Flux: Liquid Types for Rust - Technical Appendix

NICO LEHMANN, UC San Diego, USA
ADAM T. GELLER, University of British Columbia, Canada
NIKI VAZOU, IMDEA, Spain
RANJIT JHALA, UC San Diego, USA

Contents

1 SYNTAX OF $\lambda_{\mathbf{LR}}$.

Syntax of Expressions

$$a, \rho \in \mathit{RefineVar} \qquad \ell \in \mathit{Loc} \qquad x, f \in \mathit{Var} \qquad t \in \mathit{Tag}$$

Expressions	e	$::=$	$\mathbf{let}\ x = e\ \mathbf{in}\ e$	*let-bind*
		$\mid$	$\mathbf{let}\ x = \mathbf{new}(\rho)\ \mathbf{in}\ e$	*let-alloc*
		$\mid$	$\mathbf{if}\ e\ \{e\}\ \mathbf{else}\ \{e\}$	*if-then-else*
		$\mid$	$\mathbf{unpack}(x, a)\ \mathbf{in}\ e$	*unpacking*
		$\mid$	$\mathbf{call}\ e[\overline{r}](\overline{av})$	*function call*
		$\mid$	$p := e$	*assignment*
		$\mid$	$\&\mathbf{strg}\ p \mid \&\mathbf{mut}\ p \mid \&\mathbf{shr}\ p$	*(re)borrows*
		$\mid$	$*p$	*potinter dereference*
		$\mid$	x	*variables*
		$\mid$	v	*values*
Values	v	$::=$	$\mathbf{rec}\ f[\overline{a}](\overline{x})\ :=\ e$	*(recursive) functions*
		$\mid$	$\mathbf{true} \mid \mathbf{false}$	*booleans*
		$\mid$	$0, \pm 1, \ldots$	*integers*
		$\mid$	☣	*uninitialized memory*
		$\mid$	$\mathrm{ptr}(\ell, t)$	*tagged pointers*
		$\mid$	$\mathbf{vec}_{\tau}(n, v')$	*vector values*
		$\mid$	$\mathbf{Vec}_{\tau}\mathbf{::new}$	*vector new*
		$\mid$	$\mathbf{Vec}_{\tau}\mathbf{::push}$	*vector push*
		$\mid$	$\mathbf{Vec}_{\tau}\mathbf{::index\_mut}$	*vector mutable indexing*
Logic	r	$::=$	$\mathbf{true} \mid \mathbf{false} \mid 0, \pm 1, \ldots$	*constants*
		$\mid$	$a \mid r = r$	*variables and equality*
		$\mid$	$\neg r \mid r[\wedge, \vee]r \mid r[+, -, *]r$	*bool and arithmetic*
		$\mid$	ℓ	*locations*
A-values	av	$::=$	$v \mid x$	
Place	p	$::=$	$x \mid \mathrm{ptr}(\ell, t)$	

Syntax of Types

Types	τ	$:=$	$B[r]$	*indexed type*
		$\mid$	$\{a.\ B[a] \mid r\}$	*existential type*
		$\mid$	$\mathbf{ptr}(\eta)$	*pointer to location η*
		$\mid$	$\&_\mu\ \tau$	*borrowed reference (mutable or shared)*
		$\mid$	$\lightning$	*uninitialized memory of size n*
		$\mid$	$\forall \overline{a : \sigma}.\ \mathbf{fn}(\mathrm{T}; \overline{\tau}) \to \tau / \mathrm{T}$	
Base Types	B	$::=$	$\mathbf{int} \mid \mathbf{bool} \mid \mathbf{Vec}_\tau$	*integers, booleans, and vectors*
Modifier	μ	$::=$	$\mathbf{mut} \mid \mathbf{shr}$	*mutable or shared*
Locations	η	$:=$	$\rho \mid \ell$	*a concrete or abstract location*
Sorts	σ	$:=$	$\mathbf{int} \mid \mathbf{bool} \mid \mathbf{loc}$	
Refinement contexts	Δ	$:=$	$\emptyset \mid \Delta, a : \sigma \mid \Delta, r$	
Location contexts	T	$:=$	$\emptyset \mid \mathrm{T}, \eta \mapsto \tau$	
Value contexts	Γ	$:=$	$\emptyset \mid \Gamma, x : \tau$	
Dynamic Location contexts	Σ	$:=$	$\emptyset \mid \Sigma, (\ell, t) \mapsto \tau$	

2 AUXILIARY DEFINITIONS

2.1 Free Refinement Variables

Expressions

$\mathrm{FV}(\mathbf{let}\ x = e_1\ \mathbf{in}\ e_2)$	$=$	$\mathrm{FV}(e_1) \cup \mathrm{FV}(e_2)$
$\mathrm{FV}(\mathbf{let}\ x = \mathbf{new}(\rho)\ \mathbf{in}\ e)$	$=$	$\mathrm{FV}(e) \setminus \{\rho\}$
$\mathrm{FV}(\mathbf{if}\ e_1\ \{e_2\}\ \mathbf{else}\ \{e_3\})$	$=$	$\mathrm{FV}(e_1) \cup \mathrm{FV}(e_2) \cup \mathrm{FV}(e_3)$
$\mathrm{FV}(\mathbf{unpack}(x, a)\ \mathbf{in}\ e)$	$=$	$\mathrm{FV}(e) \setminus \{a\}$
$\mathrm{FV}(\mathbf{call}\ e[\overline{r}](\overline{av}))$	$=$	$\mathrm{FV}(e) \cup \bigcup_i \mathrm{FV}(r_i)$
$\mathrm{FV}(p := e)$	$=$	$\mathrm{FV}(e)$
$\mathrm{FV}([\&\mathbf{strg}\ p, \&\mathbf{mut}\ p, \&\mathbf{shr}\ p])$	$=$	$\{\}$
$\mathrm{FV}(*p)$	$=$	$\{\}$
$\mathrm{FV}(x)$	$=$	$\{\}$
$\mathrm{FV}(\mathbf{rec}\ f[\overline{a}](\overline{x})\ :=\ e)$	$=$	$\mathrm{FV}(e) \setminus \{\overline{a}\}$
$\mathrm{FV}([\mathbf{true}, \mathbf{false}])$	$=$	$\{\}$
$\mathrm{FV}(0, \pm 1, \dots)$	$=$	$\{\}$
$\mathrm{FV}(☣)$	$=$	$\{\}$
$\mathrm{FV}(\mathrm{ptr}(\ell, t))$	$=$	$\{\}$
$\mathrm{FV}(\mathbf{vec}_\tau(n, v))$	$=$	$\mathrm{FV}(v)$

Types

$$\begin{aligned}
\mathrm{FV}(B[r]) &= \mathrm{FV}(r)\\
\mathrm{FV}(\{a.\ B[a] \mid r\}) &= \mathrm{FV}(r) \setminus \{a\}\\
\mathrm{FV}(\mathbf{ptr}(\eta)) &= \mathrm{FV}(\eta)\\
\mathrm{FV}(\&_{\mu}\ \tau) &= \mathrm{FV}(\tau)\\
\mathrm{FV}(\lightning) &= \{\}\\
\mathrm{FV}(\forall \overline{a : \sigma}.\ \mathbf{fn}(\mathrm{T}_i; \overline{\tau}) \to \tau_o/\mathrm{T}_o) &= (\mathrm{FV}(r) \cup \mathrm{FV}(\mathrm{T}_i) \textstyle\bigcup_i \mathrm{FV}(\tau_i) \cup \mathrm{FV}(\tau_o) \cup \mathrm{FV}(\mathrm{T}_o)) \setminus \{\overline{a}\}
\end{aligned}$$

Refinements

$$\begin{aligned}
\mathrm{FV}(a) &= \{a\}\\
\mathrm{FV}(r_1[=, \wedge, +, -, *]r_2) &= \mathrm{FV}(r_1) \cup \mathrm{FV}(r_2)\\
\mathrm{FV}(\neg r) &= \mathrm{FV}(r)\\
\mathrm{FV}([\mathbf{true}, \mathbf{false}, \ell]) &= \{\}\\
\mathrm{FV}(n) &= \{\} \quad n \in \mathbb{Z}
\end{aligned}$$

Location Contexts

$$\begin{aligned}
\mathrm{FV}(\emptyset) &= \emptyset\\
\mathrm{FV}(\mathrm{T}, \eta \mapsto \tau) &= \mathrm{FV}(\mathrm{T}) \cup \mathrm{FV}(\tau) \cup \mathrm{FV}(\eta)
\end{aligned}$$

Value Contexts

$$\begin{aligned}
\mathrm{FV}(\emptyset) &= \emptyset\\
\mathrm{FV}(\Gamma, x : \tau) &= \mathrm{FV}(\Gamma) \cup \mathrm{FV}(\tau)
\end{aligned}$$

Dynamic Location Contexts

$$\begin{aligned}
\mathrm{FV}(\emptyset) &= \emptyset\\
\mathrm{FV}(\Sigma, (\ell, t) \mapsto \tau) &= \mathrm{FV}(\Sigma) \cup \mathrm{FV}(\tau)
\end{aligned}$$

2.2 Type Substitution

Types

$$\begin{array}{lcl}
B[r][r_a/a] & = & B[r[r_a/a]] \\
\{b.\ B[b] \mid r\}[r_a/a] & = & \begin{cases} \{b.\ B[b] \mid r\}, & \text{if } b = a \\ \{b.\ B[b] \mid r[r_a/a]\}, & \text{otherwise} \end{cases} \\
\mathbf{ptr}(\eta)[r_a/a] & = & \begin{cases} \mathbf{ptr}(r_a), & \text{if } a = \eta \\ \mathbf{ptr}(\eta), & \text{otherwise} \end{cases} \\
(\&_\mu\ \tau)[r_a/a] & = & \&_\mu\ \tau[r_a/a] \\
\lightning[r_a/a] & = & \lightning \\
(\forall \overline{a:\sigma}.\ \mathbf{fn}(\mathrm{T}_i; \overline{\tau}) \to \tau_o/\mathrm{T}_o)[r_a/a] & = & \begin{cases} \forall \overline{a:\sigma}.\ \mathbf{fn}(\mathrm{T}_i;\overline{\tau}) \to \tau_o/\mathrm{T}_o & \text{if } a \in \{\overline{a}\} \\ \forall \overline{a:\sigma}.\ \mathbf{fn}(\mathrm{T}_i[r_a/a]; \overline{\tau[r_a/a]}) \to & \\ \qquad \tau_o[r_a/a]/\mathrm{T}_o[r_a/a], & \text{otherwise} \end{cases}
\end{array}$$

Logical Expressions

$$\begin{array}{lcl}
b[r_a/a] & = & \begin{cases} r_a, & \text{if } a = b \\ b, & \text{otherwise} \end{cases} \\
(r_1[=,\wedge,+,-,*]r_2)[r_a/a] & = & r_1[r_a/a][=,\wedge,+,-,*]r_2[r_a/a] \\
(\neg r)[r_a/a] & = & \neg r[r_a/a] \\
n[r_a/a] & = & n \quad n \in \mathbb{Z} \\
\ell[r_a/a] & = & \ell \\
\mathbf{true}[r_a/a] & = & \mathbf{true} \\
\mathbf{false}[r_a/a] & = & \mathbf{false}
\end{array}$$

Location Contexts

$$\begin{array}{lcl}
\emptyset[r_a/a] & = & \emptyset \\
(\mathrm{T}, \eta \mapsto \tau)[r_a/a] & = & \mathrm{T}[r_a/a], \eta[r_a/a] \mapsto \tau[r_a/a]
\end{array}$$

Value Contexts

$$\begin{array}{lcl}
\emptyset[r_a/a] & = & \emptyset \\
(\Gamma, x{:}\tau)[r_a/a] & = & \Gamma[r_a/a], x{:}\tau[r_a/a]
\end{array}$$

Dynamic Location Contexts

$$\begin{array}{lcl}
\emptyset[r_a/a] & = & \emptyset \\
(\Sigma, (\ell, t) \mapsto \tau)[r_a/a] & = & \Sigma[r_a/a], (\ell, t) \mapsto \tau[r_a/a]
\end{array}$$

Refinement Contexts

$$\begin{array}{lcl}
\emptyset[r_a/a] & = & \emptyset \\
(\Delta, a' : \sigma)[r_a/a] & = & \Delta[r_a/a], a' : \sigma, \text{if } a \neq a' \\
(\Delta, r)[r_a/a] & = & \Delta[r_a/a], r[r_a/a]
\end{array}$$

2.3 Expression Substitution (values)

$$
\begin{array}{lcl}
(\textbf{unpack}(x, a)\ \textbf{in}\ e)[v_y/y] & = & \begin{cases} e[v_y/y][[\![v_y]\!]/a], & \text{if } x = y \\ \textbf{unpack}(x, a)\ \textbf{in}\ e[v_y/y], & \text{otherwise} \end{cases} \\
(\textbf{let}\ x = e_x\ \textbf{in}\ e)[v_y/y] & = & \begin{cases} \textbf{let}\ x = e_x[v_y/y]\ \textbf{in}\ e, & \text{if } x = y \\ \textbf{let}\ x = e_x[v_y/y]\ \textbf{in}\ e[v_y/y], & \text{otherwise} \end{cases} \\
(\textbf{let}\ x = \textbf{new}(\rho)\ \textbf{in}\ e)[v_y/y] & = & \begin{cases} \textbf{let}\ x = \textbf{new}(\rho)\ \textbf{in}\ e, & \text{if } x = y \\ \textbf{let}\ x = \textbf{new}(\rho)\ \textbf{in}\ e[v_y/y], & \text{otherwise} \end{cases} \\
x[v_y/y] & = & \begin{cases} v_y, & \text{if } x = y \\ x, & \text{otherwise} \end{cases} \\
(\textbf{if}\ e\ \{e_1\}\ \textbf{else}\ \{e_2\})[v_y/y] & = & \textbf{if}\ e[v_y/y]\ \{e_1[v_y/y]\}\ \textbf{else}\ \{e_2[v_y/y]\} \\
(S; e)[v_y/y] & = & S[v_y/y]; e[v_y/y] \\
(\textbf{call}\ e[\overline{r}](\overline{av}))[v_y/y] & = & \textbf{call}\ e[v_y/y][\overline{r}](\overline{av[v_y/y]}) \\
(\textbf{rec}\ f[\overline{a}](\overline{x})\ :=\ e)[v_y/y] & = & \begin{cases} \textbf{rec}\ f[\overline{a}](\overline{x})\ :=\ e[v_y/y], & \text{if } y \neq f \text{ and } y \neq x_i \\ \textbf{rec}\ f[\overline{a}](\overline{x})\ :=\ e, & \text{otherwise} \end{cases} \\
v[v_y/y] & = & v \\
p := e[v_y/y] & = & p[v_y/y] := e[v_y/y] \\
(\&\textbf{strg}\ p)[v_y/y] & = & \&\textbf{strg}\ p[v_y/y] \\
(\&\textbf{mut}\ p)[v_y/y] & = & \&\textbf{mut}\ p[v_y/y] \\
(\&\textbf{shr}\ p)[v_y/y] & = & \&\textbf{shr}\ p[v_y/y] \\
(*p)[v_y/y] & = & *p[v_y/y]
\end{array}
$$

2.4 Expressions Substitution (refinements)

$$
\begin{array}{lcl}
(\mathbf{unpack}(x, b)\ \mathbf{in}\ e)[r_a/a] & = & \begin{cases} \mathbf{unpack}(x, b)\ \mathbf{in}\ e[v_a/a], & \text{if } a \neq b \\ \mathbf{unpack}(x, b)\ \mathbf{in}\ e & \text{otherwise} \end{cases} \\
(\mathbf{let}\ x = e_x\ \mathbf{in}\ e)[r_a/a] & = & \mathbf{let}\ x = e_x[r_a/a]\ \mathbf{in}\ e[r_a/a] \\
(\mathbf{let}\ x = \mathbf{new}(\rho)\ \mathbf{in}\ e)[r_a/a] & = & \begin{cases} \mathbf{let}\ x = \mathbf{new}(\rho)\ \mathbf{in}\ e[r_a/a], & \text{if } a \neq \rho \\ \mathbf{let}\ x = \mathbf{new}(\rho)\ \mathbf{in}\ e[r_a/a], & \text{otherwise} \end{cases} \\
x[r_a/a] & = & x \\
(\mathbf{if}\ e\ \{e_1\}\ \mathbf{else}\ \{e_2\})[r_a/a] & = & \mathbf{if}\ e[r_a/a]\ \{e_1[r_a/a]\}\ \mathbf{else}\ \{e_2[r_a/a]\} \\
(S; e)[r_a/a] & = & S[r_a/a]; e[r_a/a] \\
(\mathbf{call}\ e[\overline{r}](\overline{av}))[r_a/a] & = & \mathbf{call}\ e[r_a/a][\overline{r[r_a/a]}](\overline{av[r_a/a]}) \\
(\mathbf{rec}\ f[\overline{b}](\overline{x})\ := \ e)[r_a/a] & = & \begin{cases} \mathbf{rec}\ f[\overline{b}](\overline{x})\ :=\ e[r_a/a], & \text{if } a \neq b_i \\ \mathbf{rec}\ f[\overline{b}](\overline{x})\ :=\ e, & \text{otherwise} \end{cases} \\
v[r_a/a] & = & v \\
p := e[r_a/a] & = & p[r_a/a] := e[r_a/a] \\
(\&\mathbf{strg}\ p)[r_a/a] & = & \&\mathbf{strg}\ p[r_a/a] \\
(\&\mathbf{mut}\ p)[r_a/a] & = & \&\mathbf{mut}\ p[r_a/a] \\
(\&\mathbf{shr}\ p)[r_a/a] & = & \&\mathbf{shr}\ p[r_a/a] \\
(*p)[r_a/a] & = & *p[r_a/a]
\end{array}
$$

2.5 Type Sorts and Value Indices

The following definitions connect values and types with the refinement logic. The function $\texttt{sort}(B)$ associates a *refinement sort* to a base type B. Values of type B must be indexed with refinements of that sort. The function $[\![\cdot]\!]$ maps values to a *refinement index*. This function is partial as only values of a base type can be refined. Integers and booleans are mapped to the exact value in the logic. Vectors are mapped to an index representing its length.

$$
\begin{array}{lcl}
\texttt{sort}(\mathbf{int}) & = & \mathbf{int} \\
\texttt{sort}(\mathbf{bool}) & = & \mathbf{bool} \\
\texttt{sort}(\mathbf{Vec}_\tau) & = & \mathbf{int} \\
[\![\mathbf{true}]\!] & = & \mathbf{true} \\
[\![\mathbf{false}]\!] & = & \mathbf{false} \\
[\![n]\!] & = & n, \text{ for } n \in \mathbb{Z} \\
[\![\mathbf{vec}_\tau(n, v)]\!] & = & n, \text{ for } n \in \mathbb{Z}
\end{array}
$$

3 DECLARATIVE JUDGEMENTS OF $\lambda_{\mathbf{LR}}$.

3.1 Well-formedness

The well-formedness judgments ($\vdash_{\mathsf{wf}}$) document the binding struct of the grammar as well as well-sortedness of refinements. In what follows, we assume well-formed implicitly in other judgments.

Well-formed Types $\boxed{\Delta \vdash_{\mathsf{wf}} \tau}$

$$\frac{\Delta \vdash r : \mathtt{sort}(B)}{\Delta \vdash_{\mathsf{wf}} B[r]}\ \textsc{WF-Idx} \qquad \frac{\Delta, a : \mathtt{sort}(B) \vdash r : \mathbf{bool}}{\Delta \vdash_{\mathsf{wf}} \{a.\ B[a] \mid r\}}\ \textsc{WF-Ex} \qquad \frac{\Delta \vdash \eta : \mathbf{loc}}{\Delta \vdash_{\mathsf{wf}} \mathbf{ptr}(\eta)}\ \textsc{WF-Ptr} \qquad \frac{\Delta \vdash_{\mathsf{wf}} \tau}{\Delta \vdash_{\mathsf{wf}} \&_{\mu}\ \tau}\ \textsc{WF-Ref} \qquad \frac{}{\Delta \vdash_{\mathsf{wf}} ⚡}\ \textsc{WF-Mem}$$

$$\frac{\begin{array}{c}\Delta, \overline{a : \sigma} \vdash r : \mathbf{bool} \\ \mathrm{dom}(\mathrm{T}_o) \subseteq \mathrm{dom}(\mathrm{T}_i) \qquad \Delta', \overline{a : \sigma} \vdash_{\mathsf{wf}} \mathrm{T}_i \qquad \forall i.\Delta', \overline{a : \sigma} \vdash_{\mathsf{wf}} \tau_i \qquad \Delta', \overline{a : \sigma} \vdash_{\mathsf{wf}} \tau_o \qquad \Delta', \overline{a : \sigma} \vdash_{\mathsf{wf}} \mathrm{T}_o\end{array}}{\Delta \vdash_{\mathsf{wf}} \forall \overline{a : \sigma}.\ \mathbf{fn}(\mathrm{T}_i; \overline{\tau}) \rightarrow \tau_o / \mathrm{T}_o}\ \textsc{WF-Fun}$$

Well-formed Value Contexts $\boxed{\Delta \vdash_{\mathsf{wf}} \Gamma}$

$$\frac{\Delta \vdash_{\mathsf{wf}} \tau \qquad \Delta \vdash_{\mathsf{wf}} \Gamma}{\Delta \vdash_{\mathsf{wf}} \Gamma, x{:}\tau}\ \textsc{WF-Bind} \qquad \frac{}{\Delta \vdash_{\mathsf{wf}} \emptyset}\ \textsc{WF-Emp}$$

Well-formed Dynamic Contexts $\boxed{\Delta \vdash_{\mathsf{wf}} \Sigma}$

$$\frac{\Delta \vdash_{\mathsf{wf}} \tau \qquad \Delta \vdash_{\mathsf{wf}} \Sigma}{\Delta \vdash_{\mathsf{wf}} \Sigma, (\ell, t) \mapsto \tau}\ \textsc{WF-Bind} \qquad \frac{}{\Delta \vdash_{\mathsf{wf}} \emptyset}\ \textsc{WF-Emp}$$

Well-formed Location contexts $\boxed{\Delta \vdash_{\mathsf{wf}} \mathrm{T}}$

$$\frac{\Delta \vdash_{\mathsf{wf}} \tau \qquad \Delta \vdash_{\mathsf{wf}} \mathrm{T} \qquad \Delta \vdash \eta : \mathbf{loc} \qquad \eta \notin \mathrm{dom}(\mathrm{T})}{\Delta \vdash_{\mathsf{wf}} \mathrm{T}, \eta \mapsto \tau}\ \textsc{WF-Bind} \qquad \frac{}{\Delta \vdash_{\mathsf{wf}} \emptyset}\ \textsc{WF-Emp}$$

Well-formed Refinements Contexts $\boxed{\vdash_{\mathsf{wf}} \Delta}$

$$\vdash_{\mathsf{wf}} \emptyset \qquad \frac{\vdash_{\mathsf{wf}} \Delta \qquad \Delta \vdash r : \mathbf{bool}}{\vdash_{\mathsf{wf}} \Delta, r} \qquad \frac{\vdash_{\mathsf{wf}} \Delta \qquad a \notin \mathrm{dom}(\Delta)}{\vdash_{\mathsf{wf}} \Delta, a : \sigma}$$

Well-sorted Refinements $\boxed{\Delta \vdash r : \sigma}$

$$\frac{a : \sigma \in \Delta}{\Delta \vdash a : \sigma} \qquad \frac{\Delta \vdash r_1 : \sigma \qquad \Delta \vdash r_2 : \sigma}{\Delta \vdash r_1 = r_2 : \mathbf{bool}} \qquad \Delta \vdash \ell : \mathbf{loc} \qquad \frac{\begin{array}{c}\oplus \in \{+, -, *\} \\ \Delta \vdash r : \mathbf{int} \qquad \Delta \vdash r_2 : \mathbf{int}\end{array}}{\Delta \vdash r_1 \oplus r_2 : \mathbf{int}}$$

$$\frac{\begin{array}{c}\oplus \in \{\land, \lor\} \\ \Delta \vdash r : \mathbf{bool} \qquad \Delta \vdash r_2 : \mathbf{bool}\end{array}}{\Delta \vdash r_1 \oplus r_2 : \mathbf{bool}} \qquad \frac{\Delta \vdash r : \mathbf{bool}}{\Delta \vdash \neg r : \mathbf{bool}} \qquad \frac{n \in \mathbb{Z}}{\Delta \vdash n : \mathbf{int}}$$

3.2 Well-typed Expressions

Well-typed Expressions $\boxed{\Delta \mid \Gamma; \mathrm{T} \mid \Sigma \vdash e : \tau \dashv \mathrm{T}}$

T-let

$$\frac{\begin{array}{c}\Delta \mid \Gamma; \mathrm{T}_i \mid \Sigma \vdash e_x : \tau_x \dashv \mathrm{T} \\ \Delta \mid \Gamma, x{:}\tau_x; \mathrm{T} \mid \Sigma \vdash e : \tau \dashv \mathrm{T}_o\end{array}}{\Delta \mid \Gamma; \mathrm{T}_i \mid \Sigma \vdash \mathbf{let}\ x = e_x\ \mathbf{in}\ e : \tau \dashv \mathrm{T}_o}$$

T-new

$$\frac{\begin{array}{c}\Delta \vdash_{\mathsf{wf}} \tau \qquad \Delta \vdash_{\mathsf{wf}} \mathrm{T}_o \\ \Delta, \rho : \mathbf{loc} \mid \Gamma, x{:}\mathbf{ptr}(\rho); \mathrm{T}_i, \rho \mapsto \lightning \mid \Sigma \vdash e : \tau \dashv \mathrm{T}_o\end{array}}{\Delta \mid \Gamma; \mathrm{T}_i \mid \Sigma \vdash \mathbf{let}\ x = \mathbf{new}(\rho)\ \mathbf{in}\ e : \tau \dashv \mathrm{T}_o}$$

T-sub

$$\frac{\begin{array}{c}\Delta \mid \Gamma; \mathrm{T}_i \mid \Sigma \vdash e : \tau_1 \dashv \mathrm{T} \\ \Delta \vdash \tau_1 \preccurlyeq \tau \qquad \Delta \vdash \mathrm{T} \Rightarrow \mathrm{T}_o\end{array}}{\Delta \mid \Gamma; \mathrm{T}_i \mid \Sigma \vdash e : \tau \dashv \mathrm{T}_o}$$

T-if

$$\frac{\begin{array}{c}\Delta \mid \Gamma; \mathrm{T}_i \mid \Sigma \vdash e : \mathbf{bool}[r] \dashv \mathrm{T}_o \\ \Delta, r \mid \Gamma; \mathrm{T}_o \mid \Sigma \vdash e_1 : \tau \dashv \mathrm{T} \\ \Delta, \neg r \mid \Gamma; \mathrm{T}_o \mid \Sigma \vdash e_2 : \tau \dashv \mathrm{T}\end{array}}{\Delta \mid \Gamma; \mathrm{T}_i \mid \Sigma_i \vdash \mathbf{if}\ e\ \{e_1\}\ \mathbf{else}\ \{e_2\} : \tau \dashv \mathrm{T}}$$

T-unpack

$$\frac{\Delta, a : \mathtt{sort}(B), r \mid \Gamma_1, x{:}B[a], \Gamma_2; \mathrm{T}_i \mid \Sigma \vdash e : \tau \dashv \mathrm{T}_o}{\Delta \mid \Gamma_1, x{:}\{a.\ B[a] \mid r\}, \Gamma_2; \mathrm{T}_i \mid \Sigma \vdash \mathbf{unpack}(x, a)\ \mathbf{in}\ e : \tau \dashv \mathrm{T}_o}$$

T-call

$$\frac{\begin{array}{c}\forall i.\Delta \mid \Gamma; \mathrm{T} \mid \Sigma \vdash av_i : \theta \cdot \tau_i \dashv \mathrm{T} \qquad \Delta \mid \Gamma; \mathrm{T} \mid \Sigma \vdash e : \forall \overline{a : \sigma}.\ \mathbf{fn}(\mathrm{T}_i; \overline{\tau}) \to \tau_o/\mathrm{T}_o \dashv \mathrm{T}_1, \mathrm{T}_2 \\ \theta = [\overline{r}/\overline{a}] \qquad \Delta \vdash \mathrm{T}_1 \Rightarrow \theta \cdot \mathrm{T}_i \qquad \forall i.\Delta \vdash r_i : \sigma_i \qquad \Delta \models \theta \cdot r \qquad \Delta \vdash_{\mathsf{wf}} \Sigma\end{array}}{\Delta \mid \Gamma; \mathrm{T} \mid \Sigma \vdash \mathbf{call}\ e[\overline{r}](\overline{av}) : \theta \cdot \tau_o \dashv \theta \cdot \mathrm{T}_o, \mathrm{T}_2}$$

Well-typed Expressions (assignment) $\boxed{\Delta \mid \Gamma; \mathrm{T} \mid \Sigma \vdash e : \tau \dashv \mathrm{T}}$

$$\textsc{T-ass}\quad \frac{\Delta \mid \Gamma; \mathrm{T}_i \mid \Sigma \vdash e : \tau_v \dashv \mathrm{T}_o \qquad \Delta \mid \Gamma; \mathrm{T}_o \mid \Sigma \vdash p : \&_{\mathbf{mut}}\tau \dashv \mathrm{T}_o \qquad \Delta \vdash \tau_v \preccurlyeq \tau}{\Delta \mid \Gamma; \mathrm{T}_i \mid \Sigma \vdash p := e : \lightning \dashv \mathrm{T}_o}$$

$$\textsc{T-ass-strg}\quad \frac{\Delta \mid \Gamma; \mathrm{T}_i \mid \Sigma \vdash e : \tau \dashv \mathrm{T}_o \qquad \Delta \mid \Gamma; \mathrm{T}_o \mid \Sigma \vdash p : \mathbf{ptr}(\eta) \dashv \mathrm{T}_o}{\Delta \mid \Gamma; \mathrm{T}_i \mid \Sigma \vdash p := e : \lightning \dashv \mathrm{T}_o[\eta \mapsto \tau]}$$

Well-typed Expressions (values) $\boxed{\Delta \mid \Gamma; \mathrm{T} \mid \Sigma \vdash e : \tau \dashv \mathrm{T}}$

$$\textsc{T-var}\quad \frac{x : \tau \in \Gamma}{\Delta \mid \Gamma; \mathrm{T} \mid \Sigma \vdash x : \tau \dashv \mathrm{T}}$$

$$\textsc{T-true}\quad \Delta \mid \Gamma; \mathrm{T} \mid \Sigma \vdash \mathbf{true} : \mathbf{bool}[\mathbf{true}] \dashv \mathrm{T}$$

$$\textsc{T-false}\quad \Delta \mid \Gamma; \mathrm{T} \mid \Sigma \vdash \mathbf{false} : \mathbf{bool}[\mathbf{false}] \dashv \mathrm{T}$$

$$\textsc{T-int}\quad \Delta \mid \Gamma; \mathrm{T} \mid \Sigma \vdash i : \mathbf{int}[i] \dashv \mathrm{T}$$

$$\textsc{T-fun}\quad \frac{\Delta, \overline{a : \sigma}, r \mid \Gamma, \overline{x : \tau}, f : \forall \overline{a : \sigma}.\ \mathbf{fn}(\mathrm{T}_i; \overline{\tau}) \to \tau / \mathrm{T}_o; \mathrm{T}_i \mid \Sigma \vdash e : \tau \dashv \mathrm{T}_o}{\Delta \mid \Gamma; \mathrm{T} \mid \Sigma \vdash \mathbf{rec}\ f[\overline{a}](\overline{x})\ := \ e : \forall \overline{a : \sigma}.\ \mathbf{fn}(\mathrm{T}_i; \overline{\tau}) \to \tau / \mathrm{T}_o \dashv \mathrm{T}}$$

Well-typed Expressions (vectors) $\boxed{\Delta \mid \Gamma; \mathrm{T} \mid \Sigma \vdash e : \tau \dashv \mathrm{T}}$

$$\textsc{T-ptr}\quad \Delta \mid \Gamma; \mathrm{T} \mid \Sigma \vdash \mathrm{ptr}(\ell, t) : \Sigma(\ell, t) \dashv \mathrm{T}$$

$$\textsc{T-mem}\quad \Delta \mid \Gamma; \mathrm{T} \mid \Sigma \vdash \text{☣} : \lightning \dashv \mathrm{T}$$

$$\textsc{T-vec-new}\quad \Delta \mid \Gamma; \mathrm{T} \mid \Sigma \vdash \mathbf{Vec}_\tau\mathbf{::new} : \mathrm{T} \dashv \mathbf{fn}() \to \mathbf{Vec}_\tau[0]$$

$$\textsc{T-vec-push}\quad \Delta \mid \Gamma; \mathrm{T} \mid \Sigma \vdash \mathbf{Vec}_\tau\mathbf{::push} : \forall \rho : \mathbf{loc}.\ \mathbf{fn}([\rho \mapsto \mathbf{Vec}_\tau[n]]; \mathbf{ptr}(\rho)) \to \lightning / [\rho \mapsto \mathbf{Vec}_\tau[n+1]] \dashv \mathrm{T}$$

$$\textsc{T-vec-index-mut}\quad \Delta \mid \Gamma; \mathrm{T} \mid \Sigma \vdash \mathbf{Vec}_\tau\mathbf{::index\_mut} : \forall a : \mathbf{int}, b : \mathbf{int}.\ \mathbf{fn}(\emptyset; \&_{\mathbf{mut}}\mathbf{Vec}_\tau[a], \mathbf{int}[b]) \to \&_{\mathbf{mut}}\tau / \emptyset \dashv \mathrm{T}$$

$$\textsc{T-vec}\quad \frac{n \geq 0 \qquad n > 0 \implies v = \mathrm{ptr}(\ell, t) \land \forall i \in [0, n).\ \Sigma(\ell + i, t) = \&_{\mathbf{mut}}\tau}{\Delta \mid \Gamma; \mathrm{T} \mid \Sigma \vdash \mathbf{vec}_\tau(n, v) : \mathbf{Vec}_\tau[n] \dashv \mathrm{T}}$$

Well-typed Expressions (borrows) $\boxed{\Delta \mid \Gamma; \mathrm{T} \mid \Sigma \vdash e : \tau \dashv \mathrm{T}}$

T-BSTRG
$$\frac{\Delta \mid \Gamma; \mathrm{T} \mid \Sigma \vdash p : \mathbf{ptr}(\eta) \dashv \mathrm{T}}{\Delta \mid \Gamma; \mathrm{T} \mid \Sigma \vdash \&\mathbf{strg}\, p : \mathbf{ptr}(\eta) \dashv \mathrm{T}}$$

T-BSMUT
$$\frac{\Delta \mid \Gamma; \mathrm{T} \mid \Sigma \vdash p : \mathbf{ptr}(\eta) \dashv \mathrm{T} \qquad \Delta \vdash \mathrm{T}(\eta) \preccurlyeq \tau}{\Delta \mid \Gamma; \mathrm{T} \mid \Sigma \vdash \&\mathbf{mut}\, p : \&_{\mathbf{mut}}\tau \dashv \mathrm{T}[\eta \mapsto \tau]}$$

T-BMUT
$$\frac{\Delta \mid \Gamma; \mathrm{T} \mid \Sigma \vdash p : \&_{\mathbf{mut}}\tau \dashv \mathrm{T}}{\Delta \mid \Gamma; \mathrm{T} \mid \Sigma \vdash \&\mathbf{mut}\, p : \&_{\mathbf{mut}}\tau \dashv \mathrm{T}}$$

T-BSHR
$$\frac{\Delta \mid \Gamma; \mathrm{T} \mid \Sigma \vdash p : \&_{\mu}\tau' \dashv \mathrm{T} \qquad \Delta \vdash \tau' \preccurlyeq \tau}{\Delta \mid \Gamma; \mathrm{T} \mid \Sigma \vdash \&\mathbf{shr}\, p : \&_{\mathbf{shr}}\tau \dashv \mathrm{T}}$$

Well-typed Expressions (dereference) $\boxed{\Delta \mid \Gamma; \mathrm{T} \mid \Sigma \vdash e : \tau \dashv \mathrm{T}}$

T-DEREF
$$\frac{\Delta \mid \Gamma; \mathrm{T} \mid \Sigma \vdash p : \&_{\mu}\tau \dashv \mathrm{T}}{\Delta \mid \Gamma; \mathrm{T} \mid \Sigma \vdash *p : \tau \dashv \mathrm{T}}$$

T-DEREF-STRG
$$\frac{\Delta \mid \Gamma; \mathrm{T} \mid \Sigma \vdash p : \mathbf{ptr}(\eta) \dashv \mathrm{T}}{\Delta \mid \Gamma; \mathrm{T} \mid \Sigma \vdash *p : \mathrm{T}(\eta) \dashv \mathrm{T}}$$

3.3 Location Context Inclusion and Subtyping

Context inclusion $\boxed{\Delta \vdash \mathrm{T} \Rightarrow \mathrm{T}}$

C-TRANS
$$\frac{\Delta \vdash \mathrm{T}_1 \Rightarrow \mathrm{T}_2 \qquad \Delta \vdash \mathrm{T}_2 \Rightarrow \mathrm{T}_3}{\Delta \vdash \mathrm{T}_1 \Rightarrow \mathrm{T}_3}$$

C-PERM
$$\frac{\mathrm{T}' \text{ is a permutation of } \mathrm{T}}{\Delta \vdash \mathrm{T} \Rightarrow \mathrm{T}'}$$

C-WEAK
$$\Delta \vdash \mathrm{T}, \mathrm{T}' \Rightarrow \mathrm{T}$$

C-FRAME
$$\frac{\Delta \vdash \mathrm{T}_1 \Rightarrow \mathrm{T}_2}{\Delta \vdash \mathrm{T}, \mathrm{T}_1 \Rightarrow \mathrm{T}, \mathrm{T}_2}$$

C-SUB
$$\frac{\Delta \vdash \tau_1 \preccurlyeq \tau_2}{\Delta \vdash \eta \mapsto \tau_1 \Rightarrow \eta \mapsto \tau_2}$$

Subtyping $\boxed{\Delta \vdash \tau \preccurlyeq \tau}$

$$\textsc{S-ptr}\quad \Delta \vdash \mathbf{ptr}(\eta) \preccurlyeq \mathbf{ptr}(\eta) \qquad\qquad \textsc{S-mem}\quad \Delta \vdash \lightning \preccurlyeq \lightning$$

$$\textsc{S-fun}\quad \frac{\Delta, \overline{a:\sigma} \models r_2 \Rightarrow r_1 \qquad \Delta, \overline{a:\sigma} \vdash \mathrm{T}_{2i} \Rightarrow \mathrm{T}_{1i} \qquad \forall i. \Delta, \overline{a:\sigma} \vdash \tau_{2i} \preccurlyeq \tau_{1i} \qquad \Delta, \overline{a:\sigma} \vdash \mathrm{T}_{1o} \Rightarrow \mathrm{T}_{2o} \qquad \Delta, \overline{a:\sigma} \vdash \tau_{1o} \preccurlyeq \tau_{2o}}{\Delta \vdash \forall \overline{a:\sigma}.\, \mathbf{fn}(\mathrm{T}_{1i}; \overline{\tau_1}) \to \tau_{1o}/\mathrm{T}_{1o} \preccurlyeq \forall \overline{a:\sigma}.\, \mathbf{fn}(\mathrm{T}_{2i}; \overline{\tau_2}) \to \tau_{2o}/\mathrm{T}_{2o}}$$

$$\textsc{S-idx}\quad \frac{\Delta \models r_1 = r_2}{\Delta \vdash B[r_1] \preccurlyeq B[r_2]}$$

$$\textsc{S-unpack}\quad \frac{\Delta, a : \mathtt{sort}(B), r \vdash B[a] \preccurlyeq \tau}{\Delta \vdash \{a.\, B[a] \mid r\} \preccurlyeq \tau} \qquad \textsc{S-ex}\quad \frac{\Delta \models r_2[r_1/a]}{\Delta \vdash B[r_1] \preccurlyeq \{a.\, B[a] \mid r_2\}} \qquad \textsc{S-shr}\quad \frac{\Delta \vdash \tau_1 \preccurlyeq \tau_2}{\Delta \vdash \&_{\mathbf{shr}}\, \tau_1 \preccurlyeq \&_{\mathbf{shr}}\, \tau_2}$$

$$\textsc{S-mut}\quad \frac{\Delta \vdash \tau_1 \preccurlyeq \tau_2 \qquad \Delta \vdash \tau_2 \preccurlyeq \tau_1}{\Delta \vdash \&_{\mathbf{mut}}\, \tau_1 \preccurlyeq \&_{\mathbf{mut}}\, \tau_2}$$

3.4 Implication Checking

Implication checking $\Delta \models r$ is implemented via SMT. Here we axiomatize the properties we need for the proof.

Assumption 1 (Weakening). *If* $\Delta_1, \Delta_2 \models r$ *then* $\Delta_1, \Delta', \Delta_2 \models r$.

Assumption 2 (Cut). *If* $\Delta_1 \models r_1$ *and* $\Delta_1, r_1, \Delta_2 \models r_2$, *then* $\Delta_1, \Delta_2 \models r_2$.

Assumption 3 (Identity). *Forall* Δ *and* r, $\Delta, r \models r$.

Assumption 4 (Substitution). *If* $\Delta_1 \vdash r_a : \sigma$ *and* $\Delta_1, a : \sigma, \Delta_2 \models r$, *then* $\Delta_1, \Delta_2[r_a/a] \models r[r_a/a]$.

Assumption 5 (Transitive). *If* $\Delta \models r_1 = r_2$ *and* $\Delta \models r_2 = r_3$, *then* $\Delta \models r_1 = r_3$.

Assumption 6 (Transitivity of Implication). *If* $\Delta \models r_1 \Rightarrow r_2$ *and* $\Delta \models r_2 \Rightarrow r_3$, *then* $\Delta \models r_1 \Rightarrow r_3$.

Assumption 7 (Congruence). *If* $\Delta \models r_1 = r_2$ *and* $\Delta \models r[r_2/a]$, *then* $\Delta \models r[r_1/a]$.

Assumption 8 (Reflexivity). $\Delta \models r = r$.

4 STACKED BORROWS

Our operational semantics is based on the *Stacked Borrows* aliasing model [Jung et al. 2020a]. We take definitions as is from the original Stacked Borrows appendix [Jung et al. 2020b]. For convenience for the reader we replicate some definitions here. As we are only modeling Rust safe fragment, we do not use all features in Stacked Borrows, for example, we do not use SharedRO permissions which are used for the semantics of raw pointers.

4.1 Stacked Borrows Domains

$$
\begin{array}{rcl}
\mathit{PtrId} & \triangleq & \mathbb{N} \\
c \in \mathit{CallId} & \triangleq & \mathbb{N} \\
f \in \mathit{FnName} & \triangleq & \mathit{String} \\
\ell \in \mathit{Lod} & \triangleq & \mathbb{N} \\
\mathit{Lod} & \triangleq & \mathbb{N} \\
X^{?} & \triangleq & X \uplus \{\bot\} \\
\iota & \triangleq & \mathit{Permission} \times \mathit{Tag} \times \mathit{CallId}^{?} \\
p \in \mathit{Permission} & \triangleq & \mathsf{Unique} \mid \mathsf{SahredRW} \mid \mathsf{SharedRO} \mid \mathsf{Disabled} \\
s \in \mathit{Scalar} & \triangleq & \mathsf{Pointer}(\ell, t) \mid z \mid \text{☣} \mid \mathsf{FnPoiner}(f) \\
S \in \mathit{Stack} & \triangleq & \mathit{List}(\mathit{Item}) \\
\xi \in \mathit{Stacks} & \triangleq & \mathit{Loc} \xrightarrow{\text{fin}} \mathit{Stack} \\
m \in \mathit{Mutability} & \triangleq & \mathsf{Mutable} \mid \mathsf{Immutable} \\
\mathit{PtrKind} & \triangleq & \mathsf{Ref}(m) \mid \mathsf{Raw}(m) \mid \mathsf{Box} \\
\tau \in \mathit{Type} & \triangleq & \mathsf{FixedSize}(n) \mid \mathsf{Ptr}(k, \tau) \quad \text{where } n \in \mathbb{N}, k \in \mathit{PtrKind} \\
 & \mid & \mathsf{UnsafeCell}(\tau) \\
 & \mid & \mathsf{Union}(\tau^{*}) \mid \mathsf{Prod}(\tau^{*}) \mid \mathsf{Sum}(\tau^{*}) \\
\mathit{Retag} & \triangleq & \mathsf{Default} \mid \mathsf{Raw} \mid \mathsf{FnEntry} \\
a \in \mathit{AccessType} & \triangleq & \mathsf{AccessRead} \mid \mathsf{AccessWrite} \\
\varsigma \in \mathit{SState} & \triangleq & \left\{ \begin{array}{rcl} \textsc{stacks} & : & \mathit{Stacks}, \\ \textsc{nextptr} & : & \mathit{PtrId}, \\ \textsc{calls} & : & \mathit{List}(\mathit{CallId}), \\ \mathsf{nextcall} & : & \mathit{CallId} \end{array} \right\}
\end{array}
$$

4.2 Relational Semantics

Stacked Borrows defines a relational semantics that makes it convenient to instrument an operational semantics with Stacked Borrows. The relational semantics, describes how different *memory events* affect the stacked borrows state ς. The rules of the relational semantics wrap underlying operations written in a functional style. We do not show the definitions of these operations and refer the reader to the original appendix [Jung et al. 2020b].

$$\begin{array}{rcl}
\varepsilon \in \mathit{Event} & ::= & \mathsf{EAccess}(a, \mathsf{Pointer}(\ell, t), \tau) \\
& | & \mathsf{ERetag}(\mathsf{Pointer}(\ell, t_{old}), t_{new}, \tau, k, k') \quad \text{where } k \in \mathit{PtrKind}, k' \in \mathit{RetagKind} \\
& | & \mathsf{EAlloc}(\mathsf{Pointer}(\ell,t), \tau) \\
& | & \mathsf{EDealloc}(\mathsf{Pointer}(\ell,t), \tau)
\end{array}$$

$$\boxed{\varsigma \xrightarrow{\varepsilon} \varsigma'}$$

OS-alloc

$$\frac{\begin{array}{c}\forall \ell' \in [\ell, \ell + |\tau|).\ \ell' \notin \mathrm{dom}(\varsigma.\textsc{stacks}) \qquad t = \varsigma.\textsc{nextptr} \\ \xi = \varsigma.\textsc{stacks}\ \mathbf{with}_{\ell' \in [\ell, \ell+|\tau|)} [\ell' := [(\mathsf{Unique}, t, \bot])]] \\ \varsigma' = \varsigma\ \mathbf{with}\ \textsc{stacks} := \xi', \textsc{nextptr} := \varsigma.\mathit{nextptr} + 1]\end{array}}{\varsigma \xrightarrow{\mathsf{EAlloc}(\mathsf{Pointer}(\ell,t),\tau)} \varsigma'}$$

OS-dealloc

$$\frac{\mathsf{MemDeallocated}(\varsigma.\textsc{stacks}, \varsigma.\textsc{calls}, \mathsf{Pointer}(\ell, t), |\tau|) = \xi' \qquad \varsigma' = \varsigma\ \mathbf{with}\ [\textsc{stacks} := \xi']}{\varsigma \xrightarrow{\mathsf{EDealloc}(\mathsf{Pointer}(\ell,t),\tau)} \varsigma'}$$

OS-access

$$\frac{\mathsf{MemAccessed}(\varsigma.\textsc{stacks}, \varsigma.\textsc{calls}, a, \mathsf{Pointer}(\ell, t), |\tau|) = \xi' \qquad \varsigma' = \varsigma\ \mathbf{with}\ [\textsc{stacks} := \xi']}{\varsigma \xrightarrow{\mathsf{EAccess}(a,\mathsf{Pointer}(\ell,t),\tau)} \varsigma'}$$

OS-retag

$$\frac{\begin{array}{c}\mathsf{Retag}(\varsigma.\textsc{stacks}, \varsigma.\textsc{nextptr}, \varsigma.\textsc{calls}, \mathsf{Pointer}(\ell, t_{old}), \tau, k, k') = (t_{new}, \xi', n') \\ \varsigma' = \varsigma\ \mathbf{with}\ [\textsc{stacks} := \xi', \textsc{nextptr} := n']\end{array}}{\varsigma \xrightarrow{\mathsf{ERetag}(\mathsf{Pointer}(\ell,t_{old}),t_{new},\tau,k,k')} \varsigma'}$$

5 OPERATIONAL SEMANTICS OF $\lambda_{\mathbf{LR}}$.

The operational semantics follows a sandard call-by-value evaluation. Note that function arguments are A-values and thus they must not be further evaluated. We use the convention that a rule with a stacked borrow transition as a premise does not get stuck if the transition does not hold but instead evaluates to an error. In O-Vec-index-mut, for example, evaluation will return an error if either the EAccess or ERetag transition does not hold, but it will get stuck if the index does not point to a valid location.

$$\begin{array}{rrcl}
\textbf{\textit{Heap}} & h & ::= & \emptyset \mid h[\ell \mapsto v] \\
\textbf{\textit{Contexts}} & C & ::= & \bullet \mid \mathbf{let}\ x = C\ \mathbf{in}\ e \mid \mathbf{if}\ C\ \{e\}\ \mathbf{else}\ \{e\} \mid \mathbf{call}\ C[\overline{r}](\overline{av}) \mid p := C
\end{array}$$

Operational Semantics

$$\boxed{\langle h, \varsigma, e\rangle \rightsquigarrow \langle h, \varsigma, e\rangle \text{ or } \langle h, \varsigma, e\rangle \rightsquigarrow \mathsf{ERR}}$$

O-Prop

$$\frac{\langle h_i, \varsigma_i, e\rangle \rightsquigarrow \langle h_o, \varsigma_o, e'\rangle}{\langle h_i, \varsigma_i, C[e]\rangle \rightsquigarrow \langle h_o, \varsigma_o, C[e']\rangle}$$

O-Prop-Err

$$\frac{\langle h_i, \varsigma_i, e\rangle \rightsquigarrow \mathsf{ERR}}{\langle h_i, \varsigma_i, C[e]\rangle \rightsquigarrow \mathsf{ERR}}$$

O-New

$$\frac{\text{fresh}_h\ \ell \qquad \varsigma \xrightarrow{\mathsf{EAlloc(Pointer(}\ell\mathsf{,}t\mathsf{),FixedSize(1))}} \varsigma'}{\langle h, \varsigma, \textbf{let } x = \textbf{new}(\rho) \textbf{ in } e\rangle \rightsquigarrow \langle h[\ell \mapsto ☣], \varsigma', e[\ell/\rho][\text{ptr}(\ell, t)/x]\rangle}$$

O-Let

$$\langle h, \varsigma, \textbf{let } x = v \textbf{ in } e\rangle \rightsquigarrow \langle h, \varsigma, e[v/x]\rangle$$

O-If-True

$$\langle h, \varsigma, \textbf{if true } \{e_1\} \textbf{ else } \{e_2\}\rangle \rightsquigarrow \langle h, \varsigma, e_1\rangle$$

O-If-False

$$\langle h, \varsigma, \textbf{if false } \{e_1\} \textbf{ else } \{e_2\}\rangle \rightsquigarrow \langle h, \varsigma, e_2\rangle$$

O-Call

$$\langle h, \varsigma, \textbf{call } (\textbf{rec } f[\overline{a}](\overline{x}) \ := \ e)[\overline{r}](\overline{v})\rangle \rightsquigarrow \langle h, \varsigma, e[\textbf{rec } f[\overline{a}](\overline{x}) \ := \ e/f, \overline{v}/\overline{x}, \overline{r}/\overline{a}]\rangle$$

O-UnPack

$$\langle h, \varsigma, \textbf{unpack}(x, a) \textbf{ in } e\rangle \rightsquigarrow \langle h, \varsigma, e\rangle$$

O-Assign

$$\frac{\varsigma \xrightarrow{\mathsf{EAccess(AccessWrite,Pointer(}\ell\mathsf{,}t\mathsf{),FixedSize(1))}} \varsigma'}{\langle h[\ell \mapsto v], \varsigma, \text{ptr}(\ell, t) := v'\rangle \rightsquigarrow \langle h[\ell \mapsto v'], \varsigma', ☣\rangle}$$

O-Strg-Rebor

$$\frac{\varsigma \xrightarrow{\mathsf{ERetag(Pointer(}\ell\mathsf{,}t\mathsf{),}t'\mathsf{,FixedSize(1),Ref(}\textbf{mut}\mathsf{),Raw)}} \varsigma'}{\langle h, \varsigma, \&\textbf{strg}\ \text{ptr}(\ell, t)\rangle \rightsquigarrow \langle h, \varsigma', \text{ptr}(\ell, t')\rangle}$$

O-Mut-Rebor

$$\frac{\varsigma \xrightarrow{\mathsf{ERetag(Pointer(}\ell\mathsf{,}t\mathsf{),}t'\mathsf{,FixedSize(1),Ref(}\textbf{mut}\mathsf{),Raw)}} \varsigma'}{\langle h, \varsigma, \&\textbf{mut}\ \text{ptr}(\ell, t)\rangle \rightsquigarrow \langle h, \varsigma', \text{ptr}(\ell, t')\rangle}$$

O-Shr-Rebor

$$\frac{\varsigma \xrightarrow{\mathsf{ERetag(Pointer(}\ell\mathsf{,}t\mathsf{),}t'\mathsf{,FixedSize(1),Ref(}\textbf{shr}\mathsf{),Raw)}} \varsigma'}{\langle h, \varsigma, \&\textbf{shr}\ \text{ptr}(\ell, t)\rangle \rightsquigarrow \langle h, \varsigma', \text{ptr}(\ell, t')\rangle}$$

O-Deref

$$\frac{\varsigma \xrightarrow{\mathsf{EAccess(AccessRead,Pointer(}\ell\mathsf{,}t\mathsf{),FixedSize(1))}} \varsigma'}{\langle h, \varsigma, *\text{ptr}(\ell, t)\rangle \rightsquigarrow \langle h, \varsigma', h(\ell)\rangle}$$

Operational Semantics (vectors) $\boxed{\langle h, \varsigma, e\rangle \rightsquigarrow \langle h, \varsigma, e\rangle}$

O-Vec-new

$$\langle h, \varsigma, \textbf{call Vec}_\tau\textbf{::new}()\rangle \rightsquigarrow \langle h, \varsigma, \textbf{vec}_\tau(0, ☣)\rangle$$

O-Vec-push

$$\frac{\begin{array}{c} \varsigma_i \xrightarrow{\text{EAccess(AccessWrite,Pointer}(\ell,t)\text{,FixedSize}(1))} \varsigma \\ \varsigma \xrightarrow{\text{EDealloc(Pointer}(\ell',t')\text{,FixedSize}(n))} \varsigma' \\ \varsigma' \xrightarrow{\text{EAlloc(Pointer}(\ell'',t'')\text{,FixedSize}(n+1))} \varsigma_o \\ [\ell'', \ell''+n] \mathbin{\#} \mathrm{dom}(h) \\ h_i = h[\ell \mapsto \textbf{vec}_\tau(n, \mathrm{ptr}(\ell', t'))][\ell' \mapsto_n \bar{v}] \\ h_o = h[\ell \mapsto \textbf{vec}_\tau(n+1, \mathrm{ptr}(\ell'', t''))][\ell'' \mapsto_{n+1} \bar{v} +\!\!+ [v]] \end{array}}{\langle h_i, \varsigma_i, \textbf{call Vec}_\tau\textbf{::push}[\ell](\mathrm{ptr}(\ell, t), v)\rangle \rightsquigarrow \langle h_o, \varsigma_o, ☣\rangle}$$

O-Vec-push-empty

$$\frac{\begin{array}{c} \varsigma_i \xrightarrow{\text{EAccess(AccessWrite,Pointer}(\ell,t)\text{,FixedSize}(1))} \varsigma \\ \varsigma \xrightarrow{\text{EAlloc(Pointer}(\ell',t')\text{,FixedSize}(1))} \varsigma_o \\ \ell' \notin \mathrm{dom}(h) \\ h_i = h[\ell \mapsto \textbf{vec}_\tau(0, v')] \\ h_o = h[\ell \mapsto \textbf{vec}_\tau(1, \mathrm{ptr}(\ell', t'))][\ell' \mapsto v] \end{array}}{\langle h_i, \varsigma_i, \textbf{call Vec}_\tau\textbf{::push}[\ell](\mathrm{ptr}(\ell, t), v)\rangle \rightsquigarrow \langle h_o, \varsigma_o, ☣\rangle}$$

O-Vec-index-mut

$$\frac{\begin{array}{c} \ell + i \in \mathrm{dom}(h) \\ \varsigma_i \xrightarrow{\text{EAccess(AccessRead,Pointer}(\ell,t)\text{,FixedSize}(1))} \varsigma \\ \varsigma \xrightarrow{\text{ERetag(Pointer}(\ell'+i,t')\text{,}t''\text{,FixedSize}(1)\text{,}\textbf{mut}\text{,Raw)}} \varsigma_o \\ h = h'[\ell \mapsto \textbf{vec}_\tau(n, \mathrm{ptr}(\ell', t'))] \end{array}}{\langle h, \varsigma_i, \textbf{call Vec}_\tau\textbf{::index\_mut}(\mathrm{ptr}(\ell, t), i)\rangle \rightsquigarrow \langle h, \varsigma_o, \mathrm{ptr}(\ell' + i, t'')\rangle}$$

5.1 Well-typed States

Intuitively, a state is well-typed if the following conditions hold: (1) for each location ℓ the stack $\varsigma.\textsc{stacks}(\ell)$ must be split into a section composed of only unique permissions followed by a section composed of only shared permissions, (2) pointers $\mathrm{ptr}(\ell, t)$ in the shared section must be associated to shared references in Σ (*i.e.,* $\Sigma(\ell, t) = \&_{\mathbf{shr}}\tau$), (3) the unique section must further be split into a section associated to strong pointers ($\Sigma(\ell, t) = \mathbf{ptr}(\ell)$) and a section associated with mutable references ($\Sigma(\ell, t) = \&_{\mathbf{mut}}\tau$), and (4) for every pointer $\mathrm{ptr}(\ell, t)$ it must be safe to read from it at type $\Sigma(\ell, t)$. This implies, for example, that for every mutable reference $\&_{\mathbf{mut}}\tau$ associated to a location ℓ the type τ must be a subtype of $\mathrm{T}(\ell)$.

Well-typed state $\boxed{\Delta \mid \mathrm{T} \mid \Sigma \vdash \langle h, \varsigma \rangle}$

$$\frac{\mathrm{dom}(h) = \mathrm{dom}(\varsigma.\textsc{stacks}) \qquad \forall \ell \in \mathrm{dom}(h).\ \Delta \mid \mathrm{T} \mid \Sigma \vdash \langle [\ell \mapsto h(\ell)], \varsigma.\textsc{stacks}(\ell) \rangle}{\Delta \mid \mathrm{T} \mid \Sigma \vdash \langle h, \varsigma \rangle}$$

Well-typed heaplet $\boxed{\Delta \mid \mathrm{T} \mid \Sigma \vdash \langle [\ell \mapsto v], S \rangle}$

Hp-tracked

$$\frac{\ell \in \mathrm{dom}(\mathrm{T}) \qquad \Delta \mid \ell \mapsto \mathrm{T}(\ell) \mid \Sigma \vdash_{\mathbf{shr}} S \qquad \Delta \mid \emptyset; \mathrm{T} \mid \Sigma \vdash v : \mathrm{T}(\ell) \dashv \mathrm{T}}{\Delta \mid \mathrm{T} \mid \Sigma \vdash \langle [\ell \mapsto v], S \rangle}$$

Hp-untracked

$$\frac{\ell \notin \mathrm{dom}(\mathrm{T}) \qquad \Delta \mid \ell \mapsto \tau \mid \Sigma \vdash_{\mathbf{shr}} S \qquad \Delta \mid \emptyset; \mathrm{T} \mid \Sigma \vdash v : \tau \dashv \mathrm{T}}{\Delta \mid \mathrm{T} \mid \Sigma \vdash \langle [\ell \mapsto v], S \rangle}$$

Well-typed stacks $\boxed{\Delta \mid \ell \mapsto \tau \mid \Sigma \vdash_{\mathbf{shr}} S, (\mathsf{SharedRO}, t, \_)}$

S-Shr

$$\frac{\Sigma(\ell, t) = \&_{\mathbf{shr}}\tau' \qquad \Delta \mid \ell \mapsto \tau \mid \Sigma \vdash_{\mathbf{shr}} S \qquad \Delta \vdash \tau \preccurlyeq \tau'}{\Delta \mid \ell \mapsto \tau \mid \Sigma \vdash_{\mathbf{shr}} S, (\mathsf{SharedRO}, t, \_)}$$

S-Shr-Mut

$$\frac{\Delta \mid \ell \mapsto \tau \mid \Sigma \vdash_{\mathbf{mut}} S}{\Delta \mid \ell \mapsto \tau \mid \Sigma \vdash_{\mathbf{shr}} S}$$

S-Mut

$$\frac{\Sigma(\ell, t) = \&_{\mathbf{mut}}\tau' \qquad \Delta \mid \ell \mapsto \tau \mid \Sigma \vdash_{\mathbf{mut}} S \qquad \Delta \vdash \tau \preccurlyeq \tau' \qquad \Delta \vdash \tau' \preccurlyeq \tau}{\Delta \mid \ell \mapsto \tau \mid \Sigma \vdash_{\mathbf{mut}} S, (\mathsf{Unique}, t, \_)}$$

S-Mut-Strg

$$\frac{\Delta \mid \ell \mapsto \tau \mid \Sigma \vdash_{\mathbf{strg}} S}{\Delta \mid \ell \mapsto \tau \mid \Sigma \vdash_{\mathbf{mut}} S}$$

S-Strg

$$\frac{\Sigma(\ell, t) = \mathbf{ptr}(\ell) \qquad \Delta \mid \ell \mapsto \tau \mid \Sigma \vdash_{\mathbf{strg}} S}{\Delta \mid \ell \mapsto \tau \mid \Sigma \vdash_{\mathbf{strg}} S, (\mathsf{Unique}, t, \_)}$$

S-Empty

$$\Delta \mid \ell \mapsto \tau \mid \Sigma \vdash_{\mathbf{strg}} \emptyset$$

S-Disabled

$$\frac{\Delta \mid \ell \mapsto \tau \mid \Sigma \vdash_{*} S}{\Delta \mid \ell \mapsto \tau \mid \Sigma \vdash_{*} S, (\mathsf{Disabled}, \_, \_)}$$

6 SOUNDNESS OF λ_{LR}.

6.1 Main theorems

We prove the following main soundness theorems. The soundness claim states that evaluation of a well-typed expression will either 1) return a value of the same type, 2) return an aliasing violation error (ERR), or 3) diverge.

Theorem 6.1 (Soundness). *If*

- $\emptyset \mid \emptyset; T_i \mid \Sigma_i \vdash e_i : \tau \dashv T$,
- $\emptyset \mid T_i \mid \Sigma_i \vdash \langle h_i, \varsigma_i \rangle$, *and*
- $\langle h_i, \varsigma_i, e_i \rangle \rightsquigarrow^\star \langle h, \varsigma, e \rangle$.

then

- *e is a value and there exist T_o and $\Sigma_o \supseteq \Sigma_i$ so that $\emptyset \mid \emptyset; T_o \mid \Sigma_o \vdash e : \tau \dashv T$, or*
- $\langle h_o, \varsigma_o, e \rangle \rightsquigarrow \mathsf{ERR}$, *or*
- *there exists T_o, $\Sigma_o \supseteq \Sigma_i$, h_o, ς_o, and e_o such that $\langle h, \varsigma, e \rangle \rightsquigarrow \langle h_o, \varsigma_o, e_o \rangle$, $\emptyset \mid T_o \mid \Sigma_o \vdash \langle h_o, \varsigma_o \rangle$, and $\emptyset \mid \emptyset; T_o \mid \Sigma_o \vdash e_o : \tau \dashv T$.*

Theorem 6.2 (Progress). *If $\emptyset \mid \emptyset; T_i \mid \Sigma_i \vdash e : \tau \dashv T_o$ and $\emptyset \mid T_i \mid \Sigma_i \vdash \langle h_i, \varsigma_i \rangle$, then*

- *e is a value,*
- $\langle h_i, \varsigma_i, e \rangle \rightsquigarrow \mathsf{ERR}$, *or*
- *there exists h_o, ς_o, and e_o so that $\langle h_i, \varsigma_i, e \rangle \rightsquigarrow \langle h_o, \varsigma_o, e_o \rangle$.*

Theorem 6.3 (Preservation). *If*

- $\Delta \mid \emptyset; T_i \mid \Sigma_i \vdash e : \tau \dashv T$,
- $\langle h_i, \varsigma_i, e \rangle \rightsquigarrow \langle h_o, \varsigma_o, e_o \rangle$, *and*
- $\Delta \mid T_i \mid \Sigma_i \vdash \langle h_i, \varsigma_i \rangle$,

then there exist T_o and $\Sigma_o \supseteq \Sigma_i$, such that

- $\Delta \mid \emptyset; T_o \mid \Sigma_o \vdash e_o : \tau \dashv T$ *and*
- $\Delta \mid T_o \mid \Sigma_o \vdash \langle h_o, \varsigma_o \rangle$

6.2 Proof of soudness

THEOREM 6.1 (SOUNDNESS). *If*

- $\emptyset \mid \emptyset; T_i \mid \Sigma_i \vdash e_i : \tau \dashv T$,
- $\emptyset \mid T_i \mid \Sigma_i \vdash \langle h_i, \varsigma_i \rangle$, *and*
- $\langle h_i, \varsigma_i, e_i \rangle \rightsquigarrow^\star \langle h, \varsigma, e \rangle$.

then

- *e is a value and there exist T_o and $\Sigma_o \supseteq \Sigma_i$ so that $\emptyset \mid \emptyset; T_o \mid \Sigma_o \vdash e : \tau \dashv T$, or*
- $\langle h_o, \varsigma_o, e \rangle \rightsquigarrow \mathsf{ERR}$, *or*
- *there exists T_o, $\Sigma_o \supseteq \Sigma_i$, h_o, ς_o, and e_o such that $\langle h, \varsigma, e \rangle \rightsquigarrow \langle h_o, \varsigma_o, e_o \rangle$, $\emptyset \mid T_o \mid \Sigma_o \vdash \langle h_o, \varsigma_o \rangle$, and $\emptyset \mid \emptyset; T_o \mid \Sigma_o \vdash e_o : \tau \dashv T$.*

PROOF. By induction on the length of the evaluation path. For path of length 0, the theorem is a direct implication of Progress (Theorem 6.2). For path of length $n+1$: $\langle h_i, \varsigma_i, e_i \rangle \rightsquigarrow^\star \langle h_n, \varsigma_n, e_n \rangle \rightsquigarrow \langle h, \varsigma, e \rangle$. By the induction hypothesis and determinism of our operational semantics, there exists T_o and $\Sigma_o \supseteq \Sigma_i$ such that $\Delta \mid \mathrm{T}_o \mid \Sigma_o \vdash \langle h_o, \varsigma_o \rangle$ and $\Delta \mid \emptyset; \mathrm{T}_o \mid \Sigma_o \vdash e : \tau \dashv \mathrm{T}$. The proof concludes by applying progress (Theorem 6.2) and then preservation (Theorem 6.3). □

6.3 Proof of preservation

THEOREM 6.3 (PRESERVATION). *If*

- $\Delta \mid \emptyset; T_i \mid \Sigma_i \vdash e : \tau \dashv T$,
- $\langle h_i, \varsigma_i, e \rangle \rightsquigarrow \langle h_o, \varsigma_o, e_o \rangle$, *and*
- $\Delta \mid T_i \mid \Sigma_i \vdash \langle h_i, \varsigma_i \rangle$,

then there exist T_o and $\Sigma_o \supseteq \Sigma_i$, such that

- $\Delta \mid \emptyset; T_o \mid \Sigma_o \vdash e_o : \tau \dashv T$ *and*
- $\Delta \mid T_o \mid \Sigma_o \vdash \langle h_o, \varsigma_o \rangle$

PROOF. By induction on the type derivation tree.

- **Rule T-SUB:** By inversion of the rule: (1) $\Delta \mid \Gamma; \mathrm{T}_i \mid \Sigma_i \vdash e : \tau_1 \dashv \mathrm{T}'$, (2) $\Delta \vdash \tau_1 \preccurlyeq \tau$, and (3) $\Delta \vdash \mathrm{T}' \Rightarrow \mathrm{T}$. By inductive hypothesis on (1): then there exist T_o and $\Sigma_o \supseteq \Sigma_i$, such that (4) $\Delta \mid \emptyset; \mathrm{T}_o \mid \Sigma_o \vdash e_o : \tau_1 \dashv \mathrm{T}$ and (5) $\Delta \mid \mathrm{T}_o \mid \Sigma_o \vdash \langle h, \varsigma \rangle$. The proof concludes by (4), (2), (3), and rule T-SUB.
- **Rule T-IF:** Let $e \equiv$ **if** e_0 $\{e_1\}$ **else** $\{e_2\}$. By inversion of the rule:
  (1) $\Delta \mid \emptyset; \mathrm{T}_i \mid \Sigma_i \vdash e_0 : \mathbf{bool}[r] \dashv \mathrm{T}_0$
  (2) $\Delta, r \mid \emptyset; \mathrm{T}_0 \mid \Sigma_i \vdash e_1 : \tau \dashv \mathrm{T}$
  (3) $\Delta, \neg r \mid \emptyset; \mathrm{T}_0 \mid \Sigma_i \vdash e_2 : \tau_2 \dashv \mathrm{T}$.
  Since e steps, there are three cases:

- **Rule O-Prop:** $\langle h_i, \varsigma_i, \textbf{if}\ e_0\ \{e_1\}\ \textbf{else}\ \{e_2\}\rangle \rightsquigarrow \langle h_o, \varsigma_o, \textbf{if}\ e'_0\ \{e_1\}\ \textbf{else}\ \{e_2\}\rangle$. By inversion: $\langle h_i, \varsigma_i, e_0\rangle \rightsquigarrow \langle h_o, \varsigma_o, e'_0\rangle$. By (1) and inductive hypothesis: there exists T_o and $\Sigma_o \supseteq \Sigma_i$, so that $\Delta \mid \emptyset; \mathrm{T}_o \mid \Sigma_o \vdash e'_0 : \textbf{bool}[r] \dashv \mathrm{T}_0$ and $\Delta \mid \mathrm{T}_o \mid \Sigma_o \vdash \langle h_o, \varsigma_o\rangle$ By (2) and (3), weakened to Σ_o using Lemma 6.5, and rule T-If: $\Delta \mid \emptyset; \mathrm{T}_o \mid \Sigma_o \vdash \textbf{if}\ e'_0\ \{e_1\}\ \textbf{else}\ \{e_2\} : \tau \dashv \mathrm{T}$, which concludes the proof.
- **Rule O-If-True:** $\langle h, \varsigma, \textbf{if true}\ \{e_1\}\ \textbf{else}\ \{e_2\}\rangle \rightsquigarrow \langle h, \varsigma, e_1\rangle$, where $h = h_i = h_o$ and $\varsigma = \varsigma_i = \varsigma_o$. Trivial by (2) and with $\mathrm{T}_o = \mathrm{T}_i$ and $\Sigma_o = \Sigma_i$.
- **Rule O-If-False:** $\langle h, \varsigma, \textbf{if false}\ \{e_1\}\ \textbf{else}\ \{e_2\}\rangle \rightsquigarrow \langle h, \varsigma, e_2\rangle$, where $h = h_i = h_o$ and $\varsigma = \varsigma_i = \varsigma_o$. Trivial by (3) and with $\mathrm{T}_o = \mathrm{T}_i$ and $\Sigma_o = \Sigma_i$.

• **Rule T-new:** Let $e \equiv \textbf{let}\ x = \textbf{new}(\rho)\ \textbf{in}\ e'$. By inversion of the rule we get:

$$(1)\ \Delta, \rho : \textbf{loc} \mid x : \textbf{ptr}(\rho); \mathrm{T}_i, \rho \mapsto \lightning \mid \Sigma_i \vdash e' : \tau \dashv \mathrm{T}$$

By assumption we have $\langle h_i, \varsigma_i, \textbf{let}\ x = \textbf{new}(\rho)\ \textbf{in}\ e'\rangle \rightsquigarrow \langle h_o, \varsigma_o, e_o\rangle$, thus the rule O-New must have been applied and we know $e_o \equiv e'[\ell/\rho][\mathrm{ptr}(\ell, t)/x]$ and $h_o \equiv h_i[\ell \mapsto ☣]$. By Lemma 6.26 we get

$$(2)\ \Delta \mid x : \textbf{ptr}(\ell); \mathrm{T}_i[\ell/\rho], \ell \mapsto \lightning \mid \Sigma_i[\ell/\rho] \vdash e'[\ell/\rho] : \tau[\ell/\rho] \dashv \mathrm{T}[\ell/\rho]$$

We know $\Delta \vdash_{\mathtt{wf}} \tau$ and $\Delta \vdash_{\mathtt{wf}} \mathrm{T}$ by inversion of T-new, thereby, we know $\tau = \tau[\ell/\rho]$ and $\mathrm{T} = \mathrm{T}[\ell/\rho]$ by lemmas 6.13 and 6.12. Then, applying Lemma 6.17 to (2) we get:

$$(3)\ \Delta \mid x : \textbf{ptr}(\ell); \mathrm{T}_i[\ell/\rho], \ell \mapsto \lightning \mid \Sigma_i[\ell/\rho], (\ell, t) \mapsto \textbf{ptr}(\ell) \vdash e'[\ell/\rho][\mathrm{ptr}(\ell, t)/x] : \tau \dashv \mathrm{T}$$

Picking $T_o = T_i[\ell/\rho]$ and $\Sigma_o = \Sigma_i, (\ell, t) \mapsto \textbf{ptr}(\ell)$ we get the first part of the conclusion. It remains to show $\Delta \mid \mathrm{T}_o \mid \Sigma_o \vdash \langle h_o, \varsigma_o\rangle$ which is a direct application of Lemma 6.38.

• **Rule T-let:** Let $e \equiv \textbf{let}\ x = e_x\ \textbf{in}\ e'$. By inversion of the rule we get: (1) $\Delta \mid \emptyset; \mathrm{T}_i \mid \Sigma_i \vdash e_x : \tau_x \dashv \mathrm{T}_0$ and (2) $\Delta \mid x : \tau_x; \mathrm{T}_0 \mid \Sigma_i \vdash e : \tau \dashv \mathrm{T}$. Since e steps, there are two cases:

- **Rule O-Let:** $\langle h_i, \varsigma_i, \textbf{let}\ x = v\ \textbf{in}\ e'\rangle \rightsquigarrow \langle h_i, \varsigma_i, e'[v/x]\rangle$. Since $e_x = v$, by (1), (2), and Lemma 6.17: $\Delta \mid \emptyset; \mathrm{T}_i \mid \Sigma_i \vdash e'[v/x] : \tau \dashv \mathrm{T}$ which concludes the proof.
- **Rule O-Prop:** $\langle h_i, \varsigma_i, \textbf{let}\ x = e_x\ \textbf{in}\ e'\rangle \rightsquigarrow \langle h_o, \varsigma_o, \textbf{let}\ x = e'_x\ \textbf{in}\ e'\rangle$. By inversion: $\langle h_i, \varsigma_i, e_x\rangle \rightsquigarrow \langle h_o, \varsigma_o, e'_x\rangle$. By (1) and inductive hypothesis: there exists T_o and $\Sigma_o \supseteq \Sigma_i$, so that $\Delta \mid \emptyset; \mathrm{T}_o \mid \Sigma_o \vdash e'_x : \tau_x \dashv \mathrm{T}_0$ and $\Delta \mid \mathrm{T}_o \mid \Sigma_o \vdash \langle h_o, \varsigma_o\rangle$. By (2), weakened to Σ_o using Lemma 6.5, and rule T-let: $\Delta \mid \emptyset; \mathrm{T}_o \mid \Sigma_o \vdash \textbf{let}\ x = e'_x\ \textbf{in}\ e' : \tau \dashv \mathrm{T}$, which concludes the proof.

• **Rule T-call:** Let $e \equiv \textbf{call}\ e'[\overline{r}](\overline{av})$. By inversion of the rule we get: (0) $\Delta \vdash_{\mathtt{wf}} \Sigma$ (1) $\forall i. \Delta \mid \emptyset; \mathrm{T} \mid \Sigma \vdash av_i : \theta \cdot \tau_i \dashv \mathrm{T}$, (2) $\Delta \mid \emptyset; \mathrm{T} \mid \Sigma \vdash e' : \forall \overline{a : \sigma}.\ \textbf{fn}(\mathrm{T}_i; \overline{\tau}) \to \tau_o/\mathrm{T}_o \dashv \mathrm{T}_1 * \mathrm{T}_2$, (3) $\theta = [\overline{r}/\overline{a}]$, (4) $\Delta \vdash \mathrm{T}_1 \Rightarrow \theta \cdot \mathrm{T}_i$, (5) $\forall i. \Delta \vdash r_i : \sigma_i$, and (6) $\Delta \models \theta \cdot r$.
Since e steps, we have the following cases:

- **Rule O-Prop**: $\langle h_i, \varsigma_i, \textbf{call}\ e'[\overline{r}](\overline{av})\rangle \rightsquigarrow \langle h_o, \varsigma_o, \textbf{call}\ e'_o[\overline{r}](\overline{av})\rangle$. By inversion of the rule $\langle h_i, \varsigma_i, e'\rangle \rightsquigarrow \langle h_o, \varsigma_o, e'_o\rangle$. By this, (2), and inductive hypothesis there exist T'_o and $\Sigma_o \supseteq \Sigma$, so that (7) $\Delta \mid \emptyset; \mathrm{T}'_o \mid \Sigma_o \vdash e'_o : \forall \overline{a : \sigma}.\ \textbf{fn}(\mathrm{T}_i; \overline{\tau}) \to \tau_o/\mathrm{T}_o \dashv \mathrm{T}_1 * \mathrm{T}_2$ and (8) $\Delta \mid \mathrm{T}'_o \mid \Sigma_o \vdash \langle h_o, \varsigma_o\rangle$. By Lemmas 6.11, 6.5, and (1): (9) $\forall i. \Delta \mid \emptyset; \mathrm{T}'_o \mid \Sigma_o \vdash av_i : \theta \cdot \tau_i \dashv \mathrm{T}'_o$, By rule T-call on (9), (7), (3), (4), (5), (6), and (0): $\Delta \mid \emptyset; \mathrm{T}'_o \mid \Sigma_o \vdash \mathbf{call}\ e'[\overline{r}](\overline{av}) : \theta \cdot \tau_o \dashv \theta \cdot \mathrm{T}_o * \mathrm{T}_2$. Which, combined with (7), concludes the proof.

– **Rule O-Call:**

$$\langle h, \varsigma, \mathbf{call}\ (\mathbf{rec}\ f[\overline{a}](\overline{x})\ :=\ e_f)[\overline{r}](\overline{v})\rangle \rightsquigarrow \langle h, \varsigma, e_f[\mathbf{rec}\ f[\overline{a}](\overline{x})\ :=\ e_f/f, \overline{v}/\overline{x}, \overline{a}/\overline{r}]\rangle$$

Since $e' \equiv \mathbf{rec}\ f[\overline{a}](\overline{x})\ :=\ e_f$, by inversion of (2), we get:

$$(7) \quad \Delta, \overline{a : \sigma}, r \mid \overline{x : \tau}, f : \forall \overline{a : \sigma}.\ \mathbf{fn}(\mathrm{T}_i; \overline{\tau}) \to \tau/\mathrm{T}_o; \mathrm{T}_i \mid \Sigma \vdash e_f : \tau_o \dashv \mathrm{T}_o$$

By Lemma 6.26 and (5):

$$(8) \quad \Delta, \theta \cdot r \mid \overline{x : \theta \cdot \tau}, f : \theta \cdot \forall \overline{a : \sigma}.\ \mathbf{fn}(\mathrm{T}_i; \overline{\tau}) \to \tau/\mathrm{T}_o; \theta \cdot \mathrm{T}_i \mid \theta \cdot \Sigma \vdash \theta \cdot e_f : \theta \cdot \tau_o \dashv \theta \cdot \mathrm{T}_o$$

The θ application is an identity in the type of the function (that is forall-quantified), and in Σ (by (0) and lemma 6.12). Further, by (6) and Lemma 6.15 we remove logical condition $\theta \cdot r$:

$$(9) \quad \Delta \mid \overline{x : \theta \cdot \tau}, f : \forall \overline{a : \sigma}.\ \mathbf{fn}(\mathrm{T}_i; \overline{\tau}) \to \tau/\mathrm{T}_o; \theta \cdot \mathrm{T}_i \mid \Sigma \vdash \theta \cdot e_f : \theta \cdot \tau_o \dashv \theta \cdot \mathrm{T}_o$$

By (1), (2), and Lemmata 6.17 and 6.11:

$$(10) \quad \Delta \mid \emptyset; \theta \cdot \mathrm{T}_i \mid \Sigma \vdash e_f[\mathbf{rec}\ f[\overline{a}](\overline{x})\ :=\ e_f/f, \overline{v}/\overline{x}, \overline{a}/\overline{r}] : \theta \cdot \tau_o \dashv \theta \cdot \mathrm{T}_o$$

By (4) and Lemma 6.6

$$(11) \quad \Delta \mid \emptyset; \mathrm{T}_1 \mid \Sigma \vdash e_f[\mathbf{rec}\ f[\overline{a}](\overline{x})\ :=\ e_f/f, \overline{v}/\overline{x}, \overline{r}/\overline{a}] : \theta \cdot \tau_o \dashv \theta \cdot \mathrm{T}_o$$

Finally, by Lemma 6.8

$$(12) \quad \Delta \mid \emptyset; \mathrm{T}_1 * \mathrm{T}_2 \mid \Sigma \vdash e_f[\mathbf{rec}\ f[\overline{a}](\overline{x})\ :=\ e_f/f, \overline{v}/\overline{x}, \overline{r}/\overline{a}] : \theta \cdot \tau_o \dashv \theta \cdot \mathrm{T}_o * \mathrm{T}_2$$

Which concludes the proof, because e' is a value, thus $\mathrm{T}_1 * \mathrm{T}_2 = \mathrm{T}$ and by assumption $\Delta \mid \mathrm{T} \mid \Sigma \vdash \langle h, \varsigma \rangle$.

– **Rule O-Vec-new:** We have

$$\langle h_i, \varsigma_i, \mathbf{call}\ \mathbf{Vec}_{\tau_v}\mathbf{::new}()\rangle \rightsquigarrow \langle h_i, \varsigma_i, \mathbf{vec}_\tau(0, ☣)\rangle$$

Because the output state does not change we know it is still well-typed. By inversion of (2) we know $\Delta \vdash \mathbf{Vec}_{\tau_v}[0] \preccurlyeq \tau_o$. We conclude by applying T-vec and T-sub to prove the result is also well-typed.

– **Rule O-Vec-push:** We have

$$\langle h_i, \varsigma_i, \mathbf{call}\ \mathbf{Vec}_{\tau_v}\mathbf{::push}[\ell](\mathrm{ptr}(\ell, t), v)\rangle \rightsquigarrow \langle h_o, \varsigma_o, ☣\rangle$$

By inversion of (1) we get $\Sigma_i(\ell, t) = \mathbf{ptr}(\ell)$ and $\Delta \mid \emptyset; \mathrm{T} \mid \Sigma_i \vdash v : \tau_v \dashv \mathrm{T}$. Then, by Lemma 6.48 we get $\Delta \mid (\ell \mapsto \mathbf{Vec}_{\tau_v}[n+1]), \mathrm{T}_2 \mid \Sigma_o \vdash \langle h_o, \varsigma_o \rangle$. Finally, by inversion of (2) we get $\Delta \vdash ↯ \preccurlyeq \tau_o$ and consequently by applying T-mem and T-sub the result is well-typed.

- **Rule O-Vec-push-empty:** We have

$$\langle h_i, \varsigma_i, \textbf{call Vec}_{\tau_v}\textbf{::push}[\ell](\text{ptr}(\ell, t), v)\rangle \rightsquigarrow \langle h_o, \varsigma_o, ☣\rangle$$

  and by inversion of the rule we also have $h(\ell) = \text{ptr}(0, v')$. By inversion of (1) we also have $\Delta \mid \emptyset; \mathrm{T} \mid \Sigma_i \vdash v : \tau_v \dashv \mathrm{T}, \Sigma_i(\ell, t) = \textbf{ptr}(\ell)$ and $\Delta \vdash \lightning \preccurlyeq \tau_o$. We pick $\Sigma_o = \Sigma_i, (\ell', t') \mapsto \&_{\textbf{mut}}\tau_v$. By Lemma 6.47 we get $\Delta \mid \ell \mapsto \textbf{Vec}_{\tau_v}[1], \mathrm{T}_2 \mid \Sigma_o \vdash \langle h_o, \varsigma_o\rangle$. We conclude by applying T-mem and T-sub to prove the result is well-typed.
- **Rule O-Vec-index-mut:** We have

$$\langle h_i, \varsigma_i, \textbf{call Vec}_{\tau_v}\textbf{::index\_mut}(\text{ptr}(\ell, t), i)\rangle \rightsquigarrow \langle h_i, \varsigma_o, \text{ptr}(\ell' + i, t'')\rangle$$

  By inversion of (1) we have $\Sigma_i(\ell, t) = \&_{\textbf{mut}}\textbf{Vec}_{\tau_v}[n]$. Therefore, by well-typedness of the input state and Lemma 6.43 we have (6) $\Delta \mid \emptyset; \emptyset \mid \Sigma \vdash h(\ell) : \textbf{Vec}_{\tau_v}[n] \dashv \emptyset$. Then, by lemmas 6.36 and 6.23 we have $h(\ell) = \textbf{vec}_\tau(n, v)$. There are two cases:
  * $n = 0$. This case is impossible because by (6) $\Delta \models 0 \le i < 0$ would need to hold which is impossible.
  * $n > 0$. By inversion of T-vec we have $v = \text{ptr}(\ell', t')$. We conclude by picking $\Sigma_o = \Sigma, (\ell', t') \mapsto \&_{\textbf{mut}}\tau_v$ and applying Lemma 6.49.

- **Rule T-var:** Impossible because the typing environment Γ is empty.
- **Rule T-unpack:** Impossible because the typing environment Γ is empty.
- **Rules T-true, T-false, T-int, T-fun, T-ptr, T-mem:** These cases are trivial, since the expressions do not step.
- **Case T-ass:** Let $e = p := e_0$. Since e steps we have two options By inversion of the rule we have

(1) $\Delta \mid \emptyset; \mathrm{T}_i \mid \Sigma_i \vdash e_0 : \tau_v \dashv \mathrm{T}$,
(2) $\Delta \mid \emptyset; \mathrm{T}_i \mid \Sigma_i \vdash p : \&_{\textbf{mut}}\tau \dashv \mathrm{T}_i$
(3) $\Delta \vdash \tau_v \preccurlyeq \tau$.

Since e takes a step there are two options:

- **Case O-Prop:** By inversion of the rule we have $\langle h_i, \varsigma_i, e_0\rangle \rightsquigarrow \langle h_o, \varsigma_o, e_1\rangle$ and $e_o = p := e_2$. By inductive hypothesis on (1) there exists Σ_o and T_o such that

  (1) $\Delta \mid \emptyset; \mathrm{T}_o \mid \Sigma' \vdash e_1 : \tau_v$
  (2) $\Delta \mid \mathrm{T}_o \mid \Sigma' \vdash \langle h_o, \varsigma_o\rangle$

  We conclude by applying T-Ass to prove $\Delta \mid \emptyset; \mathrm{T}_o \mid \Sigma_o \vdash p := e_o : \lightning \dashv \mathrm{T}$.
- **Case O-Assign:** Let $e = \text{ptr}(\ell, t) := v$, $e_o = ☣$ and $h_o = h_i[\ell \mapsto v]$. Replacing $\text{ptr}(\ell, t)$ for p in (2) we also get $\Sigma(\ell, t) = \&_{\textbf{mut}}\tau$. By using (1), (3) and subsumption we get $\Delta \mid \emptyset; \mathrm{T}_i \mid \Sigma_i \vdash v : \tau$. Let us pick $\mathrm{T}_o = \mathrm{T}$ and $\Sigma_o = \Sigma_i$. We get the first part of the conclusion by applying T-mem. We conclude by applying Lemma 6.45.

- **Case T-ass-strg:** Let $e = p := e_0$. By inversion of the rule we have

(1) $\Delta \mid \emptyset; \mathrm{T}_i \mid \Sigma_i \vdash e_0 : \tau \dashv \mathrm{T}'$ and

(2) $\mathrm{T} = \mathrm{T}'[\ell \mapsto \tau]$.

Since e takes a step there are two options:

– **Case O-PROP:** The proof is similar to the O-PROP case for the T-ASS case by applying the inductive hypothesis.

– **Case O-ASSIGN:** Let $e = \mathrm{ptr}(\ell, t) := v$, $e_o =$ ☣ and $h_o = h_i[\ell \mapsto v]$. Replacing $\mathrm{ptr}(\ell, t)$ for p in (2) we also get $\Sigma(\ell, t) = \mathbf{ptr}(\ell)$. Let us pick $\mathrm{T}_o = \mathrm{T}'$ and $\Sigma_o = \Sigma_i$. We conclude by applying Lemma 6.46

- **Case T-BSTRG:** Let $e = \&\mathbf{strg}\ p$ and $T_o = T_i$. Since e takes a step then the O-STRG-REBOR must have been applied and thus we know $p = \mathrm{ptr}(\ell, t)$ and $e_o = \mathrm{ptr}(\ell, t')$. By inversion of the typing rule we know that $\Delta \mid \emptyset; \mathrm{T}_o \mid \Sigma_i \vdash \mathrm{ptr}(\ell, t) : \mathbf{ptr}(\ell) \dashv \mathrm{T}_o$ and consequently $\Sigma_i(\ell, t) = \mathbf{ptr}(\ell)$. Picking $\Sigma_o = \Sigma_i, (\ell, t') \mapsto \mathbf{ptr}(\ell)$ we directly $\Delta \mid \emptyset; \mathrm{T}_o \mid \Sigma_i \vdash \mathrm{ptr}(\ell, t') : \mathbf{ptr}(\ell) \dashv \mathrm{T}_o$. We conclude by Lemma 6.39 to prove the output state is well-typed.
- **Case T-BSMUT:** Since e takes a step then O-MUT-REBOR must have been applied and we know $e = \&\mathbf{mut}\ \mathrm{ptr}(\ell, t)$ and $e_o = \mathrm{ptr}(\ell, t')$. We also know $\mathrm{T}_o = \mathrm{T}_i[\ell \mapsto \tau]$. By inversion of the typing rule we have that (1) $\Delta \mid \emptyset; \mathrm{T}_o \mid \Sigma_i \vdash \mathrm{ptr}(\ell, t) : \mathbf{ptr}(\ell) \dashv \mathrm{T}_o$ and $\Delta \vdash \mathrm{T}_i(\ell) \preccurlyeq \tau$. By (1) we also know $\Sigma_i(\ell, t) = \mathbf{ptr}(\ell)$. If we pick $\Sigma_o = \Sigma_i, (\ell, t') \mapsto \&_{\mathbf{mut}}\tau$ we get directly $\Delta \mid \emptyset; \mathrm{T}_o \mid \Sigma_i \vdash \mathrm{ptr}(\ell, t') : \&_{\mathbf{mut}}\tau \dashv \mathrm{T}_o$. We conclude by applying Lemma 6.40 to prove the output state is well-typed.
- **Case T-BMUT:** Since e takes a step then O-MUT-REBOR must have been applied, thus we know $e = \&\mathbf{mut}\ \mathrm{ptr}(\ell, t)$ and $e_o = \mathrm{ptr}(\ell, t')$. We also know $T = T_i$. By inversion of the typing rule we know that $\Delta \mid \emptyset; \mathrm{T}_i \mid \Sigma_i \vdash \mathrm{ptr}(\ell, t) : \&_{\mathbf{mut}}\tau \dashv \mathrm{T}_i$ and consequently $\Sigma_i(\ell, t) = \&_{\mathbf{mut}}\tau$. By definition of $\Delta \mid \mathrm{T}_i \mid \Sigma_i \vdash \langle h_i, \varsigma_i \rangle$ we know $\Delta \vdash \tau \preccurlyeq \mathrm{T}_i(\ell)\tau$ and $\Delta \mid \emptyset; \emptyset \mid \Sigma_i \vdash h(\ell) : \tau$. We conclude picking $\Sigma_o = \Sigma_i, (\ell, t') \mapsto \&_{\mathbf{mut}}\tau$ and applying Lemma 6.41.
- **Case T-BSHR:** Since e takes a step then O-SHR-REBOR must have been applied. Therefore, $e = \&\mathbf{shr}\ \mathrm{ptr}(\ell, t)$ and $e_o = \mathrm{ptr}(\ell, t')$. We also know $\mathrm{T} = \mathrm{T}_i$ and by inversion of the typing rule we have (1) $\Delta \mid \emptyset; \mathrm{T}_i \mid \Sigma_i \vdash \mathrm{ptr}(\ell, t) : \&_\mu\tau' \dashv \mathrm{T}_i$ and (2) $\Delta \vdash \tau' \preccurlyeq \tau$. By (1) we also have $\Sigma_i(\ell, t) = \&_\mu\tau'$. Since the input stack is well-typed we know $\Delta \vdash \mathrm{T}_i(\ell) \preccurlyeq \tau'$ and and transitivity of subtyping we have $\Delta \vdash \mathrm{T}_i(\ell) \preccurlyeq \tau$. We pick $\Sigma_o = \Sigma_i(\ell, t') \mapsto \&_{\mathbf{shr}}\tau$ and conclude applying Lemma 6.42.
- **Case T-DEREF:** Since e takes a step then O-DEREF must have been applied, therefore $e = *\mathrm{ptr}(\ell, t)$ and $e_o = h(\ell)$. By inversion of the typing rule we know $\Delta \mid \emptyset; \mathrm{T}_i \mid \Sigma_i \vdash \mathrm{ptr}(\ell, t) : \&_\mu\tau \dashv \mathrm{T}_i$ and consequently $\Sigma_i(\ell, t) = \&_\mu\tau$. We pick $\Sigma_o = \Sigma_i$ and conclude by applying Lemma 6.43 to pove the output state is well-typed.
- **Case T-DEREF-STRG:** Since e takes a step then O-DEREF must have been applied, therefore $e = *\mathrm{ptr}(\ell, t)$ and $e_o = h(\ell)$. By inversion of the typing rule we know $\Delta \mid \emptyset; \mathrm{T}_i \mid \Sigma_i \vdash \mathrm{ptr}(\ell, t) : \mathbf{ptr}(\ell) \dashv \mathrm{T}_i$ and $\mathrm{T}(\ell) = \tau$. Consequently, we know $\Sigma_i(\ell, t) = \mathbf{ptr}(\ell)$. We pick $\Sigma_o = \Sigma_i$ and conclude by applying Lemma 6.44.

□

6.4 Proof of progress

Theorem 6.2 (Progress). *If* $\emptyset \mid \emptyset; T_i \mid \Sigma_i \vdash e : \tau \dashv T_o$ *and* $\emptyset \mid T_i \mid \Sigma_i \vdash \langle h_i, \varsigma_i \rangle$, *then*

- *e is a value,*
- $\langle h_i, \varsigma_i, e \rangle \rightsquigarrow \mathsf{ERR}$, *or*
- *there exists h_o, ς_o, and e_o so that* $\langle h_i, \varsigma_i, e \rangle \rightsquigarrow \langle h_o, \varsigma_o, e_o \rangle$.

Proof. By induction on the type derivation tree $\emptyset \mid \emptyset; \mathrm{T}_i \mid \Sigma_i \vdash e : \tau \dashv \mathrm{T}_o$.

- **Rule T-sub:** By inversion of the rule we get $\emptyset \mid \emptyset; \mathrm{T}_i \mid \Sigma_i \vdash e : \tau_1 \dashv \mathrm{T}_o$ at which we apply the inductive hypothesis.
- **Rule T-let:** Let $e \equiv \mathbf{let}\ x = e_x\ \mathbf{in}\ e'$. By inversion of the rule, $\emptyset \mid \Gamma; \mathrm{T}_i \mid \Sigma_i \vdash e_x : \tau_x \dashv \mathrm{T}$. By inductive hypothesis, there are three cases: (1) If e_x is a value, the O-Let rule applies with h_i and ς_i; (2) If $\langle h_i, \varsigma_i, e_x \rangle \rightsquigarrow \mathsf{ERR}$, then, by rule O-Prop-Err, $\langle h_i, \varsigma_i, e \rangle \rightsquigarrow \mathsf{ERR}$; and (3) If $\langle h_i, \varsigma_i, e_x \rangle \rightsquigarrow \langle h_o, \varsigma_o, e_x' \rangle$, then, by rule O-Prop, $\langle h_i, \varsigma_i, \mathbf{let}\ x = e_x\ \mathbf{in}\ e' \rangle \rightsquigarrow \langle h_o, \varsigma_o, \mathbf{let}\ x = e_x'\ \mathbf{in}\ e' \rangle$.
- **Rule T-new:** Let $e \equiv \mathbf{let}\ x = \mathbf{new}(\rho)\ \mathbf{in}\ e'$. One of the rules O-New or O-New-Err applies.
- **Rule T-if:** Let $e \equiv \mathbf{if}\ e'\ \{e_1\}\ \mathbf{else}\ \{e_2\}$. By inversion of the rule, $\emptyset \mid \Gamma; \mathrm{T}_i \mid \Sigma_i \vdash e' : \mathbf{bool}[r] \dashv \mathrm{T}_o$. By inductive hypothesis, there are three cases: (1) If e' is a value, then, by Lemma 6.33, it is either **true** or **false** and one of the O-If-True or O-If-False rules applies with h_i and ς_i; (2) If $\langle h_i, \varsigma_i, e' \rangle \rightsquigarrow \mathsf{ERR}$, then, by rule O-Prop-Err, $\langle h_i, \varsigma_i, e \rangle \rightsquigarrow \mathsf{ERR}$; and (3) If $\langle h_i, \varsigma_i, e' \rangle \rightsquigarrow \langle h_o, \varsigma_o, e'' \rangle$, then, by rule O-Prop, $\langle h_i, \varsigma_i, e \rangle \rightsquigarrow \langle h_o, \varsigma_o, \mathbf{if}\ e''\ \{e_1\}\ \mathbf{else}\ \{e_2\} \rangle$.
- **Rule T-seq:** Let $e \equiv S; e'$. By inversion of the rule we get: $\emptyset \mid \emptyset; \mathrm{T}_i \mid \Sigma_i \vdash_s S \dashv \mathrm{T}_S$. By assumption we have $\emptyset \mid \mathrm{T}_i \mid \Sigma_i \vdash \langle h_i, \varsigma_i \rangle$. By Lemma **??**, we have that either (1) there exists h_o, ς_o such that $\langle h_i, \varsigma_i, S \rangle \rightsquigarrow \langle h_o, \varsigma_o \rangle$, (2) $\langle h_i, \varsigma_i, S \rangle \rightsquigarrow \mathsf{ERR}$ or (3) $S_i = \mathbf{skip}$. In case (1) and by rule O-Seq, there exists h_o, ς_o such that $\langle h_i, \varsigma_i, S; e' \rangle \rightsquigarrow \langle h_o, \varsigma_o, e' \rangle$. In case (2) and by rule O-Seq-Err, $\langle h_i, \varsigma_i, S; e' \rangle \rightsquigarrow \mathsf{ERR}$. Finally, in case (3) O-Skip applies.
- **Rule T-call:** Let $e \equiv \mathbf{call}\ e'[\overline{r}](\overline{av})$. By inversion of the rule, $\emptyset \mid \Gamma; \mathrm{T}_i \mid \Sigma_i \vdash e' : \forall \overline{a : \sigma}.\ \mathbf{fn}(\mathbf{E}; \overline{x : \tau}) \longrightarrow \overline{\eta}/\mathrm{T}_{fi}\tau\mathrm{T}_{fo} \dashv \mathrm{T}$. By inductive hypothesis, there are three cases: (1) If $\langle h_i, \varsigma_i, e' \rangle \rightsquigarrow \mathsf{ERR}$, then, by rule O-Prop-Err, $\langle h_i, \varsigma_i, e \rangle \rightsquigarrow \mathsf{ERR}$; and (2) If $\langle h_i, \varsigma_i, e' \rangle \rightsquigarrow \langle h_o, \varsigma_o, e'' \rangle$, then, by rule O-Prop, $\langle h_i, \varsigma_i, e \rangle \rightsquigarrow \langle h_o, \varsigma_o, \mathbf{call}\ e''[\overline{r}](\overline{av}) \rangle$. (3) If e' is a value, then, by Lemma 6.37 we have the following cases
  - $e' = \mathbf{rec}\ f[\overline{r'}](\overline{y})\ := \ r_b$. Arguments are a-values and by inversion of typing under the empty environment, they must be values, thus rule O-Call applies.
  - $e' = \mathbf{Vec}_{\tau_v}\mathbf{::new}$. In this case O-Vec-new applies.
  - $e' = \mathbf{Vec}_{\tau_v}\mathbf{::push}$. Let v_1 and v_2 be the two arguments of the function call. By inversion of T-call we have:

  (1) $\emptyset \mid \emptyset; \mathrm{T}_i \mid \Sigma_i \vdash v_1 : \mathbf{ptr}(\ell) \dashv \mathrm{T}_i$
  (2) $\emptyset \mid \emptyset; \mathrm{T}_i \mid \Sigma_i \vdash v_2 : \tau_v \dashv \mathrm{T}_i$
  (3) $\emptyset \vdash \mathrm{T}_1 \Rightarrow \ell \mapsto \mathbf{Vec}_{\tau_v}[n]$.

By lemmas 6.35 and 6.34 we know $v_1 = \mathrm{ptr}(\ell, t)$. Since the input state is well-typedness we have by (3) that $\ell \in \mathrm{dom}(h)$ and $\emptyset \mid \emptyset; \emptyset \mid \Sigma_i \vdash h(\ell) : \mathbf{Vec}_{\tau_v}[n] \dashv \emptyset$. By lemmas 6.36 and 6.23 we know $h(\ell) = \mathrm{ptr}(n, v')$. If $n = 0$ then O-VEC-PUSH-EMPTY applies. If $n > 0$ then $v' = \mathrm{ptr}(\ell', t')$ and $\forall j \in [0, n).\ell' + j \in \mathrm{dom}(h)$ and O-VEC-PUSH applies.

– $e' = \mathbf{Vec}_{\tau_v}$**::index_mut**. Let v_1 and v_2 be the two arguments of the function call. By inversion of T-CALL we have:

(1) $\emptyset \mid \emptyset; \mathrm{T}_i \mid \Sigma_i \vdash v_1 : \&_{\mathbf{mut}} \mathbf{Vec}_{\tau_v}[n] \dashv \mathrm{T}_i$

(2) $\emptyset \mid \emptyset; \mathrm{T}_i \mid \Sigma_i \vdash v_2 : \mathbf{int}[m] \dashv \mathrm{T}_i$

(3) $\emptyset \models 0 \le m < n$

By lemmas 6.35 and 6.34 we have $v_1 = \mathrm{ptr}(\ell, t)$ and $v_2 = z$. By Lemma 6.23 and (2) we have $z = m$, and by (3) we have (4) $0 \le z < n$. By well-formedness of the input state and (1) we have $\emptyset \mid \emptyset; \mathrm{T}_i \mid \Sigma_i \vdash h(\ell) : \mathbf{Vec}_{\tau_v}[n] \dashv \mathrm{T}_i$ and by lemmas 6.36 and 6.23 we have $h(\ell) = \mathbf{vec}_\tau(n, v')$. The case $n = 0$ is impossible because (4) must hold. Since $n > 0$, by inversion of T-VEC we have $v' = \mathrm{ptr}(\ell', t')$ and $\ell' + i \in \mathrm{dom}(\Sigma_i)$ for all $0 \le i < n$. By well-typeness of the input state we have $\mathrm{dom}(\Sigma_i) = \mathrm{dom}(h)$ and consequently $\ell' + z \in \mathrm{dom}(h)$. Therefore, the rule O-VEC-INDEX-MUT applies, and we conclude the proof.

- **Rule T-VAR:** Impossible because the typing environment Γ is empty.
- **Rule T-UNPACK:** Impossible because the typing environment Γ is empty.
- **Rules T-TRUE, T-FALSE, T-INT, T-FUN, T-PTR, and T-MEM:** e is a value.
- **Case T-ASS,T-ASS-STRG:** We have $e = p := e'$. Since the statement must be close since we are in an empty environment we have $p = \mathrm{ptr}(\ell, t)$. If $e' = v$ then either stacked borrow transition in the premise of O-ASSIGN holds and we take a step, or it does not hold and we step to an error by rule O-PROP-ERR. If e' is not a value then by inductive hypothesis either takes a step or goes to an error in either case we make progress by rule O-ASSIGN-RVAL or O-ASSIGN-ERR.
- **Case T-BSTRG:** $e = \&\mathbf{strg}\ p$. We know p must be equal to $\mathrm{ptr}(\ell, t)$ because we are in an empty environment, thus either O-STRG-REBOR applies or we step to an error if the stacked borrow transition does not hold.
- **Case T-BSMUT,T-BMUT:** $e = \&\mathbf{mut}\ p$. We know p must be equal to $\mathrm{ptr}(\ell, t)$ because we are in an empty environment, thus either O-MUT-REBOR applies or we step to an error if the stacked borrow transition does not hold.
- **Case T-BSHR:** $e = \&\mathbf{shr}\ p$. We know p must be equal to $\mathrm{ptr}(\ell, t)$ because we are in an empty environment, thus either O-SHR-REBOR applies or we step to an error if the stacked borrow transition does not hold.
- **Case T-DEREF,T-DEREF-STRG:** $e = *p$. We know p must be equal to $\mathrm{ptr}(\ell, t)$ because we are in an empty environment. If the stacked borrow transition in the premise of O-DEREF holds then ℓ must be in $\mathrm{dom}(\varsigma.\mathrm{STACKS})$. Then, by well-typedness of the input state we have that $\ell \in \mathrm{dom}(h)$ and the rule O-DEREF applies. If the stacked borrow transition does not hold then we step to an error.

□

6.5 Lemmata

Lemma 6.4 (Subtyping Transitive). *If $\Delta \vdash_{\mathsf{wf}} \tau_1$, then if $\Delta \vdash \tau_1 \preccurlyeq \tau_2$ and $\Delta \vdash \tau_2 \preccurlyeq \tau_3$, then $\Delta \vdash \tau_1 \preccurlyeq \tau_3$.*

Proof. By induction on the structure of the three types. We split cases on τ_1

- $\tau_1 \equiv B[r_1]$. Then, the left rule can be either S-idx or S-ex.
  - **Left rule is S-idx.** Then $\tau_2 \equiv B[r_2]$ and we know (1) $\Delta \vdash B[r_1] \preccurlyeq B[r_2]$. The right rule can be either S-idx or S-ex.
    - **Right rule is S-idx.** Then $\tau_3 \equiv B[r_3]$ and we know (2) $\Delta \vdash B[r_2] \preccurlyeq B[r_3]$. By inversion of (1) and (2) we get: (3) $\Delta \models r_1 = r_2$ and (4) $\Delta \models r_2 = r_3$. By (3), (4), and assumption 5, we have (5) $\Delta \models r_1 = r_3$. By rule S-idx we conclude the proof.
    - **Right rule is S-ex.** Then $\tau_3 \equiv \{a.\ B[a] \mid r_3\}$ and we know (2) $\Delta \vdash B[r_2] \preccurlyeq \{a.\ B[a] \mid r_3\}$. By inversion of (1) and (2) we get (3) $\Delta \models r_1 = r_2$ and (4) $\Delta \models r_3[r_2/a]$. By (3), (4), and assumption 7, we get (5) $\Delta \models r_3[r_1/a]$, which concludes the proof by rule S-ex.
  - **Left rule is S-ex.** Then $\tau_2 \equiv \{a.\ B[a] \mid r_2\}$ and we know (1) $\Delta \vdash B[r_1] \preccurlyeq \{a.\ B[a] \mid r_2\}$. The right rule can only be S-unpack: (2) $\Delta \vdash \{a.\ B[a] \mid r_2\} \preccurlyeq \tau_3$. By inversion of (2): (3) $\Delta, a : \mathtt{sort}(B), r_2 \vdash B[a] \preccurlyeq \tau_3$. Thus, τ_3 can be either singleton or existential and, respectively, the applied rules are can be either S-idx or S-ex.
    - **Right inverted rule is S-idx.** Then $\tau_3 \equiv B[r_3]$ and we know
$$(4)\Delta, a : \mathtt{sort}(B), r_2 \vdash B[a] \preccurlyeq B[r_3]$$
Since by assumption, $\Delta \vdash_{\mathsf{wf}} \tau_1$, $\Delta \vdash r_1 : \mathtt{sort}(B)$ by which and lemma 6.26, from (4) we get:
$$(5)\Delta, r_2[r_1/a] \vdash B[r_1] \preccurlyeq B[r_3]$$
By inversion of (1) we know that $\Delta \models r_2[r_1/a]$, by which and lemma 6.15 we get
$$(6)\Delta \vdash B[r_1] \preccurlyeq B[r_3]$$
Which concludes the proof.
    - **Right inverted rule is S-ex.** Then $\tau_3 \equiv \{a.\ B[a] \mid r_3\}$ and we know
$$(4)\Delta, a : \mathtt{sort}(B), r_2 \vdash B[a] \preccurlyeq \{a.\ B[a] \mid r_3\}$$
By inversion:
$$(5)\Delta, a : \mathtt{sort}(B), r_2 \models r_3$$
Since by assumption, $\Delta \vdash_{\mathsf{wf}} \tau_1$, $\Delta \vdash r_1 : \mathtt{sort}(B)$ by which and assumption 4, we get:
$$(6)\Delta, r_2[r_1/a] \models r_3[r_1/a]$$

By inversion of (1) we know that $\Delta \models r_2[r_1/a]$, by which and assumption 2 we get

$$(7)\Delta \models r_3[r_1/a]$$

Which concludes the proof by rule S-ex.

- $\tau_1 \equiv \{a.\ B[a] \mid r_1\}$. The left rule is S-unpack: (1) $\Delta \vdash \{a.\ B[a] \mid r_1\} \preccurlyeq \tau_2$. By inversion: (2) $\Delta, a : \mathtt{sort}(B), r_1 \vdash B[a] \preccurlyeq \tau_2$. So, the inverted right rule can be either S-idx or S-ex:
  - **Left inverted rule is S-idx.** Then, $\tau_2 \equiv B[r_2]$ and we know: (3): $\Delta, a : \mathtt{sort}(B), r_1 \vdash B[a] \preccurlyeq B[r_2]$. By inversion: (4): $\Delta, a : \mathtt{sort}(B), r_1 \models a = r_2$. Since $\Delta \vdash B[r_2] \preccurlyeq \tau_3$, the right rule can be either S-idx or S-ex:
    * **Right rule is S-idx.** Then, $\tau_3 \equiv B[r_3]$ and we know: (5): $\Delta \vdash B[r_2] \preccurlyeq B[r_3]$. By inversion: (6) $\Delta \models r_2 = r_3$. By assumption 7 and (6), (4) becomes: (7) $\Delta, a : \mathtt{sort}(B), r_1 \models a = r_3$. By (7) and rule S-idx: $\Delta, a : \mathtt{sort}(B), r_1 \vdash B[a] \preccurlyeq B[r_3]$, which concludes the proof by rule S-unpack.
    * **Right rule is S-ex.** Then $\tau_3 \equiv \{a_3.\ B[a_3] \mid r_3\}$ and we know (5) $\Delta \vdash B[r_2] \preccurlyeq \{a_3.\ B[a_3] \mid r_3\}$. By inversion: (6) $\Delta \models r_3[r_2/a_3]$. By assumption 1 on (6): (7) $\Delta, a : \mathtt{sort}(B), r_1 \models r_3[r_2/a_3]$. By (4) and assumption 7 (8) $\Delta, a : \mathtt{sort}(B), r_1 \models r_3[a/a_3]$. By rule S-ex $\Delta, a : \mathtt{sort}(B), r_1 \vdash B[a] \preccurlyeq \{a_3.\ B[a_3] \mid r_3\}$. The proof concludes by rule S-unpack.
  - **Left inverted rule is S-ex.** Then $\tau_2 \equiv \{a_2.\ B[a_2] \mid r_2\}$ and we know (3) $\Delta, a : \mathtt{sort}(B), r_1 \vdash B[a] \preccurlyeq \{a_2.\ B[a_2] \mid r_2\}$. By inversion, (4) $\Delta, a : \mathtt{sort}(B), r_1 \models r_2[a/a_2]$. The right rule is S-unpack and by inversion we get: (5) $\Delta, a_2 : \mathtt{sort}(B), r_2 \vdash B[a_2] \preccurlyeq \tau_3$. Thus the right inverted rule can be either S-idx or S-ex.
    * **Right inverted rule is S-idx.** Then $\tau_3 \equiv B[r_3]$ and we know $\Delta, a_2 : \mathtt{sort}(B), r_2 \vdash B[a_2] \preccurlyeq B[r_3]$. By inversion: (6) $\Delta, a_2 : \mathtt{sort}(B), r_2 \models a_2 = r_3$. By (6) and assumption 1: (7) $\Delta, a : \mathtt{sort}(B), r_1, a_2 : \mathtt{sort}(B), r_2 \models a_2 = r_3$. By assumption 4: (8) $\Delta, a : \mathtt{sort}(B), r_1, r_2[a/a_2] \models a = r_3$. By (4) and assumption 2: (9) $\Delta, a : \mathtt{sort}(B), r_1 \models a = r_3$. By rule S-idx: (10) $\Delta, a : \mathtt{sort}(B), r_1 \vdash B[a] \preccurlyeq B[r_3]$. Then, rule S-unpack concludes the proof.
    * **Right inverted rule is S-ex.** Then $\tau_3 \equiv \{a_3.\ B[a_3] \mid r_3\}$ and we know $\Delta, a_2 : \mathtt{sort}(B), r_2 \vdash B[a_2] \preccurlyeq \{a_3.\ B[a_3] \mid r_3\}$. By inversion: (6) $\Delta, a_2 : \mathtt{sort}(B), r_2 \models r_3[a_2/a_3]$. By (6) and assumption 1: (7) $\Delta, a : \mathtt{sort}(B), r_1, a_2 : \mathtt{sort}(B), r_2 \models r_3[a_2/a_3]$. By assumption 4: (8) $\Delta, a : \mathtt{sort}(B), r_1, r_2[a/a_2] \models r_3[a_2/a_3][a/a_2]$. By (4) and assumption 2: (9) $\Delta, a : \mathtt{sort}(B), r_1 \models r_3[a/a_3]$. The proof concludes by rules S-ex and S-unpack.
- $\tau_1 \equiv \mathbf{ptr}(\eta)$. The left rule is S-ptr and $\tau_2 \equiv \mathbf{ptr}(\eta)$. Similarly, the right rule is S-ptr and $\tau_3 \equiv \mathbf{ptr}(\eta)$. The proof concludes by rule S-ptr.
- $\tau_1 \equiv ⚡$. As before, $\tau_2 \equiv \tau_3 \equiv ⚡$ and the proof concludes by rule S-mem.
- $\tau_1 \equiv \&_{\mathbf{shr}}\ \tau_1'$. The left rule is S-shr and $\tau_2 \equiv \&_{\mathbf{shr}}\ \tau_2'$. By inversion, (1) $\Delta \vdash \tau_1' \preccurlyeq \tau_2'$. The right rule is S-shr and $\tau_3 \equiv \&_{\mathbf{shr}}\ \tau_3'$. By inversion, (2) $\Delta \vdash \tau_2' \preccurlyeq \tau_3'$. By inductive hypothesis on (1) and (2): $\Delta \vdash \tau_1' \preccurlyeq \tau_3'$, which concludes the proof by rule S-shr.

- $\tau_1 \equiv \&_{\mathbf{mut}}\ \tau_1'$. The left rule is S-MUT and $\tau_2 \equiv \&_{\mathbf{mut}}\ \tau_2'$. By inversion, (1) $\Delta \vdash \tau_1' \preccurlyeq \tau_2'$ and (2) $\Delta \vdash \tau_2' \preccurlyeq \tau_1'$. The right rule is S-MUT and $\tau_3 \equiv \&_{\mathbf{mut}}\ \tau_3'$. By inversion, (3) $\Delta \vdash \tau_2' \preccurlyeq \tau_3'$ and (4) $\Delta \vdash \tau_3' \preccurlyeq \tau_2'$. By inductive hypothesis on (1) and (3): (5) $\Delta \vdash \tau_1' \preccurlyeq \tau_3'$. By inductive hypothesis on (4) and (2): (6) $\Delta \vdash \tau_3' \preccurlyeq \tau_1'$. The proof concludes by (5), (6), and rule S-MUT.
- $\tau_1 \equiv \forall \overline{a : \sigma}.\, \mathbf{fn}(\mathrm{T}_{1i}; \overline{\tau_1}) \to \tau_{1o}/\mathrm{T}_{1o}$. Then, $\tau_2 \equiv \forall \overline{a : \sigma}.\, \mathbf{fn}(\mathrm{T}_{2i}; \overline{\tau_2}) \to \tau_{2o}/\mathrm{T}_{2o}$ and $\tau_3 \equiv \forall \overline{a : \sigma}.\, \mathbf{fn}(\mathrm{T}_{3i}; \overline{\tau_3}) \to \tau_{3o}/\mathrm{T}_{3o}$. By inversion of the two rules: (1) $\Delta, \overline{a : \sigma} \models r_2 \Rightarrow r_1$, (2) $\Delta, \overline{a : \sigma} \vdash \mathrm{T}_{2i} \Rightarrow \mathrm{T}_{1i}$, (3) $\forall i. \Delta, \overline{a : \sigma} \vdash \tau_{2i} \preccurlyeq \tau_{1i}$ (4) $\Delta, \overline{a : \sigma} \vdash \mathrm{T}_{1o} \Rightarrow \mathrm{T}_{2o}$, (5) $\Delta, \overline{a : \sigma} \vdash \tau_{1o} \preccurlyeq \tau_{2o}$, (6) $\Delta, \overline{a : \sigma} \models r_3 \Rightarrow r_2$, (7) $\Delta, \overline{a : \sigma} \vdash \mathrm{T}_{3i} \Rightarrow \mathrm{T}_{2i}$, (8) $\forall i. \Delta, \overline{a : \sigma} \vdash \tau_{3i} \preccurlyeq \tau_{2i}$ (9) $\Delta, \overline{a : \sigma} \vdash \mathrm{T}_{2o} \Rightarrow \mathrm{T}_{3o}$, and (10) $\Delta, \overline{a : \sigma} \vdash \tau_{2o} \preccurlyeq \tau_{3o}$.
  By (1), (5), and assumption 6: (11) $\Delta, \overline{a : \sigma} \models r_3 \Rightarrow r_1$. By (2), (6), and rule C-TRANS: (12) $\Delta, \overline{a : \sigma} \vdash \mathrm{T}_{3i} \Rightarrow \mathrm{T}_{1i}$. By (3), (8), and inductive hypothesis: (13) $\forall i. \Delta, \overline{a : \sigma} \vdash \tau_{3i} \preccurlyeq \tau_{1i}$. By (4), (9), and rule C-TRANS: (14) $\Delta, \overline{a : \sigma} \vdash \mathrm{T}_{1o} \Rightarrow \mathrm{T}_{3o}$. By (5), (10), and inductive hypothesis: (15) $\Delta, \overline{a : \sigma} \vdash \tau_{1o} \preccurlyeq \tau_{3o}$. The proof concludes by (11)-(15) and rule S-FUN.

□

Lemma 6.5 (Weakening). *If* $\Delta \mid \Gamma; T_i \mid \Sigma \vdash e : \tau \dashv T_o$, $\Delta \vdash_{\mathrm{wf}} \Sigma'$, *and* $\Sigma' \supseteq \Sigma$ *then* $\Delta \mid \Gamma; T_i \mid \Sigma' \vdash e : \tau \dashv T_o$

Proof. By induction on the expression typing derivation trees. Σ is only used in the rules T-CALL to ensure well-formedness, which is established by assumption, and T-PTR, where the theorem holds because $\Sigma' \supseteq \Sigma$. □

Lemma 6.6 (Context Inclusion). *If* $\Delta \vdash T_2 \Rightarrow T_1$ *and* $\Delta \mid \Gamma; T_1 \mid \Sigma \vdash e : \tau \dashv T_{1o}$, *then* $\Delta \mid \Gamma; T_2 \mid \Sigma \vdash e : \tau \dashv T_{2o}$ *and* $\Delta \vdash T_{2o} \Rightarrow T_{1o}$.

Proof. By induction on the derivation trees.

- **Rule T-SUB:** $\Delta \mid \Gamma; \mathrm{T}_1 \mid \Sigma \vdash e : \tau \dashv \mathrm{T}_{1o}$. By inversion: (1) $\Delta \mid \Gamma; \mathrm{T}_1 \mid \Sigma \vdash e : \tau_1 \dashv \mathrm{T}$, (2) $\Delta \vdash \tau_1 \preccurlyeq \tau$, and (3) $\Delta \vdash \mathrm{T} \Rightarrow \mathrm{T}_{1o}$. By inductive hypothesis on (1): (4) $\Delta \mid \Gamma; \mathrm{T}_2 \mid \Sigma \vdash e : \tau_1 \dashv \mathrm{T}_{2o}$ and (5) $\Delta \vdash \mathrm{T}_{2o} \Rightarrow \mathrm{T}$. By (2), (4), and rule T-SUB, $\Delta \mid \Gamma; \mathrm{T}_2 \mid \Sigma \vdash e : \tau \dashv \mathrm{T}_{2o}$. By (3), (5), and rule C-TRANS, (5) $\Delta \vdash \mathrm{T}_{2o} \Rightarrow \mathrm{T}_{1o}$.
- **Rule T-LET:** $\Delta \mid \Gamma; \mathrm{T}_{1i} \mid \Sigma \vdash \mathbf{let}\ x = e_x\ \mathbf{in}\ e : \tau \dashv \mathrm{T}_{1o}$. By inversion: (1) $\Delta \mid \Gamma; \mathrm{T}_{1i} \mid \Sigma \vdash e_x : \tau_x \dashv \mathrm{T}$, (2) $\Delta \mid \Gamma, x : \tau_x; \mathrm{T} \mid \Sigma \vdash e : \tau \dashv \mathrm{T}_{1o}$, and (3) $x \notin \mathrm{dom}(\Gamma)$. By inductive hypothesis on (1): (4) $\Delta \mid \Gamma; \mathrm{T}_{2i} \mid \Sigma \vdash e_x : \tau_x \dashv \mathrm{T}_2'$ and (5) $\Delta \vdash \mathrm{T}_2' \Rightarrow \mathrm{T}$. By inductive hypothesis on (2), with (5): (6) $\Delta \mid \Gamma, x{:}\tau_x; \mathrm{T}_2' \mid \Sigma \vdash e : \tau \dashv \mathrm{T}_{2o}$, and (5) $\Delta \vdash \mathrm{T}_{2o} \Rightarrow \mathrm{T}_{1o}$. The proof concludes by (5) and rule T-LET on (4), (6), and (3).
- **Rule T-NEW:** $\Delta \mid \Gamma; \mathrm{T}_{1i} \mid \Sigma \vdash \mathbf{let}\ x = \mathbf{new}(\rho)\ \mathbf{in}\ e : \tau \dashv \mathrm{T}_{1o}$. By inversion: (1) $\Delta, \rho : \mathbf{loc} \mid \Gamma, x : \mathbf{ptr}(\rho); \mathrm{T}_{1i}, \rho \mapsto \lightning \mid \Sigma \vdash e : \tau \dashv \mathrm{T}_{1o}$, (2) $\Delta \vdash_{\mathrm{wf}} \tau$, (3) $\Delta \vdash_{\mathrm{wf}} \mathrm{T}_{1o}$, (4) $x \notin \mathrm{dom}(\Gamma)$, (5) $\rho \notin \mathrm{dom}(\Delta)$. By lemma 6.7 and inductive hypothesis on (1): (6) $\Delta, \rho : \mathbf{loc} \mid \Gamma, x{:}\mathbf{ptr}(\rho); \mathrm{T}_{2i}, \rho \mapsto \lightning \mid \Sigma \vdash e : \tau \dashv \mathrm{T}_{2o}$ and (7) $\Delta \vdash \mathrm{T}_{2o} \Rightarrow \mathrm{T}_{1o}$. The proof concludes by (7) and rule T-NEW on (6), (3), (4), (5), and lemma 6.21.
- **Rule T-IF:** The proofs go by induction and transitive of context inclusion (as in the T-LET case).
- **Rule T-UNPACK:** The proof goes by induction and lemma 6.21.

- **Rule T-CALL:** $\Delta \mid \Gamma; \mathrm{T} \mid \Sigma \vdash \textbf{call}\ e[\overline{r}](\overline{av}) : \theta \cdot \tau_o \dashv \theta \cdot \mathrm{T}_o, \mathrm{T}_2$. By inversion: (1) $\forall i. \Delta \mid \Gamma; \mathrm{T} \mid \Sigma \vdash av_i : \theta \cdot \tau_i \dashv \mathrm{T}$, (2) $\Delta \mid \Gamma; \mathrm{T} \mid \Sigma \vdash e : \forall \overline{a : \sigma}.\ \textbf{fn}(\mathrm{T}_i; \overline{\tau}) \rightarrow \tau_o / \mathrm{T}_o \dashv \mathrm{T}_1, \mathrm{T}_2$, (3) $\theta = [\overline{r}/\overline{a}]$, (4) $\Delta \vdash \mathrm{T}_1 \Rightarrow \theta \cdot \mathrm{T}_i$, (5) $\forall i. \Delta \vdash r_i : \sigma_i$, (6) $\Delta \models \theta \cdot r$, and (7) $\Delta \vdash_{\mathsf{wf}} \Sigma$. Assume T', so that, (8) $\Delta \vdash \mathrm{T}' \Rightarrow \mathrm{T}$. By inductive hypothesis on (1) ans (2): (9) $\forall i. \Delta \mid \Gamma; \mathrm{T}' \mid \Sigma \vdash av_i : \theta \cdot \tau_i \dashv \mathrm{T}'$, (10) $\Delta \mid \Gamma; \mathrm{T}' \mid \Sigma \vdash e : \forall \overline{a : \sigma}.\ \textbf{fn}(\mathrm{T}_i; \overline{\tau}) \rightarrow \tau_o / \mathrm{T}_o \dashv \mathrm{T}'_{12}$, (11) $\Delta \vdash \mathrm{T}'_{12} \Rightarrow \mathrm{T}_1, \mathrm{T}_2$. Because of (11), $\mathrm{dom}(\mathrm{T}_1, \mathrm{T}_2) \subseteq \mathrm{dom}(\mathrm{T}'_{12})$. We split T'_{12} to $\mathrm{T}'_{12} \equiv \mathrm{T}'_1, \mathrm{T}'_2$, so that $\mathrm{dom}(\mathrm{T}_2) = \mathrm{dom}(\mathrm{T}'_2)$. Then, (12) $\Delta \vdash \mathrm{T}'_1 \Rightarrow \mathrm{T}_1$ and (13) $\Delta \vdash \mathrm{T}'_2 \Rightarrow \mathrm{T}_2$. By (9), (10), (3), (5), (6), (7), and rule C-TRANS on (4) and (12), from rule T-CALL we get: $\Delta \mid \Gamma; \mathrm{T}' \mid \Sigma \vdash \textbf{call}\ e[\overline{r}](\overline{av}) : \theta \cdot \tau_o \dashv \theta \cdot \mathrm{T}_o, \mathrm{T}'_2$. The proof concludes by C-FRAME and (13).
- **Rule T-VAR, T-TRUE, T-FALSE, T-INT, T-FUN, T-PTR, and T-MEM:** In these cases the context is not used and not changed, thus $\mathrm{T}_{2o} = \mathrm{T}_2$, $\mathrm{T}_{1o} = \mathrm{T}_1$, and by assumption: $\Delta \vdash \mathrm{T}_{2o} \Rightarrow \mathrm{T}_{1o}$.
- **Rule T-ASS:** By inversion, inductive hypothesis and application of the rule.
- **Rule T-ASS-STRG:** By inversion, inductive hypothesis, application of the rule and lemma 6.7.
- **Rule T-BSMUT:** $\Delta \mid \Gamma; \mathrm{T}_1 \mid \Sigma \vdash \&\textbf{mut}\ p : \&_{\textbf{mut}}\tau \dashv \mathrm{T}_1[\eta \mapsto \tau]$. By inversion: (1) $\Delta \mid \Gamma; \mathrm{T}_1 \mid \Sigma \vdash p : \textbf{ptr}(\eta) \dashv \mathrm{T}_1$ and (2) $\Delta \vdash \mathrm{T}_1(\eta) \preccurlyeq \tau$. By case splitting on the two typing rules for (1):(3) $\Delta \mid \Gamma; \mathrm{T}_2 \mid \Sigma \vdash p : \textbf{ptr}(\eta) \dashv \mathrm{T}_2$. By assumption, (2), and lemma 6.9: (5) $\Delta \vdash \mathrm{T}_2(\eta) \preccurlyeq \tau$. By (3), (5), and rule T-BSMUT: $\Delta \mid \Gamma; \mathrm{T}_2 \mid \Sigma \vdash \&\textbf{mut}\ p : \&_{\textbf{mut}}\tau \dashv \mathrm{T}_2[\eta \mapsto \tau]$. By rules C-TRANS, C-PERM, and C-FRAME, $\Delta \vdash \mathrm{T}_2[\eta \mapsto \tau] \Rightarrow \mathrm{T}_1[\eta \mapsto \tau]$, which concludes the proof.
- **Rule T-BSTRG, T-BMUT, T-BSHR, T-DEREF and T-DEREF-STRG:** By inductive hypothesis and the fact that in these cases the context is not used and not changed, thus $\mathrm{T}_{2o} = \mathrm{T}_2$, $\mathrm{T}_{1o} = \mathrm{T}_1$, and by assumption: $\Delta \vdash \mathrm{T}_{2o} \Rightarrow \mathrm{T}_{1o}$.

□

LEMMA 6.7 (CONTEXT INCLUSION EXTEND). *If* $\Delta \vdash T_2 \Rightarrow T_1$, *then* $\Delta \vdash T_2[\eta \mapsto \tau] \Rightarrow T_1[\eta \mapsto \tau]$ *and* $\Delta \vdash T_2, \eta \mapsto \tau \Rightarrow T_1, \eta \mapsto \tau$.

PROOF. By rules C-PERM, C-FRAME, and C-TRANS. □

LEMMA 6.8 (FRAMING). *If* $\Delta \vdash_{\mathsf{wf}} T_{o1}, T, T_{o2}$ *and* $\Delta \mid \Gamma; T_{i1}, T_{i2} \mid \Sigma \vdash e : \tau \dashv T_{o1}, T_{o2}$, *then* $\Delta \mid \Gamma; T_{i1}, T, T_{i2} \mid \Sigma \vdash e : \tau \dashv T_{o1}, T, T_{o2}$.

PROOF. By induction on the derivation tree.

- **Rule T-SUB:** $\Delta \mid \Gamma; \mathrm{T}_{i1}, \mathrm{T}_{i2} \mid \Sigma \vdash e : \tau \dashv \mathrm{T}_{o1}, \mathrm{T}_{o2}$. By inversion: (1) $\Delta \mid \Gamma; \mathrm{T}_{i1}, \mathrm{T}_{i2} \mid \Sigma \vdash e : \tau_1 \dashv \mathrm{T}_1, \mathrm{T}_2$, (2) $\Delta \vdash \tau_1 \preccurlyeq \tau$, and (3) $\Delta \vdash \mathrm{T}_1, \mathrm{T}_2 \Rightarrow \mathrm{T}_{o1}, \mathrm{T}_{o2}$. By inductive hypothesis on (1): (4) $\Delta \mid \Gamma; \mathrm{T}_{i1}, \mathrm{T}, \mathrm{T}_{i2} \mid \Sigma \vdash e : \tau_1 \dashv \mathrm{T}_1, \mathrm{T}, \mathrm{T}_2$. By lemma 6.10 on (3): (5) $\Delta \vdash \mathrm{T}_1, \mathrm{T}, \mathrm{T}_2 \Rightarrow \mathrm{T}_{o1}, \mathrm{T}, \mathrm{T}_{o2}$. The proof concludes by (4), (2), (5), and rule T-SUB.
- **Rule T-LET:** $\Delta \mid \Gamma; \mathrm{T}_{i1}, \mathrm{T}_{i2} \mid \Sigma \vdash \textbf{let}\ x = e_x\ \textbf{in}\ e : \tau \dashv \mathrm{T}_{o1}, \mathrm{T}_{o2}$. By inversion: (1) $\Delta \mid \Gamma; \mathrm{T}_{i1}, \mathrm{T}_{i2} \mid \Sigma \vdash e_x : \tau_x \dashv \mathrm{T}_1, \mathrm{T}_2$, (2) $\Delta \mid \Gamma, x : \tau_x; \mathrm{T}_1, \mathrm{T}_2 \mid \Sigma \vdash e : \tau \dashv \mathrm{T}_{o1}, \mathrm{T}_{o2}$, and (3) $x \notin \mathrm{dom}(\Gamma)$. By inductive hypothesis

on (1) and (2): (4) $\Delta \mid \Gamma; \mathrm{T}_{i1}, \mathrm{T}, \mathrm{T}_{i2} \mid \Sigma \vdash e_x : \tau_x \dashv \mathrm{T}_1, \mathrm{T}, \mathrm{T}_2$ and (5) $\Delta \mid \Gamma, x : \tau_x; \mathrm{T}_1, \mathrm{T}, \mathrm{T}_2 \mid \Sigma \vdash e : \tau \dashv \mathrm{T}_{o1}, \mathrm{T}, \mathrm{T}_{o2}$. The proof concludes by (3)-(5) and rule T-LET.

- **Rule T-NEW:** $\Delta \mid \Gamma; \mathrm{T}_{i1}, \mathrm{T}_{i2} \mid \Sigma \vdash \textbf{let } x = \textbf{new}(\rho) \textbf{ in } e : \tau \dashv \mathrm{T}_{o1}, \mathrm{T}_{o2}$. By inversion: (1) $\Delta, \rho : \textbf{loc} \mid \Gamma, x : \textbf{ptr}(\rho); \mathrm{T}_{i1}, \mathrm{T}_{i2}, \rho \mapsto \lightning \mid \Sigma \vdash e : \tau \dashv \mathrm{T}_{o1}, \mathrm{T}_{o2}$, (2) $\Delta \vdash_{\mathsf{wf}} \tau$, (3) $\Delta \vdash_{\mathsf{wf}} \mathrm{T}_{o1}, \mathrm{T}_{o2}$, (4) $x \notin \mathrm{dom}(\Gamma)$, and (5) $\rho \notin \mathrm{dom}(\Delta)$. By inductive hypothesis on (1): (6) $\Delta, \rho : \textbf{loc} \mid \Gamma, x : \textbf{ptr}(\rho); \mathrm{T}_{i1}, \mathrm{T}, \mathrm{T}_{i2}, \rho \mapsto \lightning \mid \Sigma \vdash e : \tau \dashv \mathrm{T}_{o1}, \mathrm{T}, \mathrm{T}_{o2}$. By assumption: (7) $\Delta \vdash_{\mathsf{wf}} \mathrm{T}_{o1}, \mathrm{T}, \mathrm{T}_{o2}$. The proof concludes by (6), (2), (7), (4), (5), and rule T-NEW.
- **Rule T-IF:** $\Delta \mid \Gamma; \mathrm{T}_{i1}, \mathrm{T}_{i2} \mid \Sigma_i \vdash \textbf{if } e \, \{e_1\} \textbf{ else } \{e_2\} : \tau \dashv \mathrm{T}_{o1}, \mathrm{T}_{o2}$. By inversion: (1) $\Delta \mid \Gamma; \mathrm{T}_{i1}, \mathrm{T}_{i2} \mid \Sigma \vdash e : \textbf{bool}[r] \dashv \mathrm{T}_1, \mathrm{T}_2$, (2) $\Delta, r \mid \Gamma; \mathrm{T}_1, \mathrm{T}_2 \mid \Sigma \vdash e_1 : \tau \dashv \mathrm{T}_{o1}, \mathrm{T}_{o2}$, and (3) $\Delta, \neg r \mid \Gamma; \mathrm{T}_1, \mathrm{T}_2 \mid \Sigma \vdash e_2 : \tau \dashv \mathrm{T}_{o1}, \mathrm{T}_{o2}$. The proof concludes by inductive hypothesis on (1)-(3) and the rule T-IF.
- **Rule T-UNPACK:** By inversion, inductive hypothesis and the well formedness assumption, and application of the rule.
- **Rule T-CALL:** $\Delta \mid \Gamma; \mathrm{T}_{i1}, \mathrm{T}_{i2} \mid \Sigma \vdash \textbf{call } e[\overline{r}](\overline{av}) : \theta \cdot \tau_o \dashv \theta \cdot \mathrm{T}_o, \mathrm{T}_2$ By inversion:

(1) $\forall i. \Delta \mid \Gamma; \mathrm{T}_{i1}, \mathrm{T}_{i2} \mid \Sigma \vdash av_i : \theta \cdot \tau_i \dashv \mathrm{T}_{i1}, \mathrm{T}_{i2}$,
(2) $\Delta \mid \Gamma; \mathrm{T}_{i1}, \mathrm{T}_{i2} \mid \Sigma \vdash e : \forall \overline{a : \sigma}.\ \textbf{fn}(\mathrm{T}_i; \overline{\tau}) \to \tau_o / \mathrm{T}_o \dashv \mathrm{T}_1, \mathrm{T}_2$,
(3) $\theta = [\overline{r}/\overline{a}]$,
(4) $\Delta \vdash \mathrm{T}_1 \Rightarrow \theta \cdot \mathrm{T}_i$,
(5) $\forall i. \Delta \vdash r_i : \sigma_i$,
(6) $\Delta \models \theta \cdot r$, and
(7) $\Delta \vdash_{\mathsf{wf}} \Sigma$.

By inductive hypothesis on (1) and (2):

(8) $\forall i. \Delta \mid \Gamma; \mathrm{T}_{i1}, \mathrm{T}, \mathrm{T}_{i2} \mid \Sigma \vdash av_i : \theta \cdot \tau_i \dashv \mathrm{T}_{i1}, \mathrm{T}, \mathrm{T}_{i2}$,
(9) $\Delta \mid \Gamma; \mathrm{T}_{i1}, \mathrm{T}, \mathrm{T}_{i2} \mid \Sigma \vdash e : \forall \overline{a : \sigma}.\ \textbf{fn}(\mathrm{T}_i; \overline{\tau}) \to \tau_o / \mathrm{T}_o \dashv \mathrm{T}_1, \mathrm{T}, \mathrm{T}_2$.

By (3)-(9) and rule T-CALL, we get the following:

$$\Delta \mid \Gamma; \mathrm{T}_{i1}, \mathrm{T}, \mathrm{T}_{i2} \mid \Sigma \vdash \textbf{call } e[\overline{r}](\overline{av}) : \theta \cdot \tau_o \dashv \theta \cdot \mathrm{T}_o, (\mathrm{T}, \mathrm{T}_2).$$

- **Rule T-FUN:** By inversion of the rule:

$$\Delta, \overline{a : \sigma}, r \mid \Gamma, \overline{x : \tau}, f : \forall \overline{a : \sigma}.\ \textbf{fn}(\mathrm{T}_i; \overline{\tau}) \to \tau / \mathrm{T}_o; \mathrm{T}_i \mid \Sigma \vdash e : \tau \dashv \mathrm{T}_o$$

- **Rule T-VAR, T-TRUE, T-FALSE, T-INT, T-FUN, T-PTR, and T-MEM:** These cases are trivial because typing preserves the location context. So, if $\Delta \mid \Gamma; \mathrm{T}_{i1}, \mathrm{T}_{i2} \mid \Sigma \vdash e : \tau \dashv \mathrm{T}_{i1}, \mathrm{T}_{i2}$, then $\Delta \mid \Gamma; \mathrm{T}_{i1}, \mathrm{T}, \mathrm{T}_{i2} \mid \Sigma \vdash e : \tau \dashv \mathrm{T}_{i1}, \mathrm{T}, \mathrm{T}_{i2}$.
- **Rule T-ASS:** by inversion, inductive hypothesis and application of the rule.
- **Rule T-ASS-STRG:** by inversion, inductive hypothesis and application of the rule.
- **Rule T-BSTRG:** by inversion, inductive hypothesis, and application of the rule.
- **Rule T-BSMUT:** $\Delta \mid \Gamma; \mathrm{T}_1, \mathrm{T}_2 \mid \Sigma \vdash \&\textbf{mut } p : \&_{\textbf{mut}} \tau \dashv (\mathrm{T}_1, \mathrm{T}_2)[\eta \mapsto \tau]$. By inversion: (1) $\Delta \mid \Gamma; \mathrm{T}_1, \mathrm{T}_2 \mid \Sigma \vdash p : \textbf{ptr}(\eta) \dashv \mathrm{T}_1, \mathrm{T}_2$ and (2) $\Delta \vdash (\mathrm{T}_1, \mathrm{T}_2)(\eta) \preccurlyeq \tau$. By inductive hypothesis on (1): (3)

$\Delta \mid \Gamma; \mathrm{T}_1, \mathrm{T}, \mathrm{T}_2 \mid \Sigma \vdash p : \mathbf{ptr}(\eta) \dashv \mathrm{T}_1, \mathrm{T}, \mathrm{T}_2$. By the disjointness assumption $(\mathrm{T}_1, \mathrm{T}_2)(\eta) = (\mathrm{T}_1, \mathrm{T}, \mathrm{T}_2)(\eta)$, thus (2) becomes: (4) $\Delta \vdash (\mathrm{T}_1, \mathrm{T}, \mathrm{T}_2)(\eta) \preccurlyeq \tau$. The proof concludes by (2), (4), and rule T-BSMUT.

- **Rule T-BMUT, T-BSHR, T-DEREF and T-DEREF-STRG:** by inversion, inductive hypothesis, and application of the rule.

□

LEMMA 6.9 (CONTEXT INCLUSION SUBTYPING). *If $\Delta \vdash T_2 \Rightarrow T_1$ and $\Delta \vdash T_1(\eta) \preccurlyeq \tau$, then $\Delta \vdash T_2(\eta) \preccurlyeq \tau$.*

PROOF. By induction on the context inclusion tree and by splitting cases in the C-FRAME rule. □

LEMMA 6.10 (INCLUSION FRAMING). *If $\Delta \vdash T_{i1}, T_{i2} \Rightarrow T_{o1}, T_{o2}$, then $\Delta \vdash T_{i1}, T, T_{i2} \Rightarrow T_{o1}, T, T_{o2}$.*

PROOF. By assumption:

$$(1)\Delta \vdash \mathrm{T}_{i1}, \mathrm{T}_{i2} \Rightarrow \mathrm{T}_{o1}, \mathrm{T}_{o2}.$$

By rule C-FRAME:

$$(2)\Delta \vdash \mathrm{T}, \mathrm{T}_{i1}, \mathrm{T}_{i2} \Rightarrow \mathrm{T}, \mathrm{T}_{o1}, \mathrm{T}_{o2}.$$

By rule C-PERM:

$$(3)\Delta \vdash \mathrm{T}_{i1}, \mathrm{T}, \mathrm{T}_{i2} \Rightarrow \mathrm{T}, \mathrm{T}_{i1}, \mathrm{T}_{i2} \text{ and } (4)\Delta \vdash \mathrm{T}, \mathrm{T}_{o1}, \mathrm{T}_{o2} \Rightarrow \mathrm{T}_{o1}, \mathrm{T}, \mathrm{T}_{o2}.$$

By (3), (2), and C-TRANS:

$$(5)\Delta \vdash \mathrm{T}_{i1}, \mathrm{T}, \mathrm{T}_{i2} \Rightarrow \mathrm{T}, \mathrm{T}_{o1}, \mathrm{T}_{o2}.$$

By (5), (4), and C-TRANS:

$$(6)\Delta \vdash \mathrm{T}_{i1}, \mathrm{T}, \mathrm{T}_{i2} \Rightarrow \mathrm{T}_{o1}, \mathrm{T}, \mathrm{T}_{o2}.$$

Which concludes the proof. □

LEMMA 6.11 (AVAL TYPING). *If $\Delta \mid \Gamma; T_i \mid \Sigma \vdash av : \tau \dashv T_o$, then for any T, so that $\Delta \vdash_{\mathsf{wf}} T$, $\Delta \mid \Gamma; T \mid \Sigma \vdash av : \tau \dashv T$.*

PROOF. By exhaustion: the typing rules for av and the type weakening part of rule T-SUB, do not depent on T. □

LEMMA 6.12 (WELL-FORMED VARIABLES). *Well-formedness ensures free variables are included in Δ:*

- *If $\Delta \vdash_{\mathsf{wf}} \tau$ then $FV(\tau) \subseteq dom(\Delta)$.*
- *If $\Delta \vdash_{\mathsf{wf}} \Gamma$ then $FV(\Gamma) \subseteq dom(\Delta)$.*
- *If $\Delta \vdash_{\mathsf{wf}} T$ then $FV(T) \subseteq dom(\Delta)$.*
- *If $\Delta \vdash_{\mathsf{wf}} \Sigma$ then $FV(\Sigma) \subseteq dom(\Delta)$.*

PROOF. By structural induction on the type and the environments. □

LEMMA 6.13 (ID SUBSTITUTION). *When a variable is not free, substitution is identity:*

- *If $a \notin FV(\tau)$, then $\tau[r/a] = \tau$.*
- *If $a \notin FV(\Gamma)$, then $\Gamma[r/a] = \Gamma$.*
- *If $a \notin FV(T)$, then $T[r/a] = T$.*
- *If $a \notin FV(\Sigma)$, then $\Sigma[r/a] = \Sigma$.*

Proof. By structural induction on the type and the environments. □

Lemma 6.14 (Id Var Substitution). $\tau[a/a] = \tau$

Proof. By structural induction on the type. □

Lemma 6.15 (Condition). *If $\Delta_1 \models r$, then:*

i If $\Delta_1, r, \Delta_2 \mid \Gamma; T_i \mid \Sigma \vdash e : \tau \dashv T_o$, then $\Delta_1, \Delta_2 \mid \Gamma; T_i \mid \Sigma \vdash e : \tau \dashv T_o$.
ii If $\Delta_1, r, \Delta_2 \vdash \tau_1 \preccurlyeq \tau_2$, then $\Delta_1, \Delta_2 \vdash \tau_1 \preccurlyeq \tau_2$.
iii If $\Delta_1, r, \Delta_2 \vdash T_i \Rightarrow T_o$, then $\Delta_1, \Delta_2 \vdash T_i \Rightarrow T_o$.

Proof. By mutual induction on the derivation trees.

Case i) We split cases on the type derivation.

- **Rule T-sub:** $\Delta_1, r, \Delta_2 \mid \Gamma; \mathrm{T}_i \mid \Sigma \vdash e : \tau \dashv \mathrm{T}_o$. By inversion: (1) $\Delta_1, r, \Delta_2 \mid \Gamma; \mathrm{T}_i \mid \Sigma \vdash e : \tau_1 \dashv \mathrm{T}$, (2) $\Delta_1, r, \Delta_2 \vdash \tau_1 \preccurlyeq \tau$, and (3) $\Delta_1, r, \Delta_2 \vdash \mathrm{T} \Rightarrow \mathrm{T}_o$. By inductive hypothesis on (1) we get (4) $\Delta_1, \Delta_2 \mid \Gamma; \mathrm{T}_i \mid \Sigma \vdash e : \tau_1 \dashv \mathrm{T}$. By (2) and case ii, we get (5) $\Delta_1, \Delta_2 \vdash \tau_1 \preccurlyeq \tau$. By (3) and case iii, we get (6) $\Delta_1, \Delta_2 \vdash \mathrm{T} \Rightarrow \mathrm{T}_o$. By (4), (5), (6), and rule T-sub, we conclude the proof.
- **Rule T-let:** $\Delta_1, r, \Delta_2 \mid \Gamma; \mathrm{T}_i \mid \Sigma \vdash \textbf{let } x = e_x \textbf{ in } e : \tau \dashv \mathrm{T}_o$. By inversion: (1) $\Delta_1, r, \Delta_2 \mid \Gamma; \mathrm{T}_i \mid \Sigma \vdash e_x : \tau_x \dashv \mathrm{T}$ and (2) $\Delta_1, r, \Delta_2 \mid \Gamma, x : \tau_x; \mathrm{T} \mid \Sigma \vdash e : \tau \dashv \mathrm{T}_o$. By inductive hypothesis: (3) $\Delta_1, \Delta_2 \mid \Gamma; \mathrm{T}_i \mid \Sigma \vdash e_x : \tau_x \dashv \mathrm{T}$ and (4) $\Delta_1, \Delta_2 \mid \Gamma, x{:}\tau_x; \mathrm{T} \mid \Sigma \vdash e : \tau \dashv \mathrm{T}_o$. By (3), (4), and rule T-let, we conclude the proof.
- **Rule T-new:** $\Delta_1, r, \Delta_2 \mid \Gamma; \mathrm{T}_i \mid \Sigma \vdash \textbf{let } x = \textbf{new}(\rho) \textbf{ in } e : \tau \dashv \mathrm{T}_o$. By inversion: (1) $\Delta_1, r, \Delta_2, \rho : \textbf{loc} \mid \Gamma, x : \textbf{ptr}(\rho); \mathrm{T}_i, \rho \mapsto \lightning \mid \Sigma \vdash e : \tau \dashv \mathrm{T}_o$, (2) $\Delta_1, r, \Delta_2 \vdash_{\mathtt{wf}} \tau$, and (3) $\Delta_1, r, \Delta_2 \vdash_{\mathtt{wf}} \mathrm{T}_o$. By inductive hypothesis on (1), and lemma 6.16 on (2) and (3): (4) $\Delta_1, \Delta_2, \rho : \textbf{loc} \mid \Gamma, x{:}\textbf{ptr}(\rho); \mathrm{T}_i, \rho \mapsto \lightning \mid \Sigma \vdash e : \tau \dashv \mathrm{T}_o$, (5) $\Delta_1, \Delta_2 \vdash_{\mathtt{wf}} \tau$, and (6) $\Delta_1, \Delta_2 \vdash_{\mathtt{wf}} \mathrm{T}_o$. By (4), (5), (6), and rule T-new, we conclude the proof.
- **Rule T-if:** (1) $\Delta_1, r, \Delta_2 \mid \Gamma; \mathrm{T}_i \mid \Sigma_i \vdash \textbf{if } e \; \{e_1\} \textbf{ else } \{e_2\} : \tau \dashv \mathrm{T}$. By inversion: (2) $\Delta_1, r, \Delta_2 \mid \Gamma; \mathrm{T}_i \mid \Sigma \vdash e : \textbf{bool}[r_e] \dashv \mathrm{T}_o$, (3) $\Delta_1, r, \Delta_2, r_e \mid \Gamma; \mathrm{T}_o \mid \Sigma \vdash e_1 : \tau \dashv \mathrm{T}$, and (4) $\Delta_1, r, \Delta_2, \neg r_e \mid \Gamma; \mathrm{T}_o \mid \Sigma \vdash e_2 : \tau \dashv \mathrm{T}$. By inductive hypothesis: (5) $\Delta_1, \Delta_2 \mid \Gamma; \mathrm{T}_i \mid \Sigma \vdash e : \textbf{bool}[r_e] \dashv \mathrm{T}_o$, (6) $\Delta_1, \Delta_2, r_e \mid \Gamma; \mathrm{T}_o \mid \Sigma \vdash e_1 : \tau \dashv \mathrm{T}$, and (7) $\Delta_1, \Delta_2, \neg r_e \mid \Gamma; \mathrm{T}_o \mid \Sigma \vdash e_2 : \tau \dashv \mathrm{T}$. By (5), (6), (7), and rule T-if we conclude the proof.
- **Rule T-unpack:** (1) $\Delta_1, r, \Delta_2 \mid \Gamma_1, x : \{a.\, B[a] \mid r\}, \Gamma_2; \mathrm{T}_i \mid \Sigma \vdash \textbf{unpack}(x, a) \textbf{ in } e : \tau \dashv \mathrm{T}_o$. By inversion: (2) $\Delta_1, r, \Delta_2, a : \mathtt{sort}(B), r \mid \Gamma_1, x{:}B[a], \Gamma_2; \mathrm{T}_i \mid \Sigma \vdash e : \tau \dashv \mathrm{T}_o$. By inductive hypothesis: (2) $\Delta_1, \Delta_2, a : \mathtt{sort}(B), r \mid \Gamma_1, x{:}B[a], \Gamma_2; \mathrm{T}_i \mid \Sigma \vdash e : \tau \dashv \mathrm{T}_o$. Which concludes the proof by rule T-unpack.
- **Rule T-call:** (1) $\Delta_1, r, \Delta_2 \mid \Gamma; \mathrm{T} \mid \Sigma \vdash \textbf{call } e[\overline{r}](\overline{av}) : \theta \cdot \tau_o \dashv \theta \cdot \mathrm{T}_o, \mathrm{T}_2$. By inversion, for $\theta = [\overline{r}/\overline{a}]$: (2) $\forall i. \Delta_1, r, \Delta_2 \mid \Gamma; \mathrm{T} \mid \Sigma \vdash av_i : \theta \cdot \tau_i \dashv \mathrm{T}$, (3) $\Delta_1, r, \Delta_2 \mid \Gamma; \mathrm{T} \mid \Sigma \vdash e : \forall \overline{a : \sigma}.\, \textbf{fn}(\mathrm{T}_i; \overline{\tau}) \to \tau_o / \mathrm{T}_o \dashv \mathrm{T}_1, \mathrm{T}_2$,

(4) $\Delta_1, r, \Delta_2 \vdash \mathrm{T}_1 \Rightarrow \theta \cdot \mathrm{T}_i$, (5) $\forall i. \Delta_1, r, \Delta_2 \vdash r_i : \sigma_i$, (6) $\Delta_1, r, \Delta_2 \models \theta \cdot r$, and (7) $\Delta_1, r, \Delta_2 \vdash_{\mathsf{wf}} \Sigma$. By inductive hypothesis on (2) and (3), case iii on (4), assumption 2 on (6), and lemma 6.16 on (5) and (7): (8) $\forall i. \Delta_1, \Delta_2 \mid \Gamma; \mathrm{T} \mid \Sigma \vdash av_i : \theta \cdot \tau_i \dashv \mathrm{T}$, (9) $\Delta_1, \Delta_2 \mid \Gamma; \mathrm{T} \mid \Sigma \vdash e : \forall \overline{a : \sigma}.\ \mathbf{fn}(\mathrm{T}_i; \overline{\tau}) \to \tau_o/\mathrm{T}_o \dashv \mathrm{T}_1, \mathrm{T}_2$, (10) $\Delta_1, \Delta_2 \vdash \mathrm{T}_1 \Rightarrow \theta \cdot \mathrm{T}_i$, (11) $\forall i. \Delta_1, \Delta_2 \vdash r_i : \sigma_i$, (12) $\Delta_1, \Delta_2 \models \theta \cdot r$, and (13) $\Delta_1, \Delta_2 \vdash_{\mathsf{wf}} \Sigma$. BY (8)-(13) and rule T-CALL we conclude the proof.

- **Rule T-FUN:** (1) $\Delta_1, r, \Delta_2 \mid \Gamma; \mathrm{T} \mid \Sigma \vdash \mathbf{rec}\ f[\overline{a}](\overline{x}) := e : \forall \overline{a : \sigma}.\ \mathbf{fn}(\mathrm{T}_i; \overline{\tau}) \to \tau/\mathrm{T}_o \dashv \mathrm{T}$. By inversion: (2) $\Delta_1, r, \Delta_2, \overline{a : \sigma}, r \mid \Gamma, \overline{x : \tau}, f : \forall \overline{a : \sigma}.\ \mathbf{fn}(\mathrm{T}_i; \overline{\tau}) \to \tau/\mathrm{T}_o; \mathrm{T}_i \mid \Sigma \vdash e : \tau \dashv \mathrm{T}_o$. By inductive hypothesis: (3) $\Delta_1, \Delta_2, \overline{a : \sigma}, r \mid \Gamma, \overline{x : \tau}, f : \forall \overline{a : \sigma}.\ \mathbf{fn}(\mathrm{T}_i; \overline{\tau}) \to \tau/\mathrm{T}_o; \mathrm{T}_i \mid \Sigma \vdash e : \tau \dashv \mathrm{T}_o$. (3) and rule T-FUN concludes the proof.
- **Rules T-VAR, T-TRUE, T-FALSE, T-INT, T-PTR, and T-MEM:** are trivial since they do not depend on the logical environment.
- **Rule T-ASS:** (1) $\Delta_1, r, \Delta_2 \mid \Gamma; \mathrm{T}_i \mid \Sigma \vdash p := e : \lightning \dashv \mathrm{T}_o$. By inversion: (2) $\Delta_1, r, \Delta_2 \mid \Gamma; \mathrm{T}_i \mid \Sigma \vdash e' : \tau_v \dashv \mathrm{T}_i$, (3) $\Delta_1, r, \Delta_2 \mid \Gamma; \mathrm{T}_i \mid \Sigma \vdash p : \&_{\mathbf{mut}} \tau \dashv \mathrm{T}_i$, and (4) $\Delta_1, r, \Delta_2 \vdash \tau_v \preccurlyeq \tau$. By inductive hypothesis on (2) and case ii on (4): (5) $\Delta_1, \Delta_2 \mid \Gamma; \mathrm{T}_i \mid \Sigma \vdash e' : \tau_v \dashv \mathrm{T}_i$, (6) $\Delta_1, \Delta_2 \vdash \tau_v \preccurlyeq \tau$. By (5), (3), (6), and rule T-ASS, we conclude the proof.
- **Rule T-ASS-STRG:** (1) $\Delta_1, r, \Delta_2 \mid \Gamma; \mathrm{T}_i \mid \Sigma \vdash_s p := e' \dashv \mathrm{T}_o[\eta \mapsto \tau]$. By inversion: (2) $\Delta_1, r, \Delta_2 \mid \Gamma; \mathrm{T}_i \mid \Sigma \vdash e' : \tau \dashv \mathrm{T}_o$ and (3) $\Delta_1, r, \Delta_2 \mid \Gamma; \mathrm{T}_i \mid p \vdash \mathbf{ptr}(\eta) : \mathrm{T}_i \dashv$. By inductive hypothesis on (2): (4) $\Delta_1, \Delta_2 \mid \Gamma; \mathrm{T}_i \mid \Sigma \vdash e' : \tau \dashv \mathrm{T}_o$. By (4), (3), and rule T-ASS-STRG we conclude the proof.
- **Rule T-BSMUT and T-BSHR:** By inversion, application of case ii, and application of the respective rule.
- **Rule T-BSTRG, T-BMUT, T-DEREF, and T-DEREF-STRG:** These cases are trivial since they do not depend on the logical environment.

Case ii) We split cases on the subtyping derivation:

- **Rule T-BSTRG:** $\Delta_1, r, \Delta_2, a : \mathtt{sort}(B), r_a \vdash B[a] \preccurlyeq B$y inductive hypothesis: (3) $\Delta_1, \Delta_2, a : \mathtt{sort}(B), r_a \vdash B[a] \preccurlyeq \tau$. (3) and rule S-UNPACK conclude the proof.
- **Rule S-EX:** (1) $\Delta_1, r, \Delta_2 \vdash B[r_1] \preccurlyeq \{a.\ B[a] \mid r_2\}$. By inversion: (2) $\Delta_1, r, \Delta_2 \models r_2[r_1/a]$. By hypothesis, (2), and assumption 2: (3) $\Delta_1, \Delta_2 \models r_2[r_1/a]$. (3) and rule S-EX conclude the proof.
- **Rule S-SHR:** $\Delta_1, r, \Delta_2 \vdash \&_{\mathbf{shr}}\ \tau_1 \preccurlyeq \&_{\mathbf{shr}}\ \tau_2$. By inversion: $\Delta_1, r, \Delta_2 \vdash \tau_1 \preccurlyeq \tau_2$. By inductive hypothesis: $\Delta_1, \Delta_2 \vdash \tau_1 \preccurlyeq \tau_2$, which, by rule S-SHR concludes the proof.
- **Rule S-MUT:** $\Delta_1, r, \Delta_2 \vdash \&_{\mathbf{mut}}\ \tau_1 \preccurlyeq \&_{\mathbf{mut}}\ \tau_2$. By inversion: $\Delta_1, r, \Delta_2 \vdash \tau_1 \preccurlyeq \tau_2$ and $\Delta_1, r, \Delta_2 \vdash \tau_2 \preccurlyeq \tau_1$. By inductive hypothesis: $\Delta_1, \Delta_2 \vdash \tau_1 \preccurlyeq \tau_2$ and $\Delta_1, \Delta_2 \vdash \tau_2 \preccurlyeq \tau_1$, which, by rule S-MUT concludes the proof.

Case iii) We split cases on the context derivation:

- **Rule C-PERM:** $\Delta_1, r, \Delta_2 \vdash \mathrm{T} \Rightarrow \mathrm{T}'$. By inversion T' is a permutation of T. By rule C-PERM, $\Delta_1, \Delta_2 \vdash \mathrm{T} \Rightarrow \mathrm{T}'$, which concludes the proof.

- **Rule C-Weak:** $\Delta_1, r, \Delta_2 \vdash \mathrm{T}, \mathrm{T}' \Rightarrow \mathrm{T}$. By rule C-Weak: $\Delta_1, \Delta_2 \vdash \mathrm{T}, \mathrm{T}' \Rightarrow \mathrm{T}$, which concludes the proof.
- **Rule C-Frame:** $\Delta_1, r, \Delta_2 \vdash \mathrm{T}, \mathrm{T}_1 \Rightarrow \mathrm{T}, \mathrm{T}_2$. By inversion: $\Delta_1, r, \Delta_2 \vdash \mathrm{T}_1 \Rightarrow \mathrm{T}_2$. By inductive hypothesis: $\Delta_1, \Delta_2 \vdash \mathrm{T}_1 \Rightarrow \mathrm{T}_2$. By which and rule C-Frame we conclude the proof.
- **Rule C-Sub:** $\Delta_1, r, \Delta_2 \vdash \eta \mapsto \tau_1 \Rightarrow \eta \mapsto \tau_2$. By inversion: $\Delta_1, r, \Delta_2 \vdash \tau_1 \preccurlyeq \tau_2$. By case ii, By inversion: $\Delta_1, \Delta_2 \vdash \tau_1 \preccurlyeq \tau_2$. Which concludes the proof by rule C-Sub.

□

Lemma 6.16 (Condition Weakening). .

(1) If $\Delta_1, r, \Delta_2 \vdash_{\mathsf{wf}} \tau$, *then* $\Delta_1, \Delta_2 \vdash_{\mathsf{wf}} \tau$.
(2) If $\Delta_1, r, \Delta_2 \vdash_{\mathsf{wf}} T$, *then* $\Delta_1, \Delta_2 \vdash_{\mathsf{wf}} T$.
(3) If $\Delta_1, r, \Delta_2 \vdash r : \sigma$, $\Delta_1, \Delta_2 \vdash r : \sigma$.

Proof. By mutual induction on the derivation trees. Note that all three derivations depend on the bindings and not the expressions of the logical environment. □

Lemma 6.17 (Substitution). *If* $\Delta \mid \Gamma_1; T_i \mid \Sigma \vdash v : \tau_x \dashv T_i$, *then If* $\Delta \mid \Gamma_1, x{:}\tau_x, \Gamma_2; T_i \mid \Sigma \vdash e : \tau \dashv T_o$, *then* $\Delta \mid \Gamma_1, \Gamma_2; T_i \mid \Sigma \vdash e[v/x] : \tau \dashv T_o$.

Proof. By induction on the derivation tree.

- **Rule T-sub:** (1) $\Delta \mid \Gamma_1, x : \tau_x, \Gamma_2; \mathrm{T}_i \mid \Sigma \vdash e[v/x] : \tau \dashv \mathrm{T}_o$. By inversion: (2) $\Delta \mid \Gamma_1, x : \tau_x, \Gamma_2; \mathrm{T}_i \mid \Sigma \vdash e[v/x] : \tau_1 \dashv \mathrm{T}$, (3) $\Delta \vdash \tau_1 \preccurlyeq \tau$, and (4) $\Delta \vdash \mathrm{T} \Rightarrow \mathrm{T}_o$. By inductive hypothesis on (2): (5) $\Delta \mid \Gamma_1, \Gamma_2; \mathrm{T}_i \mid \Sigma \vdash e[v/x] : \tau_1 \dashv \mathrm{T}$. By (5), (3), (4), and rule T-sub, we conclude the proof.
- **Rule T-let:** (1) $\Delta \mid \Gamma_1, x{:}\tau_x, \Gamma_2; \mathrm{T}_i \mid \Sigma \vdash \mathbf{let}\ x = e_x\ \mathbf{in}\ e : \tau \dashv \mathrm{T}_o$. By inversion we get (2) $\Delta \mid \Gamma_1, x : \tau_x, \Gamma_2; \mathrm{T}_i \mid \Sigma \vdash e_x : \tau_x \dashv \mathrm{T}$, (3) $\Delta \mid \Gamma_1, x{:}\tau_x, \Gamma_2, x{:}\tau_x; \mathrm{T} \mid \Sigma \vdash e : \tau \dashv \mathrm{T}_o$, and (4) $x \notin \mathrm{dom}(\Gamma_1, x{:}\tau_x, \Gamma_2)$. By inductive hypothesis on (2) and (3) we get: (5) $\Delta \mid \Gamma_1, \Gamma_2; \mathrm{T}_i \mid \Sigma \vdash e_x[v/x] : \tau_x \dashv \mathrm{T}$ and (6) $\Delta \mid \Gamma_1, \Gamma_2, x{:}\tau_x; \mathrm{T} \mid \Sigma \vdash e[v/x] : \tau \dashv \mathrm{T}_o$. By (7), we get $x \notin \mathrm{dom}(\Gamma_1, \Gamma_2)$. By (5)-(7) and rule T-let we conclude the proof.
- **Rule T-new:** (1) $\Delta \mid \Gamma_1, x{:}\tau_x, \Gamma_2; \mathrm{T}_i \mid \Sigma \vdash \mathbf{let}\ y = \mathbf{new}(\rho)\ \mathbf{in}\ e : \tau \dashv \mathrm{T}_o$. By inversion: (2) $\Delta, \rho : \mathbf{loc} \mid \Gamma_1, x{:}\tau_x, \Gamma_2, y{:}\mathbf{ptr}(\rho); \mathrm{T}_i, \rho \mapsto \lightning \mid \Sigma \vdash e : \tau \dashv \mathrm{T}_o$, (3) $\Delta \vdash_{\mathsf{wf}} \tau$, (4) $\Delta \vdash_{\mathsf{wf}} \mathrm{T}_o$, and (5) $y \notin \mathrm{dom}(\Gamma_1, x{:}\tau_x, \Gamma_2)$. By inductive hypothesis on (2): (6) $\Delta, \rho : \mathbf{loc} \mid \Gamma_1, \Gamma_2, y{:}\mathbf{ptr}(\rho); \mathrm{T}_i, \rho \mapsto \lightning \mid \Sigma \vdash e[v/x] : \tau \dashv \mathrm{T}_o$. By (5) we get (7): $y \notin \mathrm{dom}(\Gamma_1, \Gamma_2)$. By (6), (3), (4), (7), and rule T-new we conclude the proof.
- **Rule T-if:** (1) $\Delta \mid \Gamma_1, x{:}\tau_x, \Gamma_2; \mathrm{T}_i \mid \Sigma_i \vdash \mathbf{if}\ e\ \{e_1\}\ \mathbf{else}\ \{e_2\} : \tau \dashv \mathrm{T}$. By inversion: (2) $\Delta \mid \Gamma_1, x{:}\tau_x, \Gamma_2; \mathrm{T}_i \mid \Sigma \vdash e : \mathbf{bool}[r] \dashv \mathrm{T}_o$, (3) $\Delta, r \mid \Gamma_1, x : \tau_x, \Gamma_2; \mathrm{T}_o \mid \Sigma \vdash e_1 : \tau \dashv \mathrm{T}$, and (4) $\Delta, \neg r \mid \Gamma_1, x : \tau_x, \Gamma_2; \mathrm{T}_o \mid \Sigma \vdash e_2 : \tau \dashv \mathrm{T}$. By inductive hypothesis on (2) - (4): (5) $\Delta \mid \Gamma_1, \Gamma_2; \mathrm{T}_i \mid \Sigma \vdash e[v/x] : \mathbf{bool}[r] \dashv \mathrm{T}_o$, (6) $\Delta, r \mid \Gamma_1, \Gamma_2; \mathrm{T}_o \mid \Sigma \vdash e_1[v/x] : \tau \dashv \mathrm{T}$, and (7) $\Delta, \neg r \mid \Gamma_1, \Gamma_2; \mathrm{T}_o \mid \Sigma \vdash e_2[v/x] : \tau \dashv \mathrm{T}$. The proof concludes by (5)-(7) and rule T-if.
- **Rule T-unpack:** Let the environment of the typing rule be $\Gamma_{u1}, x_u{:}\{a.\ B[a] \mid r\}, \Gamma_{u2}$. We split three cases depending on where the substituted variable x is:

- $x \in \Gamma_{u1}$: (1) $\Delta \mid \Gamma_1, x{:}\tau_x, \Gamma_2, x_u{:}\{a.\ B[a] \mid r\}, \Gamma_{u2}; \mathrm{T}_i \mid \Sigma \vdash \textbf{unpack}(x_u, a)\ \textbf{in}\ e : \tau \dashv \mathrm{T}_o$. By inversion, (2) $\Delta, a : \mathtt{sort}(B), r \mid \Gamma_1, x : \tau_x, \Gamma_2, x_u : B[a], \Gamma_{u2}; \mathrm{T}_i \mid \Sigma \vdash e : \tau \dashv \mathrm{T}_o$. By inductive hypothesis (3) $\Delta, a : \mathtt{sort}(B), r \mid \Gamma_1, \Gamma_2, x_u : B[a], \Gamma_{u2}; \mathrm{T}_i \mid \Sigma \vdash e[v/x] : \tau \dashv \mathrm{T}_o$. The proof concludes by rule T-unpack, since $(\textbf{unpack}(x_u, a)\ \textbf{in}\ e)[v/x] = \textbf{unpack}(x_u, a)\ \textbf{in}\ e[v/x]$.
- $x = x_u$: (1) $\Delta \mid \Gamma_1, x : \{a.\ B[a] \mid r\}, \Gamma_2; \mathrm{T}_i \mid \Sigma \vdash \textbf{unpack}(x, a)\ \textbf{in}\ e : \tau \dashv \mathrm{T}_o$. By inversion: (2) $\Delta, a : \mathtt{sort}(B), r \mid \Gamma_1, x{:}B[a], \Gamma_2; \mathrm{T}_i \mid \Sigma \vdash e : \tau \dashv \mathrm{T}_o$. By inductive hypothesis:

$$(3)\ \Delta, a : \mathtt{sort}(B), r \mid \Gamma_1, \Gamma_2; \mathrm{T}_i \mid \Sigma \vdash e[v/x] : \tau \dashv \mathrm{T}_o.$$

  By hypothesis: $\Delta \mid \Gamma_1; \mathrm{T}_i \mid \Sigma \vdash v : \{a.\ B[a] \mid r\} \dashv \mathrm{T}_i$ and lemma 6.23:

$$(4)\ \Delta \vdash [\![v]\!] : \mathtt{sort}(B) \text{ and } (5)\ \Delta \models r[[\![v]\!]/a].$$

  By (3), (4), and lemma 6.26:

$$(6)\ \Delta, r[[\![v]\!]/a] \mid \Gamma_1[[\![v]\!]/a], \Gamma_2[[\![v]\!]/a]; \mathrm{T}_i[[\![v]\!]/a] \mid \Sigma[[\![v]\!]/a] \vdash e[v/x][[\![v]\!]/a] : \tau[[\![v]\!]/a] \dashv \mathrm{T}_o[[\![v]\!]/a].$$

  By (6), (5), and lemma 6.15:

$$(7)\ \Delta \mid \Gamma_1[[\![v]\!]/a], \Gamma_2[[\![v]\!]/a]; \mathrm{T}_i[[\![v]\!]/a] \mid \Sigma[[\![v]\!]/a] \vdash e[v/x][[\![v]\!]/a] : \tau[[\![v]\!]/a] \dashv \mathrm{T}_o[[\![v]\!]/a].$$

  By the well-formedness premise and lemmata 6.13 and 6.12:

$$(8)\ \Delta \mid \Gamma_1, \Gamma_2; \mathrm{T}_i \mid \Sigma \vdash e[v/x][[\![v]\!]/a] : \tau \dashv \mathrm{T}_o$$

  Which concludes the proof, since $(\textbf{unpack}(x, a)\ \textbf{in}\ e)[v/x] = e[v/x][[\![v]\!]/a]$.
- $x \in \Gamma_{u2}$: Similar to first case.

- **Rule T-call:** (1) $\Delta \mid \Gamma_1, x{:}\tau_x, \Gamma_2; \mathrm{T} \mid \Sigma \vdash \textbf{call}\ e[\overline{r}](\overline{av}) : \theta \cdot \tau_o \dashv \theta \cdot \mathrm{T}_o, \mathrm{T}_2$. By inversion, for $\theta = [\overline{r}/\overline{a}]$: (2) $\forall i. \Delta \mid \Gamma_1, x{:}\tau_x, \Gamma_2; \mathrm{T} \mid \Sigma \vdash av_i : \theta \cdot \tau_i \dashv \mathrm{T}$, (3) $\Delta \mid \Gamma_1, x{:}\tau_x, \Gamma_2; \mathrm{T} \mid \Sigma \vdash e : \forall \overline{a : \sigma}.\ \textbf{fn}(\mathrm{T}_i; \overline{\tau}) \to \tau_o/\mathrm{T}_o \dashv \mathrm{T}_1, \mathrm{T}_2$, (4) $\Delta \vdash \mathrm{T}_1 \Rightarrow \theta \cdot \mathrm{T}_i$, (5) $\forall i. \Delta \vdash r_i : \sigma_i$, (6) $\Delta \models \theta \cdot r$, and (7) $\Delta \vdash_{\mathsf{wf}} \Sigma$. The proof concludes by inductive hypothesis on (2) and (3) and application of the T-call rule.
- **Rule T-var:** (1) $\Delta \mid \Gamma_1, y{:}\tau_y, \Gamma_2; \mathrm{T} \mid \Sigma \vdash x : \tau \dashv \mathrm{T}$. By inversion: (2) $x{:}\tau \in \Gamma_1, y{:}\tau_y, \Gamma_2$. We split cases on wheather $x = y$:
  - $x \neq y$**:** $x[v/y] = x$ and by (2) $x{:}\tau \in \Gamma_1, \Gamma_2$. So, rule T-var concludes the proof.
  - $x = y$ $x[v/y] = v$ and by (2) $\tau = \tau_y$. So, the assumption, strengthened by lemma 6.20 with Γ_2, concludes the proof.
- **Rule T-fun:** (1) $\Delta \mid \Gamma_1, y : \tau_y, \Gamma_2; \mathrm{T} \mid \Sigma \vdash \textbf{rec}\ f[\overline{a}](\overline{x}) := e : \forall \overline{a : \sigma}.\ \textbf{fn}(\mathrm{T}_i; \overline{\tau}) \to \tau/\mathrm{T}_o \dashv \mathrm{T}$. By inversion, (2) $\Delta, \overline{a : \sigma}, r \mid \Gamma_1, y : \tau_y, \Gamma_2, \overline{x : \tau}, f : \forall \overline{a : \sigma}.\ \textbf{fn}(\mathrm{T}_i; \overline{\tau}) \to \tau/\mathrm{T}_o; \mathrm{T}_i \mid \Sigma \vdash e : \tau \dashv \mathrm{T}_o$. By inductive hypothesis, (3) $\Delta, \overline{a : \sigma}, r \mid \Gamma_1, \Gamma_2, \overline{x : \tau}, f : \forall \overline{a : \sigma}.\ \textbf{fn}(\mathrm{T}_i; \overline{\tau}) \to \tau/\mathrm{T}_o; \mathrm{T}_i \mid \Sigma \vdash e[v/y] : \tau \dashv \mathrm{T}_o$. By rule T-fun: (4) $\Delta \mid \Gamma_1, \Gamma_2; \mathrm{T} \mid \Sigma \vdash \textbf{rec}\ f[\overline{a}](\overline{x}) := e[v/y] : \forall \overline{a : \sigma}.\ \textbf{fn}(\mathrm{T}_i; \overline{\tau}) \to \tau/\mathrm{T}_o \dashv \mathrm{T}$. Because y is in the original typing environment, by the premise of the rule it is different than f and $\overline{x}$, so (5) $(\textbf{rec}\ f[\overline{a}](\overline{x}) := e)[v/y] = \textbf{rec}\ f[\overline{a}](\overline{x}) := e[v/y]$, which concludes the proof.

- **Rules T-TRUE, T-FALSE, T-INT, T-PTR, and T-MEM:** Trivial because $e[v/x] = e$ and typing does not depend on the environment Γ.
- **Rule T-ASS:** (1) $\Delta \mid \Gamma_1, x:\tau_x, \Gamma_2; \mathrm{T}_i \mid \Sigma \vdash_s p := e' \dashv \mathrm{T}_o$. By inversion: (2) $\Delta \mid \Gamma_1, x:\tau_x, \Gamma_2; \mathrm{T}_i \mid \Sigma \vdash e' : \tau_v \dashv \mathrm{T}_o$, (3) $\Delta \mid \Gamma_1, x:\tau_x, \Gamma_2; \mathrm{T}_i \mid \Sigma \vdash p : \&_{\mathbf{mut}}\tau \dashv \mathrm{T}_i$, and (4) $\Delta \vdash \tau_v \preccurlyeq \tau$. By inductive hypothesis on (2) and (3): (5) $\Delta \mid \Gamma_1, \Gamma_2; \mathrm{T}_i \mid \Sigma \vdash e[e'/x] : \tau_v \dashv \mathrm{T}_o$, (6) $\Delta \mid \Gamma_1, \Gamma_2; \mathrm{T}_i \mid \Sigma \vdash p[e'/x] : \&_{\mathbf{mut}}\tau \dashv \mathrm{T}_i$. The proof concludes by rule (5), (6), (4), and T-ASS.
- **Rule T-ASS-STRG:** (1) $\Delta \mid \Gamma_1, x:\tau_x, \Gamma_2; \mathrm{T}_i \mid \Sigma \vdash_s p := e' \dashv \mathrm{T}_o[\eta \mapsto \tau]$. By inversion: (2) $\Delta \mid \Gamma_1, x:\tau_x, \Gamma_2; \mathrm{T}_i \mid \Sigma \vdash e' : \tau \dashv \mathrm{T}_o$ and (3) $\Delta \mid \Gamma_1, x:\tau_x, \Gamma_2; \mathrm{T}_i \mid \Sigma \vdash p : \mathbf{ptr}(\eta) \dashv \mathrm{T}_i$. By inductive hypothesis on (2) and (3): (4) $\Delta \mid \Gamma_1, \Gamma_2; \mathrm{T}_i \mid \Sigma \vdash e[e'/x] : \tau \dashv \mathrm{T}_o$ and (5) $\Delta \mid \Gamma_1, \Gamma_2; \mathrm{T}_i \mid \Sigma \vdash p[e'/x] : \mathbf{ptr}(\eta) \dashv \mathrm{T}_i$. The proof concludes by (4), (5), and rule T-ASS-STRG.
- **Rules T-BSTRG, T-BSMUT, T-BMUT and T-BSHR** The proof goes by case splitting on the rules, inversion, application of the inductive hypothesis, and application of the respective rule.

□

LEMMA 6.18 (SUBTYPING REFLEXIVE). $\Delta \vdash \tau \preccurlyeq \tau$

PROOF. By induction on the structure of τ:

- $\tau \equiv B[r]$. Since, by assumption 8 $\Delta \models r = r$, by rule S-IDX we get $\Delta \vdash B[r] \preccurlyeq B[r]$.
- $\tau \equiv \{a.\ B[a] \mid r\}$. By assumption 3, we have (1) $\Delta, a : \mathtt{sort}(B), r \models r$. By lemma 6.14, we have: (2) $\Delta, a : \mathtt{sort}(B), r \models r[a/a]$. By (2) and rule S-EX, (3) $\Delta, a : \mathtt{sort}(B), r \vdash B[a] \preccurlyeq \{a.\ B[a] \mid r\}$. By (3) and rule S-UNPACK, we conclude the proof.
- $\tau \equiv \mathbf{ptr}(\eta)$. Trivially, by rule S-PTR.
- $\tau \equiv \lightning$. Trivially, by rule S-MEM.
- $\tau \equiv \&_{\mathbf{shr}}\ \tau'$. By inductive hypothesis on τ' and rule S-SHR.
- $\tau \equiv \&_{\mathbf{mut}}\ \tau'$. By inductive hypothesis on τ' and rule S-MUT.
- $\tau \equiv \forall \overline{a : \sigma}.\ \mathbf{fn}(\mathrm{T}_i; \overline{\tau}) \to \tau_o/\mathrm{T}_o$. By assumption 3: (1) $\Delta, \overline{a : \sigma} \models r \Rightarrow r$. By lemma 6.19 (2) $\Delta, \overline{a : \sigma} \vdash \mathrm{T}_i \Rightarrow \mathrm{T}_i$ and (3) $\Delta, \overline{a : \sigma} \vdash \mathrm{T}_o \Rightarrow \mathrm{T}_o$. By inductive hypothesis: (4) $\forall i.\Delta, \overline{a : \sigma} \vdash \tau_i \preccurlyeq \tau_i$ and (5) $\Delta, \overline{a : \sigma} \vdash \tau_o \preccurlyeq \tau_o$. The proof concludes by (1)-(5) and rule S-FUN.

□

LEMMA 6.19 (CONTEXT INCLUSION REFLEXIVE). *Forall* Δ *and* T, $\Delta \vdash T \Rightarrow T$.

PROOF. By rule C-PERM. □

LEMMA 6.20 (STRENGTHENING). *Let* Γ *be an environment that does not contain any variables bound at* e, Γ_1, *or* Γ_2. *If* $\Delta \mid \Gamma_1, \Gamma_2; T_i \mid \Sigma \vdash e : \tau \dashv T_o$, *then* $\Delta \mid \Gamma_1, \Gamma, \Gamma_2; T_i \mid \Sigma \vdash e : \tau \dashv T_o$.

PROOF. By induction on the derivation tree.
We split cases on the typing derivation tree:

- **Rule T-SUB:** By inductive hypothesis and rule T-SUB.
- **Rule T-LET:** (1) $\Delta \mid \Gamma_1, \Gamma_2; \mathrm{T}_i \mid \Sigma \vdash \textbf{let}\ x = e_x\ \textbf{in}\ e : \tau \dashv \mathrm{T}_o$. By inversion: (2) $\Delta \mid \Gamma_1, \Gamma_2; \mathrm{T}_i \mid \Sigma \vdash e_x : \tau_x \dashv \mathrm{T}$, (3) $\Delta \mid \Gamma_1, \Gamma_2, x{:}\tau_x; \mathrm{T} \mid \Sigma \vdash e : \tau \dashv \mathrm{T}_o$, and (4) $x \notin \mathrm{dom}(\Gamma_1, \Gamma_2)$. By inductive hypothesis on (2) and (3): (5) $\Delta \mid \Gamma_1, \Gamma, \Gamma_2; \mathrm{T}_i \mid \Sigma \vdash e_x : \tau_x \dashv \mathrm{T}$ and (6) $\Delta \mid \Gamma_1, \Gamma, \Gamma_2, x{:}\tau_x; \mathrm{T} \mid \Sigma \vdash e : \tau \dashv \mathrm{T}_o$. By hypothesis and (4): (7) $x \notin \mathrm{dom}(\Gamma_1, \Gamma, \Gamma_2)$. By (5)-(7) and rule T-LET we conclude the proof.
- **Rule T-NEW:** By inductive hypothesis, assumption, and rule T-NEW.
- **Rule T-IF:** By inductive hypothesis and rule T-IF.
- **Rule T-UNPACK:** We split two cases based on the location of x. Both go by inductive hypothesis and rule T-UNPACK.
- **Rule T-CALL:** By inductive hypothesis and rule T-CALL.
- **Rule T-VAR:** Since the domains of the environments are disjoint, $x{:}\tau \in \Gamma_1, \Gamma, \Gamma_2$.
- **Rule T-FUN:** By inductive hypothesis and rule T-FUN.
- **Rules T-TRUE, T-FALSE, T-INT, T-PTR, T-MEM:** Trivial since the rules do not depend on the typing environment.
- **Rules T-ASS and T-ASS-STRG** By inductive hypothesis.
- **Rules T-BSTRG, T-BSMUT, T-BMUT, T-BSHR, T-DEREF and T-DEREF-STRG:** By inductive hypothesis.

□

LEMMA 6.21 (WELL FORMEDNESS). *If* $\Delta \mid \Gamma; T_i \mid \Sigma \vdash e : \tau \dashv T_o$ *and* $\Delta \vdash_{\mathsf{wf}} \Gamma$, $\Delta \vdash_{\mathsf{wf}} T_i$, $\Delta \vdash_{\mathsf{wf}} \Sigma$, *then* $\Delta \vdash_{\mathsf{wf}} T_o$ *and* $\Delta \vdash_{\mathsf{wf}} \tau$.

PROOF. By induction on the type derivation trees. □

LEMMA 6.22 (SUBTYPING WEAKENING).

(1) If $\Delta_1, \Delta_2 \vdash \tau_1 \preccurlyeq \tau_2$ *then* $\Delta_1, \Delta', \Delta_2 \vdash \tau_1 \preccurlyeq \tau_2$.
(2) If $\Delta_1, \Delta_2 \vdash T_1 \Rightarrow T_2$ *then* $\Delta_1, \Delta', \Delta_2 \vdash T_1 \Rightarrow T_2$.

PROOF. By mutual induction in the definition of subtyping and context inclusion generalizing over Δ_2.

- **Case S-PTR, S-MEM:** Direct by applying the rules since they do not depend on the logical context.
- **Case S-MUT, S-SHR:** By applying the induction hypothesis for (1)
- **Case S-IDX:** By assumption 1
- **Case S-UNPACK:** By applying the induction hypothesis for (1) picking $\Delta_2, a : \mathtt{sort}(B), r$ for the generalized logical context.
- **Case S-EX:** By assumption 1.
- **Case S-FUN:** By inductive hypotheses and assumption 1.
- **Case C-PERM,C-WEAK:** Direct by applying the rules since they do not depend on the logical context.
- **Case C-FRAME, C-SUB:** By inductive hypotheses.

□

Lemma 6.23 (Value refinement). *Given* $\Delta \mid \Gamma; T \mid \Sigma \vdash v : \tau \dashv T$. *The following holds:*

(1) If $\tau = B[r]$ *then* $\Delta \vdash [\![v]\!] : \mathtt{sort}(B)$ *and* $\Delta \models r = [\![v]\!]$.

(2) If $\tau = \{a.\ B[a] \mid r\}$ *then* $\Delta \vdash [\![v]\!] : \mathtt{sort}(B)$ *and* $\Delta \models r[[\![v]\!]/a]$

Proof. By induction on the typing derivation generalizing over Δ. The following cases apply to values.

- **Case T-int,T-true, T-false:** Only the premise of (1) holds. The conclusion is direct since the rules assign the exact value to the index.
- **Case T-vec:** Only the premise of (1) holds. The conclusion is direct since the rule assigns the length of the vector as the index which is exactly the interpretation of vectors.
- **Case T-sub:** By inversion of the rule we have:

  (3) $\Delta \mid \Gamma; \mathrm{T} \mid \Sigma \vdash v : \tau' \dashv \mathrm{T}$

  (4) $\Delta \vdash \tau' \preccurlyeq \tau$

  We prove (1) and (2) separately, assuming the premise in each case:
  - $\tau = B[r]$. By inversion of (4) we have two cases
    - **Case S-idx:** (4) becomes $\Delta \vdash B[e'] \preccurlyeq B[e]$. The conclusion of (1) then follows from the premise of S-idx, the inductive hypotheses applied to (3) and Assumption 5.
    - **Case S-unpack:** (3) becomes $\Delta \mid \Gamma; \mathrm{T} \mid \Sigma \vdash v : \{a.\ B[a] \mid r'\} \dashv \mathrm{T}$. and (4) becomes $\Delta \vdash \{a.\ B[a] \mid r'\} \preccurlyeq B[r]$. By inversion of (4) we have $\Delta, a : \mathtt{sort}(B), r' \vdash B[a] \preccurlyeq B[r]$, and again by inversion we have $\Delta, a : \mathtt{sort}(B), r' \models a = r$. By the inductive hypothesis applied to (3) we have $\Delta \models r'[[\![v]\!]/a]$ and $\Delta \vdash [\![v]\!] : \mathtt{sort}(B)$. By Lemma 6.24 applied to (3) we have (5) $\Delta, a : \mathtt{sort}(B), r' \mid \Gamma; \mathrm{T} \mid \Sigma \vdash v : B[a] \dashv \mathrm{T}$. Applying the inductive hypothesis to (5) we get $\Delta, a : \mathtt{sort}(B), r' \models a = [\![v]\!]$. Summarizing, we have (a) $\Delta, a : \mathtt{sort}(B), r' \models a = r$, (b) $\Delta \models r'[[\![v]\!]/a]$, and (c) $\Delta, a : \mathtt{sort}(B), r' \models a = [\![v]\!]$. We can apply Assumption 4 with (a) and (b). To get $\Delta, r'[[\![v]\!]/a] \models [\![v]\!] = r$ and then apply Assumption 2 to get $\Delta \models [\![v]\!] = r$ which concludes this part of the proof.
  - $\tau = \{a.\ B[a] \mid r\}$. By inversion of (4) we have two cases:
    - **Case S-ex:** (4) becomes $\Delta \vdash B[r'] \preccurlyeq \{a.\ B[a] \mid r\}$. By inversion of (4) we have (5) $\Delta \models r[r'/a]$. By applying the inductive hypothesis to (3) we have $\Delta \vdash [\![v]\!] : \mathtt{sort}(B)$ and (6) $\Delta \models r' = [\![v]\!]$. We conclude this part of the proof by Assumption 7.
    - **Case S-unpack:** (3) becomes $\Delta \mid \Gamma; \mathrm{T} \mid \Sigma \vdash v : \{a.\ B[a] \mid r'\} \dashv \mathrm{T}$ and (4) becomes $\Delta \vdash \{a.\ B[a] \mid r'\} \preccurlyeq \{a.\ B[a] \mid r\}$. By inverting (4) twice we get $\Delta, a : \mathtt{sort}(B), r' \models r$. Applying the inductive hypothesis to (3) we get $\Delta \models r'[[\![v]\!]/a]$ and $\Delta \vdash [\![v]\!] : \mathtt{sort}(B)$. By using Assumption 4 we get $\Delta, r'[[\![v]\!]/a] \models r[[\![v]\!]/a]$. We conclude by using Assumption 2.

□

Lemma 6.24. *If* $\Delta \mid \Gamma; T \mid \Sigma \vdash v : \{a.\ B[a] \mid r\} \dashv T$ *then* $\Delta, a : \mathtt{sort}(B), r \mid \Gamma; T \mid \Sigma \vdash v : B[a] \dashv T$

PROOF. By applying T-SUB and using T-UNPACK and Lemma 6.25 to prove its premises. □

LEMMA 6.25 (LOGICAL CONTEXT WEAKENING). *If* $\Delta_1, \Delta_2 \mid \Gamma; T_i \mid \Sigma \vdash e : \tau \dashv T_o$ *then* $\Delta_1, \Delta', \Delta_2 \mid \Gamma; T_i \mid \Sigma \vdash e : \tau \dashv T_o$.

PROOF. By induction in the typing derivation generalizing over Δ_2. Most cases follow by applying the rule directly or by the inductive hypotheses. The interesting cases are:

- **Case T-ASS,T-BSHR,T-BSMUT,T-SUB** By applying the inductive hypotheses and Lemma 6.22.
- **Case T-CALL:** Follows from assumption 1, the inductive hypothesss, and Lemma 6.22.

□

LEMMA 6.26 (TYPE SUBSTITUTION). *Given* $\Delta_1 \vdash r : \sigma$ *and* $\Delta_1, a : \sigma, \Delta_2 \mid \Gamma; T_i \mid \Sigma \vdash e : \tau \dashv T_o$ *then* $\Delta_1, \Delta_2[r/a] \mid \Gamma[r/a]; T_i[r/a] \mid \Sigma[r/a] \vdash e[r/a] : \tau[r/a] \dashv T_o[r/a]$

PROOF. By induction on the typing derivation.

- **Case T-SUB:** Inverting the rules we have:
  (1) $\Delta_1, a : \sigma, \Delta_2 \mid \Gamma; \mathrm{T}_i \mid \Sigma \vdash e : \tau_1 \dashv \mathrm{T}$
  (2) $\Delta_1, a : \sigma, \Delta_2 \vdash \tau_1 \preccurlyeq \tau$
  (3) $\Delta_1, a : \sigma, \Delta_2 \vdash \mathrm{T} \Rightarrow \mathrm{T}_o$
  We apply the inductive hypothesis to (1) and Lemma 6.27 to (2) and (3) to get
  (1) $\Delta_1, \Delta_2[r/a] \mid \Gamma[r/a]; \mathrm{T}_i[r/a] \mid \Sigma[r/a] \vdash e[r/a] : \tau_1[r/a] \dashv \mathrm{T}[r/a]$
  (2) $\Delta_1, \Delta_2[r/a] \vdash \tau_1[r/a] \preccurlyeq \tau[r/a]$
  (3) $\Delta_1, \Delta_2[r/a] \vdash \mathrm{T}[r/a] \Rightarrow \mathrm{T}_o[r/a]$
- **Case T-LET:** By inversion of the rule:
  (1) $\Delta_1, a : \sigma, \Delta_2 \mid \Gamma; \mathrm{T}_i \mid \Sigma \vdash e_x : \tau_x \dashv \mathrm{T}$
  (2) $\Delta_1, a : \sigma, \Delta_2 \mid \Gamma, x : \tau_x; \mathrm{T} \mid \Sigma \vdash e_x : \tau \dashv \mathrm{T}_o$
  (3) $x \notin \mathrm{dom}(\Gamma)$
  Applying inductive hypothesis we get
  (1) $\Delta_1, \Delta_2[r/a] \mid \Gamma[r/a]; \mathrm{T}_i[r/a] \mid \Sigma[r/a] \vdash e_x[r/a] : \tau_x[r/a] \dashv \mathrm{T}[r/a]$
  (2) $\Delta_1, \Delta_2[r/a] \mid (\Gamma, x : \tau_x)[r/a]; \mathrm{T}[r/a] \mid \Sigma[r/a] \vdash e_x[r/a] : \tau[r/a] \dashv \mathrm{T}_o[r/a]$
  By definition we have $(\Gamma, x : \tau_x)[r/a] = \Gamma[r/a], x : \tau_x[r/a]$ and $\mathrm{dom}(\Gamma) = \mathrm{dom}(\Gamma[r/a])$, thus we conclude by applying T-LET.
- **Case T-NEW:** By inversion of the rule:
  (1) $\Delta_1, a : \sigma, \Delta_2, \rho : \mathbf{loc} \mid \Gamma, x : \mathbf{ptr}(\rho); \mathrm{T}_i, \rho \mapsto \lightning \mid \Sigma \vdash e : \tau \dashv \mathrm{T}_o$
  (2) $\Delta_1, a : \sigma, \Delta_2 \vdash_{\mathtt{wf}} \tau$
  (3) $\Delta_1, a : \sigma, \Delta_2 \vdash_{\mathtt{wf}} \mathrm{T}_o$
  (4) $x \notin \mathrm{dom}(\Gamma)$
  (5) $\rho \notin \mathrm{dom}(\Delta_1, a : \sigma, \Delta_2)$

Applying the inductive hypothesis and Lemma 6.28 we have

(1) $\Delta_1, (\Delta_2, \rho : \mathbf{loc})[r/a] \mid (\Gamma, x : \mathbf{ptr}(\rho))[r/a]; (\mathrm{T}_i, \rho \mapsto \lightning)[r/a] \mid \Sigma[r/a] \vdash e[r/a] : \tau[r/a] \dashv \mathrm{T}_o[r/a]$

(2) $\Delta_1, \Delta_2[r/a] \vdash_{\mathsf{wf}} \tau[r/a]$

(3) $\Delta_1, \Delta_2[r/a] \vdash_{\mathsf{wf}} \mathrm{T}_o[r/a]$

By (5) we have $\rho \neq a$, then by definition of substitution we have

$$\Delta_1, \Delta_2[r/a], \rho : \mathbf{loc} \mid \Gamma[r/a], x : \mathbf{ptr}(\rho); \mathrm{T}_i[r/a], \rho \mapsto \lightning \mid \Sigma[r/a] \vdash e[r/a] : \tau[r/a] \dashv \mathrm{T}_o[r/a]$$

We conclude by applying T-NEW.

- **Case T-IF:** Similarly, follows by applying the inductive hypothesis and definition of substitution.
- **Case T-UNPACK** By inversion of the rule:

(1) $\Delta_1, a : \sigma, \Delta_2 \mid \Gamma_1, x : B[b], \Gamma_2; \mathrm{T}_i \mid \Sigma \vdash e : \tau \dashv \mathrm{T}_o$

(2) $\Delta_1, a : \sigma, \Delta_2 \vdash_{\mathsf{wf}} \tau$

(3) $\Delta_1, a : \sigma, \Delta_2 \vdash_{\mathsf{wf}} \mathrm{T}_o$

(4) $b \notin \mathrm{dom}(\Delta_1, a : \sigma, \Delta_2)$

Applying the inductive hypothesis and Lemma 6.28 we hve

(1) $\Delta_1, (\Delta_2, b : \sigma)[r/a] \mid (\Gamma_1, x : B[b], \Gamma_2)[r/a]; \mathrm{T}_i[r/a] \mid \Sigma[r/a] \vdash e[r/a] : \tau[r/a] \dashv \mathrm{T}_o[r/a]$

(2) $\Delta_1, \Delta_2[r/a] \vdash_{\mathsf{wf}} \tau[r/a]$

(3) $\Delta_1, \Delta_2[r/a] \vdash_{\mathsf{wf}} \mathrm{T}_o[r/a]$

Because of (4) then $a \neq b$ and consequently we have $\Delta_1, \Delta_2[r/a], b : \sigma \mid (\Gamma_1, x : B[b], \Gamma_2)[r/a]; \mathrm{T}_i[r/a] \mid \Sigma[r/a] \vdash e[r/a] : \tau[r/a] \dashv \mathrm{T}_o[r/a]$. We conclude by applying T-UNPACK.

- **Case T-CALL** By inversion of the rule: By induction hypothesis, Assumption 4 and and lemmas 6.27, 6.28 and 6.29.
- **Case T-FUN** By inversion of the rule:

(1) $f \notin \mathrm{dom}(\Gamma)$

(2) $\forall i. x_i \notin \mathrm{dom}(\Gamma)$

(3) $\forall i. b_i \notin \mathrm{dom}(\Delta_1, a : \sigma, \Delta_2)$

(4) $\Delta_1, a : \sigma, \Delta_2, \overline{b : \sigma}, r' \mid \Gamma, \overline{x : \tau}, f : \forall \overline{b : \sigma}.\ \mathbf{fn}(\mathrm{T}_i; \overline{\tau}) \to \tau / \mathrm{T}_o; \mathrm{T}_i \mid \Sigma \vdash e : \tau \dashv \mathrm{T}_o$

By applying the inductive hypothesis:

(4) $\Delta_1, (\Delta_2, \overline{b : \sigma}, r')[r/a] \mid (\Gamma, \overline{x : \tau}, f : \forall \overline{b : \sigma}.\ \mathbf{fn}(\mathrm{T}_i; \overline{\tau}) \to \tau / \mathrm{T}_o)[r/a]; \mathrm{T}_i[r/a] \mid \Sigma[r/a] \vdash e[r/a] : \tau[r/a] \dashv \mathrm{T}_o[r/a]$

By (3) we have $b_i \neq a$ for all i and by definition of substitution we have:

(4) $\Delta_1, \Delta_2[r/a], b : \sigma, r' \mid \Gamma[r/a], \overline{x : \tau}, f : \forall \overline{b : \sigma}.\ \mathbf{fn}(\mathrm{T}_i[r/a]; \overline{\tau[r/a]}) \to \tau[r/a] / \mathrm{T}_o[r/a]; \mathrm{T}_i[r/a] \mid \Sigma[r/a] \vdash e[r/a] : \tau[r/a] \dashv \mathrm{T}_o[r/a]$

We conclude by applying T-FUN.

- **Case T-VAR,T-TRUE,T-FALSE,T-INT,T-MEM,T-PTR:** Direct since they apply to expressions closed under Δ.

- **Cases T-ASS-STRG, T-ASS, T-BSMUT, T-BMUT, T-BSTRG, T-DEREF and T-DEREF-STRG** All cases follow directly by inductive hypothesis and Lemma 6.27

□

LEMMA 6.27 (SUBTYPING SUBSTITUTION). *Given* $\Delta_1 \vdash r : \sigma$*, then*

i If $\Delta_1, a : \sigma, \Delta_2 \vdash \tau_1 \preccurlyeq \tau_2$ *then* $\Delta_1, \Delta_2[r/a] \vdash \tau_1[r/a] \preccurlyeq \tau_2[r/a]$.
ii If $\Delta_1, a : \sigma, \Delta_2 \vdash T_1 \Rightarrow T_2$ *then* $\Delta_1, \Delta_2[r/a] \vdash T_1[r/a] \Rightarrow T2[r/a]$.

PROOF. By mutual induction on the derivation trees:
Case i)

- **Case S-PTR and S-MEM:** By applying the rule since they apply to arbitrary context.
- **Case S-IDX:** By inversion of the rule we have $\Delta_1, a : \sigma, \Delta_2 \models r_1 = r_2$. Thus, by assumption 4 we know $\Delta_1, \Delta_2[r/a] \models (r_1 = r_2)[r/a]$. Therefore, by definition of substitution, $\Delta_1, \Delta_2[r/a] \models r[r/a] = r_[r/a]$. Applying rule S-IDX we have $\Delta_1, \Delta_2[r/a] \vdash B[r/a][r_1[r/a]] \preccurlyeq B[r/a][r_2[r/a]]$. By definition of substitution
$$\Delta_1, \Delta_2[r/a] \vdash B[r_1][r/a] \preccurlyeq B[r_2][r/a]$$
which concludes this part of the proof.
- **Case S-UNPACK:** By inversion of the rule we have
$$\Delta_1, a : \sigma, \Delta_2, b : \texttt{sort}(B), r' \vdash B[b] \preccurlyeq \tau$$
Applying, the inductive hypothesis we get
$$\Delta_1, (\Delta_2, b : \texttt{sort}(B), r')[r/a] \vdash B[b][r/a] \preccurlyeq \tau[r/a]$$
By definition of substitution and since b must be fresh:
$$\Delta_1, \Delta_2[r/a], b : \texttt{sort}(B), r'[r/a] \vdash B[b][r/a] \preccurlyeq \tau[r/a]$$
Now we apply rule S-UNPACK to get
$$\Delta_1, \Delta_2[r/a] \vdash \{b.\ B[b] \mid r'[r/a]\} \preccurlyeq \tau[r/a]$$
And by definition of substitution we have
$$\Delta_1, \Delta_2[r/a] \vdash (\{b.\ B[b] \mid r'\})[r/a] \preccurlyeq \tau[r/a]$$
which is the desired goal.
- **Case S-EX:** By inversion of the rule we have $\Delta_1, a : \sigma, \Delta_2 \models r_2[r_1/b]$ Thus, by Assumption 4, $\Delta_1, \Delta_2[r/a] \models r_2[r_1/b][r/a]$. Since b must be fresh then $b \notin \mathrm{FV}(r)$ and by Lemma 6.31 we have $\Delta_1, \Delta_2[r/a] \models r[r/a][r_1[r/a]/b]$. Now, applying S-EX we get:
$$\Delta_1, \Delta_2[r/a] \vdash B[r_1[r/a]] \preccurlyeq \{b.\ B[b] \mid r_2[r/a]\}$$

Finally, by definition of substitution we get

$$\Delta_1, \Delta_2[r/a] \vdash B[r_1][r/a] \preccurlyeq \{b.\ B[b] \mid r_2\}[r/a]$$

which concludes this part of the proof.

- **S-mut and S-shr:** By the inductive hypothesis
- **S-fun:** By inversion of the rule

(1) $\Delta_1, a : \sigma, \Delta_2, \overline{b : \sigma_b} \models r_2 \Rightarrow r_1$
(2) $\Delta_1, a : \sigma, \Delta_2, \overline{b : \sigma_b} \vdash \mathrm{T}_{2i} \Rightarrow \mathrm{T}_{1i}$
(3) $\forall i. \Delta_1, a : \sigma, \Delta_2, \overline{b : \sigma_b} \vdash \tau_{2i} \preccurlyeq \tau_{1i}$
(4) $\Delta_1, a : \sigma, \Delta_2, \overline{b : \sigma_b} \vdash \mathrm{T}_{1o} \Rightarrow \mathrm{T}_{2o}$
(5) $\Delta_1, a : \sigma, \Delta_2, \overline{b : \sigma_b} \vdash \tau_{1o} \preccurlyeq \tau_{2o}$

Applying the inductive hypotheses and Assumption 4 get get:

(1) $\Delta_1, (\Delta_2, \overline{b : \sigma_b})[r/a] \models (r_2 \Rightarrow r_1)[r/a]$
(2) $\Delta_1, (\Delta_2, \overline{b : \sigma_b})[r/a] \vdash \mathrm{T}_{2i}[r/a] \Rightarrow \mathrm{T}_{1i}[r/a]$
(3) $\forall i. \Delta_1, (\Delta_2, \overline{b : \sigma_b})[r/a] \vdash \tau_{2i}[r/a] \preccurlyeq \tau_{1i}[r/a]$
(4) $\Delta_1, (\Delta_2, \overline{b : \sigma_b})[r/a] \vdash \mathrm{T}_{1o}[r/a] \Rightarrow \mathrm{T}_{2o}[r/a]$
(5) $\Delta_1, (\Delta_2, \overline{b : \sigma_b})[r/a] \vdash \tau_{1o}[r/a] \preccurlyeq \tau_{2o}[r/a]$

Since $\overline{b}$ need to be fresh we get

(1) $\Delta_1, \Delta_2[r/a], \overline{b : \sigma_b} \models (r_2 \Rightarrow r_1)[r/a]$
(2) $\Delta_1, \Delta_2[r/a], \overline{b : \sigma_b} \vdash \mathrm{T}_{2i}[r/a] \Rightarrow \mathrm{T}_{1i}[r/a]$
(3) $\forall i. \Delta_1, \Delta_2[r/a], \overline{b : \sigma_b} \vdash \tau_{2i}[r/a] \preccurlyeq \tau_{1i}[r/a]$
(4) $\Delta_1, \Delta_2[r/a], \overline{b : \sigma_b} \vdash \mathrm{T}_{1o}[r/a] \Rightarrow \mathrm{T}_{2o}[r/a]$
(5) $\Delta_1, \Delta_2[r/a], \overline{b : \sigma_b} \vdash \tau_{1o}[r/a] \preccurlyeq \tau_{2o}[r/a]$

We conclude by applying S-fun.

Case ii)

- **Case C-trans:** By inductive hypothesis ii.
- **Case C-Perm:** Direct since if T′ is a permutation of T then T′$[r/a]$ is also a permutation of T$[r/a]$.
- **Case C-Weak:** By Lemma 6.32.
- **Case C-Frame:** By the inductive hypothesis ii. and Lemma 6.32.
- **Case C-Sub:** By the inductive hypothesis i.

□

Lemma 6.28 (Well-formedness Substitution). *Given $\Delta_1 \vdash r : \sigma$, then*

i If $\Delta_1, a : \sigma, \Delta_2 \vdash_{\mathsf{wf}} \tau$ then $\Delta_1, \Delta_2[r/a] \vdash_{\mathsf{wf}} \tau[r/a]$
ii If $\Delta_1, a : \sigma, \Delta_2 \vdash_{\mathsf{wf}} T$ then $\Delta_1, \Delta_2[r/a] \vdash_{\mathsf{wf}} T[r/a]$
iii If $\Delta_1, a : \sigma, \Delta_2 \vdash_{\mathsf{wf}} \Sigma$ then $\Delta_1, \Delta_2[r/a] \vdash_{\mathsf{wf}} \Sigma[r/a]$

Proof. Each one by induction on the derivation tree. □

Lemma 6.29 (Well-sortedness Substitution). *If* $\Delta_1 \vdash r : \sigma$*,* $\Delta_1, a : \sigma, \Delta_2 \vdash r' : \sigma'$ *and* $\vdash_{\mathtt{wf}} \Delta_1, \Delta_2$ *then* $\Delta_1, \Delta_2[r/a] \vdash r'[r/a] : \sigma'$

Proof. By induction on the derivation tree applying Lemma 6.30 and the assumption $\Delta_1 \vdash r : \sigma$ on the variable case. □

Lemma 6.30 (Well-sortedness Weakening). *If* $\Delta_1 \vdash r : \sigma$ *and* $\vdash_{\mathtt{wf}} \Delta_1, \Delta_2$ *then* $\Delta_2 \vdash r : \sigma$*.*

Proof. By induction on the derivation tree noting that if $a : \sigma \in \Delta_1$ and $\vdash_{\mathtt{wf}} \Delta_1, \Delta_2$ then $a : \sigma \in \Delta_1, \Delta_2$. □

Lemma 6.31 (Substitution composition). *If* $a \notin FV(r_b)$ *then* $(r[r_a/a])[r_b/b] = (r[r_b/b])[r_a[r_b/b]/a]$

Proof. By induction on the structure of r. The interesting case is the variable case. Let $r = c$ for some variable c. We have three cases

- $c \neq a \wedge c \neq b$: By applying the definition of substitution we get c in both sides of the equality.
- $c = a$: By applying the definition of substitution on the left side of the equality we get $r_a[r_b/b]$. Applying, the definition of substitution on the right side once we get $a[r_a[r_b/b]/a]$. Applying it a second time we get $r_a[r_b/b]$.
- $c = b$: On the left side we have

  $$(b[r_a/a])[r_b/b] = b[r_b/b] = r_b$$

  On the right side we have

  $$(b[r_b/b])[r_a[r_b/b]/a] = r_b[r_a[r_b]/a]$$

  By the assumption $a \notin \mathrm{FV}(r_b)$ we get

  $$r_b[r_a[r_b]/a] = r_b$$

  which concludes the proof.

□

Lemma 6.32 (Substitution Concatenation). $(T_1, T_2)[r/a] = T_1[r/a], T_2[r/a]$

Proof. By induction on the structure of T_2 applying the definition of substitution. □

Lemma 6.33 (Canonical Forms Booleans). *If* $\Delta \mid \Gamma; T_i \mid \Sigma \vdash v : \tau \dashv T_o$ *and* $\tau = \mathbf{bool}[r]$ *or* $\tau = \{a.\ \mathbf{bool}[a] \mid r'\}$ *then* $v = \mathbf{true}$ *or* $v = \mathbf{false}$*.*

Proof. By induction on the typing derivation. The following cases apply:

- **T-true:** Direct since the rules only applies if $v = \mathbf{true}$.
- **T-false:** Direct since the rules only applies if $v = \mathbf{false}$.

- **T-sub:** By inversion of the subtyping judgment on the premise of the rule either S-idx, S-unpack or S-ex applies. In all cases we conclude by using the inductive hypothesis choosing either the indexed or existential case.

□

Lemma 6.34 (Canonical Forms Ints). *If* $\Delta \mid \Gamma; T_i \mid \Sigma \vdash v : \tau \dashv T_o$ *and* $\tau = \mathbf{int}[n]$ *or* $\tau = \{a.\ \mathbf{int}[a] \mid r\}$ *then* $v = z$ *for some* $z \in \mathbb{Z}$.

Proof. By induction on the typing derivation. The following cases apply:

- **T-int:** Direct since the rules only applies if $v = z$.
- **T-sub:** By inversion of the subtyping judgment on the premise of the rule either S-idx, S-unpack or S-ex applies. In all cases we conclude by using the inductive hypothesis choosing either the indexed or existential case.

□

Lemma 6.35 (Canonical Forms Pointers).

(1) If $\Delta \mid \Gamma; T_i \mid \Sigma \vdash v : \mathbf{ptr}(\eta) \dashv T_o$ *then* $v = ptr(\ell, t)$.
(2) If $\Delta \mid \Gamma; T_i \mid \Sigma \vdash v : \&_{\mu}\tau \dashv T_o$ *then* $v = ptr(\ell, t)$.

Proof. Both by induction on the typing derivation. The following cases apply

- **Case T-ptr:** Direct by the form of the rule.
- **Case T-sub:** By inversion of the subtyping premise in the rule T-sub either S-ptr, S-shr or S-mut apply. In all cases we conclude by applying the inductive hypothesis.

□

Lemma 6.36 (Canonical Forms Vectors). *If* $\Delta \mid \Gamma; T_i \mid \Sigma \vdash v : \tau \dashv T_o$ *and if* $\tau = \mathbf{Vec}_{\tau}[m]$ *or* $\tau = \{a.\ \mathbf{Vec}_{\tau}[a] \mid r\}$ *then* $v = \mathbf{vec}_{\tau}(n, v')$.

Proof. By induction on the typing derivation The following cases apply:

- **Case T-vec:** Direct since the rule only applies if $v = \mathbf{vec}_{\tau}(m, v')$.
- **Case T-sub:** By inversion of the subtyping judgment on the premise of T-sub either S-idx, S-unpack or S-ex apply. In all cases we conclude by using the inductive hypothesis choosing either the indexed or existential case.

□

Lemma 6.37 (Canonical Forms Functions). *If* $\Delta \mid \Gamma; T_i \mid \Sigma \vdash v : \forall \overline{a : \sigma}.\ \mathbf{fn}(T; \overline{x : \tau}) \rightarrow \tau / T_{fi} \dashv T_o$, *then either*

(1) $v = \mathbf{rec}\ f[\overline{a}](\overline{x})\ :=\ e$,
(2) $v = \mathbf{Vec}_{\tau}\mathbf{::new}$,

(3) $v = \mathbf{Vec}_\tau\mathbf{::push}$*, or*

(4) $v = \mathbf{Vec}_\tau\mathbf{::index\_mut}$.

Proof. By induction on the typing derivation. The following cases apply:

- **Case T-fun:** In this case (1) holds.
- **Case T-vec-new** In this case (2) holds.
- **Case T-vec-push** In this case (3) holds.
- **Case T-vec-index-mut** In this case (4) holds.
- **Case T-sub:** By inversion of the subtyping judgment in the premise of the rule T-sub we have that only S-fun applies. We conclude by applying the inductive hypothesis.

□

6.6 Lemmata about memory events

Lemma 6.38 (Allocation event). *If*

- $\Delta \mid T \mid \Sigma \vdash \langle h, \varsigma_i \rangle$ *and*
- $\varsigma_i \xrightarrow{EAlloc(Pointer(\ell,t),FixedSize(1))} \varsigma_o$

then

$\Delta \mid T, \ell \mapsto \lightning \mid \Sigma, (\ell, t) \mapsto \mathbf{ptr}(\ell) \vdash \langle h[\ell \mapsto ☣], \varsigma_o \rangle$

Proof. Since $\varsigma_i \xrightarrow{\text{EAlloc(Pointer}(\ell,t)\text{,FixedSize(1))}} \varsigma_o$ we know by inversion of OS-dealloc that $\ell \notin \text{dom}(\varsigma_i.\text{stacks})$. Then, by definition of $\Delta \mid \mathrm{T} \mid \Sigma \vdash \langle h, \varsigma_i \rangle$, we have that $\ell \notin \text{dom}(h)$, consequently the update does not invalidate any of the other locations. We also have by T-mem that $\Delta \mid \emptyset; \emptyset \mid \Sigma \vdash ☣ : \lightning$. It remains to show that $\Delta \mid \ell \mapsto \lightning \mid \Sigma, (\ell, t) \mapsto \mathbf{ptr}(\ell) \vdash_{\mathbf{shr}} (\text{Unique}, t, \bot)$ which follows directly by the definition of well-typed stacks. □

Lemma 6.39 (Strong-Reborrow event). *If*

- $\Delta \mid T \mid \Sigma \vdash \langle h, \varsigma_i \rangle$,
- $\Sigma(\ell, t) = \mathbf{ptr}(\ell)$ *and*
- $\varsigma_i \xrightarrow{ERetag(Pointer(\ell,t),t',FixedSize(1),Ref(\mathbf{mut}),Raw)} \varsigma_o$

then

$\Delta \mid T \mid \Sigma, (\ell, t') \mapsto \mathbf{ptr}(\ell) \vdash \langle h, \varsigma_o \rangle$

Proof. Since $\varsigma_i \xrightarrow{\text{ERetag(Pointer}(\ell,t),t'\text{,FixedSize(1),Ref}(\mathbf{mut})\text{,Raw)}} \varsigma_o$ we know that $\varsigma_o.\text{stacks}(\ell)$ must contain $(\text{Unique}, t', \_)$ at the top followed by $(\text{Unique}, t, \_)$. Since $\Delta \mid \mathrm{T} \mid \Sigma \vdash \langle h, \varsigma_i \rangle$ and $\Sigma(\ell, t) = \mathbf{ptr}(\ell)$, then S-Strg must have been used to derive the prefix of the stack was well-typed, thus S-Strg also holds if we add $(\text{Unique}, t', \_)$ to the top of the stack. □

Lemma 6.40 (Weaken-Reborrow event). *If*

- $\Delta \vdash T(\ell) \preccurlyeq \tau$,
- $\Sigma(\ell, t) = \mathbf{ptr}(\ell)$,
- $\Delta \mid T \mid \Sigma \vdash \langle h, \varsigma_i \rangle$ *and*
- $\varsigma_i \xrightarrow{\mathit{ERetag}(\mathit{Pointer}(\ell,t),t',\mathit{FixedSize}(1),\mathit{Ref}(\mathbf{mut}),\mathit{Raw})} \varsigma_o$

then

$\Delta \mid T[\ell \mapsto \tau] \mid \Sigma, (\ell, t') \mapsto \&_{\mathbf{mut}}\tau \vdash \langle h, \varsigma_o \rangle$

Proof. Similar to Lemma 6.39 we know that $\varsigma_o.\textsc{stacks}(\ell)$ must contain $(\mathsf{Unique}, t', \_)$ at the top followed by $(\mathsf{Unique}, t, \_)$. We can apply Hp-tracked to show the heaplet is well-typed after the update applying S-Mut to show the stack is well typed. Let $\Sigma' = \Sigma, (\ell, t') \mapsto \&_{\mathbf{mut}}\tau$ It suffices to show $\Delta \mid \emptyset; \emptyset \mid \Sigma' \vdash h(\ell) : \tau$. Since $\ell \in \mathrm{T}$ then Hp-tracked must have been applied to prove well-typedness of the initial state, therefore we have $\Delta \mid \emptyset; \emptyset \mid \Sigma \vdash h(\ell) : \mathrm{T}(\ell)$. By Lemma 6.5 we have $\Delta \mid \emptyset; \emptyset \mid \Sigma' \vdash h(\ell) : \mathrm{T}(\ell)$ and by subsumption $\Delta \mid \emptyset; \emptyset \mid \Sigma' \vdash h(\ell) : \tau$. □

Lemma 6.41 (Mut-Reborrow event). *If*

- $\Sigma(\ell, t) = \&_{\mathbf{mut}}\tau$
- $\Delta \mid T \mid \Sigma \vdash \langle h, \varsigma_i \rangle$ *and*
- $\varsigma_i \xrightarrow{\mathit{ERetag}(\mathit{Pointer}(\ell,t),t',\mathit{FixedSize}(1),\mathit{Ref}(\mathbf{mut}),\mathit{Raw})} \varsigma_o$

then

$\Delta \mid T \mid \Sigma, (\ell, t') \mapsto \&_{\mathbf{mut}}\tau \vdash \langle h, \varsigma_o \rangle$

Proof. Since $\varsigma_i \xrightarrow{\mathrm{ERetag}(\mathrm{Pointer}(\ell,t),t',\mathrm{FixedSize}(1),\mathrm{Ref}(\mathbf{mut}),\mathrm{Raw})} \varsigma_o$ we have that $\varsigma_o.\textsc{stacks}(\ell)$ must contain $(\mathsf{Unique}, t', \_)$ at the top. We also know the prefix of the stack must be well-typed. We conclude by applying S-Mut to show the stack updated with the new item is well-typed. □

Lemma 6.42 (Shared-Reborrow event). *If*

- $\Sigma(\ell, t) = \&_{\mu}\tau$
- $\Delta \mid T \mid \Sigma \vdash \langle h, \varsigma_i \rangle$ *and*
- $\varsigma_i \xrightarrow{\mathit{ERetag}(\mathit{Pointer}(\ell,t),t',\mathit{FixedSize}(1),\mathit{Ref}(\mathbf{shr}),\mathit{Raw})} \varsigma_o$

then

$\Delta \mid T \mid \Sigma, (\ell, t') \mapsto \&_{\mathbf{shr}}\tau \vdash \langle h, \varsigma_o \rangle$

Proof. By well-typedness of the input stack either S-Mut or S-Shr was applied. In both cases we know $\Delta \vdash \tau \preccurlyeq \tau'$ for some τ'. We also know the retag transition only remove items from the stack, thus the prefix of the output stack must also be well-typed. We conclude by applying S-Shr to prove the output stack is well-typed after the update. □

Lemma 6.43 (Read event). *If*

- $\Delta \mid T \mid \Sigma \vdash \langle h, \varsigma_i \rangle$,
- $\Sigma(\ell, t) = \&_{\mu}\tau$ *and*
- $\varsigma_i \xrightarrow{EAccess(AccessRead, Pointer(\ell,t), FixedSize(1))} \varsigma_o$

then

- $\Delta \mid \emptyset; T \mid \Sigma \vdash h(\ell) : \tau$ *and*
- $\Delta \mid T \mid \Sigma \vdash \langle h, \varsigma_o \rangle$.

Proof. Since $\varsigma_i \xrightarrow{\text{EAccess(AccessRead,Pointer}(\ell,t)\text{,FixedSize(1))}} \varsigma_o$ the stack must have a granting item with tag t. Because the initial stack is well-typed then either one of S-Shr or S-Mut must have been used for the granting item. In both cases we know $\Delta \vdash \tau' \preccurlyeq \tau$ for some τ'. In both cases we also know $\Delta \mid \emptyset; \mathrm{T} \mid \Sigma \vdash h(\ell) : \tau'$ because either Hp-tracked or Hp-untracked must apply. By subsumption we know $\Delta \mid \emptyset; \mathrm{T} \mid \Sigma \vdash h(\ell) : \tau$ which is the first part of the conclusion. The second part follows from the fact that the read access transition can only change Unique permissions into Disabled, and thus we can apply S-Disabled to prove the output stack is well-typed. □

Lemma 6.44 (Strong read event). *If*

- $\ell \in dom(T)$
- $\Delta \mid T \mid \Sigma \vdash \langle h, \varsigma_i \rangle$,
- $\varsigma_i \xrightarrow{EAccess(AccessRead, Pointer(\ell,t), FixedSize(1))} \varsigma_o$

then

- $\Delta \mid \emptyset; T \mid \Sigma \vdash h(\ell) : T(\ell)$ *and*
- $\Delta \mid T \mid \Sigma \vdash \langle h, \varsigma_o \rangle$.

Proof. The first part of the conclusion follows directly by the definition of well-formed heaps since Hp-tracked must apply. The second part follows from the fact that the read access transition can only change Unique permissions into Disabled, and thus we can apply S-Disabled to prove the output stack is well-typed. □

Lemma 6.45 (Weak write event). *If*

- $\Delta \mid T \mid \Sigma \vdash \langle h, \varsigma_i \rangle$,
- $\Sigma\ (\ell, t) = \&_{\mathbf{mut}}\tau$,
- $\Delta \mid \emptyset; T \mid \Sigma \vdash v : \tau$, *and*
- $\varsigma_i \xrightarrow{EAccess(AccessWrite, Pointer(\ell,t), FixedSize(1))} \varsigma_o$.

then

$\Delta \mid T \mid \Sigma \vdash \langle h[\ell \mapsto v], \varsigma_o \rangle$

Proof. After the update, $\varsigma_o.\textsc{stacks}(\ell)$ must contain $(\text{Unique}, t, \_)$ at the top. Because the stack was well-typed before the update by inversion of S-Mut we have (1) $\Delta \vdash \tau \preccurlyeq \mathrm{T}(\ell)$. By (1) and subsumption we have $\Delta \mid \emptyset; \mathrm{T} \mid \Sigma \vdash v : \mathrm{T}(\ell)$, thus we can stil use Hp-tracked to prove the heaplet is well-typed after the update. We conclude by noting that the output stack must also be well-typed because items are only popped from it. □

Lemma 6.46 (Strong write event). *If*

- $\Delta \mid T \mid \Sigma \vdash \langle h, \varsigma_i \rangle$,
- $\Sigma\ (\ell, t) = \mathbf{ptr}(\ell)$,
- $\Delta \mid \emptyset; T \mid \Sigma \vdash v : \tau$
- $\varsigma_i \xrightarrow{EAccess(AccessWrite, Pointer(\ell,t), FixedSize(1))} \varsigma_o$ *and*

then

- $\Delta \mid T[\ell \mapsto \tau] \mid \Sigma \vdash \langle h[\ell \mapsto v], \varsigma_o \rangle$

Proof. After the transition, ς_o.stacks(ℓ) must contain $(\mathsf{Unique}, t, \_)$ at the top and because $\Sigma(\ell, t) = \mathbf{ptr}(\ell)$, the output stack only contains tags that are mapped to strong pointers in Σ. Therefore, there is nothing else to prove to show the output stack is well-typed. Finally, the heap is well-typed because we are updating T to point to the new type τ. □

Lemma 6.47 (Vec push empty). *Let*

- $h_i = h[\ell \mapsto \mathbf{vec}_\tau(0, v')]$ *and*
- $h_o = h[\ell \mapsto \mathbf{vec}_\tau(1, ptr(\ell', t'))][\ell' \mapsto v]$.

If

- $\Sigma(\ell, t) = \mathbf{ptr}(\ell)$
- $\Delta \mid \emptyset; \emptyset \mid \Sigma \vdash v : \tau$,
- $\Delta \mid T \mid \Sigma \vdash \langle h_i, \varsigma_i \rangle$, *and*
- $\varsigma_i \xrightarrow{EAccess(AccessWrite, Pointer(\ell,t), FixedSize(1))} \varsigma$,
- $\varsigma \xrightarrow{EAlloc(Pointer(\ell',t'), FixedSize(1))} \varsigma_o$,

then

$\Delta \mid T[\ell \mapsto \mathbf{Vec}_\tau[1]] \mid \Sigma, (\ell', t') \mapsto \&_{\mathbf{mut}}\tau \vdash \langle h_o, \varsigma_o \rangle$ *and*

Proof. Let $\Sigma_o = \Sigma, (\ell', t') \mapsto \&_{\mathbf{mut}}\tau$. We can apply T-vec to show $\Delta \mid \emptyset; \emptyset \mid \Sigma_o \vdash \mathbf{vec}_\tau(1, \mathrm{ptr}(\ell', t')) : \mathbf{Vec}_\tau[1] \dashv \emptyset$. Since v has type τ then the output stack will be well-typed in Σ_o. We conclude by using the same argument in Lemma 6.46 to prove we can update the type of ℓ in T. □

Lemma 6.48 (Vec push). *Let*

- $h_i = h[\ell \mapsto \mathbf{vec}_\tau(n, ptr(\ell', t'))][\ell' \mapsto_n \overline{v}]$,
- $h_o = h[\ell \mapsto \mathbf{vec}_\tau(n+1, ptr(\ell'', t''))][\ell'' \mapsto_{n+1} \overline{v} + [v]]$, *and*
- $\Sigma_o = (\Sigma_i - (\ell', \_))[(\ell'' + i, t') \mapsto \&_{\mathbf{mut}}\tau | i \in [0, n]]$.

If

- $\Delta \mid \emptyset;\emptyset \mid \Sigma_i \vdash v : \tau$,
- $\varsigma_i \xrightarrow{EAccess(AccessWrite, Pointer(\ell,t), FixedSize(1))} \varsigma$,
- $\varsigma \xrightarrow{EDealloc(Pointer(\ell',t'), FixedSize(n))} \varsigma'$,
- $\varsigma' \xrightarrow{EAlloc(Pointer(\ell'',t''), FixedSize(n+1))} \varsigma_o$,
- $\Delta \mid T \mid \Sigma_i \vdash \langle h_i, \varsigma_i \rangle$, *and*
- $\Sigma(\ell, t) = \mathbf{ptr}(\ell)$

then

$\Delta \mid T[\ell \mapsto \mathbf{Vec}_\tau[n+1]] \mid \Sigma_o \vdash \langle h_o, \varsigma_o \rangle$ *and*

PROOF. By well-typedness of the input state we know $\Delta \mid \emptyset;\emptyset \mid \Sigma_i \vdash \mathbf{vec}_\tau(n, \mathrm{ptr}(\ell', t')) : \mathbf{Vec}_\tau[n]$. Then, by inversion of T-VEC we know $\Sigma_i(\ell' + i, t') = \&_{\mathbf{mut}}\tau$ for $i \in [0, n)$ which together with well-typedness of the input state implies $\Delta \mid \emptyset;\emptyset \mid \Sigma_i \vdash h_i(\ell + i) : \tau$. Thus, since the new value v also has type τ, after the update all the relocated values have type τ. We can then apply T-VEC to prove $\mathbf{vec}_\tau(n+1, \mathrm{ptr}(\ell'', t''))$ has type $\mathbf{Vec}_\tau[n+1]$ under Σ_o, and conclude using the same argument in Lemma 6.46 to prove we can update the type of ℓ in T. □

LEMMA 6.49 (VEC INDEX MUT). *If*

- $\Sigma(\ell, t) = \&_{\mathbf{mut}}\mathbf{Vec}_\tau[n]$,
- $h(\ell) = \mathbf{vec}_\tau(n, ptr(\ell', t'))$,
- $\varsigma_i \xrightarrow{EAccess(AccessRead, Pointer(\ell,t), FixedSize(1))} \varsigma$, *and*
- $\varsigma \xrightarrow{ERetag(Pointer(\ell'+i,t'), t'', FixedSize(1), \mathbf{mut}, Raw)} \varsigma_o$
- $\Delta \mid T \mid \Sigma \vdash \langle h, \varsigma_i \rangle$

then

$\Delta \mid T \mid \Sigma, (\ell' + i, t'') \mapsto \&_{\mathbf{mut}}\tau \vdash \langle h, \varsigma_o \rangle$ *and*

PROOF. By Lemma 6.43 we have $\Delta \mid \emptyset;\emptyset \mid \Sigma \vdash \mathbf{vec}_\tau(n, \mathrm{ptr}(\ell', t')) : \mathbf{Vec}_\tau[n]$. By inversion of T-VEC we have $\Sigma(\ell' + i, t') = \&_{\mathbf{mut}}\tau$ for $i \in [0, n)$. We conclude by applying Lemma 6.41. □

REFERENCES

Ralf Jung, Hoang-Hai Dang, Jeehoon Kang, and Derek Dreyer. 2020a. Stacked Borrows: An Aliasing Model for Rust. *POPL*. https://doi.org/10.1145/3371109

Ralf Jung, Hoang-Hai Dang, Jeehoon Kang, and Derek Dreyer. 2020b. Stacked Borrows: Appendix. https://plv.mpi-sws.org/rustbelt/stacked-borrows