma \vdash_\Delta e : x \mid \epsilon$

$$\frac{\Gamma \vdash_\Delta v : \tau}{\Gamma \vdash_\Delta \mathsf{return}\ v : \tau \mid \cdot}\ \textsc{TExp-Ret}$$

$$\frac{\Gamma \vdash_\Delta e_1 : \tau_1 \mid \epsilon_1 \qquad \Gamma, x : \tau_1 \vdash_\Delta e_2 : \tau_2 \mid \epsilon_2}{\Gamma \vdash_\Delta \mathsf{let}\ x = e_1\ \mathsf{in}\ e_2 : \tau_2 \mid \epsilon_1 \circ \epsilon_2}\ \textsc{TExp-Let}$$

$$\frac{\Gamma \vdash_\Delta v_1 : \tau_1 \xrightarrow{\epsilon} \tau_2 \qquad \Gamma \vdash_\Delta v_2 : \tau_1}{\Gamma \vdash_\Delta v_1\ v_2 : \tau_2 \mid \epsilon}\ \textsc{TExp-AppFun}$$

$$\frac{\Gamma \vdash_\Delta v_1 : \mathsf{component}\ \tau \qquad \Gamma \vdash_\Delta v_2 : \tau}{\Gamma \vdash_\Delta v_1\ v_2 : \mathsf{view} \mid \cdot}\ \textsc{TExp-AppComp}$$

$$\frac{\Gamma \vdash_\Delta v_1 : \mathsf{setter}\ \tau \qquad \Gamma \vdash_\Delta v_2 : \tau}{\Gamma \vdash_\Delta v_1\ v_2 : \mathbf{1} \mid \mathsf{Set}}\ \textsc{TExp-AppSet}$$

$$\frac{\Gamma \vdash_\Delta v_1 : \mathsf{int} \qquad \Gamma \vdash_\Delta v_2 : \mathsf{int} \qquad \oplus : \mathbb{Z} \times \mathbb{Z} \to \mathbb{Z}}{\Gamma \vdash_\Delta v_1 \oplus v_2 : \mathsf{int}}\ \textsc{TExp-Bop}$$

$$\frac{\Gamma \vdash_\Delta v : \mathsf{bool} \qquad \Gamma \vdash_\Delta e_1 : \tau \mid \epsilon_1 \qquad \Gamma \vdash_\Delta e_2 : \tau \mid \epsilon_2}{\Gamma \vdash_\Delta \mathsf{if}\ v\ \mathsf{then}\ e_1\ \mathsf{else}\ e_2 : \tau \mid \epsilon_1 \circ \epsilon_2}\ \textsc{TExp-If}$$

Top-level expression typing: $\Gamma \Vdash_\Delta \hat{e} : \tau$

$$\frac{\Gamma \vdash_\Delta e : \tau \mid \cdot}{\Gamma \Vdash_\Delta \hat{e} : \tau}\ \textsc{THat-Lift} \qquad \frac{\Gamma \vdash_\Delta e : \tau_1 \mid \cdot \qquad \Gamma, x : \tau_1 \Vdash_\Delta \hat{e} : \tau_2}{\Gamma \Vdash_\Delta \mathsf{let}\ x = e\ \mathsf{in}\ \hat{e} : \tau_2}\ \textsc{THat-Let}$$

$$\frac{\Gamma \vdash_\Delta v : \tau_1 \qquad \Gamma, x : \tau_1, x_{\mathsf{set}} : \tau_1 \xrightarrow{\mathsf{Set}} \mathbf{1} \Vdash_\Delta \hat{e} : \tau_2}{\Gamma \Vdash_\Delta \mathsf{let}\ (x, x_{\mathsf{set}}) = \mathsf{useState}^{\ell}(v)\ \mathsf{in}\ \hat{e} : \tau_2}\ \textsc{THat-State}$$

Program judgment: $\Gamma \vdash_\Delta P : \mathsf{Prog}$

$$\frac{\Gamma \vdash_\Delta e : \tau \mid \cdot \qquad \Gamma, x : \tau \vdash P : \mathsf{Prog}}{\Gamma \vdash_\Delta \mathsf{let}\ x = e\ \mathsf{in}\ P : \mathsf{Prog}}\ \textsc{TProg-Let}$$

$$\frac{\Gamma, x : \tau \Vdash_\Delta \hat{e} : \mathsf{view} \mid \cdot \qquad \Gamma \vdash P : \mathsf{Prog}}{\Gamma \vdash_\Delta \mathsf{let}\ \mathcal{C}(x : \tau) = \hat{e}\ \mathsf{in}\ P : \mathsf{Prog}}\ \textsc{TProg-Comp}$$

$$\frac{\Gamma \vdash_\Delta e : \mathsf{view} \mid \cdot}{\Gamma \vdash_\Delta \mathsf{run}\ e : \mathsf{Prog}}\ \textsc{TProg-Run}$$

Pure expression semantics: $m\,;\,e \mapsto e'\,;\,m'$

$$\frac{m\,;\,e_1 \mapsto e_1'\,;\,m'}{m\,;\,\textsf{let}\ x = e_1\ \textsf{in}\ e_2 \mapsto \textsf{let}\ x = e_1'\ \textsf{in}\ e_2\,;\,m'}\ \text{E-LetPure}$$

$$\begin{array}{rl}
m\,;\,\textsf{let}\ x = \textsf{return}\ v\ \textsf{in}\ e \mapsto e[v/x]\,;\,m & \text{E-Ret} \\
m\,;\,(\lambda x.e)v \mapsto e[v/x]\,;\,m & \text{E-AppFun} \\
m\,;\,\mathcal{C}\ v \mapsto e[v/x]\,;\,m & \text{E-AppComp} \\
m\,;\,\textsf{set@}_p^{\ell}\ v \mapsto ()\,;\,m' & \text{E-AppSet} \\
(m' = m \mid m[p].st[\ell] = v) & \\
m\,;\,\overline{n_1} \oplus \overline{n_2} \mapsto \textsf{return}\ \overline{n_1 \oplus n_2}\,;\,m & \text{E-Bop} \\
m\,;\,\textsf{if true then}\ e_1\ \textsf{else}\ e_2 \mapsto e_1\,;\,m & \text{E-IfTrue} \\
m\,;\,\textsf{if false then}\ e_1\ \textsf{else}\ e_2 \mapsto e_2\,;\,m & \text{E-IfFalse}
\end{array}$$

Top-level expression semantics: $m\,;\,\hat{e} \mapsto_p^{\phi} \hat{e}'\,;\,m'$

$$\frac{m\,;\,e \mapsto e'\,;\,m'}{m\,;\,e \mapsto_p^{\phi} e'\,;\,m'}\ \text{E-Lift}$$

$$\frac{m\,;\,e \mapsto e'\,;\,m'}{m\,;\,\textsf{let}\ x = e\ \textsf{in}\ \hat{e} \mapsto_p^{\phi} \textsf{let}\ x = e'\ \textsf{in}\ \hat{e}\,;\,m'}\ \text{E-LetTop}$$

$$\begin{array}{rl}
m\,;\,\textsf{let}\ x = \textsf{return}\ v\ \textsf{in}\ \hat{e} \mapsto_p^{\phi} \hat{e}[v/x]\,;\,m & \text{E-RetTop} \\
m\,;\,\textsf{let}\ (x, x_{\textsf{set}}) = \textsf{useState}^{\ell}(v)\ \textsf{in}\ \hat{e} \mapsto_p^{\textsf{Init}} \hat{e}[v/x, \textsf{set@}_p^{\ell}/x_{\textsf{set}}]\,;\,m' & \text{E-StateInit} \\
(m' = m \mid m[p].\textsf{st}[\ell] = v) & \\
m\,;\,\textsf{let}\ (x, x_{\textsf{set}}) = \textsf{useState}^{\ell}(v)\ \textsf{in}\ \hat{e} \mapsto_p^{\textsf{Succ}} \hat{e}[v'/x, \textsf{set@}_p^{\ell}/x_{\textsf{set}}]\,;\,m & \text{E-StateSucc} \\
(v' = m[p].\textsf{st}[\ell]) &
\end{array}$$

Program expression semantics: $m\,;\,\mathcal{D}\,;\,P \Mapsto P'\,;\,\mathcal{D}'\,;\,m'$

$$\frac{m\,;\,e \mapsto e'\,;\,m'}{m\,;\,\mathcal{D}\,;\,\mathsf{run}\ e \mapsto \mathsf{run}\ e'\,;\,\mathcal{D}\,;\,m}\ \textsc{E-Run}$$

$$\frac{m\,;\,e \mapsto e'\,;\,m'}{m\,;\,\mathcal{D}\,;\,\mathsf{let}\ x = e\ \mathsf{in}\ P \mapsto \mathsf{let}\ x = e'\ \mathsf{in}\ P\,;\,\mathcal{D}\,;\,m'}\ \textsc{E-LetProg}$$

$$m\,;\,\mathcal{D}\,;\,\mathsf{let}\ x = \mathsf{return}\ v\ \mathsf{in}\ P \mapsto P[v/x]\,;\,\mathcal{D}\,;\,m \qquad \textsc{E-RetProg}$$

$$m\,;\,\mathcal{D}\,;\,\mathsf{let}\ \mathcal{C}(x) = \hat{e}\ \mathsf{in}\ P \mapsto P\,;\,\mathcal{D}[\mathcal{C} \mapsto x.\hat{e}]\,;\,m \qquad \textsc{E-CompDef}$$

Node initialization: $\mathcal{D}\,;\,m \vdash \mathsf{init}(v) = \langle t, m' \rangle$

$$\frac{}{\mathcal{D}\,;\,m \vdash \mathsf{init}(\mathsf{text}\ v) = \langle \mathsf{lit}\ v, m \rangle}\ \textsc{Init-Lit}$$

$$\frac{m \vdash id\ \mathsf{fresh} \qquad (\forall j)\ \mathcal{D}\,;\,m_j \vdash \mathsf{init}(v_j) = \langle t_j, m_{j+1} \rangle}{\mathcal{D}\,;\,m_0 \vdash \mathsf{init}(\mathsf{tag}(v, [\overline{v_i}], [\overline{v_j}]_{j=0}^{n-1})) = \langle \mathsf{node}(id, v, [\overline{v_i}], [\overline{t_j}]), m_n \rangle}\ \textsc{Init-Node}$$

$$\frac{\begin{array}{c} m_0 \vdash p\ \mathsf{fresh} \qquad \mathcal{D}[\mathcal{C}] = x.\hat{e} \\ m_0[p \mapsto \{\mathsf{spec} : \langle \mathcal{C}, v \rangle, \mathsf{st} : \emptyset, \mathsf{child} : \emptyset\}]\,;\,\hat{e}[v/x] \mapsto_p^{\mathsf{Init}*} \mathsf{return}\ v'\,;\,m_1 \\ \mathcal{D}\,;\,m_1 \vdash \mathsf{init}(v') = \langle t, m_2 \rangle \end{array}}{\mathcal{D}\,;\,m_0 \vdash \mathsf{init}(\langle \mathcal{C}, v \rangle) = \langle p, m_2 \mid m_2[p].\mathsf{child} = t \rangle}\ \textsc{Init-Comp}$$

(Naïve) re-render: $\mathcal{D}\,;\,m_{\mathsf{old}} \vdash \mathsf{check}(m_{\mathsf{new}}, t) = m$

$$\frac{}{\mathcal{D}\,;\,m_{\mathsf{old}} \vdash \mathsf{check}(m_{\mathsf{new}}, \mathsf{lit}\ v) = m_{\mathsf{new}}}\ \textsc{Check-Lit}$$

$$\frac{(0 \le j < n) \quad \mathcal{D}\,;\,m_{\mathsf{old}} \vdash \mathsf{check}(m_j, t_j) = \langle t'_j, m_{j+1} \rangle}{\mathcal{D}\,;\,m_{\mathsf{old}} \vdash \mathsf{check}(m_0, \mathsf{node}(id, v, [\overline{v_i}], [\overline{t_j}])) = m_n}\ \textsc{Check-Node}$$

$$\frac{\begin{array}{c} m_{\mathsf{old}}[p].\mathsf{st} = m_{\mathsf{new}}[p].\mathsf{st} \\ \mathcal{D}\,;\,m_{\mathsf{old}} \vdash \mathsf{check}(m_{\mathsf{new}}, m_{\mathsf{old}}[p].child) = m \end{array}}{\mathcal{D}\,;\,m_{\mathsf{old}} \vdash \mathsf{check}(m_{\mathsf{new}}, p) = m}\ \textsc{Check-PathConst}$$

$$\frac{\begin{array}{c} m_{\mathsf{old}}[p].\mathsf{st} = m_{\mathsf{new}}[p].\mathsf{st} \\ m'[p].\mathsf{spec} = \langle \mathcal{C}, v \rangle \qquad \mathcal{D}[\mathcal{C}] = x.\hat{e} \\ m_{\mathsf{new}}\,;\,\hat{e}[v/x] \mapsto_p^{\mathsf{Succ}*} \mathsf{return}\ v'\,;\,m \\ \mathcal{D}\,;\,m \vdash \mathsf{init}(v') = \langle t, m' \rangle \end{array}}{\mathcal{D}\,;\,m_{\mathsf{old}} \vdash \mathsf{check}(m_{\mathsf{new}}, p) = \langle p, m' \mid m[p].\mathsf{child} = t \rangle}\ \textsc{Check-PathChange}$$

Event loop semantics: $\langle \mathcal{D}, m, t, \mu \rangle \hookrightarrow \langle \mathcal{D}', m', t, \mu' \rangle$

$$\frac{\cdot\,;\,\cdot\,;\,P \Mapsto^{*} \mathsf{run}\ e'\,;\,\mathcal{D}\,;\,m_0 \qquad m_0\,;\,e \mapsto^{*} \mathsf{return}\ v\,;\,m_1 \qquad \mathcal{D}\,;\,m_1 \vdash \mathsf{init}(v) = \langle t, m_2 \rangle}{\langle \cdot, \cdot, \cdot, \mathsf{start}(P) \rangle \hookrightarrow \langle \mathcal{D}, m_2, t, \mathsf{idle} \rangle}\ \textsc{Loop-Start}$$

$$\frac{e \in \mathsf{handler}(m, id, \cdots) \qquad m\,;\,e\ v \mapsto^{*} \mathsf{return}\ ()\,;\,m'}{\langle \mathcal{D}, m, t, \mathsf{idle} \rangle \hookrightarrow \langle \mathcal{D}, m, t, \mathsf{rerender}(m') \rangle}\ \textsc{Loop-Event}$$

$$\frac{D\,;\,m_{\mathsf{old}} \vdash \mathsf{check}(m_{\mathsf{new}}, t) = m}{\langle \mathcal{D}, m_{\mathsf{old}}, t, \mathsf{rerender}(m_{\mathsf{new}}) \rangle \hookrightarrow \langle \mathcal{D}, m, t, \mathsf{idle} \rangle}\ \textsc{Loop-Rerender}$$