Linking Types: Technical Appendix:

Daniel Patterson, Andrew Wagner, Amal Ahmed

Northeastern University

July 17, 2023

$$
\begin{array}{lll}
\text{Types } \tau & := & \alpha \mid \texttt{unit} \mid \texttt{bool} \mid \texttt{int} \mid \tau \times \tau \mid \tau + \tau \mid \tau * \tau \\
& & \mid \mu\alpha.\tau \mid (\tau, \ldots, \tau) \to \tau \\
\text{Expressions } \texttt{e} & := & () \mid \texttt{true} \mid \texttt{false} \mid \texttt{if e \{e\} \{e\}} \mid \texttt{n} \mid \texttt{e = e} \\
& & \mid \texttt{e < e} \mid \texttt{e + e} \mid \texttt{x} \mid (\texttt{e}, \texttt{e}) \mid \texttt{fst e} \mid \texttt{snd e} \\
& & \mid \texttt{inl e} \mid \texttt{inr e} \mid \texttt{match e x\{e\} y\{e\}} \mid \texttt{fold e} \\
& & \mid \texttt{unfold e} \mid \texttt{fun f}(\texttt{x}_1 : \tau_1, \ldots, \texttt{x}_n : \tau_n)\{\texttt{e}\} \\
& & \mid \texttt{e}(\texttt{e}, \ldots, \texttt{e}) \mid \{\texttt{e}\}_\tau \\
\text{Imports } \texttt{I} & := & \texttt{f} : \tau \mid \texttt{f} : \tau, \texttt{I} \\
\text{Programs } \texttt{P} & := & \texttt{e} \mid \texttt{import(I) e}
\end{array}
$$

Figure 0.1: Syntax for FunLang.

0.1 A FUNCTIONAL LANGUAGE

Our core language, FunLang is a standard pure, eager, non-terminating functional language with imports (presented in Fig. 0.1 and Fig. 0.2).

$$\boxed{\vdash \mathtt{P} : \tau} \qquad \frac{\cdot;\cdot \vdash \mathtt{e} : \tau}{\vdash \mathtt{e} : \tau} \qquad \frac{\mathtt{I};\cdot \vdash \mathtt{e} : \tau}{\vdash \mathtt{import(I)\ e} : \tau}$$

$$\boxed{\mathtt{I};\Gamma \vdash \mathtt{e} : \tau} \qquad \frac{}{\mathtt{I};\Gamma \vdash () : \mathtt{unit}} \qquad \frac{}{\mathtt{I};\Gamma \vdash \mathtt{true/false} : \mathtt{bool}} \qquad \frac{}{\mathtt{I};\Gamma \vdash \mathtt{n} : \mathtt{int}}$$

$$\frac{\mathtt{I};\Gamma \vdash \mathtt{e} : \mathtt{bool} \qquad \mathtt{I};\Gamma \vdash \mathtt{e}_1 : \tau \qquad \mathtt{I};\Gamma \vdash \mathtt{e}_2 : \tau}{\mathtt{I};\Gamma \vdash \mathtt{if\ e\ \{e_1\}\ \{e_2\}} : \tau}$$

$$\frac{\mathtt{I};\Gamma \vdash \mathtt{e}_1 : \mathtt{int} \qquad \mathtt{I};\Gamma \vdash \mathtt{e}_2 : \mathtt{int}}{\mathtt{I};\Gamma \vdash \mathtt{e}_1 \ = \ \mathtt{e}_2 : \mathtt{bool}} \qquad \frac{\mathtt{I};\Gamma \vdash \mathtt{e}_1 : \mathtt{int} \qquad \mathtt{I};\Gamma \vdash \mathtt{e}_2 : \mathtt{int}}{\mathtt{I};\Gamma \vdash \mathtt{e}_1 \ < \ \mathtt{e}_2 : \mathtt{bool}}$$

$$\frac{\mathtt{I};\Gamma \vdash \mathtt{e}_1 : \mathtt{int} \qquad \mathtt{I};\Gamma \vdash \mathtt{e}_2 : \mathtt{int}}{\mathtt{I};\Gamma \vdash \mathtt{e}_1 \ + \ \mathtt{e}_2 : \mathtt{int}} \qquad \frac{\mathtt{x} : \tau \in \Gamma}{\mathtt{I};\Gamma \vdash \mathtt{x} : \tau}$$

$$\frac{\mathtt{I};\Gamma \vdash \mathtt{e}_1 : \tau_1 \qquad \mathtt{I};\Gamma \vdash \mathtt{e}_2 : \tau_2}{\mathtt{I};\Gamma \vdash (\mathtt{e}_1, \mathtt{e}_2) : \tau_1 \times \tau_2} \qquad \frac{\mathtt{I};\Gamma \vdash \mathtt{e} : \tau_1 \times \tau_2}{\mathtt{I};\Gamma \vdash \mathtt{fst/snd\ e} : \tau_1/\tau_2}$$

$$\frac{\mathtt{I};\Gamma \vdash \mathtt{e} : \tau_1/\tau_2 \qquad \vdash \tau_2/\tau_2}{\mathtt{I};\Gamma \vdash \mathtt{inl/inr\ e} : \tau_1 + \tau_2}$$

$$\frac{\mathtt{I};\Gamma \vdash \mathtt{e} : \tau_1 + \tau_1 \qquad \Gamma, x : \tau_1 \vdash \mathtt{e}_1 : \tau \qquad \Gamma, y : \tau_2 \vdash \mathtt{e}_2 : \tau}{\mathtt{I};\Gamma \vdash \mathtt{match\ e\ x\{e_1\}\ y\{e_2\}} : \tau}$$

$$\frac{\mathtt{I};\Gamma \vdash \mathtt{e} : \tau[\mu\alpha.\tau/\alpha]}{\mathtt{I};\Gamma \vdash \mathtt{fold\ e} : \mu\alpha.\tau} \qquad \frac{\mathtt{I};\Gamma \vdash \mathtt{e} : \mu\alpha.\tau}{\mathtt{I};\Gamma \vdash \mathtt{unfold\ e} : \tau[\mu\alpha.\tau/\alpha]}$$

$$\frac{\Gamma, \mathtt{f} : (\tau_1, \ldots, \tau_n) \to \tau', \mathtt{x_i} : \tau_i \vdash \mathtt{e} : \tau'}{\mathtt{I};\Gamma \vdash \mathtt{fun\ f(x_1} : \tau_1, \ldots, \mathtt{x_n} : \tau_n)\{\mathtt{e}\} : (\tau_1, \ldots, \tau_n) \to \tau'}$$

$$\frac{\mathtt{I};\Gamma \vdash \mathtt{e} : (\tau_1, \ldots, \tau_n) \to \tau' \qquad \mathtt{I};\Gamma \vdash \mathtt{e_i} : \tau_i}{\mathtt{I};\Gamma \vdash \mathtt{e}(\mathtt{e}_1, \ldots, \mathtt{e}_n) : \tau'}$$

Figure 0.2: Static semantics for FunLang.

Stack S	:=	v, . . . , v | Fail c
Error Code c	:=	TYPE | IDX | MEM | CTRL
Program P	:=	· | i; P
Value v	:=	n | thunk P | ℓ | [v, . . .]
Instruction i	:=	push v | add | less? | equal? | if0 P P | lam x.P | call | fix | idx | len | alloc | read | write | free | shift k P | reset | getlocs | noop | fail c

Figure 0.3: Syntax for StackLang

0.2 A STACK LANGUAGE

Our target language is an untyped, stack-based language called StackLang, which is derived from Kleffner (2017), which in turn derives some features from Levy (2001) (presented in Fig. 0.3 and Fig. 0.4). In the case of malformed programs that attempt to perform operations without a valid stack, we terminate with a TYPE error code via the fail instruction. We use these (and in particular, rule them out) in our logical relations to ensure only well-formed programs.

$$
\begin{array}{rcll}
\langle H \fatsemi S \fatsemi \mathsf{push\ v}; P\rangle & \to & \langle H \fatsemi S, v \fatsemi P\rangle & (S \neq \mathsf{Fail\ c}) \\
\langle H \fatsemi \mathsf{Fail\ c} \fatsemi \mathsf{push\ v}; P\rangle & \to & \langle H \fatsemi \mathsf{Fail\ c} \fatsemi \mathsf{fail\ } \textsc{Type}\rangle & \\
\langle H \fatsemi S, n_2, n_1 \fatsemi \mathsf{add}; P\rangle & \to & \langle H \fatsemi S, (n_1 + n_2) \fatsemi P\rangle & \\
\langle H \fatsemi S \fatsemi \mathsf{add}; P\rangle & \to & \langle H \fatsemi S \fatsemi \mathsf{fail\ } \textsc{Type}\rangle & (S \neq S', n_2, n_1) \\
\langle H \fatsemi S, n_2, n_1 \fatsemi \mathsf{less?}; P\rangle & \to & \langle H \fatsemi S, 0 \fatsemi P\rangle & (n_1 < n_2) \\
\langle H \fatsemi S, n_2, n_1 \fatsemi \mathsf{less?}; P\rangle & \to & \langle H \fatsemi S, 1 \fatsemi P\rangle & (n_1 \geq n_2) \\
\langle H \fatsemi S \fatsemi \mathsf{less?}; P\rangle & \to & \langle H \fatsemi S \fatsemi \mathsf{fail\ } \textsc{Type}\rangle & (S \neq S', n_2, n_1) \\
\langle H \fatsemi S, v, v \fatsemi \mathsf{equal?}; P\rangle & \to & \langle H \fatsemi S, 0 \fatsemi P\rangle & \\
\langle H \fatsemi S, v_2, v_1 \fatsemi \mathsf{equal?}; P\rangle & \to & \langle H \fatsemi S, 1 \fatsemi P\rangle & v_1 \neq v_2 \\
\langle H \fatsemi S \fatsemi \mathsf{equal?}; P\rangle & \to & \langle H \fatsemi S \fatsemi \mathsf{fail\ } \textsc{Type}\rangle & (S \neq S', v_2, v_1) \\
\langle H \fatsemi S, 0 \fatsemi \mathsf{if0\ } P_1\ P_2; P\rangle & \to & \langle H \fatsemi S \fatsemi P_1; P\rangle & \\
\langle H \fatsemi S, n \fatsemi \mathsf{if0\ } P_1\ P_2; P\rangle & \to & \langle H \fatsemi S \fatsemi P_2; P\rangle & (n \neq 0) \\
\langle H \fatsemi S \fatsemi \mathsf{if0\ } P_1\ P_2; P\rangle & \to & \langle H \fatsemi S \fatsemi \mathsf{fail\ } \textsc{Type}\rangle & (S \neq S', n) \\
\langle H \fatsemi S, v \fatsemi \mathsf{lam\ x}.P_1; P_2\rangle & \to & \langle H \fatsemi S \fatsemi [x \mapsto v]P_1; P_2\rangle & \\
\langle H \fatsemi S \fatsemi \mathsf{lam\ x}.P_1; P_2\rangle & \to & \langle H \fatsemi S \fatsemi \mathsf{fail\ } \textsc{Type}\rangle & (S \neq S', v) \\
\langle H \fatsemi S, \mathsf{thunk\ } P_1 \fatsemi \mathsf{call}; P_2\rangle & \to & \langle H \fatsemi S \fatsemi P_1; P_2\rangle & \\
\langle H \fatsemi S \fatsemi \mathsf{call}; P_2\rangle & \to & \langle H \fatsemi S \fatsemi \mathsf{fail\ } \textsc{Type}\rangle & (S \neq S', \mathsf{thunk\ } P_1) \\
\langle H \fatsemi S, \mathsf{thunk\ } P_1 \fatsemi \mathsf{fix}; P_2\rangle & \to & \langle H \fatsemi S, \mathsf{thunk\ (push\ (thunk\ } P_1), \mathsf{fix}) \fatsemi & \\
 & & P_1; P_2\rangle & \\
\langle H \fatsemi S \fatsemi \mathsf{fix}; P_2\rangle & \to & \langle H \fatsemi S \fatsemi \mathsf{fail\ } \textsc{Type}\rangle & (S \neq S', \mathsf{thunk\ } P_1) \\
\langle H \fatsemi S, [v_0, \ldots, v_{n_2}], n_1 \fatsemi \mathsf{idx}; P\rangle & \to & \langle H \fatsemi S, v_{n_1} \fatsemi P\rangle & (n_1 \in [0, n_2]) \\
\langle H \fatsemi S, [v_0, \ldots, v_{n_2}], n_1 \fatsemi \mathsf{idx}; P\rangle & \to & \langle H \fatsemi S \fatsemi \mathsf{fail\ } \textsc{Idx}\rangle & (n_1 \notin [0, n_2]) \\
\langle H \fatsemi S \fatsemi \mathsf{idx}; P\rangle & \to & \langle H \fatsemi S \fatsemi \mathsf{fail\ } \textsc{Type}\rangle & (S \neq S', [v_0, \ldots, v_{n_2}], n_1) \\
\langle H \fatsemi S, [v_0, \ldots, v_n] \fatsemi \mathsf{len}; P\rangle & \to & \langle H \fatsemi S, (n+1) \fatsemi P\rangle & \\
\langle H \fatsemi S \fatsemi \mathsf{len}; P\rangle & \to & \langle H \fatsemi S \fatsemi \mathsf{fail\ } \textsc{Type}\rangle & (S \neq S', [v_0, \ldots, v_n]) \\
\langle H \fatsemi S, v \fatsemi \mathsf{alloc}; P\rangle & \to & \langle H \uplus \{\ell \mapsto v\} \fatsemi S, \ell \fatsemi P\rangle & \\
\langle H \fatsemi \cdot \fatsemi \mathsf{alloc}; P\rangle & \to & \langle H \fatsemi \cdot \fatsemi \mathsf{fail\ } \textsc{Type}\rangle & \\
\langle H \uplus \{\ell \mapsto v\} \fatsemi S, \ell \fatsemi \mathsf{read}; P\rangle & \to & \langle H \uplus \{\ell \mapsto v\} \fatsemi S, v \fatsemi P\rangle & \\
\langle H \fatsemi S, \ell \fatsemi \mathsf{read}; P\rangle & \to & \langle H \fatsemi S \fatsemi \mathsf{fail\ } \textsc{Mem}\rangle & \ell \notin \mathrm{dom}(H) \\
\langle H \fatsemi S \fatsemi \mathsf{read}; P\rangle & \to & \langle H \fatsemi S \fatsemi \mathsf{fail\ } \textsc{Type}\rangle & (S \neq S', \ell) \\
\langle H \uplus \{\ell \mapsto \_\} \fatsemi S, \ell, v \fatsemi \mathsf{write}; P\rangle & \to & \langle H \uplus \{\ell \mapsto v\} \fatsemi S \fatsemi P\rangle & \\
\langle H \fatsemi S, \ell, v \fatsemi \mathsf{write}; P\rangle & \to & \langle H \fatsemi S \fatsemi \mathsf{fail\ } \textsc{Mem}\rangle & \ell \notin \mathrm{dom}(H) \\
\langle H \fatsemi S \fatsemi \mathsf{write}; P\rangle & \to & \langle H \fatsemi S \fatsemi \mathsf{fail\ } \textsc{Type}\rangle & (S \neq S', \ell, v) \\
\langle H \uplus \{\ell \mapsto \_\} \fatsemi S, \ell \fatsemi \mathsf{free}; P\rangle & \to & \langle H\} \fatsemi S \fatsemi P\rangle & \\
\langle H \fatsemi S, \ell \fatsemi \mathsf{free}; P\rangle & \to & \langle H \fatsemi S \fatsemi \mathsf{fail\ } \textsc{Mem}\rangle & \ell \notin \mathrm{dom}(H) \\
\langle H \fatsemi S \fatsemi \mathsf{free}; P\rangle & \to & \langle H \fatsemi S \fatsemi \mathsf{fail\ } \textsc{Type}\rangle & (S \neq S', \ell) \\
\langle H \fatsemi S \fatsemi \mathsf{shift\ k\ } P_1; P_2; & & & \\
\ldots; \mathsf{reset}; P_3\rangle & \to & \langle H \fatsemi S \fatsemi [k \mapsto \mathsf{thunk\ } P_2; \ldots]P_1; P_3\rangle & \mathsf{reset} \notin P_2; \ldots \\
\langle H \fatsemi S \fatsemi \mathsf{shift\ k\ } P_1; P_2\rangle & \to & \langle H \fatsemi S \fatsemi \mathsf{fail\ } \textsc{Ctrl}\rangle & \mathsf{reset} \notin P_2 \\
\langle H \fatsemi S \fatsemi \mathsf{reset}; P\rangle & \to & \langle H \fatsemi S \fatsemi P\rangle & \\
\langle H \fatsemi S, \mathsf{thunk\ lam\ l}.P_1, v \fatsemi & \to & \langle H \fatsemi S, \ell_1, \ldots, \ell_n \fatsemi & \\
\mathsf{getlocs}; P_2\rangle & & \mathsf{lam\ l}.P_1; \ldots; \mathsf{lam\ l}.P_1; P_2\rangle & \ell_1, \ldots, \ell_n = \mathrm{flocs}(v) \\
\langle H \fatsemi S \fatsemi \mathsf{getlocs}; P_2\rangle & \to & \langle H \fatsemi S \fatsemi \mathsf{fail\ } \textsc{Type}\rangle & S \neq S', \mathsf{thunk\ lam\ l}.P_1, \ell \\
\langle H \fatsemi S \fatsemi \mathsf{noop}; P\rangle & \to & \langle H \fatsemi S \fatsemi P\rangle & \\
\langle H \fatsemi S \fatsemi \mathsf{fail\ c}; P\rangle & \to & \langle H \fatsemi \mathsf{Fail\ c} \fatsemi \cdot\rangle &
\end{array}
$$

Figure 0.4: Operational semantics for StackLang

$\boxed{e \rightsquigarrow e^+}$

$()$	$\rightsquigarrow$	push 0
true/false	$\rightsquigarrow$	push 0/1
if e $\{e_1\}$ $\{e_2\}$	$\rightsquigarrow$	e^+; if0 $(e_1{}^+)$ $(e_2{}^+)$
n	$\rightsquigarrow$	push n
$e_1 < / = / + e_2$	$\rightsquigarrow$	$e_1{}^+$; $e_2{}^+$; less?/equal?/add
x	$\rightsquigarrow$	push x
inl e	$\rightsquigarrow$	e^+, lam x.(push $[0, x]$)
inr e	$\rightsquigarrow$	e^+; lam x.(push $[1, x]$)
match e x$\{e_1\}$ y$\{e_2\}$	$\rightsquigarrow$	e^+; DUP; push 1; idx; SWAP; push 0; idx; if0 (lam x.$e_1{}^+$) (lam y.$e_2{}^+$)
fold e	$\rightsquigarrow$	e^+
unfold e	$\rightsquigarrow$	e^+; noop
(e_1, e_2)	$\rightsquigarrow$	$e_1{}^+$; $e_2{}^+$; lam x_2.lam x_1.(push $[x_1, x_2]$)
fst/snd e	$\rightsquigarrow$	e^+; push 0/1; idx
fun f$(x_1 : \tau_1, \ldots, x_n : \tau_2)\{e\}$	$\rightsquigarrow$	push (thunk push (thunk lam f.lam x_n.... lam x_1.e^+), fix)
$e(e_1, \ldots, e_n)$	$\rightsquigarrow$	e^+; $e_1{}^+$; SWAP; $e_2{}^+$; SWAP ...; $e_n{}^+$; SWAP; call

SWAP $\triangleq$ lam x.lam y.(push x; push y)
DROP $\triangleq$ lam x.()
DUP $\triangleq$ lam x.(push x; push x)

Figure 0.5: Compiler from FunLang to StackLang

0.3 A COMPILER FOR FunLang

In Figure 0.5, we present a compiler from FunLang to StackLang, which induces the operational semantics of FunLang.

$$
\begin{array}{lcl}
\text{Core Type } \tau & := & \alpha \mid \texttt{unit} \mid \texttt{bool} \mid \texttt{int} \mid \tau \times \tau \mid \tau + \tau \\
& & \mid \mu\alpha.\tau \mid (\tau, \ldots, \tau) \to \tau \\
\text{Extended Type } \tau & := & \texttt{unit} \mid \texttt{bool} \mid \texttt{int} \mid \tau \times \tau \mid \tau + \tau \mid \mu\alpha.\tau \\
& & \mid (\tau, \ldots, \tau) \xrightarrow{\bullet} \tau \mid \textbf{ref}\ \tau
\end{array}
$$

$$
\frac{\texttt{x} : \tau \in \Gamma}{\Gamma \vdash_{\texttt{S}} \texttt{x} : \tau}
\qquad
\frac{\Gamma, \texttt{f} : (\tau_1, \ldots, \tau_n) \xrightarrow{\bullet} \tau', \overline{\texttt{x}_i : \tau_i} \vdash_{\texttt{S}} \texttt{e} : \tau'}{\Gamma \vdash_{\texttt{S}} \texttt{fun f}(\texttt{x}_1 : \tau_1, \ldots, \texttt{x}_n : \tau_n)\{\texttt{e}\} : (\tau_1, \ldots, \tau_n) \xrightarrow{\bullet} \tau'}
$$

$$
\frac{\Gamma \vdash_{\texttt{S}} \texttt{e} : (\tau_1, \ldots, \tau_n) \xrightarrow{\circ} \tau' \qquad \Gamma \vdash_{\texttt{S}} \overline{\texttt{e}_i : \tau_i}}{\Gamma \vdash_{\texttt{S}} \texttt{e}(\texttt{e}_1, \ldots, \texttt{e}_n) : \tau'}
$$

Figure 0.6: Linking types for state

$$
\frac{\texttt{I}^{+} \uplus \uparrow\Gamma \vdash_{+} \texttt{e} : \tau}{\texttt{I}; \Gamma \vdash \{\texttt{e}\}^{+}_{\downarrow\tau} : \downarrow\tau}
$$

Figure 0.7: The boundary term over an arbitrary extension, $+$

0.4 LINKING WITH STATE

With this linking types extension (defined in Fig. 0.6, Fig. 0.7, and Fig. 0.8), we can specify a more precise FFI for a mutable reference library (Fig. 0.9), which we can then `import`, using, for example, to implement a memoized fibonacci (Fig. 0.10). Since `FunLang` does not have polymorphism, note that we had to pick a concrete type τ when we import them:

$$
\begin{array}{l}
\texttt{import(}\ \texttt{alloc} : \ () \xrightarrow{\bullet} \textbf{ref}\ \tau, \\
\qquad\qquad \texttt{read} : \ (\textbf{ref}\ \tau) \xrightarrow{\bullet} \tau, \\
\qquad\qquad \texttt{write} : (\textbf{ref}\ \tau, \tau) \xrightarrow{\bullet} \texttt{unit}\ \texttt{)} \\
\quad \ldots
\end{array}
$$

$\uparrow\tau$	$\triangleq$	τ
$\uparrow\texttt{unit}$	$\triangleq$	$\texttt{unit}$
$\uparrow\texttt{bool}$	$\triangleq$	$\texttt{bool}$
$\uparrow\texttt{int}$	$\triangleq$	$\texttt{int}$
$\uparrow\tau_1 \times \tau_2$	$\triangleq$	$\uparrow\tau_1 \times \uparrow\tau_2$
$\uparrow\tau_1 + \tau_2$	$\triangleq$	$\uparrow\tau_1 + \uparrow\tau_2$
$\uparrow\mu\alpha.\tau$	$\triangleq$	$\mu\alpha.\uparrow\tau$
$\uparrow(\tau_1, \ldots, \tau_n) \to \tau'$	$\triangleq$	$(\uparrow\tau_1, \ldots, \uparrow\tau_n) \xrightarrow{\circ} \uparrow\tau'$

$\downarrow\tau$	$\triangleq$	τ
$\downarrow\texttt{unit}$	$\triangleq$	$\texttt{unit}$
$\downarrow\texttt{bool}$	$\triangleq$	$\texttt{bool}$
$\downarrow\texttt{int}$	$\triangleq$	$\texttt{int}$
$\downarrow\tau_1 \times \tau_2$	$\triangleq$	$\downarrow\tau_1 \times \downarrow\tau_2$
$\downarrow\tau_1 + \tau_2$	$\triangleq$	$\downarrow\tau_1 + \downarrow\tau_2$
$\downarrow\mu\alpha.\tau$	$\triangleq$	$\mu\alpha.\downarrow\tau$
$\downarrow(\tau_1, \ldots, \tau_n) \xrightarrow{\circ} \tau'$	$\triangleq$	$(\downarrow\tau_1, \ldots, \downarrow\tau_n) \to \downarrow\tau'$
$\downarrow\textbf{ref}\ \tau$	$\triangleq$	$\texttt{unit}$

Figure 0.8: Lift and lower functions for state extension

$$
\begin{array}{lcl}
\wr \mathbf{ref}\ \tau \int & \triangleq & \mathsf{free; push\ 0} \\
\wr (\tau_1, \ldots, \tau_n) \xrightarrow{\bullet} \tau' \int & \triangleq & \mathsf{push\ (thunk\ lam\ l.push\ l; free); getlocs} \\
\wr \tau \int & \triangleq & \cdot \quad \text{for any other } \tau
\end{array}
$$

$$
\begin{array}{lcl}
\mathsf{ALLOC} & \triangleq & \mathsf{thunk\ push\ (thunk\ lam\ falloc.lam\ f.push\ f; alloc); fix} \\
\mathsf{READ} & \triangleq & \mathsf{thunk\ push\ (thunk\ lam\ fread.lam\ r.push\ r; read); fix} \\
\mathsf{WRITE} & \triangleq & \mathsf{thunk\ push\ (thunk\ lam\ fwrite.lam\ f.lam\ r.push\ r; push\ f; write; push\ 0); fix}
\end{array}
$$

Figure 0.9: State boundary enforcement & target library code

```
import( alloc : ((int) ⇾ int) ⇾• ref ((int) ⇾ int),
        read : (ref ((int) ⇾ int)) ⇾• ((int) ⇾ int),
        write : (ref ((int) ⇾ int), ((int) ⇾ int)) ⇾• unit)
fun fastfib(y : int){
  {let mtbl = alloc(fun f(n : int){−1}) in
   fun mutfib(x : int){
     if x = 0 {0}{if x = 1{1}{
       let m = read(mtbl) in
       if m(x) = −1{
         let r = mutfib(x + −1) + mutfib(x + −2) in
         let _ = write(mtbl, fun f(n){if n = x{r}{m(x)}}) in
         r
       }{m(x)}
     }
  }(y)}S
                int
}
```

Figure 0.10: Example: fibonacci memoized with state

0.5 SOUNDNESS

0.5.1 FunLang *model*

We present the full model for core FunLang in Fig. 0.11. To distinguish the core and extension models from one another, we annotate each with an identifier; e.g., λ for core FunLang. To account for recursive types, the model is *step-indexed*, which means that every inhabitant is actually a pair of a natural number (the step index) and a term. Oftentimes, when the step index is unimportant, we refer only to the term.

We begin with the value relation, $\mathcal{V}^{\lambda}[\![\tau]\!]$. Base types are agnostic to step indices, so their interpretation should simply be consistent with the compiler. Notice that $\mathcal{V}^{\lambda}[\![\texttt{bool}]\!]$ is more liberal than the compiler, which only uses 0 and 1 for bools, but it is consistent with the StackLang eliminator if0. $\mathcal{V}^{\lambda}[\![\tau_1 \times \tau_2]\!]$ contains all two-element arrays whose first element is in $\mathcal{V}^{\lambda}[\![\tau_1]\!]$ and whose second element is in $\mathcal{V}^{\lambda}[\![\tau_2]\!]$. $\mathcal{V}^{\lambda}[\![\tau_1 + \tau_2]\!]$ contains all two-element arrays whose first element is a tag $\mathsf{n} \in \{0, 1\}$ and whose second element is in $\mathcal{V}^{\lambda}[\![\tau_{\mathsf{n}+1}]\!]$. $\mathcal{V}^{\lambda}[\![\mu\alpha.\tau]\!]$ motivates the use of step indices: naturally, any of its values should also be in the interpretation of the unfolding, $\mathcal{V}^{\lambda}[\![\tau[\mu\alpha.\tau/\alpha]]\!]$, but because this is a potentially larger type, a model defined inductively over types alone would not be well-founded. Thus, we decrement the step index before unfolding the type. Note that $\mathcal{V}^{\lambda}[\![(\tilde{\tau}) \to \tau']\!]$ contains all thunks that map well-typed inputs to well-typed outputs. Here, we additionally consider multiple arguments and recursion when considering inputs, which we only draw from smaller step indices.

The expression relation, $\mathcal{E}^{\lambda}[\![\tau]\!]$, contains pairs (k, P) of step indices and StackLang *computations*. Note we run it for fewer than k steps. P behaves like a τ if, given an *arbitrary* heap and stack, it either (i) runs too long; or (ii) halts with an error that our notion of soundness accepts; or (iii) terminates at a value in $\mathcal{V}^{\lambda}[\![\tau]\!]$ at the top of its stack. Since $\mathcal{E}^{\lambda}[\![\tau]\!]$ only contains closed computations, we also interpret contexts in $\mathcal{G}^{\lambda}[\![\Gamma]\!]$, which contain all closing substitutions γ that map the bindings $\mathtt{x} : \tau \in \Gamma$ to well-typed values in $\mathcal{V}^{\lambda}[\![\tau]\!]$.

0.5.2 *State extension model*

In Fig. 0.12, we present auxiliary definitions used in the models for the state and exceptions extensions. Like standard operational models for mutable state (Ahmed, 2004), our logical relations use a Kripke world W that is made up of a step index k and heap typing Ψ. Heap typings map locations to type interpretations, drawn from the set of valid type interpretations Typ, or to the sentinal value †, which indicates that a location has been

$$
\begin{array}{lcl}
\mathcal{V}^{\lambda}[\![\texttt{unit}]\!] & = & \{(k, 0)\} \\
\mathcal{V}^{\lambda}[\![\texttt{bool}]\!] & = & \{(k, \mathsf{n})\} \\
\mathcal{V}^{\lambda}[\![\texttt{int}]\!] & = & \{(k, \mathsf{n})\} \\
\mathcal{V}^{\lambda}[\![\tau_1 \times \tau_2]\!] & = & \{(k, [\mathsf{v}_1, \mathsf{v}_2]) \mid (k, \mathsf{v}_1) \in \mathcal{V}^{\lambda}[\![\tau_1]\!] \wedge (k, \mathsf{v}_2) \in \mathcal{V}^{\lambda}[\![\tau_2]\!]\} \\
\mathcal{V}^{\lambda}[\![\tau_1 + \tau_2]\!] & = & \{(k, [0, \mathsf{v}]) \mid (k, \mathsf{v}) \in \mathcal{V}^{\lambda}[\![\tau_1]\!]\} \\
 & & \cup\ \{(k, [1, \mathsf{v}]) \mid (k, \mathsf{v}) \in \mathcal{V}^{\lambda}[\![\tau_2]\!]\} \\
\mathcal{V}^{\lambda}[\![\mu\alpha.\tau]\!] & = & \{(k, \mathsf{v}) \mid \forall j < k.\ (j, \mathsf{v}) \in \mathcal{V}^{\lambda}[\![\tau[\mu\alpha.\tau/\alpha]]\!]\} \\
\mathcal{V}^{\lambda}[\![(\tau_1, \ldots, \tau_{\mathsf{n}}) \to \tau']\!] & = & \{(k, \mathsf{thunk\ push\ (thunk\ lam\ f.lam\ x_n.\ldots\ lam\ x_1.P; fix)} \mid \\
 & & \forall \mathsf{v_i}\ k' < k. \wedge\ (k', \mathsf{v_i}) \in \mathcal{V}^{\lambda}[\![\tau_{\mathsf{i}}]\!] \implies \\
 & & (k', [\mathsf{x_1} \mapsto \mathsf{v_1}, \ldots, \mathsf{x_n} \mapsto \mathsf{v_n}, \\
 & & \quad \mathsf{f} \mapsto (\mathsf{thunk\ push\ (thunk\ lam\ f.lam\ x_n.\ldots} \\
 & & \quad\quad \mathsf{lam\ x_1.P); fix)]P}) \in \mathcal{E}^{\lambda}[\![\tau']\!]\}
\end{array}
$$

$$
\begin{array}{l}
\mathcal{E}^{\lambda}[\![\tau]\!] = \{(k, P) \mid \forall \mathsf{H}, \mathsf{H}', \mathsf{S}, \mathsf{S}', j < k.\ \langle \mathsf{H} \fatsemi \mathsf{S} \fatsemi P \rangle \xrightarrow{j} \langle \mathsf{H}' \fatsemi \mathsf{S}' \fatsemi \cdot \rangle \\
\quad \implies (\mathsf{S}' = \mathsf{Fail\ c} \wedge \mathsf{c} \in \textsc{OkErr})\ \vee\ \exists \mathsf{v}.\ \Big(\mathsf{S}' = \mathsf{S}, \mathsf{v} \wedge (k - j, \mathsf{v}) \in \mathcal{V}^{\lambda}[\![\tau]\!]\Big)\} \\
\quad \text{where } \textsc{OkErr} \triangleq \{\textsc{Mem}\}
\end{array}
$$

$$
\begin{array}{lcl}
\mathcal{G}^{\lambda}[\![\cdot]\!] & = & \{(k, \cdot)\} \\
\mathcal{G}^{\lambda}[\![\Gamma, x : \tau]\!] & = & \{(k, \gamma[\mathsf{x} \mapsto \mathsf{v}]) \mid (k, \mathsf{v}) \in \mathcal{V}^{\lambda}[\![\tau]\!] \wedge (k, \gamma) \in \mathcal{G}^{\lambda}[\![\Gamma]\!]\}
\end{array}
$$

Figure 0.11: FunLang logical relation

$$
\begin{aligned}
\varphi &::= \{\ell, \ldots\} \\
\Psi \cap \varphi &\triangleq \{\ell \mapsto R \mid \ell \in (\mathrm{dom}(\Psi) \cap \varphi) \wedge R = \Psi(\ell)\} \\
AtomVal_n &\triangleq \{(W, \varphi, \mathsf{v}) \mid W \in World_n\} \\
HeapTy_n &\triangleq \{\Psi \mid \forall \ell \in \mathrm{dom}(\Psi).\ \Psi(\ell) = \dagger \vee \Psi(\ell) \in Typ_n\} \\
World_n &\triangleq \{(k, \Psi) \mid k < n \wedge \Psi \in HeapTy_k\} \\
Typ_n &\triangleq \{R \in 2^{AtomVal_n} \mid \forall (W, \varphi, \mathsf{v}) \in R.\ \forall W'.\ W \sqsubseteq W' \implies (W', \varphi, \mathsf{v}) \in R\} \\
World &\triangleq \bigcup_n World_n \\
Typ &\triangleq \bigcup_n Typ_n \\
\lfloor R \rfloor_j &\triangleq \{(W, \varphi, \mathsf{v}) \mid (W, \varphi, \mathsf{v}) \in R \wedge W.k < j\} \\
\lfloor \Psi \rfloor_j &\triangleq \{\ell \mapsto \lfloor R \rfloor_j \mid \ell \mapsto R \in \Psi\} \\
(k, \Psi) \sqsubseteq (j, \Psi') &\triangleq j \leq k \wedge\ \forall \ell \in \mathrm{dom}(\Psi).(\lfloor \Psi(\ell) \rfloor_j = \lfloor \Psi'(\ell) \rfloor_j \vee \Psi'(\ell) = \dagger) \\
W_1 \sqsubset W_2 &\triangleq W_1.k > W_2.k \wedge W_1 \sqsubseteq W_2 \\
\triangleright(k, \Psi) &\triangleq (k - 1, \lfloor \Psi \rfloor_{k-1}) \\
\mathsf{H} :_\varphi W &\triangleq (\forall \ell \mapsto R \in (W.\Psi \cap \varphi).\ (\triangleright W, \mathsf{H}(\ell)) \in R)
\end{aligned}
$$

Figure 0.12: State & exception extension logical relation: preliminary definitions

freed. Our type interpretations consist of tuples (W, φ, v) of worlds W, sets of *relevant* locations φ, and target values v. Rather than using the global location information in the heap typing, we locally track the relevant locations of v (i.e., its free locations) so that we can decide whether it is extensionally pure ($\varphi = \emptyset$). A similar pattern of tracking locations has been used in logical relations for linear state (Ahmed et al., 2007), though our bookkeeping is slightly different.

We define the usual restrictions on relations and heap typings to step indices, and use these to define a later ($\triangleright$) operator. Next, we define how worlds can evolve with the $\sqsubseteq$ operator: future worlds can have a lower step index, and entries in the heap typing must either be preserved (up to lower step indices) or marked as dead.

Lastly, we characterize when a heap H, which is a mapping from locations to values, *satisfies* a world W under a relevant location set φ: for any location in the heap that is relevant and alive, its contents must be in the relation specified by the heap typing in the next world (to avoid circularity and reflect that it takes a step to retrieve a value from the heap).

Next, we define the logical relations for our state extension in Fig. 0.13. We identify relations in this model with the superscript **S**. On base values, the relation is similar to the relation for `FunLang`, though it now has to

$$
\begin{array}{lcl}
\mathcal{V}^{\mathbf{S}}[\![\mathtt{unit}]\!] & = & \{(W,\emptyset,0)\} \\
\mathcal{V}^{\mathbf{S}}[\![\mathtt{bool}]\!] & = & \{(W,\emptyset,\mathsf{n})\} \\
\mathcal{V}^{\mathbf{S}}[\![\mathtt{int}]\!] & = & \{(W,\emptyset,\mathsf{n})\} \\
\mathcal{V}^{\mathbf{S}}[\![\tau_1 \times \tau_2]\!] & = & \{(W,\varphi,[\mathsf{v}_1,\mathsf{v}_2]) \mid \varphi \subset \mathrm{dom}(W.\Psi) \wedge \varphi_1 \cup \varphi_2 = \varphi \wedge \\
 & & \quad (W,\varphi_1,\mathsf{v}_1) \in \mathcal{V}^{\mathbf{S}}[\![\tau_1]\!] \wedge (W,\varphi_2,\mathsf{v}_2) \in \mathcal{V}^{\mathbf{S}}[\![\tau_2]\!]\} \\
\mathcal{V}^{\mathbf{S}}[\![\tau_1 + \tau_2]\!] & = & \{(W,\varphi,[0,\mathsf{v}]) \mid \varphi \subset \mathrm{dom}(W.\Psi) \wedge (W,\varphi,\mathsf{v}) \in \mathcal{V}^{\mathbf{S}}[\![\tau_1]\!]\} \\
 & & \cup\ \{(W,\varphi,[1,\mathsf{v}]) \mid \varphi \subset \mathrm{dom}(W.\Psi) \wedge (W,\varphi,\mathsf{v}) \in \mathcal{V}^{\mathbf{S}}[\![\tau_2]\!]\} \\
\mathcal{V}^{\mathbf{S}}[\![\mu\alpha.\tau]\!] & = & \{(W,\varphi,\mathsf{v}) \mid (W,\varphi,\mathsf{v}) \in \triangleright \mathcal{V}^{\mathbf{S}}[\![\tau[\mu\alpha.\tau/\alpha]]\!]\} \\
\mathcal{V}^{\mathbf{S}}[\![\mathbf{ref}\ \tau]\!] & = & \{(W,\{\ell\},\ell) \mid W.\Psi(\ell) = \lfloor \mathcal{V}^{\mathbf{S}}[\![\tau]\!] \rfloor_{W.k} \mid \dagger\} \\
\mathcal{V}^{\mathbf{S}}[\![(\tau_1,\ldots,\tau_n) \overset{\circ}{\to} \tau']\!] & = & \{(W,\emptyset,\mathsf{thunk\ push\ (thunk\ lam\ f.lam\ x_n.\ldots\ lam\ x_1.P);fix}) \mid \\
 & & \quad \forall \mathsf{v_i}\ \varphi_i\ W' \sqsupset W.\ \varphi_i \subset \mathrm{dom}(W'.\Psi) \wedge (W',\varphi_i,\mathsf{v_i}) \in \mathcal{V}^{\mathbf{S}}[\![\tau_i]\!] \\
 & & \quad \Longrightarrow\ (W', \bigcup_i \varphi_i, [\mathsf{x}_1 \mapsto \mathsf{v}_1,\ldots,\mathsf{x}_n \mapsto \mathsf{v}_n, \\
 & & \qquad \mathsf{f} \mapsto (\mathsf{thunk\ push\ (thunk\ lam\ f.lam\ x_n.} \\
 & & \qquad\quad \ldots\ \mathsf{lam\ x_1.P);fix)]P}) \in \mathcal{E}^{\mathbf{S}}[\![\tau']\!]\} \\
\mathcal{V}^{\mathbf{S}}[\![(\tau_1,\ldots,\tau_n) \overset{\bullet}{\to} \tau']\!] & = & \{(W,\varphi,\mathsf{thunk\ push\ (thunk\ lam\ f.lam\ x_n.\ldots\ lam\ x_1.P);fix}) \mid \\
 & & \quad \varphi \subset \mathrm{dom}(W.\Psi) \wedge \forall \mathsf{v_i}\ \varphi_i\ W' \sqsupset W.\varphi_i \subset \mathrm{dom}(W'.\Psi) \\
 & & \quad \wedge (W',\varphi_i,\mathsf{v_i}) \in \mathcal{V}^{\mathbf{S}}[\![\tau_i]\!] \\
 & & \quad \Longrightarrow\ (W', \varphi \cup \bigcup_i \varphi_i, [\mathsf{x}_1 \mapsto \mathsf{v}_1,\ldots,\mathsf{x}_n \mapsto \mathsf{v}_n, \\
 & & \qquad \mathsf{f} \mapsto (\mathsf{thunk\ push\ (thunk\ lam\ f.lam\ x_n.} \\
 & & \qquad\quad \ldots\ \mathsf{lam\ x_1.P);fix)]P}) \in \mathcal{E}^{\mathbf{S}}[\![\tau']\!]\}
\end{array}
$$

$$
\begin{array}{l}
\mathcal{E}^{\mathbf{S}}[\![\tau]\!] = \{(W,\varphi,P) \mid \forall \mathsf{H}:_{\varphi} W,\mathsf{S},\mathsf{H}',\mathsf{S}',j < W.k.\ \langle \mathsf{H} \fatsemi \mathsf{S} \fatsemi P\rangle \xrightarrow{*} j\langle \mathsf{H}' \fatsemi \mathsf{S}' \fatsemi \cdot\rangle \\
\qquad \Longrightarrow\ (\mathsf{S}' = \mathsf{Fail\ c} \wedge \mathsf{c} \in \textsc{OkErr})\ \vee\ \exists \mathsf{v}, W' \sqsupseteq W. \\
\qquad\quad (\mathsf{S}' = \mathsf{S},\mathsf{v} \wedge \mathsf{H}' :_{\varphi' \cup \varphi} W' \wedge (W',\varphi',\mathsf{v}) \in \mathcal{V}^{\mathbf{S}}[\![\tau]\!])\}
\end{array}
$$

where $\textsc{OkErr} \triangleq \{\textsc{Mem}\}$

$$
\begin{array}{lcl}
\mathcal{G}^{\mathbf{S}}[\![\cdot]\!] & = & \{(W,\emptyset,\cdot) \mid W \in World\} \\
\mathcal{G}^{\mathbf{S}}[\![\Gamma, \mathbf{x} : \tau]\!] & = & \{(W,\varphi_1 \cup \varphi_2, \gamma[\mathsf{x} \mapsto \mathsf{v}]) \mid \varphi_i \subset \mathrm{dom}(W.\Psi) \\
 & & \quad \wedge (W,\varphi_1,\mathsf{v}) \in \mathcal{V}^{\mathbf{S}}[\![\tau]\!] \wedge (W,\varphi_2,\gamma) \in \mathcal{G}^{\mathbf{S}}[\![\Gamma]\!]\}
\end{array}
$$

$$
\begin{array}{l}
[\![\mathtt{I};\Gamma \vdash \mathsf{P} : \tau]\!] \equiv \\
\forall k\,\gamma.\ \forall((k,\emptyset),\emptyset,\gamma^{\mathtt{I}^{\mathbf{S}}}) \in \mathcal{G}^{\mathbf{S}}[\![\mathtt{I}^{\mathbf{S}}]\!].\ \forall((k,\emptyset),\emptyset,\gamma^{\mathtt{I}^{\mathbf{X}}}) \in \mathcal{G}^{\mathbf{X}}[\![\mathtt{I}^{\mathbf{X}}]\!].\ (k,\gamma) \in \mathcal{G}^{\lambda}[\![\Gamma]\!]\ \Longrightarrow \\
\quad (k,\gamma^{\mathtt{I}^{\mathbf{X}}}(\gamma^{\mathtt{I}^{\mathbf{S}}}(\gamma(\mathsf{P})))) \in \mathcal{E}^{\lambda}[\![\tau]\!]
\end{array}
$$

Figure 0.13: State extension logical relation: main definition

include an arbitrary world instead of a plain step index, and also an empty relevant location set (since base values cannot close over locations). For pairs $\tau_1 \times \tau_2$, we appeal to the value relation on its component types, but we must also account for relevant locations: the relevant locations for a pair is the union of the relevant locations for its components.

This is a key difference from a linear model, which would insist on a *disjoint* union of the locations. Sums and recursive types are analogous to the relation to FunLang. For reference types $\mathbf{ref}\ \tau$, we appeal to the relation stored in the heap typing, as usual, but we also require that the location in question is the only relevant location.

We divide functions into two cases: ones that hold locations and ones that are (extensionally) pure. Functions of the latter type, $\xrightarrow{\circ}$, have no relevant locations of their own. Still, arguments passed to such functions might themselves have relevant locations, so their union is relevant to the *application*, a computation in the expression relation. Importantly, if the return type is pure, then locations relevant to the arguments (and the application) cannot be relevant to the result. Functions of the stateful type, $\xrightarrow{\bullet}$, are interpreted similarly, but they *can* have relevant locations of their own, so they are incorporated into the union of locations relevant to the application.

The expression relation is similar to that of FunLang, but we now have constraints on what the target heap can look like. In particular, our initial heap H must satisfy the world W under the relevant location set φ, and there must be a final world $W' \sqsupseteq W$ that the final heap satisfies. We require that locations φ relevant to the term and φ' relevant to the final value both be accounted for: these locations must either be freed or have their types preserved; they cannot be changed to a different type. Essentially, this means that everything we started with must be accounted for, and anything that is relevant to the value must be in the heap at the correct type.

As before, we have an environment relation $\mathcal{G}^{\mathbf{S}}[\![\mathbf{\Gamma}]\!]$ that we use to describe closing substitutions that satisfy an environment $\mathbf{\Gamma}$. The substitutions now have a relevant set of locations, which is the union of all the locations relevant to all the values in the substitution.

0.5.3 *Proving ↑ sound*

We need to prove:

Lemma 0.5.1 (lift **S**)**.** $\forall W\ \mathsf{v}.\ (W, \emptyset, \mathsf{v}) \in \mathcal{V}^{\mathbf{S}}[\![\uparrow\tau]\!] \iff (W.k, \mathsf{v}) \in \mathcal{V}^{\lambda}[\![\tau]\!]$

Proof. We note, first, that by inspection of the logical relation, all cases of $\mathcal{V}^{\mathbf{S}}[\![\tau]\!]$ for $\tau = \uparrow\tau$, $\varphi = \emptyset$. That is obviously critically important, as we wouldn't otherwise be able to account for such locations when moving to the relation for `FunLang`, but it is also part of the design of the type system and the functions ↑. This justifies the use of $\emptyset$ in the lemma statements. The proof itself then follows via induction over the step index and structure of the type, since the subset of the state relation that we are considering maps directly to the `FunLang` relation, by design:

Case `unit`/`bool`/`int`. In this case, the values are trivially in the relation, by definition.

Case $\tau_1 \times \tau_2 / \tau_1 + \tau_2$. These follow straightforwardly by appealing to the inductive hypothesis.

Case $\mu\alpha.\tau$. In this case, we can appeal to our inductive hypothesis at a smaller k (as our type may have gotten larger).

Case $(\tau_1, \ldots, \tau_n) \to \tau'$. This follows by application of the induction hypothesis.

□

0.5.4 *Proving ⌊τ⌋ satisfies ↓*

First, we prove two lemmas:

Lemma 0.5.2 (wrap closed **S**)**.** $\forall\tau.\ \mathit{fvars}(\lfloor\tau\rfloor) = \emptyset$

Proof. This follows by simple inspection of the definition. □

Lemma 0.5.3 (encapsulation **S**)**.** $\forall W\ \varphi\ \mathsf{v}\ \tau.\ (W, \varphi, \mathsf{v}) \in \mathcal{V}^{\mathbf{S}}[\![\tau]\!] \implies (W, \varphi, \mathsf{push}\ \mathsf{v}; \lfloor\tau\rfloor) \in \mathcal{E}^{\mathbf{S}}[\![\uparrow\downarrow\tau]\!]$

Proof. We proceed by case analysis on τ, handling the majority of the cases for which $\lfloor\tau\rfloor$ is empty first. In those cases, which by inspection, $\uparrow\downarrow\tau = \tau$, the proof reduces to:

$$\forall W\ \varphi\ \mathsf{v}\ \tau.\ (W, \varphi, \mathsf{v}) \in \mathcal{V}^{\mathbf{S}}[\![\tau]\!] \implies (W, \varphi, \mathsf{push}\ \mathsf{v}) \in \mathcal{E}^{\mathbf{S}}[\![\tau]\!]$$

This follows easily: we choose an arbitrary stack and a heap that satisfies W and φ, we take a single step (if no budget, in relation trivially), and result

in terminated program with stack with v on top. As needed, $(W, \varphi, \mathsf{v}) \in \mathcal{V}^{\mathbf{S}}[\![\tau]\!]$, so we are done. Now we handle the other two cases:

Case $\mathbf{ref}\ \tau$. Our obligation is to show:

$$\forall W\ \varphi\ \mathsf{v}\ \tau.\ (W, \varphi, \mathsf{v}) \in \mathcal{V}^{\mathbf{S}}[\![\mathbf{ref}\ \tau]\!] \implies (W, \varphi, \mathsf{push\ v; free; push\ 0}) \in \mathcal{E}^{\mathbf{S}}[\![\mathtt{unit}]\!]$$

By inspection of $\mathcal{V}^{\mathbf{S}}[\![\mathbf{ref}\ \tau]\!]$, we know for some ℓ, $\varphi = \{\ell\}$, $\mathsf{v} = \ell$, and $W.\Psi(\ell) = \lfloor \mathcal{V}^{\mathbf{S}}[\![\tau]\!] \rfloor_{W.k}$. This means when we choose a heap H to run with in $\mathcal{E}^{\mathbf{S}}[\![\mathtt{unit}]\!]$, we know it will have ℓ bound to some value in $\triangleright \lfloor \mathcal{V}^{\mathbf{S}}[\![\tau]\!] \rfloor_{W.k}$, though as we will see, the actual value does not matter. We will then take three steps:

$$\begin{array}{l}\langle \mathsf{H} \fatsemi \mathsf{S} \fatsemi \mathsf{push}\ \ell; \mathsf{free}; \mathsf{push}\ 0 \rangle \to \\ \langle \mathsf{H} \fatsemi \mathsf{S}, \ell \fatsemi \mathsf{free}; \mathsf{push}\ 0 \rangle \to \\ \langle \mathsf{H} \setminus \ell \fatsemi \mathsf{S} \fatsemi \mathsf{push}\ 0 \rangle \to \langle \mathsf{H} \setminus \ell \fatsemi \mathsf{S}, 0 \fatsemi \cdot \rangle\end{array}$$

Now we choose W' to be W, but with ℓ updated to be marked as dead, and choose $\varphi' = \emptyset$. This means $(\mathsf{H} \setminus \ell) :_{\varphi} W'$ (since the dead binding in the world is ignored), and by definition, $(W', \emptyset, 0) \in \mathcal{V}^{\mathbf{S}}[\![\mathtt{unit}]\!]$, so we are done with this case.

Case $(\tau_{\mathbf{1}}, \ldots, \tau_{\mathbf{n}}) \xrightarrow{\bullet} \tau'$. Our obligation is to show:

$$\begin{array}{l}\forall W\ \varphi\ \mathsf{v}\ \tau_{\mathbf{i}}\ \tau'.\ (W, \varphi, \mathsf{v}) \in \mathcal{V}^{\mathbf{S}}[\![(\tau_{\mathbf{1}}, \ldots, \tau_{\mathbf{n}}) \xrightarrow{\bullet} \tau']\!] \implies \\ (W, \varphi, \mathsf{push\ v; lam\ x.(push\ x; push\ x);} \\ \mathsf{push\ (thunk\ lam\ l.push\ l; free); getlocs}) \in \mathcal{E}^{\mathbf{S}}[\![(\tau_{\mathbf{1}}, \ldots, \tau_{\mathbf{n}}) \xrightarrow{\circ} \tau']\!]\end{array}$$

Once we pick an arbitrary stack S and a heap $\mathsf{H} :_{\varphi} W$, we take the following steps:

$$\begin{array}{l}\langle \mathsf{H} \fatsemi \mathsf{S} \fatsemi \mathsf{push\ v; lam\ x.(push\ x; push\ x); push\ (thunk\ lam\ l.push\ l; free); getlocs} \rangle \to \\ \langle \mathsf{H} \fatsemi \mathsf{S}, \mathsf{v} \fatsemi \mathsf{lam\ x.(push\ x; push\ x); push\ (thunk\ lam\ l.push\ l; free); getlocs} \rangle \xrightarrow{3} \\ \langle \mathsf{H} \fatsemi \mathsf{S}, \mathsf{v}, \mathsf{v} \fatsemi \mathsf{push\ (thunk\ lam\ l.push\ l; free); getlocs} \rangle \to \\ \langle \mathsf{H} \fatsemi \mathsf{S}, \mathsf{v}, \mathsf{v}, (\mathsf{thunk\ lam\ l.push\ l; free}) \fatsemi \mathsf{getlocs} \rangle\end{array}$$

Now, we know that getlocs will run the thunk on top of the stack once for every free location one position down the stack, which means everything reachable from our function value. Assume those locations are $\ell_1, \ldots, \ell_k$. Then we step as follows:

$$\langle \mathsf{H} \fatsemi \mathsf{S}, \mathsf{v}, \mathsf{v}, (\mathsf{thunk\ lam\ l.push\ l; free}) \fatsemi \mathsf{getlocs} \rangle \overset{3k+1}{\to} \langle \mathsf{H} \setminus \{\ell_1, \ldots, \ell_k\} \fatsemi \mathsf{S}, \mathsf{v} \fatsemi \cdot \rangle$$

Now that we have terminated, we have to fulfill the obligations of $\mathcal{E}^{\mathbf{S}}[\![(\tau_{\mathbf{1}}, \ldots, \tau_{\mathbf{n}}) \overset{\circ}{\to} \tau']\!]$. By inspection of $\mathcal{V}^{\mathbf{S}}[\![(\tau_{\mathbf{1}}, \ldots, \tau_{\mathbf{n}}) \overset{\bullet}{\to} \tau']\!]$, we know that v has form $(\mathsf{thunk\ push\ (thunk\ lam\ f.lam\ x_n. \ldots\ lam\ x_1.P); fix})$. We choose $\varphi' = \emptyset$, and W' such that every location in φ has been marked dead. By invariant of the relation, $\varphi = \{\ell_1, \ldots, \ell_k\}$. Our heap satisfies the world, by construction, and everything that should be dead is, so the only thing that remains is to show that

$$(W', \emptyset, (\mathsf{thunk\ push\ (thunk\ lam\ f.lam\ x_n. \ldots\ lam\ x_1.P); fix})) \in \mathcal{V}^{\mathbf{S}}[\![(\tau_{\mathbf{1}}, \ldots, \tau_{\mathbf{n}}) \overset{\circ}{\to} \tau']\!]$$

This follows from our hypothesis on v, once we substitute our empty relevant location set in.

□

Now we proceed to the main lemma:

Lemma 0.5.4 (boundary $\mathbf{S}$)**.** $[\![\mathtt{I}^{\mathbf{S}} \uplus \uparrow\Gamma \vdash_{\mathbf{S}} \mathsf{P} : \tau]\!] \implies [\![\mathtt{I}; \Gamma \vdash \mathsf{P}; \lbag \tau \rbag : \downarrow\tau]\!]$

Unlike soundness for lift, this is non-trivial. To start with, the intuitive statement that, for $(W, \varphi, \mathsf{P}) \in \mathcal{E}^{\mathbf{S}}[\![\tau]\!]$, show $(W.k, \mathsf{P}; \lbag \tau \rbag) \in \mathcal{E}^{\lambda}[\![\downarrow\tau]\!]$, isn't provable (or true): the problem is that P is a term which may involve locations in φ, and the relation $\mathcal{E}^{\lambda}[\![\downarrow\tau]\!]$ for `FunLang` cannot reason about such state. Indeed, that relation specifically says you choose an arbitrary heap to run under, which clearly would get stuck if P tried to access a particular location. But, of course, $\downarrow\tau$ is a type from `FunLang`, so how do we prove this? At a high-level, this relies on both soundness of lift and the lemma proved above. The detailed proof follows.

Proof. Expanding the goal, we see we need to show:

$$\forall k\, \gamma.\ \forall((k, \emptyset), \emptyset, \gamma^{\mathtt{I}^{\mathbf{S}}}) \in \mathcal{G}^{\mathbf{S}}[\![\mathtt{I}^{\mathbf{S}}]\!].\ \forall((k, \emptyset), \emptyset, \gamma^{\mathtt{I}^{\mathbf{X}}}) \in \mathcal{G}^{\mathbf{X}}[\![\mathtt{I}^{\mathbf{X}}]\!].\ (k, \gamma) \in \mathcal{G}^{\lambda}[\![\Gamma]\!] \implies (k, \gamma^{\mathtt{I}^{\mathbf{X}}}(\gamma^{\mathtt{I}^{\mathbf{S}}}(\gamma(\mathsf{P}; \lbag \tau \rbag)))) \in \mathcal{E}^{\lambda}[\![\downarrow\tau]\!]$$

From Lemma 0.5.2, we know $\lbag \tau \rbag$ is closed, so we can push the substitutions in to just over P. Further, from the hypothesis, we know that P has no free variables from $\mathtt{I}^{\mathbf{X}}$, so we can eliminated that substitution.

The hypothesis that we are working with says:

$$\forall W\, \varphi\, \gamma'\, (W, \varphi, \gamma') \in \mathcal{G}^{\mathbf{S}}[\![\mathtt{I}^{\mathbf{S}} \uplus \uparrow \Gamma]\!] \wedge \varphi = \mathrm{flocs}(\gamma(P)) \implies (k, \varphi, \gamma'(\mathsf{P})) \in \mathcal{E}^{\mathbf{S}}[\![\tau]\!]$$

To instantiate the hypothesis, we need an environment γ' that satisfies $\mathcal{G}^{\mathbf{S}}[\![\mathtt{I}^{\mathbf{S}} \uplus \uparrow \Gamma]\!]$. We argue that it is exactly γ composed with $\gamma^{\mathtt{I}^{\mathbf{S}}}$: we know they are disjoint, and we know the former can be lifted into the latter via Lemma 0.5.1. This means, in particular, that φ is $\emptyset$.

Since we have no relevant locations, any heap will satisfy the expression relation: in particular, the arbitrary H that we have to consider for our obligation, and we can similarly use the arbitrary stack S. This means that we our hypothesis tells us that:

$$\langle \mathsf{H} \fatsemi \mathsf{S} \fatsemi (\gamma^{\mathtt{I}^{\mathbf{S}}}(\gamma(\mathsf{P}))) \rangle \xrightarrow{*} \langle \mathsf{H}' \fatsemi \mathsf{S}' \fatsemi \cdot \rangle$$

Unless we run beyond our step budget, in which case we are trivially in the relation. Similarly, if we run to $\mathsf{Fail}\ \mathsf{c}$, we are also in our relation. Otherwise, we know that $\mathsf{S}' = \mathsf{S}, \mathsf{v}$ and, for a future world $W' \sqsubseteq W$ that H' satisfies with the relevant locations φ', $(W', \varphi', \mathsf{v}) \in \mathcal{V}^{\mathbf{S}}[\![\tau]\!]$.

Now, what we want to show is that this value is "contained" by the code in $\wr\tau\int$ to behave like $\downarrow\tau$. But, clearly we can't show that using the $\mathcal{E}^{\lambda}[\![\tau]\!]$ logical relation, as the value still can have locations it is closing over, etc. So, we proceed by two steps. First, we appeal to Lemma 0.5.3

This will tell us that we can evaluate the whole program at question further, to get to a point with a world $W'' \sqsubseteq W'$, φ'', $\mathsf{H}'' :_{\varphi'' \cup \varphi'} W''$ and $(W'', \varphi'', \mathsf{v}') \in \mathcal{V}^{\mathbf{S}}[\![\uparrow\downarrow\tau]\!]$:

$$\begin{array}{l} \langle \mathsf{H} \fatsemi \mathsf{S} \fatsemi (\gamma^{\mathtt{I}^{\mathbf{S}}}(\gamma(\mathsf{P})); \wr\tau\int) \rangle \xrightarrow{*} \\ \langle \mathsf{H}' \fatsemi \mathsf{S}, \mathsf{v} \fatsemi \wr\tau\int \rangle \xrightarrow{*} \\ \langle \mathsf{H}'' \fatsemi \mathsf{S}, \mathsf{v}' \fatsemi \cdot \rangle \end{array}$$

Now, we appeal to Lemma 0.5.1

This means that the value that we ran down to is in $(W''.k, \mathsf{v}') \in \mathcal{V}^{\lambda}[\![\downarrow\tau]\!]$, which is exactly what we need to show.

□

0.5.5 *Proving compatibility lemmas*

For the lemmas relating to the state extension, we use the following notation:

$$[\![\Gamma \vdash_{\mathbf{S}} \mathsf{P} : \tau]\!] \equiv \forall W\, \varphi\, \gamma.\ (W, \varphi, \gamma) \in \mathcal{G}^{\mathbf{S}}[\![\Gamma]\!] \implies (W, \mathrm{flocs}(\gamma(\mathsf{P})), \gamma(\mathsf{P})) \in \mathcal{E}^{\mathbf{S}}[\![\tau]\!]$$

0.5.6 *Supporting Lemmas*

Lemma 0.5.5 (relevant locations subset)**.** *If* $\varphi_1 \subset \varphi$, $\varphi_2 \subset \varphi$, *and* $\mathsf{H} :_\varphi W$ *then if* $\varphi_1 = \mathit{flocs}(\mathsf{P})$, $\langle \mathsf{H} \fatsemi \mathsf{S} \fatsemi \mathsf{P} \rangle \xrightarrow{*} \langle \mathsf{H}^1 \fatsemi \mathsf{S}^1 \fatsemi \mathsf{P}^1 \rangle$, *and for some* φ_1', $W^1 \sqsubseteq W$, $\mathsf{H}^1 :_{\varphi_1' \cup \varphi_1} W^1$, *then* $\mathsf{H}_1 :_{\varphi_2} W^1$.

Proof. Consider what needs to be true for $\mathsf{H}^1 :_{\varphi_2} W^1$. For every location in φ_2, either it is marked as dead in W^1, or the location must be in H^1 and must map to a value in the relation described by W^1. Since we know that $\varphi_2 \subset \varphi$ and $\mathsf{H} :_\varphi W$, we have a starting point at which these facts held. Since $W^1 \sqsubseteq W$, we know the only changes to the world can be adding locations or marking existing locations as dead. Since $\mathsf{H}^1 :_{\varphi_1} W^1$, we know that anything in $\varphi_2 \cap \varphi_1$ is satisfied. What about locations not in that set? Since $\varphi_1 = \mathrm{flocs}(\mathsf{P})$, we know the program only knew about the locations in φ_1—there is no way for an existing location to be synthesized out of thin air—and thus any locations in $\varphi_2 \setminus \varphi_1$ will have been unchanged between H and H_1, so we are done. □

Corollary 0.5.6 (Antireduction λ)**.**
If $\forall k\, \varphi'\, \mathsf{H}\, \mathsf{H}', \mathsf{S}.\ (k - j, \varphi', \mathsf{push}\ \mathsf{v}_1; \mathsf{push}\ \mathsf{v}_2; \ldots \mathsf{push}\ \mathsf{v}_\mathsf{n}; \mathsf{P}) \in \mathcal{E}^\lambda[\![\tau]\!]$, *and* $\langle \mathsf{H} \fatsemi \mathsf{S} \fatsemi \mathsf{P}'; \mathsf{P} \rangle \xrightarrow{j} \langle \mathsf{H}' \fatsemi \mathsf{S}, \mathsf{v}_1, \mathsf{v}_2, \ldots, \mathsf{v}_\mathsf{n} \fatsemi \mathsf{P} \rangle$ *then* $(k, \varphi, \mathsf{P}'; \mathsf{P}) \in \mathcal{E}^\lambda[\![\tau]\!]$.

Proof. Our obligation is to show that

$$\forall \mathsf{H}, \mathsf{H}', \mathsf{S}, \mathsf{S}', j < k.\ \langle \mathsf{H} \fatsemi \mathsf{S} \fatsemi P'; P \rangle \xrightarrow{j} \langle \mathsf{H}' \fatsemi \mathsf{S}' \fatsemi \cdot \rangle \implies (\mathsf{S}' = \mathsf{Fail}\ \mathsf{c} \wedge \mathsf{c} \in \textsc{OkErr}) \ \vee\ \exists \mathsf{v}.\ \Big(\mathsf{S}' = \mathsf{S}, \mathsf{v} \wedge (k - j, \mathsf{v}) \in \mathcal{V}^\lambda[\![\tau]\!]\Big)$$

From our second hypothesis, we know that

$$\langle \mathsf{H} \fatsemi \mathsf{S} \fatsemi P'; P \rangle \xrightarrow{j'} \langle \mathsf{H}^* \fatsemi \mathsf{v}_1, \ldots, \mathsf{v}_\mathsf{n} \fatsemi P \rangle \xrightarrow{j - j'} \langle \mathsf{H}' \fatsemi \mathsf{S}' \fatsemi \cdot \rangle$$

Our first hypothesis then tells us that

$$\big(\mathsf{S}' = \mathsf{Fail}\ \mathsf{c} \wedge \mathsf{c} \in \textsc{OkErr}\big) \vee \exists \mathsf{v}.\ \Big(\mathsf{S}' = \mathsf{S}, \mathsf{v} \wedge \big((k - j') - (j - j'), \mathsf{v}\big) \in \mathcal{V}^\lambda[\![\tau]\!]\Big)$$

which suffices to complete the proof.

□

Corollary 0.5.7 (Antireduction $\mathbf{S}$)**.**
If $\forall W'\, \varphi'\, \mathsf{H}\, \mathsf{H}', \mathsf{S}.\ (W', \varphi', \mathsf{push}\ \mathsf{v}_1; \mathsf{push}\ \mathsf{v}_2; \ldots \mathsf{push}\ \mathsf{v}_\mathsf{n}; \mathsf{P}) \in \mathcal{E}^\mathbf{S}[\![\tau]\!]$ *and* $W' \sqsubseteq W$, $\mathsf{H} :_\varphi W$, $\mathsf{H}' :_{\varphi \cup \varphi'} W'$, *and* $\langle \mathsf{H} \fatsemi \mathsf{S} \fatsemi \mathsf{P}'; \mathsf{P} \rangle \xrightarrow{*} \langle \mathsf{H}' \fatsemi \mathsf{S}, \mathsf{v}_1, \mathsf{v}_2, \ldots, \mathsf{v}_\mathsf{n} \fatsemi \mathsf{P} \rangle$ *then* $(W, \varphi, \mathsf{P}'; \mathsf{P}) \in \mathcal{E}^\mathbf{S}[\![\tau]\!]$.

Proof. We consider heap $\mathsf{H} :_{\varphi} W$, arbitrary stack S. We know that if the term in question does not run forever (which, if it does, then the suffix P does as well, so we are done), then it steps to a terminal configuration $\langle \mathsf{H}^F ; \mathsf{S}^F ; \cdot \rangle$. We need to show that, assuming that is not an error, $\mathsf{S}^F = \mathsf{S}, \mathsf{v}$ and for some φ^F and $W^F \sqsubseteq W$, $\mathsf{H}^F :_{\varphi^F} W^F$ and $(W^F, \varphi^F, \mathsf{v}) \in \mathcal{V}^{\mathbf{S}}[\![\tau]\!]$. We know that $\langle \mathsf{H} ; \mathsf{S} ; \mathsf{P}'; \mathsf{P} \rangle \xrightarrow{*} \langle \mathsf{H}' ; \mathsf{S}, \mathsf{v}_1, \ldots, \mathsf{v}_n ; \mathsf{P} \rangle$ and that for some $W' \sqsubseteq W$, $\mathsf{H}' :_{\varphi \cup \varphi'} W'$. So we instantiate our first hypothesis with H' and S. After n steps, it is in exactly the configuration our term left off in. We know it doesn't run forever, and if it errors, similarly, our overall term must error, so we conclude that it runs to a terminal configuration which due to confluence, will be the same one. Thus, we know $\mathsf{H}^F :_{\varphi \cup \varphi' \cup \varphi^F} W^F$, which is stronger than we need, and $(W^F, \varphi^F, \mathsf{v}) \in \mathcal{V}^{\mathbf{S}}[\![\tau]\!]$, exactly as needed. □

0.5.7 FunLang *Compatibility Lemmas*

$$
\begin{aligned}
&[\![\mathtt{I}; \Gamma \vdash \mathsf{P} : \tau]\!] \equiv \\
&\forall k\, \gamma.\ \forall ((k, \emptyset), \emptyset, \gamma^{\mathtt{I}^{\mathbf{S}}}) \in \mathcal{G}^{\mathbf{S}}[\![\mathtt{I}^{\mathbf{S}}]\!].\ \forall ((k, \emptyset), \emptyset, \gamma^{\mathtt{I}^{\mathbf{X}}}) \in \mathcal{G}^{\mathbf{X}}[\![\mathtt{I}^{\mathbf{X}}]\!].\ (k, \gamma) \in \mathcal{G}^{\lambda}[\![\Gamma]\!] \implies \\
&\quad (k, \gamma^{\mathtt{I}^{\mathbf{X}}}(\gamma^{\mathtt{I}^{\mathbf{S}}}(\gamma(\mathsf{P})))) \in \mathcal{E}^{\lambda}[\![\tau]\!]
\end{aligned}
$$

We now state and prove all the compatibility lemmas for our source language. Note that we have to prove these *three times*: once for each model, though the boundary terms only exist at the top level, and they are the most challenging/interesting.

Lemma 0.5.8 (unit)**.** *Show that* $[\![\mathtt{I}; \Gamma \vdash \mathsf{push}\ 0 : \mathtt{unit}]\!]$.

Proof. Since 0 has no free variables, what we need to show is that $(k, \mathsf{push}\ 0) \in \mathcal{E}^{\lambda}[\![\mathtt{unit}]\!]$. Given any H, γ, we can see that we take one step from $\langle \mathsf{H} ; \gamma ; \mathsf{push}\ 0 \rangle$ to $\langle \mathsf{H} ; \gamma, 0 ; \cdot \rangle$, and thus provided k was larger than 1 (else, trivial), what remains to show is that $(k-1, 0) \in \mathcal{V}^{\lambda}[\![\mathtt{unit}]\!]$. But this is trivial by the definition of the value relation.

□

Lemma 0.5.9 (bool)**.** *Show for any* n, $[\![I; \Gamma \vdash \mathsf{push}\ \mathsf{n} : \mathtt{bool}]\!]$.

Proof. This proof is identical to that of `unit`. □

Lemma 0.5.10 (if)**.** *If* $[\![\mathtt{I}; \Gamma \vdash \mathsf{P} : \mathtt{bool}]\!]$, $[\![\mathtt{I}; \Gamma \vdash \mathsf{P}_1 : \tau]\!]$, *and* $[\![\mathtt{I}; \Gamma \vdash \mathsf{P}_2 : \tau]\!]$ *then*
$[\![\mathtt{I}; \Gamma \vdash \mathsf{P}; \mathsf{if0}\ \mathsf{P}_1\ \mathsf{P}_2 : \tau]\!]$.

Proof. We are given $(k, \gamma) \in \mathcal{G}^{\lambda}[\![\Gamma]\!]$, $((k, \emptyset), \emptyset, \gamma^{\mathtt{I}^{\mathbf{S}}}) \in \mathcal{G}^{\mathbf{S}}[\![\mathtt{I}^{\mathbf{S}}]\!]$, $((k, \emptyset), \emptyset, \gamma^{\mathtt{I}^{\mathbf{X}}}) \in \mathcal{G}^{\mathbf{X}}[\![\mathtt{I}^{\mathbf{X}}]\!]$, and need to show that $(k, \gamma^{\mathtt{I}^{\mathbf{X}}}(\gamma^{\mathtt{I}^{\mathbf{S}}}(\gamma(\mathsf{P}; \mathsf{if0}\ \mathsf{P}_1\ \mathsf{P}_2)))) \in \mathcal{E}^{\lambda}[\![\tau]\!]$.

Pushing the substitutions in and combining $\gamma^{\mathtt{I}^{\mathbf{X}}} \circ \gamma^{\mathtt{I}^{\mathbf{S}}} \circ \gamma$ (for compactness) to $\gamma^{\mathtt{I}}$, we refine this slightly to:

$$(k, \gamma^{\mathtt{I}}(\mathsf{P}); \mathsf{if0}\ \gamma^{\mathtt{I}}(\mathsf{P}_1)\ \gamma^{\mathtt{I}}(\mathsf{P}_2)) \in \mathcal{E}^{\lambda}[\![\tau]\!]$$

Applying Lemma 0.5.6, it suffices to show

$$(k - j, \mathsf{push}\ \mathsf{v}_1; \mathsf{if0}\ \gamma^{\mathtt{I}}(\mathsf{P}_1)\ \gamma^{\mathtt{I}}(\mathsf{P}_2)) \in \mathcal{E}^{\lambda}[\![\tau]\!]$$

, since from the first hypothesis we know $\gamma^{\mathtt{I}}(\mathsf{P})$ will reduce to some value v_1 in $\mathcal{V}^{\lambda}[\![\mathtt{bool}]\!]$. We now appeal to Lemma 0.5.6 again, finishing the proof by noting that if v_1 is 0 then the induction hypothesis on $\gamma^{\mathtt{I}}(\mathsf{P}_1)$ suffices, and if it is not, the induction hypothesis on $\gamma^{\mathtt{I}}(\mathsf{P}_2)$) suffices.

□

Lemma 0.5.11 (int)**.** *For any* n*, show* $[\![I; \Gamma \vdash \mathsf{push}\ \mathsf{n} : \mathtt{int}]\!]$.

Proof. This case is analogous to `unit` and `bool`. □

Lemma 0.5.12 (op-=)**.** *If* $[\![I; \Gamma \vdash \mathsf{P}_1 : \mathtt{int}]\!]$ *and* $[\![I; \Gamma \vdash \mathsf{P}_2 : \mathtt{int}]\!]$*, show that* $[\![I; \Gamma \vdash \mathsf{P}_1; \mathsf{P}_2; \mathsf{equal?} : \mathtt{bool}]\!]$.

Proof. We are given $(k, \gamma) \in \mathcal{G}^{\lambda}[\![\Gamma]\!]$, $((k, \emptyset), \emptyset, \gamma^{\mathtt{I}^{\mathtt{S}}}) \in \mathcal{G}^{\mathtt{S}}[\![\mathtt{I}^{\mathtt{S}}]\!]$, $((k, \emptyset), \emptyset, \gamma^{\mathtt{I}^{\mathtt{X}}}) \in \mathcal{G}^{\mathtt{X}}[\![\mathtt{I}^{\mathtt{X}}]\!]$, and need to show that $(k, \gamma^{\mathtt{I}^{\mathtt{X}}}(\gamma^{\mathtt{I}^{\mathtt{S}}}(\gamma(\mathsf{P}_1; \mathsf{P}_1; \mathsf{equal?})))) \in \mathcal{E}^{\lambda}[\![\mathtt{bool}]\!]$.

Pushing the substitutions in and combining $\gamma^{\mathtt{I}^{\mathtt{X}}} \circ \gamma^{\mathtt{I}^{\mathtt{S}}} \circ \gamma$ (for compactness) to $\gamma^{\mathtt{I}}$, we refine this slightly to:

$(k, \gamma^{\mathtt{I}}(\mathsf{P}_1); \gamma^{\mathtt{I}}(\mathsf{P}_2); \mathsf{equal?})) \in \mathcal{E}^{\lambda}[\![\mathtt{bool}]\!]$

We apply Lemma 0.5.6 twice, appealing to our inductive hypotheses to reduce our obligation to showing that

$(k', \mathsf{push}\ \mathsf{v}_1; \mathsf{push}\ \mathsf{v}_2; \mathsf{equal?})) \in \mathcal{E}^{\lambda}[\![\mathtt{bool}]\!]$

for some v_1 and v_2 in $\mathcal{V}^{\lambda}[\![\mathtt{int}]\!]$. Since v_1 and v_2 are both integers, the term steps to either 0 or 1 on the stack, which means we satisfy our requirement to be in $\mathcal{V}^{\lambda}[\![\mathtt{bool}]\!]$, sufficient to complete the proof. □

Lemma 0.5.13 (op-¡)**.** *If* $[\![I; \Gamma \vdash \mathsf{P}_1 : \mathtt{int}]\!]$ *and* $[\![I; \Gamma \vdash \mathsf{P}_2 : \mathtt{int}]\!]$*, show that* $[\![I; \Gamma \vdash \mathsf{P}_1; \mathsf{P}_2; \mathsf{less?} : \mathtt{bool}]\!]$.

Proof. This proof is identical to that of =. □

Lemma 0.5.14 (op-+)**.** *If* $[\![I; \Gamma \vdash \mathsf{P}_1 : \mathtt{int}]\!]$ *and* $[\![I; \Gamma \vdash \mathsf{P}_2 : \mathtt{int}]\!]$*, show that* $[\![I; \Gamma \vdash \mathsf{P}_1; \mathsf{P}_2; \mathsf{add} : \mathtt{int}]\!]$.

Proof. This proof is identical to that of =. □

Lemma 0.5.15 (var)**.** *For any* $\mathtt{x} : \tau \in \Gamma$*, show that* $[\![\mathtt{I}; \Gamma \vdash \mathsf{push}\ \mathsf{x} : \tau]\!]$.

Proof. We are given $(k, \gamma) \in \mathcal{G}^{\lambda}[\![\Gamma]\!]$, $((k, \emptyset), \emptyset, \gamma^{\mathtt{I}^{\mathtt{S}}}) \in \mathcal{G}^{\mathtt{S}}[\![\mathtt{I}^{\mathtt{S}}]\!]$, $((k, \emptyset), \emptyset, \gamma^{\mathtt{I}^{\mathtt{X}}}) \in \mathcal{G}^{\mathtt{X}}[\![\mathtt{I}^{\mathtt{X}}]\!]$, and need to show that $(k, \gamma^{\mathtt{I}^{\mathtt{X}}}(\gamma^{\mathtt{I}^{\mathtt{S}}}(\gamma(\mathsf{push}\ \mathsf{x})))) \in \mathcal{E}^{\lambda}[\![\tau]\!]$.

Since $x \in \Gamma$, it isn't in $\mathtt{I}$, and thus we can eliminate the other substitutions. Further, we know from the definition of $\mathcal{G}^\lambda[\![\Gamma]\!]$ that there exists some v with $(k, \mathsf{v}) \in \mathcal{V}^\lambda[\![\tau]\!]$ such that $\gamma(\mathsf{x}) = \mathsf{v}$. This means we can substitute, yielding this as a goal:

$$(k, \mathsf{push\ v}) \in \mathcal{E}^\lambda[\![\tau]\!]$$

Now we can choose an arbitrary heap H and stack S, take one step, and end up in a terminal state with stack S, v. Since $(k, \mathsf{v}) \in \mathcal{V}^\lambda[\![\tau]\!]$, we are done.

□

Lemma 0.5.16 (pair)**.** *If* $[\![I; \Gamma \vdash \mathsf{P}_1 : \tau_1]\!]$ *and* $[\![I; \Gamma \vdash \mathsf{P}_2 : \tau_2]\!]$*, show that* $[\![I; \Gamma \vdash \mathsf{P}_1; \mathsf{P}_2; \mathsf{lam\ x_2.\ lam\ x_1.push\ [x_1, x_2]} : \tau_1 \times \tau_2]\!]$.

Proof. We are given $(k, \gamma) \in \mathcal{G}^\lambda[\![\Gamma]\!]$, $((k, \emptyset), \emptyset, \gamma^{\mathtt{I}^\mathtt{S}}) \in \mathcal{G}^\mathtt{S}[\![\mathtt{I}^\mathtt{S}]\!]$, $((k, \emptyset), \emptyset, \gamma^{\mathtt{I}^\mathtt{X}}) \in \mathcal{G}^\mathtt{X}[\![\mathtt{I}^\mathtt{X}]\!]$, and need to show that $(k, \gamma^{\mathtt{I}^\mathtt{X}}(\gamma^{\mathtt{I}^\mathtt{S}}(\gamma(\mathsf{P}_1; \mathsf{P}_2; \mathsf{lam\ x_2.lam\ x_1.push\ [x_1, x_2]})))) \in \mathcal{E}^\lambda[\![\tau_1 \times \tau_2]\!]$.

Pushing the substitutions in and combining $\gamma^{\mathtt{I}^\mathtt{X}} \circ \gamma^{\mathtt{I}^\mathtt{S}} \circ \gamma$ (for compactness) to $\gamma^{\mathtt{I}}$, we refine this slightly to:

$(k, \gamma^{\mathtt{I}}(\mathsf{P}_1); \gamma^{\mathtt{I}}(\mathsf{P}_2); \mathsf{lam\ x_2.lam\ x_1.push\ [x_1, x_2]})) \in \mathcal{E}^\lambda[\![\tau_1 \times \tau_2]\!]$

We apply Lemma 0.5.6 twice, appealing to both induction hypotheses, to reduce our obligation to showing

$(k', \mathsf{push\ v_1; push\ v_2; lam\ x_2.lam\ x_1.push\ [x_1, x_2]})) \in \mathcal{E}^\lambda[\![\tau_1 \times \tau_2]\!]$

where (k', v_1) is in $\mathcal{V}^\lambda[\![\tau_1]\!]$ and (k', v_2) is in $\mathcal{V}^\lambda[\![\tau_2]\!]$. The term then takes three steps, resulting in the value $[\mathsf{v}_1, \mathsf{v}_2]$ on the stack, which suffices to finish the proof.

□

Lemma 0.5.17 (fst)**.** *If* $[\![\mathtt{I}; \Gamma \vdash \mathsf{P} : \tau_1 \times \tau_2]\!]$*, show that* $[\![\mathtt{I}; \Gamma \vdash \mathsf{P}_1; \mathsf{push\ 0; idx} : \tau_1]\!]$.

Proof. We are given $(k, \gamma) \in \mathcal{G}^\lambda[\![\Gamma]\!]$, $((k, \emptyset), \emptyset, \gamma^{\mathtt{I}^\mathtt{S}}) \in \mathcal{G}^\mathtt{S}[\![\mathtt{I}^\mathtt{S}]\!]$, $((k, \emptyset), \emptyset, \gamma^{\mathtt{I}^\mathtt{X}}) \in \mathcal{G}^\mathtt{X}[\![\mathtt{I}^\mathtt{X}]\!]$, and need to show that $(k, \gamma^{\mathtt{I}^\mathtt{X}}(\gamma^{\mathtt{I}^\mathtt{S}}(\gamma(\mathsf{P}_1; \mathsf{push\ 0; idx})))) \in \mathcal{E}^\lambda[\![\tau_1]\!]$.

Pushing the substitutions in and combining $\gamma^{\mathtt{I}^\mathtt{X}} \circ \gamma^{\mathtt{I}^\mathtt{S}} \circ \gamma$ (for compactness) to $\gamma^{\mathtt{I}}$, we refine this slightly to:

$(k, \gamma^{\mathtt{I}}(\mathsf{P}_1); \mathsf{push\ 0; idx})) \in \mathcal{E}^\lambda[\![\tau_1]\!]$

We apply Lemma 0.5.6 to reduce this to showing

$(k', \mathsf{push\ v; push\ 0; idx})) \in \mathcal{E}^\lambda[\![\tau_1]\!]$

where $(k', \mathsf{v}) \in \mathcal{V}^\lambda[\![\tau_1 \times \tau_2]\!]$, and thus has shape $[\mathsf{v}_1, \mathsf{v}_2]$. The term takes three steps to result in v_1 on top of the stack, which suffices to finish the proof.

□

Lemma 0.5.18 (snd). *If* $[\![\mathtt{I}; \Gamma \vdash \mathsf{P} : \tau_1 \times \tau_2]\!]$, *show that* $[\![\mathtt{I}; \Gamma \vdash \mathsf{P}_1; \mathsf{push}\ 1; \mathsf{idx} : \tau_2]\!]$.

Proof. This proof is nearly identical to that of `fst`. □

Lemma 0.5.19 (inl). *If* $[\![\mathtt{I}; \Gamma \vdash \mathsf{P} : \tau_1]\!]$, *show that* $[\![\mathtt{I}; \Gamma \vdash \mathsf{P}; \mathsf{lam}\ \mathsf{x}.\mathsf{push}\ [0, \mathsf{x}] : \tau_1 + \tau_2]\!]$.

Proof. We are given $(k, \gamma) \in \mathcal{G}^{\lambda}[\![\Gamma]\!]$, $((k, \emptyset), \emptyset, \gamma^{\mathtt{I}^{\mathtt{S}}}) \in \mathcal{G}^{\mathtt{S}}[\![\mathtt{I}^{\mathtt{S}}]\!]$, $((k, \emptyset), \emptyset, \gamma^{\mathtt{I}^{\mathtt{X}}}) \in \mathcal{G}^{\mathtt{X}}[\![\mathtt{I}^{\mathtt{X}}]\!]$, and need to show that $(k, \gamma^{\mathtt{I}^{\mathtt{X}}}(\gamma^{\mathtt{I}^{\mathtt{S}}}(\gamma(\mathsf{P}; \mathsf{lam}\ \mathsf{x}.\mathsf{push}\ [0, \mathsf{x}])))) \in \mathcal{E}^{\lambda}[\![\tau_1 + \tau_2]\!]$.

Pushing the substitutions in and combining $\gamma^{\mathtt{I}^{\mathtt{X}}} \circ \gamma^{\mathtt{I}^{\mathtt{S}}} \circ \gamma$ (for compactness) to $\gamma^{\mathtt{I}}$, we refine this slightly to:

$(k, \gamma^{\mathtt{I}}(\mathsf{P}_1); \mathsf{lam}\ \mathsf{x}.\mathsf{push}\ [0, \mathsf{x}]) \in \mathcal{E}^{\lambda}[\![\tau_1 + \tau_2]\!]$

We apply Lemma 0.5.6 to reduce this to

$(k', \mathsf{push}\ \mathsf{v}_1; \mathsf{lam}\ \mathsf{x}.\mathsf{push}\ [0, \mathsf{x}]) \in \mathcal{E}^{\lambda}[\![\tau_1 + \tau_2]\!]$

where $(k', \mathsf{v}_1) \in \mathcal{V}^{\lambda}[\![\tau_1]\!]$. This takes three steps to result in $[0, \mathsf{v}_1]$ on the stack, which suffices to complete the proof. □

Lemma 0.5.20 (inr). *If* $[\![\mathtt{I}; \Gamma \vdash \mathsf{P} : \tau_2]\!]$, *show that* $[\![\mathtt{I}; \Gamma \vdash \mathsf{P}; \mathsf{lam}\ \mathsf{x}.\mathsf{push}\ [1, \mathsf{x}] : \tau_1 + \tau_2]\!]$.

Proof. This proof is nearly identical to that of `inl`. □

Lemma 0.5.21 (match). *If* $[\![\mathtt{I}; \Gamma \vdash \mathsf{P} : \tau_1 + \tau_2]\!]$, $[\![\mathtt{I}; \Gamma, x : \tau_1 \vdash \mathsf{P}_1 : \tau]\!]$, *and* $[\![\mathtt{I}; \Gamma, y : \tau_2 \vdash \mathsf{P}_2 : \tau]\!]$, *show that*

$$[\![\mathtt{I}; \Gamma \vdash \begin{array}{l} \mathsf{P}; \mathsf{DUP}; \mathsf{push}\ 1; \mathsf{idx}; \mathsf{SWAP}; \\ \mathsf{push}\ 0; \mathsf{idx}; \mathsf{if0}\ (\mathsf{lam}\ \mathsf{x}.\mathsf{P}_1)\ (\mathsf{lam}\ \mathsf{y}.\mathsf{P}_2) \end{array} : \tau]\!]$$

Proof. We are given $(k, \gamma) \in \mathcal{G}^{\lambda}[\![\Gamma]\!]$, $((k, \emptyset), \emptyset, \gamma^{\mathtt{I}^{\mathtt{S}}}) \in \mathcal{G}^{\mathtt{S}}[\![\mathtt{I}^{\mathtt{S}}]\!]$, $((k, \emptyset), \emptyset, \gamma^{\mathtt{I}^{\mathtt{X}}}) \in \mathcal{G}^{\mathtt{X}}[\![\mathtt{I}^{\mathtt{X}}]\!]$, and need to show that, after pushing substitutions and combining $\gamma^{\mathtt{I}^{\mathtt{X}}} \circ \gamma^{\mathtt{I}^{\mathtt{S}}} \circ \gamma$ (for compactness) to $\gamma^{\mathtt{I}}$,

$$(k, \gamma^{\mathtt{I}}(\mathsf{P}); \mathsf{DUP}; \mathsf{push}\ 1; \mathsf{idx}; \mathsf{SWAP}; \mathsf{push}\ 0; \mathsf{idx}; \mathsf{if0}\ (\mathsf{lam}\ \mathsf{x}.\gamma^{\mathtt{I}}(\mathsf{P}_1))\ (\mathsf{lam}\ \mathsf{y}.\gamma^{\mathtt{I}}(\mathsf{P}_2))) \in \mathcal{E}^{\lambda}[\![\tau]\!]$$

We apply Lemma 0.5.6 to reduce this to showing

$$(k', \mathsf{v}; \mathsf{DUP}; \mathsf{push}\ 1; \mathsf{idx}; \mathsf{SWAP}; \mathsf{push}\ 0; \mathsf{idx}; \mathsf{if0}\ (\mathsf{lam}\ \mathsf{x}.\gamma^{\mathtt{I}}(\mathsf{P}_1))\ (\mathsf{lam}\ \mathsf{y}.\gamma^{\mathtt{I}}(\mathsf{P}_2))) \in \mathcal{E}^{\lambda}[\![\tau]\!]$$

where $(k', \mathsf{push}\ \mathsf{v}) \in \mathcal{V}^{\lambda}[\![\tau_1 + \tau_2]\!]$. We appeal to Lemma 0.5.6 again, noting that after seven steps we will either have a $\mathsf{v}_1, 0$ where v_1 is in $\mathcal{V}^{\lambda}[\![\tau_1]\!]$ or $\mathsf{v}_2, 1$ where v_2 is in $\mathcal{V}^{\lambda}[\![\tau_2]\!]$ on the top of the stack, and thus in either case,

after two more steps we can appeal to one of our induction hypotheses to complete the proof.

□

Lemma 0.5.22 (fold)**.** *If* $[\![\mathtt{I}; \Gamma \vdash \mathsf{P} : \tau[\mu\alpha.\tau/\alpha]]\!]$*, show that* $[\![\mathtt{I}; \Gamma \vdash \mathsf{P} : \mu\alpha.\tau]\!]$.

Proof. We are given $(k, \gamma) \in \mathcal{G}^{\lambda}[\![\Gamma]\!]$, $((k, \emptyset), \emptyset, \gamma^{\mathtt{I}^{\mathtt{S}}}) \in \mathcal{G}^{\mathtt{S}}[\![\mathtt{I}^{\mathtt{S}}]\!]$, $((k, \emptyset), \emptyset, \gamma^{\mathtt{I}^{\mathtt{X}}}) \in \mathcal{G}^{\mathtt{X}}[\![\mathtt{I}^{\mathtt{X}}]\!]$, and need to show that $(k, \gamma^{\mathtt{I}^{\mathtt{X}}}(\gamma^{\mathtt{I}^{\mathtt{S}}}(\gamma(\mathsf{P})))) \in \mathcal{E}^{\lambda}[\![\mu\alpha.\tau]\!]$.

This means we need to pick an arbitrary heap H and stack S and show that this runs down to a value in the value relation (or else runs forever or to a well-defined error).

We can instantiate our hypothesis with the same substitutions, combining $\gamma^{\mathtt{I}^{\mathtt{X}}} \circ \gamma^{\mathtt{I}^{\mathtt{S}}} \circ \gamma$ (for compactness) to $\gamma^{\mathtt{I}}$, and heap and stack. This means that (assuming no divergence beyond k, or error, which would finish the proof immediately):

$$\langle \mathsf{H} \fatsemi \mathsf{S} \fatsemi \gamma^{\mathtt{I}}(\mathsf{P}) \rangle \xrightarrow{j} \langle \mathsf{H} \fatsemi \mathsf{S}, \mathsf{v} \fatsemi \cdot \rangle$$

Now, we know from the hypothesis that $(k - j, \mathsf{v}) \in \mathcal{V}^{\lambda}[\![\tau[\mu\alpha.\tau/\alpha]]\!]$. What we need to show is that $(k - j, \mathsf{v})$ is also in $\mathcal{V}^{\lambda}[\![\mu\alpha.\tau]\!]$. But this is fine, since that definition only requires that the value be in $\mathcal{V}^{\lambda}[\![\tau[\mu\alpha.\tau/\alpha]]\!]$ for smaller step index, and our relations are closed under smaller step index.

□

Lemma 0.5.23 (unfold)**.** *If* $[\![\mathtt{I}; \Gamma \vdash \mathsf{P} : \mu\alpha.\tau]\!]$*, show that* $[\![\mathtt{I}; \Gamma \vdash \mathsf{P}; \mathsf{noop} : \tau[\mu\alpha.\tau/\alpha]]\!]$.

Proof. We are given $(k, \gamma) \in \mathcal{G}^{\lambda}[\![\Gamma]\!]$, $((k, \emptyset), \emptyset, \gamma^{\mathtt{I}^{\mathtt{S}}}) \in \mathcal{G}^{\mathtt{S}}[\![\mathtt{I}^{\mathtt{S}}]\!]$, $((k, \emptyset), \emptyset, \gamma^{\mathtt{I}^{\mathtt{X}}}) \in \mathcal{G}^{\mathtt{X}}[\![\mathtt{I}^{\mathtt{X}}]\!]$, and need to show that $(k, \gamma^{\mathtt{I}^{\mathtt{X}}}(\gamma^{\mathtt{I}^{\mathtt{S}}}(\gamma(\mathsf{P}; \mathsf{noop})))) \in \mathcal{E}^{\lambda}[\![\tau[\mu\alpha.\tau/\alpha]]\!]$.

This means we need to pick an arbitrary heap H and stack S and show that this runs down to a value in the value relation (or else runs forever or to a well-defined error).

We can instantiate our hypothesis with the same substitutions, combining $\gamma^{\mathtt{I}^{\mathtt{X}}} \circ \gamma^{\mathtt{I}^{\mathtt{S}}} \circ \gamma$ (for compactness) to $\gamma^{\mathtt{I}}$, and heap and stack. This means that (assuming no divergence beyond k, or error, which would finish the proof immediately):

$$\langle \mathsf{H} \fatsemi \mathsf{S} \fatsemi \gamma^{\mathtt{I}}(\mathsf{P}) \rangle \xrightarrow{j} \langle \mathsf{H} \fatsemi \mathsf{S}, \mathsf{v} \fatsemi \cdot \rangle$$

Now, we return to our original program, which runs as:

$$\langle \mathsf{H} \fatsemi \mathsf{S} \fatsemi \gamma^{\mathtt{I}}(\mathsf{P}); \mathsf{noop} \rangle \xrightarrow{j} \langle \mathsf{H} \fatsemi \mathsf{S}, \mathsf{v} \fatsemi \mathsf{noop} \rangle \to \langle \mathsf{H} \fatsemi \mathsf{S}, \mathsf{v} \fatsemi \cdot \rangle$$

Now, we know from the hypothesis that $(k - j, \mathsf{v}) \in \mathcal{V}^{\lambda}[\![\mu\alpha.\tau]\!]$. What we need to show is that $(k - j - 1, \mathsf{v})$ (since we took one more step) is in

$\mathcal{V}^{\lambda}[\![\tau[\mu\alpha.\tau/\alpha]]\!]$. But, the definition of $\mathcal{V}^{\lambda}[\![\mu\alpha.\tau]\!]$ gives us this immediately, as our step index is lower. □

Lemma 0.5.24 (fun)**.** *If* $[\![\mathtt{I}; \Gamma, \mathtt{f} : (\tau_1, \ldots, \tau_n) \to \tau', \mathtt{x_i} : \tau_i \vdash \mathsf{P} : \tau']\!]$*, show that* $[\![\mathtt{I}; \Gamma \vdash \mathsf{push}\ (\mathsf{thunk}\ \mathsf{push}\ (\mathsf{thunk}\ \mathsf{lam}\ \mathsf{f}.\mathsf{lam}\ \mathsf{x_n}.\ldots\ \mathsf{lam}\ \mathsf{x_1}.\mathsf{P}); \mathsf{fix}) : (\tau_1, \ldots, \tau_n) \to \tau']\!]$

Proof. We are given $(k, \gamma) \in \mathcal{G}^{\lambda}[\![\Gamma]\!]$, $((k, \emptyset), \emptyset, \gamma^{\mathtt{I}^{\mathtt{S}}}) \in \mathcal{G}^{\mathtt{S}}[\![\mathtt{I}^{\mathtt{S}}]\!]$, $((k, \emptyset), \emptyset, \gamma^{\mathtt{I}^{\mathtt{X}}}) \in \mathcal{G}^{\mathtt{X}}[\![\mathtt{I}^{\mathtt{X}}]\!]$, and need to show, after pushing in substitutions and combining $\gamma^{\mathtt{I}^{\mathtt{X}}} \circ \gamma^{\mathtt{I}^{\mathtt{S}}} \circ \gamma$ (for compactness) to $\gamma^{\mathtt{I}}$:

$$(k, \mathsf{push}\ (\mathsf{thunk}\ \mathsf{push}\ (\mathsf{thunk}\ \mathsf{lam}\ \mathsf{f}.\mathsf{lam}\ \mathsf{x_n}.\ldots\ \mathsf{lam}\ \mathsf{x_1}.\gamma^{\mathtt{I}}(\mathsf{P})); \mathsf{fix})) \in \mathcal{E}^{\lambda}[\![(\tau_1, \ldots, \tau_n) \to \tau']\!]$$

Following the definition of $\mathcal{E}^{\lambda}[\![\tau]\!]$, we choose an arbitrary H and S and run the term, which after one step, results in the thunk on the stack. That means what we need to show is:

$$(k, \mathsf{thunk}\ \mathsf{push}\ (\mathsf{thunk}\ \mathsf{lam}\ \mathsf{f}.\mathsf{lam}\ \mathsf{x_n}.\ldots\ \mathsf{lam}\ \mathsf{x_1}.\gamma^{\mathtt{I}}(\mathsf{P})); \mathsf{fix}) \in \mathcal{V}^{\lambda}[\![(\tau_1, \ldots, \tau_n) \to \tau']\!]$$

Syntactically, this clearly satisfies the value relation; that means what we need to show is:

$$\begin{array}{l} \forall \mathsf{v_i}\ k' < k. \wedge\ (k', \mathsf{v_i}) \in \mathcal{V}^{\lambda}[\![\tau_i]\!] \\ \implies (k', [\mathsf{x_1} \mapsto \mathsf{v_1}, \ldots, \mathsf{x_n} \mapsto \mathsf{v_n}, \\ \quad \mathsf{f} \mapsto (\mathsf{thunk}\ \mathsf{push}\ (\mathsf{thunk}\ \mathsf{lam}\ \mathsf{f}.\mathsf{lam}\ \mathsf{x_n}.\ldots\ \mathsf{lam}\ \mathsf{x_1}.\gamma^{\mathtt{I}}(\mathsf{P})); \mathsf{fix})]\mathsf{P}) \in \mathcal{E}^{\lambda}[\![\tau']\!] \end{array}$$

We do this by appeal to our hypothesis. Specifically, we construct an extended substitution γ':

$$\gamma^{\mathtt{I}}, \mathsf{x_1}{:}\mathsf{v_1}, \ldots, \mathsf{x_n}{:}\mathsf{v_n}, \mathsf{f} : (\mathsf{thunk}\ \mathsf{push}\ (\mathsf{thunk}\ \mathsf{lam}\ \mathsf{f}.\mathsf{lam}\ \mathsf{x_n}.\ldots\ \mathsf{lam}\ \mathsf{x_1}.\gamma^{\mathtt{I}}(\mathsf{P})); \mathsf{fix})$$

Note that f can be given the needed type in the relation because we are only considering step $k' < k$, and our overall induction is over step indices. Further, our relations are closed under step indices, which means our substitution $\gamma^{\mathtt{I}}$ is still valid when restricted to k'. This means that we know:

$$(k', \gamma'(P)) \in \mathcal{E}^{\lambda}[\![\tau']\!]$$

Which, expanding out γ', is exactly what we needed to show.

□

Lemma 0.5.25 (app). *If* $[\![\mathtt{I}; \Gamma \vdash \mathsf{P} : (\tau_1, \ldots, \tau_n) \to \tau']\!]$ *and for* $i \in \{1, \ldots, n\}$ $[\![\mathtt{I}; \Gamma \vdash \mathsf{P}_i : \tau_i]\!]$ *then*
$[\![\mathtt{I}; \Gamma \vdash \mathsf{P}; \mathsf{P}_1; \mathsf{SWAP} \ldots \mathsf{P}_n; \mathsf{SWAP}; \mathsf{call} : \tau']\!]$

Proof. We are given $(k, \gamma) \in \mathcal{G}^{\lambda}[\![\Gamma]\!]$, $((k, \emptyset), \emptyset, \gamma^b \mathtt{I}^{\mathtt{S}}) \in \mathcal{G}^{\mathtt{S}}[\![\mathtt{I}^{\mathtt{S}}]\!]$, $((k, \emptyset), \emptyset, \gamma^{\mathtt{I}^{\mathtt{X}}}) \in \mathcal{G}^{\mathtt{X}}[\![\mathtt{I}^{\mathtt{X}}]\!]$, and need to show that $(k, \gamma^{\mathtt{I}^{\mathtt{X}}}(\gamma^{\mathtt{I}^{\mathtt{S}}}(\gamma(\mathsf{P}; \mathsf{P}_1; \mathsf{SWAP} \ldots \mathsf{P}_n; \mathsf{SWAP}; \mathsf{call})))) \in \mathcal{E}^{\lambda}[\![\tau']\!]$. Pushing the substitutions in and combining $\gamma^{\mathtt{I}^{\mathtt{X}}} \circ \gamma^{\mathtt{I}^{\mathtt{S}}} \circ \gamma$ (for compactness) to $\gamma^{\mathtt{I}}$, we refine this slightly to:

$$(k, \gamma^{\mathtt{I}}(\mathsf{P}); \gamma^{\mathtt{I}}(\mathsf{P}_1); \mathsf{SWAP} \ldots \gamma^{\mathtt{I}}(\mathsf{P}_n); \mathsf{SWAP}; \mathsf{call}) \in \mathcal{E}^{\lambda}[\![\tau']\!]$$

Following the definition of $\mathcal{E}^{\lambda}[\![\tau]\!]$, we choose an arbitrary H and S and run the term. To figure out how it steps, we instantiate our first hypothesis with $\gamma^{\mathtt{I}}$, H, and S. This tells us that either P runs forever (in which case, the term is in the relation trivially), or:

$$\langle \mathsf{H} \fatsemi \mathsf{S} \fatsemi \gamma^{\mathtt{I}}(\mathsf{P}) \rangle \xrightarrow{j} \langle \mathsf{H}' \fatsemi \mathsf{S}' \fatsemi \cdot \rangle$$

And either S' is a well-defined error (in which case, the entire program would have run to the same error, and we are again done), or $\mathsf{S}, \mathsf{v_f}$ with $(k - j, \mathsf{v_f}) \in \mathcal{V}^{\lambda}[\![(\tau_1, \ldots, \tau_n) \to \tau']\!]$.

Then, we instantiate the second hypothesis with $\gamma^{\mathtt{I}}$, H', and S, resulting in a similar result for a smaller step index k_1 and H_1 and value v_1. We can repeat this process another $n-1$ times. This results in an overall evaluation of:

$$\begin{aligned}
&\langle \mathsf{H} \fatsemi \mathsf{S} \fatsemi \gamma^{\mathtt{I}}(\mathsf{P}); \gamma^{\mathtt{I}}(\mathsf{P}_1); \mathsf{SWAP} \ldots \gamma^{\mathtt{I}}(\mathsf{P}_n); \mathsf{SWAP}; \mathsf{call} \rangle \\
&\quad \xrightarrow{*} \langle \mathsf{H}' \fatsemi \mathsf{S}, \mathsf{v_f} \fatsemi \gamma^{\mathtt{I}}(\mathsf{P}_1); \mathsf{SWAP} \ldots \gamma^{\mathtt{I}}(\mathsf{P}_n); \mathsf{SWAP}; \mathsf{call} \rangle \\
&\quad \xrightarrow{*} \langle \mathsf{H}_1 \fatsemi \mathsf{S}, \mathsf{v_f}, \mathsf{v}_1 \fatsemi \mathsf{SWAP} \ldots \gamma^{\mathtt{I}}(\mathsf{P}_n); \mathsf{SWAP}; \mathsf{call} \rangle \\
&\quad \xrightarrow{*} \langle \mathsf{H}_1 \fatsemi \mathsf{S}, \mathsf{v}_1, \mathsf{v_f} \fatsemi \ldots \gamma^{\mathtt{I}}(\mathsf{P}_n); \mathsf{SWAP}; \mathsf{call} \rangle \\
&\ldots \\
&\quad \xrightarrow{*} \langle \mathsf{H}_n \fatsemi \mathsf{S}, \mathsf{v}_1, \mathsf{v}_2, \ldots, \mathsf{v}_n, \mathsf{v_f} \fatsemi \mathsf{call} \rangle
\end{aligned}$$

From $\mathcal{V}^{\lambda}[\![(\tau_1, \ldots, \tau_n) \to \tau']\!]$, we know the shape of $\mathsf{v_f}$, so we can expand that out and step further:

$$\begin{aligned}
&\langle \mathsf{H}_n \fatsemi \mathsf{S}, \mathsf{v}_1, \mathsf{v}_2, \ldots, \mathsf{v}_n, (\mathsf{thunk\ push\ (thunk\ lam\ f.lam\ x_n. \ldots\ lam\ x_1.P); fix}) \fatsemi \mathsf{call} \rangle \\
&\quad \to \langle \mathsf{H}_n \fatsemi \mathsf{S}, \mathsf{v}_1, \mathsf{v}_2, \ldots, \mathsf{v}_n \fatsemi \mathsf{push\ (thunk\ lam\ f.lam\ x_n. \ldots\ lam\ x_1.P); fix} \rangle \\
&\quad \to \langle \mathsf{H}_n \fatsemi \mathsf{S}, \mathsf{v}_1, \mathsf{v}_2, \ldots, \mathsf{v}_n, \mathsf{thunk\ lam\ f.lam\ x_n. \ldots\ lam\ x_1.P} \fatsemi \mathsf{fix} \rangle \\
&\quad \to \langle \mathsf{H}_n \fatsemi \mathsf{S}, \mathsf{v}_1, \mathsf{v}_2, \ldots, \mathsf{v}_n, \mathsf{thunk(push(thunk\ lam\ f.lam\ x_n. \ldots\ lam\ x_1.P); fix)} \fatsemi \\
&\qquad \mathsf{lam\ f.lam\ x_n. \ldots\ lam\ x_1.P} \rangle \\
&\quad \xrightarrow{n+1} \langle \mathsf{H}_n \fatsemi \mathsf{S} \fatsemi [\mathsf{x_i} \mapsto \mathsf{v_i}, \mathsf{f} \mapsto \mathsf{thunk(push(thunk\ lam\ f.lam\ x_n. \ldots\ lam\ x_1.P); fix)}]\mathsf{P} \rangle
\end{aligned}$$

Now we can appeal to the definition of $\mathcal{V}^{\lambda}[\![(\tau_1, \ldots, \tau_n) \to \tau']\!]$, which tells us that this term is in $\mathcal{E}^{\lambda}[\![\tau']\!]$, which is exactly what we need to complete the proof: we can instantiate that relation with H_n, S, and compose the two reductions together to produce the result needed. □

Lemma 0.5.26 (boundary $\mathbf{S}$)**.** $[\![\mathtt{I}^{\mathbf{S}} \uplus \uparrow\Gamma \vdash_{\mathbf{S}} \mathsf{P} : \tau]\!] \implies [\![\mathtt{I}; \Gamma \vdash \mathsf{P}; \wr\tau\int : \downarrow\tau]\!]$

Proof. Expanding the goal, we see we need to show:

$$\forall k\, \gamma.\ \forall((k,\emptyset),\emptyset,\gamma^{\mathtt{I}^{\mathbf{S}}}) \in \mathcal{G}^{\mathbf{S}}[\![\mathtt{I}^{\mathbf{S}}]\!].\ \forall((k,\emptyset),\emptyset,\gamma^{\mathtt{I}^{\mathbf{X}}}) \in \mathcal{G}^{\mathbf{X}}[\![\mathtt{I}^{\mathbf{X}}]\!].\ (k,\gamma) \in \mathcal{G}^{\lambda}[\![\Gamma]\!] \implies (k, \gamma^{\mathtt{I}^{\mathbf{X}}}(\gamma^{\mathtt{I}^{\mathbf{S}}}(\gamma(\mathsf{P}; \wr\tau\int)))) \in \mathcal{E}^{\lambda}[\![\downarrow\tau]\!]$$

From Lemma 0.5.2, we know $\wr\tau\int$ is closed, so we can push the substitutions in to just over P. Further, from the hypothesis, we know that P has no free variables from $\mathtt{I}^{\mathbf{X}}$, so we can eliminated that substitution.

The hypothesis that we are working with says:

$$\forall W\, \varphi\, \gamma'\, (W, \varphi, \gamma') \in \mathcal{G}^{\mathbf{S}}[\![\mathtt{I}^{\mathbf{S}} \uplus \uparrow\Gamma]\!] \land \varphi = \mathrm{flocs}(\gamma(P)) \implies (k, \varphi, \gamma'(\mathsf{P})) \in \mathcal{E}^{\mathbf{S}}[\![\tau]\!]$$

To instantiate the hypothesis, we need an environment γ' that satisfies $\mathcal{G}^{\mathbf{S}}[\![\mathtt{I}^{\mathbf{S}} \uplus \uparrow\Gamma]\!]$. We argue that it is exactly γ composed with $\gamma^{\mathtt{I}^{\mathbf{S}}}$: we know they are disjoint, and we know the former can be lifted into the latter via Lemma 0.5.1. This means, in particular, that φ is $\emptyset$.

Since we have no relevant locations, any heap will satisfy the expression relation: in particular, the arbitrary H that we have to consider for our obligation, and we can similarly use the arbitrary stack S. This means that we our hypothesis tells us that:

$$\langle \mathsf{H} \fatsemi \mathsf{S} \fatsemi (\gamma^{\mathtt{I}^{\mathbf{S}}}(\gamma(\mathsf{P})) \rangle \xrightarrow{*} \langle \mathsf{H}' \fatsemi \mathsf{S}' \fatsemi \cdot \rangle$$

Unless we run beyond our step budget, in which case we are trivially in the relation. Similarly, if we run to $\mathsf{Fail}\ \mathsf{c}$, we are also in our relation. Otherwise, we know that $\mathsf{S}' = \mathsf{S}, \mathsf{v}$ and, for a future world $W' \sqsubseteq W$ that H' satisfies with the relevant locations φ', $(W', \varphi', \mathsf{v}) \in \mathcal{V}^{\mathbf{S}}[\![\tau]\!]$.

Now, what we want to show is that this value is "contained" by the code in $\wr\tau\int$ to behave like $\downarrow\tau$. But, clearly we can't show that using the $\mathcal{E}^{\lambda}[\![\tau]\!]$ logical relation, as the value still can have locations it is closing over, etc. So, we proceed by two steps. First, we appeal to Lemma 0.5.3

This will tell us that we can evaluate the whole program at question further, to get to a point with a world $W'' \sqsubseteq W'$, φ'', $\mathsf{H}'' :_{\varphi'' \cup \varphi'} W''$ and $(W'', \varphi'', \mathsf{v}') \in \mathcal{V}^{\mathbf{S}}[\![\uparrow\downarrow\tau]\!]$:

$$\langle \mathsf{H} \fatsemi \mathsf{S} \fatsemi (\gamma^{\mathbf{I^S}}(\gamma(\mathsf{P}); \lbag \tau \rbag \rangle \xrightarrow{*}$$
$$\langle \mathsf{H}' \fatsemi \mathsf{S}, \mathsf{v} \fatsemi \lbag \tau \rbag \rangle \xrightarrow{*}$$
$$\langle \mathsf{H}'' \fatsemi \mathsf{S}, \mathsf{v}' \fatsemi \cdot \rangle$$

Now, we appeal to Lemma 0.5.1

This means that the value that we ran down to is in $(W''.k, \mathsf{v}') \in \mathcal{V}^{\lambda}[\![\downarrow\tau]\!]$, which is exactly what we need to show.

□

0.5.8 FunLang *with* S *Compatibility Lemmas*

$$[\![\Gamma \vdash_{\mathbf{S}} \mathsf{P} : \tau]\!] \equiv \forall W\, \varphi\, \gamma.\ (W, \varphi, \gamma) \in \mathcal{G}^{\mathbf{S}}[\![\Gamma]\!] \implies (W, \mathrm{flocs}(\gamma(\mathsf{P})), \gamma(\mathsf{P})) \in \mathcal{E}^{\mathbf{S}}[\![\tau]\!]$$

Lemma 0.5.27 (unit)**.** *Show that* $[\![\Gamma \vdash_{\mathbf{S}} \mathsf{push}\ 0 : \mathtt{unit}]\!]$.

Proof. We expand the goal, pushing out substitution through and simplifying φ, given there are no free variables in push 0, to get an obligation:

$$(W, \emptyset, \mathsf{push}\ 0) \in \mathcal{E}^{\mathbf{S}}[\![\mathtt{unit}]\!]$$

To satisfy this, we note that we can take 1 step (if $W.k = 1$, we are in the relation trivially) to having 0 on top of the stack, with a world that has the same heap typing and, still, no relevant locations, thus satisfying $\mathcal{V}^{\mathbf{S}}[\![\mathtt{unit}]\!]$, as needed. □

Lemma 0.5.28 (bool)**.** *Show for any* n, $[\![\Gamma \vdash_{\mathbf{S}} \mathsf{n} : \mathtt{bool}]\!]$.

Proof. This proof is identical to that of unit. □

Lemma 0.5.29 (if)**.** *If* $[\![\Gamma \vdash_{\mathbf{S}} \mathsf{P} : \mathtt{bool}]\!]$, $[\![\Gamma \vdash_{\mathbf{S}} \mathsf{P}_1 : \tau]\!]$, *and* $[\![\Gamma \vdash_{\mathbf{S}} \mathsf{P}_2 : \tau]\!]$ *then*
$[\![\Gamma \vdash_{\mathbf{S}} \mathsf{P}; \mathsf{if0}\ \mathsf{P}_1\ \mathsf{P}_2 : \tau]\!]$.

Proof. We are given $(W, \varphi^{\dagger}, \gamma) \in \mathcal{G}^{\mathbf{S}}[\![\gamma]\!]$, where $\varphi = \mathrm{flocs}(\gamma(\mathsf{P}; \mathsf{if0}\ \mathsf{P}_1\ \mathsf{P}_2))$, and need to show that $(W, \varphi, \gamma(\mathsf{P}; \mathsf{if0}\ \mathsf{P}_1\ \mathsf{P}_2)) \in \mathcal{E}^{\mathbf{S}}[\![\tau]\!]$. Pushing the substitutions in, we refine this slightly to:

$$(W, \varphi, \gamma(\mathsf{P}); \mathsf{if0}\ \gamma(\mathsf{P}_1)\ \gamma(\mathsf{P}_2))) \in \mathcal{E}^{\mathbf{S}}[\![\tau]\!]$$

We appeal to Lemma 0.5.7, which reduces our obligation to

$$(W', \varphi', \mathsf{v}; \mathsf{if0}\ \gamma(\mathsf{P}_1)\ \gamma(\mathsf{P}_2))) \in \mathcal{E}^{\mathbf{S}}[\![\tau]\!]$$

where from our induction hypothesis we know that for $\mathsf{H} :_{\varphi} W$ and arbitrary S, $\langle \mathsf{H} \fatsemi \mathsf{S} \fatsemi \gamma(\mathsf{P}) \rangle \xrightarrow{*} \langle \mathsf{H}^1 \fatsemi \mathsf{S}^1 \fatsemi \cdot \rangle$ and either S^1 is a dynamic failure, in which case we are done, or it is v above, where for some $W' \sqsubseteq W$, φ', $\mathsf{H}^1 :_{\varphi \cup \varphi'} W'$.

From the definition of $\mathcal{V}^{\mathbf{S}}[\![\texttt{bool}]\!]$, we know v is either 0 or non-zero. In either case, we appeal to Lemma 0.5.7 again, relying on the corresponding hypotheses in the corresponding case that the term reduces to. □

Lemma 0.5.30 (int)**.** *For any* n*, show* $[\![\Gamma \vdash_{\mathbf{S}} \mathsf{push\ n} : \texttt{int}]\!]$.

Proof. This proof is essentially equivalent to that of `unit` and `bool`. □

Lemma 0.5.31 (op-=)**.** *If* $[\![\Gamma \vdash_{\mathbf{S}} \mathsf{P}_1 : \texttt{int}]\!]$ *and* $[\![\Gamma \vdash_{\mathbf{S}} \mathsf{P}_2 : \texttt{int}]\!]$*, show that* $[\![\Gamma \vdash_{\mathbf{S}} \mathsf{P}_1; \mathsf{P}_2; \mathsf{equal?} : \texttt{bool}]\!]$.

Proof. We are given $(W, \varphi^\dagger, \gamma) \in \mathcal{G}^{\mathbf{S}}[\![\gamma]\!]$, where $\varphi = \mathrm{flocs}(\gamma(\mathsf{P}_1; \mathsf{P}_2; \mathsf{equal?}))$, and need to show that $(W, \varphi, \gamma(\mathsf{P}_1; \mathsf{P}_2; \mathsf{equal?})) \in \mathcal{E}^{\mathbf{S}}[\![\texttt{bool}]\!]$. Pushing the substitutions in, we refine this slightly to:

$$(W, \varphi, \gamma(\mathsf{P}_1); \gamma(\mathsf{P}_2)\ \mathsf{equal?}) \in \mathcal{E}^{\mathbf{S}}[\![\texttt{bool}]\!]$$

We then apply Lemma 0.5.7 twice, relying on our two hypotheses to reduce our obligation to

$$(W', \varphi', \mathsf{push\ v}_1; \mathsf{push\ v}_2; \mathsf{equal?}) \in \mathcal{E}^{\mathbf{S}}[\![\texttt{bool}]\!]$$

Note that we instantiate the second hypothesis with $\varphi'' = \mathrm{flocs}(\gamma(\mathsf{P}_2)) \subset \varphi$, noting that $\mathsf{H}^1 :_{\varphi''} W^1$ via Lemma 0.5.5.

Since v_1 and v_2 are integers, this takes three steps to either 0 or 1 on top of the stack (with unchanged heap), which is sufficient to complete the proof.

□

Lemma 0.5.32 (op-¡)**.** *If* $[\![\Gamma \vdash_{\mathbf{S}} \mathsf{P}_1 : \texttt{int}]\!]$ *and* $[\![\Gamma \vdash_{\mathbf{S}} \mathsf{P}_2 : \texttt{int}]\!]$*, show that* $[\![\Gamma \vdash_{\mathbf{S}} \mathsf{P}_1; \mathsf{P}_2; \mathsf{less?} : \texttt{bool}]\!]$.

Proof. This proof is identical to that of =. □

Lemma 0.5.33 (op-+)**.** *If* $[\![\Gamma \vdash_{\mathbf{S}} \mathsf{P}_1 : \texttt{int}]\!]$ *and* $[\![\Gamma \vdash_{\mathbf{S}} \mathsf{P}_2 : \texttt{int}]\!]$*, show that* $[\![\Gamma \vdash_{\mathbf{S}} \mathsf{P}_1; \mathsf{P}_2; \mathsf{add} : \texttt{int}]\!]$.

Proof. This proof is identical to that of =. □

Lemma 0.5.34 (var)**.** *For any* $\mathtt{x} : \tau \in \Gamma$*, show that* $[\![\Gamma \vdash_{\mathbf{S}} \mathsf{push\ x} : \tau]\!]$.

Proof. We are given $(W, \varphi^\dagger, \gamma) \in \mathcal{G}^{\mathbf{S}}[\![\gamma]\!]$, where $\varphi = \mathrm{flocs}(\gamma(\mathsf{push\ x}))$, and need to show that $(W, \varphi, \gamma(\mathsf{push\ x})) \in \mathcal{E}^{\mathbf{S}}[\![\tau]\!]$.

Based on the definition of $\mathcal{G}^{\mathbf{S}}[\![\gamma]\!]$, we know that $\gamma(\mathsf{x}) = \mathsf{v}$ for some v where $(W, \varphi', \mathsf{v}) \in \mathcal{V}^{\mathbf{S}}[\![\tau]\!]$ and $\varphi' \subset \varphi$. Substituting, we can refine our proof goal to:

$$(W, \varphi, \mathsf{push\ v}) \in \mathcal{E}^{\mathbf{S}}[\![\tau]\!]$$

And since $\varphi = \mathrm{flocs}(\mathsf{v})$, we know $\varphi' = \varphi$. This means that after one step starting from a heap satisfying W and φ', we terminate with v on the top of the stack, and we are done. □

Lemma 0.5.35 (pair)**.** *If* $[\![\Gamma \vdash_{\mathbf{S}} \mathsf{P}_1 : \tau_1]\!]$ *and* $[\![\Gamma \vdash_{\mathbf{S}} \mathsf{P}_2 : \tau_2]\!]$ *then* $[\![\Gamma \vdash_{\mathbf{S}} \mathsf{P}_1; \mathsf{P}_2; \mathsf{lam}\ \mathsf{x}_2.\mathsf{lam}\ \mathsf{x}_1.\mathsf{push}\ [\mathsf{x}_1, \mathsf{x}_2] : \tau_1 \times \tau_2]\!]$

Proof. We are given $(W, \varphi^\dagger, \gamma) \in \mathcal{G}^{\mathbf{S}}[\![\gamma]\!]$, where $\varphi = \mathrm{flocs}(\gamma(\mathsf{P}_1; \mathsf{P}_2; \mathsf{lam}\ \mathsf{x}_2.\mathsf{lam}\ \mathsf{x}_1.\mathsf{push}\ [\mathsf{x}_1, \mathsf{x}_2]))$, and need to show that $(W, \varphi, \gamma(\mathsf{P}_1; \mathsf{P}_2; \mathsf{lam}\ \mathsf{x}_2.\mathsf{lam}\ \mathsf{x}_1.\mathsf{push}\ [\mathsf{x}_1, \mathsf{x}_2])) \in \mathcal{E}^{\mathbf{S}}[\![\tau_1 \times \tau_2]\!]$.

Pushing the substitutions in, we refine this slightly to:

$(W, \varphi, \gamma(\mathsf{P}_1); \gamma(\mathsf{P}_2); \mathsf{lam}\ \mathsf{x}_2.\mathsf{lam}\ \mathsf{x}_1.\mathsf{push}\ [\mathsf{x}_1, \mathsf{x}_2])) \in \mathcal{E}^{\mathbf{S}}[\![\tau_1 \times \tau_2]\!]$

This follows from two applications of Lemma 0.5.7 and the operational semantics, relying on Lemma 0.5.5 for the choice of relevant locations. □

Lemma 0.5.36 (fst)**.** *If* $[\![\Gamma \vdash_{\mathbf{S}} \mathsf{P} : \tau_1 \times \tau_2]\!]$*, show that* $[\![\Gamma \vdash_{\mathbf{S}} \mathsf{P}_1; \mathsf{push}\ 0; \mathsf{idx} : \tau_1]\!]$.

Proof. We are given $(W, \varphi^\dagger, \gamma) \in \mathcal{G}^{\mathbf{S}}[\![\gamma]\!]$, where $\varphi = \mathrm{flocs}(\gamma(\mathsf{P}_1; \mathsf{push}\ 0; \mathsf{idx}))$, and need to show that $(W, \varphi, \gamma(\mathsf{P}_1; \mathsf{push}\ 0; \mathsf{idx})) \in \mathcal{E}^{\mathbf{S}}[\![\tau_1]\!]$.

Pushing the substitutions in, we refine this slightly to:

$(W, \varphi, \gamma(\mathsf{P}_1); \mathsf{push}\ 0; \mathsf{idx}) \in \mathcal{E}^{\mathbf{S}}[\![\tau_1]\!]$

We apply Lemma 0.5.7, which, combined with the hypothesis, the operational semantics, and definition of the value relation is sufficient to complete the proof. □

Lemma 0.5.37 (snd)**.** *If* $[\![\Gamma \vdash_{\mathbf{S}} \mathsf{P} : \tau_1 \times \tau_2]\!]$*, show that* $[\![\Gamma \vdash_{\mathbf{S}} \mathsf{P}_1; \mathsf{push}\ 1; \mathsf{idx} : \tau_2]\!]$.

Proof. This proof is identical to `fst`. □

Lemma 0.5.38 (inl)**.** *If* $[\![\Gamma \vdash_{\mathbf{S}} \mathsf{P} : \tau_1]\!]$*, show that* $[\![\Gamma \vdash_{\mathbf{S}} \mathsf{P}; \mathsf{lam}\ \mathsf{x}.\mathsf{push}\ [0, \mathsf{x}] : \tau_1 + \tau_2]\!]$.

Proof. We are given $(W, \varphi^\dagger, \gamma) \in \mathcal{G}^{\mathbf{S}}[\![\gamma]\!]$, where $\varphi = \mathrm{flocs}(\gamma(\mathsf{P}; \mathsf{lam}\ \mathsf{x}.\mathsf{push}\ [0, \mathsf{x}]))$, and need to show that $(W, \varphi, \gamma(\mathsf{P}_1; \mathsf{lam}\ \mathsf{x}.\mathsf{push}\ [0, \mathsf{x}])) \in \mathcal{E}^{\mathbf{S}}[\![\tau_1 + \tau_2]\!]$.

Pushing the substitutions in, we refine this slightly to:

$(W, \varphi, \gamma(\mathsf{P}); \mathsf{lam}\ \mathsf{x}.\mathsf{push}\ [0, \mathsf{x}]) \in \mathcal{E}^{\mathbf{S}}[\![\tau_1 + \tau_2]\!]$

As in other cases, this follows from Lemma 0.5.7 and our hypothesis. □

Lemma 0.5.39 (inr)**.** *If* $[\![\Gamma \vdash_{\mathbf{S}} \mathsf{P} : \tau_2]\!]$*, show that* $[\![\Gamma \vdash_{\mathbf{S}} \mathsf{P}; \mathsf{lam}\ \mathsf{x}.\mathsf{push}\ [1, \mathsf{x}] : \tau_1 + \tau_2]\!]$.

Proof. This proof is identical to that of `inl`. □

Lemma 0.5.40 (match)**.** *If* $[\![\Gamma \vdash_{\mathbf{S}} \mathsf{P} : \tau_1 + \tau_2]\!]$*,* $[\![\Gamma, x : \tau_1 \vdash_{\mathbf{S}} \mathsf{P}_1 : \tau]\!]$*, and* $[\![\Gamma, y : \tau_2 \vdash_{\mathbf{S}} \mathsf{P}_2 : \tau]\!]$*, show that*

$$[\![\Gamma \vdash_{\mathbf{S}} \mathsf{P}; \mathsf{DUP}; \mathsf{push}\ 1; \mathsf{idx}; \mathsf{SWAP}; \mathsf{push}\ 0; \mathsf{idx}; \mathsf{if0}\ (\mathsf{lam}\ \mathsf{x}.\mathsf{P}_1)\ (\mathsf{lam}\ \mathsf{y}.\mathsf{P}_2) : \tau]\!]$$

Proof. We are given $(W, \varphi^\dagger, \gamma) \in \mathcal{G}^{\mathbf{S}}[\![\gamma]\!]$, where

$$\varphi = \mathrm{flocs}(\gamma(\mathsf{P}; \mathsf{DUP}; \mathsf{push\ 1}; \mathsf{idx}; \mathsf{SWAP}; \mathsf{push\ 0}; \mathsf{idx}; \mathsf{if0}\ (\mathsf{lam\ x.P_1})\ (\mathsf{lam\ y.P_2})))$$

and need to show that, after pushing in substitutions: Pushing the substitutions in, we refine this slightly to:

$$(W, \varphi, \gamma(\mathsf{P}); \mathsf{DUP}; \mathsf{push\ 1}; \mathsf{idx}; \mathsf{SWAP}; \mathsf{push\ 0}; \mathsf{idx}; \mathsf{if0}\ (\mathsf{lam\ x.}\gamma(\mathsf{P_1}))\ (\mathsf{lam\ y.}\gamma(\mathsf{P_2})))) \in \mathcal{E}^{\mathbf{S}}[\![\tau_1 + \tau_2]\!]$$

We appeal to Lemma 0.5.7 and the operational semantics to reduce this to considering the two possible branches: when $\mathcal{V}^{\mathbf{S}}[\![\tau_1 + \tau_1]\!]$ is $[0, \mathsf{v}]$ and when it is $[1, \mathsf{v}]$. In both cases, we again appeal to Lemma 0.5.7, but to the second or third hypothesis respectively, as operationally that is what we will reduce to, with appropriate substitution. □

Lemma 0.5.41 (fold)**.** *If* $[\![\Gamma \vdash_{\mathbf{S}} \mathsf{P} : \tau[\mu\alpha.\tau/\alpha]]\!]$*, show that* $[\![\Gamma \vdash_{\mathbf{S}} \mathsf{P} : \mu\alpha.\tau]\!]$.

Proof. We are given $(W, \varphi^\dagger, \gamma) \in \mathcal{G}^{\mathbf{S}}[\![\gamma]\!]$, where $\varphi = \mathrm{flocs}(\gamma(\mathsf{P}))$, and need to show that $(W, \varphi, \gamma(\mathsf{P})) \in \mathcal{E}^{\mathbf{S}}[\![\mu\alpha.\tau]\!]$.

This means we are given an heap $\mathsf{H} :_\varphi W$, stack γ, and, assuming we don't run forever or out of steps (in $W.k$ budget), we run down to $\langle \mathsf{H}' \fatsemi \gamma' \fatsemi \cdot \rangle$. We instantiate our first hypothesis with W, H, γ, and φ, to get that:

$$\langle \mathsf{H} \fatsemi \mathsf{S} \fatsemi \gamma(\mathsf{P}) \rangle \xrightarrow{j^1} \langle \mathsf{H}^1 \fatsemi \mathsf{S}^1 \fatsemi \cdot \rangle$$

Now, either S^1 is $\mathsf{Fail\ c}$ for appropriate c, in which case the entire program will be and we are done, or $\mathsf{S}^1 = \mathsf{S}, \mathsf{v}$ and for $W^1 \sqsubseteq W$, $\mathsf{H}^1 :_{\varphi^1 \cup \varphi} W^1$, and $(W^1, \varphi^1, \mathsf{v}) \in \mathcal{V}^{\mathbf{S}}[\![\tau[\mu\alpha.\tau/\alpha]]\!]$. Now, our obligation only needs us to prove that the resulting value, which is the same value, is in this relation at lower step index, so we are done. □

Lemma 0.5.42 (unfold)**.** *If* $[\![\Gamma \vdash_{\mathbf{S}} \mathsf{P} : \mu\alpha.\tau]\!]$*, show that* $[\![\Gamma \vdash_{\mathbf{S}} \mathsf{P}; \mathsf{noop} : \tau[\mu\alpha.\tau]]\!]$.

Proof. We are given $(W, \varphi^\dagger, \gamma) \in \mathcal{G}^{\mathbf{S}}[\![\gamma]\!]$, where $\varphi = \mathrm{flocs}(\gamma(\mathsf{P}; \mathsf{noop}))$, and need to show that $(W, \varphi, \gamma(\mathsf{P}; \mathsf{noop})) \in \mathcal{E}^{\mathbf{S}}[\![\tau[\mu\alpha.\tau]]\!]$.

This means we are given an heap $\mathsf{H} :_\varphi W$, stack γ, and, assuming we don't run forever or out of steps (in $W.k$ budget), we run down to $\langle \mathsf{H}' \fatsemi \gamma' \fatsemi \cdot \rangle$. We instantiate our first hypothesis with W, H, γ, and φ, to get that:

$$\langle \mathsf{H} \fatsemi \mathsf{S} \fatsemi \gamma(\mathsf{P}) \rangle \xrightarrow{j^1} \langle \mathsf{H}^1 \fatsemi \mathsf{S}^1 \fatsemi \cdot \rangle$$

Now, either S^1 is $\mathsf{Fail\ c}$ for appropriate c, in which case the entire program will be and we are done, or $\mathsf{S}^1 = \mathsf{S}, \mathsf{v}$ and for $W^1 \sqsubseteq W$, $\mathsf{H}^1 :_{\varphi^1 \cup \varphi} W^1$, and $(W^1, \varphi^1, \mathsf{v}) \in \mathcal{V}^{\mathbf{S}}[\![\mu\alpha.\tau]\!]$. Now, our original term steps as follows:

$$\langle \mathsf{H} \fatsemi \mathsf{S} \fatsemi \gamma(\mathsf{P}; \mathsf{noop}) \rangle \xrightarrow{j^1}$$
$$\langle \mathsf{H}^1 \fatsemi \mathsf{S}, \mathsf{v} \fatsemi \mathsf{noop} \rangle \to$$
$$\langle \mathsf{H}^1 \fatsemi \mathsf{S}, \mathsf{v} \fatsemi \cdot \rangle$$

We need to fulfill $\mathcal{E}^{\mathbf{S}}[\![\tau[\mu\alpha.\tau]]\!]$, which means we need to choose $W' \sqsubseteq W$, φ' such that $\mathsf{H}^1 :_{\varphi'} W'$ and $(W', \varphi', \mathsf{v}) \in \mathcal{V}^{\mathbf{S}}[\![\tau[\mu\alpha.\tau]]\!]$. We choose W' to be W^1 with the step index decreased by one. Because this is a strictly future world of W^1, this follows directly from the definition of $\mathcal{V}^{\mathbf{S}}[\![\mu\alpha.\tau]\!]$.

□

Lemma 0.5.43 (fun). *If* $[\![\Gamma, \mathtt{f} : (\tau_{\mathbf{1}}, \ldots, \tau_{\mathbf{n}}) \xrightarrow{\bullet} \tau', \mathtt{x_i} : \tau_{\mathbf{i}} \vdash_{\mathbf{S}} \mathsf{P} : \tau']\!]$, *show that* $[\![\Gamma \vdash_{\mathbf{S}} \mathsf{push\ (thunk\ push\ (thunk\ lam\ f.lam\ x_n. \ldots\ lam\ x_1.\gamma(P)); fix)} : (\tau_{\mathbf{1}}, \ldots, \tau_{\mathbf{n}}) \xrightarrow{\bullet} \tau']\!]$

Proof. We are given $(W, \varphi^\dagger, \gamma) \in \mathcal{G}^{\mathbf{S}}[\![\gamma]\!]$, where

$$\varphi = \mathrm{flocs}(\gamma(\mathsf{push\ (thunk\ push\ (thunk\ lam\ f.lam\ x_n. \ldots\ lam\ x_1.P); fix)}))$$

and need to show that

$$(W, \varphi, \gamma(\mathsf{push\ (thunk\ push\ (thunk\ lam\ f.lam\ x_n. \ldots\ lam\ x_1.P); fix)})) \in \mathcal{E}^{\mathbf{S}}[\![(\tau_{\mathbf{1}}, \ldots, \tau_{\mathbf{n}}) \xrightarrow{\circ} \tau']\!]$$

We can then push the substitution in to refine that to:

$$(W, \varphi, \mathsf{push\ (thunk\ push\ (thunk\ lam\ f.lam\ x_n. \ldots\ lam\ x_1.\gamma(P); fix)}) \in \mathcal{E}^{\mathbf{S}}[\![(\tau_{\mathbf{1}}, \ldots, \tau_{\mathbf{n}}) \xrightarrow{\circ} \tau']\!]$$

This means we are given a $\mathsf{H} :_\varphi W$, stack S, and, assuming we don't run forever or out of steps (in $W.k$ budget), we run down to $\langle \mathsf{H}' \fatsemi \mathsf{S}' \fatsemi \cdot \rangle$.

This clearly takes a single step to put thunk push (thunk lam f.lam $\mathsf{x_n}. \ldots$ lam $\mathsf{x_1}.\gamma(\mathsf{P}); \mathsf{fix}$ on the stack. We can thus choose W' to be W with k decreased by 1, the same relevant location set $\emptyset$. Thus we need to satisfy the value relation, which amounts to:

$$\begin{array}{l}
\forall \mathsf{v_i}\ \varphi_i\ W' \sqsupset W. \\
\varphi_i \subset \mathrm{dom}(W'.\Psi) \wedge (W', \varphi_i, \mathsf{v_i}) \in \mathcal{V}^{\mathbf{S}}[\![\tau_{\mathbf{i}}]\!] \\
\implies (W', \varphi \cup \bigcup_i \varphi_i, [\mathsf{x_1} \mapsto \mathsf{v_1}, \ldots, \mathsf{x_n} \mapsto \mathsf{v_n}, \\
\qquad \mathsf{f} \mapsto (\mathsf{thunk\ push\ (thunk\ lam\ f.lam\ x_n. \ldots\ lam\ x_1.\gamma(P)); fix})]\gamma(\mathsf{P})) \in \mathcal{E}^{\mathbf{S}}[\![\tau']\!]
\end{array}$$

Thus we choose an arbitrary future world $W'' \sqsubset W'$, and construct an extended substitution $\gamma' = \gamma, \mathsf{x_1} \mapsto \mathsf{v_1}, \ldots, \mathsf{x_n} \mapsto \mathsf{v_n}, \mathsf{f} \mapsto (\mathsf{thunk...})$. We argue that $(W'', \varphi \cup \bigcup_i \varphi_i, \gamma') \in \mathcal{G}^{\mathbf{S}}[\![\Gamma, \mathtt{f} : (\tau_{\mathbf{1}}, \ldots, \tau_{\mathbf{n}}) \xrightarrow{\circ} \tau', \mathtt{x_i} : \tau_{\mathbf{i}}]\!]$. Clearly,

all of the values $\mathsf{v_i}$ are in the value relation at the correct type. And, since W'' is a strict world extension, it has a smaller step index, which means that we can appeal to our inductive hypothesis to get that our function has the correct semantic type at that world.

That means we can instantiate our first hypothesis with W'', $\varphi \cup \bigcup_i \varphi_i$ and γ' to complete the proof.

□

Lemma 0.5.44 (app pure)**.** *If* $[\![\Gamma \vdash_{\mathbf{S}} \mathsf{P} : (\tau_{\mathbf{1}}, \ldots, \tau_{\mathbf{n}}) \xrightarrow{\circ} \tau']\!]$ *and for* $i \in \{1, \ldots, n\}$ $[\![\Gamma \vdash_{\mathbf{S}} \mathsf{P_i} : \tau_{\mathbf{i}}]\!]$ *then*
$[\![\Gamma \vdash_{\mathbf{S}} \mathsf{P}; \mathsf{P_1}; \mathsf{SWAP} \ldots \mathsf{P_n}; \mathsf{SWAP}; \mathsf{call} : \tau']\!]$

Proof. We are given $(W, \varphi^\dagger, \gamma) \in \mathcal{G}^{\mathbf{S}}[\![\gamma]\!]$, where

$$\varphi = \mathrm{flocs}(\gamma(\mathsf{P}; \mathsf{P_1}; \mathsf{SWAP} \ldots \mathsf{P_n}; \mathsf{SWAP}; \mathsf{call}))$$

and need to show that

$$(W, \varphi, \gamma(\mathsf{P}; \mathsf{P_1}; \mathsf{SWAP} \ldots \mathsf{P_n}; \mathsf{SWAP}; \mathsf{call})) \in \mathcal{E}^{\mathbf{S}}[\![\tau']\!]$$

We can then push the substitution in to refine that to:

$$(W, \varphi, \mathsf{P}; \mathsf{P_1}; \mathsf{SWAP} \ldots \mathsf{P_n}; \mathsf{SWAP}; \mathsf{call} \in \mathcal{E}^{\mathbf{S}}[\![\tau']\!]$$

This means we are given a $\mathsf{H} :_\varphi W$, stack S, and, assuming we don't run forever or out of steps (in $W.k$ budget), we run down to $\langle \mathsf{H}' \fatsemi \mathsf{S}' \fatsemi \cdot \rangle$.

To figure out how it steps, we instantiate our first hypothesis with W, γ, H, S and φ', where $\varphi' = \mathrm{flocs}(\gamma(\mathsf{P_1})) \subset \varphi$, noting that the heap will still satisfy the same world with the smaller φ', to get that:

$$\langle \mathsf{H} \fatsemi \mathsf{S} \fatsemi \gamma(\mathsf{P}) \rangle \xrightarrow{j^1} \langle \mathsf{H}^1 \fatsemi \mathsf{S}^1 \fatsemi \cdot \rangle$$

Now, either S^1 is $\mathsf{Fail}\ \mathsf{c}$ for appropriate c, in which case the entire program will be and we are done, or $\mathsf{S}^1 = \mathsf{S}, \mathsf{v_f}$ and for $W^1 \sqsubseteq W$, $\mathsf{H}^1 :_{\varphi^f \cup \varphi'} W^1$, and $(W^1, \varphi^f, \mathsf{v_f}) \in \mathcal{V}^{\mathbf{S}}[\![(\tau_{\mathbf{1}}, \ldots, \tau_{\mathbf{n}}) \xrightarrow{\circ} \tau']\!]$. From the value relation, we note that φ^f is $\emptyset$.

We then instantiate out second hypothesis with W^1, H^1, S^1, and $\varphi'' = \mathrm{flocs}(\gamma(\mathsf{P_2})) \subset \varphi$. Note that $\mathsf{H}^1 :_{\varphi''} W^1$ from Lemma 0.5.5.

This means that:

$$\langle \mathsf{H}^1 \fatsemi \mathsf{S}, \mathsf{v_f} \fatsemi \gamma(\mathsf{P_2}) \rangle \xrightarrow{j^2} \langle \mathsf{H}^2 \fatsemi \mathsf{S}^2 \fatsemi \cdot \rangle$$

Since this program began running in the same state as the previous one stopped, and the previous one began at the beginning of our whole program, again, we are either trivially in the relation or else we know that

$\mathsf{S}^2 = \mathsf{S}^1, \mathsf{v}_1 = \mathsf{S}, \mathsf{v}_\mathsf{f}, \mathsf{v}_1$ and for $W^2 \sqsubseteq W^1$, $\mathsf{H}^2 :_{\varphi^1 \cup \varphi''} W^2$, and $(W^2, \varphi^1, \mathsf{v}_1) \in \mathcal{V}^{\mathbf{S}}[\![\tau_1]\!]$.

We can repeat this process another $n-1$ times. This results in an overall evaluation of:

$$\begin{aligned}
&\langle \mathsf{H} \fatsemi \mathsf{S} \fatsemi \gamma(\mathsf{P}); \gamma(\mathsf{P}_1); \mathsf{SWAP} \ldots \gamma(\mathsf{P}_\mathsf{n}); \mathsf{SWAP}; \mathsf{call} \rangle \\
&\quad \xrightarrow{*} \langle \mathsf{H}' \fatsemi \mathsf{S}, \mathsf{v}_\mathsf{f} \fatsemi \gamma(\mathsf{P}_1); \mathsf{SWAP} \ldots \gamma(\mathsf{P}_\mathsf{n}); \mathsf{SWAP}; \mathsf{call} \rangle \\
&\quad \xrightarrow{*} \langle \mathsf{H}_1 \fatsemi \mathsf{S}, \mathsf{v}_\mathsf{f}, \mathsf{v}_1 \fatsemi \mathsf{SWAP} \ldots \gamma(\mathsf{P}_\mathsf{n}); \mathsf{SWAP}; \mathsf{call} \rangle \\
&\quad \xrightarrow{*} \langle \mathsf{H}_1 \fatsemi \mathsf{S}, \mathsf{v}_1, \mathsf{v}_\mathsf{f} \fatsemi \ldots \gamma(\mathsf{P}_\mathsf{n}); \mathsf{SWAP}; \mathsf{call} \rangle \\
&\ldots \\
&\quad \xrightarrow{*} \langle \mathsf{H}_n \fatsemi \mathsf{S}, \mathsf{v}_1, \mathsf{v}_2, \ldots, \mathsf{v}_\mathsf{n}, \mathsf{v}_\mathsf{f} \fatsemi \mathsf{call} \rangle
\end{aligned}$$

From $\mathcal{V}^{\mathbf{S}}[\![(\tau_1, \ldots, \tau_\mathbf{n}) \xrightarrow{\circ} \tau']\!]$, we know the shape of v_f, so we can expand that out and step further:

$$\begin{aligned}
&\langle \mathsf{H}_n \fatsemi \mathsf{S}, \mathsf{v}_1, \mathsf{v}_2, \ldots, \mathsf{v}_\mathsf{n}, (\mathsf{thunk\ push\ (thunk\ lam\ f.lam\ x_n.\ldots\ lam\ x_1.P); fix}) \fatsemi \mathsf{call} \rangle \\
&\quad \to \langle \mathsf{H}_n \fatsemi \mathsf{S}, \mathsf{v}_1, \mathsf{v}_2, \ldots, \mathsf{v}_\mathsf{n} \fatsemi \mathsf{push\ (thunk\ lam\ f.lam\ x_n.\ldots\ lam\ x_1.P); fix} \rangle \\
&\quad \to \langle \mathsf{H}_n \fatsemi \mathsf{S}, \mathsf{v}_1, \mathsf{v}_2, \ldots, \mathsf{v}_\mathsf{n}, \mathsf{thunk\ lam\ f.lam\ x_n.\ldots\ lam\ x_1.P} \fatsemi \mathsf{fix} \rangle \\
&\quad \to \langle \mathsf{H}_n \fatsemi \mathsf{S}, \mathsf{v}_1, \mathsf{v}_2, \ldots, \mathsf{v}_\mathsf{n}, \mathsf{thunk(push(thunk\ lam\ f.lam\ x_n.\ldots\ lam\ x_1.P); fix)} \fatsemi \\
&\qquad \mathsf{lam\ f.lam\ x_n.\ldots\ lam\ x_1.P} \rangle \\
&\quad \xrightarrow{n+1} \langle \mathsf{H}_n \fatsemi \mathsf{S} \fatsemi [\mathsf{x_i} \mapsto \mathsf{v_i}, \mathsf{f} \mapsto \mathsf{thunk(push(thunk\ lam\ f.lam\ x_n.\ldots\ lam\ x_1.P); fix)}]\mathsf{P} \rangle
\end{aligned}$$

Now we can appeal to the definition of $\mathcal{V}^{\mathbf{S}}[\![(\tau_1, \ldots, \tau_\mathbf{n}) \xrightarrow{\circ} \tau']\!]$, which tells us that this term is in $\mathcal{E}^{\mathbf{S}}[\![\tau']\!]$, given the values were in the value relation, which we know from each instantiated hypothesis. We then instantiate that relation with W^{n+1}, $\varphi^f \cup \bigcup_i \varphi^i$, and compose the reductions together to produce the result needed. □

Lemma 0.5.45 (app state)**.** *If* $[\![\Gamma \vdash_\mathbf{S} \mathsf{P} : (\tau_1, \ldots, \tau_\mathbf{n}) \xrightarrow{\bullet} \tau']\!]$ *and for* $i \in \{1, \ldots, n\}$ $[\![\Gamma \vdash_\mathbf{S} \mathsf{P_i} : \tau_\mathbf{i}]\!]$ *then*
$[\![\Gamma \vdash_\mathbf{S} \mathsf{P}; \mathsf{P}_1; \mathsf{SWAP} \ldots \mathsf{P}_\mathsf{n}; \mathsf{SWAP}; \mathsf{call} : \tau']\!]$

Proof. This proof is nearly identical to the previous one: the only difference is that φ^f is not empty, but that just carries down to the final instantiation which we left unsimplified in the above proof for clarity. □

0.5.9 *Proving libraries satisfy types*

The next step we need to do is prove that the library code that we are linking with satisfies the types that we are importing it as. We note that a single library may have multiple types that it can be given—and may be usable from different extensions, which have different reasoning principles. This is most noticeable in our case because our source language

and extensions lack polymorphism, while many of our library functions are naturally polymorphic: e.g., **ref** τ, not **ref** `int` or **ref** `bool`. However, when we prove the libraries sound, we can prove that they satisfy the more general pattern, and thus include them at whatever more concrete type is appropriate.

All the library code we used is repeated below. We show the types that we want to prove that the code has.

$$\begin{array}{lcll} \mathsf{ALLOC} & : & (\tau) \xrightarrow{\bullet} \mathbf{ref}\ \tau & \triangleq\ \mathsf{t{-}p}\ (\mathsf{t{-}l\ falloc.lam\ x.push\ x; alloc}); \mathsf{fix} \\ \mathsf{READ} & : & (\mathbf{ref}\ \tau) \xrightarrow{\bullet} \tau & \triangleq\ \mathsf{t{-}p}\ (\mathsf{t{-}l\ fread.lam\ l.push\ l; read}); \mathsf{fix} \\ \mathsf{WRITE} & : & (\mathbf{ref}\ \tau, \tau) \xrightarrow{\bullet} \mathtt{unit} & \triangleq\ \mathsf{t{-}p}\ (\mathsf{t{-}l\ fwrite.lam\ x.lam\ l.push\ l; push\ x; write; push\ 0}); \mathsf{fix} \end{array}$$

where $\mathsf{t{-}p} = \mathsf{thunk\ push}$ and $\mathsf{t{-}l} = \mathsf{thunk\ lam}$

Lemma 0.5.46 (alloc sound $\mathbf{S}$)**.**
$\forall W\ \tau.\ (W, \emptyset, \mathsf{thunk\ push\ (thunk\ lam\ falloc.lam\ x.push\ x; alloc); fix}) \in \mathcal{V}^{\mathbf{S}}[\![(\tau^+) \xrightarrow{\bullet} \mathbf{ref}\ \tau^+]\!]$

Proof. It suffices to show:

$$\begin{array}{l} \forall \mathsf{v}\ \varphi\ W' \sqsupset W.\varphi \subset \mathrm{dom}(W'.\Psi) \wedge (W', \varphi, \mathsf{v}) \in \mathcal{V}^{\mathbf{S}}[\![\tau^+]\!] \\ \quad \Longrightarrow\ (W', \varphi, \mathsf{push\ v; alloc}) \in \mathcal{E}^{\mathbf{S}}[\![\mathbf{ref}\ \tau^+]\!] \end{array}$$

Thus, we choose a $\mathsf{H} :_{\varphi} W'$, S, and take two steps $\langle \mathsf{H} \fatsemi \mathsf{S} \fatsemi \mathsf{push\ v; alloc} \rangle \xrightarrow{2} \langle \mathsf{H}, \ell \mapsto \mathsf{v} \fatsemi \mathsf{S}, \ell \fatsemi \cdot \rangle$, for fresh ℓ. To complete the proof, we choose W'' to be W' extended with ℓ mapping to $\mathcal{V}^{\mathbf{S}}[\![\tau^+]\!]$, appropriately restricted, and $\varphi' = \{\ell\}$, which means $(W'', \varphi', \ell) \in \mathcal{V}^{\mathbf{S}}[\![\mathbf{ref}\ \tau^+]\!]$ as needed. □

Lemma 0.5.47 (read sound $\mathbf{S}$)**.**
$\forall W\ \tau.\ (W, \emptyset, \mathsf{thunk\ push\ (thunk\ lam\ fread.lam\ l.push\ l; read); fix}) \in \mathcal{V}^{\mathbf{S}}[\![(\mathbf{ref}\ \tau^+) \xrightarrow{\bullet} \tau^+]\!]$

Proof. It suffices to show:

$$\begin{array}{l} \forall \ell\ W' \sqsupset W.\ell \in \mathrm{dom}(W'.\Psi) \wedge (W', \{\ell\}, \ell) \in \mathcal{V}^{\mathbf{S}}[\![\mathbf{ref}\ \tau^+]\!] \\ \quad \Longrightarrow\ (W', \{\ell\}, \mathsf{push}\ \ell; \mathsf{read}) \in \mathcal{E}^{\mathbf{S}}[\![\tau^+]\!] \end{array}$$

Thus, we choose a $\mathsf{H} :_{\{\ell\}} W'$, S, and if $W'.\Psi(\ell) \neq \dagger$, take two steps $\langle \mathsf{H} \fatsemi \mathsf{S} \fatsemi \mathsf{push}\ \ell; \mathsf{read} \rangle \xrightarrow{2} \langle \mathsf{H} \fatsemi \mathsf{S}, \mathsf{v} \fatsemi \cdot \rangle$, which from the invariant on the heap, $(\triangleright W', \emptyset, \mathsf{v}) \in \mathcal{V}^{\mathbf{S}}[\![\tau^+]\!]$ as needed. If $W'.\Psi(\ell) = \dagger$, then the location has been freed and we will reduce to fail MEM, which is also in the relation.

□

Lemma 0.5.48 (write sound $\mathbf{S}$)**.**
$\forall W\ \tau.\ (W, \emptyset, \mathsf{thunk\ push\ (thunk\ lam\ fwrite.lam\ x.lam\ l.push\ l; push\ x; write; push\ 0); fix}) \in \mathcal{V}^{\mathbf{S}}[\![(\mathbf{ref}\ \tau^+, \tau^+) \xrightarrow{\bullet} \mathtt{unit}]\!]$

Proof. It suffices to show:

$$\forall \ell\ \mathsf{v}\ W' \sqsupset W.\ \{\ell\} \cup \varphi \subset \mathrm{dom}(W'.\Psi) \wedge (W', \{\ell\}, \ell) \in \mathcal{V}^{\mathbf{S}}[\![\mathbf{ref}\ \tau^+]\!] \wedge (W', \varphi, \mathsf{v}) \in \mathcal{V}^{\mathbf{S}}[\![\tau^+]\!]$$
$$\implies (W', \{\ell\} \cup \varphi, \mathsf{push}\ \ell; \mathsf{push}\ \mathsf{v}; \mathsf{write}; \mathsf{push}\ 0) \in \mathcal{E}^{\mathbf{S}}[\![\mathtt{unit}]\!]$$

Thus, we choose a $\mathsf{H} :_{\{\ell\}\cup\varphi} W'$, S, and, if $W'.\Psi(\ell) \neq \dagger$, take four steps:

$$\langle \mathsf{H} \fatsemi \mathsf{S} \fatsemi \mathsf{push}\ \ell; \mathsf{push}\ \mathsf{v}; \mathsf{write}; \mathsf{push}\ 0\rangle \to$$
$$\langle \mathsf{H} \fatsemi \mathsf{S}, \ell \fatsemi \mathsf{push}\ \mathsf{v}; \mathsf{write}; \mathsf{push}\ 0\rangle \to$$
$$\langle \mathsf{H} \fatsemi \mathsf{S}, \ell, \mathsf{v} \fatsemi \mathsf{write}; \mathsf{push}\ 0\rangle \to$$
$$\langle \mathsf{H}[\ell \mapsto \mathsf{v}] \fatsemi \mathsf{S}, \ell, \mathsf{v} \fatsemi \mathsf{push}\ 0\rangle \to$$
$$\langle \mathsf{H}[\ell \mapsto \mathsf{v}] \fatsemi \mathsf{S}, 0 \fatsemi \cdot\rangle$$

Note that the third step succeeds because the invariant on the heap means that ℓ is bound in it. To complete the proof, it suffices to choose φ' as $\emptyset$, W'' as an extension that simply decreases the step index, since $(W'', \emptyset, 0) \in \mathcal{V}^{\mathbf{S}}[\![\mathtt{unit}]\!]$. We know $\mathsf{H}[\ell \mapsto \mathsf{v}] :_{\{\ell\}\cup\varphi} W''$ and that W'' is an extension, since the value we updated the location with had the same type as what was at ℓ. If $W'.\Psi(\ell) = \dagger$, then the location has been freed and we will reduce to fail MEM, which is also in the relation.

□

0.5.10 *Finally, proving soundness*

With all of the compatibility lemmas proved, we can prove the fundamental property of the logical relation:

Theorem 0.5.49 (fundamental property)**.**
If $\mathtt{I}; \Gamma \vdash \mathtt{e} : \tau$ *then* $[\![\mathtt{I}; \Gamma \vdash \mathtt{e}^+ : \tau]\!]$.

Proof. We prove this by induction over the typing derivation, using a corresponding compatibility lemma for each typing rule. Note that when we cross the boundary, we will switch to using compatibility lemmas for the corresponding extension. □

With that, we can prove type soundness. Note that this references the exception extension covered later, as we close with libraries that could reference it.

Corollary 0.5.50 (type soundness)**.** *If* $\mathtt{I}; \cdot \vdash \mathtt{e} : \tau$ *then given libraries* $\gamma^{\mathtt{I}^{\mathbf{S}}}$ *(where* $((k, \emptyset), \emptyset, \gamma^{\mathtt{I}^{\mathbf{S}}}) \in \mathcal{G}^{\mathbf{S}}[\![\mathtt{I}^{\mathbf{S}}]\!]$*) and* $\gamma^{\mathtt{I}^{\mathbf{X}}}$ *(where* $((k, \emptyset), \emptyset, \gamma^{\mathtt{I}^{\mathbf{X}}}) \in \mathcal{G}^{\mathbf{X}}[\![\mathtt{I}^{\mathbf{X}}]\!]$*), for any heap* H*, stack* S*, if* $\langle \mathsf{H} \fatsemi \mathsf{S} \fatsemi \gamma^{\mathtt{I}^{\mathbf{S}}}(\gamma^{\mathtt{I}^{\mathbf{X}}}(\mathtt{e}^+))\rangle \xrightarrow{*} \langle \mathsf{H}' \fatsemi \mathsf{S}' \fatsemi \mathsf{P}'\rangle$ *then one of:*

- $\mathsf{P}' = \cdot$ *and* $\mathsf{S}' = \mathsf{Fail}\ \mathsf{c}$ *and* $c \in$ OKERR

- $\mathsf{P}' = \cdot$ *and* $\mathsf{S}' = \mathsf{S}, \mathsf{v}$ *and* $\exists j.\ (j, \mathsf{v}) \in \mathcal{V}^{\lambda}[\![\tau]\!]$
- $\exists \mathsf{H}^*\ \mathsf{S}^*\ \mathsf{P}^*.\ \langle \mathsf{H}' \fatsemi \mathsf{S}' \fatsemi \mathsf{P}' \rangle \to \langle \mathsf{H}^* \fatsemi \mathsf{S}^* \fatsemi \mathsf{P}^* \rangle$

Proof. This is simply a combination of the fundamental property with the definition of $\mathcal{E}^{\lambda}[\![\tau]\!]$. □

$$\text{Extended Type } \tau \quad := \quad \tau \mid \texttt{unit} \mid \texttt{bool} \mid \texttt{int} \mid \tau \times \tau \mid \tau + \tau \mid \mu\alpha.\tau \mid (\tau, \ldots, \tau) \xrightarrow{\blacksquare} \tau \mid \textbf{ref}\ \tau$$

$$\frac{\texttt{x} : \tau \in \Gamma}{\Gamma \vdash_{\mathbf{X}} \texttt{x} : \tau} \qquad \frac{\Gamma, \texttt{f} : (\tau_1, \ldots, \tau_n) \xrightarrow{\blacksquare} \tau', \texttt{x}_i : \tau_i \vdash_{\mathbf{X}} \texttt{e} : \tau'}{\Gamma \vdash_{\mathbf{X}} \texttt{fun f}(\texttt{x}_1 : \tau_1, \ldots, \texttt{x}_n : \tau_n)\{\texttt{e}\} : (\tau_1, \ldots, \tau_n) \xrightarrow{\blacksquare} \tau'}$$

$$\frac{\Gamma \vdash_{\mathbf{X}} \texttt{e} : (\tau_1, \ldots, \tau_n) \xrightarrow{\blacksquare} \tau' \qquad \Gamma \vdash_{\mathbf{X}} \texttt{e}_i : \tau_i}{\Gamma \vdash_{\mathbf{X}} \texttt{e}(\texttt{e}_1, \ldots, \texttt{e}_n) : \tau'}$$

Figure 0.14: Linking types for exceptions and state

This extension reuses the same core language, FunLang, as in our state case study, and thus the same target, StackLang. We do not reproduce the FunLang static semantics, the operational semantics of StackLang, or the compiler between them.

0.6 SOUNDNESS

0.6.1 *Exception extension model*

0.6.2 *Proving* $\uparrow$ *sound*

Lemma 0.6.1 (lift **X**)**.** $\forall W\ \mathsf{v}.\ (W, \emptyset, \mathsf{v}) \in \mathcal{V}^{\mathbf{X}}[\![\uparrow\tau]\!] \iff (W.k, \mathsf{v}) \in \mathcal{V}^{\lambda}[\![\tau]\!]$

Proof. We prove this by simultaneous induction over the size k and the structure of τ.

Case unit/bool/int. In this case, the values are trivially in the relation, by definition.

Case $\tau_1 \times \tau_2 / \tau_1 + \tau_2$. These follow straightforwardly by appealing to the inductive hypothesis.

Case $\mu\alpha.\tau$. In this case, we can appeal to our inductive hypothesis at a smaller k (as our type may have gotten larger).

Case $(\tau_1, \ldots, \tau_n) \to \tau'$. This follows by application of the induction hypothesis.

□

0.6.3 *Proving* $\lceil\tau\rfloor$ *satisfies* $\downarrow$

Lemma 0.6.2 (wrap closed **X**)**.** $\forall\tau.\ \mathit{fvars}(\lceil\tau\rfloor) = \emptyset$

$\uparrow\tau$	$\triangleq$	τ
$\uparrow\texttt{unit}$	$\triangleq$	$\texttt{unit}$
$\uparrow\texttt{bool}$	$\triangleq$	$\texttt{bool}$
$\uparrow\texttt{int}$	$\triangleq$	$\texttt{int}$
$\uparrow\tau_1 \times \tau_2$	$\triangleq$	$\uparrow\tau_1 \times \uparrow\tau_2$
$\uparrow\tau_1 + \tau_2$	$\triangleq$	$\uparrow\tau_1 + \uparrow\tau_2$
$\uparrow\mu\alpha.\tau$	$\triangleq$	$\mu\alpha.\uparrow\tau$
$\uparrow(\tau_1,\ldots,\tau_n)\to\tau'$	$\triangleq$	$(\uparrow\tau_1,\ldots,\uparrow\tau_n)\xrightarrow{\square}\uparrow\tau'$

$\downarrow\tau$	$\triangleq$	τ
$\downarrow\texttt{unit}$	$\triangleq$	$\texttt{unit}$
$\downarrow\texttt{bool}$	$\triangleq$	$\texttt{bool}$
$\downarrow\texttt{int}$	$\triangleq$	$\texttt{int}$
$\downarrow\tau_1\times\tau_2$	$\triangleq$	$\downarrow\tau_1 \times \downarrow\tau_2$
$\downarrow\tau_1+\tau_2$	$\triangleq$	$\downarrow\tau_1 + \downarrow\tau_2$
$\downarrow\mu\alpha.\tau$	$\triangleq$	$\mu\alpha.\downarrow\tau$
$\downarrow(\tau_1,\ldots,\tau_n) \xrightarrow{\square} \tau'$	$\triangleq$	$(\downarrow\tau_1,\ldots,\downarrow\tau_n)\to\downarrow\tau'$
$\downarrow(\tau_1,\ldots,\tau_n) \xrightarrow{\blacksquare} \tau'$	$\triangleq$	$(\downarrow\tau_1,\ldots,\downarrow\tau_n)\to\texttt{U} + (\downarrow\tau')$
$\downarrow\textbf{ref}\ \tau$	$\triangleq$	$\texttt{unit}$
$\downarrow\tau$	$\triangleq$	$\texttt{U} + \downarrow\tau$

where $\texttt{U} = \mu\alpha.\texttt{unit} + \texttt{int} + (\alpha \times \alpha) + (\alpha + \alpha) + ((\alpha) \to \alpha) + \alpha$

Figure 0.15: Lift and lower functions for exceptions extension

CATCH $\triangleq$ thunk push (thunk lam fcatch.lam f.push f; call; lam res.push [1, res]; reset); fix
THROW $\triangleq$ thunk push (thunk lam fthrow.lam exn.push [0, exn]; shift _ ()); fix

$\wr\textbf{ref}\ \tau\int$ $\triangleq$ free; push [1, 0]; reset
$\wr(\tau_1,\ldots,\tau_n) \xrightarrow{\blacksquare} \tau'\int$ $\triangleq$ DUP; push (thunk lam l.push l; free); getlocs; lam res. push [1, res]; reset
$\wr\tau\int$ $\triangleq$ lam res.push [1, res]; reset where τ not in above

Figure 0.16: Exception target library & boundary enforcement

```
import( alloc : ((int) →□ int) →■ ref ((int) →□ int),
        read : (ref ((int) →□ int)) →■ ((int) →□ int),
        write : (ref ((int) →□ int), ((int) →□ int)) →■ unit,
        catch : (() →■ U) →□ U + U,
        throw : (U) →■ int)
fun fiblist(lst : μα.(int × α) + unit){                          ①
 {let mtbl = alloc(fun f(n : int){−1}) in                        ②
  let mf = fun mutfib(y : int){
   if x = 0 {0}{if x = 1{1}{
    let m = read(mtbl) in
    if m(x) = −1{
     let r = mutfib(x − 1) + mutfib(x − 2) in
     let _ = write(mtbl, fun f(n){if n = x{r}{m(x)}}) in
     r
    }{m(x)}
   } in
  fun mutfiblist(l : μα.(int × α) + unit){
   match unfold l
    x {if fst x < 0 {throw(fold inl ())} {                       ③
     fold inl(mf(fst x), mutfiblist(snd x))}
    y {fold inr()}
   }
 }(lst)}^x
 U + μα.(int × α) + unit
}
```

Figure 0.17: Example: fibonacci with input checks and memoization

$$
\begin{array}{lcl}
\mathcal{V}^{\mathbf{X}}[\![\mathtt{unit}]\!] & = & \{(W,\emptyset,0)\} \\
\mathcal{V}^{\mathbf{X}}[\![\mathtt{bool}]\!] & = & \{(W,\emptyset,\mathsf{n})\} \\
\mathcal{V}^{\mathbf{X}}[\![\mathtt{int}]\!] & = & \{(W,\emptyset,\mathsf{n})\} \\
\mathcal{V}^{\mathbf{X}}[\![\tau_1 \times \tau_2]\!] & = & \{(W,\varphi,[\mathsf{v}_1,\mathsf{v}_2]) \mid \varphi \subset \mathrm{dom}(W.\Psi) \wedge \varphi_1 \cup \varphi_2 = \varphi \wedge \\
 & & \quad (W,\varphi_1,\mathsf{v}_1) \in \mathcal{V}^{\mathbf{X}}[\![\tau_1]\!] \wedge (W,\varphi_2,\mathsf{v}_2) \in \mathcal{V}^{\mathbf{X}}[\![\tau_2]\!]\text{`}\} \\
\mathcal{V}^{\mathbf{X}}[\![\tau_1 + \tau_2]\!] & = & \{(W,\varphi,[0,\mathsf{v}]) \mid \varphi \subset \mathrm{dom}(W.\Psi) \wedge (W,\varphi,\mathsf{v}) \in \mathcal{V}^{\mathbf{X}}[\![\tau_1]\!]\} \\
 & \cup & \{(W,\varphi,[1,\mathsf{v}]) \mid \varphi \subset \mathrm{dom}(W.\Psi) \wedge (W,\varphi,\mathsf{v}) \in \mathcal{V}^{\mathbf{X}}[\![\tau_2]\!]\} \\
\mathcal{V}^{\mathbf{X}}[\![\mu\alpha.\tau]\!] & = & \{(W,\varphi,\mathsf{v}) \mid (W,\varphi,\mathsf{v}) \in \triangleright \mathcal{V}^{\mathbf{X}}[\![\tau[\mu\alpha.\tau/\alpha]]\!]\} \\
\mathcal{V}^{\mathbf{X}}[\![\mathtt{ref}\ \tau]\!] & = & \{(W,\{\ell\},\ell) \mid W.\Psi(\ell) = \lfloor \mathcal{V}^{\mathbf{X}}[\![\tau]\!] \rfloor_{W.k} \mid \dagger\} \\
\mathcal{V}^{\mathbf{X}}[\![(\tau_1,\ldots,\tau_n) \xrightarrow{\square} \tau']\!] & = & \{(W,\emptyset,\mathsf{thunk\ push\ (thunk\ lam\ f.lam\ x_n.\ldots.\ lam\ x_1.P);fix}) \mid \\
 & & \forall \mathsf{v_i}\ \varphi_i\ W' \sqsupseteq W.\ \varphi_i \subset \mathrm{dom}(W'.\Psi) \wedge (W',\varphi_i,\mathsf{v_i}) \in \mathcal{V}^{\mathbf{X}}[\![\tau_i]\!] \\
 & & \implies (W', \bigcup_i \varphi_i, [\mathsf{x_1} \mapsto \mathsf{v_1},\ldots,\mathsf{x_n} \mapsto \mathsf{v_n}, \\
 & & \quad \mathsf{f} \mapsto (\mathsf{thunk\ push\ (thunk\ lam\ f.lam\ x_n.\ldots.\ lam\ x_1.P);fix})]\mathsf{P}) \\
 & & \quad \in \mathcal{E}^{\mathbf{X}}[\![\tau']\!]\} \\
\mathcal{V}^{\mathbf{X}}[\![(\tau_1,\ldots,\tau_n) \xrightarrow{\blacksquare} \tau']\!] & = & \{(W,\varphi,\mathsf{thunk\ push\ (thunk\ lam\ f.lam\ x_n.\ldots.\ lam\ x_1.P);fix}) \mid \\
 & & \varphi \subset \mathrm{dom}(W.\Psi) \wedge \forall \mathsf{v_i}\ \varphi_i\ W' \sqsupseteq W. \\
 & & \varphi_i \subset \mathrm{dom}(W'.\Psi) \wedge (W',\varphi_i,\mathsf{v_i}) \in \mathcal{V}^{\mathbf{X}}[\![\tau_i]\!] \\
 & & \implies (W', \varphi \cup \bigcup_i \varphi_i, [\mathsf{x_1} \mapsto \mathsf{v_1},\ldots,\mathsf{x_n} \mapsto \mathsf{v_n}, \\
 & & \quad \mathsf{f} \mapsto (\mathsf{thunk\ push\ (thunk\ lam\ f.lam\ x_n.\ldots.\ lam\ x_1.P);fix})]\mathsf{P}) \\
 & & \quad \in \mathcal{E}^{\mathbf{X}}[\![\tau']\!]\lightning\}
\end{array}
$$

$$
\begin{array}{l}
\mathsf{K} ::= \mathsf{push\ v_1; push\ v_2; \ldots push\ v_n; [\cdot]; P} \\
\mathcal{E}^{\mathbf{X}}[\![\tau]\!]\lightning = \{(W,\varphi_p,\mathsf{P}) \mid \mathsf{reset} \notin \mathsf{P} \ \wedge\ \forall (W,\varphi_k,K) \in \mathcal{K}[\![\tau \Rightarrow \tau']\!].\ (W,\varphi_p \cup \varphi_k, \mathsf{K[P]}) \in \mathcal{E}^{\mathbf{X}}[\![\tau']\!]\} \\
\mathcal{R}[\![\tau]\!] = \ \{(W,\varphi,\mathsf{push\ v}) \mid (W,\varphi,\mathsf{v}) \in \mathcal{V}^{\mathbf{X}}[\![\tau]\!]\} \\
\qquad \cup \{(W,\varphi_p \cup \varphi_v, \mathsf{push\ [0,v]; shift\ \_\ ();\ P}) \mid (W,\varphi_v,\mathsf{v}) \in \mathcal{V}^{\mathbf{X}}[\![\mathtt{U}]\!] \wedge \mathsf{reset} \notin \mathsf{P}\} \\
\mathcal{K}[\![\tau \Rightarrow \tau']\!] = \{(W,\varphi,\mathsf{K}) \mid \varphi = \mathrm{flocs}(\mathsf{K}) \ \wedge \forall W' \sqsupseteq W, (W',\varphi',P) \in \mathcal{R}[\![\tau]\!].\ \\
\qquad (W',\varphi \cup \varphi', K[P]) \in \mathcal{E}^{\mathbf{X}}[\![\tau']\!]\} \\
\mathcal{E}^{\mathbf{X}}[\![\tau]\!] = \{(W,\varphi,\mathsf{P}) \mid \forall \mathsf{H}{:}_{\varphi} W, \mathsf{S}.\ \mathrm{running}_{W.k}(\langle \mathsf{H} \fatsemi \mathsf{S} \fatsemi P\rangle) \ \vee\ \exists j < W.k, \mathsf{H'}, \mathsf{S'}. \\
\qquad \langle \mathsf{H} \fatsemi \mathsf{S} \fatsemi P\rangle \xrightarrow{*} j\langle \mathsf{H'} \fatsemi \mathsf{S'} \fatsemi \cdot\rangle \wedge ((\mathsf{S'} = \mathsf{Fail\ c} \wedge \mathsf{c} \in \textsc{OkErr}) \\
\qquad \vee\ \exists \mathsf{v}\ \varphi'\ W' \sqsupseteq W.\ \left(\mathsf{S'} = \mathsf{S}, \mathsf{v} \wedge \mathsf{H'} :_{\varphi' \cup \varphi} W' \ \wedge\ (W',\varphi',v) \in \mathcal{V}^{\mathbf{X}}[\![\tau]\!]\right))\}
\end{array}
$$

where $\textsc{OkErr} \triangleq \{\textsc{Mem}\}$

Figure 0.18: Exception extension logical relation: main definition

Proof. This follows by simple inspection of the definition. □

Lemma 0.6.3 (encapsulation continuation)**.** $\forall W\ \tau.\ (W, \emptyset, \wr\tau\int) \in \mathcal{K}[\![\tau \to \uparrow\downarrow\tau]\!]$

Proof. We proceed by case analysis on τ, handling the majority of the cases for which $\wr\tau\int$ is the default, exception catching case first.

In those cases, which by inspection, $\uparrow\downarrow\tau = \mathtt{U} + \tau$, the proof obligation is to show that

$$(W, \emptyset, \mathtt{lam\ res.push\ [1, res]; reset}) \in \mathcal{K}[\![\tau \to \mathtt{U} + \tau]\!]$$

That means:

$$\begin{array}{l}
\forall \mathsf{v}\ \varphi.\ (W, \varphi, \mathsf{v}) \in \mathcal{V}^{\mathbf{X}}[\![\mathtt{U}]\!] \implies \\
\qquad (W, \varphi, \mathsf{push\ [0, v]; shift\ _(); lam\ res.push\ [1, res]; reset}) \in \mathcal{E}^{\mathbf{X}}[\![\mathtt{U} + \tau]\!]_\circ \\
\wedge\ \forall \mathsf{v}\ \varphi.\ (W, \varphi, \mathsf{v}) \in \mathcal{V}^{\mathbf{X}}[\![\tau]\!] \implies \\
\qquad (W, \varphi, \mathsf{push\ v; lam\ res.push\ [1, res]; reset}) \in \mathcal{E}^{\mathbf{X}}[\![\mathtt{U} + \tau]\!]_\circ
\end{array}$$

We consider each case in turn. First, we consider the case when an exception value is raised. We choose an heap $\mathsf{H} :_\varphi W$, and a stack S, and see that the term runs as follows:

$$\begin{array}{l}
\langle \mathsf{H} \fatsemi \mathsf{S} \fatsemi \mathsf{push\ [0, v]; shift\ _(); lam\ res.push\ [1, res]; reset} \rangle \to \\
\langle \mathsf{H} \fatsemi \mathsf{S}, [0, \mathsf{v}] \fatsemi \mathsf{shift\ _(); lam\ res.push\ [1, res]; reset} \rangle \to \\
\langle \mathsf{H} \fatsemi \mathsf{S}, [0, \mathsf{v}] \fatsemi \cdot \rangle
\end{array}$$

At this point, we clearly satisfy the requirements of the expression relation. In the other case, when no exception is raised, we again choose a heap $\mathsf{H} :_\varphi W$ (note this is a differest set of relevant locations!), and run as follows:

$$\begin{array}{l}
\langle \mathsf{H} \fatsemi \mathsf{S} \fatsemi \mathsf{push\ v; lam\ res.push\ [1, res]; reset} \rangle \to \\
\langle \mathsf{H} \fatsemi \mathsf{S}, \mathsf{v} \fatsemi \mathsf{lam\ res.push\ [1, res]; reset} \rangle \to \\
\langle \mathsf{H} \fatsemi \mathsf{S} \fatsemi \mathsf{push\ [1, v]; reset} \rangle \to \\
\langle \mathsf{H} \fatsemi \mathsf{S}, [1, \mathsf{v}] \fatsemi \mathsf{reset} \rangle \to \\
\langle \mathsf{H} \fatsemi \mathsf{S}, [1, \mathsf{v}] \fatsemi \cdot \rangle
\end{array}$$

Again, we satisfy the relation, this time in the other disjunct.

Now, we consider the other two types, which use unique wrapping code:

Case $\mathbf{ref}\ \tau$. Our obligation is to show:

$$\begin{array}{l}
\forall \mathsf{v}\ \varphi.\ (W, \varphi, \mathsf{v}) \in \mathcal{V}^{\mathbf{X}}[\![\mathtt{U}]\!] \\
\qquad \implies (W, \varphi, \mathsf{push\ [0, v]; shift\ _\ (); free; push\ [1, 0]; reset}) \in \mathcal{E}^{\mathbf{X}}[\![\mathtt{U} + \mathtt{unit}]\!]_\circ \\
\wedge\ \forall \mathsf{v}\ \varphi.\ (W, \varphi, \mathsf{v}) \in \mathcal{V}^{\mathbf{X}}[\![\mathbf{ref}\ \tau]\!] \\
\qquad \implies (W, \varphi, \mathsf{push\ v; free; push\ [1, 0]; reset}) \in \mathcal{E}^{\mathbf{X}}[\![\mathtt{U} + \mathtt{unit}]\!]_\circ
\end{array}$$

The first case is identical to our first one, so we only consider the second case. By inspection of $\mathcal{V}^{\mathbf{X}}[\![\mathbf{ref}\ \tau]\!]$, we know for some ℓ, $\varphi = \{\ell\}$, $\mathsf{v} = \ell$, and $W.\Psi(\ell) = \lfloor \mathcal{V}^{\mathbf{X}}[\![\tau]\!] \rfloor_{W.k}$. This means when we choose a heap $\mathsf{H} :_{\varphi} W$, we know it will have ℓ bound to some value in $\triangleright \lfloor \mathcal{V}^{\mathbf{X}}[\![\tau]\!] \rfloor_{W.k}$, though as we will see, the actual value does not matter. We will then take four steps:

$$\begin{array}{l}
\langle \mathsf{H} \fatsemi \mathsf{S} \fatsemi \mathsf{push}\ \ell; \mathsf{free}; \mathsf{push}\ [1,0]; \mathsf{reset} \rangle \to \\
\langle \mathsf{H} \fatsemi \mathsf{S}, \ell \fatsemi \mathsf{free}; \mathsf{push}\ [1,0]; \mathsf{reset} \rangle \to \\
\langle \mathsf{H} \setminus \ell \fatsemi \mathsf{S} \fatsemi \mathsf{push}\ [1,0]; \mathsf{reset} \rangle \to \\
\langle \mathsf{H} \setminus \ell \fatsemi \mathsf{S}, [1,0] \fatsemi \mathsf{reset} \rangle \to \\
\langle \mathsf{H} \setminus \ell \fatsemi \mathsf{S}, [1,0] \fatsemi \cdot \rangle
\end{array}$$

Now we choose W' to be W, but with ℓ updated to be marked as dead, and choose $\varphi' = \emptyset$. Still, $(\mathsf{H} \setminus \ell) :_{\varphi} W'$ (as dead elements in the world are ignored), and by definition, $(W', \emptyset, 0) \in \mathcal{V}^{\mathbf{X}}[\![\mathtt{unit}]\!]$, which, along with the appropriate tag, is sufficient to satisfy $\mathcal{V}^{\mathbf{X}}[\![\mathtt{U} + \mathtt{unit}]\!]$, so we are done with this case.

Case $(\tau_1, \ldots, \tau_n) \xrightarrow{\bullet} \tau'$. Our obligation is to show:

$$\begin{array}{l}
\forall \mathsf{v}\, \varphi.\ (W, \varphi, \mathsf{v}) \in \mathcal{V}^{\mathbf{X}}[\![\mathtt{U}]\!] \\
\quad \Longrightarrow\ (W, \varphi, \mathsf{push}\ [0, \mathsf{v}]; \mathsf{shift}\ _\ (); \mathsf{lam}\ \mathsf{x}.(\mathsf{push}\ \mathsf{x}; \mathsf{push}\ \mathsf{x}); \\
\qquad\qquad \mathsf{push}\ (\mathsf{thunk}\ \mathsf{lam}\ \mathsf{l}.\mathsf{push}\ \mathsf{l}; \mathsf{free}); \mathsf{getlocs}; \mathsf{lam}\ \mathsf{res}.\ \mathsf{push}\ [1, \mathsf{res}]; \mathsf{reset}) \\
\qquad \in \mathcal{E}^{\mathbf{X}}[\![\mathtt{U} + (\tau_1, \ldots, \tau_n) \xrightarrow{\bullet} \tau']\!]_{\circ} \\
\wedge\ \forall \mathsf{v}\, \varphi.\ (W, \varphi, \mathsf{v}) \in \mathcal{V}^{\mathbf{X}}[\![(\tau_1, \ldots, \tau_n) \xrightarrow{\bullet} \tau']\!] \\
\quad \Longrightarrow\ (W, \varphi, \mathsf{push}\ \mathsf{v}; \mathsf{lam}\ \mathsf{x}.(\mathsf{push}\ \mathsf{x}; \mathsf{push}\ \mathsf{x}); \\
\qquad\qquad \mathsf{push}\ (\mathsf{thunk}\ \mathsf{lam}\ \mathsf{l}.\mathsf{push}\ \mathsf{l}; \mathsf{free}); \mathsf{getlocs}; \mathsf{lam}\ \mathsf{res}.\ \mathsf{push}\ [1, \mathsf{res}]; \mathsf{reset}) \\
\qquad \in \mathcal{E}^{\mathbf{X}}[\![\mathtt{U} + (\tau_1, \ldots, \tau_n) \xrightarrow{\bullet} \tau']\!]_{\circ}
\end{array}$$

As before, the first case is identical to previous, so we only consider the second case. Once we pick an arbitrary stack S and a heap $\mathsf{H} :_{\varphi} W$, we take the following steps:

$$\begin{array}{l}
\langle \mathsf{H} \fatsemi \mathsf{S} \fatsemi\ \mathsf{push}\ \mathsf{v}; \mathsf{lam}\ \mathsf{x}.(\mathsf{push}\ \mathsf{x}; \mathsf{push}\ \mathsf{x}); \mathsf{push}\ (\mathsf{thunk}\ \mathsf{lam}\ \mathsf{l}.\mathsf{push}\ \mathsf{l}; \mathsf{free});\ \rangle \xrightarrow{4} \\
\qquad \mathsf{getlocs}; \mathsf{lam}\ \mathsf{res}.\ \mathsf{push}\ [1, \mathsf{res}]; \mathsf{reset} \\
\langle \mathsf{H} \fatsemi \mathsf{S}, \mathsf{v}, \mathsf{v} \fatsemi \mathsf{push}\ (\mathsf{thunk}\ \mathsf{lam}\ \mathsf{l}.\mathsf{push}\ \mathsf{l}; \mathsf{free}); \mathsf{getlocs}; \mathsf{lam}\ \mathsf{res}.\ \mathsf{push}\ [1, \mathsf{res}]; \mathsf{reset} \rangle \to \\
\langle \mathsf{H} \fatsemi \mathsf{S}, \mathsf{v}, \mathsf{v}, (\mathsf{thunk}\ \mathsf{lam}\ \mathsf{l}.\mathsf{push}\ \mathsf{l}; \mathsf{free}) \fatsemi \mathsf{getlocs}; \mathsf{lam}\ \mathsf{res}.\ \mathsf{push}\ [1, \mathsf{res}]; \mathsf{reset} \rangle
\end{array}$$

Now, we know that getlocs will run the thunk on top of the stack once for every free location one position down the stack, which means

everything reachable from our function value. Assume those locations are $\ell_1, \ldots, \ell_k$. Then we step as follows:

$$
\begin{array}{l}
\langle \mathsf{H} \fatsemi \mathsf{S}, \mathsf{v}, \mathsf{v}, (\mathsf{thunk\ lam\ l.push\ l}; \mathsf{free}) \fatsemi \mathsf{getlocs}; \mathsf{lam\ res.\ push}\ [1, \mathsf{res}]; \mathsf{reset} \rangle \overset{3k+1}{\to} \\
\langle \mathsf{H} \setminus \{\ell_1, \ldots, \ell_k\} \fatsemi \mathsf{S}, \mathsf{v} \fatsemi \mathsf{lam\ res.\ push}\ [1, \mathsf{res}]; \mathsf{reset} \rangle \to \\
\langle \mathsf{H} \setminus \{\ell_1, \ldots, \ell_k\} \fatsemi \mathsf{S} \fatsemi \mathsf{push}\ [1, \mathsf{v}]; \mathsf{reset} \rangle \to \\
\langle \mathsf{H} \setminus \{\ell_1, \ldots, \ell_k\} \fatsemi \mathsf{S}, [1, \mathsf{v}] \fatsemi \mathsf{reset} \rangle \to \\
\langle \mathsf{H} \setminus \{\ell_1, \ldots, \ell_k\} \fatsemi \mathsf{S}, [1, \mathsf{v}] \fatsemi \cdot \rangle
\end{array}
$$

Now that we have terminated, we have to fulfill the obligations of $\mathcal{E}^{\mathbf{X}}[\![\mathtt{U} + (\tau_1, \ldots, \tau_n) \overset{\circ}{\to} \tau']\!]$. We choose $\varphi' = \emptyset$, and W' such that every location in φ has been marked dead. By invariant of the relation, $\varphi = \{\ell_1, \ldots, \ell_k\}$. Our heap satisfies the world, by construction, and everything that should be dead is, so the only thing that remains is to show that $(W', \emptyset, [1, \mathsf{v}]) \in \mathcal{V}^{\mathbf{X}}[\![\mathtt{U} + (\tau_1, \ldots, \tau_n) \overset{\circ}{\to} \tau']\!]$. This follows from the definition of $\mathcal{V}^{\mathbf{X}}[\![\tau + \tau]\!]$ and our hypothesis on v, once we substitute our empty relevant location set in.

□

0.6.4 *Proving libraries satisfy types*

We now need to show that our exception library satisfies the proper semantic types. We present the definitions and their intended types first, after which we show the proofs.

$$
\begin{array}{lcl}
\mathsf{CATCH} & : & (() \overset{\blacksquare}{\to} \tau) \overset{\square}{\to} \mathtt{U} + \tau \\
& \triangleq & \mathsf{t{-}p}\ (\mathsf{t{-}l\ fcatch.lam\ f.push\ f}; \mathsf{call}; \mathsf{lam\ res.push}\ [1, \mathsf{res}]; \mathsf{reset}); \mathsf{fix} \\
\mathsf{THROW} & : & (\mathtt{U}) \overset{\blacksquare}{\to} \tau \\
& \triangleq & \mathsf{t{-}p}\ (\mathsf{t{-}l\ fthrow.lam\ exn.push}\ [0, \mathsf{exn}]; \mathsf{shift}\ \_\ ()); \mathsf{fix}
\end{array}
$$

where $\mathsf{t{-}p} = \mathsf{thunk\ push}$ and $\mathsf{t{-}l} = \mathsf{thunk\ lam}$

Lemma 0.6.4 (catch sound $\mathbf{X}$)**.**

$$
\begin{array}{l}
\forall W\, \tau.\ (W, \emptyset,\ \mathsf{thunk\ push\ (thunk\ lam\ fcatch.lam\ f.push\ f}; \mathsf{call}; \\
\qquad\qquad \mathsf{lam\ res.push}\ [1, \mathsf{res}]; \mathsf{reset}); \mathsf{fix}) \\
\qquad \in \mathcal{V}^{\mathbf{X}}[\![(() \overset{\blacksquare}{\to} \tau) \overset{\square}{\to} \mathtt{U} + \tau]\!]
\end{array}
$$

Proof. It suffices to show:

$$\varphi \subset \mathrm{dom}(W.\Psi) \wedge \forall \mathsf{v}\ \varphi\ W' \sqsupseteq W.\ \varphi \subset \mathrm{dom}(W'.\Psi) \wedge (W', \varphi, \mathsf{v}) \in \mathcal{V}^{\mathbf{X}}[\![() \xrightarrow{\blacksquare} \tau]\!]$$
$$\implies (W', \varphi, \mathsf{push\ v; call; lam\ res.push\ [1, res]; reset}) \in \mathcal{E}^{\mathbf{X}}[\![\mathtt{U} + \tau]\!]$$

Thus, we need to consider heap $\mathsf{H} :_{\varphi \cup \varphi^k} W'$, stack S, and after two steps, are running the body of v. From the definition of $\mathcal{V}^{\mathbf{X}}[\![() \xrightarrow{\blacksquare} \tau]\!]$, we know that the body, which has to arguments, is in $\mathcal{E}^{\mathbf{X}}[\![\tau]\!]\lightning$. We instantiate that with $K = [\cdot]; \mathsf{lam\ res.push\ [1, res]; reset}$, and thus it suffices to show that $(W'', \emptyset, K) \in \mathcal{K}[\![\tau \Rightarrow \mathtt{U} + \tau]\!]$. In the case that a normal value is returned, this tags it in with 1 and returns it, satisfying the relation. In the case that an exceptional value is produced, it is already tagged with 0, and immediately reduces to the value, so we are done.

□

Lemma 0.6.5 (throw sound $\mathbf{X}$)**.**
$\forall W\ \tau.\ (W, \emptyset, \mathsf{thunk\ push\ (thunk\ lam\ fthrow.lam\ exn.push\ [0, exn]; shift\ \_\ ()); fix}) \in \mathcal{V}^{\mathbf{X}}[\![(\mathtt{U}) \xrightarrow{\blacksquare} \tau]\!]$

Proof. It suffices to show:

$$\varphi \subset \mathrm{dom}(W.\Psi) \wedge \forall \mathsf{v}\ \varphi\ W' \sqsupseteq W.\ \varphi \subset \mathrm{dom}(W'.\Psi) \wedge (W', \varphi, \mathsf{v}) \in \mathcal{V}^{\mathbf{X}}[\![\mathtt{U}]\!]$$
$$\implies (W', \varphi, \mathsf{push\ [0, v]; shift\ \_\ ()}) \in \mathcal{E}^{\mathbf{X}}[\![\tau]\!]\lightning$$

We choose an arbitrary τ', K, and the result is now immediate from the definition of $\mathcal{K}[\![\tau \Rightarrow \tau']\!]$, since our term is already in the form of the exception result.

□

0.6.5 *Compatibility lemmas & type soundness*

We use the following shorthand for typing rules for the exception notation, which supplements the notation defined already.

$$[\![\Gamma \vdash_{\mathbf{X}} \mathsf{P} : \tau]\!] \equiv \forall W\ \varphi\ \gamma.\ (W, \varphi, \gamma) \in \mathcal{G}^{\mathbf{X}}[\![\Gamma]\!] \implies (W, \mathrm{flocs}(\gamma(\mathsf{P})), \gamma(\mathsf{P})) \in \mathcal{E}^{\mathbf{X}}[\![\tau]\!]\lightning$$

Lemma 0.6.6 (boundary $\mathbf{X}$)**.** $[\![\mathtt{I}^{\mathbf{X}} \uplus \uparrow\Gamma \vdash_{\mathbf{X}} \mathsf{P} : \tau]\!] \implies [\![\mathtt{I}; \Gamma \vdash \mathsf{P}; \wr\tau\int : \downarrow\tau]\!]$

Proof. The general approach of this proof is similar to the one for $\mathbf{S}$ (Lemma 0.5.4); the difference, of course, is that the $\mathbf{X}$ logical relation has a different shape, and so some details are different.

Expanding the goal, we see we need to show:

$$\forall k\,\gamma.\ \forall((k,\emptyset),\emptyset,\gamma^{\mathtt{I}^{\mathbf{S}}}) \in \mathcal{G}^{\mathbf{S}}[\![\mathtt{I}^{\mathbf{S}}]\!].\ \forall((k,\emptyset),\emptyset,\gamma^{\mathtt{I}^{\mathbf{X}}}) \in \mathcal{G}^{\mathbf{X}}[\![\mathtt{I}^{\mathbf{X}}]\!].\ (k,\gamma) \in \mathcal{G}^{\lambda}[\![\Gamma]\!] \implies (k, \gamma^{\mathtt{I}^{\mathbf{X}}}(\gamma^{\mathtt{I}^{\mathbf{S}}}(\gamma(\mathsf{P}; \wr\tau\int)))) \in \mathcal{E}^{\lambda}[\![\downarrow\tau]\!]$$

We note due to Lemma 0.6.2 that $\wr\tau\int$ is closed, so can push the substitution in to only around P.

The hypothesis that we are working with says:

$$\forall W\,\varphi\,\gamma\,(W,\varphi,\gamma) \in \mathcal{G}^{\mathbf{X}}[\![\mathtt{I}^{\mathbf{X}} \uplus \uparrow\Gamma]\!] \implies (W,\varphi,\gamma(\mathsf{P})) \in \mathcal{E}^{\mathbf{X}}[\![\tau]\!]_{\bullet}$$

To instantiate the hypothesis, we need an environment γ' that satisfies $\mathcal{G}^{\mathbf{S}}[\![\mathtt{I}^{\mathbf{X}} \uplus \uparrow\Gamma]\!]$. We argue that it is exactly γ composed with $\gamma^{\mathtt{I}^{\mathbf{X}}}$: we know they are disjoint, and we know the former can be lifted into the latter via Lemma 0.6.1. This means, in particular, that φ is $\emptyset$. Now, we need to choose a continuation and return type $\tau_{\mathbf{A}}$ from $\mathcal{K}[\![\tau \to \tau_{\mathbf{A}}]\!]$. We choose $\wr\tau\int$, with $\tau_{\mathbf{A}}$ set to $\uparrow\downarrow\tau$, which we know, with any world and empty φ, is in the relation from Lemma 0.6.3. This then tells us that $(W,\emptyset,\gamma(\mathsf{P});\wr\tau\int)$ is in $\mathcal{E}^{\mathbf{X}}[\![\uparrow\downarrow\tau]\!]_{\circ}$.

This means we can use the arbitrary heap H we are given initially to instantiate this relation, as our world and set of relevant locations makes no restriction on the heap. We also use the arbitrary stack S we are given. This means that we know that either we run past our step index budget (in which case we are trivially in $\mathcal{E}^{\lambda}[\![\downarrow\tau]\!]\rho$, our overall goal), or after some number of steps we have either run to an acceptable failure state (also okay), or we have terminated in a value v, at a future world W', with relevant locations φ' such that $(W',\varphi',\mathsf{v}) \in \mathcal{V}^{\mathbf{X}}[\![\uparrow\downarrow\tau]\!]$. By inspection of the value relation, we can see for all types $\uparrow\downarrow\tau$, φ' will be $\emptyset$. At this point, the result follows from Lemma 0.6.1. □

0.6.6 *Supporting Lemmas*

Lemma 0.6.7 ($\mathcal{E}^{\mathbf{X}}[\![\tau]\!]$ Embeds $\mathcal{V}^{\mathbf{X}}[\![\tau]\!]$)**.** *If* $(W,\varphi,\mathsf{v}) \in \mathcal{V}^{\mathbf{X}}[\![\tau]\!]$, *then*

$$(W,\varphi,\mathsf{push}\ \mathsf{v}) \in \mathcal{E}^{\mathbf{X}}[\![\tau]\!]$$

Proof. We choose heap $\mathsf{H} :_{\varphi} W$, arbitrary stack S, take a single step and the result is immediate. □

Lemma 0.6.8 ($\mathcal{E}^{\mathbf{X}}[\![\tau]\!]\lightning$ Embeds $\mathcal{E}^{\mathbf{X}}[\![\tau]\!]$)**.** *If* $(W,\varphi,\mathsf{P}) \in \mathcal{E}^{\mathbf{X}}[\![\tau]\!]$, *then*

$$(W,\varphi,\mathsf{P}) \in \mathcal{E}^{\mathbf{X}}[\![\tau]\!]\lightning$$

Proof. Our obligation is to show that for arbitrary τ', K, where $(W,\varphi^k,K) \in \mathcal{K}[\![\tau \Rightarrow \tau']\!]$, $(W,\varphi \cup \varphi^k, K[P]) \in \mathcal{E}^{\mathbf{X}}[\![\tau]\!]$. We do this by appealing to our

hypothesis, as we know that if we do not run forever, or result in an acceptable error, we will reduce to a final value on the stack. In that case, we simply appeal to the first case of $\mathcal{R}[\![\tau]\!]$ and we are done. □

Lemma 0.6.9 (Monotonicity $\mathbf{X}$)**.** *If* $(W, \varphi, \mathsf{v}) \in \mathcal{V}^{\mathbf{X}}[\![\tau]\!]$ *and* $W' \sqsupseteq W$*, then* $(W', \varphi, \mathsf{v}) \in \mathcal{V}^{\mathbf{X}}[\![\tau]\!]$

Proof. This follows from the definition of world extension: step indices can decrease, which can only have the effect of bringing more terms into the relation, in the case that we run out of steps before we can rule our membership, and the heap typing can expand or mark existing locations as dead, neither of which rules out existing values being in the relation. □

Lemma 0.6.10 (Antireduction$_\circ$ $\mathbf{X}$)**.**
If $\forall W'\, \varphi'\, \mathsf{H}\, \mathsf{H}', \mathsf{S}.\ (W', \varphi', \mathsf{push}\ \mathsf{v}_1; \mathsf{push}\ \mathsf{v}_2; \ldots \mathsf{push}\ \mathsf{v}_n; \mathsf{P}) \in \mathcal{E}^{\mathbf{X}}[\![\tau]\!]_\circ$ *and* $W' \sqsubseteq W$, $\mathsf{H} :_\varphi W$, $\mathsf{H}' :_{\varphi \cup \varphi'} W'$, *and* $\langle \mathsf{H} \fatsemi \mathsf{S} \fatsemi \mathsf{P}'; \mathsf{P} \rangle \xrightarrow{*} \langle \mathsf{H}' \fatsemi \mathsf{S}, \mathsf{v}_1, \mathsf{v}_2, \ldots, \mathsf{v}_n \fatsemi \mathsf{P} \rangle$ *then* $(W, \varphi, \mathsf{P}'; \mathsf{P}) \in \mathcal{E}^{\mathbf{X}}[\![\tau]\!]_\circ$.

Proof. We consider heap $\mathsf{H} :_\varphi W$, arbitrary stack S. We know that if the term in question does not run forever (which, if it does, then the suffix P does as well, so we are done), then it steps to a terminal configuration $\langle \mathsf{H}^F \fatsemi \mathsf{S}^F \fatsemi \cdot \rangle$. We need to show that, assuming that is not an error, $\mathsf{S}^F = \mathsf{S}, \mathsf{v}$ and for some φ^F and $W^F \sqsubseteq W$, $\mathsf{H}^F :_{\varphi^F} W^F$ and $(W^F, \varphi^F, \mathsf{v}) \in \mathcal{V}^{\mathbf{X}}[\![\tau]\!]_\circ$. We know that $\langle \mathsf{H} \fatsemi \mathsf{S} \fatsemi \mathsf{P}'; \mathsf{P} \rangle \xrightarrow{*} \langle \mathsf{H}' \fatsemi \mathsf{S}, \mathsf{v}_1, \ldots, \mathsf{v}_n \fatsemi \mathsf{P} \rangle$ and that for some $W' \sqsubseteq W$, $\mathsf{H}' :_{\varphi \cup \varphi'} W'$. So we instantiate our first hypothesis with H' and S. After n steps, it is in exactly the configuration our term left off in. We know it doesn't run forever, and if it errors, similarly, our overall term must error, so we conclude that it runs to a terminal configuration which due to confluence, will be the same one. Thus, we know $\mathsf{H}^F :_{\varphi \cup \varphi' \cup \varphi^F} W^F$, which is stronger than we need, and $(W^F, \varphi^F, \mathsf{v}) \in \mathcal{V}^{\mathbf{X}}[\![\tau]\!]$, exactly as needed. □

Lemma 0.6.11 (Monadic Bind $\mathbf{X}$)**.** *If* $(W, \varphi_p, \mathsf{P}) \in \mathcal{E}^{\mathbf{X}}[\![\tau]\!]_\bullet$*, and* $(W', \varphi_k \cup \varphi'_p, \mathsf{K}[\mathsf{P}']) \in \mathcal{E}^{\mathbf{X}}[\![\tau']\!]_\bullet$ *whenever* $(W', \varphi'_p, \mathsf{P}') \in \mathcal{R}[\![\tau]\!]$ *and* $W' \sqsupseteq W$*, then* $(W, \varphi_k \cup \varphi_p, \mathsf{K}[\mathsf{P}]) \in \mathcal{E}^{\mathbf{X}}[\![\tau']\!]_\bullet$.

Proof. Given $(W, \varphi'_k, \mathsf{K}') \in \mathcal{K}[\![\tau' \Rightarrow \tau'']\!]$, we must show $(W, \varphi'_k \cup \varphi_k \cup \varphi_p, \mathsf{K}'[\mathsf{K}[\mathsf{P}]]) \in \mathcal{E}^{\mathbf{X}}[\![\tau'']\!]_\circ$. Because $(W, \varphi_p, \mathsf{P}) \in \mathcal{E}^{\mathbf{X}}[\![\tau]\!]_\bullet$, it suffices if $(W, \varphi_k \cup \varphi'_k, \mathsf{K}'[\mathsf{K}]) \in \mathcal{K}[\![\tau \Rightarrow \tau'']\!]$. Given $(W', \varphi'_p, \mathsf{P}') \in \mathcal{R}[\![\tau]\!]$ where $W' \sqsupseteq W$, we must show $(W', \varphi'_k \cup \varphi_k \cup \varphi'_p, \mathsf{K}'[\mathsf{K}[\mathsf{P}']]) \in \mathcal{E}^{\mathbf{X}}[\![\tau'']\!]_\circ$. By assumption, $(W', \varphi_k \cup \varphi'_p, \mathsf{K}[\mathsf{P}']) \in \mathcal{E}^{\mathbf{X}}[\![\tau']\!]_\bullet$, so $(W', \varphi'_k \cup \varphi'_k \cup \varphi'_p, \mathsf{K}'[\mathsf{K}[\mathsf{P}']]) \in \mathcal{E}^{\mathbf{X}}[\![\tau'']\!]_\circ$ by definition of $\mathcal{E}^{\mathbf{X}}[\![\tau']\!]_\bullet$. □

Corollary 0.6.12 (Antireduction$_\bullet$ $\mathbf{X}$)**.**
If $\forall W'\, \varphi'\, \mathsf{H}\, \mathsf{H}', \mathsf{S}.\ (W', \varphi', \mathsf{push}\ \mathsf{v}_1; \mathsf{push}\ \mathsf{v}_2; \ldots \mathsf{push}\ \mathsf{v}_n; \mathsf{P}) \in \mathcal{E}^{\mathbf{X}}[\![\tau]\!]_\bullet$ *and* $W' \sqsubseteq W$, $\mathsf{H} :_\varphi W$, $\mathsf{H}' :_{\varphi \cup \varphi'} W'$, *and* $\langle \mathsf{H} \fatsemi \mathsf{S} \fatsemi \mathsf{P}'; \mathsf{P} \rangle \xrightarrow{*} \langle \mathsf{H}' \fatsemi \mathsf{S}, \mathsf{v}_1, \mathsf{v}_2, \ldots, \mathsf{v}_n \fatsemi \mathsf{P} \rangle$ *then* $(W, \varphi, \mathsf{P}'; \mathsf{P}) \in \mathcal{E}^{\mathbf{X}}[\![\tau]\!]_\bullet$.

Proof. In Lemma 0.6.10, the only cases are divergence, (type-sound) termination, and failure. Here, we must also consider exceptions, but we can use Lemma 0.6.11 as needed. Otherwise, the proof proceeds as in Lemma 0.6.10. □

Lemma 0.6.13 (Thread X)**.** *If* $(W, \varphi_p, \mathsf{P}) \in \mathcal{E}^{\mathbf{X}}[\![\tau]\!]_\bullet$*, and* $(W', \varphi_k \cup \varphi_v, \mathsf{K}[\mathsf{push}\ \mathsf{v}]) \in \mathcal{E}^{\mathbf{X}}[\![\tau']\!]_\bullet$ *whenever* $(W', \varphi_v, \mathsf{v}) \in \mathcal{V}^{\mathbf{X}}[\![\tau]\!]$ *and* $W' \sqsupseteq W$*, then* $(W, \varphi_k \cup \varphi_p, \mathsf{K}[\mathsf{P}]) \in \mathcal{E}^{\mathbf{X}}[\![\tau']\!]_\bullet$.

Proof. By Lemma 0.6.11, it suffices if $(W', \varphi_k \cup \varphi'_p, \mathsf{K}[\mathsf{P}']) \in \mathcal{E}^{\mathbf{X}}[\![\tau']\!]_\bullet$ given $(W', \varphi'_p, \mathsf{P}') \in \mathcal{R}[\![\tau]\!]$ where $W' \sqsupseteq W$. Unfolding $\mathcal{R}[\![\tau]\!]$, there are two cases.

- $\mathsf{P}' = \mathsf{push}\ \mathsf{v}$ for $(W', \varphi'_p, \mathsf{v}) \in \mathcal{V}^{\mathbf{X}}[\![\tau]\!]$. Then apply the second premise.
- $\mathsf{P}' = \mathsf{push}\ [0, \mathsf{v}];\ \mathsf{shift}\ \_\ ();\mathsf{P}''$ for $(W', \varphi_v, \mathsf{v}) \in \mathcal{V}^{\mathbf{X}}[\![\mathtt{U}]\!]$, $\varphi_v \subseteq \varphi'_p$. Given $(W', \varphi'_k, \mathsf{K}') \in \mathcal{K}[\![\tau' \Rightarrow \tau'']\!]$, we must show $(W', \varphi'_k \cup \varphi_k \cup \varphi'_p, \mathsf{K}'[\mathsf{K}[\mathsf{push}\ [0, \mathsf{v}];\ \mathsf{shift}\ \_\ ();\mathsf{P}'']]) \in \mathcal{E}^{\mathbf{X}}[\![\tau'']\!]_\circ$. Since $\mathsf{K} = \overline{\mathsf{push}\ \mathsf{v}_\mathsf{k}};[\cdot];\mathsf{P}_\mathsf{k}$, we must show $(W', \varphi'_k \cup \varphi_k \cup \varphi'_p, \mathsf{K}'[\overline{\mathsf{push}\ \mathsf{v}};\mathsf{push}\ [0, \mathsf{v}];\ \mathsf{shift}\ \_\ ();\mathsf{P}'';\mathsf{P}_\mathsf{k}]) \in \mathcal{E}^{\mathbf{X}}[\![\tau'']\!]_\circ$. Applying Lemma 0.6.10, it suffices if $(W'', \varphi'_k \cup \varphi_k \cup \varphi'_p, \mathsf{K}'[\mathsf{push}\ [0, \mathsf{v}];\ \mathsf{shift}\ \_\ ();\mathsf{P}'';\mathsf{P}_\mathsf{k}]) \in \mathcal{E}^{\mathbf{X}}[\![\tau'']\!]_\circ$. But notice that $(W'', \varphi_k \cup \varphi'_p, \mathsf{push}\ [0, \mathsf{v}];\ \mathsf{shift}\ \_\ ();\mathsf{P}'';\mathsf{P}_\mathsf{k}) \in \mathcal{R}[\![\tau']\!]$, so applying the definition of $\mathcal{K}[\![\tau' \Rightarrow \tau'']\!]$ is sufficient.

□

0.6.7 FunLang *with* X *Compatibility Lemmas*

$$[\![\Gamma \vdash_{\mathbf{X}} \mathsf{P} : \tau]\!] \equiv \forall W\, \varphi\, \gamma.\ (W, \varphi, \gamma) \in \mathcal{G}^{\mathbf{X}}[\![\Gamma]\!] \implies (W, \mathrm{flocs}(\gamma(\mathsf{P})), \gamma(\mathsf{P})) \in \mathcal{E}^{\mathbf{X}}[\![\tau]\!]_\lightning$$

Lemma 0.6.14 (unit)**.** *Show that* $[\![\Gamma \vdash_{\mathbf{X}} \mathsf{push}\ 0 : \mathtt{unit}]\!]$.

Proof. We are given $(W, \varphi^\dagger, \gamma) \in \mathcal{G}^{\mathbf{X}}[\![\gamma]\!]$, where $\varphi = \mathrm{flocs}(\gamma(\mathsf{push}\ 0)) = \emptyset$, and need to show that $(W, \emptyset, \gamma(\mathsf{push}\ 0)) \in \mathcal{E}^{\mathbf{X}}[\![\mathtt{unit}]\!]_\lightning$.

Thus, we consider arbitrary continuation K with $(W, \varphi^k, K) \in \mathcal{K}[\![\mathtt{unit} \Rightarrow \tau]\!]$. We need to show that $(W, \varphi^k, \mathsf{K}[\gamma(\mathsf{push}\ 0)]) \in \mathcal{E}^{\mathbf{X}}[\![\tau]\!]$. But this follows exactly from the definition of $\mathcal{R}[\![\tau]\!]$. □

Lemma 0.6.15 (bool)**.** *Show for any* n*,* $[\![\Gamma \vdash_{\mathbf{X}} \mathsf{n} : \mathtt{bool}]\!]$.

Proof. This proof is essentially identical to that of `unit`. □

Lemma 0.6.16 (if)**.** *If* $[\![\Gamma \vdash_{\mathbf{X}} \mathsf{P}_1 : \mathtt{bool}]\!]$, $[\![\Gamma \vdash_{\mathbf{X}} \mathsf{P}_2 : \tau]\!]$*, and* $[\![\Gamma \vdash_{\mathbf{X}} \mathsf{P}_3 : \tau]\!]$ *then*

$[\![\Gamma \vdash_{\mathbf{X}} \mathsf{P}_1;\mathsf{if0}\ \mathsf{P}_2\ \mathsf{P}_3 : \tau]\!]$.

Proof. Unfolding $[\![\cdot]\!]$ and pushing substitutions, we must show

$$(W, \varphi, \gamma(\mathsf{P}_1);\mathsf{if0}\ \gamma(\mathsf{P}_2)\ \gamma(\mathsf{P}_3)) \in \mathcal{E}^{\mathbf{X}}[\![\tau]\!]_\lightning$$

given $(W, \varphi^\dagger, \gamma) \in \mathcal{G}^{\mathbf{X}}[\![\Gamma]\!]$ where $\varphi = \bigcup \varphi_i$ and $\varphi_i = \mathrm{flocs}(\gamma(\mathsf{P_i}))$. Applying Lemma 0.6.13 with the first premise, it suffices if

$$(W', \varphi_2 \cup \varphi_3, \mathsf{push\ n; if0\ } \gamma(\mathsf{P_2})\ \gamma(\mathsf{P_3})) \in \mathcal{E}^{\mathbf{X}}[\![\tau]\!]_{\lightning}$$

given $(W', \emptyset, \mathsf{n}) \in \mathcal{V}^{\mathbf{X}}[\![\mathtt{bool}]\!]$ and $W' \sqsupseteq W$. There are two cases.

- Suppose $\mathsf{n} = 0$. Then by Lemma 0.6.12, it suffices if

  $$(W', \varphi_2, \gamma(\mathsf{P_2})) \in \mathcal{E}^{\mathbf{X}}[\![\tau]\!]_{\lightning}$$

  Applying Lemma 0.6.13 with the second premise, it suffices if

  $$(W'', \varphi', \mathsf{push\ v}) \in \mathcal{E}^{\mathbf{X}}[\![\tau]\!]_{\lightning}$$

  where $(W'', \varphi', \mathsf{v}) \in \mathcal{V}^{\mathbf{X}}[\![\tau]\!]$, $W'' \sqsupseteq W'$. Then apply Lemmas 0.6.7, 0.6.8.

- Suppose $\mathsf{n} \neq 0$. Then by Lemma 0.6.12, it suffices if

  $$(W', \varphi_3, \gamma(\mathsf{P_3})) \in \mathcal{E}^{\mathbf{X}}[\![\tau]\!]_{\lightning}$$

  Applying Lemma 0.6.13 with the third premise, it suffices if

  $$(W'', \varphi', \mathsf{push\ v}) \in \mathcal{E}^{\mathbf{X}}[\![\tau]\!]_{\lightning}$$

  where $(W'', \varphi', \mathsf{v}) \in \mathcal{V}^{\mathbf{X}}[\![\tau]\!]$, $W'' \sqsupseteq W'$. Then apply Lemmas 0.6.7, 0.6.8.

□

Lemma 0.6.17 (int)**.** *For any* n*, show* $[\![\Gamma \vdash_{\mathbf{X}} \mathsf{push\ n} : \mathtt{int}]\!]$.

Proof. This proof is essentially identical to that of `unit`. □

Lemma 0.6.18 (op-=)**.** *If* $[\![\Gamma \vdash_{\mathbf{X}} \mathsf{P_1} : \mathtt{int}]\!]$ *and* $[\![\Gamma \vdash_{\mathbf{X}} \mathsf{P_2} : \mathtt{int}]\!]$*, then* $[\![\Gamma \vdash_{\mathbf{X}} \mathsf{P_1; P_2; equal?} : \mathtt{bool}]\!]$.

Proof. Unfolding $[\![\cdot]\!]$ and pushing substitutions, we are to show

$$(W, \varphi, \gamma(\mathsf{P_1}); \gamma(\mathsf{P_2}); \mathsf{equal?}) \in \mathcal{E}^{\mathbf{X}}[\![\mathtt{bool}]\!]_{\lightning}$$

given $(W, \varphi^\dagger, \gamma) \in \mathcal{G}^{\mathbf{X}}[\![\Gamma]\!]$ where $\varphi = \bigcup \varphi_i$ and $\varphi_i = \mathrm{flocs}(\gamma(\mathsf{P_i}))$. Applying Lemma 0.6.13 twice, it suffices if

$$(W', \emptyset, \mathsf{push\ n_1; push\ n_2; equal?}) \in \mathcal{E}^{\mathbf{X}}[\![\mathtt{bool}]\!]_{\lightning}$$

given $(W', \emptyset, \mathsf{n_i}) \in \mathcal{V}^{\mathbf{X}}[\![\mathtt{int}]\!]$ and $W' \sqsupseteq W$. Applying Lemma 0.6.12, there are two cases:

- Suppose $\mathsf{n}_1 = \mathsf{n}_2$. Then we must show

$$(W', \emptyset, \mathsf{push}\ 0) \in \mathcal{E}^{\mathbf{X}}[\![\mathtt{bool}]\!]_{\lightning}$$

  which we have by Lemmas 0.6.7, 0.6.8 and the definition of $\mathcal{V}^{\mathbf{X}}[\![\mathtt{bool}]\!]$.

- Suppose $\mathsf{n}_1 \neq \mathsf{n}_2$. Then we must show

$$(W', \emptyset, \mathsf{push}\ 1) \in \mathcal{E}^{\mathbf{X}}[\![\mathtt{bool}]\!]_{\lightning}$$

  which we have by Lemmas 0.6.7, 0.6.8 and the definition of $\mathcal{V}^{\mathbf{X}}[\![\mathtt{bool}]\!]$.

□

Lemma 0.6.19 (op-¡). *If* $[\![\Gamma \vdash_{\mathbf{X}} \mathsf{P}_1 : \mathtt{int}]\!]$ *and* $[\![\Gamma \vdash_{\mathbf{X}} \mathsf{P}_2 : \mathtt{int}]\!]$, *then* $[\![\Gamma \vdash_{\mathbf{X}} \mathsf{P}_1; \mathsf{P}_2; \mathsf{less?} : \mathtt{bool}]\!]$.

Proof. This proof is essentially identical to that of $=$. □

Lemma 0.6.20 (op-+). *If* $[\![\Gamma \vdash_{\mathbf{X}} \mathsf{P}_1 : \mathtt{int}]\!]$ *and* $[\![\Gamma \vdash_{\mathbf{X}} \mathsf{P}_2 : \mathtt{int}]\!]$, *then* $[\![\Gamma \vdash_{\mathbf{X}} \mathsf{P}_1; \mathsf{P}_2; \mathsf{add} : \mathtt{int}]\!]$.

Proof. This proof is essentially identical to that of $=$. □

Lemma 0.6.21 (var). $[\![\Gamma \vdash_{\mathbf{X}} \mathsf{push}\ \mathsf{x} : \tau]\!]$

Proof. Unfolding $[\![\cdot]\!]$ and pushing substitutions, we are to show

$$(W, \varphi, \mathsf{push}\ \mathsf{v}) \in \mathcal{E}^{\mathbf{X}}[\![\tau]\!]_{\lightning}$$

given $(W, \varphi^\dagger \cup \varphi, \gamma[\mathsf{x} \mapsto \mathsf{v}]) \in \mathcal{G}^{\mathbf{X}}[\![\Gamma]\!]$ where $(W, \varphi, \mathsf{v}) \in \mathcal{V}^{\mathbf{X}}[\![\tau]\!]$. Then apply Lemmas 0.6.7, 0.6.8. □

Lemma 0.6.22 (pair). *If* $[\![\Gamma \vdash_{\mathbf{X}} \mathsf{P}_1 : \tau_1]\!]$ *and* $[\![\Gamma \vdash_{\mathbf{X}} \mathsf{P}_2 : \tau_2]\!]$ *then*

$$[\![\Gamma \vdash_{\mathbf{X}} \mathsf{P}_1; \mathsf{P}_2; \mathsf{lam}\ \mathsf{x}_2.\mathsf{lam}\ \mathsf{x}_1.\mathsf{push}\ [\mathsf{x}_1, \mathsf{x}_2] : \tau_1 \times \tau_2]\!]$$

Proof. Unfolding $[\![\cdot]\!]$ and pushing substitutions, we are to show

$$(W, \varphi, \gamma(\mathsf{P}_1); \gamma(\mathsf{P}_2); \mathsf{lam}\ \mathsf{x}_2.\mathsf{lam}\ \mathsf{x}_1.\mathsf{push}\ [\mathsf{x}_1, \mathsf{x}_2]) \in \mathcal{E}^{\mathbf{X}}[\![\tau_1 \times \tau_2]\!]_{\lightning}$$

given $(W, \varphi^\dagger, \gamma) \in \mathcal{G}^{\mathbf{X}}[\![\Gamma]\!]$ where $\varphi = \bigcup \varphi_i$ and $\varphi_i = \mathrm{flocs}(\gamma(\mathsf{P}_i))$.
Applying Lemma 0.6.13 twice, it sufficies if

$$(W', \varphi', \mathsf{push}\ \mathsf{v}_1; \mathsf{push}\ \mathsf{v}_2; \mathsf{lam}\ \mathsf{x}_2.\mathsf{lam}\ \mathsf{x}_1.\mathsf{push}\ [\mathsf{x}_1, \mathsf{x}_2]) \in \mathcal{E}^{\mathbf{X}}[\![\tau_1 \times \tau_2]\!]_{\lightning}$$

given $(W', \varphi_i', \mathsf{v_i}) \in \mathcal{V}^{\mathbf{X}}[\![\tau_i]\!]$ and $\varphi' = \bigcup \varphi_i'$ and $W' \sqsupseteq W$. Applying Lemma 0.6.12, it suffices if

$$(W'', \varphi', \mathsf{push}\ [\mathsf{v_1}, \mathsf{v_2}]) \in \mathcal{E}^{\mathbf{X}}[\![\tau_1 \times \tau_2]\!]_{\lightning}$$

given $W'' \sqsupseteq W'$, which we have by Lemmas 0.6.7, 0.6.8 and the definition of $\mathcal{V}^{\mathbf{X}}[\![\tau_1 \times \tau_2]\!]$. □

Lemma 0.6.23 (fst). *If* $[\![\Gamma \vdash_{\mathbf{X}} \mathsf{P} : \tau_1 \times \tau_2]\!]$, *then* $[\![\Gamma \vdash_{\mathbf{X}} \mathsf{P}; \mathsf{push}\ 0; \mathsf{idx} : \tau_1]\!]$.

Proof. Unfolding $[\![\cdot]\!]$ and pushing substitutions, we are to show

$$(W, \varphi, \gamma(\mathsf{P}); \mathsf{push}\ 0; \mathsf{idx}) \in \mathcal{E}^{\mathbf{X}}[\![\tau_1]\!]_{\lightning}$$

given $(W, \varphi^\dagger, \gamma) \in \mathcal{G}^{\mathbf{X}}[\![\Gamma]\!]$ where $\varphi = \mathrm{flocs}(\gamma(\mathsf{P}))$.
Applying Lemma 0.6.13, it sufficies if

$$(W', \varphi', \mathsf{push}\ [\mathsf{v_1}, \mathsf{v_2}]; \mathsf{push}\ 0; \mathsf{idx}) \in \mathcal{E}^{\mathbf{X}}[\![\tau_1]\!]_{\lightning}$$

where $(W', \varphi_i', \mathsf{v_i}) \in \mathcal{V}^{\mathbf{X}}[\![\tau_i]\!]$ and $\varphi' = \bigcup \varphi_i'$ and $W' \sqsupseteq W$. Applying Lemma 0.6.12, it suffices if

$$(W'', \varphi_1', \mathsf{push}\ \mathsf{v_1}) \in \mathcal{E}^{\mathbf{X}}[\![\tau_1]\!]_{\lightning}$$

where $W'' \sqsupseteq W'$, which we have by Lemmas 0.6.7, 0.6.8. □

Lemma 0.6.24 (snd). *If* $[\![\Gamma \vdash_{\mathbf{X}} \mathsf{P} : \tau_1 \times \tau_2]\!]$, *then* $[\![\Gamma \vdash_{\mathbf{X}} \mathsf{P}_1; \mathsf{push}\ 1; \mathsf{idx} : \tau_2]\!]$.

Proof. As in Lemma 0.6.23. □

Lemma 0.6.25 (inl). *If* $[\![\Gamma \vdash_{\mathbf{X}} \mathsf{P} : \tau_1]\!]$, *then* $[\![\Gamma \vdash_{\mathbf{X}} \mathsf{P}; \mathsf{lam}\ \mathsf{x}.\mathsf{push}\ [0, \mathsf{x}] : \tau_1 + \tau_2]\!]$.

Proof. Unfolding $[\![\cdot]\!]$ and pushing substitutions, we are to show

$$(W, \varphi, \gamma(\mathsf{P}); \mathsf{lam}\ \mathsf{x}.\mathsf{push}\ [0, \mathsf{x}]) \in \mathcal{E}^{\mathbf{X}}[\![\tau_1 + \tau_2]\!]_{\lightning}$$

given $(W, \varphi^\dagger, \gamma) \in \mathcal{G}^{\mathbf{X}}[\![\Gamma]\!]$ where $\varphi = \mathrm{flocs}(\gamma(\mathsf{P}))$.
Applying Lemma 0.6.13, it sufficies if

$$(W', \varphi', \mathsf{push}\ \mathsf{v}; \mathsf{lam}\ \mathsf{x}.\mathsf{push}\ [0, \mathsf{x}]) \in \mathcal{E}^{\mathbf{X}}[\![\tau_1 + \tau_2]\!]_{\lightning}$$

given $(W', \varphi', \mathsf{v}) \in \mathcal{V}^{\mathbf{X}}[\![\tau_1]\!]$ and $W' \sqsupseteq W$. Applying Lemma 0.6.12, it suffices if

$$(W'', \varphi', \mathsf{push}\ [0,\ \mathsf{v}]) \in \mathcal{E}^{\mathbf{X}}[\![\tau_1 + \tau_2]\!]_{\lightning}$$

given $W'' \sqsupseteq W'$, which we have by Lemmas 0.6.7, 0.6.8 and the definition of $\mathcal{V}^{\mathbf{X}}[\![\tau_1 + \tau_2]\!]$. □

Lemma 0.6.26 (inr)**.** *If* $[\![\Gamma \vdash_{\mathbf{X}} \mathsf{P} : \tau_2]\!]$*, then* $[\![\Gamma \vdash_{\mathbf{X}} \mathsf{P}; \mathsf{lam\ x.push}\ [1, \mathsf{x}] : \tau_1 + \tau_2]\!]$.

Proof. As in Lemma 0.6.25. □

Lemma 0.6.27 (match)**.** *If* $[\![\Gamma \vdash_{\mathbf{X}} \mathsf{P}_0 : \tau_1 + \tau_2]\!]$*,* $[\![\Gamma, x : \tau_1 \vdash_{\mathbf{X}} \mathsf{P}_1 : \tau]\!]$*, and* $[\![\Gamma, y : \tau_2 \vdash_{\mathbf{X}} \mathsf{P}_2 : \tau]\!]$*, then*

$$[\![\Gamma \vdash_{\mathbf{X}} \mathsf{P}_0; \mathsf{DUP}; \mathsf{push\ 1}; \mathsf{idx}; \mathsf{SWAP}; \mathsf{push\ 0}; \mathsf{idx}; \mathsf{if0\ (lam\ x.P_1)\ (lam\ y.P_2)} : \tau]\!]$$

Proof. Unfolding $[\![\cdot]\!]$ and pushing substitutions, we must show

$$(W, \varphi, \gamma(\mathsf{P}_0); \mathsf{DUP}; \mathsf{push\ 1}; \mathsf{idx}; \mathsf{SWAP}; \mathsf{push\ 0}; \mathsf{idx}; \mathsf{if0\ (lam\ x.}\gamma(\mathsf{P}_1)\mathsf{)\ (lam\ y.}\ \gamma(\mathsf{P}_2))) \in \mathcal{E}^{\mathbf{X}}[\![\tau]\!]_{\lightning}$$

given $(W, \varphi^\dagger, \gamma) \in \mathcal{G}^{\mathbf{X}}[\![\Gamma]\!]$ where $\varphi = \bigcup \varphi_i$ and $\varphi_i = \mathrm{flocs}(\gamma(\mathsf{P_i}))$. Applying Lemma 0.6.13 with the first premise, it suffices if

$$(W', \varphi', \mathsf{push\ [n, v]}; \mathsf{DUP}; \mathsf{push\ 1}; \mathsf{idx}; \mathsf{SWAP}; \mathsf{push\ 0}; \mathsf{idx}; \mathsf{if0\ (lam\ x.}\gamma(\mathsf{P}_1)\mathsf{)\ (lam\ y.}\ \gamma(\mathsf{P}_2))) \in \mathcal{E}^{\mathbf{X}}[\![\tau]\!]_{\lightning}$$

given $(W', \varphi'_0, [\mathsf{n}, \mathsf{v}]) \in \mathcal{V}^{\mathbf{X}}[\![\tau_1 + \tau_2]\!]$ and $W' \sqsupseteq W$ where $\varphi' = \varphi'_0 \cup \varphi_1 \cup \varphi_2$. There are two cases.

- Suppose $\mathsf{n} = 0$ and $(W', \varphi'_0, \mathsf{v}) \in \mathcal{V}^{\mathbf{X}}[\![\tau_1]\!]$. Then by Lemma 0.6.12 and pushing substitutions, it suffices if

  $$(W', \varphi_1, \gamma[\mathsf{x} \mapsto \mathsf{v}](\mathsf{P}_1)) \in \mathcal{E}^{\mathbf{X}}[\![\tau]\!]_{\lightning}$$

  Applying Lemma 0.6.13 with the second premise, it suffices if

  $$(W'', \varphi'', \mathsf{push\ v}') \in \mathcal{E}^{\mathbf{X}}[\![\tau]\!]_{\lightning}$$

  where $(W'', \varphi'', \mathsf{v}) \in \mathcal{V}^{\mathbf{X}}[\![\tau]\!]$, $W'' \sqsupseteq W'$. Then apply Lemmas 0.6.7, 0.6.8.

- Suppose $\mathsf{n} = 1$ and $(W', \varphi'_0, \mathsf{v}) \in \mathcal{V}^{\mathbf{X}}[\![\tau_2]\!]$. Then by Lemma 0.6.12 and pushing substitutions, it suffices if

  $$(W', \varphi_2, \gamma[\mathsf{y} \mapsto \mathsf{v}](\mathsf{P}_2)) \in \mathcal{E}^{\mathbf{X}}[\![\tau]\!]_{\lightning}$$

  Applying Lemma 0.6.13 with the third premise, it suffices if

  $$(W'', \varphi'', \mathsf{push\ v}') \in \mathcal{E}^{\mathbf{X}}[\![\tau]\!]_{\lightning}$$

where $(W'', \varphi'', \mathsf{v}) \in \mathcal{V}^{\mathbf{X}}[\![\tau]\!]$, $W'' \sqsupseteq W'$. Then apply Lemmas 0.6.7, 0.6.8.

□

Lemma 0.6.28 (fold)**.** *If* $[\![\Gamma \vdash_{\mathbf{X}} \mathsf{P} : \tau[\mu\alpha.\tau/\alpha]]\!]$, *then* $[\![\Gamma \vdash_{\mathbf{X}} \mathsf{P} : \mu\alpha.\tau]\!]$.

Proof. Unfolding $[\![\cdot]\!]$ and pushing substitutions, we are to show

$$(W, \varphi, \gamma(\mathsf{P})) \in \mathcal{E}^{\mathbf{X}}[\![\mu\alpha.\tau]\!]^{\lightning}$$

given $(W, \varphi^{\dagger}, \gamma) \in \mathcal{G}^{\mathbf{X}}[\![\Gamma]\!]$ where $\varphi = \mathrm{flocs}(\gamma(\mathsf{P}))$.
Applying Lemma 0.6.13 with the first premise, it sufficies if

$$(W', \varphi', \mathsf{push}\ \mathsf{v}) \in \mathcal{E}^{\mathbf{X}}[\![\mu\alpha.\tau]\!]^{\lightning}$$

given $(W', \varphi', \mathsf{v}) \in \mathcal{V}^{\mathbf{X}}[\![\tau[\mu\alpha.\tau/\alpha]]\!]$ and $W' \sqsupseteq W$. Applying Lemmas 0.6.7, 0.6.8, it suffices if

$$(W', \varphi', \mathsf{v}) \in \mathcal{V}^{\mathbf{X}}[\![\mu\alpha.\tau]\!]$$

which is immediate from the assumption, the definition of $\mathcal{V}^{\mathbf{X}}[\![\mu\alpha.\tau]\!]$, and Lemma 0.6.9. □

Lemma 0.6.29 (unfold)**.** *If* $[\![\Gamma \vdash_{\mathbf{X}} \mathsf{P} : \mu\alpha.\tau]\!]$, *then* $[\![\Gamma \vdash_{\mathbf{X}} \mathsf{P}; \mathsf{noop} : \tau[\mu\alpha.\tau]]\!]$.

Proof. Unfolding $[\![\cdot]\!]$ and pushing substitutions, we are to show

$$(W, \varphi, \gamma(\mathsf{P}); \mathsf{noop}) \in \mathcal{E}^{\mathbf{X}}[\![\tau[\mu\alpha.\tau]]\!]^{\lightning}$$

given $(W, \varphi^{\dagger}, \gamma) \in \mathcal{G}^{\mathbf{X}}[\![\Gamma]\!]$ where $\varphi = \mathrm{flocs}(\gamma(\mathsf{P}))$.
Applying Lemma 0.6.13 with the first premise, it suffices if

$$(W', \varphi', \mathsf{push}\ \mathsf{v}; \mathsf{noop}) \in \mathcal{E}^{\mathbf{X}}[\![\tau[\mu\alpha.\tau]]\!]^{\lightning}$$

given $(W', \varphi', \mathsf{v}) \in \mathcal{V}^{\mathbf{X}}[\![\mu\alpha.\tau]\!]$ and $W' \sqsupseteq W$. Applying Lemma 0.6.12, it suffices if

$$(W'', \varphi', \mathsf{push}\ \mathsf{v}) \in \mathcal{E}^{\mathbf{X}}[\![\tau[\mu\alpha.\tau]]\!]^{\lightning}$$

given $W'' \sqsupset W'$ (N.B., we take care to strictly advance the world, here). Applying Lemmas 0.6.7, 0.6.8, it suffices if

$$(W'', \varphi', \mathsf{v}) \in \mathcal{V}^{\mathbf{X}}[\![\tau[\mu\alpha.\tau]]\!]$$

which is immediate from the assumption, the definition of $\mathcal{V}^{\mathbf{X}}[\![\mu\alpha.\tau]\!]$, and Lemma 0.6.9. □

Lemma 0.6.30 (fun). *If* $[\![\Gamma, \mathtt{f} : (\tau_1, \ldots, \tau_n) \xrightarrow{\blacksquare} \tau', \mathtt{x_i} : \tau_i \vdash_{\mathbf{X}} \mathsf{P} : \tau']\!]$, *then* $[\![\Gamma \vdash_{\mathbf{X}} \mathsf{push}\ (\mathsf{thunk}\ \mathsf{push}\ (\mathsf{thunk}\ \mathsf{lam}\ \mathsf{f}.\mathsf{lam}\ \mathsf{x_n}.\ldots\ \mathsf{lam}\ \mathsf{x_1}.\mathsf{P}); \mathsf{fix}) : (\tau_1, \ldots, \tau_n) \xrightarrow{\blacksquare} \tau']\!]$

Proof. Unfolding $[\![\cdot]\!]$ and pushing substitutions, we are to show

$$(W, \varphi, \mathsf{push}\ (\mathsf{thunk}\ \mathsf{push}\ (\mathsf{thunk}\ \mathsf{lam}\ \mathsf{f}.\mathsf{lam}\ \mathsf{x_n}.\ldots\ \mathsf{lam}\ \mathsf{x_1}.\gamma(\mathsf{P})); \mathsf{fix})) \in \mathcal{E}^{\mathbf{X}}[\![(\tau_1, \ldots, \tau_n) \xrightarrow{\blacksquare} \tau']\!]_{\lightning}$$

given $(W, \varphi^\dagger, \gamma) \in \mathcal{G}^{\mathbf{X}}[\![\Gamma]\!]$ where $\varphi = \mathrm{flocs}(\gamma(\mathsf{P}))$.
Applying Lemmas 0.6.7, 0.6.8, it suffices if

$$(W, \varphi, \mathsf{thunk}\ \mathsf{push}\ (\mathsf{thunk}\ \mathsf{lam}\ \mathsf{f}.\mathsf{lam}\ \mathsf{x_n}.\ldots\ \mathsf{lam}\ \mathsf{x_1}.\gamma(\mathsf{P})); \mathsf{fix}) \in \mathcal{V}^{\mathbf{X}}[\![(\tau_1, \ldots, \tau_n) \xrightarrow{\blacksquare} \tau']\!]$$

Unfolding the definition of $\mathcal{V}^{\mathbf{X}}[\![(\tau_1, \ldots, \tau_n) \xrightarrow{\blacksquare} \tau']\!]$ and pushing substitutions, we must show

$$(W', \varphi', \gamma[\mathsf{x_i} \mapsto \mathsf{v_i}, \mathsf{f} \mapsto \mathsf{thunk}\ \mathsf{push}\ (\mathsf{thunk}\ \mathsf{lam}\ \mathsf{f}.\mathsf{lam}\ \mathsf{x_n}.\ldots\ \mathsf{lam}\ \mathsf{x_1}.\gamma(\mathsf{P})); \mathsf{fix}](\mathsf{P})) \in \mathcal{E}^{\mathbf{X}}[\![\tau']\!]_{\lightning}$$

given $W' \sqsupset W$ and $(W', \varphi_i, v_i) \in \mathcal{V}^{\mathbf{X}}[\![\tau_i]\!]$ where $\varphi' = \bigcup \varphi_i \cup \varphi \subset W'.\Psi$, which is immediate from the premise. □

Lemma 0.6.31 (app). *If* $[\![\Gamma \vdash_{\mathbf{X}} \mathsf{P}_0 : (\tau_1, \ldots, \tau_n) \xrightarrow{\blacksquare} \tau']\!]$ *and for* $i \in \{1, \ldots, n\}$ $[\![\Gamma \vdash_{\mathbf{X}} \mathsf{P_i} : \tau_i]\!]$ *then*
$[\![\Gamma \vdash_{\mathbf{X}} \mathsf{P}_0; \mathsf{P}_1; \mathsf{SWAP} \ldots \mathsf{P_n}; \mathsf{SWAP}; \mathsf{call} : \tau']\!]$

Proof. Unfolding $[\![\cdot]\!]$ and pushing substitutions, we are to show

$$(W, \varphi, \gamma(\mathsf{P}_0); \gamma(\mathsf{P}_1); \mathsf{SWAP} \ldots \gamma(\mathsf{P_n}); \mathsf{SWAP}; \mathsf{call}) \in \mathcal{E}^{\mathbf{X}}[\![\tau']\!]_{\lightning}$$

given $(W, \varphi^\dagger, \gamma) \in \mathcal{G}^{\mathbf{X}}[\![\Gamma]\!]$ where $\varphi_i = \mathrm{flocs}(P_i)$ and $\varphi = \bigcup P_i$.
Applying Lemmas 0.6.13, 0.6.12 with the premises in order, it suffices if

$$(W', \varphi', \mathsf{push}\ \mathsf{v_1}; \ldots; \mathsf{push}\ \mathsf{v_n}; \mathsf{push}\ (\mathsf{thunk}\ \mathsf{push}\ (\mathsf{thunk}\ \mathsf{lam}\ \mathsf{f}.\mathsf{lam}\ \mathsf{x_n}.\ldots\ \mathsf{lam}\ \mathsf{x_1}.\mathsf{P}); \mathsf{fix}); \mathsf{call}) \in \mathcal{E}^{\mathbf{X}}[\![\tau']\!]_{\lightning}$$

given $W' \sqsupseteq W$, $(W', \varphi'_i, \mathsf{v_i}) \in \mathcal{V}^{\mathbf{X}}[\![\tau_i]\!]$ for $i > 0$, and

$$(W', \varphi'_0, \mathsf{thunk}\ \mathsf{push}\ (\mathsf{thunk}\ \mathsf{lam}\ \mathsf{f}.\mathsf{lam}\ \mathsf{x_n}.\ldots\ \mathsf{lam}\ \mathsf{x_1}.\mathsf{P}); \mathsf{fix}) \in \mathcal{V}^{\mathbf{X}}[\![(\tau_1, \ldots, \tau_n) \xrightarrow{\blacksquare} \tau']\!]$$

where $\varphi' = \bigcup \varphi'_i$. Applying Lemma 0.6.12, it suffices if

$$(W', \varphi', [\mathsf{x_i} \mapsto \mathsf{v_i}, \mathsf{f} \mapsto \mathsf{thunk}\ \mathsf{push}\ (\mathsf{thunk}\ \mathsf{lam}\ \mathsf{f}.\mathsf{lam}\ \mathsf{x_n}.\ldots\ \mathsf{lam}\ \mathsf{x_1}.\mathsf{P}); \mathsf{fix}](\mathsf{P})) \in \mathcal{E}^{\mathbf{X}}[\![\tau']\!]_{\lightning}$$

which is immediate from the definition of $\mathcal{V}^{\mathbf{X}}[\![(\tau_1, \ldots, \tau_n) \xrightarrow{\blacksquare} \tau']\!]$. □

0.6.8 *Finally, soundness*

To account for our new extension, we need to update our existing proof of the fundamantal property for FunLang to include the boundary term for X. The proof itself, of course, still simply dispatches to the appropriate compatibility lemma.

Theorem 0.6.32 (fundamental property)**.** *If* $\mathtt{I}; \Gamma \vdash \mathtt{e} : \tau$ *then* $[\![\mathtt{I}; \Gamma \vdash \mathtt{e}^+ : \tau]\!]$.

Proof. As before, by induction over the typing derivation, using a corresponding compatibility lemma for each typing rule. □

Type soundness again is a corollary, and thus follows from our re-proven fundamental property:

Corollary 0.6.33 (type soundness)**.** *If* $\mathtt{I}; \cdot \vdash \mathtt{e} : \tau$ *then given libraries* $\gamma^{\mathtt{I}^{\mathbf{S}}}$ *(where* $((k, \emptyset), \emptyset, \gamma^{\mathtt{I}^{\mathbf{S}}}) \in \mathcal{G}^{\mathbf{S}}[\![\mathtt{I}^{\mathbf{S}}]\!]$*) and* $\gamma^{\mathtt{I}^{\mathbf{X}}}$ *(where* $((k, \emptyset), \emptyset, \gamma^{\mathtt{I}^{\mathbf{X}}}) \in \mathcal{G}^{\mathbf{X}}[\![\mathtt{I}^{\mathbf{X}}]\!]$*), for any heap* H*, stack* S*, if* $\langle \mathsf{H} ; \mathsf{S} ; \gamma^{\mathtt{I}^{\mathbf{S}}}(\gamma^{\mathtt{I}^{\mathbf{X}}}(\mathtt{e}^+)) \rangle \xrightarrow{*} \langle \mathsf{H}' ; \mathsf{S}' ; \mathsf{P}' \rangle$ *then one of:*

- $\mathsf{P}' = \cdot$ *and* $\mathsf{S}' = \mathsf{Fail}\ \mathsf{c}$ *and* $c \in \textsc{OkErr}$
- $\mathsf{P}' = \cdot$ *and* $\mathsf{S}' = \mathsf{S}, \mathsf{v}$ *and* $\exists j.\ (j, \mathsf{v}) \in \mathcal{V}^{\lambda}[\![\tau]\!]$
- $\exists \mathsf{H}^*\ \mathsf{S}^*\ \mathsf{P}^*.\ \langle \mathsf{H}' ; \mathsf{S}' ; \mathsf{P}' \rangle \to \langle \mathsf{H}^* ; \mathsf{S}^* ; \mathsf{P}^* \rangle$

Proof. This is simply a combination of the fundamental property with the definition of $\mathcal{E}^{\lambda}[\![\tau]\!]$. □

BIBLIOGRAPHY

A. Ahmed, M. Fluet, and G. Morrisett. L3 : A linear language with locations. *Fundamenta Informaticae*, 77(4):397–449, June 2007.

A. J. Ahmed. *Semantics of Types for Mutable State*. PhD thesis, Princeton University, Nov. 2004.

R. Kleffner. A foundation for typed concatenative languages. Master's thesis, Northeastern University, 2017.

P. B. Levy. *Call-by-Push-Value*. Ph. D. dissertation, Queen Mary, University of London, London, UK, Mar. 2001.