Verifying Object-Oriented Code Using Object Propositions

Ligia Nistor+  Jonathan Aldrich+  Hannes Mehnert*

School of Computer Science, Carnegie Mellon University+
*IT University of Copenhagen

{[nistor,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 \((\Gamma, y : T_y), (\Pi_1, y \leadsto R_y) \vdash e \equiv e_1 : \exists x : T, R_{\Gamma}y\) then 
\((\Gamma, (\Pi_1, \Pi_2)) \vdash e_1/y[e : \exists x : T, e_1/y]R_y\).

Proof of Substitution Lemma
The proof is by induction on the derivations of \((\Gamma, y : T_y), (\Pi_1, y \leadsto R_y) \vdash e \equiv e_1 : T, R_{\Gamma}y\).

1. \(e\) is a value \(v\). The values that \(e\) can take in this case are \(\text{true}\), \(\text{false}\), or \(\text{nil}\). We know \((\Gamma, y : T_y), (\Pi_1, y \leadsto R_y) \vdash e = v : \exists x : T, R_{\Gamma}y\) and we see that \(R\) might contain object propositions referring to \(y\), which will have to be substituted when \(y\) is not in the linear context any more. Since \(e_1/y[e = v, \Gamma, (\Pi_1, \Pi_2) \vdash v \equiv \exists x : T, e_1/y]R_{\Gamma}y\) directly.

2. \(e\) is a variable \(z \neq y\). We know \((\Gamma, y : T_y), (\Pi_1, y \leadsto R_y) \vdash z \equiv \exists x : T, R_{\Gamma}y\). We see that \(R\) might contain object propositions referring to \(y\), which will have to be substituted when \(y\) is not in the linear context any more. Since \(e_1/y\equiv z\), we conclude that \(\Gamma, (\Pi_1, \Pi_2) \vdash \exists x : T, e_1/y]R_{\Gamma}y\).

3. \(e\) is the variable \(y\). Now \(e_1/y[e = e_1 \equiv \exists x : T, e_1/y]R_{\Gamma}y\).

4. \(e\) is \(t.f_i\). We know \((\Gamma, y : T_y), (\Pi_1, y \leadsto R_y) \vdash t.f_i : \exists x : T, R_{\Gamma}y\) and \(\Pi_1, y \leadsto R_y\) \vdash t.f_i \rightarrow r\). We also know by inversion that \(\Pi_1, y \leadsto R_y\) \vdash t.f_i \rightarrow r\) and that \((\Gamma, y \leadsto R_y) \vdash r \rightarrow R_{\Gamma}y\). The third thing that we know, by inversion, is that \(\Gamma, (\Pi_1, \Pi_2) \vdash \exists x : T, e_1/y]R_{\Gamma}y\).

5. \(e\) is \text{new} \(C(T)\). We know \((\Gamma, y : T_y), (\Pi_1, y \leadsto R_y) \vdash \text{new} \(C(T)\) \equiv \exists z : C.unpacked(z, \text{unique}(z) \in Q_0(T)) \rightarrow T_{\Gamma}y\). We also know by inversion that \((\Gamma, y : T_y) \vdash t : T_{\Gamma}y\). The induction hypothesis we have \(\Gamma, (\Pi_1, \Pi_2) \vdash \exists x : T, e_1/y]R_{\Gamma}y\). Using the rule (NEW), we obtain that \(G, (\Pi_1, \Pi_2) \vdash \exists z \in Q_0(e_1/y)\) exactly what we wanted.

6. \(e\) is \text{if}(t, e_1, e_2). We know \((\Gamma, y : T_y), (\Pi_1, y \leadsto R_y) \vdash \text{if}(t, e_1, e_2) \equiv \exists x : T, e_1/y]R_{\Gamma}y\) and \(\exists x : T, \Pi_1, \Pi_2 \vdash \exists x : T, e_1/y]R_{\Gamma}y\). The third thing that we know, by inversion, is that \(\Gamma, (\Pi_1, \Pi_2) \vdash \exists x : T, e_1/y]R_{\Gamma}y\).

7. \(e\) is \text{let} \(x = e_1 \in e_2\). We know \((\Gamma, y : T_y), (\Pi_1, y \leadsto R_y) \vdash \text{let} \(x = e_1 \in e_2 : \exists w : T_{\Gamma}e_1/y]R_{\Gamma}y\).

8. \(e\) is \text{pack} \(r \rightarrow \text{Perm}(r)\) in \(Q(\Pi)\). We know that \((\Gamma, z : T_z), (\Pi_1, \Pi_2, z \leadsto R_z) \vdash \text{pack} \(r \rightarrow \text{Perm}(r)\) in \(Q(\Pi)\).
\[ \begin{array}{ll}
\frac{\mu, \rho, l \rightarrow \mu, \rho, l}{\text{LOOKUP}} \\
\frac{\mu, \rho, e_1 \rightarrow \mu', \rho', e'}{\mu, \rho, l \rightarrow \mu', \rho', l', e'} & \text{LET-E} \\
\frac{v \in \{n, \text{true}, \text{false}\}}{\mu, \rho, \text{let } x = e_1 \rightarrow \mu, \rho, \text{let } x = e'_1 \rightarrow e_2} & \text{LET-V} \\
\frac{\mu, \rho, \text{if (true, } e_1, e_2) \rightarrow \mu, \rho, e_1}{\mu, \rho, \text{if (false, } e_1, e_2) \rightarrow \mu, \rho, e_2} & \text{IF-TRUE} \\
\frac{\mu(l_1) = C(\overline{\sigma})}{\mu, \rho, l_1.m(l_2) \rightarrow \mu, \rho, l_2} & \text{INVOKED} \\
\frac{n_1 \text{ binop } n_2 = n_3}{\mu, \rho, n_1 \text{ binop } n_2 \rightarrow \mu, \rho, n_3} & \text{BINOP} \\
\frac{bo_1 \text{ and } bo_2 = bo_3}{\mu, \rho, bo_1 \text{ and } bo_2 \rightarrow \mu, \rho, bo_3} & \text{AND} \\
\frac{\text{localFields}(C) = f : T}{\Gamma; \Pi \vdash f : \exists \overline{x} : \overline{T} \text{ none}} & \text{NONE} \\
\end{array} \]

\[ \begin{array}{ll}
\frac{o \notin \text{dom}(\mu)}{\mu' = \mu[o \rightarrow C(\overline{\rho(l)})]} & \text{NEW} \\
\frac{l \notin \text{dom}(\rho)}{\mu, \rho, l \rightarrow o \rightarrow e_2 \rightarrow \mu, \rho, l \rightarrow o \rightarrow e_2} & \text{LET-O} \\
\frac{\mu(\rho(l_1) = C(\overline{\sigma}))}{\mu, \rho, \text{assign } l := l_2 \rightarrow \mu, \rho, l \rightarrow o \rightarrow e_2} & \text{ASSIGN} \\
\frac{\mu(\rho(l)) = C(\overline{\sigma})}{\mu, \rho, \text{if } (false, e_1, e_2) \rightarrow \mu, \rho, e_2} & \text{IF-FALSE} \\
\frac{\mu(\rho(l)) = C(\overline{\sigma})}{\mu, \rho, \text{if } (true, e_1, e_2) \rightarrow \mu, \rho, e_1} & \text{IF-TRUE} \\
\frac{\mu(\rho(l)) = C(\overline{\sigma})}{\mu, \rho, \text{pack } r \rightarrow R_1 \rightarrow \mu, \rho, e_1} & \text{PACK} \\
\frac{\mu, \rho, \text{unpack } r \rightarrow R_1 \rightarrow \mu, \rho, e_1}{\mu, \rho, \text{unpack } r \rightarrow R_1 \rightarrow \mu, \rho, e_1} & \text{UNPACK} \\
\frac{bo_1 \text{ or } bo_2 = bo_3}{\mu, \rho, bo_1 \text{ or } bo_2 \rightarrow \mu, \rho, bo_3} & \text{OR} \\
\frac{\neg bo_1 = bo_2}{\mu, \rho, \neg bo_1 \rightarrow \mu, \rho, bo_2} & \text{NOT} \\
\end{array} \]

**Figure 1. Dynamic Semantics Rules**

1. \( \exists x : T.R \). We also know by inversion that \((\Gamma, z : T_2) ; (\Pi_2, z \rightsquivalence R_2, \text{Perm}(r) \in Q(\overline{\sigma})) \vdash e : \exists x : T.R \). Using the induction hypothesis and knowing that \(\Gamma, \Pi_3 \vdash e_1 : \exists x : T_2.R_2\), we obtain that

   \(\Gamma ; (\Pi_1, \Pi_2, \Pi_3, \text{Perm}(r) \in Q(\overline{\sigma})) \vdash [e_1/z]e : \exists x : T_2.[e_1/z]R\).

   The other two premises of the (PACK-SHMM) rule can also be obtained by inversion and they remain the same. So now we can apply the (PACK-SHMM) rule again and we get that

   \(\Gamma ; (\Pi_1, \Pi_2, \Pi_3) \vdash \text{pack } r \rightarrow \text{Perm}(r) \in Q(\overline{\sigma})\).

2. \( Q(\overline{\sigma}) \in [e_1/z]/e : \exists x : T.[e_1/z]R \). All the free variables in \(Q \) have been replaced by the argument \(\overline{\tau} \), there will be no more free \(z\) variables in \(\text{pack } r \rightarrow \text{Perm}(r) \in Q(\overline{\sigma})\), so \(\text{pack } r \rightarrow \text{Perm}(r) \in Q(\overline{\sigma}) \in [e_1/z]/e : \exists x : T_2.[e_1/z]R\).

3. Thus, \(\Gamma ; (\Pi_1, \Pi_2, \Pi_3) \vdash [e_1/z][\text{pack } r \rightarrow \text{Perm}(r) \in Q(\overline{\sigma}) \in [e_1/z]/e : \exists x : T_2.[e_1/z]R\), exactly what we wanted.

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

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

6. \( e \) is \( \text{pack } r \rightarrow \text{unique } (r) \in Q_2(\overline{\sigma}) \) in \( e \). The proof in this case is analogous to the one for the previous case, but \(\text{Perm} \) will be replaced by \(\text{unique} \) across the proof.
11. $e$ is unpack $r$ from immutable$(r)$ in $Q(\bar{\tau})$ in $e$. The proof in this case is analogous to the one for the previous case, but $\text{Perm}$ will be replaced by immutable across the proof.

12. $e$ is $t_0.m(\bar{t})$. We know that $(\Gamma, y : T_y); (\Pi, y \leadsto R_y) \vdash t_0.m(\bar{t}) : \exists \: T_r.[e_1/y][t_0/this]\in\bar{t}/\bar{x}.R$. We know by inversion that $(\Gamma, y : T_y); (\Pi, y \leadsto R_y) \vdash t_0/this\in\bar{t}/\bar{x}.R_1$. Using the induction hypothesis and knowing that $\Gamma; \Pi_3 \vdash e_1 : \exists x : T_y, R_0, R_1, R_2 \Rightarrow R$, we obtain that $\Gamma; \Pi_3 \vdash [e_1/y][t_0/this]\in\bar{t}/\bar{x}.R_1$. This is equivalent to writing $\Gamma; \Pi_3 \vdash [t_0/this][\in\bar{t}/\bar{x}.\bar{t}]/\bar{e}_1/y.R_1$.

By inversion we know that $(\Gamma, y : T_y) \vdash t_0 : C_0 \quad (\Gamma, y : T_y) \vdash \bar{t} : T$. Using the induction hypothesis we obtain that $\Gamma \vdash t_0 : C_0 \quad \Gamma \vdash \bar{t} : T$. Also by inversion we know that $\text{mtype}(m, C_0) = \forall \bar{x} : T. R$ and $R_i \Rightarrow R$ and $R_i$ implies $R_1$.

We can infer that $[e_1/y].R_1 \Rightarrow R \Rightarrow T$ and that $\text{mtype}(m, C_0) = \forall \bar{x} : T. R$.

13. $e$ is $\text{assign} t_1.f_1 := t$. We know that $(\Gamma, y : T_y); (\Pi, y \leadsto R_y, \Pi_1, \Pi_2) \vdash \text{assign} t_1.f_1 := t : \exists x : T_r. \text{Perm}'(x) \in Q(\bar{\tau}) \otimes \text{Perm}_0(t) \in Q(\bar{\tau}_0) \otimes p \otimes t_1.f_1 \Rightarrow t$. We know by inversion that $(\Gamma, y : T_y); (\Pi, y \leadsto R_y) \vdash t_1.f_1 \Rightarrow t$. Using the induction hypothesis and knowing that $\Gamma; \Pi_4 \vdash e_1 : \exists x : T_y, R_0, R_1, R_2 \Rightarrow R$, we obtain that $\Gamma; \Pi_1, \Pi_4 \vdash [e_1/y].t : T_r.[e_1/y].(\text{Perm}_0(t) \in Q(\bar{\tau}_0))$. Since all the free variables in $Q(\bar{\tau}_0)$ have been replaced by $\bar{\tau}_0$, 

$[e_1/y].(\text{Perm}_0(t) \in Q(\bar{\tau}_0)) = \text{Perm}_0([e_1/y].t) \in Q(\bar{\tau}_0)$. 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 $\Gamma; \Pi_1, \Pi_2, \Pi_3 \vdash \text{assign} t_1.f_1 := [e_1/y].t : \exists x : T_r. (\text{Perm}'(x) \in Q(\bar{\tau}) \otimes \text{Perm}_0([e_1/y].t) \in Q(\bar{\tau}_0)) \otimes p \otimes t_1.f_1 \Rightarrow [e_1/y].t$. Since $\text{assign} t_1.f_1 := [e_1/y].t : \exists x : T_r. (\text{Perm}'(x) \in Q(\bar{\tau}) \otimes \text{Perm}_0([e_1/y].t) \in Q(\bar{\tau}_0)) \otimes p \otimes t_1.f_1 \Rightarrow [e_1/y].t$, we finally obtain that $\Gamma; \Pi_1, \Pi_2, \Pi_3 \vdash [e_1/y].(\text{assign} t_1.f_1 := t) : \exists x : T_r.[e_1/y](\text{Perm}'(x) \in Q(\bar{\tau}) \otimes \text{Perm}_0(t) \in Q(\bar{\tau}_0) \otimes p \otimes t_1.f_1 \Rightarrow t)$. 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 $\mu, (\Sigma, l \leadsto (Q, i)), (\Pi, l \leadsto R), \rho ok$ and $\mu, (\Sigma, l \leadsto (Q, i)), (\Pi, l \leadsto R), \rho ok$, where $R = \text{Perm}(x) in Q$.

2. If $\mu, (\Sigma, \Pi, \rho ok$ and $o \notin \text{dom}(\mu) and \text{init}(C) = (Q(\bar{\tau}))$ then $\mu[\{0 \sim C(\rho)(\bar{t})\}], (\Sigma, o \sim (\text{unpack}(i), (\Pi, o \sim \text{unpack}(o, unique(\Sigma) in Q(\bar{\tau}))), \rho ok$.

3. If $\mu, (\Sigma, l \leadsto (Q, i)), (\Pi, l \leadsto R), \rho ok and o \notin \text{dom}(\mu) and \text{init}(C) = (Q(\bar{\tau}))$ then $\mu, (\Sigma, l \leadsto (Q, i)), (\Pi, l \leadsto R), \rho' \leadsto \rho(l)$ ok, where $P = \text{Perm}(x) in Q$.

4. If $\mu, (\Sigma, \Pi_1, \Pi_2, \rho ok$ and $\text{unpack}(r, \text{Perm}(r) in Q(\bar{\tau})) \in \Pi_1, then $\mu, (\Sigma, \Pi_2, \rho ok$ and $\text{Perm}(r) in Q(\bar{\tau})) \in \Pi_1$ and $\rho ok$.

5. If $\mu, (\Sigma, \Pi_1, \Pi_2, \rho ok$ and $\text{unpack}(r, \text{unique}(r) in Q(\bar{\tau})) \in \Pi_1$ and $\rho ok$.

6. If $\mu, (\Sigma, \Pi_1, \Pi_2, \rho ok$ and $\text{Perm}(r) in Q(\bar{\tau}) \in \Pi_0 \text{ and } Q(\bar{\tau}) \Rightarrow \Pi_0 \text{ and } \Pi_1 \Rightarrow r \neq r')$ then $\mu, (\Sigma', \Pi_0 \leadsto \text{unpack}(r'), \text{Perm}(r') \in Q(\bar{\tau}) \in (\Pi_0 \cup \Pi_2) \Rightarrow \Pi_0, \Pi_2 \Rightarrow r \neq r')$.

7. If $\mu, (\Sigma, \Pi, \rho ok$ and $\rho(l)(\bar{t}) = C(\bar{\tau}) \text{ and } \text{fields}(C) = \bar{\tau}$ then $\mu' = (\mu + o_1: T_1, \Sigma' = (\Sigma, o_1 \rightarrow (Q, j)), \Pi' = (\Pi, o_1 \rightarrow R), \rho ok$, where $R = \text{Perm}(x) in Q$.

8. If $\mu, (\Sigma, \Pi_0, \Pi_2 \leadsto (Q(\bar{\tau}), j)), \Pi = (\Pi_0, \Pi_2 \leadsto (\text{Perm}(y) in Q(\bar{\tau}) \otimes \text{Perm}_0(x) in Q(\bar{\tau}_0) \otimes p \otimes t_1.f_1 \Rightarrow x), \rho ok$ and $\rho(l_2) = o_2$, then $\mu' = (\mu[l_1 \rightarrow o_2], \Sigma = (\Sigma, o_2 \rightarrow (Q(\bar{\tau}), j)), \Pi' = (\Pi, o_2 \rightarrow \text{Perm}(y) in Q(\bar{\tau}) \otimes \text{Perm}_0(x) in Q(\bar{\tau}_0) \otimes p \otimes t_1.f_1 \Rightarrow x), \rho ok$.

**Proof of memory consistency lemma**

1. Environment map

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

Assuming \( \mu, \Sigma, \Pi, \rho \in \text{o} \) and \( \text{o} \notin \text{dom}(\mu) \), we have to show that \( \mu' = [\mu(\text{o} \rightarrow C(\rho))] \), \( \Sigma' = (\Sigma, \text{o} \rightarrow (\text{unpack}(i))) \), \( \Pi' = (\Pi, \text{o} \rightarrow \text{unpack}(\text{o}, \text{unique}(\text{o}) \in Q_0(\Pi)) \). It must be that \( \rho(l) = \sigma^j \) for some objects \( \sigma' \). We know that \( \text{init}(C) = (Q_0(\Pi)) \). This means that \( Q_0(\Pi) \) is of the form \( f \rightarrow x \) (this is the requirement 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 \( o, \sigma' \). Since \( \mu(o') = \mu'(o') \) and \( \text{props}(\mu, \Sigma, \Pi, \rho, o') = \text{props}(\mu', \Sigma', \Pi', \rho, o') \) we can deduce that

\[
\mu', \Sigma', \Pi', \rho \rightarrow o' \in \text{ok}.
\]

The only object proposition refering to \( o \) in \( \Pi' \) is unpacked(\( o, \text{unique}(o) \in Q_0(\Pi) \)), which means that the heap invariants are satisfied and we can deduce that \( \mu', \Sigma', \Pi', \rho \rightarrow o \in \text{ok} \). Thus, \( \mu', \Sigma', \Pi', \rho \in \text{ok} \).

3. Environment rename

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

4. Packing to Perm

Assuming \( \Omega_1 = [\mu, \Sigma = (\Sigma_1, \Sigma_2), \Pi = (\Pi_1, \Pi_2), \rho \in \text{ok} \) we have to show that \( \Omega_2 = [\mu, \Sigma' = (\Sigma_2, \Pi_2 \rightarrow (Q(\Pi), i)), \Pi' = (\Pi_2, \Pi_2 \rightarrow \text{Perm}(x) \in Q(\Pi)), \rho \in \text{ok} \), where \( \text{Perm} \) is share or \( \text{Perm} \) is immutable. Let’s take an arbitrary \( o \). Since \( \mu \) and \( \rho \) don’t change, the only changes in the \( \tilde{P} \)'s corresponding to \( \Omega_1 \) to \( \Omega_2 \) come from the different \( o \rightarrow R \) extracted from \( \Pi \) and from \( \Pi' \). We have to show that the heap invariants are preserved by the different \( o \rightarrow R \) in \( \Pi' \), knowing that the invariants are preserved by the different \( o \rightarrow R \) in \( (\Pi_1, \Pi_2) \). Knowing this, we deduce that the invariants cannot be broken by the assertions in \( \Pi_2 \). Thus, we only have to see if \( r \rightarrow \text{Perm}(x) \in Q(\Pi) \) is in contradiction with any assertions about \( r \) in \( \Pi_2 \). We also know that \( \text{unpack}(r, \text{Perm}(x) \in Q(\Pi)) \in \Pi_1 \). Since \( \Omega_1 \in \text{ok} \), the only object proposition in \( \Pi_2 \) about \( r \) has to be \( \text{none}(r) \in Q_3(\Pi) \), according to the heap invariants. It follows that

\[
(\Pi_2, r \rightarrow \text{unique}(r) \in Q_2(\Pi)) \text{ satisfies the heap invariants. Since } [\tilde{P}/x, \tilde{P}/\tilde{R}]R2 \in \Pi_1 \text{ and the primitives corresponding to } (\Pi_1, \Pi_2) \text{ are ok, there can be no primitives in } \Pi_2 \text{ that contradict } [\tilde{P}/x, \tilde{P}/\tilde{R}]R2 \text{. We know that } Q_2(\Pi) = \exists_{R2} R2 \in C \text{ and we can deduce that the primitives corresponding to } \mu, \Sigma', \Pi', \rho \text{ are ok. Thus }
\]

\[
\mu, \Sigma', \Pi', \rho \in \text{ok}.
\]

6. Unpacking from Perm

Assuming \( \Omega_1 = [\mu, \Sigma = (\Sigma_0, \Sigma_2), \Pi = (\Pi_0, \Pi_2), \rho \in \text{ok} \) we have to show that \( \Omega_2 = [\mu, \Sigma' = (\Sigma_2, r \rightarrow \text{unpack}(i)), \Pi' = (\Pi_2, [\tilde{P}/x, \tilde{P}/\tilde{R}]R1, r \rightarrow \text{unpack}(x, \text{Perm}(x) \in Q(\Pi))) \in \Pi_1 \). Let’s take an arbitrary \( o \). Since \( \mu \) and \( \rho \) don’t change, the only changes in the \( \tilde{P} \)'s corresponding to \( \Omega_1 \) to \( \Omega_2 \) come from the different \( o \rightarrow R \) extracted from \( \Pi_0 \) and from \( \Pi_2 \). We have to show that the heap invariants are preserved by the different \( o \rightarrow R \) in \( \Pi' \), knowing that the invariants are preserved by the different \( o \rightarrow R \) in \( \Pi \). Knowing this, we deduce that the invariants cannot be broken by the assertions in \( \Pi_2 \). Thus, we only have to see if \( r \rightarrow \text{unpack}(x, \text{Perm}(x) \in Q(\Pi)) \) and \( [\tilde{P}/x, \tilde{P}/\tilde{R}]R1 \) are in contradiction with any assertions about \( r \) in \( \Pi_2 \). Since \( \forall x', \Pi, \text{Perm} : (\text{unpack}(r', \text{Perm}(x') \in Q(\Pi)) \in \Pi_0 \cup \Pi_2 \Rightarrow \Pi_0 \cup \Pi_2 \vdash r \neq r' \) the heap invariants allow us to infer that \( \Pi_2 \) does not contain any object that is unpacked from the predicate \( Q \) and aliases with \( r \). We also know that \( \text{Perm}(r) \in Q(\Pi) \in \Pi_1 \). Using the heap invariants, we deduce that if there is an object proposition refering to \( r \) in \( \Pi_2 \), this object proposition must be \( \text{Perm}(r) \in Q(\Pi) \) or \( \text{none}(r) \in Q_2(\Pi) \). The \( \text{none}(r) \) in \( Q_2(\Pi) \) object proposition expresses the fact that the predicate \( Q_2(\Pi) \) holds for \( r \), but there is no alias to \( r \) that can conflict with other aliases. This \textit{none} permission can be ignored in our proof.
The formula $[\tau/\varnothing] R_1$ corresponds to $r$, after it got unpacked. In this formula there might be object propositions referring to $r$ or to other references that appear in $\Pi_2$. Since $r$ was packed to $Q$, using object propositions from $\Pi_0$, right before being unpacked and since $Q(\varnothing) = \exists \varnothing R_1$, we deduce that $[\tau/\varnothing] R_1$ will only contain object propositions that are already in $\Pi_0$. This means that the different $\alpha \rightarrow R$ extracted from $(\Pi_0, \Pi_2)$ are compatible with each other and with $r \rightarrow \text{unpacked}(x, \Perm(x) in Q(\tau))$ (same reasoning as in the previous paragraph). If $\Perm = \text{immutable}$ 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 $\Pi'$ because: there is no object that aliases with $r$ that is unpacked from $Q$ in $\Pi'$, and also because $r \rightarrow \text{unpacked}(x, \Perm(x) in Q(\tau))$, $\Perm(r)$ in $Q(\tau)$ and $[\tau/\varnothing] R_1$ do not contain object propositions or primitives that are not compatible. Thus, $\mu, \Sigma', \Pi', \rho ok$.

7. Field read
Assuming $\mu, \Sigma, \Pi, \rho ok$ and $\mu(\rho(l)) = C(\varnothing)$ and fields$(C) = T_{\varnothing}$, we have to show that $\mu' = (\mu + \alpha_i : T_i, \Sigma' = (\Sigma, \alpha_i \rightarrow (Q, j)), \Pi' = (\Pi, \alpha_i \rightarrow R, \rho ok)$, where $R = \Perm(x) in Q$. The only object affected is $\alpha_i$. Because of the way fieldProps$(\mu', \Sigma')$ is defined, any object proposition about $\alpha_i$ will be extracted from the object propositions referring to $\mu(\rho(l))$, which are already in $\Pi$. This means that $\props(\mu, \Sigma, \Pi, \rho, \alpha_i) = \props(\mu', \Sigma', \Pi', \rho, \alpha_i)$ and $\mu', \Sigma', \Pi', \rho \vdash \alpha_i ok$. Thus $\mu', \Sigma', \Pi', \rho ok$.

8. Assignment
Assuming $\mu, \Sigma = (\Sigma_0, \alpha_0 \rightarrow (Q(\tau), j)), \Pi = (\Pi_0, \alpha_2 \rightarrow \Perm(\rho(x) in Q_0(\tau) \otimes \exists t_1, f_1 \rightarrow x), \rho ok$ and $\rho(l_2) = \alpha_2$, we have to prove that $\mu' = (\mu + \rho(\alpha_1) \rightarrow \varnothing, C(\varnothing)), \Sigma' = (\Sigma, \alpha_2 \rightarrow (Q(\tau'), j)), \Pi' = (\Pi, \alpha_2 \rightarrow \Perm(\rho(y) in Q(\tau') \otimes \Perm(y) in Q_0(\tau) \otimes \exists t_1, f_1 \rightarrow x), \rho ok$.

The only object that changes is $\alpha_1$. Since $\props(\mu, \Sigma, \Pi, \rho, \alpha_i) = \props(\mu', \Sigma', \Pi', \rho, \alpha_i)$ and $\mu, \Sigma, \Pi, \rho \vdash \alpha_i ok$, we can conclude that $\mu', \Sigma', \Pi', \rho \vdash \alpha_i ok$ and thus $\mu', \Sigma', \Pi', \rho ok$.

The proof for the Preservation Theorem is done by induction on the dynamic semantics rules. The rule $(\text{LET})$ can be applied as the first step in each derivation. This is because in the static rules $\Pi$ could incorporate a number of $\Pi_i$, as we do not know which of the $\Pi_i$ 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) $\mu, \Sigma, \Pi, \rho ok$
(c) $\mu, \rho, l \rightarrow \mu, \rho, \rho(l)$

2. By inversion on $1a$
(a) $\Gamma = (\Gamma_1, l : T)$
(b) $\Pi = (\Pi_1, l \rightarrow R), \Sigma = (\Sigma_1, l \rightarrow (Q, i))$
(c) $\Sigma = (\Sigma_1, l \rightarrow (Q, i))$, where $\Sigma_1$ is the store type corresponding to $\Pi_1$. $i$ represents the index of $\Pi$, that contains the $R$. The value of $i$ will be determined at runtime.

3. $\mu, \Gamma_1, l : T), (\Pi_1, l \rightarrow R), (\Sigma_1, l \rightarrow (Q, i)) \text{o ak}$ -by 2,

4. $\rho(l) = o$, for some $o - \text{by Object Proposition Consistency}$

5. Let $\Gamma' = (\Gamma', \rho(l) : T), \Pi' = (\Pi_1, l \rightarrow R)$ and $\Sigma' = (\Sigma_1, l \rightarrow (Q, i))$

6. $(\Gamma, o : T), (\Pi_1, o \rightarrow R), (\Sigma_1, l \rightarrow (Q, i)) \text{o ak}$ -by (TERM)

7. $\Gamma', \Pi' \vdash \rho(l) : \exists x : T.R$ -by 5, 6

8. $\mu, (\Pi_1, l \rightarrow R), (\Sigma_1, l \rightarrow (Q, i)) \text{o ak}$ -by 3, 4, memory consistency lemma

9. $\mu, \Pi, \Sigma', \rho ok$ -by 5, 8

10. q.e.d -by 7, 9

Case (NEW)

1. By assumption
(a) $\Gamma, \Pi \vdash \text{new } C(\varnothing) : \exists y : T.R$
(b) $\mu, \Sigma, \Pi, \rho ok$
(c) $\mu, \rho, \text{new } C(\varnothing) \rightarrow \mu', \rho, o$
(d) $o \notin \text{dom}(\mu)$
(e) $\mu' = \mu[o \rightarrow C(\rho(l))]$

2. By inversion on $1a$
(a) $\exists y : T.R = \exists z : C, [\text{unpacked}(z, \text{unique}(z) in Q_0(\tau)) \otimes \overline{\tau} \rightarrow \overline{\tau}]$
(b) $\Gamma = (\Gamma_1, l : T)$
(c) $\text{fields}(C) = T_{\overline{\tau}}$
(d) $Q_0(\tau) = \emptyset \subset C$
(e) $\text{init}(C) = \langle Q_0(\tau) \rangle$

3. Let $\Gamma' = (\Gamma, o : C), \Pi' = (\Pi, o \rightarrow [\text{unpacked}(z, \text{unique}(z) in Q_0(\tau)) \otimes \overline{\tau} \rightarrow \overline{\tau}])$ -by (TERM)

4. Let $\Sigma' = (\Sigma, o \rightarrow (\text{unpacked}, i))$

5. $\Gamma', \Pi' \vdash o : \exists z : C, [\text{unpacked}(z, \text{unique}(z) in Q_0(\tau)) \otimes \overline{\tau} \rightarrow \overline{\tau}]$ -by (TERM)

6. $\mu[o \rightarrow C(\rho(l))], (\Sigma, o \rightarrow (\text{unpacked}, i)), (\Pi, o \rightarrow [\text{unpacked}(z, \text{unique}(z) in Q_0(\tau)) \otimes \overline{\tau} \rightarrow \overline{\tau}]), \rho ok$ -by memory consistency lemma

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

Case (LET-o)
1. By assumption
   (a) $\Gamma, \Pi \vdash o : e_2 : \exists y : T.R$
   (b) $\mu, \Sigma, \Pi, \rho ok$
   (c) $\mu, \rho, \text{let } x = o \text{ in } e_2 \rightarrow \mu, \rho[l \leadsto o], [l/x]e_2$
   (d) $l \notin \text{dom}(\rho)$

2. By inversion on 1a
   (a) $\Gamma, \Pi \vdash o : \exists x : T_1.R_1$
   (b) $R_1 \vdash \exists z : T_2.R_2$
   (c) $(\Gamma, x : T_1, z : T_2), R_2 \vdash e_2 : \exists y : T_2.R$
   (d) $\exists y : T.R = \exists w : T_2,[o/x]R$

3. $\Gamma = (\Gamma_1, o : T_1), \Pi = (\Pi_1, o \rightarrow R_1)$ - by inversion on 2a
   (b) $\Pi_1 = \permi{Pi}$
   (c) $\Gamma' = (\Gamma, l : T_1), \Pi' = (R_2, l \rightarrow R_1), \Sigma' = (Q_2, l \leadsto (Q_1, i))$, where $\Sigma_2$ corresponds to $R_2$

4. Let $\Gamma' = (\Gamma, l : T_1); \Pi' = (R_2, l \rightarrow R_1); [l/x]e_2 : \exists y : T_2,[l/x]R_2$ - by 1d, 2c, Substitution Lemma

5. $\Gamma', \Pi' \vdash e_2 : \exists y : T.R$ - by 6, 2d

6. $\mu, (\Sigma_2, o \leadsto (Q_1, i)), (\Pi_2, o \rightarrow R_1), \rho ok$ - by memory consistency lemma

7. $\mu, \Pi', \Sigma', \rho[l \leadsto o] ok$

8. q.e.d. - by 10, 7
   Case (LET-E)

   1. By assumption
      (a) $\Gamma, \Pi \vdash let x = e_1 \text{ in } e_2 : \exists y : T.R$
      (b) $\mu, \Sigma, \Pi, \rho ok$
      (c) $\mu, \rho, \text{let } x = e_1 \text{ in } e_2 \rightarrow \mu', \rho', \text{let } x = e_1' \text{ in } e_2$
      (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) $R_1 \vdash \exists z : T_2.R_2$
   (c) $(\Gamma, x : T_1, z : T_2), R_2 \vdash e_2 : \exists y : T_2.R$
   (d) $\exists y : T.R = \exists w : T_2,[e_1/x]R$

3. By induction on 1b, 1d, 2a
   (a) $\exists \Gamma_0, \Pi' \text{ such that } \Gamma_0, \Pi' \vdash e' : \exists x : T_1.R_1$
   (b) $\exists \Sigma', \Pi', \rho ok$

4. Let $\Gamma' = \Gamma \cup \Gamma_0$

5. $\Gamma', \Pi' \vdash let x = e_1' \text{ in } e_2 : \exists y : T.R$ - 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) $\Gamma, \Pi \vdash \text{pack } r \text{ to } R_1 \text{ in } e_1 : \exists x : T.R$
   (b) $\mu, \Sigma, \Pi, \rho ok$
   (c) $\mu, \rho, \text{pack } r \text{ to } R_1 \text{ in } e_1 \rightarrow \mu, \rho, e_1$

2. By inversion on (PACK-SH-IMM)
   (a) $(\Gamma, \exists y : T_y, \Pi_1 \vdash r : T_1, \exists y : T_y, \exists y : T_y) \vdash \exists y : T.R$
   (b) $\exists y : T.R \vdash \exists y : T.R$
   (d) $\Pi = (\Pi_1, \Pi_2)$

3. Let $\Pi' = (\Pi_2, r \leadsto \text{Perm}(x) \text{ in } Q(\pi))$, $\Sigma' = (\Sigma_2, r \leadsto (Q(\pi), i))$, $\Pi_2, r \leadsto \text{Perm}(x) \text{ in } Q(\pi)$

4. q.e.d. - by 1a, 2b, 3
   Case (UNPACK) Subcase: the static semantics rule corresponding to (UNPACK) is (UNPACK-SH-UNI).

1. By assumption
   (a) $\Gamma, \Pi \vdash \text{unpack } r \text{ from } R_1 \text{ in } e_1 : \exists x : T.R$
   (b) $\mu, \Sigma, \Pi, \rho ok$
   (c) $\mu, \rho, \text{unpack } r \text{ from } R_1 \text{ in } e_1 \rightarrow \mu, \rho, e_1$

2. By inversion on (UNPACK-SH-UNI)
   (a) $\Gamma, \Pi_0 \vdash r : T_1, \text{Perm}(r) \text{ in } Q(\pi)$
   (b) $(\Gamma, \exists y : T_y, \Pi_2) \vdash \exists y : T_y \vdash \exists y : T.R$
unpacked(r, Perm(r) in Q(\(\overline{T}\))) \vdash e : \exists x : T.R
(c) \(\forall r', \bar{T}, Perm' : (unpacked(r', Perm'(r') in Q(\(\overline{T}\))).: (Pi_0 \cup Pi_2) \Rightarrow Pi_0, Pi_2 \vdash r \neq r')
(d) Q(\(\overline{T}\)) = \exists \bar{y}. R_1 \in C \quad Perm \in \{unique, share\}
(e) Pi = (Pi_0, Pi_2)
3. Let Pi' = (Pi_2, (\(\overline{T_1}\)/\(\overline{T}\))R_1)
   \(\vdash r \rightarrow \text{unpacked}(x, Perm(x) in Q(\(\overline{T}\))))
   \(\Sigma' = (\Sigma_2, r \rightarrow \text{unpacked}(i),)\)
   \(\Gamma' = (\Gamma, \bar{y} : \overline{T_1})\)
4. \(\Gamma', Pi' \vdash e : \exists x : T.R \text{-by 2b, 3}\)
5. \(\mu, (\Sigma_2, r \rightarrow \text{unpacked}(i),)\)
   \((\Pi_2, (\(\overline{T_1}\)/\(\overline{T}\))R_1, r \rightarrow \text{unpacked}(x, Perm(x) in Q(\(\overline{T}\))))\)
   \(\vdash e : \exists x : T.R \text{-by memory consistency lemma}\)
6. q.e.d. -by 4, 5
Subcase: the static semantics rule corresponding to (UNPACK) is
   (UNPACK-IMM).
   1. By assumption
      (a) \(\Gamma, Pi \vdash \text{unpack} r \text{ from } R_1 \text{ in } e_1 : \exists x : T.R\)
      (b) \(\mu, \Sigma, Pi, \rho o_k\)
      (c) \(\mu, \rho, \text{unpack} r \text{ from } R_1 \text{ in } e_1 \rightarrow \mu, \rho, e_1\)
   2. By inversion on (UNPACK-IMM)
      (a) \(\Gamma, Pi_0 \vdash r : T_i, \text{immutable}(r) in Q(\(\overline{T}\))\)
      (b) \((\Gamma, \bar{y} : \overline{T_i}); (Pi_2, (\(\overline{T_1}\)/\(\overline{T}\))R_1,\)
          \(\vdash e : \exists x : T.R\)
      (c) \(\forall r', \bar{T}, Perm' : \text{unpacked}(r', Perm'(r') in Q(\(\overline{T}\))) \in (Pi_0 \cup Pi_2) \Rightarrow Pi_0, Pi_2 \vdash r \neq r')
      (d) Q(\(\overline{T}\)) = \exists \bar{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, (\(\overline{T_1}\)/\(\overline{T}\))R_1,
   \(\vdash r \rightarrow \text{unpacked}(x, \text{immutable}(x) in Q(\(\overline{T}\)))\)),
   \(\Sigma' = (\Sigma_2, r \rightarrow \text{unpacked}(i),)\)
   \(\Gamma' = (\Gamma, \bar{y} : \overline{T_1})\)
4. \(\Gamma', Pi' \vdash e : \exists x : T.R \text{-by 2b, 3}\)
5. \(\mu, (\Sigma_2, r \rightarrow \text{unpacked}(i),)\)
   \((\Pi_2, (\(\overline{T_1}\)/\(\overline{T}\))R_1, r \rightarrow \text{unpacked}(x, \text{immutable}(x) in Q(\(\overline{T}\))))\)
   \(\vdash e : \exists x : T.R \text{-by memory consistency lemma}\)
6. q.e.d. -by 4, 5
Case (IF-TRUE)
1. By assumption
   (a) \(\Gamma, Pi \vdash \text{if}(true, e_1, e_2) : \exists x : T.R\)
   (b) \(\mu, \Sigma, Pi, \rho o_k\)
   (c) \(\mu, \rho, \text{if}(true, e_1, e_2) \rightarrow \mu, \rho, e_1\)
   (d) \(\exists x : T.R = \exists x : T.R_1 \oplus T_2\)
2. By inversion on the static semantics rule (IF): \(\Gamma, Pi \vdash e_1 : \exists x : T.R_1\)
\[ \mu[\rho(l_1) \rightsquigarrow [\rho(l_2)/o_1]C(\sigma)], \rho, \rho(l_2) \]

(d) \[ \mu(\rho(l_1)) = C(\sigma) \]

(e) \[ fields(C) = T \]

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

(a) \[ \Gamma; \Pi \vdash l_2 : \exists x : T_r. \text{Perm}_0(x) \text{ in } Q_0(\tau_0), \text{ thus } T = T_l \]

(b) \[ \Gamma; \Pi \vdash l_1.f : T_r. \text{Perm}'(r_1) \text{ in } Q'(\tau) \otimes p \]

(c) \[ p = \text{unpacked}(l_1, \text{Perm}(l_1) \text{ in } Q(\tau)) \]

(d) \[ \Pi_i \vdash l_1.f \rightarrow r_i \]

(e) \[ \text{Perm} \neq \text{immutable} \]

(f) \[ \exists x : T.R = \exists x : T_r. \text{Perm}(x) \text{ in } Q'(\tau) \otimes \text{Perm}_0(l_2) \text{ in } Q_0(\tau_0) \otimes p \otimes l_1.f \rightarrow l_2 \]

(g) \[ \Pi = (\Pi_1, \Pi_2, \Pi_3) \]

3. \[ \exists o_2 \text{ such that } \rho(l_2) = o_2. \]

4. Let \[ \Gamma' = (G, o_2 : T_l), \Pi' = (\Pi, o_2 \rightsquigarrow \text{Perm}(x) \text{ in } Q'(\tau) \otimes \text{Perm}_0(l_2) \text{ in } Q_0(\tau_0) \otimes p \otimes t_1.f_1 \rightarrow l_2), \Sigma' = (\Sigma, o_2 \rightsquigarrow (Q'(\tau), j)). \]

5. \[ \Gamma', \Pi' \vdash o_2 : \exists x : T_r. R \text{-by (TERM)} \]

6. \[ \mu' = \mu[\rho(l_1) \rightsquigarrow [\rho(l_2)/o_1]C(\sigma)], \Sigma', \Pi', \rho \text{ ok} \text{-by memory consistency lemma} \]

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

Case (INVOLVE)

(a) By assumption
   i. \[ \Gamma, \Pi \vdash l_1.m(l_2) : \exists x : T.R' \]
   ii. \[ \mu, \Sigma, \Pi, \rho \text{ ok} \]
   iii. \[ \mu, \rho, l_1.m(l_2) \rightarrow \mu, \rho, [l_1/\text{this}, l_2/\tau]e \]
   iv. \[ \vdash PR \]
   v. \[ \mu(\rho(l_1)) = C(\sigma) \]
   vi. \[ \text{method}(m, C) = T_r m(\tau)\{\text{return } e\} \]

(b) By inversion on the static semantics rule (CALL)
   i. \[ \Gamma \vdash l_1 : C \quad \Gamma \vdash l_2 : T \]
   ii. \[ \Gamma; \Pi \vdash [l_1/\text{this}]l_2/\tau]R_1 \]
   iii. \[ \text{mtype}(m, C) = \forall x.T.R \text{-res} : T_r. R \rightarrow R \]
   iv. \[ R_1 \text{ implies } R' \]
   v. \[ \exists x : T.R' = \exists \text{ result : } T_r.[l_1/\text{this}][l_2/\tau]R \]

(c) From 7(a)iv we know that the body \{\text{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 \( [l_1/\text{this}, l_2/\tau]e \) will be of the type \( \exists x : T_r. R' \). Since \( \mu, \Sigma, \Pi, \rho \) do not change, \( \mu, \Sigma, \Pi, \rho \text{ ok} \).

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