Existential Types for Imperative Languages: Technical Results

Dan Grossman

18 October 2001

1 Introduction

This technical report contains the full type-safety proof for the language presented in the paper *Existential Types for Imperative Languages*, originally submitted for publication in October 2001. Because this report should be read only after the paper, effectively as an appendix, we do not repeat the motivation, examples, and informal presentation contained there. Also refer to the paper for related work and a bibliography. We do repeat the figures and definitions so that this report comprises a self-contained proof.

2 Language

In this section, we define the language—syntax, dynamic semantics, and static semantics—that we prove sound. If M is a partial map, then we write $M[a \mapsto b]$ for the map that is like M except that $M(a) = b$. Later we will talk about maps *extending* other maps. By definition, M' *extends* M when $dom(M) \subseteq \mathrm{dom}(M')$ and $M(a) = b$ implies $M'(a) = b$.

Figure 1 contains the syntax for the language. We consider all terms with binding occurrences to be α-convertible. The binding occurrences are α in $\exists^\phi \alpha.\tau$, x in $\textbf{let}\ x : \tau = e\ \textbf{in}\ s$, α and x in $\textbf{open}\ e\ \textbf{as}\ \alpha, x\ \textbf{in}\ s$, and α and x in $\textbf{open}\ e\ \textbf{as}\ \alpha, *x\ \textbf{in}\ s$.

The dynamic semantics is divided into judgments for manipulating heap records (Figure 2), expressions (Figure 3), and statements (Figure 4). Expression evaluation is defined in terms of mutually inductive judgments for evaluating "left expressions" ($\Downarrow_{\mathrm{L}}$) and "right expressions" ($\Downarrow_{\mathrm{R}}$). Statement evaluation uses substitution: We write $s\{\ell p/x\}$ for the substitution of ℓp for x in s and $s\{\tau/\alpha\}$ for the capture-avoiding substitution of τ for α in s. We extend substitution to maps (such as Γ : Var $\rightharpoonup$ Type) in a point-wise manner. We omit the straightforward but tedious definitions of substitution, which of course involve auxiliary definitions for substitution through expressions and types. Substitution simply replaces all free occurrences of the type/term variable with the type/location.

$$\ell \in \mathrm{Lab} \qquad c \in \mathrm{Int} \qquad x \in \mathrm{Var} \qquad \alpha \in \mathrm{Tyvar} \qquad H : \mathrm{Lab} \rightharpoonup \mathrm{Value}$$

$$\begin{array}{llcl}
\text{Type} & \tau & ::= & \mathsf{int} \mid \alpha \mid \tau* \mid \tau_1 \times \tau_2 \mid \exists^\phi \alpha.\tau \\
\text{Exp} & e & ::= & c \mid x \mid \ell p \mid (e_1, e_2) \mid e.i \mid \&e \mid *e \mid \textbf{pack}\ \tau', e\ \textbf{as}\ \exists^\phi \alpha.\tau \\
\text{Stmt} & s & ::= & \textbf{skip} \mid e_1 := e_2 \mid s_1; s_2 \mid \textbf{let}\ x : \tau = e\ \textbf{in}\ s \\
 & & \mid & \textbf{open}\ e\ \textbf{as}\ \alpha, x\ \textbf{in}\ s \mid \textbf{open}\ e\ \textbf{as}\ \alpha, *x\ \textbf{in}\ s \\
\text{Path} & p & ::= & \cdot \mid ip \mid \mathrm{u}p \\
\text{Field} & i & ::= & 0 \mid 1 \\
\text{Style} & \phi & ::= & \delta \mid \& \\
\\
\text{Value} & v & ::= & c \mid \&\ell p \mid \textbf{pack}\ \tau', v\ \textbf{as}\ \exists^\phi \alpha.\tau \mid (v_1, v_2)
\end{array}$$

Figure 1: Syntax

$$\frac{}{\mathrm{get}(v,\cdot,v)} \qquad \frac{\mathrm{get}(v_0,p,v)}{\mathrm{get}((v_0,v_1),0p,v)} \qquad \frac{\mathrm{get}(v_1,p,v)}{\mathrm{get}((v_0,v_1),1p,v)}$$

$$\frac{\mathrm{get}(v_1,p,v)}{\mathrm{get}(\mathbf{pack}\ \tau',v_1\ \mathbf{as}\ \exists^{\&}\alpha.\tau,\mathrm{u}p,v)}$$

$$\frac{}{\mathrm{set}(v_{old},\cdot,v,v)} \qquad \frac{\mathrm{set}(v_0,p,v,v')}{\mathrm{set}((v_0,v_1),0p,v,(v',v_1))} \qquad \frac{\mathrm{set}(v_1,p,v,v')}{\mathrm{set}((v_0,v_1),1p,v,(v_0,v'))}$$

$$\frac{\mathrm{set}(v_1,p,v,v')}{\mathrm{set}(\mathbf{pack}\ \tau',v_1\ \mathbf{as}\ \exists^{\phi}\alpha.\tau,\ \mathrm{u}p,\ v,\ \mathbf{pack}\ \tau',v'\ \mathbf{as}\ \exists^{\phi}\alpha.\tau)}$$

Figure 2: Dynamic Semantics: Heap Objects

$$\frac{}{H\vdash \ell p \Downarrow_{\mathrm{L}} \ell p} \qquad \frac{H\vdash e \Downarrow_{\mathrm{L}} \ell p}{H\vdash e.i \Downarrow_{\mathrm{L}} \ell p i} \qquad \frac{H\vdash e \Downarrow_{\mathrm{R}} \&\ell p}{H\vdash *e \Downarrow_{\mathrm{L}} \ell p}$$

$$\frac{}{H\vdash c \Downarrow_{\mathrm{R}} c} \qquad \frac{\mathrm{get}(H(\ell),p,v)}{H\vdash \ell p \Downarrow_{\mathrm{R}} v} \qquad \frac{H\vdash e \Downarrow_{\mathrm{R}} (v_0,v_1)}{H\vdash e.0 \Downarrow_{\mathrm{R}} v_0} \qquad \frac{H\vdash e \Downarrow_{\mathrm{R}} (v_0,v_1)}{H\vdash e.1 \Downarrow_{\mathrm{R}} v_1}$$

$$\frac{H\vdash e_0 \Downarrow_{\mathrm{R}} v_0 \quad H\vdash e_1 \Downarrow_{\mathrm{R}} v_1}{H\vdash (e_0,e_1) \Downarrow_{\mathrm{R}} (v_0,v_1)} \qquad \frac{H\vdash e \Downarrow_{\mathrm{L}} \ell p}{H\vdash \&e \Downarrow_{\mathrm{R}} \&\ell p} \qquad \frac{H\vdash e \Downarrow_{\mathrm{R}} \&\ell p \quad \mathrm{get}(H(\ell),p,v)}{H\vdash *e \Downarrow_{\mathrm{R}} v}$$

$$\frac{H\vdash e \Downarrow_{\mathrm{R}} v}{H\vdash \mathbf{pack}\ \tau',e\ \mathbf{as}\ \exists^{\phi}\alpha.\tau \Downarrow_{\mathrm{R}} \mathbf{pack}\ \tau',v\ \mathbf{as}\ \exists^{\phi}\alpha.\tau}$$

Figure 3: Dynamic Semantics: Expressions

$$\frac{H\vdash e_1 \Downarrow_{\mathrm{L}} \ell p \quad H\vdash e_2 \Downarrow_{\mathrm{R}} v \quad \mathrm{set}(H(\ell),p,v,v')}{(H,e_1:=e_2)\rightarrow(H[\ell\mapsto v'],\mathbf{skip})}$$

$$\frac{}{(H,\mathbf{skip};s)\rightarrow(H,s)} \qquad \frac{(H,s_1)\rightarrow(H',s_1')}{(H,s_1;s_2)\rightarrow(H',s_1';s_2)}$$

$$\frac{H\vdash e \Downarrow_{\mathrm{R}} v \quad \ell\notin\mathrm{dom}(H)}{(H,\mathbf{let}\ x:\tau=e\ \mathbf{in}\ s)\rightarrow(H[\ell\mapsto v],s\{\ell\cdot/x\})}$$

$$\frac{H\vdash e \Downarrow_{\mathrm{R}} \mathbf{pack}\ \tau',v\ \mathbf{as}\ \exists^{\phi}\alpha.\tau \quad \ell\notin\mathrm{dom}(H)}{(H,\mathbf{open}\ e\ \mathbf{as}\ \alpha,x\ \mathbf{in}\ s)\rightarrow(H[\ell\mapsto v],s\{\tau'/\alpha\}\{\ell\cdot/x\})}$$

$$\frac{H\vdash e \Downarrow_{\mathrm{L}} \ell' p \quad \mathrm{get}(H(\ell'),p,\mathbf{pack}\ \tau',v\ \mathbf{as}\ \exists^{\phi}\alpha.\tau) \quad \ell\notin\mathrm{dom}(H)}{(H,\mathbf{open}\ e\ \mathbf{as}\ \alpha,*x\ \mathbf{in}\ s)\rightarrow(H[\ell\mapsto\&\ell' p\mathrm{u}],s\{\tau'/\alpha\}\{\ell\cdot/x\})}$$

Figure 4: Dynamic Semantics: Statements

Heap Type	$\Psi : \mathrm{Lab} \rightharpoonup \mathrm{Type}$
Pack Map	$\Upsilon : \mathrm{Lab} \times \mathrm{Path} \rightharpoonup \mathrm{Type}$
Tyvar Set	$\Delta \subset \mathrm{Tyvar}$
Var Map	$\Gamma : \mathrm{Var} \rightharpoonup \mathrm{Type}$
$\Delta \vdash \tau$	τ closed under Δ
Type Context	$C = \Psi; \Upsilon; \Delta; \Gamma$
$\vdash \Psi$	for all $\tau \in \mathrm{rng}(\Psi)$, τ closed
$\vdash \Upsilon$	for all $\tau \in \mathrm{rng}(\Upsilon)$, τ closed
$\Delta \vdash \Gamma$	for all $\tau \in \mathrm{rng}(\Gamma)$, $\Delta \vdash \tau$
$\vdash \Psi; \Upsilon; \Delta; \Gamma$	$\vdash \Psi$, $\vdash \Upsilon$ and $\Delta \vdash \Gamma$

Figure 5: Static Semantics: Contexts and Well-Formedness

$$\frac{}{\vdash \mathsf{int}\ \text{packable}} \qquad \frac{}{\vdash \alpha\ \text{packable}} \qquad \frac{}{\vdash \tau*\ \text{packable}}$$

$$\frac{\vdash \tau\ \text{packable}}{\vdash \tau\ \text{assignable}} \qquad \frac{\vdash \tau_0\ \text{assignable} \quad \vdash \tau_1\ \text{assignable}}{\vdash \tau_0 \times \tau_1\ \text{assignable}} \qquad \frac{\vdash \tau\ \text{assignable}}{\vdash \exists^{\delta}\alpha.\tau\ \text{assignable}}$$

$$\frac{}{\vdash x\ \text{lval}} \qquad \frac{}{\vdash \ell p\ \text{lval}} \qquad \frac{}{\vdash *e\ \text{lval}} \qquad \frac{\vdash e\ \text{lval}}{\vdash e.i\ \text{lval}}$$

Figure 6: Static Semantics: Auxiliary Judgments

$$\frac{}{\Upsilon; \ell \vdash \mathrm{gettype}(p, \tau, \cdot, \tau)} \qquad \frac{\Upsilon; \ell \vdash \mathrm{gettype}(p\mathrm{u}, \tau'\{\Upsilon(\ell, p)/\alpha\}, p', \tau)}{\Upsilon; \ell \vdash \mathrm{gettype}(p, \exists^{\&}\alpha.\tau', \mathrm{u}p', \tau)}$$

$$\frac{\Upsilon; \ell \vdash \mathrm{gettype}(p0, \tau_0, p', \tau)}{\Upsilon; \ell \vdash \mathrm{gettype}(p, \tau_0 \times \tau_1, 0p', \tau)} \qquad \frac{\Upsilon; \ell \vdash \mathrm{gettype}(p1, \tau_1, p', \tau)}{\Upsilon; \ell \vdash \mathrm{gettype}(p, \tau_0 \times \tau_1, 1p', \tau)}$$

Figure 7: Static Semantics: Heap Objects

$$\frac{\vdash C}{C \vdash c : \mathsf{int}} \qquad \frac{\vdash \Psi; \Upsilon; \Delta; \Gamma}{\Psi; \Upsilon; \Delta; \Gamma \vdash x : \Gamma(x)} \qquad \frac{\Upsilon; \ell \vdash \mathrm{gettype}(\cdot, \Psi(\ell), p, \tau) \quad \vdash \Psi; \Upsilon; \Delta; \Gamma}{\Psi; \Upsilon; \Delta; \Gamma \vdash \ell p : \tau}$$

$$\frac{C \vdash e_0 : \tau_0 \quad C \vdash e_1 : \tau_1}{C \vdash (e_0, e_1) : \tau_0 \times \tau_1} \qquad \frac{C \vdash e : \tau_0 \times \tau_1}{C \vdash e_i : \tau_i} \qquad \frac{C \vdash e : \tau \quad \vdash e\ \text{lval}}{C \vdash \&e : \tau*} \qquad \frac{C \vdash e : \tau*}{C \vdash *e : \tau}$$

$$\frac{C \vdash e : \tau\{\tau'/\alpha\} \quad \vdash \tau'\ \text{packable}}{C \vdash \mathbf{pack}\ \tau', e\ \mathbf{as}\ \exists^{\phi}\alpha.\tau : \exists^{\phi}\alpha.\tau}$$

Figure 8: Static Semantics: Expressions

$$\frac{\vdash C}{C \vdash \mathbf{skip}} \qquad \frac{C \vdash s_1 \quad C \vdash s_2}{C \vdash s_1; s_2}$$

$$\frac{C \vdash e_1 : \tau \quad C \vdash e_2 : \tau \quad \vdash e_1 \text{ lval} \quad \vdash \tau \text{ assignable}}{C \vdash e_1 := e_2}$$

$$\frac{\Psi; \Upsilon; \Delta; \Gamma[x \mapsto \tau] \vdash s \quad \Psi; \Upsilon; \Delta; \Gamma \vdash e : \tau \quad x \notin \mathrm{dom}(\Gamma)}{\Psi; \Upsilon; \Delta; \Gamma \vdash \mathbf{let}\ x : \tau = e\ \mathbf{in}\ s}$$

$$\frac{\Psi; \Upsilon; \Delta, \alpha; \Gamma[x \mapsto \tau] \vdash s \quad \Psi; \Upsilon; \Delta; \Gamma \vdash e : \exists^{\phi}\alpha.\tau \quad \alpha \notin \Delta \quad x \notin \mathrm{dom}(\Gamma)}{\Psi; \Upsilon; \Delta; \Gamma \vdash \mathbf{open}\ e\ \mathbf{as}\ \alpha, x\ \mathbf{in}\ s}$$

$$\frac{\Psi; \Upsilon; \Delta, \alpha; \Gamma[x \mapsto \tau*] \vdash s \quad \Psi; \Upsilon; \Delta; \Gamma \vdash e : \exists^{\&}\alpha.\tau \quad \alpha \notin \Delta \quad x \notin \mathrm{dom}(\Gamma)}{\Psi; \Upsilon; \Delta; \Gamma \vdash \mathbf{open}\ e\ \mathbf{as}\ \alpha, *x\ \mathbf{in}\ s}$$

Figure 9: Static Semantics: Statements

$$\begin{array}{l} \Psi : \mathrm{Lab} \rightharpoonup \mathrm{Type} \\ \Upsilon : \mathrm{Lab} \times \mathrm{Path} \rightharpoonup \mathrm{Type} \end{array} \qquad \frac{\begin{array}{l} \vdash H : \Psi; \Upsilon \qquad \Psi; \Upsilon; \emptyset; \emptyset \vdash s \\ \text{for all } \tau \in \mathrm{rng}(\Psi) \cup \mathrm{rng}(\Upsilon),\ \tau \text{ is closed} \end{array}}{\vdash (H, s)}$$

$$\frac{\begin{array}{l} \mathrm{dom}(H) = \mathrm{dom}(\Psi) \\ \text{for all } \ell \in \mathrm{dom}(H),\ \Psi; \Upsilon; \emptyset; \emptyset \vdash H(\ell) : \Psi(\ell) \\ \text{for all } (\ell, p) \in \mathrm{dom}(\Upsilon),\ \mathrm{get}(H(\ell), p, \mathbf{pack}\ \Upsilon(\ell, p), v\ \mathbf{as}\ \exists^{\&}\alpha.\tau) \end{array}}{\vdash H : \Psi; \Upsilon}$$

Figure 10: Static Semantics: States and Heaps

The static semantics is divided into judgments for context well-formedness (Figure 5), type-checking heap records (Figure 7), expressions (Figure 8), statements (Figure 9), and program states (Figure 10), which comprise a heap and a statement. We also have auxiliary judgments (Figure 6) for restricting the types that may serve as witness types, the types of values that may be assigned to, and the forms of expressions that are "left expressions".

3 Canonical Forms and Substitution

In the section, we prove the canoncial forms and substitution properties that we will need when proving the preservation and progress lemmas in a later section. The canonical-forms lemma simply determines the forms of values given their types. The substitution lemmas indicate that certain properties are preserved when an appropriate type or location is substituted for a type variable or term variable, respectively.

Lemma 3.1 (Canonical Forms) *If* $\Psi;\Upsilon;\emptyset;\emptyset \vdash v : \tau$ *and* τ *is*

- int, *then* $v = c$
- $\tau'*$, *then* $v = \&\ell p$
- $\tau_0 \times \tau_1$, *then* $v = (v_1, v_2)$
- $\exists^{\&}\alpha.\tau'$, *then* $v = \mathbf{pack}\ \tau'', v'\ \mathbf{as}\ \exists^{\&}\alpha.\tau'$
- $\exists^{\delta}\alpha.\tau'$, *then* $v = \mathbf{pack}\ \tau'', v'\ \mathbf{as}\ \exists^{\delta}\alpha.\tau'$

Futhermore, $\tau \neq \alpha$.

Proof: By inspection of the static semantics for expressions and the form of values. □

Lemma 3.2 (Substitution Lemmas)

1. *If* $\Delta, \alpha \vdash \tau$ *and* $\Delta \vdash \tau'$, *then* $\Delta \vdash \tau\{\tau'/\alpha\}$.
2. *If* $\vdash \tau$ packable *and* $\vdash \tau'$ packable, *then* $\vdash \tau\{\tau'/\alpha\}$ packable.
3. *If* $\vdash \tau$ assignable *and* $\vdash \tau'$ assignable, *then* $\vdash \tau\{\tau'/\alpha\}$ assignable.
4. *If* $\vdash e$ lval, *then* $\vdash e\{\tau/\alpha\}$ lval.
5. *If* $\Psi;\Upsilon;\Delta,\alpha;\Gamma \vdash e : \tau$, $\Delta \vdash \tau'$, *and* $\vdash \tau'$ packable, *then* $\Psi;\Upsilon;\Delta;\Gamma\{\tau'/\alpha\} \vdash e\{\tau'/\alpha\} : \tau\{\tau'/\alpha\}$.
6. *If* $\Psi;\Upsilon;\Delta,\alpha;\Gamma \vdash s$, $\Delta \vdash \tau'$, *and* $\vdash \tau'$ packable, *then* $\Psi;\Upsilon;\Delta;\Gamma\{\tau'/\alpha\} \vdash s\{\tau'/\alpha\}$.
7. *If* $\vdash e$ lval, *then* $\vdash e\{\ell p/x\}$ lval.
8. *If* $\Psi;\Upsilon;\Delta;\Gamma[x \mapsto \tau'] \vdash e : \tau$ *and* $\Psi;\Upsilon;\Delta;\Gamma \vdash \ell p : \tau'$, *then* $\Psi;\Upsilon;\Delta;\Gamma \vdash e\{\ell p/x\} : \tau$.
9. *If* $\Psi;\Upsilon;\Delta;\Gamma[x \mapsto \tau'] \vdash s$ *and* $\Psi;\Upsilon;\Delta;\Gamma \vdash \ell p : \tau'$, *then* $\Psi;\Upsilon;\Delta;\Gamma \vdash s\{\ell p/x\}$.

Proof:

1. By induction on the structure of τ.
2. By inspection of the derivation of $\vdash \tau$ packable.
3. By induction on the derivation of $\vdash \tau$ assignable and Substitution Lemma 2.
4. By induction on the derivation of $\vdash e$ lval.
5. By induction on the derivation of $\Psi;\Upsilon;\Delta,\alpha;\Gamma \vdash e : \tau$, proceeding by cases on the last rule used:

c By applying Substitution Lemma 1 to each type in the range of Γ, the context is still well-formed. And c still has type int.

x By applying Substitution Lemma 1 to each type in the range of Γ, the context is still well-formed. And x trivially has type $\Gamma\{\tau'/\alpha\}(x)$.

ℓp Same argument as previous case, plus an inductive argument on the gettype relation that τ is closed because Ψ and Υ are well-formed.

(e_1, e_2) By induction.

$e'.i$ By induction.

$\&e'$ By induction and Substitution Lemma 4.

$*e'$ By induction.

pack By assumption, we have a derivation of the form

$$\frac{\Psi;\Upsilon;\Delta,\alpha;\Gamma \vdash e_1 : \tau_2\{\tau_1/\beta\} \quad \vdash \tau_1 \text{ packable}}{\Psi;\Upsilon;\Delta,\alpha;\Gamma \vdash \textbf{pack } \tau_1, e_1 \textbf{ as } \exists^\phi\beta.\tau_2 : \exists^\phi\beta.\tau_2}$$

where α-equivalence allows us to choose a $\beta \notin \Delta, \alpha$. By assumption, $\Delta \vdash \tau'$, so β is not free in τ'. By induction,

$$\Psi;\Upsilon;\Delta;\Gamma\{\tau'/\alpha\} \vdash e_1\{\tau'/\alpha\} : \tau_2\{\tau_1/\beta\}\{\tau'/\alpha\}$$

Because β is not free in τ', we can show that $\tau_2\{\tau_1/\beta\}\{\tau'/\alpha\} = \tau_2\{\tau'/\alpha\}\{\tau_1\{\tau'/\alpha\}/\beta\}$. Hence

$$\Psi;\Upsilon;\Delta;\Gamma\{\tau'/\alpha\} \vdash e_1\{\tau'/\alpha\} : \tau_2\{\tau'/\alpha\}\{\tau_1\{\tau'/\alpha\}/\beta\}$$

By assumption, $\vdash \tau'$ packable and by the original derivation $\vdash \tau_1$ packable. So by Substitution Lemma 1, $\vdash \tau_1\{\tau'/\alpha\}$ packable. Therefore, we can derive

$$\frac{\Psi;\Upsilon;\Delta;\Gamma\{\tau'/\alpha\} \vdash e_1\{\tau'/\alpha\} : \tau_2\{\tau'/\alpha\}\{\tau_1\{\tau'/\alpha\}/\beta\} \quad \vdash \tau_1\{\tau'/\alpha\} \text{ packable}}{\Psi;\Upsilon;\Delta;\Gamma\{\tau'/\alpha\} \vdash \textbf{pack } \tau_1\{\tau'/\alpha\}, e_1\{\tau'/\alpha\} \textbf{ as } \exists^\phi\beta.\tau_2\{\tau'/\alpha\} : \exists^\phi\beta.\tau_2\{\tau'/\alpha\}}$$

By the definition of substitution, the conclusion is the desired result.

6. By induction on the derivation of $\Psi;\Upsilon;\Delta,\alpha;\Gamma \vdash s$, proceeding by cases on the last rule used:

skip By applying Substitution Lemma 1 to each type in the range of Γ, the context is still well-formed.

$s_1; s_2$ By induction.

$e_1 := e_2$ By Substitution Lemmas 3, 4, and 5. (Note $\vdash \tau'$ assignable because $\vdash \tau'$ packable.)

let By induction and Substitution Lemma 5.

open Using α-conversion to ensure that the α in the derivation is distinct from the α in the statement of the lemma, the result follows from induction and Substitution Lemma 5.

7. By induction on the derivation of $\vdash e$ lval.

8. By induction on the derivation of $\Psi;\Upsilon;\Delta;\Gamma[x \mapsto \tau'] \vdash e : \tau$, proceeding by cases on the last rule used:

c Substitution has no effect.

x Trivial.

ℓp Substitution has no effect.

(e_1, e_2) By induction.

$e'.i$ By induction.

$\&e'$ By induction and Substitution Lemma 7.

$*e'$ By induction.

pack By induction.

9. By induction on the derivation of $\Psi; \Upsilon; \Delta; \Gamma[x \mapsto \tau'] \vdash s$, proceeding by cases on the last rule used:

skip Substitution has no effect.

$s_1; s_2$ By induction.

$e_1 := e_2$ By Substitution Lemmas 7 and 8.

let,**open** Using α-conversion to ensure that the x in the derivation is distinct from the x in the statement of the lemma, the result follows from induction. and Substitution Lemma 8.

$\square$

4 Path and Heap Lemmas

In this section, we prove technical lemmas about paths and the get, set, and gettype relations.

The path-extension lemmas let us reason about the existence of derivations using pi and $p\mathrm{u}$ given derivations using p. These lemmas are necessary because the get, set, and gettype relations are defined inductively in terms of leftmost path elements, but the dynamic semantics often adds path elements to the right.

The heap-extension lemmas are a standard part of syntactic type-safety proofs for languages with heaps. They indicate that as the heap grows, terms that do not refer to the new heap elements continue to type-check.

Heap lemmas 1 and 2 state fairly obvious properties about the get relation that we need during the proof of preservation. Heap lemma 3 indicates what given a gettype derivation, we can assume about get and set derivations involving the same path. In particular, the second conclusion in heap lemma 3 is the "progress lemma" for heap records. Heap lemma 4 is very important for the case of the preservation proof for assignment statements. It indicates that if we can use the gettype relation to determine that at the end of a path we have an assignable type, then no value at an extension of that path can be in the domain of Υ. This lemma relates the gettype relation (particularly the case for $\mathrm{u}p$) to the heap-wellformedness rule for Υ and the $\vdash \tau$ assignable judgment.

Lemma 4.1 (Path Extension Lemmas)

1. *Suppose* $\mathrm{get}(v, p, v')$.
   - *If* $v' = (v_0, v_1)$, *then* $\mathrm{get}(v, p0, v_0)$ *and* $\mathrm{get}(v, p1, v_1)$. *Else there are no derivations of the form* $\mathrm{get}(v, pip', v'')$.
   - *If* $v' = \mathbf{pack}\ \tau', v_u\ \mathbf{as}\ \exists^{\phi}\alpha.\tau$, *then* $\mathrm{get}(v, p\mathrm{u}, v_u)$. *Else there are no derivations of the form* $\mathrm{get}(v, p\mathrm{u}p', v'')$.
2. *Suppose* $\Upsilon; \ell \vdash \mathrm{gettype}(p, \tau, p', \tau')$. *Then if* $\tau' =$
   - $\tau_0 \times \tau_1$, *then* $\Upsilon; \ell \vdash \mathrm{gettype}(p, \tau, p'0, \tau_0)$ *and* $\Upsilon; \ell \vdash \mathrm{gettype}(p, \tau, p'1, \tau_1)$.
   - $\exists^{\&}\alpha.\tau_1$ *and* $(\ell, p) \in \mathrm{dom}(\Upsilon)$, *then* $\Upsilon; \ell \vdash \mathrm{gettype}(p, \tau, p'\mathrm{u}, \tau_1\{\Upsilon(\ell, p)/\alpha\})$.

Proof:

1. By induction on the length of p. If $p = \cdot$, then $v = v'$ and the result follows from inspection of the get relation (using the fact that $\cdot p_1 = p_1$ for all p_1). For longer p, we proceed by cases on the leftmost element of p. In each case, inversion on the $\mathrm{get}(v, p, v')$ derivation and the induction hypothesis suffice to conclude the desired results.

2. By induction on the length of p'. If $p' = \cdot$, then $\tau = \tau'$ and the result follows from inspection of the gettype relation (using the fact that $\cdot p_1 = p_1$ for all p_1). For longer p', we proceed by cases on the leftmost element of p'. In each case, inversion on the $\Upsilon; \ell \vdash \mathrm{gettype}(p, \tau, p', \tau')$ derivation and the induction hypothesis suffice to conclude the desired results.

□

Lemma 4.2 (Heap Extension Lemmas)
Suppose Υ' and Ψ' are well-formed extensions of well-formed Υ and Ψ, respectively.

1. *If* $\Upsilon; \ell \vdash \mathrm{gettype}(p, \tau, p', \tau')$*, then* $\Upsilon'; \ell \vdash \mathrm{gettype}(p, \tau, p', \tau')$.
2. *If* $\Psi; \Upsilon; \Delta; \Gamma \vdash e : \tau$*, then* $\Psi'; \Upsilon'; \Delta; \Gamma \vdash e : \tau$.
3. *If* $\Psi; \Upsilon; \Delta; \Gamma \vdash s$*, then* $\Psi'; \Upsilon'; \Delta; \Gamma \vdash s$.

Proof:

1. By induction on the derivation of $\Upsilon; \ell \vdash \mathrm{gettype}(p, \tau, p', \tau')$. The case for paths beginning with $\mathtt{u}$ uses the fact that Υ' extends Υ.
2. By induction on the derivation of $\Psi; \Upsilon; \Delta; \Gamma \vdash e : \tau$. For the base cases, it is trivial to show $\vdash \Psi'; \Upsilon'; \Delta; \Gamma$. For the label case, we need Heap Extension Lemma 1 and the fact that Ψ' extends Ψ.
3. By induction on the derivation of $\Psi; \Upsilon; \Delta; \Gamma \vdash s$ and Heap Extension Lemma 2.

□

Lemma 4.3 (Heap Lemmas)

1. *There is at most one v_2 such that* $\mathrm{get}(v_1, p, v_2)$.
2. *Suppose* $\mathrm{get}(v, p_1, v_1)$ *and* $\mathrm{get}(v, p_1 p_2, v_2)$*. Then* $\mathrm{get}(v_1, p_2, v_2)$.
3. *Suppose*
   - $\vdash H : \Psi; \Upsilon$
   - $\Psi; \Upsilon; \emptyset; \emptyset \vdash v_1 : \tau_1$
   - $\Upsilon; \ell \vdash \mathrm{gettype}(p_1, \tau_1, p_2, \tau_2)$
   - $\mathrm{get}(H(\ell), p_1, v_1)$

   Then:

   - *There exists a v_2 such that* $\mathrm{get}(H(\ell), p_1 p_2, v_2)$*. Furthermore,* $\Psi; \Upsilon; \emptyset; \emptyset \vdash v_2 : \tau_2$.
   - *For all v_2' there exists a v_1' such that* $\mathrm{set}(v_1, p_2, v_2', v_1')$.
4. *Suppose that in addition to the assumptions of Heap Lemma 3,* $\vdash \tau_2$ assignable*. Then for all p',* $(\ell, p_1 p_2 p') \notin \mathrm{dom}(\Upsilon)$.

Proof:

1. By straightforward induction on the length of p.
2. By straightforward induction on the length of p_1.
3. By induction on the length of p_2. If $p_2 = \cdot$, then the gettype relation ensures that $\tau_1 = \tau_2$ and the get relation ensures that $\mathrm{get}(H(\ell), p_1 \cdot, v_1)$. So letting $v_2 = v_1$, the assumption $\Psi; \Upsilon; \emptyset; \emptyset \vdash v_1 : \tau_1$ means $\Psi; \Upsilon; \emptyset; \emptyset \vdash v_2 : \tau_2$. We can trivially derive $\mathrm{set}(v_1, \cdot, v_2', v_2')$. For longer paths, we proceed by cases on the leftmost element:

$p_2 = 0p_2'$ – By assumption $\Upsilon; \ell \vdash \mathrm{gettype}(p_1, \tau_1, 0p_2', \tau_2)$, which by inversion ensures $\Upsilon; \ell \vdash \mathrm{gettype}(p_1 0, \tau_{10}, p_2', \tau_2)$ where $\tau_1 = \tau_{10} \times \tau_{11}$.

- By assumption $\Psi; \Upsilon; \emptyset; \emptyset \vdash v_1 : \tau_{10} \times \tau_{11}$, which by inversion ensures $v_1 = (v_{10}, v_{11})$ and $\Psi; \Upsilon; \emptyset; \emptyset \vdash v_{10} : \tau_{10}$.
- By assumption $\mathrm{get}(H(\ell), p_1, (v_{10}, v_{11}))$. By Path Extension Lemma 1, we know we can derive $\mathrm{get}(H(\ell), p_1 0, v_{10})$.

Given these facts, the induction hypothesis applies by substituting:

$p_1 0$	for	p_1
p_2'	for	p_2
τ_{10}	for	τ_1
τ_2	for	τ_2
v_{10}	for	v_1

Therefore, there exists a v_2 such that $\mathrm{get}(H(\ell), p_1 0 p_2', v_2)$ and $\Psi; \Upsilon; \emptyset; \emptyset \vdash v_2 : \tau_2$, as desired. Furthermore, for all v_2' there exists a v_{10}' such that $\mathrm{set}(v_{10}, p_2', v_2', v_{10}')$. Hence we can derive $\mathrm{set}((v_{10}, v11), 0p_2', v_2', (v_{10}', v_{11}))$, which satisfies the desired result (letting $v_1' = (v_{10}', v_{11})$).

$p_2 = 1p_2'$ Analogous to the previous case.

$p_2 = \mathtt{u}p_2'$

- By assumption $\Upsilon; \ell \vdash \mathrm{gettype}(p_1, \tau_1, \mathtt{u}p_2', \tau_2)$, which by inversion ensures $\Upsilon; \ell \vdash \mathrm{gettype}(p_1\mathtt{u}, \tau_{1u}, p_2', \tau_2)$ where $\tau_1 = \exists^{\&}\alpha.\tau_1'$ and $\tau_{1u} = \tau_1'\{\Upsilon(\ell, p_1)/\alpha\}$. Therefore, $(\ell, p_1) \in \mathrm{dom}(\Upsilon)$.
- By assumption $\Psi; \Upsilon; \emptyset; \emptyset \vdash v_1 : \exists^{\&}\alpha.\tau_1'$, which by inversion ensures $v_1 = \textbf{pack } \tau_p, v_{1u} \textbf{ as } \exists^{\&}\alpha.\tau_1'$ and $\Psi; \Upsilon; \emptyset; \emptyset \vdash v_{1u} : \tau_1'\{\tau_p/\alpha\}$.
- By assumption $\mathrm{get}(H(\ell), p_1, \textbf{pack } \tau_p, v_{1u} \textbf{ as } \exists^{\&}\alpha.\tau_1')$. By Path Extension Lemma 1, we know we can derive $\mathrm{get}(H(\ell), p_1\mathtt{u}, v_{1u})$.
- By assumption $\vdash H : \Psi; \Upsilon$. In particular, $(\ell, p_1) \in \mathrm{dom}(\Upsilon)$, so we know we can derive $\mathrm{get}(H(\ell), p_1, \textbf{pack } \Upsilon(\ell, p_1), v'' \textbf{ as } \exists^{\&}\alpha.\tau'')$. By Heap Lemma 1, that means $\tau_p = \Upsilon(\ell, p_1)$. Hence we have $\Upsilon; \ell \vdash \mathrm{gettype}(p_1\mathtt{u}, \tau_1'\{\tau_p/\alpha\}, p_2', \tau_2)$.

Given all these facts, the induction hypothesis applies by substituting:

$p_1\mathtt{u}$	for	p_1
p_2'	for	p_2
$\tau_1'\{\tau_p/\alpha\}$	for	τ_1
τ_2	for	τ_2
v_{1u}	for	v_1

Therefore, there exists a v_2 such that $\mathrm{get}(H(\ell), p_1\mathtt{u}p_2', v_2)$ and $\Psi; \Upsilon; \emptyset; \emptyset \vdash v_2 : \tau_2$, as desired. Furthermore, for all v_2' there exists a v_{1u}' such that $\mathrm{set}(v_{1u}, p_2', v_2', v_{1u}')$. Hence we can derive $\mathrm{set}(v_1, \mathtt{u}p_2', v_2', \textbf{pack } \tau_p, v_{1u}' \textbf{ as } \exists^{\&}\alpha.\tau_1')$, which satisfies the desired result (letting $v_1' = \textbf{pack } \tau_p, v_{1u}' \textbf{ as } \exists^{\&}\alpha.\tau_1')$.

4. By Heap Lemmas 1 and 3, we know there is exactly one v_2 such that $\mathrm{get}(H(\ell), p_1 p_2, v_2)$. Furthermore $\Psi; \Upsilon; \emptyset; \emptyset \vdash v_2 : \tau_2$. We continue by induction on the derivation of $\vdash \tau_2$ assignable, proceeding by cases on the last rule used.

   int By Canonical Forms, $v_2 = c$. By Path Extension Lemma 1, the only derivation of the form $\mathrm{get}(H(\ell), p_1 p_2 p', v'')$ is for $p' = \cdot$, that is, $v'' = c$. Hence $(\ell, p_1 p_2 p') \in \mathrm{dom}(\Upsilon)$ would violate the well-formedness of H.

   $\tau*$ Same argument as int case, replacing c with $\&\ell p$.

   $\exists^{\delta}\alpha.\tau'$ Same argument as int case, replacing c with $\textbf{pack } \tau_p, v_p \textbf{ as } \exists^{\delta}\alpha.\tau'$.

   α There are no values of type α, so by (the contrapositive of) Heap Lemma 3, we know that we cannot derive $\Upsilon; \ell \vdash \mathrm{gettype}(p_1, \tau_1, p_2, \tau_2)$. So the claim holds vacuously.

   $\tau_{20} \times \tau_{21}$ By Canoncial Forms, $v_2 = (v_{20}, v_{21})$. Hence by Path Extension Lemma 1, the only derivations of the form $\mathrm{get}(H(\ell), p_1 p_2 p', v'')$ are:

   - $p' = \cdot$. That is, $v_2 = v''$. In this case, $(\ell, p_1 p_2 p') \in \mathrm{dom}(\Upsilon)$ would violate the well-formedness of H.

– $p' = 0p''$. By Path Extension Lemma 1, $\mathrm{get}(H(\ell), p_1p_20, v_{20})$. By Path Extension Lemma 2, $\Upsilon; \ell \vdash \mathrm{gettype}(p_1, \tau_1, p_20, \tau_2)$. By inversion on $\vdash \tau_2$ assignable, we know $\vdash \tau_{20}$ assignable. So the induction hypothesis applies (substituting p_20 for p_2 and τ_{20} for τ_2), so $(\ell, p_1p_20p'') \notin \mathrm{dom}(\Upsilon)$, as desired.

– $p' = 1p''$. Analogous to previous case.

□

5 Preservation and Progress

The first preservation lemma relates the get and set judgments to the gettype judgment, just as the ensuing preservation lemmas relate the dynamic semantics for expressions and statements to the static semantics for expressions and statements. By moving most of the novel issues to preceding lemmas, the proofs of preservation and progress are at this point largely routine.

Lemma 5.1 (Preservation Lemmas)

1. *Suppose that in addition to the assumptions of Heap Lemma 4,* $\Psi; \Upsilon; \emptyset; \emptyset \vdash v_2 : \tau_2$ *and* $\mathrm{set}(v_1, p_2, v_2, v_1')$. *Then* $\Psi; \Upsilon; \emptyset; \emptyset \vdash v_1' : \tau_1$. *Furthermore, for all* $(\ell, p_1p') \in \mathrm{dom}(\Upsilon)$ *there are* v'' *and* τ'' *such that* $\mathrm{get}(v_1', p', \textbf{pack } \Upsilon(\ell, p_1p'), v'' \textbf{ as } \exists^{\&}\alpha.\tau'')$.
2. *Suppose* $\vdash H : \Psi; \Upsilon$ *and* $\Psi; \Upsilon; \emptyset; \emptyset \vdash e : \tau$. *If* $H \vdash e \Downarrow_{\mathrm{L}} \ell p$, *then* $\Psi; \Upsilon; \emptyset; \emptyset \vdash \ell p : \tau$ *and if* $H \vdash e \Downarrow_{\mathrm{R}} v$, *then* $\Psi; \Upsilon; \emptyset; \emptyset \vdash v : \tau$.
3. *Suppose* $\vdash H : \Psi; \Upsilon$, $\Psi; \Upsilon; \emptyset; \emptyset \vdash s$, *and* $(H, s) \to (H', s')$. *Then there exist* Ψ' *and* Υ' *extending* Ψ *and* Υ *respectively such that* $\vdash H' : \Psi'; \Upsilon'$ *and* $\Psi'; \Upsilon'; \emptyset; \emptyset \vdash s'$.

Proof:

1. By induction on the length of p_2. If $p_2 = \cdot$, then the set relation ensures that $v_1' = v_2$ and the gettype relation ensures that $\tau_2 = \tau_1$. Hence the assumption $\Psi; \Upsilon; \emptyset; \emptyset \vdash v_2 : \tau_2$ means $\Psi; \Upsilon; \emptyset; \emptyset \vdash v_1 : \tau_1$. Furthermore, by Heap Lemma 4, for all p', $(\ell, p_1 \cdot p') \notin \mathrm{dom}(\Upsilon)$, so the second conclusion holds vacuously. For longer paths, we proceed by cases on the leftmost element.

$p_2 = 0p_2'$

– By assumption $\Upsilon; \ell \vdash \mathrm{gettype}(p_1, \tau_1, 0p_2', \tau_2)$, which by inversion ensures $\Upsilon; \ell \vdash \mathrm{gettype}(p_10, \tau_{10}, p_2', \tau_2)$ where $\tau_1 = \tau_{10} \times \tau_{11}$.

– By assumption $\mathrm{set}(v_1, 0p_2', v_2, v_1')$, which by inversion ensures $\mathrm{set}(v_{10}, p_2', v_2, v_{10}')$ where $v_1 = (v_{10}, v_{11})$ and $v_1' = (v_{10}', v_{11})$.

– By assumption $\Psi; \Upsilon; \emptyset; \emptyset \vdash (v_{10}, v_{11}) : \tau_{10} \times \tau_{11}$, which by inversion ensures $\Psi; \Upsilon; \emptyset; \emptyset \vdash v_{10} : \tau_{10}$ and $\Psi; \Upsilon; \emptyset; \emptyset \vdash v_{11} : \tau_{11}$.

– By assumption $\mathrm{get}(H(\ell), p_1, (v_{10}, v_{11}))$. By Path Extension Lemma 1, we know we can derive $\mathrm{get}(H(\ell), p_10, v_{10})$.

Given these facts, the induction hypothesis applies by substituting:

p_10	for	p_1
p_2'	for	p_2
τ_{10}	for	τ_1
τ_2	for	τ_2
v_{10}	for	v_1
v_{10}'	for	v_1'
v_2	for	v_2

Hence by induction, $\Psi; \Upsilon; \emptyset; \emptyset \vdash v_{10}' : \tau_{10}$ and if $(\ell, p_10p'') \in \mathrm{dom}(\Upsilon)$ then $\mathrm{get}(v_{10}', p_10p'', \textbf{pack } \Upsilon(\ell, p_10p''), v'' \textbf{ as } \exists^{\&}\alpha.\tau'')$. So by the static semantics, we can derive $\Psi; \Upsilon; \emptyset; \emptyset \vdash (v_{10}', v_{11}) : \tau_{10} \times \tau_{11}$, as desired. If $(\ell, p_1p') \in \mathrm{dom}(\Upsilon)$, then

$\mathrm{get}(H(\ell), p_1 p', \mathbf{pack}\ \Upsilon(\ell, p_1 p'), v''\ \mathbf{as}\ \exists^{\&}\alpha.\tau'')$. We can use this fact to show $\mathrm{get}(v_1', p', \mathbf{pack}\ \Upsilon(\ell, p_1 p'), v''\ \mathbf{as}\ \exists^{\&}\alpha.\tau'')$ as follows: Because $\mathrm{get}(H(\ell), p_1, (v_{10}, v_{11}))$, Path Extension Lemma 1 ensures that p' has the form $\cdot$, $0p''$, or $1p''$. The first case is impossible because get is a function. The second case is derivable from the result of the induction. For the third case, we can use Heap Lemma 2 to conclude $\mathrm{get}(v_1, 1p'', \mathbf{pack}\ \Upsilon(\ell, p_1 1p''), v''\ \mathbf{as}\ \exists^{\&}\alpha.\tau'')$. By inversion, that means $\mathrm{get}(v_{11}, p'', \mathbf{pack}\ \Upsilon(\ell, p_1 1p''), v''\ \mathbf{as}\ \exists^{\&}\alpha.\tau'')$. Hence we can derive $\mathrm{get}(v_1', 1p'', \mathbf{pack}\ \Upsilon(\ell, p_1 1p''), v''\ \mathbf{as}\ \exists^{\&}\alpha.\tau'')$, as desired.

$p_2 = 1p_2'$ Analogous to the previous case.

$p_2 = \mathtt{u}p_2'$

- By assumption $\Upsilon; \ell \vdash \mathrm{gettype}(p_1, \tau_1, \mathtt{u}p_2', \tau_2)$, which by inversion ensures $\Upsilon; \ell \vdash \mathrm{gettype}(p_1\mathtt{u}, \tau_{1u}, p_2', \tau_2)$ where $\tau_1 = \exists^{\&}\alpha.\tau_1'$ and $\tau_{1u} = \tau_1'\{\Upsilon(\ell, p_1)/\alpha\}$. Therefore, $(\ell, p_1) \in \mathrm{dom}(\Upsilon)$.
- By assumption $\mathrm{set}(v_1, \mathtt{u}p_2', v_2, v_1')$, which by inversion ensures $\mathrm{set}(v_{1u}, p_2', v_2, v_{1u}')$ where $v_1 = \mathbf{pack}\ \tau_p, v_{1u}\ \mathbf{as}\ \exists^{\&}\alpha.\tau_1''$ and $v_1' = \mathbf{pack}\ \tau_p, v_{1u}'\ \mathbf{as}\ \exists^{\&}\alpha.\tau_1''$.
- By assumption, $\Psi; \Upsilon; \emptyset; \emptyset \vdash \mathbf{pack}\ \tau_p, v_{1u}\ \mathbf{as}\ \exists^{\&}\alpha.\tau_1'' : \exists^{\&}\alpha.\tau_1'$, which by inversion ensures $\tau_1'' = \tau_1'$ and $\Psi; \Upsilon; \emptyset; \emptyset \vdash v_{1u} : \tau_1'\{\tau_p/\alpha\}$.
- By assumption $\mathrm{get}(H(\ell), p_1, \mathbf{pack}\ \tau_p, v_{1u}\ \mathbf{as}\ \exists^{\&}\alpha.\tau_1')$. By Path Extension Lemma 1, we know we can derive $\mathrm{get}(H(\ell), p_1\mathtt{u}, v_{1u})$.
- By assumption $\vdash H : \Psi; \Upsilon$. In particular, $(\ell, p_1) \in \mathrm{dom}(\Upsilon)$, so we know we can derive $\mathrm{get}(H(\ell), p_1, \mathbf{pack}\ \Upsilon(\ell, p_1), v''\ \mathbf{as}\ \exists^{\&}\alpha.\tau'')$. But the get relation is a function, so $\tau_p = \Upsilon(\ell, p_1)$. Hence we have $\Upsilon; \ell \vdash \mathrm{gettype}(p_1\mathtt{u}, \tau_1'\{\tau_p/\alpha\}, p_2', \tau_2)$.

Given all these facts, the induction hypothesis applies by substituting:

$p_1\mathtt{u}$	for	p_1
p_2'	for	p_2
$\tau_1'\{\tau_p/\alpha\}$	for	τ_1
τ_2	for	τ_2
v_{1u}	for	v_1
v_{1u}'	for	v_1'
v_2	for	v_2

Hence by induction $\Psi; \Upsilon; \emptyset; \emptyset \vdash v_{1u}' : \tau_1'\{\tau_p/\alpha\}$ and if $(\ell, p_1\mathtt{u}p'') \in \mathrm{dom}(\Upsilon)$, then $\mathrm{get}(v_{1u}', p', \mathbf{pack}\ \Upsilon(\ell, p_1\mathtt{u}p''), v''\ \mathbf{as}\ \exists^{\&}\alpha.\tau'')$. So by the static semantics, $\Psi; \Upsilon; \emptyset; \emptyset \vdash \mathbf{pack}\ \tau_p, v_{1u}'\ \mathbf{as}\ \exists^{\&}\alpha.\tau_1' : \exists^{\&}\alpha.\tau_1'$, as desired. If $(\ell, p_1 p') \in \mathrm{dom}(\Upsilon)$, then $\mathrm{get}(H(\ell), p_1 p', \mathbf{pack}\ \Upsilon(\ell, p_1 p'), v''\ \mathbf{as}\ \exists^{\&}\alpha.\tau'')$. We can use this fact to show as follows: $\mathrm{get}(v_1', p', \mathbf{pack}\ \Upsilon(\ell, p_1 p'), v''\ \mathbf{as}\ \exists^{\&}\alpha.\tau'')$. Because $\mathrm{get}(H(\ell), p_1, \mathbf{pack}\ \tau_p, v_{1u}\ \mathbf{as}\ \exists^{\&}\alpha.\tau_1'')$, Path Extension Lemma 1 ensures that p' has the form $\cdot$ or $\mathtt{u}p''$. The first case is trivial because $\mathrm{get}(v_1', \cdot, v_1')$. The second case is derivable from the result of the induction.

2. By mutual induction on the derivations of the form $H \vdash e \Downarrow_{\mathrm{L}} \ell p$ and $H \vdash e \Downarrow_{\mathrm{R}} v$, proceeding by cases on the last step:

$H \vdash \ell p \Downarrow_{\mathrm{L}} \ell p$ Trivial.

$H \vdash e'.i \Downarrow_{\mathrm{L}} \ell p i$ By assumption, $\Psi; \Upsilon; \emptyset; \emptyset \vdash e'.i : \tau$, which by inversion ensures $\Psi; \Upsilon; \emptyset; \emptyset \vdash e' : \tau_0 \times \tau_1$ where $\tau = \tau_i$. By assumption, $H \vdash e' \Downarrow_{\mathrm{L}} \ell p'$. So by induction, $\Psi; \Upsilon; \emptyset; \emptyset \vdash \ell p' : \tau_0 \times \tau_1$. By inversion, it must be that $\Upsilon; \ell \vdash \mathrm{gettype}(\cdot, \Psi(\ell), p', \tau_0 \times \tau_1)$. Hence by Path Extension Lemma 2, we know we can derive $\Upsilon; \ell \vdash \mathrm{gettype}(\cdot, \Psi(\ell), p'i, \tau_i)$, from which we derive $\Psi; \Upsilon; \emptyset; \emptyset \vdash \ell p' i : \tau_i$, as desired.

$H \vdash *e' \Downarrow_{\mathrm{L}} \ell p$ By assumption, $\Psi; \Upsilon; \emptyset; \emptyset \vdash *e' : \tau$, which by inversion ensures $\Psi; \Upsilon; \emptyset; \emptyset \vdash e' : \tau*$. By assumption, $H \vdash *e' \Downarrow_{\mathrm{L}} \ell p$. So by induction, $\Psi; \Upsilon; \emptyset; \emptyset \vdash \&\ell p : \tau*$, which by inversion ensures that $\Psi; \Upsilon; \emptyset; \emptyset \vdash \ell p : \tau$, as desired.

$H \vdash c \Downarrow_{\mathrm{R}} c$ Trivial.

$H \vdash \ell p \Downarrow_{\mathrm{R}} v$ By assumption, $\Psi; \Upsilon; \emptyset; \emptyset \vdash \ell p : \tau$, which by inversion ensures $\Upsilon; \ell \vdash \mathrm{gettype}(\cdot, \Psi(\ell), p, \tau)$. By assumption, $H \vdash \ell p \Downarrow_{\mathrm{R}} v$, which by inversion ensures $\mathrm{get}(H(\ell), p, v)$. Trivially, $\mathrm{get}(H(\ell), \cdot, H(\ell))$. So Heap Lemma 3 applies (letting $v_1 = H(\ell)$, $v_2 = v$, $p_1 = \cdot$, $p_2 = p$, $\tau_1 = \Psi(\ell)$, and $\tau_2 = \tau$), ensuring $\Psi; \Upsilon; \emptyset; \emptyset \vdash v : \tau$, as desired.

$H \vdash *e' \Downarrow_{\mathrm{R}} v$ By assumption, $\Psi;\Upsilon;\emptyset;\emptyset \vdash *e' : \tau$, which by inversion ensures $\Psi;\Upsilon;\emptyset;\emptyset \vdash e' : \tau*$. By assumption, $H \vdash *e' \Downarrow_{\mathrm{R}} v$, which by inversion ensures $H \vdash e' \Downarrow_{\mathrm{R}} \&\ell p$ and $\mathrm{get}(H(\ell), p, v)$. So by induction, $\Psi;\Upsilon;\emptyset;\emptyset \vdash \&\ell p : \tau*$. So by inversion $\Psi;\Upsilon;\emptyset;\emptyset \vdash \ell p : \tau$ and therefore $\Upsilon;\ell \vdash \mathrm{gettype}(\cdot, \Psi(\ell), p, \tau)$. Using the latter fact and $\mathrm{get}(H(\ell), p, v)$, we can use Heap Lemma 2 as in the proof of the previous case to show $\Psi;\Upsilon;\emptyset;\emptyset \vdash v : \tau$, as desired.

others All other cases are straightforward inductive arguments, much like the argument for the $H \vdash *e' \Downarrow_{\mathrm{L}} \ell p$ case.

3. By induction on the derivation of $(H, s) \to (H', s')$, proceeding by cases on the last step. Several cases use the fact that if Ψ' and Υ' extend Ψ and Υ, then for all $\ell \in \mathrm{dom}(\Psi)$, $\Psi';\Upsilon';\emptyset;\emptyset \vdash H(\ell) : \Psi'(\ell)$. This follows from the definition of $\vdash H : \Psi;\Upsilon$ and Heap Extension Lemma 2. Furthermore, if $\ell \neq \ell'$ and $\mathrm{get}(H(\ell'), p, v')$, then trivially $\mathrm{get}(H[\ell \mapsto v](\ell'), p, v')$. It follows that each element of Υ of the form (ℓ', p) still satisfies the requirements for heap well-formedness. In particular, we can add new heap elements and let $\Upsilon' = \Upsilon$.

skip; s By inversion.

$s_1; s_2$ By induction.

let Let $\Psi' = \Psi[\ell \mapsto \tau]$ (an extension because $\ell \notin \mathrm{dom}(H)$) and $\Upsilon' = \Upsilon$. By inversion and Preservation Lemma 2, $\Psi;\Upsilon;\emptyset;\emptyset \vdash v : \tau$, so by Heap Extension Lemma 2, $\Psi';\Upsilon';\emptyset;\emptyset \vdash v : \tau$. Along with the facts proven above, this suffices to show $\vdash H[\ell \mapsto v] : \Psi';\Upsilon'$. By inversion, $\Psi;\Upsilon;\emptyset;[x \mapsto \tau] \vdash s$, which by Heap Extension Lemma 3 means $\Psi';\Upsilon';\emptyset;[x \mapsto \tau] \vdash s$. Because $\Psi'(\ell) = \tau$, it is easy to show $\Psi';\Upsilon';\emptyset;\emptyset \vdash \ell\cdot : \tau$. Hence by Substitution Lemma 9, $\Psi';\Upsilon';\emptyset;\emptyset \vdash s\{\ell\cdot/x\}$, as desired.

open Let $\Psi' = \Psi[\ell \mapsto \tau\{\tau'/\alpha\}]$ (an extension because $\ell \notin \mathrm{dom}(H)$) and $\Upsilon' = \Upsilon$. By inversion and Preservation Lemma 2, $\Psi;\Upsilon;\emptyset;\emptyset \vdash v : \tau\{\tau'/\alpha\}$ and $\vdash \tau'$ packable, so by Heap Extension Lemma 2, $\Psi';\Upsilon';\emptyset;\emptyset \vdash v : \tau\{\tau'/\alpha\}$. Along with the facts proven above, this suffices to show $\vdash H[\ell \mapsto v] : \Psi';\Upsilon'$. By inversion, $\Psi;\Upsilon;\alpha;[x \mapsto \tau] \vdash s$, which by Heap Extension Lemma 3 means $\Psi';\Upsilon';\alpha;[x \mapsto \tau] \vdash s$. Because $\vdash \tau'$ packable, Substitution Lemma 6 allows us to conclude $\Psi';\Upsilon';\emptyset;[x \mapsto \tau\{\tau'/\alpha\}] \vdash s\{\tau'/\alpha\}$. Because $\Psi'(\ell) = \tau\{\tau'/\alpha\}$, it is easy to show $\Psi';\Upsilon';\emptyset;\emptyset \vdash \ell\cdot : \tau\{\tau'/\alpha\}$. Hence by Substitution Lemma 9, $\Psi';\Upsilon';\emptyset;\emptyset \vdash s\{\tau'/\alpha\}\{\ell\cdot/x\}$, as desired.

open* Let $\Psi' = \Psi[\ell \mapsto \tau\{\tau'/\alpha\}*]$ (an extension because $\ell \notin \mathrm{dom}(H)$) and $\Upsilon' = \Upsilon[(\ell', p) \mapsto \tau']$. Υ' is a (not necessarily proper) extension of Υ because we know by inversion that $\mathrm{get}(H(\ell'), p, \textbf{pack}\ \tau', v\ \textbf{as}\ \exists^{\&}\alpha.\tau)$. Furthermore, get is a function and the derivation of $\vdash H : \Psi;\Upsilon$ guarantees that if $(\ell', p) \in \Upsilon$, then $\mathrm{get}(H(\ell'), p, \textbf{pack}\ \Upsilon(\ell', p), v''\ \textbf{as}\ \exists^{\&}\alpha.\tau'')$. By Path Extension Lemma 1, $\mathrm{get}(H(\ell'), p\mathrm{u}, v'')$. By inversion and Preservation Lemma 2, $\Psi;\Upsilon;\emptyset;\emptyset \vdash \ell' p : \exists^{\&}\alpha.\tau$ and $\vdash \tau'$ packable, which means $\Upsilon;\ell' \vdash \mathrm{gettype}(\cdot, \Psi(\ell'), p, \exists^{\&}\alpha.\tau)$. Hence by Heap Extension Lemma 1, $\Upsilon';\ell' \vdash \mathrm{gettype}(\cdot, \Psi(\ell'), p, \exists^{\&}\alpha.\tau)$. So by Path Extension Lemma 2, $\Upsilon';\ell' \vdash \mathrm{gettype}(\cdot, \Psi(\ell'), p\mathrm{u}, \tau\{\tau'/\alpha\})$. Hence we can derive, $\Psi';\Upsilon';\emptyset;\emptyset \vdash \&\ell' p\mathrm{u} : \tau\{\tau'/\alpha\}*$. Along with the facts proven above, and the fact that $\mathrm{get}(H(\ell'), p, \textbf{pack}\ \tau', v\ \textbf{as}\ \exists^{\&}\alpha.\tau)$, this suffices to show $\vdash H[\ell \mapsto \&\ell' p\mathrm{u}] : \Psi';\Upsilon'$. By inversion, $\Psi;\Upsilon;\alpha;[x \mapsto \tau*] \vdash s$, which by Heap Extension Lemma 3 means $\Psi';\Upsilon';\alpha;[x \mapsto \tau*] \vdash s$. Because $\vdash \tau'$ packable, Substitution Lemma 6 allows us to conclude $\Psi';\Upsilon';\emptyset;[x \mapsto \tau\{\tau'/\alpha\}*] \vdash s\{\tau'/\alpha\}$. Because $\Psi'(\ell) = \tau\{\tau'/\alpha\}*$, it is easy to show $\Psi';\Upsilon';\emptyset;\emptyset \vdash \ell\cdot : \tau\{\tau'/\alpha\}*$. Hence by Substitution Lemma 9, $\Psi';\Upsilon';\emptyset;\emptyset \vdash s\{\tau'/\alpha\}\{\ell\cdot/x\}$, as desired.

$e_1 := e_2$ Let $\Psi' = \Psi$ and $\Upsilon' = \Upsilon$. By inversion and Preservation Lemma 2, $\Psi;\Upsilon;\emptyset;\emptyset \vdash \ell p : \tau$, $\Psi;\Upsilon;\emptyset;\emptyset \vdash v : \tau$, $\mathrm{set}(H(\ell), p, v, v')$, and $\vdash \tau$ assignable. By inversion on $\Psi;\Upsilon;\emptyset;\emptyset \vdash \ell p : \tau$, we know $\Upsilon;\ell \vdash \mathrm{gettype}(\cdot, \Psi(\ell), p, \tau)$. We can trivially derive $\mathrm{get}(H(\ell), \cdot, H(\ell))$. Hence Preservation Lemma 1 applies (substituting $\cdot$ for p_1, $\Psi(\ell)$ for τ_1, p for p_2, τ for τ_2, $H(\ell)$ for v_1, v for v_2, and v' for v_1'). So $\Psi;\Upsilon;\emptyset;\emptyset \vdash v' : \Psi(\ell)$ and for all $(\ell, \cdot p') \in \mathrm{dom}(\Upsilon)$ the conditions for heap well-formedness still hold. Along with the arguments above, this suffices to show $\vdash H : \Psi;\Upsilon$. We can derive $\Psi;\Upsilon;\emptyset;\emptyset \vdash \textbf{skip}$ (a trivial induction shows $\vdash \Psi;\Upsilon;\emptyset;\emptyset$), so we are done.

$\square$

Lemma 5.2 (Progress) *Suppose* $\vdash H : \Psi; \Upsilon$. *Then:*

1. *If* $\Psi; \Upsilon; \emptyset; \emptyset \vdash e : \tau$, *then there exists a* v *such that* $H \vdash e \Downarrow_{\mathrm{R}} v$. *If we further assume* $\vdash e$ lval, *then there exist* ℓ *and* p *such that* $H \vdash e \Downarrow_{\mathrm{L}} \ell p$.

2. *If* $\Psi; \Upsilon; \emptyset; \emptyset \vdash s$, *then either* $s = \textbf{skip}$ *or there exist* H' *and* s' *such that* $(H, s) \rightarrow (H', s')$.

Proof:

1. By induction on the structure of e. Proceeding by cases:

   c Trivial because c is a value, $H \vdash c \Downarrow_{\mathrm{R}} c$, and we cannot derive $\vdash c$ lval.

   x Trivial because we cannot derive $\Psi; \Upsilon; \emptyset; \emptyset \vdash x : \tau$.

   ℓp The second desired result is trivial because $H \vdash \ell p \Downarrow_{\mathrm{L}} \ell p$. For the first, inversion of the typing derivation ensures $\Upsilon; \ell \vdash \mathrm{gettype}(\cdot, \Psi(\ell), p, \tau)$. Furthermore, $\mathrm{get}(H(\ell), \cdot, H(\ell))$ and $\Psi; \Upsilon; \emptyset; \emptyset \vdash H(\ell) : \Psi(\ell)$. So by Heap Lemma 3, there exists a v such that $\mathrm{get}(H(\ell), p, v)$. Hence we can derive $H \vdash \ell p \Downarrow_{\mathrm{R}} v$, as desired.

   (e_1, e_2) For the first result, a straightforward inductive argument suffices because (v_1, v_2) is a value. The second result is trivial because we cannot derive $\vdash (e_1, e_2)$ lval.

   $e'.i$ For the first result, the induction hypothesis ensures that $H \vdash e' \Downarrow_{\mathrm{R}} v'$. By inversion of the typing derivation $\Psi; \Upsilon; \emptyset; \emptyset \vdash e' : \tau_0 \times \tau_1$. So by Preservation, $\Psi; \Upsilon; \emptyset; \emptyset \vdash v' : \tau_0 \times \tau_1$. So by Canonical Forms, $v' = (v_0, v_1)$. So we can derive $H \vdash e'.i \Downarrow_{\mathrm{R}} v_i$, as desired. For the second result, inversion of $\vdash e'.i$ lval ensures $\vdash e'$ lval. Hence by induction, $H \vdash e' \Downarrow_{\mathrm{L}} \ell p'$. Hence we can derive $H \vdash e'.i \Downarrow_{\mathrm{L}} \ell p' i$, as desired.

   $\&e'$ For the first result, the typing derivation ensures that $\vdash e'$ lval and $\Psi; \Upsilon; \emptyset; \emptyset \vdash e' : \tau'$. Hence by induction $H \vdash e' \Downarrow_{\mathrm{L}} \ell p$. Hence we can derive $H \vdash \&e' \Downarrow_{\mathrm{R}} \&\ell p$ and $\&\ell p$ is a value, as desired. The second result is trivial because we cannot derive $\vdash \&e'$ lval.

   $*e'$ For the first result, the typing derivation ensures that $\Psi; \Upsilon; \emptyset; \emptyset \vdash e' : \tau*$. Hence by induction $H \vdash e' \Downarrow_{\mathrm{R}} v'$. So by Preservation, $\Psi; \Upsilon; \emptyset; \emptyset \vdash v' : \tau*$. So by Canonical Forms, $v' = \&\ell p$. So inversion of $\Psi; \Upsilon; \emptyset; \emptyset \vdash v' : \tau*$ ensures $\Upsilon; \ell \vdash \mathrm{gettype}(\cdot, \Psi(\ell), p, \tau)$. So using the same argument as in the ℓp case, there exists a v such that $\mathrm{get}(H(\ell), p, v)$. Hence we can derive $H \vdash *e \Downarrow_{\mathrm{R}} v$, as desired. For the second result, we just showed $H \vdash e' \Downarrow_{\mathrm{R}} \&\ell p$. Hence we can derive $H \vdash *e \Downarrow_{\mathrm{L}} \ell p$.

   $\textbf{pack}\ \tau', e'\ \textbf{as}\ \exists^{\phi}\alpha.\tau''$ For the first result, a straightforward inductive argument suffices because $\textbf{pack}\ \tau', v'\ \textbf{as}\ \exists^{\phi}\alpha.\tau''$ is a value. The second result is trivial because we cannot derive $\vdash \textbf{pack}\ \tau', e'\ \textbf{as}\ \exists^{\phi}\alpha.\tau''$ lval.

2. By induction on the structure of s. Proceeding by cases:

   skip Trivial.

   $e_1 := e_2$ By inversion of the typing derivation and Progress Lemma 1, we know $H \vdash e_1 \Downarrow_{\mathrm{L}} \ell p$ and $H \vdash e_2 \Downarrow_{\mathrm{R}} v$ for some ℓ, p, and v. By Preservation $\Psi; \Upsilon; \emptyset; \emptyset \vdash \ell p : \tau$, which by inversion ensures that $\Upsilon; \ell \vdash \mathrm{gettype}(\cdot, \Psi(\ell), p, \tau)$. Furthermore, $\Psi; \Upsilon; \emptyset; \emptyset \vdash H(\ell) : \Psi(\ell)$ and $\mathrm{get}(H(\ell), \cdot, H(\ell))$, so by Heap Lemma 3, we know there exists a v' such that $\mathrm{set}(H(\ell), p, v, v')$. Hence we can derive $(H, e_1 := e_2) \rightarrow (H', \textbf{skip})$.

   $s_1; s_2$ Trivial if $s_1 = \textbf{skip}$. Else by induction.

   let By inversion of the typing derivation and Progress Lemma 1, we know $H \vdash e \Downarrow_{\mathrm{R}} v$ for some v. Hence we can derive $(H, \textbf{let}\ x : \tau = e\ \textbf{in}\ s) \rightarrow (H[\ell \mapsto v], s\{\ell \cdot / x\})$ for any $\ell \notin \mathrm{dom}(H)$.

   open By inversion of the typing derivation and Progress Lemma 1, we know $H \vdash e \Downarrow_{\mathrm{R}} v$. By Preservation $\Psi; \Upsilon; \emptyset; \emptyset \vdash v : \exists^{\phi}\alpha.\tau$, so by Canonical Forms, $v = \textbf{pack}\ \tau', v'\ \textbf{as}\ \exists^{\phi}\alpha.\tau$. Hence we can derive $(H, \textbf{open}\ e\ \textbf{as}\ \alpha, x\ \textbf{in}\ s) \rightarrow (H[\ell \mapsto v], s\{\tau'/\alpha\}\{\ell \cdot / x\})$ for any $\ell \notin \mathrm{dom}(H)$.

   open* By inversion of the typing derivation and Progress Lemma 1, we know $H \vdash e \Downarrow_{\mathrm{L}} \ell' p$. By Preservation, $\Psi; \Upsilon; \emptyset; \emptyset \vdash \ell' p : \exists^{\&}\alpha.\tau$, which by inversion ensures that $\Upsilon; \ell \vdash \mathrm{gettype}(\cdot, \Psi(\ell'), p, \exists^{\&}\alpha.\tau)$. Furthermore, $\Psi; \Upsilon; \emptyset; \emptyset \vdash H(\ell') : \Psi(\ell')$ and $\mathrm{get}(H(\ell'), \cdot, H(\ell'))$, so by Heap Lemma 3, we know there exists a v_2 such that $\mathrm{get}(H(\ell'), p, v_2)$ and $\Psi; \Upsilon; \emptyset; \emptyset \vdash v_2 : \exists^{\&}\alpha.\tau$. So by Canonical Forms, $v_2 = \textbf{pack}\ \tau', v'\ \textbf{as}\ \exists^{\&}\alpha.\tau$. Hence
   we can derive $(H, \textbf{open}\ e\ \textbf{as}\ \alpha, *x\ \textbf{in}\ s) \rightarrow (H[\ell \mapsto \&\ell' p\mathrm{u}], s\{\tau'/\alpha\}\{\ell \cdot / x\})$ for any $\ell \notin \mathrm{dom}(H)$.

$\square$

6 Conclusion: Type Soundness

We can now conclude type soundness as a simple corollary to Preservation Lemma 3 and Progress Lemma 2. Specifically, if $\rightarrow^*$ is the reflexive, transitive closure of $\rightarrow$, then given $\Psi; \Upsilon; \emptyset; \emptyset \vdash (H, s)$, there is no (H', s') such that $\vdash (H, s) \rightarrow^* (H', s')$ where $s' \neq \textbf{skip}$ and there is no (H'', s'') such that $(H', s') \rightarrow (H'', s'')$. That is, well-typed programs cannot get stuck. The proof is a straightforward induction on the length of the evaluation sequence.