Verifying Object-Oriented Code Using Object Propositions

Ligia Nistor+ Jonathan Aldrich+ Hannes Mehnert*
School of Computer Science, Carnegie Mellon University+
*IT University of Copenhagen
{lnistor,aldrich}@cs.cmu.edu, hame@itu.dk

1. Dynamic Semantics Rules

The complete set of dynamic semantics rules is presented in Figure 1.

2. Proving the Preservation Theorem

LEMMA 2.1. (Substitution) *If* $(\Gamma, y : T_y), (\Pi_1, y \rightsquigarrow R_y) \vdash e : \exists x : T.R$ *and* $\Gamma, \Pi_2 \vdash e_1 : \exists x : T_y.R_y$ *then* $\Gamma, (\Pi_1, \Pi_2) \vdash [e_1/y]e : \exists x : T.[e_1/y]R$.

Proof of Substitution Lemma

The proof is by induction on the derivation of $(\Gamma, y : T_y), (\Pi_1, y \rightsquigarrow R_y) \vdash e : \exists x : T.R$.

1. e is a value v. The values that e can take in this case are $o|true|false|n$. We know $(\Gamma, y : T_y), (\Pi_1, y \rightsquigarrow R_y) \vdash v : \exists x : T.R$ and we see that R might contain object propositions refering to y, which will have to be substituted when y is not in the linear context any more. Since $[e_1/y]e = v$, $\Gamma, (\Pi_1, \Pi_2) \vdash v : \exists x : T.[e_1/y]R$ directly.
2. e is a variable z,$z \neq y$. We know $(\Gamma, y : T_y), (\Pi_1, y \rightsquigarrow R_y) \vdash z : \exists x : T.R$. We see that R might contain object propositions refering to y, which will have to be substituted when y is not in the linear context any more. Since $[e_1/y]e = z$, and $z : T$ must be in Γ, and $z \rightsquigarrow R$ must be in Π_1(by inversion), $\Gamma, (\Pi_1, \Pi_2) \vdash z : \exists x : T.[e_1/y]R$.
3. e is the variable y. Now $[e_1/y]e = e_1$ and $T = T_y$ and $R = R_y$. Thus, $\Gamma, (\Pi_1, \Pi_2) \vdash e_1 : \exists x : T.[e_1/y]R$.
4. e is $t.f_i$. We know that $(\Gamma, y : T_y); (\Pi, y \rightsquigarrow R_y) \vdash t.f_i : \exists x : T.R$. We also know by inversion that $(\Pi, y \rightsquigarrow R_y) \vdash t.f_i \rightarrow r$ *or* $(\Pi, y \rightsquigarrow R_y) \vdash t.f_i \multimap r$ and that $\Gamma, (\Pi, y \rightsquigarrow R_y) \vdash [r/x]R$. Using the induction hypothesis we have: $(\Pi, \Pi_2) \vdash [e_1/y](t.f_i \rightarrow r)$ *or* $(\Pi, \Pi_2) \vdash [e_1/y](t.f_i \multimap r)$ and $\Gamma, (\Pi, \Pi_2) \vdash [e_1/y][r/x]R$. Since $t.f_i$ is just the syntactic representation of a field, the substitution will happen in r: $(\Pi, \Pi_2) \vdash (t.f_i \rightarrow [e_1/y]r)$ *or* $(\Pi, \Pi_2) \vdash (t.f_i \multimap [e_1/y]r)$. Also, we can rewrite $[e_1/y][r/x]R$ as $[([e_1/y]r)/x][e_1/y]R$ and so we have $\Gamma, (\Pi, \Pi_2) \vdash [([e_1/y]r)/x][e_1/y]R$. Using the rule (FIELD), we obtain that $G, (\Pi, \Pi_2) \vdash [e_1/y]t.f_i : \exists x : T.[e_1/y]R$, exactly what we wanted.
5. e is $\texttt{new}\ C(\overline{t})$. We know $(\Gamma, y : T_y); (\Pi, y \rightsquigarrow R_y) \vdash \texttt{new}\ C(\overline{t}) : \exists z : C.\texttt{unpacked}(z, \texttt{unique}(z)\ \texttt{in}\ Q_0(\overline{t})) \otimes \overline{f} \rightarrow \overline{t}$. We also know by inversion that $(\Gamma, y : T_y) \vdash \overline{t : T}$. Using the induction hypothesis we have $\Gamma \vdash \overline{[e_1/y]t : T}$. Using the rule (NEW), we obtain that
   $G, (\Pi, \Pi_2) \vdash [e_1/y]\texttt{new}\ C(\overline{t}) : \exists z :$
   $C.\texttt{unpacked}(z, \texttt{unique}(z)\ \texttt{in}\ Q_0(\overline{[e_1/y]t})) \otimes \overline{f} \rightarrow \overline{[e_1/y]t}$,
   exactly what we wanted.
6. e is $\texttt{if}(t, e_1, e_2)$. We know $(\Gamma, y : T_y); (\Pi, y \rightsquigarrow R_y) \vdash \texttt{if}(t, e_1, e_2)\exists x : T.R_1 \oplus R_2$. We also know by inversion that $(\Gamma, y : T_y); (\Pi, y \rightsquigarrow R_y, t = true) \vdash e_1 : \exists x : T.R_1$ and that $(\Gamma, y : T_y); (\Pi, y \rightsquigarrow R_y, t = false) \vdash e_1 : \exists x : T.R_2$. Using the induction hypothesis and knowing that $\Gamma, \Pi_2 \vdash e_0 : \exists x : T_y.R_y$, we have $\Gamma; (\Pi, \Pi_2, t = true) \vdash [e_0/y]e_1 : \exists x : T.[e_0/y]R_1$ and that $\Gamma; (\Pi, \Pi_2, t = false) \vdash [e_0/y]e_1 : \exists x : T.[e_0/y]R_2$. By applying the (IF) rule, we obtain that $\Gamma; (\Pi, \Pi_2) \vdash \texttt{if}(t, e_1, e_2)\exists x : T.[e_0/y]R_1 \oplus [e_0/y]R_2$. Since $[e_0/y]R_1 \oplus [e_0/y]R_2 = [e_0/y](R_1 \oplus R_2)$, $\Gamma; (\Pi, \Pi_2) \vdash \texttt{if}(t, e_1, e_2)\exists x : T.[e_0/y](R_1 \oplus R_2)$, exactly what we wanted.
7. e is $\texttt{let}\ x = e_1\ \texttt{in}\ e_2$. We know $(\Gamma, y : T_y); (\Pi, y \rightsquigarrow R_y) \vdash \texttt{let}\ x = e_1\ \texttt{in}\ e_2 : \exists w : T_2.[e_1/x]R$. We also know by inversion that $(\Gamma, y : T_y); (\Pi, y \rightsquigarrow R_y) \vdash e_1 : \exists x : T_1.R_1$. Using the induction hypothesis and knowing that $\Gamma, \Pi_2 \vdash e_0 : \exists x : T_y.R_y$, we obtain that $\Gamma; (\Pi, \Pi_2) \vdash [e_0/y]e_1 : \exists x : T_1.[e_0/y]R_1$. We also know, by inversion, that $R_1 \vdash \exists \overline{z} : \overline{T_z}.R_2$. This means that $[e_0/y]R_1 \vdash \exists \overline{z} : \overline{T_z}.[e_0/y]R_2$. The third thing that we know, by inversion, is that $(\Gamma, x : T_1, \overline{z} : \overline{T_z}, y : T_y), R_2 \vdash e_2 : \exists w : T_2.R$. This means that $(\Gamma, x : T_1, \overline{z} : \overline{T_z}, \Pi_2), [e_0/y]R_2 \vdash [e_0/y]e_2 : \exists w : T_2.[e_0/y]R$. Now, we can apply the (LET) rule and we obtain that $\Gamma; (\Pi, \Pi_2) \vdash [e_0/y](\texttt{let}\ x = e_1\ \texttt{in}\ e_2) : \exists w : T_2.[[e_0/y]e_1/x][e_0/y]R$. Since $[[e_0/y]e_1/x][e_0/y]R = [e_0/y][e_1/x]R$, we conclude that
   $\Gamma; (\Pi, \Pi_2) \vdash [e_0/y](\texttt{let}\ x = e_1\ \texttt{in}\ e_2) : \exists w : T_2.[e_0/y][e_1/x]R$, which is exactly what we wanted.
8. e is $\texttt{pack}\ r\ \texttt{to}\ Perm(r)\ in\ Q(\overline{r_1})\ \texttt{in}\ e$. We know that $(\Gamma, z : T_z); (\Pi_1, \Pi_2, z \rightsquigarrow R_z) \vdash \texttt{pack}\ r\ \texttt{to}\ Perm(r)\ in\ Q(\overline{r_1})\ \texttt{in}\ e :$

$$\frac{}{\mu, \rho, l \ \rightarrow \ \mu, \rho, \rho(l)} \ \textsc{Lookup}$$

$$\frac{o \notin dom(\mu) \quad \mu' = \mu[o \rightarrow C(\overline{\rho(l)})]}{\mu, \rho, new\ C(\bar{l}) \ \rightarrow \ \mu', \rho, o} \ \textsc{New}$$

$$\frac{\mu, \rho, e_1 \rightarrow \mu', \rho', e'}{\mu, \rho, let\ x = e_1\ in\ e_2 \ \rightarrow \ \mu', \rho', let\ x = e'\ in\ e_2} \ \textsc{Let-e}$$

$$\frac{l \notin dom(\rho)}{\mu, \rho, let\ x = o\ in\ e_2 \ \rightarrow \ \mu, \rho[l \rightsquigarrow o], [l/x]e_2} \ \textsc{Let-o}$$

$$\frac{v \in \{n, true, false\}}{\mu, \rho, let\ x = v\ in\ e_2 \ \rightarrow \ \mu, \rho, [v/x]e_2} \ \textsc{Let-v}$$

$$\frac{\mu(\rho(l_1)) = C(\overline{o}) \quad fields(C) = \overline{Tf}}{\mu, \rho, assign\ l_1.f := l_2 \ \rightarrow \ \mu[\rho(l_1) \rightsquigarrow [\rho(l_2)/o_i]C(\overline{o})], \rho, \rho(l_2)} \ \textsc{Assign}$$

$$\frac{}{\mu, \rho, if(true, e_1, e_2) \ \rightarrow \ \mu, \rho, e_1} \ \textsc{If-true}$$

$$\frac{}{\mu, \rho, if(false, e_1, e_2) \ \rightarrow \ \mu, \rho, e_2} \ \textsc{If-false}$$

$$\frac{\mu(\rho(l_1)) = C(\overline{o}) \quad method(m, C) = T_r\ m(\overline{x})\{return\ e\}}{\mu, \rho, l_1.m(\overline{l_2}) \ \rightarrow \ \mu, \rho, [l_1/this, \overline{l_2}/\overline{x}]e} \ \textsc{Invoke}$$

$$\frac{\mu(\rho(l)) = C(\overline{o}) \quad fields(C) = \overline{Tf} \quad \mu' = \mu + o_i : T_i}{\mu, \rho, l.f_i \ \rightarrow \ \mu', \rho, o_i} \ \textsc{Field}$$

$$\frac{}{\mu, \rho, pack\ r\ to\ R_1\ in\ e_1 \ \rightarrow \ \mu, \rho, e_1} \ \textsc{Pack}$$

$$\frac{}{\mu, \rho, unpack\ r\ from\ R_1\ in\ e_1 \ \rightarrow \ \mu, \rho, e_1} \ \textsc{Unpack}$$

$$\frac{n_1\ \widehat{binop}\ n_2 \ = n_3}{\mu, \rho, n_1\ binop\ n_2 \ \rightarrow \ \mu, \rho, n_3} \ \textsc{Binop}$$

$$\frac{bo_1\ \widehat{or}\ bo_2 \ = bo_3}{\mu, \rho, bo_1\ or\ bo_2 \ \rightarrow \ \mu, \rho, bo_3} \ \textsc{Or}$$

$$\frac{bo_1\ \widehat{and}\ bo_2 \ = bo_3}{\mu, \rho, bo_1\ and\ bo_2 \ \rightarrow \ \mu, \rho, bo_3} \ \textsc{And}$$

$$\frac{\widehat{not}\ bo_1 \ = bo_2}{\mu, \rho, not\ bo_1 \ \rightarrow \ \mu, \rho, bo_2} \ \textsc{Not}$$

$$\frac{\mathsf{localFields}(C) = \overline{f : T}}{\Gamma; \Pi \vdash \overline{f} : \exists \overline{x} : \overline{T}.none} \ \textsc{None}$$

Figure 1. Dynamic Semantics Rules

$\exists x : T.R$. We also know by inversion that $(\Gamma, z : T_z); (\Pi_2, z \rightsquigarrow R_z, Perm(r)\ in\ Q(\overline{r_1})) \vdash e : \exists x : T.R$. Using the induction hypothesis and knowing that $\Gamma, \Pi_3 \vdash e_1 : \exists x : T_z.R_z$, we obtain that

$\Gamma; (\Pi_2, \Pi_3, Perm(r)\ in\ Q(\overline{r_1})) \vdash [e_1/z]e : \exists x : T.[e_1/z]R$. The other two premises of the (PACK-SH-IMM) rule can also be obtained by inversion and they remain the same. So now we can apply the (PACK-SH-IMM) rule again and we get that

$\Gamma; (\Pi_1, \Pi_2, \Pi_3) \vdash \texttt{pack}\ r\ \texttt{to}\ Perm(r)\ in$

$Q(\overline{r_1})\ \texttt{in}\ [e_1/z]e : \exists x : T.[e_1/z]R$. All the free variables in Q have been replaced by the argument $\overline{r_1}$, there will be no more free z variables in $\texttt{pack}\ r\ \texttt{to}\ Perm(r)\ in\ Q(\overline{r_1})$, so $\texttt{pack}\ r\ \texttt{to}\ Perm(r)\ in\ Q(\overline{r_1})\ \texttt{in}\ [e_1/z]e : \exists x : T.[e_1/z]R = [e_1/z](\texttt{pack}\ r\ \texttt{to}\ Perm(r)\ in\ Q(\overline{r_1})\ \texttt{in}\ e)$. Thus, $\Gamma; (\Pi_1, \Pi_2, \Pi_3) \vdash$

$[e_1/z](\texttt{pack}\ r\ \texttt{to}\ Perm(r)\ in\ Q(\overline{r_1})\ \texttt{in}\ e) : \exists x : T.[e_1/z]R$, exactly what we wanted.

9. e is $\texttt{pack}\ r\ \texttt{to}\ \texttt{unique}(r)\ in\ Q_2(\overline{r_2})\ \texttt{in}\ e$. The proof in this case is analogous to the one for the previous case, but $Perm$ will be replaced by unique across the proof.

10. e is $\texttt{unpack}\ r\ \texttt{from}\ Perm(r)\ in\ Q(\overline{r_1})\ \texttt{in}\ e$. We know that $(\Gamma, z : T_z); (\Pi_0, \Pi_2, z \rightsquigarrow R_z) \vdash \texttt{unpack}\ r\ \texttt{from}\ Perm(r)\ in\ Q(\overline{r_1}$ $\exists x : T.R$. We also know by inversion that $(\Gamma, z : T_z, \overline{y} : \overline{T_y}); (\Pi_2, z \rightsquigarrow R_z, [\overline{r_1}/\overline{x}]R_1,$

$\texttt{unpacked}(r, Perm(r)\ in\ Q(\overline{r_1})) \vdash e : \exists x : T.R$. Using the induction hypothesis and knowing that $\Gamma, \Pi_3 \vdash e_1 : \exists x : T_z.R_z$, we obtain that

$(\Gamma, \overline{y} : \overline{T_y}); (\Pi_2, \Pi_3, [\overline{r_1}/\overline{x}]R_1,$

$\texttt{unpacked}(r, Perm(r)\ in\ Q(\overline{r_1})) \vdash [e_1/z]e : \exists x : T.[e_1/z]R$. The other premises of the (UNPACK-SH-UNI) rule can also be obtained by inversion and they remain the same. So now we can apply the (UNPACK-SH-UNI) rule again and we get that $\Gamma; (\Pi_0, \Pi_2, \Pi_3) \vdash \texttt{unpack}\ r\ \texttt{from}\ Perm(r)\ in\ Q(\overline{r_1})\ \texttt{in}\ [e_1/z]e : \exists x : T.[e_1/z]R$. All the free variables in Q have been replaced by the argument $\overline{r_1}$, there will be no more free z variables in $\texttt{unpack}\ r\ \texttt{from}\ Perm(r)\ in\ Q(\overline{r_1})$, so $\texttt{unpack}\ r\ \texttt{from}\ Perm(r)\ in\ Q(\overline{r_1})\ \texttt{in}\ [e_1/z]e = [e_1/z](\texttt{unpack}\ r\ \texttt{from}\ Perm(r)\ in\ Q(\overline{r_1})\ \texttt{in}\ e)$. Thus, $\Gamma; (\Pi_0, \Pi_2, \Pi_3) \vdash [e_1/z](\texttt{unpack}\ r\ \texttt{from}$

$Perm(r)\ in\ Q(\overline{r_1})\ \texttt{in}\ [e_1/z]e : \exists x : T.R)$, exactly what we wanted to prove.

11. e is unpack r from immutable(r) *in* $Q(\overline{r_1})$ in e. The proof in this case is analogous to the one for the previous case, but $Perm$ will be replaced by immutable across the proof.

12. e is $t_0.m(\bar{t})$. We know that $(\Gamma, y : T_y); (\Pi, y \rightsquigarrow R_y) \vdash t_0.m(\bar{t}) : \exists$ *result* $: T_r.[t_0/this][\bar{t}/\overline{x}]R$. We know by inversion that $(\Gamma, y : T_y); (\Pi, y \rightsquigarrow R_y) \vdash [t_0/this][\bar{t}/\overline{x}]R_1$. Using the induction hypothesis and knowing that $\Gamma, \Pi_3 \vdash e_1 : \exists x : T_y.R_y$, we obtain that

    $\Gamma; (\Pi, \Pi_3) \vdash [e_1/y]([t_0/this][\bar{t}/\overline{x}]R_1)$. This is equivalent to writing

    $\Gamma; (\Pi, \Pi_3) \vdash [t_0/this][\overline{[e_1/y]t}/\overline{x}][e_1/y]R_1$.

    By inversion we know that $(\Gamma, y : T_y) \vdash t_0 : C_0 \quad (\Gamma, y : T_y) \vdash \overline{t : T}$. Using the induction hypothesis we obtain that $\Gamma \vdash t_0 : C_0 \quad \Gamma \vdash \overline{[e_1/y]t : T}$. Also by inversion we know that $mtype(m, C_0) = \forall \overline{x : T}.\exists result : T_r.R_1' \multimap R$ and $R_1 implies R_1'$.

    We can infer that $[e_1/y]R_1 implies [e_1/y]R_1'$ and that $mtype(m, C_0) = \forall \overline{x : T}.\exists result :$

    $T_r.R_1' \multimap R$ will hold for $\overline{[e_1/y]t : T}$ (because of the $\forall$ quantifier of the (MTYPE) judgement. We can now apply the (CALL) rule again and we obtain that $\Gamma; (\Pi, \Pi_3) \vdash (([e_1/y]t_0).m(\overline{[e_1/y]t})) : \exists$ *result* $: T_r.[[e_1/y]t_0/this][\overline{[e_1/y]t}/\overline{x}][e_1/y]R$.

    Since $(t_0).m(\overline{[e_1/y]t}) = [e_1/y](t_0.m(\bar{t})$ and

    $[t_0/this][\overline{[e_1/y]t}/\overline{x}][e_1/y]R = [e_1/y]([t_0/this][\bar{t}/\overline{x}]R)$, we obtain that $\Gamma; (\Pi, \Pi_3) \vdash [e_1/y](t_0.m(\bar{t})) : \exists$ *result* $: T_r.[e_1/y]([t_0/this][\bar{t}/\overline{x}]R)$. This is exactly what we wanted to prove.

13. e is assign $t_1.f_i := t$. We know that $(\Gamma, y : T_y); (\Pi_1, y \rightsquigarrow R_y, \Pi_2, \Pi_3) \vdash (\texttt{assign}\ t_1.f_i := t)$:

    $\exists x : T_i.Perm'(x)\ in\ Q'(\overline{r'}) \otimes Perm_0(t)\ in\ Q_0(\overline{r_0}) \otimes p \otimes \ t_1.f_i \rightarrow t$. We know by inversion that $(\Gamma, y : T_y); (\Pi_1, y \rightsquigarrow R_y) \vdash t : \exists x : T_i.Perm_0(x)\ in\ Q_0(\overline{r_0})$. Using the induction hypothesis and knowing that $\Gamma, \Pi_4 \vdash e_1 : \exists x : T_y.R_y$, we obtain that $\Gamma; (\Pi_1, \Pi_4) \vdash [e_1/y]t : T_i.[e_1/y](Perm_0(t)\ in\ Q_0(\overline{r_0}))$. Since all the free variables in Q_0 have been replaced by $\overline{r_0}$,

    $[e_1/y](Perm_0(t)\,in\,Q_0(\overline{r_0})) = Perm_0([e_1/y]t)\,in\,Q_0(\overline{r_0})$. The other premises of the (ASSIGN) rule can also be obtained by inversion and they remain the same. So now we can apply the (ASSIGN) rule again and we get that $\Gamma; (\Pi_1, \Pi_4, \Pi_2, \Pi_3) \vdash \texttt{assign}\ t_1.f_i := [e_1/y]t \ : \exists x : T_i.(Perm'(x)\,in\,Q'(\overline{r'}) \otimes Perm_0([e_1/y]t)\,in\,Q_0(\overline{r_0}) \otimes p \otimes \ t_1.f_i \rightarrow [e_1/y]t)$. Since $\texttt{assign}\ t_1.f_i := [e_1/y]t = [e_1/y](\texttt{assign}\ t_1.f_i := t)$ and $(Perm'(x)\ in\ Q'(\overline{r'}) \otimes Perm_0([e_1/y]t)\ in\ Q_0(\overline{r_0}) \otimes p \otimes \ t_1.f_i \rightarrow [e_1/y]t) = [e_1/y](Perm'(x)\ in\ Q'(\overline{r'}) \otimes Perm_0(t)\ in\ Q_0(\overline{r_0}) \otimes p \otimes \ t_1.f_i \rightarrow t)$, we finally obtain that

    $\Gamma; (\Pi_1, \Pi_4, \Pi_2, \Pi_3) \vdash [e_1/y](\texttt{assign}\ t_1.f_i := t) \ : \exists x : T_i.[e_1/y](Perm'(x)\ in\ Q'(\overline{r'}) \otimes Perm_0(t)\ in\ Q_0(\overline{r_0}) \otimes p \otimes \ t_1.f_i \rightarrow t)$. This is exactly what we wanted to prove.

    We have now gone through all the induction cases and the proof of the Substitution Lemma is finished.

We also need to define the following lemma:

LEMMA 2.2. (Memory Consistency)

1. *If* $\mu, (\Sigma, l \rightsquigarrow (Q, i)), (\Pi, l \rightsquigarrow R), \rho\ \underline{ok}$ *then* $\mu, (\Sigma, \rho(l) \rightsquigarrow (Q, i)), (\Pi, \rho(l) \rightsquigarrow R), \rho\ \underline{ok}$, *where* $R = Perm(x)\ in\ Q$.
2. *If* $\mu, \Sigma, \Pi, \rho\ \underline{ok}$ *and* $o \notin dom(\mu)$ *and* $init(C) = \langle Q_0(\overline{x}) \rangle$ *then* $\mu[o \rightsquigarrow C(\overline{\rho(l)})], (\Sigma, o \rightsquigarrow (unpacked, i)), (\Pi, o \rightsquigarrow unpacked(o, unique(o)\ in\ Q_0(\bar{t}))), \rho\ \underline{ok}$.
3. *If* $\mu, (\Sigma, l \rightsquigarrow (Q, i)), (\Pi, l \rightsquigarrow R), \rho\ \underline{ok}$ *and* $l' \notin dom(\rho)$ *then* $\mu, (\Sigma, l' \rightsquigarrow (Q, i)), (\Pi, l' \rightsquigarrow R), \rho[l' \rightsquigarrow \rho(l)]\ \underline{ok}$, *where* $P = perm(x)\ in\ Q$.
4. *If* $\mu, (\Sigma_1, \Sigma_2), (\Pi_1, \Pi_2), \rho\ \underline{ok}$ *and* $unpacked(r, Perm(r)\ in\ Q(\overline{r_1})) \in \Pi_1$, *then* $\mu, (\Sigma_2, r \rightarrow (Q(\overline{r_1}), i)), (\Pi_2, r \rightsquigarrow Perm(r)\ in\ Q(\overline{r_1})), \rho\ \underline{ok}$, *where* $Perm =$ share *or* $Perm =$ immutable.
5. *If* $\mu, (\Sigma_1, \Sigma_2), (\Pi_1, \Pi_2), \rho\ \underline{ok}$, $unpacked(r, unique(r)\ in\ Q_1(\overline{r_1})) \in \Pi_1$ *and* $[\overline{r'}/\overline{z},\ \overline{r_2}/\overline{x}]R_2 \in \Pi_1$, *with* $Q_2(\overline{x}) = \exists \overline{z}.R_2 \in C$, *then* $\mu, (\Sigma_2, r \rightsquigarrow (Q_2(\overline{r_2}), i)), (\Pi_2, r \rightsquigarrow unique(x)\ in\ Q_2(\overline{r_2})), \rho\ \underline{ok}$.
6. *If* $\mu, (\Sigma_0, \Sigma_2), (\Pi_0, \Pi_2), \rho\ \underline{ok}$ *and* $Perm(r)\ in\ Q(\overline{r_1}) \in \Pi_0$ *and* $Q(\overline{x}) = \exists \overline{y}.R_1 \in C$ *and* $\forall r', \overline{x}, Perm' : (\ unpacked(r', Perm'(r')\ in\ Q(\overline{x})) \in (\Pi_0 \cup \Pi_2) \Rightarrow \Pi_0, \Pi_2 \vdash r \neq r')$ *then* $\mu, \Sigma' = (\Sigma_2, r \rightsquigarrow (unpacked, i))$, $\Pi' = (\Pi_2, [\overline{r_1}/\overline{x}]R_1$, $r \rightsquigarrow unpacked(x, Perm(x)\ in\ Q(\overline{r_1}))), \rho\ \underline{ok}$, *where* $Perm \in \{unique, share, immutable\}$. *If* $Perm =$ immutable *then all permissions present in* R_1 *must be immutable.*
7. *If* $\mu, \Sigma, \Pi, \rho\ \underline{ok}$ *and* $\mu(\rho(l)) = C(\overline{o})$ *and* $fields(C) = \overline{Tf}$ *then* $\mu' = (\mu + o_i : T_i), \Sigma' = (\Sigma, o_i \rightarrow (Q, j)), \Pi' = (\Pi, o_i \rightsquigarrow R), \rho\ \underline{ok}$, *where* $R = Perm(x)\ in\ Q$.
8. *If* $\mu, \Sigma = (\Sigma_0, l_2 \rightsquigarrow (Q'(\overline{r'}), j)), \Pi = (\Pi_0, l_2 \rightsquigarrow Perm'(y)\,in\,Q'(\overline{r'}) \otimes Perm_0(x)\,in\,Q_0(\overline{r_0}) \otimes p \otimes \ t_1.f_i \rightarrow x), \rho\ \underline{ok}$ *and* $\rho(l_2) = o_2$, *then* $\mu' = \mu[\rho(l_1) \rightsquigarrow [o_2/o_i]C(\overline{o})], \Sigma' = (\Sigma, o_2 \rightsquigarrow (Q'(\overline{r'}), j))$, $\Pi' = (\Pi, o_2 \rightsquigarrow Perm'(y)\ in\ Q'(\overline{r'}) \otimes Perm_0(x)\ in\ Q_0(\overline{r_0}) \otimes p \otimes \ t_1.f_i \rightarrow x), \rho\ \underline{ok}$

Proof of memory consistency lemma

1. Environment map

   Assuming $\mu, (\Sigma, l \rightsquigarrow (Q, i)), (\Pi, l \rightarrow R), \rho\ \underline{ok}$ we need to show that $\mu, (\Sigma, \rho(l) \rightsquigarrow Q), (\Pi, \rho(l) \rightsquigarrow R), \rho\ \underline{ok}$, where $R = Perm(x)\ in\ Q$. Memory does not change. The only object potentially affected is $\rho(l)$, which is equal to o, say. Since $props(\mu, (\Sigma, l \rightsquigarrow Q), (\Pi, l \rightsquigarrow R), \rho, o) = props(\mu, (\Sigma, o \rightsquigarrow Q), (\Pi, o \rightsquigarrow P), \rho, o)$, we can conclude that $\mu, (\Sigma, \rho(l) \rightarrow Q), (\Pi, \rho(l) \rightsquigarrow R), \rho \vdash o\ \underline{ok}$, and therefore $\mu, (\Sigma, o \rightsquigarrow Q), (\Pi, o \rightsquigarrow R), \rho\ \underline{ok}$.

2. New object

   Assuming μ, Σ, Π, ρ $\underline{ok}$ and $o \notin dom(\mu)$, we have to show that $\mu' = \mu[o \rightsquigarrow C(\overline{\rho(l)})], \Sigma' = (\Sigma, o \rightsquigarrow (\mathsf{unpacked}, i)), \Pi' = (\Pi, o \rightsquigarrow \mathsf{unpacked}(o, \mathsf{unique}(o)\ \mathtt{in}\ Q_0(\overline{t}))), \rho$ $\underline{ok}$.
   It must be that $\overline{\rho(l) = o'}$ for some objects o'. We know that $init(C) = \langle Q_0(\overline{x}) \rangle$. This means that $Q_0(\overline{x})$ is of the form $\overline{f \rightsquigarrow x}$ (this is a requierement for the initial predicate in each class) and when the predicate Q_0 is unpacked, the heap invariants will not be affected . The only objects affected are $o, \overline{o'}$. Since $\overline{\mu(o') = \mu'(o')}$ and $props(\mu, \Sigma, \Pi, \rho, o') = props(\mu', \Sigma', \Pi', \rho, o')$ we can deduce that
   $\overline{\mu', \Sigma', \Pi', \rho \vdash o'\ \underline{ok}}$.

   The only object proposition refering to o in Π' is
   $\mathsf{unpacked}(o, \mathsf{unique}(o)\ \mathtt{in}\ Q_0(\overline{t}))$, which means that the heap invariants are satisfied and we can deduce that $\mu', \Sigma', \Pi', \rho \vdash o\ \underline{ok}$. Thus, $\mu', \Sigma', \Pi', \rho\ \underline{ok}$.

3. Environment rename

   Assuming $\mu, (\Sigma, l \rightsquigarrow (Q, i)), (\Pi, l \rightsquigarrow R), \rho$ $\underline{ok}$ and $l' \notin dom(\rho)$, we have to show that $\mu, (\Sigma, l' \rightsquigarrow (Q, i)), (\Pi, l' \rightsquigarrow R), \rho[l' \rightsquigarrow \rho(l)]$ $\underline{ok}$, where $R = perm(x)\ in\ Q$. The only object affected can be $\rho(l)$. By the same argument above, that the $props$ sets are identical, w can conclude that $\mu, (\Sigma, l' \rightsquigarrow (Q, i)), (\Pi, l' \rightsquigarrow R), \rho[l' \rightsquigarrow \rho(l)]$ $\underline{ok}$.

4. Packing to Perm

   Assuming $\Omega_1 = [\mu, \Sigma = (\Sigma_1, \Sigma_2), \Pi = (\Pi_1, \Pi_2), \rho]$ $\underline{ok}$, we have to show that $\Omega_2 = [\mu, \Sigma' = (\Sigma_2, r \rightsquigarrow (Q(\overline{r_1}), i)), \Pi' = (\Pi_2, r \rightsquigarrow Perm(r)\ in\ Q(\overline{r_1})), \rho]$ $\underline{ok}$, where $Perm = \mathsf{share}$ or $Perm = \mathsf{immutable}$. Let's take an arbitrary o. Since μ and ρ don't change, the only changes in the $\overline{P}$'s corresponding to Ω_1 and to Ω_2 come from the different $o \rightsquigarrow R$ extracted from Π and from Π'. We have to show that the heap invariants are preserved by the different $o \rightsquigarrow R$ in Π', knowing that the invariants are preserved by the different $o \rightsquigarrow R$ in (Π_1, Π_2). Knowing this, we deduce that the invariants cannot be broken by the assertions in Π_2. Thus, we only have to see if $r \rightsquigarrow Perm(x)\ in\ Q(\overline{r_1})$ is in contradiction with any assertions about r in Π_2. We also know that $unpacked(r, Perm(r)\ in\ Q(\overline{r_1})$ is in Π_1.

   Since Ω_1 $\underline{ok}$, the only object proposition in Π_2 about r has to be $Perm(r)\ in\ Q(\overline{r_1})$, according to the heap invariants. Thus, $(\Pi_2, r \rightsquigarrow Perm(r)\ in\ Q(\overline{r_1}))$ satisfies the heap invariants, Σ' is compatible with Π' and the primitives are preserved, so μ, Σ', Π', ρ $\underline{ok}$.

5. Packing to unique

   Assuming $\Omega_1 = [\mu, (\Sigma_1, \Sigma_2), (\Pi_1, \Pi_2), \rho]$ $\underline{ok}$, we have to show that $\Omega_2 = [\mu, \Sigma' = (\Sigma_2, r \rightsquigarrow (Q_2(\overline{r_2}), i)), \Pi' = (\Pi_2, r \rightsquigarrow \mathsf{unique}(r)\ in\ Q_2(\overline{r_2})), \rho]$ $\underline{ok}$,

   where $unpacked(r, \mathsf{unique}(r)\ in\ Q_1(\overline{r_1})) \in \Pi_1$ and $[\overline{r'}/\overline{z},\ \overline{r_2}/\overline{x}]R_2 \in \Pi_1$, with $Q_2(\overline{x}) = \exists \overline{z}.R_2 \in C$. Let's take an arbitrary o. Since μ and ρ don't change, the only changes in the $\overline{R}$'s corresponding to Ω_1 and to Ω_2 come from the different $o \rightsquigarrow R$ extracted from (Π_1, Π_2) and from $(\Pi_2, r \rightsquigarrow \mathsf{unique}(r)\ in\ Q_2(\overline{r_2}))$. We have to show that the heap invariants are preserved by the different $o \rightsquigarrow R$ in $(\Pi_2, r \rightsquigarrow \mathsf{unique}(r)\ in\ Q_2(\overline{r_2}))$, knowing that the invariants are preserved by the different $o \rightsquigarrow R$ in (Π_1, Π_2). Knowing this, we deduce that the invariants cannot be broken by the assertions in Π_2. Thus, we only have to see if $r \rightsquigarrow \mathsf{unique}(x)\ in\ Q_2(\overline{r_2})$ is in contradiction with any assertions about r in Π_2. We also know that

   $unpacked(r, \mathsf{unique}(r)\ in\ Q_1(\overline{r_1})) \in \Pi_1$.

   Since Ω_1 $\underline{ok}$, the only object proposition in Π_2 about r has to be $none(r)\ in\ Q_3(\overline{r_3})$, according to the heap invariants. It follows that

   $(\Pi_2, r \rightsquigarrow \mathsf{unique}(r)\ in\ Q_2(\overline{r_2}))$ satisfies the heap invariants. Since $[\overline{r'}/\overline{z},\ \overline{r_2}/\overline{x}]R_2 \in \Pi_1$ and the primitives corresponding to (Π_1, Π_2) are ok, there can be no primitives in Π_2 that contradict $[\overline{r'}/\overline{z},\ \overline{r_2}/\overline{x}]R_2$. We know that $Q_2(\overline{x}) = \exists \overline{z}.R_2 \in C$ and we can deduce that the primitives corresponding to μ, Σ', Π', ρ are ok. Thus μ, Σ', Π', ρ $\underline{ok}$.

6. Unpacking from Perm

   Assuming $\Omega_1 = [\mu, \Sigma = (\Sigma_0, \Sigma_2), \Pi = (\Pi_0, \Pi_2), \rho]$ $\underline{ok}$, we have to show that

   $\Omega_2 = [\mu, \Sigma' = (\Sigma_2, r \rightsquigarrow (unpacked, i)),$
   $\Pi' = (\Pi_2, [\overline{r_1}/\overline{x}]R_1,$
   $r \rightsquigarrow \mathsf{unpacked}(x, Perm(x)\ in\ Q(\overline{r_1}))), \rho]$ $\underline{ok}$.

   Let's take an arbitrary o. Since μ and ρ don't change, the only changes in the $\overline{P}$'s corresponding to Ω_1 and to Ω_2 come from the different $o \rightsquigarrow R$ extracted from (Π_0, Π_2) and from Π'. We have to show that the heap invariants are preserved by the different $o \rightsquigarrow R$ in Π', knowing that the invariants are preserved by the different $o \rightsquigarrow R$ in Π. Knowing this, we deduce that the invariants cannot be broken by the assertions in Π_2. Thus, we only have to see if $r \rightsquigarrow \mathsf{unpacked}(x, Perm(x)\ in\ Q(\overline{r_1}))$ and $[\overline{r_1}/\overline{x}]R_1$ are in contradiction with any assertions about r in Π_2.

   Since $\forall r', \overline{x}, Perm' : (\ unpacked(r', Perm'(r')\ in\ Q(\overline{x})) \in (\Pi_0 \cup \Pi_2) \Rightarrow \Pi_0, \Pi_2 \vdash r \neq r')$ the heap invariants allow us to infer that Π_2 does not contain any object that is unpacked from the predicate Q and aliases with r. We also know that $Perm(r)\ in\ Q(\overline{r_1}) \in \Pi_0$. Using the heap invariants, we deduce that if there is an object proposition refering to r in Π_2, this object proposition must be $Perm(r)\ in\ Q(\overline{r_1})$ or $none(r)\ in\ Q_2(\overline{r_2})$. The $none(r)\ in\ Q_2(\overline{r_2})$ object proposition expresses the fact that the predicate $Q_2(\overline{r_2})$ holds of r, but there is no alias to r that can conflict with other aliases. This $none$ permission can be ignored in our proof.

The formula $[\overline{r_1}/\overline{x}]R_1$ corresponds to r, after it got unpacked. In this formula there might be object propositions refering to r or to other references that appear in Π_2. Since r was packed to Q, using object propositions from Π_0, right before being unpacked and since $Q(\overline{x}) = \exists \overline{y}.R_1$, we deduce that $[\overline{r_1}/\overline{x}]R_1$ will only contain object propositions that are already in Π_0. This means that the different $o \rightsquigarrow R$ extracted from (Π_0, Π_2) are compatible with each other and with $r \rightsquigarrow \mathsf{unpacked}(x, Perm(x)\ in\ Q(\overline{r_1}))$ (same reasoning as in the previous paragraph). If $Perm = \mathsf{immutable}$ then all permissions present in R_1 are immutable and thus the heap invariants will hold in this case also.

The heap invariants hold of Π' because: there is no object that aliases with r that is unpacked from Q in Π', and also because $r \rightsquigarrow \mathsf{unpacked}(x, Perm(x)\ in\ Q(\overline{r_1}))$, $Perm(r)\ in\ Q(\overline{r_1})$ and $[\overline{r_1}/\overline{x}]R_1$ do not contain object propositions or primitives that are not compatible. Thus, $\mu, \Sigma', \Pi', \rho\ \underline{ok}$

7. Field read

   Assuming $\mu, \Sigma, \Pi, \rho\ \underline{ok}$ and $\mu(\rho(l)) = C(\overline{o})$ and $fields(C) = \overline{Tf}$, we have to show that $\mu' = (\mu + o_i : T_i), \Sigma' = (\Sigma, o_i \rightsquigarrow (Q, j)), \Pi' = (\Pi, o_i \rightsquigarrow R), \rho\ \underline{ok}$, where $R = Perm(x)\ in\ Q$. The only object affected is o_i. Because of the way $fieldProps(\mu', \Sigma')$ is defined, any object proposition about o_i will be extracted from the object propositions refering to $\mu(\rho(l))$, which are already in Π. This means that $props(\mu, \Sigma, \Pi, \rho, o_i) = props(\mu', \Sigma', \Pi', \rho, o_i)$ and $\mu', \Sigma', \Pi', \rho \vdash o_i\ \underline{ok}$. Thus $\mu', \Sigma', \Pi', \rho\ \underline{ok}$.

8. Assignment

   Assuming $\mu, \Sigma = (\Sigma_0, l_2 \rightsquigarrow (Q'(\overline{r'}), j)), \Pi = (\Pi_0, l_2 \rightsquigarrow Perm'(y)\ in\ Q'(\overline{r'}) \otimes Perm_0(x)\ in\ Q_0(\overline{r_0}) \otimes p \otimes\ t_1.f_i \rightarrow x), \rho\ \underline{ok}$ and $\rho(l_2) = o_2$, we have to prove that $\mu' = \mu[\rho(l_1) \rightsquigarrow [o_2/o_i]C(\overline{o})], \Sigma' = (\Sigma, o_2 \rightsquigarrow (Q'(\overline{r'}), j)), \Pi' = (\Pi, o_2 \rightsquigarrow Perm'(y)\ in\ Q'(\overline{r'}) \otimes Perm_0(x)\ in\ Q_0(\overline{r_0}) \otimes p \otimes\ t_1.f_i \rightarrow x), \rho\ \underline{ok}$. The only object that changes is o_i. Since $props(\mu, \Sigma, \Pi, \rho, o_i) = props(\mu', \Sigma', \Pi', \rho, o_i)$ and

   $\mu, \Sigma, \Pi, \rho \vdash o_i\ \underline{ok}$, we can conclude that $\mu', \Sigma', \Pi', \rho \vdash o_i\ \underline{ok}$ and thus $\mu', \Sigma', \Pi', \rho\ \underline{ok}$.

The proof for the Preservation Theorem is done by induction on the dynamic semantics rules. The rule ($\oplus$) can be applied as the first step in each derivation. This is because in the static rules Π could incorporate a number of Π_i, as we do not know which of the Π_i is the one that will be used at runtime.

Proof of the Preservation Theorem

Case (LOOKUP)

1. By assumption
   (a) $\Gamma, \Pi \vdash l : \exists\, x : T.R$
   (b) $\mu, \Sigma, \Pi, \rho\ \underline{ok}$
   (c) $\mu, \rho, l \rightarrow \mu, \rho, \rho(l)$
2. By inversion on 1a
   (a) $\Gamma = (\Gamma_1,\ l : T)$
   (b) $\Pi = (\Pi_1, l \rightsquigarrow R)$, where $R = Perm(x)\ in\ Q$
   (c) $\Sigma = (\Sigma_1, l \rightsquigarrow (Q, i))$, where Σ_1 is the store type corresponding to Π_1. i represents the index of Π_i that contains the R. The value of i will be determined at runtime.
3. $\mu, (\Gamma_1, l : T), (\Pi_1, l \rightsquigarrow R), (\Sigma_1, l \rightsquigarrow (Q, i))\ \underline{ok}$ -by 2
4. $\rho(l) = o$, for some o - by Object Proposition Consistency
5. Let $\Gamma' = (\Gamma, \rho(l) : T)$, $\Pi' = (\Pi_1, \rho(l) \rightsquigarrow R)$ and $\Sigma' = (\Sigma_1, \rho(l) \rightsquigarrow (Q, i))$
6. $(\Gamma, o : T), (\Pi_1, o \rightsquigarrow R) \vdash o : \exists x : T.R$ -by (TERM)
7. $\Gamma', \Pi' \vdash \rho(l) : \exists x : T.R$ -by 5,6
8. $\mu, (\Pi_1, \rho(l) \rightsquigarrow R), (\Sigma_1, \rho(l) \rightsquigarrow (Q, i)), \rho\ \underline{ok}$ -by 3,4, memory consistency lemma
9. $\mu, \Pi', \Sigma', \rho\ \underline{ok}$ -by 5, 8
10. q.e.d -by 7, 9

Case (NEW)

1. By assumption
   (a) $\Gamma, \Pi \vdash new\ C(\bar{l})\ : \exists y : T.R$
   (b) $\mu, \Sigma, \Pi, \rho\ \underline{ok}$
   (c) $\mu, \rho, new\ C(\bar{l}) \rightarrow \mu', \rho, o$
   (d) $o \notin dom(\mu)$
   (e) $\mu' = \mu[o \rightarrow C(\overline{\rho(l)})]$
2. By inversion on 1a
   (a) $\exists y : T.R = \exists z : C.[\mathsf{unpacked}(z, \mathsf{unique}(z)\ \mathtt{in}\ Q_0(\bar{t})) \otimes \overline{f} \rightarrow \bar{t}]$
   (b) $\Gamma = (\Gamma_1, \overline{l : T})$
   (c) $fields(C) = \overline{T\ f}$
   (d) $Q_0(\overline{x}) = R \in C$
   (e) $init(C) = \langle Q_0(\overline{x}) \rangle$
3. Let $\Gamma' = (G, o : C), \Pi' = (\Pi, o \rightsquigarrow [\mathsf{unpacked}(z, \mathsf{unique}(z)\ \mathtt{in}\ Q_0(\bar{t})) \otimes \overline{f} \rightarrow \bar{t}])$
4. Let $\Sigma' = (\Sigma, o \rightsquigarrow (\mathsf{unpacked}, i))$
5. $\Gamma', \Pi' \vdash o : \exists z : C.[\mathsf{unpacked}(z, \mathsf{unique}(z)\ \mathtt{in}\ Q_0(\bar{t})) \otimes \overline{f} \rightarrow \bar{t}]$ -by (TERM)
6. $\mu[o \rightsquigarrow C(\overline{\rho(l)})], (\Sigma, o \rightsquigarrow (\mathsf{unpacked}, i)), (\Pi, o \rightsquigarrow [\mathsf{unpacked}(z, \mathsf{unique}(z)\ \mathtt{in}\ Q_0(\bar{t})) \otimes \overline{f} \rightarrow \bar{t})], \rho\ \underline{ok}$ -by memory consistency lemma
7. q.e.d. -by 5, 6

Case (LET-O)

1. By assumption
   (a) $\Gamma, \Pi \vdash let\ x = o\ in\ e_2 : \exists\ y : T.R$
   (b) $\mu, \Sigma, \Pi, \rho\ \underline{ok}$
   (c) $\mu, \rho, let\ x = o\ in\ e_2 \rightarrow \mu, \rho[l \rightsquigarrow o], [l/x]e_2$
   (d) $l \notin dom(\rho)$
2. By inversion on 1a
   (a) $\Gamma; \Pi \vdash o : \exists x : T_1.R_1$
   (b) $R_1 \vdash \exists \overline{z} : \overline{T_z}.R_2$
   (c) $(\Gamma, x : T_1, \overline{z} : \overline{T_z}), R_2 \vdash e_2 : \exists y : T_2.R$
   (d) $\exists\ y : T.R = \exists\ w : T_2.[o/x]R$
3. $\Gamma = (\Gamma_1, o : T_1), \Pi = (\Pi_1, o \rightsquigarrow R_1)$ -by inversion on 2a
4. Also, $\Sigma = (\Sigma_1, o \rightsquigarrow (Q_1, i))$, where $R_1 = Perm(\overline{x})\ in\ Q_1$
5. Let $\Gamma' = (\Gamma, l : T_1), \Pi' = (R_2, l \rightsquigarrow R_1), \Sigma' = (\Sigma_2, l \rightsquigarrow (Q_1, i))$, where Σ_2 corresponds to R_2
6. $(\Gamma, l : T_1); (\Pi_1, l \rightsquigarrow P_1) \vdash [l/x]e_2 : \exists y : T_2.[l/x]R_2$ -by 1d, 2c, Substitution Lemma
7. $\Gamma', \Pi' \vdash e_2 : \exists\ y : T.R$ -by 6, 2d
8. $\mu, (\Sigma_2, o \rightsquigarrow (Q_1, i)), (\Pi_2, o \rightsquigarrow R_1), \rho\ \underline{ok}$ -by 2a, 2b
9. $\mu, (\Sigma_2, l \rightsquigarrow (Q_1, i)), (\Pi_2, l \rightsquigarrow R_1), \rho[l \rightsquigarrow o]\ \underline{ok}$ -by memory consistency lemma
10. $\mu, \Pi', \Sigma', \rho[l \rightsquigarrow o]\ \underline{ok}$
11. q.e.d. -by 10, 7

Case (LET-E)

1. By assumption
   (a) $\Gamma, \Pi \vdash let\ x = e_1\ in\ e_2 : \exists y : T.R$
   (b) $\mu, \Sigma, \Pi, \rho\ \underline{ok}$
   (c) $\mu, \rho, let\ x = e_1\ in\ e_2 \rightarrow \mu', \rho', let\ x = e_1'\ in\ e_2$
   (d) $\mu, \rho, e_1 \rightarrow \mu', \rho', e'$
2. By inversion on 1a
   (a) $\Gamma, \Pi \vdash e_1 : \exists x : T_1.R_1$
   (b) $R_1 \vdash \exists \overline{z} : \overline{T_z}.R_2$
   (c) $(\Gamma, x : T_1, \overline{z} : \overline{T_z}), R_2 \vdash e_2 : \exists y : T_2.R$
   (d) $\exists y : T.R = \exists w : T_2.[e_1/x]R$
3. By induction on 1b, 1d, 2a
   (a) $\exists \Gamma_0, \Pi'$ such that $\Gamma_0, \Pi' \vdash e' : \exists x : T_1.R_1$
   (b) $\exists \Sigma'$ ausch that $\mu', \Sigma', \Pi', \rho'\ \underline{ok}$
4. Let $\Gamma' = \Gamma \cup \Gamma_0$
5. $\Gamma', \Pi' \vdash let\ x = e_1'\ in\ e_2 : \exists y : T.R$ -by 3a,2c, 2d, (LET)
6. q.e.d. -by 3b,5

Case (PACK) Subcase: the static semantics rule corresponding to (PACK) is (PACK-SH-IMM).

1. By assumption
   (a) $\Gamma, \Pi \vdash pack\ r\ to\ R_1\ in\ e_1 : \exists x : T.R$
   (b) $\mu, \Sigma, \Pi, \rho\ \underline{ok}$
   (c) $\mu, \rho, pack\ r\ to\ R_1\ in\ e_1 \rightarrow \mu, \rho, e_1$
2. By inversion on (PACK-SH-IMM)
   (a) $(\Gamma, \overline{y} : \overline{T_y}); \Pi_1 \vdash r : T_1.[\overline{r'}/\overline{y},\ \overline{r_1}/\overline{x}]R_1 \otimes \mathsf{unpacked}(r, Perm(r)\ in\ Q(\overline{r_1}))$
   (b) $\Gamma; (\Pi_2, Perm(r)\ \textit{in}\ Q(\overline{r_1})) \vdash e : \exists x : T.R$
   (c) $Q(\overline{x}) = \exists \overline{y}.R_1$
   (d) $\Pi = (\Pi_1, \Pi_2)$
3. Let $\Pi' = (\Pi_2, r \rightsquigarrow Perm(x)\ \textit{in}\ Q(\overline{r_1}))$, $\Sigma' = (\Sigma_2, r \rightsquigarrow (Q(\overline{r_1}, i))$, $\Gamma' = \Gamma$
4. $\Gamma', \Pi' \vdash e : \exists x : T.R$ -by 2b, 3
5. $\mu, (\Sigma_2, r \rightsquigarrow (Q(\overline{r_1}), i)), (\Pi_2, r \rightsquigarrow Perm(x)\ \textit{in}\ Q(\overline{r_1})), \rho\ \underline{ok}$ -by memory consistency lemma
6. q.e.d. -by 4, 5

Subcase: the static semantics rule corresponding to (PACK) is (PACK-UNI).

1. By assumption
   (a) $\Gamma, \Pi \vdash pack\ r\ to\ R_1\ in\ e_1 : \exists x : T.R$
   (b) $\mu, \Sigma, \Pi, \rho\ \underline{ok}$
   (c) $\mu, \rho, pack\ r\ to\ R_1\ in\ e_1 \rightarrow \mu, \rho, e_1$
2. By inversion on (PACK-UNI)
   (a) $(\Gamma, \overline{z} : \overline{T_z}); \Pi_1 \vdash r : T_1.[\overline{r'}/\overline{z},\ \overline{r_2}/\overline{x}]R_2 \otimes \mathsf{unpacked}(r, \mathsf{unique}(r)\ in\ Q_1(\overline{r_1}))$
   (b) $\Gamma; (\Pi_2, \mathsf{unique}(r)\ \textit{in}\ Q_2(\overline{r_2})) \vdash e : \exists x : T.R$
   (c) $Q_1(\overline{x}) = R_1 \in C \qquad Q_2(\overline{x}) = \exists \overline{z}.R_2 \in C$
   (d) $\Pi = (\Pi_1, \Pi_2)$
3. Let $\Pi' = (\Pi_2, r \rightsquigarrow \mathsf{unique}(x)\ \textit{in}\ Q_2(\overline{r_2}))$, $\Sigma' = (\Sigma_2, r \rightsquigarrow (Q_2(\overline{r_2}), i)$, $\Gamma' = \Gamma$
4. $\Gamma', \Pi' \vdash e : \exists x : T.R$ -by 2b, 3
5. $\mu, (\Sigma_2, r \rightsquigarrow (Q_2(\overline{r_2}), i)), (\Pi_2, r \rightsquigarrow \mathsf{unique}(x)\ \textit{in}\ Q_2(\overline{r_2})), \rho\ \underline{ok}$ -by memory consistency lemma
6. q.e.d. -by 4, 5

Case (UNPACK) Subcase: the static semantics rule corresponding to (UNPACK) is (UNPACK-SH-UNI).

1. By assumption
   (a) $\Gamma, \Pi \vdash unpack\ r\ from\ R_1\ in\ e_1 : \exists x : T.R$
   (b) $\mu, \Sigma, \Pi, \rho\ \underline{ok}$
   (c) $\mu, \rho, unpack\ r\ from\ R_1\ in\ e_1 \rightarrow \mu, \rho, e_1$
2. By inversion on (UNPACK-SH-UNI)
   (a) $\Gamma; \Pi_0 \vdash r : T_1.Perm(r)\ in\ Q(\overline{r_1})$
   (b) $(\Gamma, \overline{y} : \overline{T_y}); (\Pi_2, [\overline{r_1}/\overline{x}]R_1,$

$\mathsf{unpacked}(r, Perm(r)\ in\ Q(\overline{r_1}))) \vdash e : \exists x : T.R$

(c) $\forall r', \overline{x}, Perm' : (\ unpacked(r', Perm'(r')\ in\ Q(\overline{x})) \in (\Pi_0 \cup \Pi_2) \Rightarrow \Pi_0, \Pi_2 \vdash r \neq r')$

(d) $Q(\overline{x}) = \exists \overline{y}.R_1 \in C \quad Perm \in \{\mathsf{unique}, \mathsf{share}\}$

(e) $\Pi = (\Pi_0, \Pi_2)$

3. Let $\Pi' = (\Pi_2, [\overline{r_1}/\overline{x}]R_1,$
$r \rightsquigarrow \mathsf{unpacked}(x, Perm(x)\ in\ Q(\overline{r_1}))),$
$\Sigma' = (\Sigma_2, r \rightsquigarrow (unpacked, i)), \Gamma' = (\Gamma, \overline{y} : \overline{T_y})$
4. $\Gamma', \Pi' \vdash e : \exists x : T.R$ -by 2b, 3
5. $\mu, (\Sigma_2, r \rightsquigarrow (unpacked, i)), (\Pi_2, [\overline{r_1}/\overline{x}]R_1, r \rightsquigarrow \mathsf{unpacked}(x, Perm(x)\ in\ Q(\overline{r_1}))), \rho\ \underline{ok}$ -by memory consistency lemma
6. q.e.d. -by 4, 5

Subcase: the static semantics rule corresponding to (UNPACK) is (UNPACK-IMM).

1. By assumption
   (a) $\Gamma, \Pi \vdash unpack\ r\ from\ R_1\ in\ e_1 : \exists x : T.R$
   (b) $\mu, \Sigma, \Pi, \rho\ \underline{ok}$
   (c) $\mu, \rho, unpack\ r\ from\ R_1\ in\ e_1 \rightarrow \mu, \rho, e_1$
2. By inversion on (UNPACK-IMM)
   (a) $\Gamma; \Pi_0 \vdash r : T_1.\mathsf{immutable}(r)\ in\ Q(\overline{r_1})$
   (b) $(\Gamma, \overline{y} : \overline{T_y}); (\Pi_2, [\overline{r_1}/\overline{x}]R_1,$
   $\mathsf{unpacked}(r, \mathsf{immutable}(r)\ in\ Q(\overline{r_1}))) \vdash e : \exists x : T.R$
   (c) $\forall r', \overline{x}, Perm' : (\ unpacked(r', Perm'(r')\ in\ Q(\overline{x})) \in (\Pi_0 \cup \Pi_2) \Rightarrow \Pi_0, \Pi_2 \vdash r \neq r')$
   (d) $Q(\overline{x}) = \exists \overline{y}.R_1 \in C$
   (e) all permissions present in R_1 must be immutable
   (f) $\Pi = (\Pi_0, \Pi_2)$
3. Let $\Pi' = (\Pi_2, [\overline{r_1}/\overline{x}]R_1,$
$r \rightsquigarrow \mathsf{unpacked}(x, \mathsf{immutable}(x)\ in\ Q(\overline{r_1}))),$
$\Sigma' = (\Sigma_2, r \rightsquigarrow (unpacked, i)), \Gamma' = (\Gamma, \overline{y} : \overline{T_y})$
4. $\Gamma', \Pi' \vdash e : \exists x : T.R$ -by 2b, 3
5. $\mu, (\Sigma_2, r \rightsquigarrow (unpacked, i)), (\Pi_2, [\overline{r_1}/\overline{x}]R_1, r \rightsquigarrow \mathsf{unpacked}(x, \mathsf{immutable}(x)\ in\ Q(\overline{r_1}))), \rho\ \underline{ok}$ -by memory consistency lemma
6. q.e.d. -by 4, 5

Case (IF-TRUE)

1. By assumption
   (a) $\Gamma, \Pi \vdash\ if(true, e_1, e_2) : \exists x : T.R$
   (b) $\mu, \Sigma, \Pi, \rho\ \underline{ok}$
   (c) $\mu, \rho, if(true, e_1, e_2) \rightarrow \mu, \rho, e_1$
   (d) $\exists x : T.R = \exists x : T.R_1 \oplus R_2$
2. By inversion on the static semantics rule (IF): $\Gamma, \Pi \vdash e_1 : \exists x : T.R_1$
3. Let $\Gamma' = \Gamma, \Pi' = \Pi, \Sigma' = \Sigma$
4. $R_1 \oplus R_2$ is true if R_1 is true or if R_2 is true
5. $\Gamma', \Pi' \vdash e_1 : \exists x : T.R_1 \oplus R_2$ -by 2,3,4
6. $\mu, \Sigma', \Pi', \rho\ \underline{ok}$ -by 3,1b
7. q.e.d. -by 5,6

Case (IF-FALSE)

1. By assumption
   (a) $\Gamma, \Pi \vdash\ if(false, e_1, e_2) : \exists x : T.R$
   (b) $\mu, \Sigma, \Pi, \rho\ \underline{ok}$
   (c) $\mu, \rho, if(false, e_1, e_2) \rightarrow \mu, \rho, e_2$
   (d) $\exists x : T.R = \exists x : T.R_1 \oplus R_2$
2. By inversion on the static semantics rule (IF): $\Gamma, \Pi \vdash e_2 : \exists x : T.R_2$
3. Let $\Gamma' = \Gamma, \Pi' = \Pi, \Sigma' = \Sigma$
4. $R_1 \oplus R_2$ is true if R_1 is true or if R_2 is true
5. $\Gamma', \Pi' \vdash e_2 : \exists x : T.R_1 \oplus R_2$ -by 2,3,4
6. $\mu, \Sigma', \Pi', \rho\ \underline{ok}$ -by 3,1b
7. q.e.d. -by 5,6

Case (FIELD)

1. By assumption
   (a) $\Gamma, \Pi \vdash l.f_i : \exists x : T_r.R$
   (b) $\mu, \Sigma, \Pi, \rho\ \underline{ok}$
   (c) $\mu, \rho, l.f_i\ \rightarrow\ \mu', \rho, o_i$
   (d) $\mu' = \mu + o_i : T_i$
   (e) $\mu(\rho(l)) = C(\overline{o})$
   (f) $fields(C) = \overline{Tf}$
   (g) $T_r = T_i$
2. By inversion on the static semantics rule (FIELD)
   (a) $l.f_i : T_r\ is\ a\ local\ field\ of\ C$
   (b) $\Gamma; \Pi \vdash [l.f_i/x]R$
3. Let $\Gamma' = (\Gamma, o_i : T_i), \Pi' = (\Pi, o_i \rightsquigarrow R)$
4. Let $\Sigma' = (\Sigma, o_i \rightsquigarrow (Q, j))$, where $R = Perm(x)\ in\ Q$
5. $\Gamma', \Pi' \vdash o_i : \exists x : T_i.R$ -by (TERM)
6. $\mu' = (\mu + o_i : T_i), \Sigma' = (\Sigma, o_i \rightsquigarrow (Q, j)), \Pi' = (\Pi, o_i \rightsquigarrow R)), \rho\ \underline{ok}$ -by memory consistency lemma
7. q.e.d. -by 5,6

Case (ASSIGN)

1. By assumption
   (a) $\Gamma, \Pi \vdash assign\ l_1.f := l_2 : \exists x : T.R$
   (b) $\mu, \Sigma, \Pi, \rho\ \underline{ok}$
   (c) $\mu, \rho, assign\ l_1.f := l_2\ \rightarrow$

$\mu[\rho(l_1) \rightsquigarrow [\rho(l_2)/o_i]C(\overline{o})], \rho, \rho(l_2)$

(d) $\mu(\rho(l_1)) = C(\overline{o})$

(e) $fields(C) = \overline{Tf}$

2. By inversion on the static semantics rule (ASSIGN)

(a) $\Gamma; \Pi_1 \vdash l_2 : \exists x : T_i.Perm_0(x)\ in\ Q_0(\overline{r_0})$, thus $T = T_i$

(b) $\Gamma; \Pi_2 \vdash l_1.f : T_i.Perm'(r_i)\ in\ Q'(\overline{r'}) \otimes p$

(c) $p = \mathsf{unpacked}(l_1, Perm(l_1)\ in\ Q(\overline{r}))$

(d) $\Pi_3 \vdash l_1.f \rightarrow r_i$

(e) $Perm \neq immutable$

(f) $\exists x : T.R = \exists x : T_i.Perm'(x)\ in\ Q'(\overline{r'}) \otimes Perm_0(l_2)\ in\ Q_0(\overline{r_0}) \otimes p \otimes\ l_1.f \rightarrow l_2$

(g) $\Pi = (\Pi_1, \Pi_2, \Pi_3)$

3. $\exists\ o_2$ such that $\rho(l_2) = o_2$.

4. Let $\Gamma' = (G, o_2 : T_i), \Pi' = (\Pi, o_2 \rightsquigarrow Perm'(x)\ in\ Q'(\overline{r'}) \otimes Perm_0(l_2)\ in\ Q_0(\overline{r_0}) \otimes p \otimes\ t_1.f_i \rightarrow l_2), \Sigma' = (\Sigma, o_2 \rightsquigarrow (Q'(\overline{r'}), j))$.

5. $\Gamma', \Pi' \vdash o_2 : \exists x : T_i.R$ -by (TERM)

6. $\mu' = \mu[\rho(l_1) \rightsquigarrow [\rho(l_2)/o_i]C(\overline{o})], \Sigma', \Pi', \rho\ \underline{ok}$ -by memory consistency lemma

7. q.e.d. -by 5, 6

Case (INVOKE)

(a) By assumption

i. $\Gamma, \Pi \vdash l_1.m(\overline{l_2}) : \exists x : T.R'$

ii. $\mu, \Sigma, \Pi, \rho\ \underline{ok}$

iii. $\mu, \rho, l_1.m(\overline{l_2}) \rightarrow \mu, \rho, [l_1/this, \overline{l_2}/\overline{x}]e$

iv. $\vdash PR$

v. $\mu(\rho(l_1)) = C(\overline{o})$

vi. $method(m, C) = T_r\ m(\overline{x})\{return\ e\}$

(b) By inversion on the static semantics rule (CALL)

i. $\Gamma \vdash l_1 : C \quad \Gamma \vdash \overline{l_2 : T}$

ii. $\Gamma; \Pi \vdash [l_1/this][\overline{l_2}/\overline{x}]R_1$

iii. $mtype(m, C) = \forall \overline{x : T}.\exists result : T_r.R_1' \multimap R$

iv. $R_1 \mathit{implies} R_1'$

v. $\exists x : T.R' = \exists\ \mathit{result} : T_r.[l_1/this][\overline{l_2}/\overline{x}]R$

(c) From 7(a)iv we know that the body $\{return\ e\}$ of the method m implements its specification, so the result will be of the type $\exists x : T_r.R'$, given the arguments of the right type.

(d) By the substitution Lemma, we know that $[l_1/this, \overline{l_2}/\overline{x}]e$ will be of the type $\exists x : T_r.R'$. Since μ, Σ, Π, ρ do not change, $\mu, \Sigma, \Pi, \rho\ \underline{ok}$.

(e) q.e.d., by 7d, 7(a)iv.

The cases LET-V, BINOP, AND, OR, NOT are trivial and they preserve soundness. After having proved the Preservation Theorem, we should prove the Progress Theorem for our soundness proof to be complete. Since our system is similar to Featherweight Java and we did not add new features to the language, the Progress Theorem automatically holds. This concludes out soundness proof.