

15-312 Assignment 1

Andrew Carnegie
(andrew)

February 26, 2018

1 Introduction

In this paper, we propose a model for deriving asymptotically tight bounds for first order functional programs. We choose a fragment of OCaml as the target language. The abstract and concrete syntax of the language is shown below. Note that we only allow first order functions of type $\tau_1 \rightarrow \tau_2$, where τ_1 and τ_2 are base types: unit, bool, product, or lists.

BTypes	$\tau ::=$		
	nat	nat	naturals
	unit	unit	unit
	bool	bool	boolean
	prod ($\tau_1; \tau_2$)	$\tau_1 \times \tau_2$	product
	list (τ)	$L(\tau)$	list
FTypes	$\rho ::=$		
	arr ($\tau_1; \tau_2$)	$\tau_1 \rightarrow \tau_2$	first order function
Exp	$e ::=$		
	x	x	variable
	nat [n]	\bar{n}	number
	unit	()	unit
	T	T	true
	F	F	false
	if ($x; e_1; e_2$)	if x then e_1 else e_2	if
	lam ($x : \tau.e$)	$\lambda x : \tau.e$	abstraction
	ap ($f; x$)	$f(x)$	application
	tpl ($x_1; x_2$)	$\langle x_1, x_2 \rangle$	pair
	case ($x_1, x_2.e_1$)	case $p \{ (x_1; x_2) \hookrightarrow e_1 \}$	match pair
	nil	\square	nil
	cons ($x_1; x_2$)	$x_1 :: x_2$	cons
	case { l }($e_1; x, xs.e_2$)	case $l \{ \text{nil} \hookrightarrow e_1 \mid \text{cons}(x; xs) \hookrightarrow e_2 \}$	match list
	let ($e_1; x : \tau.e_2$)	let $x = e_1$ in e_2	let
	share ($x; x_1, x_2.e$)	share x as x_1, x_2 in e	share
Val	$v ::=$		
	val (n)	n	numeric value
	val (T)	T	true value
	val (F)	F	false value
	val (Null)	Null	null value
	val (cl ($V; x.e$))	($V, x.e$)	function value
	val (l)	l	loc value
	val (pair ($v_1; v_2$))	$\langle v_1, v_2 \rangle$	pair value
State	$s ::=$		
	alive	alive	live value
	dead	dead	dead value
Loc	$l ::=$		
	loc (l)	l	location
Var	$l ::=$		
	var (x)	x	variable

2 Paths and aliasing

Model dynamics using judgement of the form:

$$\boxed{V, H, R, F \vdash_{P:\Sigma} e \Downarrow v, H', F'}$$

Where $V : \text{Var} \rightarrow \text{Val} \times \text{State}$, $H : \text{Loc} \rightarrow \text{Val}$, $R \subseteq \text{Loc}$, $F \subseteq \text{Loc}$, and $\Sigma : \text{Var} \rightarrow \text{FTypes}$. This can be read as: under stack V , heap H , roots R , freelist F , and program P with signature Σ , the expression e evaluates to v , and engenders a new heap H' and freelist F' .

A *program* is then a Σ indexed map P from Var to pairs $(y_f, e_f)_{f \in \Sigma}$, where $\Sigma(y_f) = A \rightarrow B$, and $\Sigma; y_f : A \vdash e_f : B$ (typing rules are discussed in 7). We write $P : \Sigma$ to mean P is a program with signature Σ . Because the signature Σ for the mapping of function names to first order functions does not change during evaluation, we drop the subscript Σ from \vdash_Σ when the context of evaluation is clear. It is convenient to think of the evaluation judgement \vdash as being indexed by a family of signatures Σ 's, each of which is a set of “top-level” first-order declarations to be used during evaluation.

For a partial map $f : A \rightarrow B$, we write dom for the defined values of f . Sometimes we shorten $x \in \text{dom}(f)$ to $x \in f$. We write $f[x \mapsto y]$ for the extension of f where x is mapped to y , with the constraint that $x \notin \text{dom}(f)$.

Roots represents the set of locations required to compute the continuation *excluding* the current expression. We can think of roots as the heap allocations necessary to compute the context with a hole that will be filled by the current expression.

In order prove soundness of the type system, we need some auxiliary judgements to defining properties of a heap. Below we define $\text{reach} : \text{Val} \rightarrow \{\{\text{Loc}\}\}$ that maps stack values its the root *multiset*, the multiset of locations that's already on the stack.

Next we define reachability of values:

$$\begin{aligned} \text{reach}_H(\langle v_1, v_2 \rangle) &= \text{reach}_H(v_1) \uplus \text{reach}_H(v_2) \\ \text{reach}_H(l) &= \{l\} \uplus \text{reach}_H(H(l)) \\ \text{reach}_H(-) &= \emptyset \end{aligned}$$

For a multiset S , we write $\mu_S : S \rightarrow \mathbb{N}$ for the multiplicity function of S , which maps each element to the count of its occurence. If $\mu_S(x) \geq 1$ for a multiset S , then we write $x \in S$ as in the usual set membership relation. If for all $s \in S$, $\mu(s) = 1$, then S is a property set, and we denote it by $\text{set}(S)$. Additionally, $A \uplus B$ denotes counting union of sets where $\mu_{A \uplus B}(s) = \mu_A(s) + \mu_B(s)$, and $A \cup B$ denotes the usual union where $\mu_{A \cup B}(s) = \max(\mu_A(s), \mu_B(s))$. For the disjoint union of sets A and B , we write $A \sqcup B$.

Next, we define the predicates **no_alias**, **stable**, and **disjoint**:

no_alias(V, H): $\forall x, y \in V, x \neq y. \text{ Let } r_x = \text{reach}_H(V(x)), r_y = \text{reach}_H(V(y)). \text{ Then:}$

1. $\text{set}(r_x), \text{set}(r_y)$
2. $r_x \cap r_y = \emptyset$

stable(R, H, H'): $\forall l \in R. H(l) = H'(l).$

safe(V, H, F): $\forall x \in V. \text{reach}_H(V(x)) \cap F = \emptyset$

disjoint(\mathcal{C}): $\forall X, Y \in \mathcal{C}. X \cap Y = \emptyset$

For a stack V and a heap H , whenever **no_alias**(V, H) holds, visually, one can think of the situation as the following: the induced graph of heap H with variables on the stack as additional leaf nodes is a forest: a disjoint union of arborescences (directed trees); consequently, there is at most one path from a live variable on the stack V to a

location in H by following the pointers.

First, we define $FV^*(e)$, the multiset of free variables of e . As the usual FV , it is defined inductively over the structure of e ; the only unusual thing is that multiple occurrences of a free variable x in e will be reflected in the multiplicity of $FV^*(e)$.

Next, we define $locs_{V,H}$ using the previous notion of reachability.

$$locs_{V,H}(e) = \bigcup_{x \in FV(e)} reach_H(V(x))$$

$size$ calculates the *literal size* of a value, e.g. the size to store its address.

$$\begin{aligned} size(\langle v_1, v_2 \rangle) &= size(v_1) + size(v_2) \\ size(-) &= 1 \end{aligned}$$

$\|\cdot\|_H$ calculates the *semantic* or *heap size* of a value, e.g. the size of the heap structure induced by the value.

$$\begin{aligned} \|\langle v_1, v_2 \rangle\|_H &= \|v_1\|_H + \|v_2\|_H \\ \|l\|_H &= 1 + \|H(l)\|_H \end{aligned}$$

As usual, we extend it to stacks V : $\|V\|_H = \sum_{V(x)=v} \|v\|_H$

$copy(H, L, v)$ takes a heap H , a set of locations L , and a value v , and returns a new heap H' and a location l such that l maps to v in H' .

$$\begin{aligned} copy(H, L, \langle v_1, v_2 \rangle) &= \\ \text{let } L_1 \sqcup L_2 &\subseteq L \\ \text{where } |L_1| &= size(v_1), |L_2| = size(v_2) \\ \text{let } H_1 &= copy(H, L_1, v_1) \\ \text{let } H_2 &= copy(H_1, L_2, v_2) \text{ in} \\ H_2\{l \mapsto v\} & \\ copy(H, L, v) &= \\ \text{let } l \in H &\text{ in} \\ H\{l \mapsto v\} & \end{aligned}$$

3 Garbage collection semantics

$$\frac{V(x) = v}{V, H, R, F \vdash x \Downarrow v, H, F} (S_1) \quad \frac{}{V, H, R, F \vdash \bar{n} \Downarrow \text{val}(n), H, F} (S_2) \quad \frac{}{V, H, R, F \vdash T \Downarrow \text{val}(T), H, F} (S_3)$$

$$\frac{}{V, H, R, F \vdash F \Downarrow \text{val}(F), H, F} (S_4) \quad \frac{}{V, H, R, F \vdash () \Downarrow \text{val}(\text{Null}), H, F} (S_5)$$

$$\frac{V = V'[x \mapsto T] \quad g = \{l \in H \mid l \notin F \cup R \cup \text{locs}_{V,H}(e_1)\} \quad V', H, R, F \cup g \vdash e_1 \Downarrow v, H', F'}{V, H, R, F \vdash \text{if}(x; e_1; e_2) \Downarrow v, H', F'} (S_6)$$

$$\frac{V = V'[x \mapsto F] \quad g = \{l \in H \mid l \notin F \cup R \cup \text{locs}_{V,H}(e_2)\} \quad V', H, R, F \cup g \vdash e_2 \Downarrow v, H', F'}{V, H, R, F \vdash \text{if}(x; e_1; e_2) \Downarrow v, H', F'} (S_7)$$

$$\frac{P(f) = (y_f, e_f) \quad g = \{l \in H \mid l \notin F \cup R \cup \text{locs}_{V,H}(e_f)\} \quad [y_f \mapsto v'], H, R, F \cup g \vdash e_f \Downarrow v, H', F' \quad V(x) = v'}{V, H, R, F \vdash f(x) \Downarrow v, H', F'} (S_8)$$

$$\frac{V(x_1) = v_1 \quad V(x_2) = v_2}{V, H, R, F \vdash \langle x_1, x_2 \rangle \Downarrow \langle v_1, v_2 \rangle, H, F} (S_9)$$

$$\frac{g = \{l \in H \mid l \notin F \cup R \cup \text{locs}_{V,H}(e)\} \quad V = V'[x \mapsto \langle v_1, v_2 \rangle] \quad V'' = V'[x_1 \mapsto v_1, x_2 \mapsto v_2] \quad V'', H, R, F \cup g \vdash e \Downarrow v, H', F'}{V, H, R, F \vdash \text{case } x \{ (x_1; x_2) \hookrightarrow e \} \Downarrow v, H', F'} (S_{10})$$

$$\frac{}{V, H, R, F \vdash \text{nil} \Downarrow \text{val}(\text{Null}), H, F} (S_{11})$$

$$\frac{V = V'[x \mapsto v'] \quad g = \text{reach}_H(v') \quad V', H, R, F \cup g \vdash e \Downarrow v, H', F'}{V, H, R, F \vdash \text{drop}(x; e) \Downarrow v, H', F'} (S_{12})$$

$$\frac{V = V'[x \mapsto v'] \quad V'[x_1 \mapsto v', x_2 \mapsto v'], H, R, F \vdash e \Downarrow v, H', F'}{V, H, R, F \vdash \text{share } x \text{ as } x_1, x_2 \text{ in } e \Downarrow v, H', F'} (S_{13})$$

$$\frac{L \sqcup \{l\} \subseteq F \quad |L| = \text{size}_H(v) \quad v = \langle V(x_1), V(x_2) \rangle \quad F' = F \setminus (L \sqcup \{l\}) \quad H' = \text{copy}(H, L, v) \quad H'' = H' \{l \mapsto v\}}{V, H, R, F \vdash \text{cons}(x_1; x_2) \Downarrow l, H'', F'} (S_{14})$$

$$\frac{V(x) = \text{Null} \quad V' \subseteq V \quad \text{dom}(V') = FV(e_1) \quad g = \{l \in H \mid l \notin F \cup R \cup \text{locs}_{V,H}(e_1)\} \quad V', H, R, F \cup g \vdash e_1 \Downarrow v, H', F'}{V, H, R, F \vdash \text{case } x \{ \text{nil} \hookrightarrow e_1 \mid \text{cons}(x_h; x_t) \hookrightarrow e_2 \} \Downarrow v, H', F'} (S_{15})$$

$$\frac{V(x) = l \quad H(l) = \langle v_h, v_t \rangle \quad V' \subseteq V \quad \text{dom}(V') = FV(e_2) \setminus \{x_h, x_t\} \quad V'' = V'[x_h \mapsto v_h, x_t \mapsto v_t] \quad g = \{l \in H \mid l \notin F \cup R \cup \text{locs}_{V'',H}(e_2)\} \quad V'', H, R, F \cup g \vdash e_2 \Downarrow v, H', F'}{V, H, R, F \vdash \text{case } x \{ \text{nil} \hookrightarrow e_1 \mid \text{cons}(x_h; x_t) \hookrightarrow e_2 \} \Downarrow v, H', F'} (S_{16})$$

$$\frac{V = V_1 \sqcup V_2 \quad \text{dom}(V_1) = FV(e_1) \quad \text{dom}(V_2) = FV(\text{lam}(x : \tau.e_2)) \quad R' = R \cup \text{locs}_{V_2,H}(\text{lam}(x : \tau.e_2)) \quad V_1, H, R', F \vdash e_1 \Downarrow v_1, H_1, F_1 \quad V_2' = V_2[x \mapsto v_1] \quad g = \{l \in H_1 \mid l \notin F_1 \cup R \cup \text{locs}_{V_2',H_1}(e_2)\} \quad V_2', H_1, R, F_1 \cup g \vdash e_2 \Downarrow v_2, H_2, F_2}{V, H, R, F \vdash \text{let}(e_1; x : \tau.e_2) \Downarrow v_2, H_2, F_2} (S_{17})$$

4 Operational semantics

In order to prove the soundness of the type system, we also define a simplified operational semantics that does not account for garbage collection.

$$\boxed{V, H \vdash e \Downarrow v, H'}$$

This can be read as: under stack V , heap H the expression e evaluates to v , and engenders a new heap H' . We write the representative rules.

$$\frac{v = \langle V(x_1), V(x_2) \rangle \quad (L \sqcup \{l\}) \cap \text{dom}(H) = \emptyset \quad H', l = \text{copy}(H, L, v)}{V, H \vdash \mathbf{cons}(x_1; x_2) \Downarrow l, H'} \text{(S}_{18}\text{)}$$

$$\frac{V' \subseteq V \quad \text{dom}(V') = FV(e_2) \setminus \{x_h, x_t\} \quad V(x) = l \quad H(l) = \langle v_h, v_t \rangle \quad V'' = V'[x_h \mapsto v_h, x_t \mapsto v_t] \quad V'', H \vdash e_2 \Downarrow v, H'}{V, H \vdash \mathbf{case } x \{ \mathbf{nil} \hookrightarrow e_1 \mid \mathbf{cons}(x_h; x_t) \hookrightarrow e_2 \} \Downarrow v, H'} \text{(S}_{19}\text{)}$$

$$\frac{\text{dom}(V_2) = FV(\mathbf{lam}(x : \tau.e_2)) \quad V = V_1 \sqcup V_2 \quad \text{dom}(V_1) = FV(e_1) \quad V_1, H \vdash e_1 \Downarrow v_1, H_1 \quad V'_2 = V_2[x \mapsto v_1] \quad V'_2, H_1 \vdash e_2 \Downarrow v_2, H_2}{V, H \vdash \mathbf{let}(e_1; x : \tau.e_2) \Downarrow v_2, H_2} \text{(S}_{20}\text{)}$$

5 Well Defined Environments

In order to define the potential for first-order types, we need a notion of well-define environments, one that relates heap values to semantic values of a type. We first give a denotational semantics for the first-order types:

$$\begin{aligned} () &\in \llbracket \mathbf{unit} \rrbracket \\ \perp &\in \llbracket \mathbf{bool} \rrbracket \\ \top &\in \llbracket \mathbf{bool} \rrbracket \\ 0 &\in \llbracket \mathbf{nat} \rrbracket \\ n + 1 &\in \llbracket \mathbf{nat} \rrbracket \text{ if } n \in \llbracket \mathbf{nat} \rrbracket \\ [] &\in \llbracket L(A) \rrbracket \\ \pi(a, l) &\in \llbracket L(A) \rrbracket \text{ if } a \in \llbracket A \rrbracket \text{ and } l \in \llbracket L(A) \rrbracket \end{aligned}$$

Where semantic set for each type is the least set such that the above holds. Note $\pi(x, y)$ is the usual set-theoretic pairing function, and write $[a_1, \dots, a_n]$ for $\pi(a_1, \dots, \pi(a_n, []))$.

Now we give the judgements relating heap values to semantic values, in the form $\boxed{H \models v \mapsto a : A}$, which can be read as: under heap H , heap value v defines the semantic value $a \in \llbracket A \rrbracket$.

$$\begin{aligned} \frac{n \in \mathbb{Z}}{H \models n \mapsto n : \mathbf{nat}} \text{(V:ConstI)} \quad & \frac{}{H \models \mathbf{Null} \mapsto n : \mathbf{unit}} \text{(V:ConstI)} \quad & \frac{A \in \mathbf{BType}}{H \models \mathbf{Null} \mapsto n : L(A)} \text{(V:Nil)} \\ \frac{}{H \models \mathbf{T} \mapsto \top : \mathbf{bool}} \text{(V:True)} \quad & \frac{}{H \models \mathbf{F} \mapsto \perp : \mathbf{bool}} \text{(V:False)} \\ \frac{l \in \mathbf{Loc} \quad H(l) = \langle v_h, v_t \rangle \quad H \models v_h \mapsto a_1 : A \quad H \models v_t \mapsto [a_2, \dots, a_n] : L(A)}{H \models l \mapsto [a_1, \dots, a_n] : L(A)} \text{(V:Cons)} \end{aligned}$$

6 Stack vs Heap Allocated Types

In order to share variables, we need to distinguish between types that are allocated on the stack and the heap. We write $\boxed{\text{stack}(A)}$ to denote that values of type A will be allocated *entirely* on the stack at run time (no references into the heap).

$$\frac{A \in \{\text{unit}, \text{bool}, \text{nat}\}}{\text{stack}(A)} (\text{S:Const}) \quad \frac{\text{stack}(A_1) \quad \text{stack}(A_2)}{\text{stack}(A_1 \times A_2)} (\text{S:Product})$$

7 Linear Garbage Collection Type Rules

The linear version of the type system takes into account of garbage collected cells by returning potential locally in a match construct. Since we are interested in the number of heap cells, all constants are assumed to be nonnegative. The second let rule expresses the fact that since stack types don't reference heap cells, any heap cells used in the evaluation of e_1 can be deallocated, as there are no longer references to them in v_1 .

$$\begin{array}{c} \frac{n \in \mathbb{Z}}{\Sigma; \emptyset \mid \frac{q}{q'} n : \text{nat}} (\text{L:ConstI}) \quad \frac{}{\Sigma; \emptyset \mid \frac{q}{q'} () : \text{unit}} (\text{L:ConstU}) \quad \frac{}{\Sigma; \emptyset \mid \frac{q}{q'} \text{T} : \text{bool}} (\text{L:ConstT}) \\ \\ \frac{}{\Sigma; \emptyset \mid \frac{q}{q'} \text{F} : \text{bool}} (\text{L:ConstF}) \quad \frac{}{\Sigma; x : B \mid \frac{q}{q'} x : B} (\text{L:Var}) \quad \frac{\Sigma(f) = A \xrightarrow{q/q'} B}{\Sigma; x : A \mid \frac{q}{q'} f(x) : B} \\ \\ \frac{\Sigma; \Gamma \mid \frac{q}{q'} e_t : B \quad \Sigma; \Gamma \mid \frac{q}{q'} e_f : B}{\Sigma; \Gamma, x : \text{bool} \mid \frac{q}{q'} \text{if } x \text{ then } e_t \text{ else } e_f : B} (\text{L:Cond}) \quad \frac{}{\Sigma; x_1 : A_1, x_2 : A_2 \mid \frac{q}{q'} \langle x_1, x_2 \rangle : A_1 \times A_2} (\text{L:Pair}) \\ \\ \frac{\Sigma; \Gamma, x_1 : A_1, x_2 : A_2 \mid \frac{q}{q'} e : B}{\Sigma; \Gamma, x : (A_1, A_2) \mid \frac{q}{q'} \text{case } x \{ (x_1; x_2) \hookrightarrow e \} : B} (\text{L:MatP}) \quad \frac{}{\Sigma; \emptyset \mid \frac{q}{q'} \text{nil} : L^p(A)} (\text{L:Nil}) \\ \\ \frac{}{\Sigma; x_h : A, x_t : L^p(A) \mid \frac{q+p+1}{q} \text{cons}(x_h; x_t) : L^p(A)} (\text{L:Cons}) \\ \\ \frac{\Sigma; \Gamma \mid \frac{q}{q'} e_1 : B \quad \Sigma; \Gamma, x_h : A, x_t : L^p(A) \mid \frac{q+p+1}{q'} e_2 : B}{\Sigma; \Gamma, x : L^p(A) \mid \frac{q}{q'} \text{case } x \{ \text{nil} \hookrightarrow e_1 \mid \text{cons}(x_h; x_t) \hookrightarrow e_2 \} : B} (\text{L:MatL}) \\ \\ \frac{\Sigma; \Gamma_1 \mid \frac{q}{p} e_1 : A \quad \Sigma; \Gamma_2, x : A \mid \frac{p}{q'} e_2 : B}{\Sigma; \Gamma_1, \Gamma_2 \mid \frac{q}{q'} \text{let}(e_1; x : \tau.e_2) : B} (\text{L:Let}) \\ \\ \frac{\Sigma; \Gamma_1 \mid \frac{q}{p} e_1 : A \quad \text{stack}(A) \quad \Sigma; \Gamma_2, x : A \mid \frac{\max(p,q)}{q'} e_2 : B}{\Sigma; \Gamma_1, \Gamma_2 \mid \frac{q}{q'} \text{let}(e_1; x : \tau.e_2) : B} (\text{L:LetS}) \quad \frac{\Sigma; \Gamma \mid \frac{q}{q'} e : B}{\Sigma; \Gamma, x : A \mid \frac{q}{q'} \text{drop}(x; e) : B} (\text{L:Drop}) \\ \\ \frac{\text{stack}(A) \quad \Sigma; \Gamma, x_1 : A, x_2 : A \mid \frac{q}{q'} e : B}{\Sigma; \Gamma, x : A \mid \frac{q}{q'} \text{share } x \text{ as } x_1, x_2 \text{ in } e : B} (\text{L:Share}) \end{array}$$

Now if we take $\dagger : L^p(A) \mapsto L(A)$ as the map that erases resource annotations, we obtain a simpler typing judgement $\boxed{\Sigma^\dagger; \Gamma^\dagger \vdash e : B^\dagger}$.

8 Type Rules For Sharing

$$\begin{aligned}
 L^p(A) - n &= L^{\max(p-n,0)}(A - n) \\
 A_1 \times A_2 - n &= A_1 - n \times A_2 - n \\
 A - n &= A
 \end{aligned}$$

$$\frac{A \text{ with } A_1, A_2 \quad \Sigma; \Gamma, x_1 : A_1, x_2 : A_2 \mid_{\frac{q}{q'}} e : B}{\Sigma; \Gamma, x : A \mid_{\frac{q}{q'}} \text{share } x \text{ as } x_1, x_2 \text{ in } e : B} (\text{M:Share})$$

$$\frac{\Sigma; \Gamma_1 \mid_{\frac{q}{p}} e_1 : A \quad \Sigma; \Gamma_1 \mid_{\text{cf}} e_1 : A' \quad \Sigma; \Gamma_2, x : (A' - 1) \mid_{\frac{p}{q'}} e_2 : B}{\Sigma; \Gamma_1, \Gamma_2 \mid_{\frac{q}{q'}} \text{let}(e_1; x : \tau.e_2) : B} (\text{M:Let})$$

9 Soundness for Linear GC

We simplify the soundness proof of the type system for the general metric to one with monotonic resource. (No function types for now)

Lemma 1.1. *If $\Sigma; \Gamma \vdash_{q'}^q e : B$, then $\Sigma^\dagger; \Gamma^\dagger \vdash e : B^\dagger$.*

Lemma 1.2. *If $\Sigma; \Gamma \vdash_{q'}^q e : B$, then $\text{set}(FV^*(e))$ and $\text{dom}(\Gamma) = FV(e)$.*

Proof. Induction on the typing judgement. □

Lemma 1.3. *Let $H \models v \mapsto a : A$. For all sets of locations R , if $\text{reach}_H(v) \subseteq R$ and $\text{stable}(R, H, H')$, then $H' \models v \mapsto a : A$ and $\text{reach}_H(v) = \text{reach}_{H'}(v)$.*

Proof. Induction on the structure of v . □

Corollary 1.3.1. *Let $H \models V : \Gamma$. For all sets of locations R , if $\bigcup_{x \in V} \text{reach}_H(V(x)) \subseteq R$ and $\text{stable}(R, H, H')$, then $H' \models V : \Gamma$.*

Proof. Follows from Lemma 1.3. □

Lemma 1.4. *Let $H \models v \mapsto a : A$. If $\text{stack}(A)$, then $\Phi_H(v : A) = 0$.*

Proof. Induction on $H \models v \mapsto a : A$. □

Lemma 1.5 (heap conservation). *Let $V, H, R, F \vdash e \Downarrow v, H', F'$, and $g = H'RF'$. Then $\|V\|_H + |F| = \|v\|_{H'} + |F' \cup g|$.*

Lemma 1.6 (main lemma). *For all stacks V and heaps H , let $V, H, R, F \vdash e \Downarrow v, H', F'$ and $\Sigma; \Gamma \vdash e : B$. Then given the following:*

1. $\text{dom}(V) = FV(e)$
2. $\text{no_alias}(V, H)$, and
3. $\text{disjoint}(\{R, F, \text{locs}_{V,H}(e)\})$

We have the following:

1. $\text{set}(\text{reach}_{H'}(v))$
2. $\text{disjoint}(\{R, F', \text{reach}_{H'}(v)\})$, and
3. $\text{stable}(R, H, H')$

Proof. Nested induction on the evaluation judgement and the typing judgement.

Case 1: E:Var

Suppose $H \models V : \Gamma, \text{dom}(V) = FV(e), \text{no_alias}(V, H), \text{disjoint}(\{R, F, \text{locs}_{V,H}(e)\})$	
$\text{set}(\text{reach}_H(v))$	($\text{no_alias}(V, H)$)
$\text{disjoint}(\{R, F, \text{reach}_H(v)\})$	($\text{disjoint}(\{R, F, \text{locs}_{V,H}(e)\})$)
$\text{no_alias}(V, H)$	(Sp.)
$\text{stable}(R, H, H')$	($H = H'$)

Case 2: E:Const* Due to similarity, we show only for E:ConstI

$$\begin{array}{ll}
\text{Suppose } H \models V : \Gamma, \text{dom}(V) = FV(e), \text{no_alias}(V, H), \text{disjoint}(\{R, F, \text{locs}_{V,H}(e)\}) & \\
\text{set}(\text{reach}_H(v)) & (\text{reach}_H(v) = \emptyset) \\
\text{disjoint}(\{R, F, \emptyset\}) & (\text{disjoint}(R, F)) \\
\text{no_alias}(V, H) & (\text{Sp.}) \\
\text{stable}(R, H, H') & (H = H')
\end{array}$$

Case 4: E:App

Case 5: E:CondT Similar to E:MatNil

Case 6: E:CondF Similar to E:CondT

Case 7: E:Let

$$\begin{array}{ll}
V, H, R, F \vdash \text{let}(e_1; x : \tau.e_2) \Downarrow v_2, H_2, F_2 & (\text{case}) \\
V, H, R', F \vdash e_1 \Downarrow v_1, H_1, F_1 & (\text{ad.}) \\
\Sigma; \Gamma_1, \Gamma_2 \vdash \text{let}(e_1; x : \tau.e_2) : B & (\text{case}) \\
\Sigma; \Gamma_1 \vdash e_1 : A & (\text{ad.}) \\
\text{Suppose } H \models V : \Gamma, \text{dom}(V) = FV(e), \text{no_alias}(V, H), \text{disjoint}(\{R, F, \text{locs}_{V,H}(e)\}) & \\
H \models V_1 : \Gamma_1 & (\text{def of W.D.E and Lemma 1.2}) \\
\text{By IH, we have invariant on } J_1 & \\
\text{NTS (1) - (3) to instantiate invariant on } J_1 & \\
(1) \quad \text{dom}(V_1) = FV(e_1) & (\text{def of } V_1) \\
(2) \quad \text{no_alias}(V_1, H) & (\text{no_alias}(V, H) \text{ and } V_1 \subseteq V) \\
(3) \quad \text{disjoint}(R', F, \text{locs}_{V,H}(e_1)) & \\
F \cap R' = \emptyset & (F \cap \text{locs}_{V,H}(e) = \emptyset \text{ and } \text{locs}_{V_2,H}(\text{lam}(x : \tau.e_2)) \subseteq \text{locs}_{V,H}(e)) \\
FV(e_1) \cap FV(\text{lam}(x : \tau.e_2)) = \emptyset & (\text{Lemma 1.2}) \\
\text{locs}_{V,H}(e_1) \cap \text{locs}_{V_2,H}(\text{lam}(x : \tau.e_2)) = \emptyset & (\text{no_alias}(V, H)) \\
R' \cap \text{locs}_{V,H}(e_1) = \emptyset & (\text{disjoint}(\{R, \text{locs}_{V,H}(e)\})) \\
F \cap \text{locs}_{V,H}(e_1) = \emptyset & (\text{Sp.}) \\
\text{Thus we have } \text{disjoint}(R', F, \text{locs}_{V,H}(e_1)) & \\
\text{By IH,} & \\
(1) \quad \text{set}(\text{reach}_{H_1}(v_1)) & \\
(2) \quad \text{disjoint}(\{R', F_1, \text{reach}_{H_1}(v_1)\}) & \\
(3) \quad \text{stable}(R', H, H_1) & \\
V'_2, H_1, R, F_1 \cup g \vdash e_2 \Downarrow v_2, H_2, F_2 & (\text{ad.}) \\
\Sigma; \Gamma_2, x : A \vdash e_2 : B & (\text{ad.}) \\
H_1 \models V'_2 : (\Gamma_2, x : A) & (???) \\
\text{By IH, we have invariant on } J_2 & \\
\text{NTS (1) - (3) to instantiate invariant on } J_2 & \\
(1) \quad \text{dom}(V'_2) = FV(e_2) & (\text{def of } V'_2) \\
(2) \quad \text{no_alias}(V'_2, H_1) & \\
\text{Let } x_1, x_2 \in V'_2, x_1 \neq x_2 \text{ be arb.} & \\
\text{case: } x_1 \neq x, x_2 \neq x & \\
\text{reach}_H(V'_2(x_1)) \subseteq R' & (\text{reach}_H(V'_2(x_1)) \subseteq \text{locs}_{V'_2,H}(\text{lam}(x : \tau.e_2)))
\end{array}$$

$$\begin{aligned}
reach_H(V'_2(x_2)) &\subseteq R' & (reach_H(V'_2(x_2)) \subseteq locs_{V'_2, H}(\mathbf{lam}(x : \tau.e_2))) \\
reach_H(V'_2(x_1)) &= reach_{H_1}(V'_2(x_1)), reach_H(V'_2(x_2)) = reach_{H_1}(V'_2(x_2)) \\
& & (\text{stable}(R', H, H_1) \text{ and Lemma 1.3}) \\
reach_{H_1}(V'_2(x_1)) &= reach_H(V(x_1)), reach_{H_1}(V'_2(x_2)) = reach_H(V(x_2)) \\
& & (\text{stable}(R', H, H_1) \text{ and Lemma 1.3}) \\
no_alias(V'_2, H_1) & & (no_alias(V, H))
\end{aligned}$$

case: $x_1 = x, x_2 \neq x$

$$\begin{aligned}
reach_{H_1}(V'_2(x_1)) &= reach_{H_1}(v_1) & (\text{def of } V'_2) \\
reach_{H_1}(V'_2(x_2)) &\subseteq R' & (\text{same as above}) \\
set(reach_{H_1}(v_1)) & & (\text{IH 1.1}) \\
reach_{H_1}(V'_2(x_2)) &= reach_H(V(x_2)) & (\text{same as above}) \\
set(reach_{H_1}(V'_2(x_2))) & & (no_alias(V, H)) \\
reach_{H_1}(V'_2(x_1)) \cap reach_{H_1}(V'_2(x_2)) &= \emptyset & (\text{disjoint}(\{R', reach_{H_1}(v_1)\}))
\end{aligned}$$

Thus we have $no_alias(V'_2, H_1)$

$$(3) \quad \text{disjoint}(\{R, F_1 \cup g, locs_{V'_2, H_1}(e_2)\})$$

$$R \cap F_1 = \emptyset \quad (\text{disjoint}(\{R', F_1\}) \text{ from 1.2 and } R \subseteq R')$$

$$R \cap (F_1 \cup g) = \emptyset \quad (\text{def of } g)$$

$$NTS (F_1 \cup g) \cap locs_{V'_2, H_1}(e_2) = \emptyset$$

Let $l \in locs_{V'_2, H_1}(e_2)$ be arb.

$$l \in reach_{H_1}(V'_2(x')) \text{ for some } x' \in V'_2$$

case: $x' \neq x$

$$\begin{aligned}
reach_H(V_2(x')) &= reach_{H_1}(V'_2(x')) & (\text{same as above}) \\
reach_{H_1}(V'_2(x')) &\subseteq R' & (\text{def of } R') \\
reach_{H_1}(V'_2(x')) \cap F_1 &= \emptyset & (\text{disjoint}(\{R', F_1\}) \text{ from 1.2})
\end{aligned}$$

case: $x' = x$

$$\begin{aligned}
reach_{H_1}(V'_2(x')) &= reach_{H_1}(v_1) & (\text{def of } V'_2) \\
reach_{H_1}(V'_2(x')) \cap F_1 &= \emptyset & (\text{disjoint}(\{F_1, reach_{H_1}(v_1)\}) \text{ from 1.2}) \\
reach_{H_1}(V'_2(x')) &\subseteq locs_{V'_2, H_1}(e_2) & (\text{def of } locs_{V, H}) \\
reach_{H_1}(V'_2(x')) \cap g &= \emptyset & (\text{def of } g)
\end{aligned}$$

$$\text{Thus } reach_{H_1}(V'_2(x')) \cap (F_1 \cup g) = \emptyset$$

$$NTS R \cap locs_{V'_2, H_1}(e_2) = \emptyset$$

Let $l \in locs_{V'_2, H_1}(e_2)$ be arb.

$$l \in reach_{H_1}(V'_2(x')) \text{ for some } x' \in V'_2$$

case: $x' \neq x$

$$\begin{aligned}
reach_H(V_2(x')) &= reach_{H_1}(V'_2(x')) & (\text{same as above}) \\
l \in locs_{V, H}(e) & & (\text{def of } locs_{V, H}) \\
l \notin R & & (\text{disjoint}(\{R, locs_{V, H}(e)\}) \text{ from 0.3})
\end{aligned}$$

case: $x' = x$

$$\begin{aligned}
reach_{H_1}(V'_2(x')) &= reach_{H_1}(v_1) & (\text{def of } V'_2) \\
reach_{H_1}(V'_2(x')) \cap R &= \emptyset & (\text{disjoint}(\{R', reach_{H_1}(v_1)\}) \text{ from 1.2 and } R \subseteq R')
\end{aligned}$$

$$\text{Thus } reach_{H_1}(V'_2(x')) \cap R = \emptyset$$

$$\text{Hence we have } (3) \quad \text{disjoint}(R, F_1 \cup g, locs_{V'_2, H_1}(e_2))$$

By instantiating the invariant on J_2 , we have

- (1) $\text{set}(\text{reach}_{H_2}(v_2))$
- (2) $\text{disjoint}(\{R, F_2, \text{reach}_{H_2}(v_2)\})$
- (3) $\text{stable}(R, H_1, H_2)$

Lastly, showing (1) - (3) holds for the original case J_0 :

- (1) $\text{set}(\text{reach}_{H_2}(v_2))$ (By 2.1)
- (2) $\text{disjoint}(\{R, F_2, \text{reach}_{H_2}(v_2)\})$ (By 2.2)
- (3) $\text{stable}(R, H_1, H_2)$

Let $l \in R$ be arb.

$$H(l) = H_1(l) \quad (\text{stable}(R', H, H_1) \text{ from 1.3})$$

$$H_1(l) = H_2(l) \quad (\text{stable}(R, H_1, H_2) \text{ from 2.3})$$

$$H(l) = H_2(l)$$

Hence $\text{stable}(R, H, H_2)$

Case 8: E:Pair Similar to E:Var

Case 9: E:MatP Similar to E:MatCons

Case 10: E:Nil Similar to E:Const*

Case 11: E:Cons

$$V, H, R, F \vdash e \Downarrow l, H'', F' \quad (\text{case})$$

Suppose $H \models V : \Gamma, \text{dom}(V) = FV(e), \text{no_alias}(V, H), \text{disjoint}(\{R, F, \text{locs}_{V,H}(e)\})$

NTS (1) - (3) holds after evaluation

- (1) $\text{set}(\text{reach}_{H''}(l))$
- $\text{stable}(\{\text{locs}_{V,H}(e)\}, H, H'')$ (disjoint($\{F, \text{locs}_{V,H}(e)\}$) and *copy* only updates $l \in L \subseteq F$)
- $\text{reach}_H(V(x_i)) = \text{reach}_{H''}(V(x_i))$ ($\text{reach}_H(V(x_i)) \subseteq \text{locs}_{V,H}(e)$ and 1.3 for $i = 1, 2$)
- $\text{reach}_{H''}(l) = \{l\} \cup \text{reach}_{H''}(V(x_1)) \cup \text{reach}_{H''}(V(x_2))$ (def of reach_H)
- $\text{set}(\text{reach}_{H''}(l))$ ($l \notin \text{locs}_{V,H}(e)$ and $\text{no_alias}(V, H)$)
- (2) $\text{disjoint}(\{R, F', \text{reach}_{H''}(l)\})$
- $R \cap F' = \emptyset$ ($F' \subseteq F$ and $\text{disjoint}(\{R, F\})$)
- $R \cap \text{reach}_{H''}(l) = \emptyset$ ($l \in F$ and $\text{disjoint}(\{R, \text{locs}_{V,H}(e)\})$)
- $F' \cap \text{reach}_{H''}(l) = \emptyset$ ($F' \subseteq F$ and $\text{disjoint}(\{F, \text{locs}_{V,H}(e)\})$)
- Thus we have (2) $\text{disjoint}(\{R, F', \text{reach}_{H''}(l)\})$
- (3) $\text{stable}(R, H, H'')$ (since copy only updates $l \in L \subseteq F$ and $F \cap R = \emptyset$)

Case 12: E:MatNil

Suppose $H \models V : \Gamma, \text{dom}(V) = FV(e), \text{no_alias}(V, H), \text{disjoint}(\{R, F, \text{locs}_{V,H}(e)\})$

$\Sigma; \Gamma' \vdash e_1 : B$ (ad.)

$V, H, R, F \cup g \vdash e_1 \Downarrow v, H', F'$ (ad.)

$H \models V' : \Gamma'$ (def of W.D.E)

By IH, we have invariant on J_1

NTS (1) - (3) to instantiate invariant on J_1

- (1) $\text{dom}(V') = FV(e_1)$ (def of V')

- (2) $\text{no_alias}(V', H)$ ($\text{no_alias}(V, H)$ and $V' \subseteq V$)
(3) $\text{disjoint}(\{R, F, \text{locs}_{V', H}(e_1)\})$ ($\text{disjoint}(\{R, F, \text{locs}_{V, H}(e)\})$ and $\text{locs}_{V', H}(e_1) \subseteq \text{locs}_{V, H}(e)$)
Instantiating invariant on J_1 ,
(1) $\text{set}(\text{reach}_{H'}(v))$
(2) $\text{disjoint}(\{R, F_1, \text{reach}_{H'}(v)\})$
(3) $\text{stable}(R, H, H')$

Case 13: E:MatCons

- $V(x) = l$ (ad.)
 $H(l) = \langle v_h, v_t \rangle$ (ad.)
 $\Gamma = \Gamma', x : L(A)$ (ad.)
 $\Sigma; \Gamma', x_h : A, x_t : L(A) \vdash e_2 : B$ (ad.)
 $V'', H, R, F \cup g \vdash e_2 \Downarrow v_2, H_2, F'$ (ad.)
Suppose $H \models V : \Gamma, \text{dom}(V) = FV(e), \text{no_alias}(V, H), \text{disjoint}(\{F, R, \text{locs}_{V, H}(e)\})$
 $H \models V(x) : L(A)$ (def of W.D.E)
 $H'' \models v_h : A, H'' \models v_t : L(A)$ (ad.)
 $H \models v_h : A, H \models v_t : L(A)$ (???)
 $H \models V'' : \Gamma', x_h : A, x_t : L(A)$ (def of W.D.E)
By IH, we have invariant on J_1
NTS (1) - (3) to instantiate invariant on J_1
(1) $\text{dom}(V'') = FV(e_2)$ (def of V'')
(2) $\text{no_alias}(V'', H)$
Let $x_1, x_2 \in V'', x_1 \neq x_2, r_{x_1} = \text{reach}_H(V''(x_1)), r_{x_2} = \text{reach}_H(V''(x_2))$
case: $x_1 \notin \{x_h, x_t\}, x_2 \notin \{x_h, x_t\}$
(1), (2) from $\text{no_alias}(V, H)$
case: $x_1 = x_h, x_2 \notin \{x_h, x_t\}$
 $\text{set}(r_{x_1})$ (since $\text{set}(\text{reach}_H(V(x)))$ from $\text{no_alias}(V, H)$)
 $\text{set}(r_{x_2})$ (since $\text{no_alias}(V, H)$)
 $x_2 \in FV(e)$ (def of FV)
 $\text{reach}_H(V(x)) \cap r_{x_2} = \emptyset$ (def of reach and $\text{no_alias}(V, H)$)
hence $r_{x_1} \cap r_{x_2} = \emptyset$
case: $x_1 = x_h, x_2 = x_t$
 $\text{set}(r_{x_1})$ since $\text{set}(\text{reach}_H(V(x)))$ from $\text{no_alias}(V, H)$
 $\text{set}(r_{x_2})$ since $\text{set}(\text{reach}_H(V(x)))$ from $\text{no_alias}(V, H)$
 $r_{x_1} \cap r_{x_2} = \emptyset$ ($\text{set}(\text{reach}_H(V(x)))$)
case: otherwise
similar to the above
Thus we have $\text{no_alias}(V'', H)$
(3) $\text{disjoint}(\{R, F \cup g, \text{locs}_{V'', H}(e_2)\})$
 $(F \cup g) \cap R = \emptyset$ (since $F \cap R = \emptyset$ and by def of g)
NTS $R \cap \text{locs}_{V'', H}(e_2) = \emptyset$
Let $l' \in \text{locs}_{V'', H}(e_2)$ be arb.
case: $l' \in \text{reach}_H(V''(x'))$ for some $x' \in FV(e_2)$ where $x' \notin \{x_h, x_t\}$
 $x' \in V$ (def of V'')

$l' \in reach_H(V(x'))$
 $x' \in FV(e)$ (def of FV)
 $l' \in locs_{V,H}(e)$ (def of $locs_{V,H}$)
 $l' \notin R$ (disjoint($\{R, F, locs_{V,H}(e)\}$))
case: $l' \in reach_H(V''(x_h))$
tom $l' \in reach_H(v_h)$
 $l' \in reach_H(V(x))$ (def of $reach$)
 $l' \in locs_{V,H}(e)$ (def of $locs_{V,H}$)
 $l' \notin R$ (since disjoint($\{F, R, locs_{V,H}(e)\}$))
case: $l' \in reach_H(V''(x_t))$
 similar to above
 Hence $R \cap locs_{V'',H}(e_2) = \emptyset$
 $F \cap locs_{V'',H}(e_2) = \emptyset$ (Similar to above)
 $g \cap locs_{V'',H}(e_2) = \emptyset$ (def. of g)
 $(F \cup g) \cap locs_{V'',H}(e_2) = \emptyset$
 Thus disjoint($\{R, F \cup g, locs_{V'',H}(e_2)\}$)
 Instantiating invariant on J_1 ,
 (1) $set(reach_{H'}(v))$
 (2) $disjoint(\{R, F', reach_{H'}(v)\})$
 (3) $stable(R, H, H')$

Case 13: E:Drop

$e = \text{drop}(x; e')$ (case)
 $V', H, R, F \cup g \vdash e' \Downarrow v, H', F'(\mathcal{J}_1)$ (ad.)
 $\Gamma = \Gamma', x : A$ (case)
 $\Sigma; \Gamma' \vdash_{q'}^q e' : B$
 Suppose $dom(V) = FV(e)$, $no_alias(V, H)$, disjoint($\{R, F, locs_{V,H}(e)\}$)
 By IH, we have invariant on \mathcal{J}_1
 NTS (1) - (3) for \mathcal{J}_1
 (1) $dom(V') = FV(e')$ ($dom(V) = FV(e)$ and def of FV)
 (2) $no_alias(V', H)$ ($no_alias(V, H)$ and $V' \subseteq V$)
 (3) $disjoint(\{R, F \cup g, locs_{V',H}(e')\})$
 $g = reach_H(v')$ (case)
 $g \subseteq locs_{V,H}(e)$ (def of $locs_{V,H}$)
 $R \cap (F \cup g) = \emptyset$ (disjoint($\{R, F\}$) and disjoint($\{R, locs_{V,H}(e)\}$))
 $R \cap locs_{V',H}(e') = \emptyset$ (disjoint($\{R, locs_{V,H}(e)\}$) and $locs_{V',H}(e) \subseteq locs_{V,H}(e)$)
 $F \cap locs_{V',H}(e') = \emptyset$ (disjoint($\{F, locs_{V,H}(e)\}$) and $locs_{V',H}(e) \subseteq locs_{V,H}(e)$)
 $g \cap locs_{V',H}(e') = \emptyset$ ($no_alias(V, H)$)
 Instantiating invariant on \mathcal{J}_1 ,
 (1) $set(reach_{H'}(v))$
 (2) $\{R, F', reach_{H'}(v)\}$
 (3) $stable(R, H, H')$

Case 13: E:Share

$$e = \text{share}(x; x_1, x_2.e) \quad (\text{case})$$

□

Task 1.7 (Soundness). *let $H \models V : \Gamma, \Sigma; \Gamma \mid_{q'}^q e : B$, and $V, H \vdash e \Downarrow v, H'$. Then $\forall C \in \mathbb{Q}^+$ and $\forall F, R \subseteq \text{Loc}$, if we have the following (existence lemma):*

1. $\text{dom}(V) = \text{FV}(e)$
2. $\text{no_alias}(V, H)$
3. $\text{disjoint}(\{R, F, \text{locs}_{V,H}(e)\})$, and
4. $|F| \geq \Phi_{V,H}(\Gamma) + q + C$

then there exists $F' \subseteq \text{Loc}$ s.t.

1. $V, H, R, F \vdash e \Downarrow v, H', F'$
2. $|F'| \geq \Phi_{H'}(v : B) + q' + C$

Proof. Nested induction on the evaluation judgement and the typing judgement.

Case 1: E:Var

$$\begin{aligned}
V, H, R, F \vdash x \Downarrow V(x), H, F & \quad (\text{admissibility}) \\
\Sigma; x : B \mid_{q'}^q x : B & \quad (\text{admissibility}) \\
|F| - |F'| & \quad (1) \\
= |F| - |F| & \quad (\text{ad.}) \\
= 0 & \quad (2) \\
\Phi_{V,H}(\Gamma) + q - (\Phi_{H'}(v : B) + q') & \quad (3) \\
= \Phi_{V,H}(x : B) + q - (\Phi_H(V(x) : B) + q) & \quad (\text{ad.}) \\
= \Phi_H(V(x) : B) + q - (\Phi_H(V(x) : B) + q) & \quad (\text{def. of } \Phi_{V,H}) \\
= 0 & \quad (4) \\
|F| - |F'| \leq \Phi_{V,H}(\Gamma) + q - (\Phi_{H'}(v : B) + q') & \quad ((3),(5))
\end{aligned}$$

Case 2: E:Const* Due to similarity, we show only for E:ConstI

$$\begin{aligned}
|F| - |F'| &= |F| - |F| & (\text{ad.}) \\
&= 0 \\
\Phi_{V,H}(\Gamma) + q - (\Phi_{H'}(v : B) + q') &= \Phi_{V,H}(\emptyset) + q - (\Phi_H(v : \text{int}) + q) & (\text{ad.}) \\
&= 0 & (\text{def of } \Phi_{V,H}) \\
|F| - |F'| &\leq \Phi_{V,H}(\Gamma) + q - (\Phi_{H'}(v : B) + q')
\end{aligned}$$

Case 4: E:App

Case 5: E:CondT

$$\begin{aligned}
& \Gamma = \Gamma', x : \text{bool} && (\text{ad.}) \\
& H \models V : \Gamma' && (\text{def of W.F.E}) \\
& \Sigma; \Gamma' \mid_{q'}^q e_t : B && (\text{ad.}) \\
& V, H, R, F \cup g \vdash e_t \Downarrow v, H', F' && (\text{ad.}) \\
& |F \cup g| - |F'| \leq \Phi_{V,H}(\Gamma) + q - (\Phi_{H'}(v : B) + q') && (\text{IH}) \\
& |F| - |F'| \leq \Phi_{V,H}(\Gamma) + q - (\Phi_{H'}(v : B) + q')
\end{aligned}$$

Case 6: E:CondF Similar to E:CondT

Case 7: E:Let

$$\begin{aligned}
& V, H \vdash e \Downarrow v_2, H_2 && (\text{case}) \\
& V, H \vdash e_1 \Downarrow v_1, H_1 && (\text{ad.}) \\
& \Sigma; \Gamma_1 \mid_p^q e_1 : A && (\text{ad.}) \\
& H \models V_1 : \Gamma_1 && (\text{def of W.D.E})
\end{aligned}$$

Let $C \in \mathbb{Q}^+$, $F, R \subseteq \text{Loc}$ be arb.

Suppose $\text{dom}(V) = FV(e)$, $\text{no_alias}(V, H)$, $\text{disjoint}(\{R, F, \text{locs}_{V,H}(e)\})$, and $|F| \geq \Phi_{V,H}(\Gamma) + q + C$

NTF F' s.t.

1. $V, H, R, F \vdash e \Downarrow v_2, H_2, F'$ and
2. $|F'| \geq \Phi_{H_2}(v_2 : B) + q' + C$

Let $R' = R \cup \text{locs}_{V,H}(\text{lam}(x : \tau.e_2))$

$\text{disjoint}(\{R', F, \text{locs}_{V,H}(e_1)\})$

(Similar to case in Lemma 1.6)

Instantiate IH with $C = C + \Phi_{V_2,H}(\Gamma_2)$, $F = F, R = R'$, we get existence lemma on J_1 :

NTS (1) - (4) to instantiate existence lemma on J_1

- (1) $\text{dom}(V_1) = FV(e_1)$
- (2) $\text{no_alias}(V_1, H)$
- (3) $\text{disjoint}(\{R, F, \text{locs}_{V,H}(e)\})$ ((1) - (3) all verbatim as in Lemma 1.6)
- (4) $|F| \geq \Phi_{V_1,H}(\Gamma_1) + q + C + \Phi_{V,H}(\Gamma_2)$ ($|F| \geq \Phi_{V,H}(\Gamma) + q + C$ and $\Phi_{V,H}(\Gamma) \geq \Phi_{V_1,H}(\Gamma_1) + \Phi_{V,H}(\Gamma_2)$)

Instantiating existence lemma on J_1 , we get F'' s.t.

1. $V, H, R', F \vdash e_1 \Downarrow v_1, H_1, F''$ and
2. $|F''| \geq \Phi_{H_1}(v_1 : A) + p + C + \Phi_{V_2,H_1}(\Gamma_2)$

For the second premise:

$$\begin{aligned}
& \Sigma; \Gamma_2, x : A \mid_{q'}^p e_2 : B && (\text{ad.}) \\
& H_1 \models v_1 : A \text{ and} && (\text{Theorem 3.3.4}) \\
& H_1 \models V : \Gamma_2 && (???) \\
& H_1 \models V' : \Gamma_2, x : A && (\text{def of } \models) \\
& V', H_1 \vdash e_2 \Downarrow v_2, H_2 && (\text{ad.})
\end{aligned}$$

Let $g = \{l \in H_1 \mid l \notin F_1 \cup R \cup \text{locs}_{V',H_1}(e_2)\}$

Instantiate IH with $C = C, F = F'' \cup g, R = R$, we get existence lemma on J_2 :

NTS (1) - (4) to instantiate existence lemma on J_1

- (1) $\text{dom}(V'_2) = FV(e_2)$
- (2) $\text{no_alias}(V'_2, H_1)$

$$\begin{aligned}
(3) \quad & \text{disjoint}(\{R, F'' \cup g, \text{locs}_{V'_2, H_1}(e_2)\}) & ((1) - (3) \text{ all verbatim as in Lemma 1.6}) \\
(4) \quad & |F'' \cup g| \geq \Phi_{V'_2, H_1}(\Gamma_2, x : A) + p + C \\
& |F'' \cup g| \geq |F''| \\
& \geq \Phi_{H_1}(v_1 : A) + p + C + \Phi_{V_2, H}(\Gamma_2) & (\text{IH}) \\
& = \Phi_{H_1}(v_1 : A) + p + C + \Phi_{V'_2, H_1}(\Gamma_2) & (\text{Lemma 4.3.3}) \\
& = \Phi_{V'_2, H_1}(\Gamma_2, x : A) + p + C & (\text{def of } \Phi)
\end{aligned}$$

Instantiating existence lemma on J_2 , we get $F^{(3)}$ s.t.

$$\begin{aligned}
1. & V'_2, H_1, R, F'' \cup g \vdash e_2 \Downarrow v_2, H_2, F^{(3)} \\
2. & |F^{(3)}| \geq \Phi_{H_2}(v_2 : B) + q' + C
\end{aligned}$$

Take $F' = F^{(3)}$

$$\begin{aligned}
V, H, R, F \vdash e \Downarrow v_2, H_2, F' & \text{ and} & (\text{E:Let}) \\
|F'| \geq \Phi_{H_2}(v_2 : B) + q' + C & & (\text{from IH})
\end{aligned}$$

Case 14: E:Let1

$$\begin{aligned}
V, H \vdash e \Downarrow v_2, H_2 & & (\text{case}) \\
V, H \vdash e_1 \Downarrow v_1, H_1 & & (\text{ad.}) \\
\Sigma; \Gamma_1 \mid \frac{q}{p} e_1 : A & & (\text{ad.}) \\
H \models V_1 : \Gamma_1 & & (\text{def of W.D.E})
\end{aligned}$$

Let $C \in \mathbb{Q}^+$, $F, R \subseteq \text{Loc}$ be arb.

Suppose $\text{dom}(V) = FV(e)$, $\text{no_alias}(V, H)$, $\text{disjoint}(\{R, F, \text{locs}_{V, H}(e)\})$, and $|F| \geq \Phi_{V, H}(\Gamma) + q + C$

NTF F' s.t.

$$\begin{aligned}
1. & V, H, R, F \vdash e \Downarrow v_2, H_2, F' \text{ and} \\
2. & |F'| \geq \Phi_{H_2}(v_2 : B) + q' + C
\end{aligned}$$

Let $R' = R \cup \text{locs}_{V, H}(\text{lam}(x : \tau.e_2))$

$\text{disjoint}(\{R', F, \text{locs}_{V, H}(e_1)\})$ (Similar to case in Lemma 1.6)

Instantiate IH with $C = C + \Phi_{V_2, H}(\Gamma_2)$, $F = F$, $R = R'$, we get existence lemma on J_1 :

NTS (1) - (4) to instantiate existence lemma on J_1

$$\begin{aligned}
(1) \quad & \text{dom}(V_1) = FV(e_1) \\
(2) \quad & \text{no_alias}(V_1, H) \\
(3) \quad & \text{disjoint}(\{R, F, \text{locs}_{V, H}(e)\}) & ((1) - (3) \text{ all verbatim as in Lemma 1.6}) \\
(4) \quad & |F| \geq \Phi_{V_1, H}(\Gamma_1) + q + C + \Phi_{V, H}(\Gamma_2) \quad (|F| \geq \Phi_{V, H}(\Gamma) + q + C \text{ and } \Phi_{V, H}(\Gamma) \geq \Phi_{V_1, H}(\Gamma_1) + \Phi_{V, H}(\Gamma_2))
\end{aligned}$$

Instantiating existence lemma on J_1 , we get F'' s.t.

$$\begin{aligned}
1. & V, H, R', F \vdash e_1 \Downarrow v_1, H_1, F'' \text{ and} \\
2. & |F''| \geq \Phi_{H_1}(v_1 : A) + p + C + \Phi_{V_2, H_1}(\Gamma_2)
\end{aligned}$$

For the second premise:

$$\begin{aligned}
\Sigma; \Gamma_2, x : A \mid \frac{\max(p, q)}{q'} e_2 : B & & (\text{ad.}) \\
H_1 \models v_1 : A \text{ and} & & (\text{Theorem 3.3.4}) \\
H_1 \models V : \Gamma_2 & & (???) \\
H_1 \models V' : \Gamma_2, x : A & & (\text{def of } \models) \\
V', H_1 \vdash e_2 \Downarrow v_2, H_2 & & (\text{ad.}) \\
\text{Let } g = \{l \in H_1 \mid l \notin F'' \cup R \cup \text{locs}_{V', H_1}(e_2)\}
\end{aligned}$$

Instantiate IH with $C = C, F = F'' \cup g, R = R$, we get existence lemma on J_2 :

NTS (1) - (4) to instantiate existence lemma on J_1

- (1) $\text{dom}(V'_2) = FV(e_2)$
 - (2) $\text{no_alias}(V'_2, H_1)$
 - (3) $\text{disjoint}(\{R, F'' \cup g, \text{locs}_{V'_2, H_1}(e_2)\})$ ((1) - (3) all verbatim as in Lemma 1.6)
 - (4) $|F'' \cup g| \geq \Phi_{V'_2, H_1}(\Gamma_2, x : A) + q + C$
- $$\begin{aligned}
 & |F'' \cup g| \geq |F''| \\
 & \geq \Phi_{H_1}(v_1 : A) + p + C + \Phi_{V_2, H}(\Gamma_2) \quad (\text{IH}) \\
 & = \Phi_{H_1}(v_1 : A) + p + C + \Phi_{V'_2, H_1}(\Gamma_2) \quad (\text{Lemma 4.3.3}) \\
 & = \Phi_{V'_2, H_1}(\Gamma_2, x : A) + p + C \quad (\text{def of } \Phi)
 \end{aligned}$$

Instantiating existence lemma on J_2 , we get $F^{(3)}$ s.t.

- 1. $V'_2, H_1, R, F'' \cup g \vdash e_2 \Downarrow v_2, H_2, F^{(3)}$
- 2. $|F^{(3)}| \geq \Phi_{H_2}(v_2 : B) + q' + C$

Take $F' = F^{(3)}$

$V, H, R, F \vdash e \Downarrow v_2, H_2, F'$ and (E:Let)
 $|F'| \geq \Phi_{H_2}(v_2 : B) + q' + C$ (from IH)

Case 8: E:Pair Similar to E:Const*

Case 9: E:MatP Similar to E:MatCons

Case 10: E:Nil Similar to E:Const*

Case 11: E:Cons

$V, H \vdash \text{cons}(x_1; x_2) \Downarrow l, H'$ (case)
 Let $C \in \mathbb{Q}^+, F, R \subseteq \text{Loc}$ be arb.
 Suppose $\text{dom}(V) = FV(e), \text{no_alias}(V, H), \text{disjoint}(\{R, F, \text{locs}_{V, H}(e)\}), |F| \geq \Phi_{V, H}(\Gamma) + q + C$
 NTF F' s.t.
 1. $V, H, R, F \vdash e \Downarrow v, H', F'$ and
 2. $|F'| \geq \Phi_{H'}(v : B) + q' + C$
 Let $F' = F$

Case 12: E:MatNil Similar to E:Cond*

Case 13: E:MatCons

$V(x) = (l, \text{alive})$ (ad.)
 $H(l) = \langle v_h, v_t \rangle$ (ad.)
 $\Gamma = \Gamma', x : L^p(A)$ (ad.)
 $\Sigma; \Gamma', x_h : A, x_t : L^p(A) \mid \frac{q+p+1}{q'} e_2 : B$ (ad.)
 $V'', H \vdash e_2 \Downarrow v, H'$ (ad.)
 Let $C \in \mathbb{Q}^+, F, R \subseteq \text{Loc}$ be arb.
 $H \models V(x) : L^p(A)$ (def of W.D.E)
 $H'' \models v_h : A, H'' \models v_t : L^p(A)$ (ad.)
 $H \models v_h : A, H \models v_t : L^p(A)$ (???)

$H \models V'' : \Gamma', x_h : A, x_t : L^p(A)$ (def of W.D.E)

Suppose $\text{no_alias}(V, H)$, $\text{disjoint}(\{R, F, \text{locs}_{V,H}(e)\})$, and $|F| \geq \Phi_{V,H}(\Gamma) + q + C$

NTF F' s.t.

1. $V, H, R, F \vdash e \Downarrow v, H', F'$ and

2. $|F'| \geq \Phi_{H'}(v : B) + q' + C$

Let $g = \{l \in H \mid l \notin F \cup R \cup \text{locs}_{V'',H}(e_2)\}$

We want to g nonempty, in particular, that $l \in g$

$l \notin F \cup R$ ($\text{disjoint}(\{R, F, \text{locs}_{V,H}(e)\})$)

AFSOC $l \in \text{locs}_{V'',H}(e_2)$

Then $l \in \text{reach}_H(\bar{V}''(x'))$ for some $x' \neq x$

$x' \in \{x_h, x_t\}$ (since $\text{reach}_H(\bar{V}(x')) \cap \text{reach}_H(\bar{V}(x)) = \emptyset$ from $\text{no_alias}(V, H)$)

WLOG let $x' = x_h$

But then $\mu_{\text{reach}_H(\bar{V}(x))}(l) \geq 2$ and $\text{set}(\text{reach}_H(\bar{V}(x)))$ doesn't hold

$l \notin \text{locs}_{V'',H}(e_2)$

Hence $l \in g$

Next, we have $\text{no_alias}(V'', H)$ and $\text{disjoint}(\{R, F \cup g, \text{locs}_{V'',H}(e_2)\})$ (similar to case in Lemma 1.2)

By IH with $C' = C, F'' = F \cup g$ and the above conditions, we have: $F^{(3)}$ s.t.

1. $V'', H, R, F \cup g \vdash e_2 \Downarrow v, H', F^{(3)}$

2. $|F^{(3)}| \geq \Phi_{H'}(v : B) + q' + C$

Where we also verify the precondition that $|F''| \geq \Phi_{V'',H}(\Gamma', x_h : A, x_t : L^p(A)) + q + p + 1 + C' :$

$|F''| = |F \cup g|$

$= |F| + |g|$

(F and g disjoint)

$\geq \Phi_{V,H}(\Gamma) + q + C + |g|$

(Sp.)

$= \Phi_{V,H}(\Gamma', x_h : A, x_t : L^p(A)) + p + q + C + |g|$

(Lemma 4.1.1)

$= \Phi_{V,H}(\Gamma', x_h : A, x_t : L^p(A)) + p + q + C + 1$

(g nonempty)

Now take $F' = F^{(3)}$

$V, H, R, F \vdash e \Downarrow v, H', F'$

(E:MatCons)

$|F'| \geq \Phi_{H'}(v : B) + q' + C$

(From the IH)

□