Verifying Correct Usage of Atomic Blocks and Typestate: Technical Companion

Nels E. Beckman[†] **Jonathan Aldrich**[†]

August 2008
CMU-ISR-08-126

Institute for Software Research
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

[†]Institute for Software Research, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA

This work was supported by a University of Coimbra Joint Research Collaboration Initiative, DARPA grant #HR00110710019, Army Research Office grant #DAAD19-02-1-0389 entitled "Perpetually Available and Secure Information Systems", the Department of Defense, and the Software Industry Center at CMU and its sponsors, especially the Alfred P. Sloan Foundation.

Keywords: transactional memory, typestate, proof

Abstract

In this technical report, we present a static and dynamic semantics as well as a proof of soundness for a programming language presented in the paper entitled, *Verifying Correct Usage of Atomic Blocks and Typestate* [1]. The proof of soundness consists of a proof of preservation, which shows that well-typed expressions evaluate to other well-typed expressions, and a proof of progress, which shows that well-typed expressions are either values or can take an evaluation step in the dynamic semantics. The notion of progress is complicated by a specific notion of a well-typed heap, which ensures that only one reference in the entire thread-pool can know the exact state of an object of share or pure permission.

1 Proof of Soundness

Our soundness criterion is as follows: It is either the case that all of the threads in a program are values, or their exists one thread such that the expression this thread is evaluating is well-typed and can take a step to another well-typed expression. If one of the threads in the thread-pool is currently executing within a transaction, then that thread must step, and if no threads in the thread-pool are currently executing within a transaction, then any threads that is not a value must be able to step. The dynamic semantics track typestate, and there is no evaluation rule to allow a method call if method preconditions are not met. In order to prove that method preconditions are always met for well-typed programs, our store typing judgment requires the invariant that only one thread can pinpoint the state of a share or pure object at a time.

The language of proof differs from the language used in the paper in a few ways. We have restored the original effects system used by Bierhoff and Aldrich [2]. This system was removed from the paper for purposes of clarity. The effects system keeps track of the fields that are modified in a subexpression to ensure that only the fields of the unpacked object are modified, and no field permissions "escape" beyond the packing of that object. Otherwise, our proof language resembles their proof language in most ways. As it was for Bierhoff and Aldrich, we have simplified the language of proof by removing linear disjunction ($\oplus$) and additive conjunction ($\&$). In the paper, an object is known to be unpacked if there is an $\mathsf{unpacked}(k, s)$ permission inside of the linear context. In the system shown here, we use a separate context u. One u appears on the left-hand side of the judgment. This shows us which object is unpacked before the expression takes an evaluation step. The other u appears on the right-hand side, and shows us which object is unpacked after the expression has finished an evaluation step.

For the majority of the proof, things proceed much as they did in the proof of soundness presented by Bierhoff and Aldrich [2] with many of the multi-threaded features coming from Moore and Grossman [3]. Our system is different in a few ways. In Bierhoff and Aldrich the stack permissions, that is the dynamic representation of permissions that are currently available for use by the evaluating expression, were actually stored inside the heap. Because we have many threads, we have a separate environment S_p attached to each thread expression which holds these stack permissions. Additionally, when typing a pool of threads, T (essentially a list of expressions and their stack permissions), we associate each with their own linear context Δ and incoming unpacking flag u. We often must refer to the entire collection of linear contexts and packing flags, and this will usually be written $\overline{\Delta}$ and $\overline{u}$. Keep in mind that each linear context and unpacking flag is associated with one specific thread. This would most accurately be written as a list of tuples except that our Δ and u usually appear on the left-hand side of the rule, while the thread itself will appear on the right-hand side, and so treating them as a tuple would be notationally awkward.

When type-checking the top-level thread pool, the members of $\overline{\Delta}$ and $\overline{u}$ are tagged with an additional bit of information, and are written $\overline{\Delta^{\mathcal{E}}}$ and $\overline{u^{\mathcal{E}}}$. At most one Δ and u pair are allowed to contain specific state information about pure and share permissions. If this is the case, that Δ and u will be tagged with wt, whereas others may not be. The fact that at most one linear context and unpacking flag is allowed to contain state information about share and pure permissions is checked by the $a; \langle\overline{\Delta^{\mathcal{E}}}, \overline{u^{\mathcal{E}}}\rangle$ ok judgment.

1.1 Proof Language

$$
\begin{array}{rrcl}
\textit{program} & \textit{PG} & ::= & \langle \overline{\textit{CL}}, e \rangle \\
\textit{class decls.} & \textit{CL} & ::= & \texttt{class}\ C\ \{\ \overline{F}\ I\ \overline{N}\ \overline{M} \} \\
\textit{methods} & M & ::= & C_r\ m(\overline{C\ x}) : P_1 \multimap \exists result : C_r.P_2 = e \\
\textit{terms} & t & ::= & x, y, z \mid o \\
\textit{expressions} & e & ::= & k \cdot t \mid k \cdot t.f \mid t_1.f := k \cdot t_2 \\
 & & \mid & \texttt{new}\ C(\overline{k \cdot t}) \mid k \cdot t.m(\overline{k \cdot t}) \\
 & & \mid & \texttt{inatomic}\ (e) \\
 & & \mid & \texttt{let}\ x = e_1\ \texttt{in}\ e_2 \\
 & & \mid & \texttt{spawn}\ (k \cdot t.m(\overline{k \cdot t})) \mid \texttt{atomic}\ e \\
 & & \mid & \texttt{unpack}_{\mathcal{E}} k \cdot t@s\ \texttt{in}\ e \mid \texttt{pack}\ t\ \texttt{to}\ s'\ \texttt{in}\ e \\
\textit{expression types} & E & ::= & \exists x : C.P \\
 & I & ::= & \mathsf{init}\langle \exists \overline{f : C}.P, s \rangle \\
\textit{atomic} & \mathcal{E} & ::= & \mathsf{wt} \mid \mathsf{ot} \mid \mathsf{emp} \\
\textit{states} & S & ::= & s \mid \mathsf{unpacked}(k) \mid \mathsf{unpacked}(s) \\
\textit{Predicates} & P & ::= & k \cdot r@\$ \mid P_1 \otimes P_2 \\
 & \$ & ::= & s \mid ? \\
 & N & ::= & s = P \\
\textit{valid contexts} & \Gamma & ::= & \cdot \mid \Gamma, x : C \\
\textit{linear contexts} & \Delta^{\mathcal{E}} & ::= & \cdot \mid \Delta^{\mathcal{E}}, P \\
\textit{stores} & \Sigma & ::= & \cdot \mid \Sigma, o : C \\
\textit{heaps} & H & ::= & \cdot \mid H, o \mapsto C(\overline{f = k \cdot o})@S \\
 & k & ::= & \mathsf{full} \mid \mathsf{pure} \mid \mathsf{share} \mid \mathsf{immutable} \mid \mathsf{unique} \\
 & u & ::= & - \mid k \cdot t@s \\
 & \omega & ::= & \emptyset \mid \{t.f\} \mid \omega_1 \cup \omega_2
\end{array}
$$

1.2 Judgment Forms

Judgment	Judgment form	Description
Top-Level Evaluation	$a; H; T \to a'; H'; T'$	Under transaction state a and heap H the thread-pool T evaluates to T', which may modify an expression and add a new expression, while possibly modifying the heap H' and changing the transaction state a'.
Expr. Evaluation	$a; H; \langle e, S_p \rangle \to a'; H'; \langle e', S_p' \rangle; T$	In heap H, with transaction state a and stack permissions S_p, the expression e takes a step to e', potentially modifying each and potentially adding a new thread.
Expression typing	$\Gamma; \Sigma; \Delta; \mathcal{E}; u \vdash e : E \setminus \omega \| u'$	In variable context Γ, store Σ, linear context Δ, transaction effect $\mathcal{E}$, and unpacking flag u, expression e has type E and may assign to fields in ω and and changes unpacking to u'.
Store typing (definition 1.5.1)	$\Sigma; \overline{\Delta^{\mathcal{E}}}; \overline{u^{\mathcal{E}}} \vdash H; \overline{S_p}$	In store context Σ with lists of linear contexts $\overline{\Delta^{\mathcal{E}}}$ and packing flags $\overline{u^{\mathcal{E}}}$, each tagged with a transaction effect, the heap H and the list of all stack permissions $\overline{S_p}$ is well-typed.
Linear logic entailment (figure 6)	$\Gamma; \Sigma; \Delta \vdash P$	In variable context Γ and store Σ, linear context Δ proves P.
Runtime property check (definition 1.5.2)	$H; S_p \| \overline{k \cdot o} \vdash P$	Heap H with stack permissions S_p restricted to stack permissions $\overline{k \cdot o}$ satisfies property P.

$$\boxed{\Gamma; \Sigma; \Delta \vdash_{\mathcal{E}} P}$$

$$\frac{\begin{array}{c}\mathcal{E} = \mathsf{ot} | \mathsf{emp} \quad \Gamma; \Sigma; \Delta \vdash P \\ k \cdot o@s \notin \Delta, P \ where\ k = \mathsf{pure} | \mathsf{share}\end{array}}{\Gamma; \Sigma; \Delta \vdash_{\mathcal{E}} P} \qquad \frac{\Gamma; \Sigma; \Delta \vdash P}{\Gamma; \Sigma; \Delta \vdash_{\mathsf{wt}} P}$$

Figure 1: Transaction-aware linear judgement

1.3 Thread Pool and Expression Typing

$$\boxed{k \cdot o@s \notin \Delta}$$

$$\frac{}{k \cdot o@s \notin \cdot} \qquad \frac{k \cdot o@s \notin P \quad k \cdot o@s \notin \Delta}{k \cdot o@s \notin \Delta, P}$$

$$\boxed{k \cdot o@s \notin P}$$

$$\frac{}{k \cdot o@s \notin k \cdot o@?} \qquad \frac{(k \neq k' | o \neq o' | s \neq s')}{k \cdot o@s \notin k' \cdot o'@s'}$$

$$\boxed{a; \overline{\langle \Delta^{\mathcal{E}}, u^{\mathcal{E}} \rangle}\ \textsf{ok}}$$

$$\frac{\textsf{not-wt}(\overline{\langle \Delta^{\mathcal{E}}, u^{\mathcal{E}} \rangle}_1) \quad \textsf{not-wt}(\overline{\langle \Delta^{\mathcal{E}}, u^{\mathcal{E}} \rangle}_2)}{\bullet; (\overline{\langle \Delta^{\mathcal{E}}, u^{\mathcal{E}} \rangle}_1 \langle \Delta^{\textsf{wt}}, u^{\textsf{wt}} \rangle, \overline{\langle \Delta^{\mathcal{E}}, u^{\mathcal{E}} \rangle}_2)\ \textsf{ok}} \qquad \frac{\textsf{not-wt}(\overline{\langle \Delta^{\mathcal{E}}, u^{\mathcal{E}} \rangle})}{\circ; \overline{\langle \Delta^{\mathcal{E}}, u^{\mathcal{E}} \rangle}\ \textsf{ok}}$$

Figure 2: Well-formedness of all linear contexts.

$$\boxed{\textsf{not-wt}(\overline{\langle \Delta^{\mathcal{E}}, u^{\mathcal{E}} \rangle})}$$

$$\frac{}{\textsf{not-wt}(\cdot)} \qquad \frac{\textsf{not-wt}(\langle \Delta^{\mathcal{E}}, u^{\mathcal{E}} \rangle) \quad \textsf{not-wt}(\overline{\langle \Delta^{\mathcal{E}}, u^{\mathcal{E}} \rangle})}{\textsf{not-wt}(\langle \Delta^{\mathcal{E}}, u^{\mathcal{E}} \rangle, \overline{\langle \Delta^{\mathcal{E}}, u^{\mathcal{E}} \rangle})}$$

$$\boxed{\textsf{not-wt}(\langle \Delta^{\mathcal{E}}, u^{\mathcal{E}} \rangle)}$$

$$\frac{\textsf{not-wt}(\Delta^{\mathcal{E}}) \quad where\ \mathcal{E} = \textsf{share}|\textsf{pure}}{\textsf{not-wt}(\langle \Delta^{\mathcal{E}}, u^{\mathcal{E}} \rangle)}$$

$$\boxed{\textsf{not-wt}(\Delta^{\mathcal{E}})}$$

$$\frac{k \cdot o@s \notin \Delta^{\mathcal{E}}\ where\ k = \textsf{share}|\textsf{pure}}{\textsf{not-wt}(\Delta^{\mathcal{E}})}$$

$$\boxed{\textsf{not-active}(e)}$$

$$\frac{\textsf{mbody}(C, m) = \overline{x}.e_m}{\textsf{not-active}(e_m)} \qquad \frac{}{\textsf{not-active}(k \cdot t)}$$

$$\frac{}{\textsf{not-active}(k \cdot t.f)} \qquad \frac{}{\textsf{not-active}(t_1.f := k \cdot t_2)}$$

$$\frac{}{\textsf{not-active}(\texttt{new}\ C(\overline{k \cdot t}))} \qquad \frac{}{\textsf{not-active}(k \cdot t.m(\overline{k \cdot t}))}$$

$$\frac{\textsf{not-active}(e_1) \quad \textsf{not-active}(e_2)}{\textsf{not-active}(\texttt{let}\ x = e_1\ \texttt{in}\ e_2)} \qquad \frac{\textsf{not-active}(e)}{\textsf{not-active}(\texttt{unpack}_{\mathcal{E}} k \cdot t@s\ \texttt{in}\ e)}$$

$$\frac{\textsf{not-active}(e)}{\textsf{not-active}(\texttt{pack}_{\mathcal{E}}\ t\ \texttt{to}\ s\ \texttt{in}\ e)} \qquad \frac{\textsf{not-active}(e)}{\textsf{not-active}(\texttt{atomic}\ (e))}$$

Figure 3: Expressions with no active subexpressions.

$$\boxed{\textsf{active}(e)}$$

$$\frac{}{\textsf{active}(\texttt{inatomic}\ (e))} \qquad \frac{\textsf{active}(e_1) \quad \textsf{not-active}(e_2)}{\textsf{active}(\texttt{let}\ x = e_1\ \texttt{in}\ e_2)}$$

Figure 4: Expressions with an active subexpression.

$$\boxed{\texttt{forget}(P) = P'}$$

$$\frac{k = \textsf{immutable}|\textsf{unique}|\textsf{full}}{\texttt{forget}(k \cdot o@s) = k \cdot o@s} \qquad \frac{k = \textsf{pure}|\textsf{share}}{\texttt{forget}(k \cdot o@s) = k \cdot o@?}$$

$$\frac{\texttt{forget}(P_1) = P_1' \quad \texttt{forget}(P_2) = P_2'}{\texttt{forget}(P_1 \otimes P_2) = P_1' \otimes P_2'}$$

Figure 5: The forget judgement.

$$\boxed{\texttt{forget}_{\mathcal{E}}(P) = P'}$$

$$\frac{\mathcal{E} = \mathsf{wt}}{\texttt{forget}_{\mathcal{E}}(P) = P} \quad \frac{\mathcal{E} \neq \mathsf{wt} \quad \texttt{forget}(P) = P'}{\texttt{forget}_{\mathcal{E}}(P) = P'}$$

$$\boxed{\mathsf{writes}(k)}$$

$$\frac{}{\mathsf{writes}(\mathsf{unique})} \quad \frac{}{\mathsf{writes}(\mathsf{full})} \quad \frac{}{\mathsf{writes}(\mathsf{share})}$$

$$\boxed{\mathsf{readonly}(k)}$$

$$\frac{}{\mathsf{readonly}(\mathsf{pure})} \quad \frac{}{\mathsf{readonly}(\mathsf{immutable})}$$

$$\boxed{S \leq S'}$$

$$\frac{}{S \leq S} \qquad \frac{}{S \leq ?}$$

$$\frac{}{\mathsf{unpacked}(s) \leq s} \quad \frac{}{s \leq \mathsf{unpacked}(s)}$$

$$\boxed{k \leq k'}$$

$$\frac{k \cdot o@s \Rrightarrow k' \cdot o@s}{k \leq k'} \quad \frac{k \cdot o@s \Rrightarrow k' \cdot o@s \otimes k'' \cdot o@s}{k \leq k'}$$

$$\frac{}{\Gamma; P \vdash P}\ \textsc{LinHyp} \qquad \frac{\Gamma; \Delta \vdash P' \quad P' \Rrightarrow P}{\Gamma; \Delta \vdash P}\ \textsc{Subst}$$

$$\frac{\Gamma; \Delta_1 \vdash P_1 \quad \Gamma; \Delta_2 \vdash P_2}{\Gamma; (\Delta_1, \Delta_2) \vdash P_1 \otimes P_2}\ \otimes I \qquad \frac{\Gamma; \Delta \vdash P_1 \otimes P_2 \quad \Gamma; (\Delta', P_1, P_2) \vdash P}{\Gamma; (\Delta, \Delta') \vdash P}\ \otimes E$$

$$\frac{}{\Gamma; \cdot \vdash \mathbf{1}}\ \mathbf{1}I \qquad \frac{\Gamma; \Delta \vdash \mathbf{1} \quad \Gamma; \Delta' \vdash P}{\Gamma; (\Delta, \Delta') \vdash P}\ \mathbf{1}E$$

$$\frac{\Gamma; \Delta \vdash P_1 \quad \Gamma; \Delta \vdash P_2}{\Gamma; \Delta \vdash P_1 \,\&\, P_2}\ \& I \qquad \frac{\Gamma; \Delta \vdash P_1 \,\&\, P_2}{\Gamma; \Delta \vdash P_1}\ \& E_L \qquad \frac{\Gamma; \Delta \vdash P_1 \,\&\, P_2}{\Gamma; \Delta \vdash P_2}\ \& E_R$$

$$\frac{}{\Gamma; \Delta \vdash \top}\ \top I$$

no $\top$ ***elimination***

$$\frac{\Gamma; \Delta \vdash P_1}{\Gamma; \Delta \vdash P_1 \oplus P_2}\ \oplus I_L \qquad \frac{\Gamma; \Delta \vdash P_2}{\Gamma; \Delta \vdash P_1 \oplus P_2}\ \oplus I_R$$

$$\frac{\Gamma; \Delta \vdash P_1 \oplus P_2 \quad \begin{array}{c}\Gamma; (\Delta', P_1) \vdash P \\ \Gamma; (\Delta', P_2) \vdash P\end{array}}{\Gamma; (\Delta, \Delta') \vdash P}\ \oplus E$$

no 0 introduction

$$\frac{\Gamma; \Delta \vdash \mathbf{0}}{\Gamma; (\Delta, \Delta') \vdash P}\ \mathbf{0}E$$

$$\frac{(\Gamma, z : H); \Delta \vdash P}{\Gamma; \Delta \vdash \forall z : H.P}\ \forall I \qquad \frac{\Gamma \vdash h : H \quad \Gamma; \Delta \vdash \forall z : H.P}{\Gamma; \Delta \vdash [h/z]P}\ \forall E$$

$$\frac{\Gamma \vdash h : H \quad \Gamma; \Delta \vdash [h/z]P}{\Gamma; \Delta \vdash \exists z : H.P}\ \exists I \qquad \frac{\Gamma; \Delta \vdash \exists z : H.P \quad (\Gamma, z : H), (\Delta', P) \vdash P'}{\Gamma; (\Delta, \Delta') \vdash P'}\ \exists E$$

Figure 6: Linear logic for permission reasoning

$$\boxed{\vdash a; H; T}$$

$$\frac{\begin{array}{c}\cdot; \Sigma; \overline{\Delta^{\mathcal{E}}}; \overline{u} \vdash H; \overline{S_p} \quad \Sigma; \overline{\Delta^{\mathcal{E}}}; \overline{u} \vdash T \\ \textsf{correct-atomic}(a, T) \quad \mathit{where}\ T = \overline{\langle e, S_p, \rangle}\end{array}}{\vdash a; H; T}$$

Figure 7: Top-level typing rules

$$\boxed{\Sigma; \overline{\Delta}; \overline{u} \vdash T}$$

$$\frac{}{\Sigma; \cdot; \cdot \vdash \cdot} \qquad \frac{\Sigma; \Delta_1^{\mathcal{E}}; \mathcal{E}_1; u_1 \vdash e_1 : E_1 \setminus \omega | u \quad \Sigma; \Delta_2^{\mathcal{E}}, \ldots, \Delta_n^{\mathcal{E}}; u_2, \ldots, u_n \vdash T}{\Sigma; \Delta_1^{\mathcal{E}}, \Delta_2^{\mathcal{E}}, \ldots, \Delta_n^{\mathcal{E}}; u_1, u_2, \ldots, u_n \vdash \langle e, S_p \rangle_1, T}$$

Figure 8: Well-typed thread-pool

$$\frac{(o : C) \in \Sigma \quad \cdot; \Sigma; \Delta \vdash_{\mathcal{E}} P}{\cdot; \Sigma; \Delta; \mathcal{E}; u \vdash k \cdot o : \exists x : C.[x/o]P \setminus \emptyset | u} \text{ T-LOC}$$

$$\frac{\mathsf{readonly}(k_u) \textit{ implies } \mathsf{readonly}(k) \quad \cdot; \Sigma; \Delta \vdash_{\mathcal{E}} P \quad \mathsf{localFields}(C) = \overline{f : C}}{\cdot; \Sigma; \Delta; \mathcal{E}; k_u \cdot o@S_u \vdash k \cdot o.f_i : \exists x : T_i.[x/f_i]P \setminus \emptyset | k_u \cdot o@s_u} \text{ T-READ}$$

$$\frac{\begin{array}{c}\mathsf{localFields}(C'') = \overline{f : C} \quad (o' : C') \in \Sigma \quad \mathsf{writes}(k') \\ \cdot; \Sigma; \Delta \vdash_{\mathcal{E}} k \cdot o : \exists x : C_i.P \quad \cdot; \Sigma; \Delta' \vdash_{\mathcal{E}} [o'.f_i/x']P'\end{array}}{\cdot; \Sigma; \Delta, \Delta'; \mathcal{E}; k' \cdot o'@s' \vdash o'.f_i' := k \cdot o : \exists x' : C_i.P' \otimes [o'.f_i/x]P \setminus \{o_i.f\} | k' \cdot o'@s'} \text{ T-ASSIGN}$$

$$\frac{\cdot; \Sigma; \Delta \vdash_{\mathcal{E}} [\overline{o}/\overline{f}]P \quad \overline{o : C} \subseteq \Sigma \quad \mathsf{init}(C) = \langle \exists \overline{f : C}.P, s \rangle}{\cdot; \Sigma; \Delta; \mathcal{E}; u \vdash \texttt{new } C(\overline{k \cdot o}) : \exists x : C.\mathsf{unique} \cdot x@s \setminus \emptyset | u} \text{ T-NEW}$$

$$\frac{\begin{array}{c}\texttt{forget}_{\mathcal{E}}(k \cdot o@s) = k \cdot o@\$ \\ k = \mathsf{immutable} \mid \mathsf{pure} \textit{ implies } s = s' \quad \cdot; \Sigma; \Delta', k \cdot o@\$; \mathcal{E}; - \vdash e' : E \setminus \emptyset | - \\ \mathsf{localFields}(C) = \overline{f : C} \quad (o : C) \in \Sigma \quad \cdot; \Sigma; \Delta \vdash_{\mathcal{E}} [o/\textit{this}]\mathsf{inv}_C(s, k) \\ \textit{No temporary permissions for } o.\overline{f} \textit{ in } \Delta'\end{array}}{\cdot; \Sigma; (\Delta, \Delta'); \mathcal{E}; k \cdot o@s \vdash \texttt{pack } o \texttt{ to } s' \texttt{ in } e' : E \setminus \{o\overline{f}\} | -} \text{ T-PACK}$$

$$\frac{\begin{array}{c}k = \mathsf{unique} \mid \mathsf{full} \mid \mathsf{immutable} \quad (o : C) \in \Sigma \quad \cdot; \Sigma; \Delta \vdash_{\mathcal{E}} k \cdot o@s \\ \mathcal{E} = \mathsf{emp}|\mathsf{ot} \quad \cdot; \Sigma; \Delta', [o/\textit{this}]\mathsf{inv}_C(s, k); \mathcal{E}; k \cdot o@s \vdash e' : E \setminus \omega | -\end{array}}{\cdot; \Sigma; (\Delta, \Delta'); \mathcal{E}; - \vdash \texttt{unpack}_{\mathcal{E}}\ k \cdot o@s \texttt{ in } e' : E \setminus \emptyset | -} \text{ T-UNPACK}$$

$$\frac{\begin{array}{c}(o : C) \in \Sigma \quad \cdot; \Sigma; \Delta \vdash_{\mathsf{wt}} k \cdot o@s \\ \cdot; \Sigma; \Delta', [o/\textit{this}]\mathsf{inv}_C(s, k); \mathsf{wt}; k \cdot o@s \vdash e' : E \setminus \omega | -\end{array}}{\cdot; \Sigma; (\Delta, \Delta'); \mathsf{wt}; - \vdash \texttt{unpack}_{\mathsf{wt}}\ k \cdot o@s \texttt{ in } e' : E \setminus \emptyset | -} \text{ T-UNPACK-WT}$$

$$\frac{\begin{array}{c}(o : C) \in \Sigma \quad \overline{o : C} \subseteq \Sigma \\ \cdot; \Sigma; \Delta \vdash_{\mathcal{E}} [o/\textit{this}][\overline{o}/\overline{f}]P \quad \textsf{mtype}(C, m) = \forall \overline{x : C}.P \multimap \exists x : C.P_r \\ \texttt{forget}_{\mathcal{E}}(P_r) = P'_r\end{array}}{\cdot; \Sigma; \Delta; \mathcal{E}; - \vdash k \cdot o.m(\overline{k \cdot o}) : \exists x : C.P'_r \setminus \emptyset | -} \text{ T-CALL}$$

$$\frac{\begin{array}{c}o : C \in \Sigma \quad \overline{o : C} \in \Sigma \quad \textsf{mtype}(C, m) = \forall \overline{x : C}.P \multimap E \\ \cdot; \Sigma; \Delta^{\textsf{ot}} \vdash_{\textsf{ot}} [o/\textit{this}][\overline{o}/\overline{f}]P\end{array}}{\cdot; \Sigma; \Delta; \textsf{ot}; - \vdash \texttt{spawn}\ (k \cdot o.m(\overline{k \cdot o})) : \exists_{-} : C_d.\textsf{immutable} \cdot o_d@s_d \setminus \emptyset | -} \text{ T-SPAWN}$$

$$\frac{\begin{array}{c}\Sigma; \Delta_2, P \vdash_{\mathcal{E}} P' \\ \cdot; \Sigma; \Delta_1; \mathcal{E}; u \vdash e_1 : \exists x : T.P \setminus \omega_1 | u_2 \quad x : C; \Sigma; P'; \mathcal{E}; u_2 \vdash e_2 : E\omega_2 | u' \\ \textit{No permissions for } \omega_1 \textit{ in } \Delta_2\end{array}}{\cdot; \Sigma; (\Delta_1, \Delta_2); \mathcal{E}; u \vdash \texttt{let}\ x = e_1\ \texttt{in}\ e_2 : E \setminus \omega_1 \cup \omega_2 | u'} \text{ T-LET}$$

$$\frac{\cdot; \Sigma; \Delta; \textsf{wt}; u \vdash e : \exists x : C.P \setminus \omega | u' \quad \texttt{forget}_{\mathcal{E}}(P) = P'}{\cdot; \Sigma; \Delta; \mathcal{E}; u \vdash \texttt{inatomic}\ (e) : \exists x : C.P' \setminus \omega | u'} \text{ T-INATOMIC}$$

$$\frac{\cdot; \Sigma; \Delta; \textsf{wt}; u \vdash e : \exists x : C.P \setminus \omega | u' \quad \texttt{forget}_{\mathcal{E}}(P) = P'}{\cdot; \Sigma; \Delta; \mathcal{E}; u \vdash \texttt{atomic}\ (e) : \exists x : C.P' \setminus \omega | u'} \text{ T-ATOMIC}$$

1.4 Dynamic Semantics

$$\boxed{a; H; T \to a'; H'; T'}$$

$$\frac{a; H; e \to a'; H'; e'; T'}{a; H; T_a, e, T_b \to a'; H'; T_a, e', T_b, T'}$$

Figure 9: Top-level Dynamic Semantics

$$\frac{k = \mathsf{pure} \mid \mathsf{immutable} \quad o \mapsto C(\ldots, f_i = k' \cdot o')@\mathsf{unpacked}(s'') \in H}{a; H; \langle k \cdot o.f_i, S_p\rangle \to a; H[o \mapsto C(\ldots, f_i = (k' - k) \cdot o)]; \langle k \cdot o', (S_p + k \cdot o')\rangle; \cdot} \textsc{ E-Read-R}$$

$$\frac{k \leq k' \quad o \mapsto C(\ldots, f_i = k' \cdot o')@\mathsf{unpacked}(k'') \in H}{\begin{array}{c} a; H; \langle k \cdot o.f_i, S_p\rangle \to \\ a; H[o \mapsto C(\ldots, f_i = (k' - k) \cdot o')@\mathsf{unpacked}(k'')]; \langle k \cdot o', (S_p + k \cdot o')\rangle; \cdot \end{array}} \textsc{ E-Read-RW}$$

$$\frac{\begin{array}{c} k_1 \cdot o_1 \in S_p \quad o_1 \mapsto C(\ldots, f = k' \cdot o', \ldots)@\mathsf{unpacked}(k'') \in H \\ k_2 \cdot o_2 \in S_p \quad o_2 \mapsto C(\ldots)@S_2 \in H \\ H' = H[o_1 \mapsto C(\ldots, f = k \cdot o_2, \ldots)@\mathsf{unpacked}(k'')] \quad S_p' = S_p[(k_2 - k) \cdot o_2], k' \cdot o' \end{array}}{a; H; \langle o_1.f := k \cdot o_2, S_p\rangle \to a; H'; \langle k' \cdot o', S_p'\rangle \cdot} \textsc{ E-Assign}$$

$$\frac{H; S_p \vdash [\overline{o}/\overline{f}]P \quad \mathsf{init}(C) = \langle \exists \overline{f : C}.P, s\rangle \quad S_p' = S_p - \overline{k \cdot o} \quad o_n \notin \mathsf{dom}(H)}{a; H; \langle \mathtt{new}\ C(\overline{k \cdot o}), S_p\rangle \to a; H, o_n \mapsto C(\overline{f = k \cdot o})@s; \langle \mathsf{unique} \cdot o_n, (S_p', \mathsf{unique} \cdot o_n)\rangle; \cdot} \textsc{ E-New}$$

$$\frac{\begin{array}{c} \mathcal{E} = \mathsf{ot} \mid \mathsf{emp} \quad k' \cdot o \in S_p \quad \mathsf{readonly}(k) \quad o \mapsto C(\ldots)@S \in H \quad k \leq k' \\ k = \mathsf{immutable} \supset S = (\mathsf{unpacked}(s) | s), k = \mathsf{pure} \supset S = s \end{array}}{\begin{array}{c} a; H; \langle \mathtt{unpack}_{\mathcal{E}}\ k \cdot o@S\ \mathtt{in}\ e', S_p\rangle \to \\ a; H[o \mapsto C(\ldots)@\mathsf{unpacked}(s)]; \langle e', S_P[(k' - k) \cdot o]\rangle; \cdot \end{array}} \textsc{ E-Unpack-R}$$

$$\frac{\begin{array}{c} k' \cdot o \in S_p \quad \mathsf{readonly}(k) \quad o \mapsto C(\ldots)@S \in H \quad k \leq k' \\ k = \mathsf{immutable} \supset S = (\mathsf{unpacked}(s) | s), k = \mathsf{pure} \supset S = s \end{array}}{\begin{array}{c} \circ; H; \langle \mathtt{unpack}_{\mathsf{wt}}\ k \cdot o@s\ \mathtt{in}\ e', S_p\rangle \to \\ \circ; H[o \mapsto C(\ldots)@\mathsf{unpacked}(s)]; \langle e', S_P[(k' - k) \cdot o]\rangle; \cdot \end{array}} \textsc{ E-Unpack-R-Wt}$$

$$\frac{\mathcal{E} = \mathsf{ot} \mid \mathsf{emp} \quad k' \cdot o \in S_p \quad \mathsf{writes}(k) \quad o \mapsto C(\ldots)@s \in H \quad k \leq k'}{\begin{array}{c} a; H; \langle \mathtt{unpack}_{\mathcal{E}}\ k \cdot o@s\ \mathtt{in}\ e', S_p\rangle \to \\ a; H[o \mapsto C(\ldots)@\mathsf{unpacked}(k)]; \langle e', S_P[(k' - k) \cdot o]\rangle; \cdot \end{array}} \textsc{ E-Unpack-RW}$$

$$\frac{k' \cdot o \in S_p \quad \mathsf{writes}(k) \quad o \mapsto C(\ldots)@s \in H \quad k \leq k'}{\begin{array}{c} \circ; H; \langle \mathtt{unpack}_{\mathsf{wt}}\ k \cdot o@s\ \mathtt{in}\ e', S_p\rangle \to \\ \circ; H[o \mapsto C(\ldots)@\mathsf{unpacked}(k)]; \langle e', S_P[(k' - k) \cdot o]\rangle; \cdot \end{array}} \textsc{ E-Unpack-RW-Wt}$$

$$\frac{\begin{array}{c} \mathsf{inv}_C(s)\ satisfied\ by\ o's\ fields \\ k_o \cdot o \in S_p \quad o \mapsto C(\overline{f = k \cdot o})@\mathsf{unpacked}(s) \in H \end{array}}{a; H; \langle \mathtt{pack}\ o\ \mathtt{to}\ s'\ \mathtt{in}\ e', S_p\rangle \to a; H[o \mapsto C(\overline{f = k \cdot o})@s]; \langle e', S_p\rangle} \textsc{ E-Pack-R}$$

$$\frac{\begin{array}{c} \mathsf{inv}_C(s)\ satisfied\ by\ o's\ fields \\ k_o \cdot o \in S_p \quad o \mapsto C(\overline{f = k \cdot o})@\mathsf{unpacked}(k) \in H \end{array}}{a; H; \langle \mathtt{pack}\ o\ \mathtt{to}\ s\ \mathtt{in}\ e', S_p\rangle \to a; H[o \mapsto C(\overline{f = k \cdot o})@s]; \langle e', S_p[(k + k_o) \cdot o]\rangle} \textsc{ E-Pack-RW}$$

$$\frac{\mathsf{mbody}(C,m) = \overline{x}.e_m \quad \mathsf{mtype}(C,m) = \forall \overline{x:C}.P \multimap E \qquad H; S_p | k \cdot o, \overline{k \cdot o} \vdash [o/\textit{this}][\overline{o}/\overline{x}]P}{a; H; \langle k \cdot o.m(\overline{k \cdot o}), S_p \rangle \rightarrow a; H; \langle [o/\textit{this}][\overline{o}/\overline{x}]e_m, S_p \rangle; \cdot} \text{ E-CALL}$$

$$\frac{\mathsf{mbody}(C,m) = \overline{x}.e_m \quad \mathsf{mtype}(C,m) = \forall \overline{x:C}.P \multimap E \qquad H; S_{p_2} | k \cdot o, \overline{k \cdot o} \vdash [o/\textit{this}][\overline{o}/\overline{x}]P}{\circ; H; \langle \texttt{spawn}\ (k \cdot o.m(\overline{k \cdot o})), (S_{p_1}, S_{p_2}) \rangle \rightarrow \circ; H; \langle o_d, S_{p_1} \rangle; \langle [o/\textit{this}][\overline{o}/\overline{x}]e_m, S_{p_2} \rangle} \text{ E-SPAWN}$$

$$\frac{a; H; \langle e_1, S_p \rangle \rightarrow a'; H'; \langle e_1', S_p' \rangle; T}{a; H; \langle \texttt{let}\ x = e_1\ \texttt{in}\ e_2, S_p \rangle \rightarrow a'; H'; \langle \texttt{let}\ x = e_1'\ \texttt{in}\ e_2, S_p' \rangle; T} \text{ E-LET-E}$$

$$\frac{k' \cdot o \in S_p \quad o \mapsto C(\ldots)@S \in H \quad k \leq k'}{a; H; \langle \texttt{let}\ x = k \cdot o\ \texttt{in}\ e_2, S_p \rangle \rightarrow a; H; \langle [o/x]e_2, S_p \rangle; \cdot} \text{ E-LET-V}$$

$$\frac{}{\circ; H; \langle \texttt{atomic}\ (e), S_p \rangle \rightarrow \bullet; H; \langle \texttt{inatomic}\ (e), S_p \rangle; \cdot} \text{ E-ATOMIC-BEGIN}$$

$$\frac{}{\bullet; H; \langle \texttt{inatomic}\ (k \cdot o), S_p \rangle \rightarrow \circ; H; \langle k \cdot o, S_p \rangle; \cdot} \text{ E-ATOMIC-EXIT}$$

$$\frac{a; H; \langle e, S_p \rangle \rightarrow a'; H'; \langle e', S_p' \rangle; T}{\bullet; H; \langle \texttt{inatomic}\ (e), S_p \rangle \rightarrow \bullet; H'; \langle \texttt{inatomic}\ (e'), S_p' \rangle; T} \text{ E-INATOMIC}$$

1.5 Preservation

1.5.1 Definition of Store Typing

$$\Sigma; \overline{\Delta^{\mathcal{E}}}; \overline{u^{\mathcal{E}}} \vdash H; \overline{S_p}$$

The above judgement is true if:

1. $\Sigma; \overline{\Delta^{\mathcal{E}} \vdash S_p}$
2. $\Sigma; \overline{\Delta^{\mathcal{E}}}; \overline{S_p}; \overline{u^{\mathcal{E}}} \vdash H$

$$\boxed{\Sigma; \overline{\Delta^{\mathcal{E}} \vdash S_p}}$$

$$\frac{\Sigma; \Delta_1^{\mathcal{E}} \vdash S_{p1} \quad \Sigma; \Delta_2^{\mathcal{E}}, \ldots, \Delta_n^{\mathcal{E}} \vdash S_{p2} \ldots S_{pn}}{\Sigma; \Delta_1^{\mathcal{E}}, \Delta_2^{\mathcal{E}}, \ldots, \Delta_n^{\mathcal{E}} \vdash S_{p1}, S_{p2}, \ldots, S_{pn}} \qquad \frac{}{\Sigma; \cdot \vdash \cdot}$$

$$\boxed{\Sigma; \Delta^{\mathcal{E}} \vdash S_p}$$

$$\frac{\{o|k \cdot o@\$ \in \Delta\} \subseteq \{o|k \cdot o \in S_p\} \quad \forall k \cdot o \in S_p \cdot; \Sigma; \Delta \vdash k' \cdot o@\$ \otimes \top \supset k' \leq k}{\Sigma; \Delta^{\mathcal{E}} \vdash S_p}$$

Where the above rule ignores permissions on fields.

$$\boxed{\Sigma; \overline{\Delta^{\mathcal{E}}}; \overline{S_p}; \overline{u^{\mathcal{E}}} \vdash H}$$

The above judgement is true if:

1. $\mathsf{dom}(\Sigma) = \mathsf{dom}(H)$
2. $a; \langle\overline{\Delta^{\mathcal{E}}, u^{\mathcal{E}}}\rangle$ ok
3. $\forall u^{\mathcal{E}} \in \overline{u^{\mathcal{E}}}, u^{\mathcal{E}} = k \cdot o@s \supset \mathcal{E} = \mathsf{wt}|k = \mathsf{immutable}|\mathsf{unique}|\mathsf{full}$ and $o \mapsto C(\ldots)@S \in H$, where $S = \mathsf{unpacked}(s)$ if $\mathsf{readonly}(k)$ or $S = \mathsf{unpacked}(k)$ if $\mathsf{writes}(k)$. Also, $o \notin \overline{u} \supset$ o is packed in H and $\mathsf{inv}_C(o, \mathsf{unique})$.
4. $\forall o \in \mathsf{dom}(\Sigma), \forall \Delta \in \overline{\Delta}$: if $o \mapsto C(\overline{f = k \cdot o})@S \in H$ then
   (a) $(o : C) \in \Sigma$
   (b) Either $S = \mathsf{unpacked}(k)$ or $S = \mathsf{unpacked}(s)$ and $[o/\textit{this}]\mathsf{inv}_C(s, \mathsf{immutable})$ is satisfied by o's fields, or $S = s$ and $[o/\textit{this}]\mathsf{inv}_C(s, \mathsf{unique})$ is satisfied by o's fields.
   (c) If $\cdot; \Sigma; \Delta \vdash k \cdot o@\$ \otimes \top$ then $S \leq \$$.
   (d) If $\cdot; \Sigma; \Delta \vdash k_i' \cdot o.f_i@\$ \otimes \top$, then $k_i' \leq k_i$ (and $o = o_{unp}$) and $o_i \mapsto C_o(\ldots)@s_o \in H$ and either $S = \mathsf{unpacked}(s)$, which implies $\mathsf{readonly}(k_i')$, or $S = \mathsf{unpacked}(k')$. If $S = \mathsf{unpacked}(s)$ then $\$ = s_o$ or $\$ =?$.
   (e) $\mathsf{unique} \cdot o@s \in \Delta, u \supset k \cdot o@\$$ not in any other Δ or u in $\overline{\Delta}$ or $\overline{u}$. Also, $\mathsf{full} \cdot o@s \in \Delta, u \supset \mathsf{full} \cdot o@\$$ and $k \cdot o@s$ not in any other Δ or u in $\overline{\Delta}$ or $\overline{u}$.
   (f) $\mathsf{immutable} \cdot o@s \in \Delta, u \supset (k \cdot o@\$ \in \overline{\Delta}, \overline{u} \supset k = \mathsf{immutable}\&(\$ = s|\$ =?))$
   (g) Where $k_i = \mathsf{unique}$ implies $k \cdot o_i \notin \overline{\Delta}, \overline{u}$, where $k_i = \mathsf{full}$ implies $\mathsf{full} \cdot o_i@\$$ and $k \cdot o_i@s \notin \overline{\Delta}, \overline{u}$ and where $k_i = \mathsf{immutable}$ and $o \mapsto C(\ldots)@S$, where $S = s|\mathsf{unpacked}(s)$ implies $k' \cdot o_i@s' \notin \overline{\Delta}, \overline{u}$, where $k' \neq \mathsf{immutable}|s' \neq s$.

1.5.2 Property Satisfied at Runtime

If

- $\overline{o \mapsto C(\ldots)@s} \subseteq H$ and $\overline{k' \cdot o} \in S_p$
- $\cdot|\overline{o : C}|\overline{k \cdot o@s} \vdash P$ (an instance of $\Gamma|\Sigma|\Delta \vdash P$)
- $\overline{k \leq k'}$

then $H; S_p|\overline{k \cdot o} \vdash P$

1.5.3 Lemma: Compositionality

If $\Sigma; \overline{\Delta^{\mathcal{E}}}; \overline{u^{\mathcal{E}}} \vdash H; \overline{S_p}$ and $\Delta_i = \Delta_{i1}, \Delta_{i2}$ then $\Sigma; \overline{\Delta^{\mathcal{E}'}}; \overline{u^{\mathcal{E}}} \vdash H; \overline{S_p}$ where Δ_i is replaced with Δ_{i1} and and $\Sigma; \overline{\Delta^{\mathcal{E}''}}; \overline{u^{\mathcal{E}}} \vdash H; \overline{S_p}$ where Δ_i is replaced with Δ_{i2}.
Proof: Immediate from the definition of store typing. We are always allowed to know less statically about permissions than what is true at run-time, so long as what we know statically is consistent with the run-time information.

1.5.4 Lemma: Packing Flag

If $\Gamma; \Sigma; \Delta; \mathcal{E}; u \vdash e : E \setminus \omega|u'$ then either (a) $u = -$ and $\omega = \emptyset$ or (b) $u = k \cdot t@s$ and ω contains only fields of t.
Proof:(a) $u = -$ is not a valid precondition for producing effects (using assignment or packing). (b) By induction on typing derivations, using (a). Only one object can be unpacked at a time, permission for unpacked object is needed for assignments and packing, and effect of `unpack` expression is $\emptyset$.

1.5.5 Object Weight

- $w(o, \Delta) = \Sigma_{k \cdot o \in \Delta} k$, ignoring fields.
- $w(o, u) = k$, if $u = k \cdot o@s$, and 0 otherwise.
- $w(o, S_p) = \Sigma_{k \cdot o \in S_p} k$

  Where:

$$k + k'$$

is defined as:

- $\mathsf{full} + \mathsf{pure} = \mathsf{full}$
- $\mathsf{share} + \mathsf{pure} = \mathsf{share}$, $\mathsf{share} + \mathsf{share} = \mathsf{share}$
- $\mathsf{immutable} + \mathsf{immutable} = \mathsf{immutable}$
- $\mathsf{pure} + \mathsf{pure} = \mathsf{pure}$

1.5.6 Preservation for Thread Pools

If

- $a; H; T \rightarrow a'; H'; T'$
- $\vdash a; H; T$

Then there exists

- $\Sigma' \supseteq \Sigma$
- $\overline{\Delta^{\mathcal{E}}}'$
- $\overline{u^{\mathcal{E}}}'$
- $\overline{\omega'}$

such that

- $\Sigma'; \overline{\Delta}'; \mathsf{ot}; \overline{u}' \vdash \overline{e'} : \overline{E} \setminus \overline{\omega'} | \overline{u''}$
- $\Sigma'; \overline{\Delta^{\mathcal{E}}}'; \overline{u^{\mathcal{E}}}' \vdash H'; \overline{S_p}'$, where $T' = \overline{< e', S'_p >}$
- correct-atomic(a',T')
- $\forall o \in \mathsf{dom}(H) : w(o, S_p) - w(o, \Delta) - w(o, u) \leq w(o, S'_p) - w(o, \Delta') - w(o, u')$, for each Δ in $\overline{\Delta}$, S_p in $\overline{S_p}$, Δ in $\overline{\Delta}'$, and S_p in $\overline{S_p}'$

Proof: By structural induction on the derivation of $a; H; T \rightarrow a'; H'; T'$.

CASE TOP-LEVEL

$\vdash a; H; T$ Assumption
$a; H; e \rightarrow a'; H'; e'; T'$ where $T_a || e || T_b$
Inversion of only eval rule.
$\cdot; \Sigma; \overline{\Delta^{\mathcal{E}}}; \overline{u} \vdash H; \overline{S_p}$
$\Sigma; \overline{\Delta^{\mathcal{E}}}; \overline{u} \vdash T$
$\mathsf{correct\text{-}atomic}(a, T) \quad where\ T = \overline{\langle e, S_p, \rangle}$
Inversion of only typing rule. $\cdot; \Sigma; \Delta; \mathsf{ot}; u \vdash e : E \setminus \omega | u''$
From well-typed thread pool.
Invoke preservation for single threads.
Σ',$\Delta^{\mathcal{E}'}$, $u^{\mathcal{E}'}$, ω', a', s.t.
$\cdot; \Sigma'; \Delta'; \mathsf{ot}; u' \vdash e' : E \setminus \omega' | u''$
Single-threaded lemma.
If $T \neq \cdot$:

$\cdot; \Sigma'; \Delta_t^{\mathsf{ot}}; \mathsf{ot}; - \vdash e_t : E_t \setminus \omega_t | u'''$

Single-threaded lemma.

$\Sigma; \overline{\Delta^{\mathcal{E}}}; \overline{u^{\mathcal{E}}} \vdash H; \overline{S_p}$ Single-threaded lemma.

$\forall o \in \mathsf{dom}(H) : w(o, S_p) - w(o, \Delta) - w(o, u) \leq w(o, S_p') - w(o, \Delta') - w(o, u')$,
for each Δ in $\overline{\Delta}$, S_p in $\overline{S_p}$, Δ in $\overline{\Delta}'$, and S_p in $\overline{S_p}'$

Single-threaded lemma.

$\mathsf{not\text{-}active}(T')$ by single-threaded lemma.

If $a = \circ$ implies $\mathsf{not\text{-}active}(T)$. If $a' = \bullet$, then by single-threaded lemma $\mathsf{active}(e')$. If $a' = \circ$ the by single-threaded lemma $\mathsf{not\text{-}active}(e')$. Thus, $\mathsf{correct\text{-}atomic}(a', T')$.
If $a = \bullet$ and $\mathsf{active}(e)$ implies $\mathsf{not\text{-}active}(T_a || T_b)$. If $a' = \bullet$ then by single-threaded lemma $\mathsf{active}(e')$. If $a' = \circ$ then by the single-threaded lemma $\mathsf{not\text{-}active}(e')$ Thus, $\mathsf{correct\text{-}atomic}(a', T')$.
If $a = \bullet$ and $\mathsf{not\text{-}active}(e)$ implies $\mathsf{active}(T_a)$ or $\mathsf{active}(T_b)$. Only one may be active but neither will change during e's step, so $a' = \bullet$. Single-threaded lemma gives us $\mathsf{not\text{-}active}(e')$ Thus, $\mathsf{correct\text{-}atomic}(a', T')$.

1.5.7 Preservation for Single Threads

If

- $\Sigma; \Delta; \mathcal{E}; u \vdash e : E \setminus \omega | u''$
- $\Sigma; \overline{\Delta^{\mathcal{E}}}; \overline{u^{\mathcal{E}}} \vdash H; \overline{S_p}$, where $\overline{\Delta^{\mathcal{E}}} = (\Delta_1, \Delta_1^*)^{\mathcal{E}}, (\Delta_2, \Delta_2^*)^{\mathcal{E}}, \ldots (\Delta_n, \Delta_n^*)^{\mathcal{E}}$, where Δ_i^* contains extra permissions that contain no temporary state information and no permissions for fields in ω.
- $a; H; \langle e, S_p \rangle \rightarrow a'; H'; \langle e', S_p' \rangle; T$
- And exactly one of the following:
  - $a = \circ$ and $\mathsf{not\text{-}active}(e)$
  - $a = \bullet$ and $\mathsf{not\text{-}active}(e)$
  - $a = \bullet$ and $\mathsf{active}(e)$

  Then there exists

- $\Sigma' \supseteq \Sigma$
- u' tagged with $\mathcal{E}$, written $u^{\mathcal{E}'}$.
- ω', where either (a) $a; H; \langle e, S_p \rangle \rightarrow a'; H'; \langle e', S_p' \rangle; T$ unpacks an object o, i.e., $u^{\mathcal{E}} = -$ and $u^{\mathcal{E}'} = k \cdot o@s$ and $\omega' - \omega$ only mentions fields of o, or (b) $\omega' \subseteq \omega$.

- Δ' tagged with $\mathcal{E}$, written $\Delta^{\mathcal{E}'}$.
- S_{pt}, $\Delta_t^{\mathcal{E}}$ and $u_t^{\mathcal{E}}$.

such that

- T is either e_t or $\cdot$
- $\cdot; \Sigma'; \Delta'; \mathcal{E}; u' \vdash e' : E \setminus \omega' | u''$
- $\Sigma; \overline{\Delta^{\mathcal{E}}}; \overline{u^{\mathcal{E}}} \vdash H; \overline{S_p}$, where $\overline{\Delta}'$ and $\overline{S_p}'$ are $\overline{\Delta}$ and $\overline{S_p}$ with (Δ', Δ^*) swapped for (Δ, Δ^*) and S_p' swapped for S_p (and including S_{pt} and Δ_t if $T = e_t$).
- $\forall o \in \mathsf{dom}(H) : w(o, S_p) - w(o, \Delta) - w(o, u) \leq w(o, S_p') - w(o, \Delta') - w(o, u')$, for each Δ in $\overline{\Delta}$, S_p in $\overline{S_p}$, Δ in $\overline{\Delta}'$, and S_p in $\overline{S_p}'$
- If T is e_t then $\cdot; \Sigma'; \Delta_t; \mathsf{ot}; - \vdash e_t : E_t \setminus \omega | -$
- As well as all of the following, although exactly one will not be vaccuous:
  - if $a = a'$ and $\mathsf{not\text{-}active}(e)$ then $\mathsf{not\text{-}active}(e')$
  - if $a = a'$ and $\mathsf{active}(e)$ then $\mathsf{active}(e')$
  - if $a = \circ$ and $a' = \bullet$ and $\mathsf{not\text{-}active}(e)$ then $\mathsf{active}(e')$
  - if $a = \bullet$ and $a' = \circ$ and $\mathsf{active}(e)$ then $\mathsf{not\text{-}active}(e')$

Proof: By structural induction on the derivation of $a; H; \langle e, S_p \rangle \to a'; H'; \langle e', S_p' \rangle; T$.

CASE E-UNPACK-RW-WT

So $e = \texttt{unpack}_{\mathsf{wt}}\ k \cdot o@s\ \texttt{in}\ e_2$, $e' = e_2$, $a = a' = \circ$, $H' = H[o \mapsto C(\ldots)@\mathsf{unpacked}(k)]$, $S_p' = S_p[(k' - k) \cdot o]$ and $T = \cdot$.

$\cdot; \Sigma; \Delta; \mathcal{E}; u \vdash e : E \setminus \omega | u''$ — Assumption

$\Sigma; \overline{\Delta^{\mathcal{E}}}; \overline{u^{\mathcal{E}}} \vdash H; \overline{S_p}$ — Assumption

$\mathsf{writes}(k) \quad o \mapsto C(\ldots)@s \in H \quad k' \cdot o \in S_p\ k \leq k'$

Inversion of only eval rule

$o \in \Sigma \quad \Delta^{\mathcal{E}} = \Delta^{\mathsf{wt}} = (\Delta_1^{\mathsf{wt}}, \Delta_2^{\mathsf{wt}}) = (k \cdot o@s, \Delta_2)$

$\cdot; \Sigma; \Delta_1 \vdash_{\mathsf{wt}} k \cdot o@s \quad u = u'' = - \quad \omega = \emptyset$

$\cdot; \Sigma; \Delta_2, [o/\mathit{this}]\mathsf{inv}_C(s, k); \mathsf{wt}; k \cdot o@s \vdash e_2 : E \setminus \omega_2 | -$ — Inversion of only typing rule

Let $\Sigma' = \Sigma$, $\quad \Delta^{\mathsf{wt}'} = \Delta_2, [o/\mathit{this}]\mathsf{inv}_C(s, k)$, $\quad u^{\mathsf{wt}'} = k \cdot o@s$, $\quad \omega' = \omega_2$.

$\cdot; \Sigma'; \Delta'; \mathsf{wt}; u' \vdash e' : E \setminus \omega' | -$ — Substitution

Must show $\Sigma; \overline{\Delta^{\mathcal{E}}}; \overline{u^{\mathcal{E}}} \vdash H; \overline{S_p}$

$\Sigma'; (\Delta^{\mathsf{wt}'}, \Delta^*); \vdash S_p'$

We have removed $k \cdot o$ from Δ and S_p, and added field perms to Δ which are ignored.

$\Sigma'; \overline{\Delta'} \vdash \overline{S_p'}$

No other Δ or S_p changed.

Must show $\Sigma'; \overline{\Delta^{\mathcal{E}'}; S'_p; u^{\mathcal{E}'}} \vdash H'$

1.) ok — No change to $\mathsf{dom}(\Sigma)$ or $\mathsf{dom}(H)$

2.) ok — $\Delta^{\mathsf{wt}'}$ and $u^{\mathsf{wt}'}$ were and remain the only wt elements.

3.) ok — For $u^{\mathsf{wt}'}$, $\mathcal{E} = \mathsf{wt}$. $o \mapsto C(\ldots)@\mathsf{unpacked}(k) \in H'$ and $\mathsf{writes}(k)$.

4.a.) ok — No change

4.b.) ok — $S = \mathsf{unpacked}(k)$

4.c.) ok — No new stack perms in Δ'.

4.d.) ok — 4.b. was true before step. Fields added to Δ' are given by $\mathsf{inv}_C(s, k)$.

4.e.) ok — 4.g. was true before step. Any unique or full fields cannot be in other Δs and u.

4.f.) ok — 4.g. was true before step. Other permissions to fields must agree with state.

4.g.) ok — No fields altered.

o was unpacked. — $u = -$ and $u' = k \cdot o@s$.

$\omega' - \omega = \omega'$ only contains fields of o. — Packing flag lemma

$\forall o \in \mathsf{dom}(H) : w(o, S_p) - w(o, \Delta) - w(o, u) \leq w(o, S'_p) - w(o, \Delta') - w(o, u')$

Net is unchanged. Permission moved from Δ to u'.

$T = \cdot$

$\mathsf{not\text{-}active}(\texttt{unpack})$ implies $\mathsf{not\text{-}active}(e_2)$ — Inversion of $\mathsf{not\text{-}active}$.

$\mathsf{active}(\texttt{unpack})$ cannot be derrived.

$a = a'$ and $\mathsf{not\text{-}active}(e)$ implies $\mathsf{not\text{-}active}(e')$ — Above

CASE E-UNPACK-RW

So $e = \texttt{unpack}_{\mathcal{E}}\ k \cdot o@s\ \texttt{in}\ e_2$, $e' = e_2$, $a = a'$, $H' = H[o \mapsto C(\ldots)@\mathsf{unpacked}(k)]$, $S'_p = S_p[(k' - k) \cdot o]$ and $T = \cdot$.

$\cdot; \Sigma; \Delta; \mathcal{E}; u \vdash e : E \setminus \omega | u''$ — Assumption

$\Sigma; \overline{\Delta^{\mathcal{E}}}; \overline{u^{\mathcal{E}}} \vdash H; \overline{S_p}$ — Assumption

$\mathsf{writes}(k) \quad o \mapsto C(\ldots)@s \in H \quad k' \cdot o \in S_p\ k \leq k' \quad \mathcal{E} = \mathsf{ot}|\mathsf{emp}$

Inversion of only eval rule

$o \in \Sigma \quad \Delta^{\mathcal{E}} = (\Delta_1, \Delta_2)$

$\cdot; \Sigma; \Delta_1 \vdash_{\mathcal{E}} k \cdot o@s \quad u = u'' = - \quad \omega = \emptyset$

$k = \mathsf{unique}|\mathsf{full}|\mathsf{immutable} \quad \cdot; \Sigma; \Delta_2, [o/\mathit{this}]\mathsf{inv}_C(s, k); \mathcal{E}; k \cdot o@s \vdash e_2 : E \setminus \omega_2 | -$

Inversion of only typing rule

Let $\Sigma' = \Sigma, \quad \Delta^{\mathcal{E}'} = \Delta_2, [o/\mathit{this}]\mathsf{inv}_C(s, k), \quad u^{\mathcal{E}'} = k \cdot o@s, \quad \omega' = \omega_2.$

$\cdot; \Sigma'; \Delta'; \mathcal{E}; u' \vdash e' : E \setminus \omega' | -$ — Substitution

Must show $\Sigma; \overline{\Delta^{\mathcal{E}}}; \overline{u^{\mathcal{E}}} \vdash H; \overline{S_p}$

$\Sigma'; (\Delta^{\mathcal{E}'}, \Delta^*); \vdash S'_p$

We have removed $k \cdot o$ from Δ and S_p, and added field perms to Δ which are ignored.

$\Sigma'; \overline{\Delta'} \vdash \overline{S'_p}$

No other Δ or S_p changed.

Must show $\Sigma'; \overline{\Delta^{\mathcal{E}'}; S'_p; u^{\mathcal{E}'}} \vdash H'$

1.) ok — No change to $\mathsf{dom}(\Sigma)$ or $\mathsf{dom}(H)$

2.) ok We have not changed the number of wt elements from before.

If $\mathcal{E} \neq \mathsf{wt}$, then $\mathsf{not\text{-}wt}(\Delta')$ because invariants cannot contain pure and shared information.

3.) ok $k = \mathsf{immutable}|\mathsf{full}|\mathsf{unique}$. $o \mapsto C(\ldots)@\mathsf{unpacked}(k) \in H'$ and $\mathsf{writes}(k)$.

4.a.) ok No change

4.b.) ok $S = \mathsf{unpacked}(k)$

4.c.) ok No new stack perms in Δ'.

4.d.) ok 4.b. was true before step. Fields added to Δ' are given by $\mathsf{inv}_C(s, k)$.

4.e.) ok 4.g. was true before step. Any unique or full fields cannot be in other Δs and u.

4.f.) ok 4.g. was true before step. Other permissions to fields must agree with state.

4.g.) ok No fields altered.

o was unpacked. $u = -$ and $u' = k \cdot o@s$.

$\omega' - \omega = \omega'$ only contains fields of o. packing flag lemma

$\forall o \in \mathsf{dom}(H) : w(o, S_p) - w(o, \Delta) - w(o, u) \leq w(o, S'_p) - w(o, \Delta') - w(o, u')$

Net is unchanged. Permission moved from Δ to u'.

$T = \cdot$

$\mathsf{not\text{-}active}(\texttt{unpack})$ implies $\mathsf{not\text{-}active}(e_2)$ Inversion of $\mathsf{not\text{-}active}$.

$\mathsf{active}(\texttt{unpack})$ cannot be derrived.

$a = a'$ and $\mathsf{not\text{-}active}(e)$ implies $\mathsf{not\text{-}active}(e')$ Above

CASE E-UNPACK-R

So $e = \texttt{unpack}_{\mathcal{E}}\ k \cdot o@s\ \texttt{in}\ e_2$, $e' = e_2$, $a = a'$, $H' = H[o \mapsto C(\ldots)@\mathsf{unpacked}(s)]$, $S'_p = S_p[(k' - k) \cdot o]$ and $T = \cdot$.

$\cdot; \Sigma; \Delta; \mathcal{E}; u \vdash e : E \setminus \omega | u''$ Assumption

$\Sigma; \overline{\Delta^{\mathcal{E}}}; \overline{u^{\mathcal{E}}} \vdash H; \overline{S_p}$ Assumption

$\mathcal{E} = \mathsf{ot}\ |\ \mathsf{emp} \quad k' \cdot o \in S_p \quad \mathsf{readonly}(k) \quad o \mapsto C(\ldots)@S \in H \quad k \leq k'$

$k = \mathsf{immutable} \supset S = (\mathsf{unpacked}(s)|s), k = \mathsf{pure} \supset S = s$

Inversion of only eval rule

$\cdot; \Sigma; (\Delta_1, \Delta_2); \mathcal{E}; - \vdash \texttt{unpack}_{\mathcal{E}}\ k \cdot o@s\ \texttt{in}\ e_2 : E \setminus \emptyset | -$

$k = \mathsf{unique}\ |\ \mathsf{full}\ |\ \mathsf{immutable} \quad (o : C) \in \Sigma \quad \cdot; \Sigma; \Delta_1 \vdash_{\mathcal{E}} k \cdot o@s$

$\mathcal{E} = \mathsf{emp}|\mathsf{ot} \quad \cdot; \Sigma; \Delta_2, [o/\mathit{this}]\mathsf{inv}_C(s, k); \mathcal{E}; k \cdot o@s \vdash e_2 : E \setminus \omega_2 | -$

Inversion of only typing rule

Let $\Sigma' = \Sigma, \quad \Delta^{\mathcal{E}'} = \Delta_2, [o/\mathit{this}]\mathsf{inv}_C(s, k), \quad u^{\mathcal{E}'} = k \cdot o@s, \quad \omega' = \omega_2.$

$\cdot; \Sigma'; \Delta'; \mathcal{E}; u' \vdash e' : E \setminus \omega' | -$ Substitution

Must show $\Sigma; \overline{\Delta^{\mathcal{E}}}; \overline{u^{\mathcal{E}}} \vdash H; \overline{S_p}$

$\Sigma'; (\Delta^{\mathcal{E}'}, \Delta^*); \vdash S'_p$

We have removed $k \cdot o$ from Δ and S_p, and added field perms to Δ which are ignored.

$\Sigma'; \overline{\Delta'} \vdash \overline{S'_p}$

No other Δ or S_p changed.

Must show $\Sigma'; \overline{\Delta^{\mathcal{E}'}; S'_p; u^{\mathcal{E}'}} \vdash H'$

1.) ok No change to $\mathsf{dom}(\Sigma)$ or $\mathsf{dom}(H)$

2.) ok We have not changed the number of wt elements from before.

If $\mathcal{E} \neq \mathsf{wt}$, then $\mathsf{not\text{-}wt}(\Delta')$ because invariants cannot contain pure and shared information.

3.) ok $k = \mathsf{immutable}|\mathsf{full}|\mathsf{unique}$. $o \mapsto C(\ldots)@\mathsf{unpacked}(k) \in H'$ and $\mathsf{writes}(k)$.

4.a.) ok No change

4.b.) ok Before step, either $S = \mathsf{unpacked}(s)$ and invariant holds by this rule, or $S = s$ and invariant held by this rule. Fields have not changed.

4.c.) ok No new stack perms in Δ'.

4.d.) ok 4.b. was true before step. Fields added to Δ' are given by $\mathsf{inv}_C(s,k)$.

4.e.) ok 4.g. was true before step. Any unique or full fields cannot be in other Δs and u.

4.f.) ok 4.g. was true before step. Other permissions to fields must agree with state.

4.g.) ok No fields altered.

o was unpacked. $u = -$ and $u' = k \cdot o@s$.

$\omega' - \omega = \omega'$ only contains fields of o. Packing Flag lemma

Δ' does not contain any fields in $\omega - \omega'$ $\omega - \omega' = \emptyset$

$\forall o \in \mathsf{dom}(H) : w(o, S_p) - w(o, \Delta) - w(o, u) \leq w(o, S_p') - w(o, \Delta') - w(o, u')$

Net is unchanged. Permission moved from Δ to u'.

$T = \cdot$

$\mathsf{not\text{-}active}(\mathtt{unpack})$ implies $\mathsf{not\text{-}active}(e_2)$ Inversion of $\mathsf{not\text{-}active}$.

$\mathsf{active}(\mathtt{unpack})$ cannot be derrived.

$a = a'$ and $\mathsf{not\text{-}active}(e)$ implies $\mathsf{not\text{-}active}(e')$ Above

CASE E-UNPACK-R-WT

So $e = \mathtt{unpack}_{\mathsf{wt}}\ k \cdot o@s\ \mathtt{in}\ e_2$, $e' = e_2$, $a = a'$, $H' = H[o \mapsto C(\ldots)@\mathsf{unpacked}(s)]$, $S_p' = S_p[(k' - k) \cdot o]$ and $T = \cdot$.

$\cdot; \Sigma; \Delta; \mathcal{E}; u \vdash e : E \setminus \omega | u''$ Assumption

$\Sigma; \overline{\Delta^{\mathcal{E}}}; \overline{u^{\mathcal{E}}} \vdash H; \overline{S_p}$ Assumption

$k' \cdot o \in S_p \quad \mathsf{readonly}(k) \quad o \mapsto C(\ldots)@S \in H \quad k \leq k'$

$k = \mathsf{immutable} \supset S = (\mathsf{unpacked}(s)|s), k = \mathsf{pure} \supset S = s$

Inversion of only evaluation rule.

$\cdot; \Sigma; (\Delta_1, \Delta_2); \mathsf{wt}; - \vdash \mathtt{unpack}_{\mathsf{wt}}\ k \cdot o@s\ \mathtt{in}\ e' : E \setminus \emptyset | -$

$(o : C) \in \Sigma \quad \cdot; \Sigma; \Delta_1 \vdash_{\mathsf{wt}} k \cdot o@s \quad \cdot; \Sigma; \Delta_2, [o/\mathit{this}]\mathsf{inv}_C(s,k); \mathsf{wt}; k \cdot o@s \vdash e' : E \setminus \omega_2 | -$

Only typing rule and its inversion.

Let $\Sigma' = \Sigma, \quad \Delta^{\mathsf{wt}'} = \Delta_2, [o/\mathit{this}]\mathsf{inv}_C(s,k), \quad u^{\mathsf{wt}'} = k \cdot o@s, \quad \omega' = \omega_2$.

$\cdot; \Sigma'; \Delta'; \mathcal{E}; u' \vdash e' : E \setminus \omega' | -$ Substitution

Must show $\Sigma; \overline{\Delta^{\mathcal{E}}}; \overline{u^{\mathcal{E}}} \vdash H; \overline{S_p}$

$\Sigma'; (\Delta^{\mathsf{wt}'}, \Delta^*); \vdash S_p'$

We have removed $k \cdot o$ from Δ and S_p, and added field perms to Δ which are ignored.

$\Sigma'; \overline{\Delta'} \vdash \overline{S_p'}$

No other Δ or S_p changed.

Must show $\Sigma'; \overline{\Delta^{\mathcal{E}'}; S_p'; u^{\mathcal{E}'}} \vdash H'$

1.) ok No change to $\mathsf{dom}(\Sigma)$ or $\mathsf{dom}(H)$

2.) ok Still the only wt, no need to prove not-active.

3.) ok $\mathcal{E} = \mathsf{wt}$.

4.a.) ok No change

4.b.) ok Before step, either $S = \mathsf{unpacked}(s)$ and invariant holds by this rule, or $S = s$ and invariant held by this rule. Fields have not changed.

4.c.) ok No new stack perms in Δ'.

4.d.) ok 4.b. was true before step. Fields added to Δ' are given by $\mathsf{inv}_C(s, k)$.

4.e.) ok 4.g. was true before step. Any unique or full fields cannot be in other Δs and u.

4.f.) ok 4.g. was true before step. Other permissions to fields must agree with state.

4.g.) ok No fields altered.

o was unpacked. $u = -$ and $u' = k \cdot o@s$.

$\omega' - \omega = \omega'$ only contains fields of o. Packing Flag lemma

Δ' does not contain any fields in $\omega - \omega'$ $\omega - \omega' = \emptyset$

$\forall o \in \mathsf{dom}(H) : w(o, S_p) - w(o, \Delta) - w(o, u) \leq w(o, S_p') - w(o, \Delta') - w(o, u')$

Net is unchanged. Permission moved from Δ to u'.

$T = \cdot$

$\mathsf{not\text{-}active}(\texttt{unpack})$ implies $\mathsf{not\text{-}active}(e_2)$ Inversion of not-active.

$\mathsf{active}(\texttt{unpack})$ cannot be derrived.

$a = a'$ and $\mathsf{not\text{-}active}(e)$ implies $\mathsf{not\text{-}active}(e')$ Above

CASE E-PACK-R

So $e = \texttt{pack}\ o\ \texttt{to}\ s\ \texttt{in}\ e_2$, $e' = e_2$, $a = a'$, $H' = H[o \mapsto C(\ldots)@s]$, $S_p' = S_p[(k + k_o) \cdot o]$ and $T = \cdot$.

$\cdot; \Sigma; \Delta; \mathcal{E}; u \vdash e : E \setminus \omega | u''$ Assumption

$\Sigma; \overline{\Delta^{\mathcal{E}}}; \overline{u^{\mathcal{E}}} \vdash H; \overline{S_p}$ Assumption

$\mathsf{inv}_C(s)$ satisfied by o's fields

$k_o \cdot o \in S_p \quad o \mapsto C(\overline{f = k \cdot o})@\mathsf{unpacked}(s) \in H$

Inversion of only evaluation rule

$o \in \Sigma \quad \Delta^{\mathcal{E}} = (\Delta_1, \Delta_2)$

$\Sigma; \Delta_1 \vdash_{\mathcal{E}} [o/\textit{this}]\mathsf{inv}_C(s, k)$

$\Sigma; k \cdot o@s \vdash_{\mathcal{E}} k \cdot o@\$$

$\cdot; \Sigma; \Delta_2, k \cdot o@\$; \mathcal{E}; - \vdash e_2 : E \setminus \{l.\overline{f}\} | -$

No temporary permissions for $o.\overline{f}$ in Δ_2

Inversion of only typing rule

Let $\Sigma' = \Sigma$, $\quad D^{\mathcal{E}'} = \Delta_2, k \cdot o@\$ \quad u^{\mathcal{E}'} = - \quad \omega' = \emptyset$

$\cdot; \Delta'; \mathcal{E}; u' \vdash e' : E \setminus \omega' | -$ Substitution

Must show $\Sigma; \overline{\Delta^{\mathcal{E}}}; \overline{u^{\mathcal{E}}} \vdash H; \overline{S_p}$

$\Sigma'; (\Delta^{\mathcal{E}'}, \Delta^*); \vdash S_p'$

k added back to Δ, S_p $\Sigma'; \overline{\Delta'} \vdash \overline{S_p'}$ No other Δ or S_p changed.

1.) ok No change to $\mathsf{dom}(\Sigma)$ or $\mathsf{dom}(H)$

2.) ok We have not added a wt that was not previously there.
If $\mathcal{E} \neq$ wt, not-wt$(\Delta^{\mathcal{E}'})$ by inversion of $\Sigma; \Delta' \vdash_{\mathcal{E}} k \cdot o@\$$.

3.) ok

We have only removed a permission from $u^{\mathcal{E}}$. This o is packed and $\mathsf{inv}_C(o, \mathsf{unique})$ from above.

4.a.) ok No change

4.b.) ok

For only modified o, $S = s$ and invariant satisfied from assumption and 4.d being true before step.

4.c.) ok Only one new permission added to Δ, and $S = s$.

4.d.) ok We have only removed fields from Δ.

4.e.) ok True before step. Can be no other full or uniques to u, now in Δ'.

4.f.) ok True before step. u, now in Δ, must be consistent.

4.g.) ok From 4.e. and 4.f before step.

$\omega' = \emptyset \subset \omega$

$\forall o \in \mathsf{dom}(H) : w(o, S_p) - w(o, \Delta) - w(o, u) \leq w(o, S_p') - w(o, \Delta') - w(o, u')$

Net is unchanged. Permission moved from u to Δ'.

$T = \cdot$

not-active(`pack`) implies not-active(e_2) Inversion of not-active.

active(`pack`) cannot be derrived.

$a = a'$ and not-active(e) implies not-active(e') Above

CASE E-PACK-RW

So $e = \texttt{pack}\ o\ \texttt{to}\ s\ \texttt{in}\ e_2$, $e' = e_2$, $a = a'$, $H' = H[o \mapsto C(\ldots)@s]$, $S_p' = S_p[(k + k_o) \cdot o]$ and $T = \cdot$.

$\cdot; \Sigma; \Delta; \mathcal{E}; u \vdash e : E \setminus \omega | u''$ Assumption

$\Sigma; \overline{\Delta^{\mathcal{E}}}; \overline{u^{\mathcal{E}}} \vdash H; \overline{S_p}$ Assumption

$o \mapsto C(\ldots)@\mathsf{unpacked}(k) \in H \quad k_o \cdot o \in S_p \quad \mathsf{inv}_C(s)$ satisfied by o's fields

Inversion of only eval rule

$o \in \Sigma \quad \Delta^{\mathcal{E}} = (\Delta_1, \Delta_2)$

$\Sigma; \Delta_1 \vdash_{\mathcal{E}} [o/\mathit{this}]\mathsf{inv}_C(s, k)$

$\Sigma; k \cdot o@s \vdash_{\mathcal{E}} k \cdot o@\$$

$\cdot; \Sigma; \Delta_2, k \cdot o@\$; \mathcal{E}; - \vdash e_2 : E \setminus \{l.\overline{f}\} | -$

No temporary permissions for $o.\overline{f}$ in Δ_2

Inversion of only typing rule

Let $\Sigma' = \Sigma, \quad D^{\mathcal{E}'} = \Delta_2, k \cdot o@\$ \quad u^{\mathcal{E}'} = - \quad \omega' = \emptyset$

$\cdot; \Delta'; \mathcal{E}; u' \vdash e' : E \setminus \omega' | -$ Substitution

Must show $\Sigma; \overline{\Delta^{\mathcal{E}}}; \overline{u^{\mathcal{E}}} \vdash H; \overline{S_p}$

$\Sigma'; (\Delta^{\mathcal{E}'}, \Delta^*); \vdash S_p'$

k added back to Δ, S_p $\Sigma'; \overline{\Delta'} \vdash \overline{S_p'}$ No other Δ or S_p changed.

Must show $\Sigma'; \overline{\Delta^{\mathcal{E}'}; S_p'; u^{\mathcal{E}'}} \vdash H'$

1.) ok No change to $\mathsf{dom}(\Sigma)$ or $\mathsf{dom}(H)$

2.) ok We have not added a wt that was not previously there.

If $\mathcal{E} \neq$ wt, not-wt$(\Delta^{\mathcal{E}'})$ by inversion of $\Sigma; \Delta' \vdash_{\mathcal{E}} k \cdot o@\$$.

3.) ok

We have only removed a permission from $u^{\mathcal{E}}$. This o is packed and $\mathsf{inv}_C(o, \mathsf{unique})$ from above.

4.a.) ok — No change

4.b.) ok

For only modified o, $S = s$ and invariant satisfied from assumption and 4.d being true before step.

4.c.) ok — Only one new permission added to Δ, and $S = s$.

4.d.) ok — We have only removed fields from Δ.

4.e.) ok — True before step. Can be no other full or uniques to u, now in Δ'.

4.f.) ok — True before step. u, now in Δ, must be consistent.

4.g.) ok — From 4.e. and 4.f before step.

$\omega' = \emptyset \subset \omega$

$\forall o \in \mathsf{dom}(H) : w(o, S_p) - w(o, \Delta) - w(o, u) \leq w(o, S'_p) - w(o, \Delta') - w(o, u')$

Net is unchanged. Permission moved from u to Δ'.

$T = \cdot$

not-active$(\mathtt{pack})$ implies not-active(e_2) — Inversion of not-active.

active$(\mathtt{pack})$ cannot be derrived.

$a = a'$ and not-active(e) implies not-active(e') — Above

CASE E-ASSIGN

So $e = o_1.f := k{\cdot}o_2$, $e' = k'{\cdot}o'$, $a = a'$, $H' = H[o_1 \mapsto C(\ldots, f = k{\cdot}o_2, \ldots)@\mathsf{unpacked}(k'')]$, $S'_p = S_p[(k_2 - k) \cdot o_2], k' \cdot o'$ and $T = \cdot$.

$\cdot; \Sigma; \Delta; \mathcal{E}; u \vdash e : E \setminus \omega | u''$ — Assumption

$\Sigma; \overline{\Delta^{\mathcal{E}}}; \overline{u^{\mathcal{E}}} \vdash H; \overline{S_p}$ — Assumption

$k_1 \cdot o_1 \in S_p$

$o_1 \mapsto C(\ldots, f = k' \cdot o', \ldots)@\mathsf{unpacked}(k'') \in H$

$k_2 \cdot o_2 \in S_p \quad o_2 \mapsto C(\ldots)@S_2 \in H$

Inversion of only eval rule

$\Delta^{\mathcal{E}} = (\Delta_1, \Delta_2) \quad \omega = \{o_i.f\}$

$\mathsf{localFields}(C'') = \overline{f : C} \quad (o' : C') \in \Sigma \quad \mathsf{writes}(k')$

$\cdot; \Sigma; \Delta_1 \vdash_{\mathcal{E}} k \cdot o : \exists x : C_i.P$

$\cdot; \Sigma; \Delta_2 \vdash_{\mathcal{E}} [o'.f_i/x']P'$

Inversion of only typing rule

Let $\Sigma' = \Sigma, \quad \Delta^{\mathcal{E}'} = [o'/x']P' \otimes [o_i.f/x]P \quad u^{\mathcal{E}'} = u^{\mathcal{E}} = k' \cdot o'@s' \quad \omega' = \emptyset$

$\cdot; \Sigma'; \Delta'; \mathcal{E}; u' \vdash e' : E \setminus \omega' | u''$ — By rule T-LOC. Must show $\Sigma; \overline{\Delta^{\mathcal{E}}}; \overline{u^{\mathcal{E}}} \vdash H; \overline{S_p}$

$\Delta' = [o_i.f/o_2]([o'/o_i.f]\Delta))$ — From above.

$\Sigma'; (\Delta^{\mathsf{wt}'}, \Delta^*); \vdash S'_p$

$k' \cdot o'$ went into both Δ', as subst. for field permission and S'_p.
Field permissions inserted, which are ignored.

$\Sigma'; \overline{\Delta'} \vdash \overline{S'_p}$

No other Δ or S_p changed.

Must show $\Sigma'; \overline{\Delta^{\mathcal{E}'}; S'_p; u^{\mathcal{E}'}} \vdash H'$

1.) ok — No change to $\mathsf{dom}(\Sigma)$ or $\mathsf{dom}(H)$

2.) ok — We have not changed the number of wt elements from before. The permissions added to Δ' were cleansed, by the inverse of transaction-aware linear judgment.

3.) ok — $u^{\mathcal{E}}$ unchanged.

4.a.) ok — No change

4.b.) ok — $S = \mathsf{unpacked}(k)$

4.c.) ok — From 4.d. true before step.

4.d.) ok — From 4.c. true before step.

4.e.) ok — From 4.g. true before step.

4.f.) ok — From 4.g. true before step.

4.g.) ok — From 4.d,e,f. true before step.

$\omega' = \supset \omega = \{o_1.f\}$ $\forall o \in \mathsf{dom}(H) : w(o, S_p) - w(o, \Delta) - w(o, u) \leq w(o, S'_p) - w(o, \Delta') - w(o, u')$

$k \cdot o_2$ and $k' \cdot o'$ move between field and stack.

$T = \cdot$

$a' = a$ and only $\mathsf{not\text{-}active}(e)$ can be derrived.

$\mathsf{not\text{-}active}$ rules for field.

$\mathsf{not\text{-}active}(e')$ — $\mathsf{not\text{-}active}$ rules for loc.

CASE E-CALL

So $e = k \cdot o.m(\overline{k \cdot o})$, $e' = [o/\mathit{this}][\overline{o}/\overline{f}]e_m$, $H' = H'$, $S_p = S_p$, $a' = a$,

$\cdot; \Sigma; \Delta; \mathcal{E}; u \vdash e : E \setminus \omega | u''$ — Assumption

$\Sigma; \overline{\Delta^{\mathcal{E}}}; \overline{u^{\mathcal{E}}} \vdash H; \overline{S_p}$ — Assumption

$\mathsf{mbody}(C, m) = \overline{x}.e_m$ $\mathsf{mtype}(C, m) = \forall \overline{x : C}.P \multimap E$

$H; S_p | k \cdot o, \overline{k \cdot o} \vdash [o/\mathit{this}][\overline{o}/\overline{x}]P$

Inversion only eval rule

$\cdot; \Sigma; \Delta; \mathcal{E}; - \vdash k \cdot o.m(\overline{k \cdot o}) : \exists x : C.P'_r \setminus \emptyset | -$

$(o : C) \in \Sigma$ $\overline{o : C} \subseteq \Sigma$

$\cdot; \Sigma; \Delta \vdash_{\mathcal{E}} [o/\mathit{this}][\overline{o}/\overline{f}]P$ $\mathsf{mtype}(C, m) = \forall \overline{x : C}.P \multimap \exists x : C.P_r$

$\mathtt{forget}_{\mathcal{E}}(P_r) = P'_r$

Only typing rule and its inversion.

$\overline{x : C}, \mathit{this} : C; \cdot; P; \mathsf{wt}; - \vdash e_m : \exists x : C.P_r \setminus \emptyset | -$

$\overline{x : C}, \mathit{this} : C; \cdot; P; \mathsf{ot}; - \vdash e_m : \exists x : C.P''_r \setminus \emptyset | -$

Inversion of M ok

$\mathcal{E} = \mathsf{wt}$ implies $P_r = P'_r$

$\mathcal{E} \neq \mathsf{wt}$ implies $P''_r = P'_r$

Inversion of forget

$\cdot; \Sigma'; \Delta'; \mathcal{E}; u' \vdash [o/\mathit{this}][\overline{o}/\overline{f}]e_m : E \setminus \omega' | u''$

Above and substituion.

$\Sigma'; \overline{\Delta^{\mathcal{E}'}; S'_p; u^{\mathcal{E}'}} \vdash H'$

No changes

$\forall o \in \mathsf{dom}(H) : w(o, S_p) - w(o, \Delta) - w(o, u) \leq w(o, S_p') - w(o, \Delta') - w(o, u')$

No changes

$a' = a$ and $\mathsf{not\text{-}active}(e)$ Only notactive can be derrived for call.

$\mathsf{not\text{-}active}(e')$ Well formed method body cannot be active.

CASE E-SPAWN

So $e = \texttt{spawn}\ (k \cdot o.m(\overline{k \cdot o}))$, $H' = H'$, $\overline{S_p}' = \overline{S_p}$,$S_{p2}$ with $S_{p1}, \mathsf{immutable} \cdot o_d @ S_d$ replacing S_p.

$\cdot; \Sigma; \Delta; \mathcal{E}; u \vdash e : E \setminus \omega | u''$ Assumption

$\Sigma; \overline{\Delta^{\mathcal{E}}}; \overline{u^{\mathcal{E}}} \vdash H; \overline{S_p}$ Assumption

$\mathsf{mbody}(C, m) = \overline{x}.e_m \quad \mathsf{mtype}(C, m) = \forall \overline{x : C}.P \multimap E$

$H; S_{p_2} | k \cdot o, \overline{k \cdot o} \vdash [o/\mathit{this}][\overline{o}/\overline{x}]P$

Inversion of only eval rule

$o : C \in \Sigma \quad \overline{o : C} \in \Sigma \quad \mathsf{mtype}(C, m) = \forall \overline{x : C}.P \multimap E$

$\cdot; \Sigma; \Delta^{\mathsf{ot}} \vdash_{\mathsf{ot}} [o/\mathit{this}][\overline{o}/\overline{f}]P$

Inversion of only typing rule

Let $e' = \mathsf{immutable} \cdot o_d$, $T = \langle [o/\mathit{this}][\overline{o}/\overline{f}]e_m, S_{p2} \rangle$, $\Sigma' = \Sigma$, $\Delta^{\mathsf{ot}'} = \mathsf{immutable} \cdot o_d @ S_d$,

$\Delta_t^{\mathsf{ot}} = \Delta$, $u_t^{\mathsf{ot}} = -$, $u^{\mathsf{ot}'} = -$

$\cdot; \Sigma'; \Delta_t; \mathsf{ot}; - \vdash e_t : E_t \setminus \emptyset | -$

$\overline{x : C}, \mathit{this} : C; \cdot; P; \mathsf{ot}; - \vdash e_m : E_t \setminus \emptyset | -$ Inversion of mtype.

$\cdot; \Sigma'; \Delta'; \mathsf{ot}; - \vdash \mathsf{immutable} \cdot o_d : \exists_- : C_d.\mathsf{immutable} \cdot o_d @ S_d$

Always true of o_d, which is implicitly in all Δ.

Must show $\Sigma; \overline{\Delta^{\mathcal{E}}}; \overline{u^{\mathcal{E}}} \vdash H; \overline{S_p}$

$\Sigma'; (\Delta^{\mathsf{ot}'}, \Delta^*); \vdash S_p'$ Only one permission in Δ' and we added it to S_p'.

$\Sigma'; \Delta_t^{\mathsf{ot}}; \vdash S_{p2}$ From above

$\Sigma'; \overline{\Delta'} \vdash \overline{S_p'}$ No other Δ or S_p changed.

Must show $\Sigma'; \overline{\Delta^{\mathcal{E}'}; S_p'; u^{\mathcal{E}'}} \vdash H'$

1.) ok No change to $\mathsf{dom}(\Sigma)$ or $\mathsf{dom}(H)$

2.) okNew Δs are tagged with ot. Δ' on has $\mathsf{immutable}$ objects and Δ_t is clean, inverse of TALJ.

3.) ok New us are both $-$.

4.a.) ok No change

4.b.) ok No states or fields changed.

4.c.) ok Nothing new in Δs w.r.t. the heap.

4.d.) ok Nothing new in Δs w.r.t. the heap.

4.e.) ok Nothing new in Δs w.r.t. the heap or u.

4.f.) ok Special default object, o_d, is always in state s_d.

4.g.) ok No fields modified.

$\omega' \subseteq \omega$ $\omega' = \omega = \emptyset$

$\omega_t \subseteq \omega$ $\omega_t = \omega = \emptyset$

Δ' contains no permissions for fields in $\emptyset$

Δ_t contains no permissions for fields in $\emptyset$

$\forall o \in \mathsf{dom}(H) : w(o, S_p) - w(o, \Delta) - w(o, u) \leq w(o, S'_p \cup S_{tp}) - w(o, \Delta', \Delta_t) - w(o, u', u_t)$
Net is unchanged. $\mathsf{immutable} \cdot o_d@s_d$ added to S'_p and Δ'
$a' = a$ and $\mathsf{not\text{-}active}(e)$ — $\mathsf{not\text{-}active}$ rule for Spawn.
$\mathsf{not\text{-}active}(e')$ — $\mathsf{not\text{-}active}$ rule for o_d.
$\mathsf{not\text{-}active}(e_t)$ — Property of well-typed method body.

CASE E-READ-R

So $e = k \cdot o.f_i$, $e' = k \cdot o'$, $T = \cdot$, $a' = a$, $H' = H[o \mapsto C(\ldots, f_i = (k' - k) \cdot o, \ldots)@\mathsf{unpacked}(s'')]$, $S'_p = S_p, (k \cdot o')$.

$\cdot; \Sigma; \Delta; \mathcal{E}; u \vdash e : E \setminus \omega | u''$ — Assumption
$\Sigma; \overline{\Delta^{\mathcal{E}}}; \overline{u^{\mathcal{E}}} \vdash H; \overline{S_p}$ — Assumption
$k = \mathsf{pure} \mid \mathsf{immutable} \quad o \mapsto C(\ldots, f_i = k' \cdot o')@\mathsf{unpacked}(s'') \in H$
Inversion of only eval rule
$\cdot; \Sigma; \Delta; \mathcal{E}; k_u \cdot o@S_u \vdash k \cdot o.f_i : \exists x : T_i.[x/f_i]P \setminus \emptyset | k_u \cdot o@s_u$
$\mathsf{readonly}(k_u)$ *implies* $\mathsf{readonly}(k) \quad \cdot; \Sigma; \Delta \vdash_{\mathcal{E}} P \quad \mathsf{localFields}(C) = \overline{f : C}$
Only typing rule and its inversion
Let $\Sigma' = \Sigma$, $u^{\mathcal{E}'} = u^{\mathcal{E}}$, $\Delta^{\mathcal{E}'} = [o'/o.f_i]P$, $\omega' = \omega = \emptyset$.
$\cdot; \Sigma'; \Delta'; \mathcal{E}'; u' \vdash k \cdot o' : E \setminus \omega' | u'$
Rule T-LOC.
Must show $\Sigma; \overline{\Delta^{\mathcal{E}}}; \overline{u^{\mathcal{E}}} \vdash H; \overline{S_p}$
$\Sigma'; (\Delta^{\mathcal{E}'}, \Delta^*); \vdash S'_p$ — Δ' only has permissions for o', this object was added to S_p.
$\Sigma'; \overline{\Delta'} \vdash \overline{S'_p}\ \Sigma'; \overline{\Delta'} \vdash \overline{S'_p}$ — No other Δ or S_p changed.
Must show $\Sigma'; \overline{\Delta^{\mathcal{E}'}; S'_p; u^{\mathcal{E}'}} \vdash H'$
1.) ok — No change to $\mathsf{dom}(\Sigma)$ or $\mathsf{dom}(H)$
2.) ok — We have not added a wt that was not previously there.
If $\mathcal{E} \neq \mathsf{wt}$, $\mathsf{not\text{-}wt}(\Delta^{\mathcal{E}'})$ by inversion of $\Sigma; \Delta' \vdash_{\mathcal{E}} k \cdot o@\$$.
3.) ok — u' has not changed.
4.a.) ok — No change
4.b.) ok — By inversion of $-$ on permissions and $\mathsf{inv}_C(s, \mathsf{immutable})$
4.c.) ok — States are correct by $\mathsf{inv}_C(s, \mathsf{immutable})$ of o's fields.
4.d.) ok — We have only removed field permissions from Δ.
4.e.) ok — There can be no full or unique perm in P after downgrading.
4.f.) ok — From 4.g. true before step.
4.g.) ok — True by inversion of subtraction on permissions.
$\omega' \subseteq \omega$ — $\omega' = \omega = \emptyset$
Δ' contains no permissions for fields in $\emptyset$
$\forall o \in \mathsf{dom}(H) : w(o, S_p) - w(o, \Delta) - w(o, u) \leq w(o, S'_p) - w(o, \Delta') - w(o, u')$
Net unchanged. $k \cdot o'$ added to S_p and Δ.
$T = \cdot$
$a' = a$ and only $\mathsf{not\text{-}active}(e)$ can be derrived. — $\mathsf{not\text{-}active}$ rule for field reads.
Only $\mathsf{not\text{-}active}(e')$ can be derrived. — $\mathsf{not\text{-}active}$ rule for location reads.

CASE E-READ-RW

So $e = k \cdot o.f_i$, $e' = k \cdot o'$, $T = \cdot$, $a' = a$, $H' = H[o \mapsto C(\ldots, f_i = (k' - k) \cdot o, \ldots)@\textsf{unpacked}(k'')]$, $S_p' = S_p, (k \cdot o')$.

$\cdot; \Sigma; \Delta; \mathcal{E}; u \vdash e : E \setminus \omega | u''$ — Assumption

$\Sigma; \overline{\Delta^{\mathcal{E}}}; \overline{u^{\mathcal{E}}} \vdash H; \overline{S_p}$ — Assumption

$k \leq k$' $\quad o \mapsto C(\ldots, f_i = k' \cdot o')@\textsf{unpacked}(k'') \in H$

Inversion of only eval rule.

$\cdot; \Sigma; \Delta \vdash_{\mathcal{E}} P \quad \textsf{localFields}(C) = \overline{f : C}$

$\cdot; \Sigma; \Delta; \mathcal{E}; k_u \cdot o@S_u \vdash k \cdot o.f_i : \exists x : T_i.[x/f_i]P \setminus \emptyset | k_u \cdot o@s_u$

Inversion of only typing rule.

Let $\Sigma' = \Sigma$, $u^{\mathcal{E}'} = u^{\mathcal{E}}$, $\Delta^{\mathcal{E}'} = [o'/o.f_i]P, \omega' = \omega = \emptyset$.

$\cdot; \Sigma'; \Delta'; \mathcal{E}'; u' \vdash k \cdot o' : E \setminus \omega' | u'$

Rule T-LOC.

Must show $\Sigma; \overline{\Delta^{\mathcal{E}}}; \overline{u^{\mathcal{E}}} \vdash H; \overline{S_p}$

$\Sigma'; (\Delta^{\mathcal{E}'}, \Delta^*); \vdash S_p'$ — Δ' only has permissions for o', this object was added to S_p.

$\Sigma'; \overline{\Delta'} \vdash \overline{S_p'}$ — No other Δ or S_p changed.

Must show $\Sigma'; \overline{\Delta^{\mathcal{E}'}; S_p'; u^{\mathcal{E}'}} \vdash H'$

1.) ok — No change to $\textsf{dom}(\Sigma)$ or $\textsf{dom}(H)$

2.) ok — We have not added a wt that was not previously there.

If $\mathcal{E} \neq \textsf{wt}$, $\textsf{not-wt}(\Delta^{\mathcal{E}'})$ by inversion of $\Sigma; \Delta' \vdash_{\mathcal{E}} k \cdot o@\$$.

3.) ok — u' has not changed.

4.a.) ok — No change

4.b.) ok — $S = \textsf{unpacked}(k)$

4.c.) ok — 4.d.) was true before step.

4.d.) ok — We have only removed field permissions from Δ.

4.e.) ok — 4.g.) was true before step.

4.f.) ok — 4.g.) was true before step.

4.g.) ok — True by inversion of subtraction on permissions.

$\omega' \subseteq \omega$ — $\omega' = \omega = \emptyset$

Δ' contains no permissions for fields in $\emptyset$

$\forall o \in \textsf{dom}(H) : w(o, S_p) - w(o, \Delta) - w(o, u) \leq w(o, S_p') - w(o, \Delta') - w(o, u')$

Net unchanged. $k \cdot o'$ added to S_p and Δ.

$T = \cdot$

$a' = a$ and only $\textsf{not-active}(e)$ can be derrived. — not-active rule for field reads.

Only $\textsf{not-active}(e')$ can be derrived. — not-active rule for location reads.

CASE E-INATOMIC

So $e = \texttt{inatomic}\ (e_1)$, $e' = \texttt{inatomic}\ (e_1')$, $a' = a$, $H' = H'$ from I.H., $S_p' = S_p'$ from I.H., $\omega' = \omega'$ from I.H.

$\cdot; \Sigma; \Delta; \mathcal{E}; u \vdash e : E \setminus \omega | u''$ Assumption

$\Sigma; \overline{\Delta^{\mathcal{E}}}; \overline{u^{\mathcal{E}}} \vdash H; \overline{S_p}$ Assumption

$a; H; \langle e_1, S_p \rangle \to a'; H'; \langle e_1', S_p' \rangle; T$

Inversion of only eval rule

$\cdot; \Sigma; \Delta; \mathcal{E}; u \vdash \texttt{inatomic}\ (e_1) : \exists x : C.P' \setminus \omega | u'$

$\cdot; \Sigma; \Delta; \mathsf{wt}; u \vdash e_1 : \exists x : C.P \setminus \omega | u' \quad \texttt{forget}_{\mathcal{E}}(P) = P'$

Inversion of only typing rule

Apply induction hypothesis.

$\Sigma; \overline{\Delta^{\mathcal{E}}}; \overline{u^{\mathcal{E}}} \vdash H; \overline{S_p}$ I.H.

T ok I.H.

ω' ok I.H.

$a' = a$ and active$(\texttt{inatomic}\ (e_1))$ active rule for `inatomic`.

active(e') active rule for `inatomic`.

CASE E-ATOMIC-EXIT

So $e = \texttt{inatomic}\ (k \cdot o)$, $e' = k \cdot o$, $S_p' = S_p$, $H' = H$, $a' = \circ$.

$\cdot; \Sigma; \Delta; \mathcal{E}; u \vdash e : E \setminus \omega | u''$ Assumption

$\Sigma; \overline{\Delta^{\mathcal{E}}}; \overline{u^{\mathcal{E}}} \vdash H; \overline{S_p}$ Assumption

$\cdot; \Sigma; \Delta; \mathcal{E}; u \vdash \texttt{inatomic}\ (e) : \exists x : C.P' \setminus \omega | u''$

$\cdot; \Sigma; \Delta; \mathsf{wt}; u \vdash e : \exists x : C.P \setminus \omega | u'' \quad \texttt{forget}_{\mathcal{E}}(P) = P'$

Only typing rule and its inverse. Let $\Sigma' = \Sigma$, $\omega' = \omega$

Case: $\mathcal{E} = \mathsf{wt}$

Let $\Delta^{\mathsf{wt}'} = \Delta, u^{\mathsf{wt}}$.

$\cdot; \Sigma'; \Delta'; \mathsf{wt}; u' \vdash k \cdot o : \exists X : C.P \setminus \omega | u''$

By sustitution, and P'=P when $\mathcal{E} = \mathsf{wt}$

Tag for u and Δ did not change.

$\langle \overline{\Delta^{\mathcal{E}}, u^{\mathcal{E}}} \rangle$ ok Above

Case: $\mathcal{E} \neq \mathsf{wt}$

Let $\Delta^{\mathcal{E}} = P'$, $u^{\mathcal{E}} = u$.

$\cdot; \Sigma'; \Delta'; \mathcal{E}; u' \vdash k \cdot o : \exists X : C.P' \setminus \omega | u''$

Rules T-LOC

$\Delta^{\mathcal{E}}$ contains no share or pure perms. inv. forget.

$u^{\mathcal{E}}$ contains no share or pure permissions.

Unpacking share or pure requires $\mathcal{E} = \mathsf{wt}$

$\langle \overline{\Delta^{\mathcal{E}}, u^{\mathcal{E}}} \rangle$ ok Above

Heap cond 3 satisfied. Above

$\Sigma; \overline{\Delta^{\mathcal{E}}}; \overline{u^{\mathcal{E}}} \vdash H; \overline{S_p}$

Rest of heap unchanged. $T = \cdot$ $a' = \circ \neq \bullet = a$, active$(e)$ Active rule for `inatomic`

not-active(e') Only derivable rule for $k \cdot o$

CASE E-ATOMIC

So $e = \texttt{atomic}\ (e_1)$, $e' = \texttt{inatomic}\ (e_1)$, $H' = H$, $S'_p = S_p$, $a' = \bullet$, $\omega' = \omega$.

$\cdot; \Sigma; \Delta; \mathcal{E}; u \vdash e : E \setminus \omega | u''$ — Assumption

$\Sigma; \overline{\Delta^{\mathcal{E}}}; \overline{u^{\mathcal{E}}} \vdash H; \overline{S_p}$ — Assumption

$\cdot; \Sigma; \Delta; \mathcal{E}; u \vdash \texttt{atomic}\ (e_1) : \exists x : C.P' \setminus \omega | u''$

$\cdot; \Sigma; \Delta; \mathsf{wt}; u \vdash e_1 : \exists x : C.P \setminus \omega | u'' \quad \texttt{forget}_{\mathcal{E}}(P) = P'$

only typing rule and its inversion. Let $\Sigma' = \Sigma$, $u' = u$, $\Delta' = \Delta$, $\omega' = \omega$.

$\cdot; \Sigma'; \Delta'; \mathcal{E} \vdash e' : \exists x : C.P \setminus \omega' | u''$

By rule T-INATOMIC. Let u' and Δ' be tagged with wt.

$\langle \overline{\Delta^{\mathcal{E}}, u^{\mathcal{E}}} \rangle\ \mathsf{ok}$

$a = \circ$ implies no u or Δ tagged with wt before step.

$\Sigma; \overline{\Delta^{\mathcal{E}}}; \overline{u^{\mathcal{E}}} \vdash H; \overline{S_p}$

No other changes to heap.

$a' = \circ \neq \bullet = a$. Given $\mathsf{not\text{-}active}(e)$.

$\mathsf{active}(e')$ — active rule for $\texttt{inatomic}$.

CASE E-NEW

So $e = \texttt{new}\ C(\overline{k \cdot o})$, $e' = \mathsf{unique} \cdot o_n$, $H' = H, o_n \mapsto C(\overline{f = k \cdot o})@s$, $S'_p = (S_p - \overline{k \cdot o}), \mathsf{unique} \cdot o_n$, $a' = a$.

$\cdot; \Sigma; \Delta; \mathcal{E}; u \vdash e : E \setminus \omega | u''$ — Assumption

$\Sigma; \overline{\Delta^{\mathcal{E}}}; \overline{u^{\mathcal{E}}} \vdash H; \overline{S_p}$ — Assumption

$H; S_p \vdash [\overline{o}/\overline{f}]P \quad \mathsf{init}(C) = \langle \exists \overline{f : C}.P, s \rangle$

Inversion of only evaluation rule

$\cdot; \Sigma; \Delta; \mathcal{E}; u \vdash \texttt{new}\ C(\overline{k \cdot o}) : \exists x : C.\mathsf{unique} \cdot x@s \setminus \emptyset | u$

$\cdot; \Sigma; \Delta \vdash_{\mathcal{E}} [\overline{o}/\overline{f}]P \quad \overline{o : C} \subseteq \Sigma \quad \mathsf{init}(C) = \langle \exists \overline{f : C}.P, s \rangle$

Only typing rule and its inversion.

Let $\Sigma' = \Sigma, o_n : C$, $u^{\mathcal{E}'} = u$, $\Delta^{\mathcal{E}'} = \mathsf{unique} \cdot o_n@s$, where $\mathcal{E}$ tag is the same as before step, $\omega' = \omega = \emptyset$.

$\cdot; \Sigma'; \Delta'; \mathcal{E}; u' \vdash \mathsf{unique} \cdot o_n : \exists x : C.\mathsf{unique} \cdot x@s$

By rule T-LOC

Must show $\Sigma; \overline{\Delta^{\mathcal{E}}}; \overline{u^{\mathcal{E}}} \vdash H; \overline{S_p}$

$\Sigma'; (\Delta^{\mathcal{E}'}, \Delta^*); \vdash S'_p$

We removed $\overline{k \cdot o}$ from both S_p and Δ and added $\mathsf{unique} \cdot o_n@s$ to both.

$\Sigma'; \overline{\Delta'} \vdash \overline{S'_p}$

No other Δ or S_p changed.

Must show $\Sigma'; \overline{\Delta^{\mathcal{E}'}; S'_p; u^{\mathcal{E}'}} \vdash H'$

1.) ok — Added o_n to both.

2.) ok We have not changed $\mathcal{E}$ tagging. Only new permission is unique, so invariant holds, if nec.

3.) ok — o_n is packed. inv_C holds b/c inverse of init and runtime proof of P.

4.a.) ok $o : C$ added to both.
4.b.) ok $S = s$ for o_n and invariant holds from above.
4.c.) ok We know $o_n@s$ in Δ' and H' b/c we added them.
4.d.) ok No fields added to Δ.
4.e.) ok True b/c $o_n \notin \mathsf{dom}(\Sigma)$ until now.
4.f.) ok None added.
4.g.) ok Fields were all in Δ before step, therefore by 4.e and 4.f property now holds for fields.
$\forall o \in \mathsf{dom}(H) : w(o, S_p) - w(o, \Delta) - w(o, u) \leq w(o, S'_p) - w(o, \Delta') - w(o, u')$
$\overline{k \cdot o}$ removed from S'_p and Δ'.
$T = \cdot$
$a' = a$ and only $\mathsf{not\text{-}active}(e)$ can be derrived. inv on $\mathsf{not\text{-}active}$ rule.
$\mathsf{not\text{-}active}(e')$ $\mathsf{not\text{-}active}$ rule for locations.

CASE E-LET-E

So $e = \texttt{let}\ x = e_1\ \texttt{in}\ e_2$, $e' = \texttt{let}\ x = e'_1\ \texttt{in}\ e_2$.

$\cdot; \Sigma; \Delta; \mathcal{E}; u \vdash e : E \setminus \omega | u''$ Assumption
$\Sigma; \overline{\Delta^{\mathcal{E}}}; \overline{u^{\mathcal{E}}} \vdash H; \overline{S_p}$ Assumption
$a; H; \langle e_1, S_p \rangle \rightarrow a'; H'; \langle e'_1, S'_p \rangle; T$ Inversion of only eval rule
$\Delta^{\mathcal{E}} = (\Delta_1, \Delta_2) \quad \cdot; \Sigma; \Delta_1; \mathcal{E}; u \vdash e_1 : \exists x : C.P \setminus \omega_1 | u_2$
$\Sigma; \Delta_2, P \vdash_{\mathcal{E}} P'$
$x : C; \Sigma; P'; \mathcal{E}; u_2 \vdash e_2 : E \setminus \omega_2 | u''$
Inversion of only typing rule
$\Sigma; \overline{\Delta^{\mathcal{E}}}; \overline{u^{\mathcal{E}}} \vdash H; \overline{S_p}$ where $\overline{\Delta^{\mathcal{E}}}$ has Δ_1 instead of Δ.
Compositionality
Apply induction hypothesis where (Δ_2, Δ^*) is the additional linear context.
$\Sigma'; \overline{\Delta}'; u' \vdash H'; \overline{S_p}'$
$\overline{\Delta}'$ is the same as $\overline{\Delta}$ except Δ is now $\Delta'_1, \Delta_2, \Delta^*$. I.H.
Gives us $\Sigma' \supseteq \Sigma$. $u^{\mathcal{E}'}$ and ω'_1
I.H.
Either (a) $u = -$ and $u' = k \cdot o@s$ and ω_1-ω'_1 only contains fields of o or (b) $\omega'_1 \supseteq \omega_1$.
I.H.
$\cdot; \Sigma'; \Delta_1; \mathcal{E}; u' \vdash e'_1 : \exists x : C.P \setminus \omega'_1 | u_2$
I.H.
$\forall o \in \mathsf{dom}(H) : w(o, S_p) - w(o, \Delta) - w(o, u) \leq w(o, S'_p) - w(o, \Delta') - w(o, u')$
I.H. Fractions in Δ_2 unchanged.
T ok
I.H.
SUBCASE: $u = -$ and $u' = k \cdot o@s$ and $\omega'_1 - \omega_1$ only contains fields of o.

Δ_2, Δ^* do not contain permissions for fields of o.
Definition of well-typed store.
Δ_2, Δ^* do not contain permissions for fields in ω'_1 $\omega'_1 - \omega_1$ contains only fields of o

$\cdot; \Sigma'; \Delta'; \mathcal{E}; u' \vdash e' : E \setminus \omega' | u''$

By rule T-LET

SUBCASE: $\omega_1' \supseteq \omega_1$

Δ_2, Δ^* do not contain permissions for fields in ω_1'

$\omega_1' \supseteq \omega_1$

$\cdot; \Sigma'; \Delta'; \mathcal{E}; u' \vdash e' : E \setminus \omega' | u''$

By rule T-LET

If $a = a'$ and $\mathsf{active}(e)$, then $\mathsf{active}(e_1)$, $\mathsf{not\text{-}active}(e_2)$ — Inversion active

$\mathsf{active}(e_1)$ implies $\mathsf{active}(e_1')$ — Induction

$\mathsf{active}(e_1')$ and $\mathsf{not\text{-}active}(e_2)$ imply $\mathsf{active}(e')$ — Active rule, Let

If $a = a'$ and $\mathsf{not\text{-}active}(e)$ then $\mathsf{not\text{-}active}(e_1)$ and $\mathsf{not\text{-}active}(e_2)$ — Inversion $\mathsf{not\text{-}active}(e)$

$a = a'$ and $\mathsf{not\text{-}active}(e_1)$ implies $\mathsf{not\text{-}active}(e_1')$ — Induction

$\mathsf{not\text{-}active}(e_1')$ and $\mathsf{not\text{-}active}(e_2)$ imply $\mathsf{not\text{-}active}(e')$ — Not-active rule, Let

If $a = \circ$ and $a' = \bullet$, then $\mathsf{not\text{-}active}(e_1)$ and $\mathsf{active}(e_1')$ — Induction

$a = \circ$ implies $\mathsf{not\text{-}active}(e)$ — Assumption

$\mathsf{not\text{-}active}(e)$ imples $\mathsf{not\text{-}active}(e_2)$ — Inversion, not-active Let

$\mathsf{active}(e_1')$ and $\mathsf{not\text{-}active}(e_2)$ imply $\mathsf{active}(e')$ — Active rule, Let

If $a = \bullet$ and $a' = \circ$ then $\mathsf{active}(e_1)$ and $\mathsf{not\text{-}active}(e_1)$ — Induction

$a = \bullet$ implies either $\mathsf{active}(e)$ or $\mathsf{not\text{-}active}(e)$ — Assumption

Given $\mathsf{active}(e_1)$, $\mathsf{not\text{-}active}(e)$ impossible — Definition of active for Let

$\mathsf{active}(e)$ — Above

$\mathsf{active}(e)$ implies $\mathsf{not\text{-}active}(e_2)$ — Active rule, Let

$\mathsf{not\text{-}active}(e_1')$ and $\mathsf{not\text{-}active}(e_2$ implies $\mathsf{not\text{-}active}(e')$ — Not active rule, Let.

CASE E-LET-V

So $e = \texttt{let}\ x = k \cdot o\ \texttt{in}\ e_2$, $e' = [o/x]e_2$, $H' = H$, $S_p' = S_p$, $a' = a$.

$\cdot; \Sigma; \Delta; \mathcal{E}; u \vdash e : E \setminus \omega | u''$ — Assumption

$\Sigma; \overline{\Delta^{\mathcal{E}}}; \overline{u^{\mathcal{E}}} \vdash H; \overline{S_p}$ — Assumption

$k' \cdot o \in S_p \quad o \mapsto C(\ldots)@S \in H \quad k \leq k'$

Inversion of only eval rule.

$\cdot; \Sigma; (\Delta_1, \Delta_2); \mathcal{E}; u \vdash \texttt{let}\ x = e_1\ \texttt{in}\ e_2 : E \setminus \omega_1 \cup \omega_2 | u'$

$\Sigma; \Delta_2, P \vdash_{\mathcal{E}} P'$

$\cdot; \Sigma; \Delta_1; \mathcal{E}; u \vdash e_1 : \exists x : T.P \setminus \omega_1 | u_2 \quad x : C; \Sigma; P'; \mathcal{E}; u_2 \vdash e_2 : E\omega_2 | u'$

No permissions for eff_1 in Δ_2

Only typing rule and its inversion.

Let $\Sigma' = \Sigma$, $\Delta^{\mathcal{E}} = P'$, $u^{\mathcal{E}} = u$, $\omega' = \omega$. $\cdot; \Sigma'; \Delta'; \mathcal{E}; u' \vdash e_2 : E \setminus \omega_1 | u_2$

Substitution

Must show $\Sigma; \overline{\Delta^{\mathcal{E}}}; \overline{u^{\mathcal{E}}} \vdash H; \overline{S_p}$

No change at all except forgetting permissions in Δ.

$\forall o \in \mathsf{dom}(H) : w(o, S_p) - w(o, \Delta) - w(o, u) \leq w(o, S_p') - w(o, \Delta') - w(o, u')$

No changes

$T = \cdot$

$a' = a$.
$\textsf{not-active}(e)$ No active rule for locations, let rule.
$\textsf{not-active}(e_2)$ notactive Let rule.

1.6 Progress

1.6.1 Top-Level Progress

If $\vdash a; H; T$

Then there exists either:

- $\overline{v}$ such that $T =< \overline{v}, \overline{S_p} >$, or
- $a'; H'; T'$ such that $a; H; T \rightarrow a'; H'; T'$

Proof: By structural induction on the derivation of $\vdash a; H; T$

CASE T-TOP-LEVEL

$\vdash a; H; T$ Asssumed
$\textsf{correct-atomic}(a, T)$ Inversion of above
SUBCASE: $a = \circ$
Every e_i in $\langle ole, S_p \rangle$ is $\textsf{not-active}(e_i)$. Inversion of correct-atomic
SUBCASE: Every e in $\overline{e}$ is a value
Proof satisfied
SUBCASE: $\exists e_i$ in $\overline{e}$ s.t. e_i not a value
e_i must take a step Single-threaded lemma
Global thread pool steps rule E-THREAD-POOL
SUBCASE: $a = \bullet$
There is a e_i in $\overline{e}$ such that $\textsf{active}(e_i)$.
Inversion of correct-atomic
e_i must take a step Single-threaded lemma
Global thread pool steps rule E-THREAD-POOL

1.6.2 Thread-Level Progress

If

- $\Sigma; \overline{\Delta^{\mathcal{E}}}; \overline{u^{\mathcal{E}}} \vdash H; \overline{S_p}$

Then the following three items must hold true:

1. If $\cdot; \Sigma; \Delta; \mathcal{E}; u \vdash e : E \setminus \omega | u$ and $\textsf{active}(e)$, then $\exists e', a', H', T, S'_p$ such that $\bullet; H; \langle e, S_p \rangle \rightarrow a'; H'; \langle e', S'_p \rangle; T$, where Δ and S_p come from $\overline{\Delta}$ and $\overline{S_p}$ respectively and are associated.

2. If $\cdot;\Sigma;\Delta;\mathcal{E};u \vdash e : E \setminus \omega|u$, and $\textsf{not-active}(e)$, then e is a value, or $\exists e', a', H', T, S'_p$ such that $\circ; H; \langle e, S_p\rangle \rightarrow a'; H'; \langle e', S'_p\rangle; T$, where Δ and S_p come from $\overline{\Delta}$ and $\overline{S_p}$ respectively and are associated.

Proof: By structural induction on the derivation of $\Gamma;\Sigma;\Delta;\mathcal{E};u \vdash e : E \setminus \omega|u''$

CASE T-LOC $k \cdot o$ is already a value.
CASE T-CALL
So $e = k \cdot o.m(\overline{k \cdot o})$.

$\Sigma;\overline{\Delta^{\mathcal{E}}};\overline{u^{\mathcal{E}}} \vdash H;\overline{S_p}$ — Assumption
$\cdot;\Sigma;\Delta;\mathcal{E};- \vdash k \cdot o.m(\overline{k \cdot}) : \exists x : C.P'_r|-$ — Assumption
$(o : C) \in \Sigma \quad \overline{o : C} \subseteq \Sigma$
$\cdot;\Sigma;\Delta \vdash_{\mathcal{E}} [o/\textit{this}][\overline{o}/\overline{f}]P \quad \textsf{mtype}(C, m) = \forall \overline{x : C}.P \multimap \exists x : C.P_r$
$\texttt{forget}_{\mathcal{E}}(P_r) = P'_r$
Inversion of typing rule. $o, \overline{o} \in \textsf{dom}(H)$ — Heap condition 1
$o, \overline{o} \in \textsf{dom}(S_p)$ — $\Sigma;\Delta \vdash S_p$
$\{k' \cdot o, \overline{k \cdot o}\} \subseteq S_p$ — Above
$k \leq k', \overline{k \leq k'}$ — $\Sigma;\Delta \vdash S_p$
$H, S_p|k \cdot o, \overline{k \cdot o} \vdash [o/\textit{this}][\overline{o}/\overline{f}]P$
— $\Sigma;\Delta \vdash S_p$ and heap well-typed
$a; H; \langle e, S_p\rangle \rightarrow a'; H'; \langle e', S'_p\rangle; T'$
— By rule E-CALL
$\textsf{not-active}(e)$ — No rule for active Call.
Rule works for $a = \circ$

CASE T-SPAWN
So $e = \texttt{spawn}\ (k \cdot o.m(\overline{k \cdot o}))$.

$\Sigma;\overline{\Delta^{\mathcal{E}}};\overline{u^{\mathcal{E}}} \vdash H;\overline{S_p}$ — Assumption
$\cdot;\Sigma;\Delta;\textsf{ot};- \vdash \texttt{spawn}\ (k \cdot o.m(\overline{k \cdot o})) : \exists\_ : C_d.\textsf{immutable} \cdot o_d@s_d \setminus \emptyset|-$
— Assumption
$o : C \in \Sigma \quad \overline{o : C} \in \Sigma \quad \textsf{mtype}(C, m) = \forall \overline{x : C}.P \multimap E$
$\cdot;\Sigma;\Delta^{\textsf{ot}} \vdash_{\textsf{ot}} [o/\textit{this}][\overline{o}/\overline{f}]P$
— Inversion of only typing rule.
$o, \overline{o} \in \textsf{dom}(H)$ — Heap condition 1
$o, \overline{o} \in \textsf{dom}(S_p)$ — $\Sigma;\Delta \vdash S_p$
$\{k' \cdot o, \overline{k \cdot o}\} \subseteq S_p$ — Above
$k \leq k', \overline{k \leq k'}$ — $\Sigma;\Delta \vdash S_p$
$H, S_p|k \cdot o, \overline{k \cdot o} \vdash [o/\textit{this}][\overline{o}/\overline{f}]P$
— $\Sigma;\Delta \vdash S_p$ and heap well-typed

$a; H; \langle e, S_p \rangle \to a'; H'; \langle e', S_p' \rangle; T'$

By rule E-SPAWN

$\mathsf{not\text{-}active}(e)$ No rule for active Spawn.

Rule works for $a = \circ$

CASE T-UNPACK-WT

So $e = \texttt{unpack}_{\mathsf{wt}}\ k \cdot o@s\ \texttt{in}\ e_2$.

$\Sigma; \overline{\Delta^{\mathcal{E}}}; \overline{u^{\mathcal{E}}} \vdash H; \overline{S_p}$ Assumption

$\cdot; \Sigma; (\Delta, \Delta'); \mathsf{wt}; - \vdash \texttt{unpack}_{\mathsf{wt}}\ k \cdot o@s\ \texttt{in}\ e' : E \setminus \emptyset | -$ Assumption

$(o : C) \in \Sigma \quad \overline{o : C} \subseteq \Sigma$

$\cdot; \Sigma; \Delta \vdash_{\mathcal{E}} [o/\mathit{this}][\overline{o}/\overline{f}]P \quad \mathsf{mtype}(C, m) = \forall \overline{x : C}.P \multimap \exists x : C.P_r$

$\texttt{forget}_{\mathcal{E}}(P_r) = P_r'$

Inversion of only typing rule

$k' \cdot o \in S_p$ $\Sigma; \Delta \vdash S_p$

$o \in \mathsf{dom}(H)$ $\Sigma; \Delta \vdash S_p$

$k \leq k'$ From $\Sigma; \Delta \vdash S_p$

SUBCASE: $\mathsf{readonly}(k)$

$k = \mathsf{immutable}$ implies $o \mapsto C(\ldots)@s \in H$ or $o \mapsto C(\ldots)@\mathsf{unpacked}(s) \in H$

From heap condition 4.c and $\leq$. $k = \mathsf{pure}$ implies $o \mapsto C(\ldots)@s \in H$

From heap conditions 4.c, 2 and 3.

$a; H; \langle e, S_p \rangle \to a'; H'; \langle e', S_p' \rangle; T'$

By rule E-UNPACK-R-WT

Only $\mathsf{not\text{-}active}(e)$ can be derrived, and we can step when $a = \circ$.

SUBCASE: $\mathsf{writes}(k)$

$k = \mathsf{share}|\mathsf{full}|\mathsf{unique}$ implies $o \mapsto C(\ldots)@s \in H$

Heap condition 4.c.

$k' \in S_p \quad k \leq k'$ $\Sigma; \Delta \vdash S_p$

$a; H; \langle e, S_p \rangle \to a'; H'; \langle e', S_p' \rangle; T'$

By rule E-UNPACK-RW-WT

Only $\mathsf{not\text{-}active}(e)$ can be derrived, and we can step when $a = \circ$.

CASE T-UNPACK

So $e = \texttt{unpack}_{\mathcal{E}}\ k \cdot o@s\ \texttt{in}\ e_2$.

$\Sigma; \overline{\Delta^{\mathcal{E}}}; \overline{u^{\mathcal{E}}} \vdash H; \overline{S_p}$ Assumption

$\cdot; \Sigma; (\Delta, \Delta'); \mathcal{E}; - \vdash \texttt{unpack}_{\mathcal{E}}\ k \cdot o@s\ \texttt{in}\ e' : E \setminus \emptyset | -$

Assumption

$k = \mathsf{unique} \mid \mathsf{full} \mid \mathsf{immutable} \quad (o : C) \in \Sigma \quad \cdot; \Sigma; \Delta \vdash_{\mathcal{E}} k \cdot o@s$

$\mathcal{E} = \mathsf{emp}|\mathsf{ot} \quad \cdot; \Sigma; \Delta', [o/\mathit{this}]\mathsf{inv}_C(s, k); \mathcal{E}; k \cdot o@s \vdash e' : E \setminus \omega | -$

Inversion of only typing rule

$k' \cdot o \in S_p$ $\Sigma;\Delta \vdash S_p$
$o \in \mathsf{dom}(H)$ $\Sigma;\Delta \vdash S_p$
$k \leq k'$ From $\Sigma;\Delta \vdash S_p$

SUBCASE: $\mathsf{readonly}(k)$

$k = \mathsf{immutable}$ implies $o \mapsto C(\ldots)@s \in H$ or $o \mapsto C(\ldots)@\mathsf{unpacked}(s) \in H$

From heap condition 4.c and $\leq$.

$a; H; \langle e, S_p\rangle \to a'; H'; \langle e', S_p'\rangle; T'$

By rule E-UNPACK-R

Only $\mathsf{not\text{-}active}(e)$ can be derrived, and we can step when $a = \circ$.

SUBCASE: $\mathsf{writes}(k)$

$k = \mathsf{full}|\mathsf{unique}$ implies $o \mapsto C(\ldots)@s \in H$

Heap condition 4.c.

$k' \in S_p \quad k \leq k'$ $\Sigma;\Delta \vdash S_p$
$a; H; \langle e, S_p\rangle \to a'; H'; \langle e', S_p'\rangle; T'$

By rule E-UNPACK-RW

Only $\mathsf{not\text{-}active}(e)$ can be derrived, and we can step when $a = \circ$.

CASE T-PACK

So $e = \texttt{pack}\ o\ \texttt{to}\ s'\ \texttt{in}\ e_2$.

$\Sigma; \overline{\Delta^{\mathcal{E}}}; \overline{u^{\mathcal{E}}} \vdash H; \overline{S_p}$ Assumption
$\cdot; \Sigma; (\Delta, \Delta'); \mathcal{E}; k \cdot o@s \vdash \texttt{pack}\ o\ \texttt{to}\ s'\ \texttt{in}\ e' : E \setminus \{o\overline{f}\}|-$ Assumption
$\texttt{forget}_{\mathcal{E}}(k \cdot o@s) = k \cdot o@\$$
$k = \mathsf{immutable} \mid \mathsf{pure}\ \textit{implies}\ s = s' \quad \cdot; \Sigma; \Delta', k \cdot o@\$; \mathcal{E}; - \vdash e' : E \setminus \emptyset|-$
$\mathsf{localFields}(C) = \overline{f : C} \quad (o : C) \in \Sigma \quad \cdot; \Sigma; \Delta \vdash_{\mathcal{E}} [o/\textit{this}]\mathsf{inv}_C(s, k)$
No temporary permissions for $o.\overline{f}$ *in* Δ'

Inversion of only typing rule.

SUBCASE: $\mathsf{writes}(k)$

$o \mapsto C(\ldots)@\mathsf{unpacked}(k) \in H$ Heap condition 3
o's fields satisfy $[o/\textit{this}]\mathsf{inv}_C(s, k)$

Above and heap condition 4.d.

$k' \cdot o \in S_p$ $\Sigma;\Delta \vdash S_p$
$a; H; \langle e, S_p\rangle \to a'; H'; \langle e', S_p'\rangle; T'$

By rule E-PACK-RW

Only $\mathsf{not\text{-}active}(e)$ can be derrived.
We can step when $a = \circ$.

SUBCASE: $\mathsf{readonly}(k)$

$o \mapsto C(\ldots)@\mathsf{unpacked}(s) \in H$ Heap condition 3
o's fields satisfy $[o/\textit{this}]\mathsf{inv}_C(s, k)$

Above and heap condition 4.d.

$k' \cdot o \in S_p$ $\Sigma;\Delta \vdash S_p$
$a; H; \langle e, S_p\rangle \to a'; H'; \langle e', S_p'\rangle; T'$

By rule E-PACK-R

Only $\mathsf{not\text{-}active}(e)$ can be derrived.
We can step when $a = \circ$.

CASE T-ATOMIC
$e = \texttt{atomic}\ (e_1)$

$\Sigma; \overline{\Delta^{\mathcal{E}}}; \overline{u^{\mathcal{E}}} \vdash H; \overline{S_p}$ Assumption
$\cdot; \Sigma; \Delta; \mathcal{E}; u \vdash \texttt{atomic}\ (e_1) : \exists x : C.P' \setminus \omega | u'$ Assumption
$\cdot; \Sigma; \Delta; \mathsf{wt}; u \vdash e_1 : \exists x : C.P \setminus \omega | u' \quad \texttt{forget}_{\mathcal{E}}(P) = P'$
Inversion of only typing rule

$a; H; \langle e, S_p \rangle \to a'; H'; \langle e', S'_p \rangle; T'$
By rule E-ATOMIC-BEGIN

Only $\mathsf{active}(e)$ can be derrived.
e can step when $e = \circ$

CASE T-INATOMIC
$e = \texttt{inatomic}\ (e_1)$

$\Sigma; \overline{\Delta^{\mathcal{E}}}; \overline{u^{\mathcal{E}}} \vdash H; \overline{S_p}$ Assumption
$\cdot; \Sigma; \Delta; \mathcal{E}; u \vdash \texttt{inatomic}\ (e) : \exists x : C.P' \setminus \omega | u'$ Assumption
$\cdot; \Sigma; \Delta; \mathsf{wt}; u \vdash e : \exists x : C.P \setminus \omega | u' \quad \texttt{forget}_{\mathcal{E}}(P) = P'$
Inversion of typing rule.

SUBCASE: e_1 is a value It is only possible to derrive $\mathsf{active}(e)$.
When $a = \bullet$, we can step.
$a; H; \langle e, S_p \rangle \to a'; H'; \langle e', S'_p \rangle; T'$
By rule E-ATOMIC-EXIT

SUBCASE: e_1 is not a value.
e_2 can take a step Induction hypothesis
It is only possible to derrive $\mathsf{active}(e)$.
When $a = \bullet$, we can step.
$a; H; \langle e, S_p \rangle \to a'; H'; \langle e', S'_p \rangle; T'$
By rule E-INATOMIC

CASE T-READ
So $e = k \cdot o.f_i$.

$\Sigma; \overline{\Delta^{\mathcal{E}}}; \overline{u^{\mathcal{E}}} \vdash H; \overline{S_p}$ Assumption
$\cdot; \Sigma; \Delta; \mathcal{E}; u \vdash k \cdot o : \exists x : C.[x/o]P \setminus \emptyset | u$ Assumption
$\mathsf{readonly}(k_u)$ *implies* $\mathsf{readonly}(k) \quad \cdot; \Sigma; \Delta \vdash_{\mathcal{E}} P \quad \mathsf{localFields}(C) = \overline{f : C}$
Inversion of typing rule

$o \mapsto C(\ldots, f_i = k_i \cdot o_i, \ldots)@S$

Heap condition 3

$k \leq k_i$ Heap condition 4.d

SUBCASE: $\mathsf{writes}(k_u)$

$S = \mathsf{unpacked}(k_u)$ Heap condition 3

Only $\mathsf{not\text{-}active}(e)$ can be derrived.

We can step when $a = \circ$

$a; H; \langle e, S_p \rangle \to a'; H'; \langle e', S_p' \rangle; T'$

By rule E-READ-RW

SUBCASE: $\mathsf{readonly}(k_u)$

$S = \mathsf{unpacked}(s)$ Heap condition 3

$k = \mathsf{immutable}|\mathsf{pure}$ Above

Only $\mathsf{not\text{-}active}(e)$ can be derrived.

We can step when $a = \circ$

$a; H; \langle e, S_p \rangle \to a'; H'; \langle e', S_p' \rangle; T'$

By rule E-READ-R

CASE T-LET

So $e = \texttt{let}\ x = e_1\ \texttt{in}\ e_2$.

$\Sigma; \overline{\Delta^{\mathcal{E}}}; \overline{u^{\mathcal{E}}} \vdash H; \overline{S_p}$ Assumption

$\cdot; \Sigma; (\Delta_1, \Delta_2); \mathcal{E}; u \vdash \texttt{let}\ x = e_1\ \texttt{in}\ e_2 : E \setminus \omega_1 \cup \omega_2 | u'$ Assumption

$\Sigma; \Delta_2, P \vdash_{\mathcal{E}} P'$

$\cdot; \Sigma; \Delta_1; \mathcal{E}; u \vdash e_1 : \exists x : T.P \setminus \omega_1 | u_2 \quad x : C; \Sigma; P'; \mathcal{E}; u_2 \vdash e_2 : E\omega_2 | u'$

No permissions for ω_1 *in* Δ_2

Inversion of typing rule

SUBCASE: e_1 is a value.

$e_1 = k \cdot o$ No other values.

$k' \cdot o \in S_p \qquad k \leq k'$

By inversion of T-LOC and $\Sigma; \Delta \vdash S_p$

$o \in H$ Heap condition 1

Only $\mathsf{not\text{-}active}(e)$ possible when e_1 is a value.

We can step when $a = \circ$.

$a; H; \langle e, S_p \rangle \to a'; H'; \langle e', S_p' \rangle; T'$

By rule E-LET-V

SUBCASE: e_1 is not a value.

e_1 is well-typed Above

e_1 must step Induction hypothesis

If $\mathsf{active}(e)$ then $\mathsf{active}(e_1)$ active for Let

e_1 must step when $a = \bullet$ I.H

If $\mathsf{not\text{-}active}(e)$ then $\mathsf{not\text{-}active}(e_1)$ $\mathsf{not\text{-}active}$ for Let

e_1 must step when $a = \circ$ I.H

$a; H; \langle e, S_p \rangle \rightarrow a'; H'; \langle e', S'_p \rangle; T'$

By rule E-LET-E

CASE T-NEW

So $e = \texttt{new}\ C(\overline{k \cdot o})$.

$\Sigma; \overline{\Delta^{\mathcal{E}}}; \overline{u^{\mathcal{E}}} \vdash H; \overline{S_p}$ — Assumption

$\cdot; \Sigma; \Delta; \mathcal{E}; u \vdash \texttt{new}\ C(\overline{k \cdot o}) : \exists x : C.\mathsf{unique} \cdot x@s \setminus \emptyset | u$

Assumption

$\cdot; \Sigma; \Delta \vdash_{\mathcal{E}} [\overline{o}/\overline{f}]P \quad \overline{o : C} \subseteq \Sigma \quad \mathsf{init}(C) = \langle \exists \overline{f : C}.P, s \rangle$

Inversion of typing rule

$H; S_p \vdash [\overline{o}/\overline{f}]P$

Heap condition 4.c.

$\overline{k \cdot o} \in S_p$ — $\Sigma; \Delta \vdash S_p$

$\overline{k \leq k'}$ — $\Sigma; \Delta \vdash S_p$

We can only derrive $\mathsf{not\text{-}active}(e)$

We can step when $a = \circ$

$a; H; \langle e, S_p \rangle \rightarrow a'; H'; \langle e', S'_p \rangle; T'$

By rule E-NEW-E

CASE T-ASSIGN

$\Sigma; \overline{\Delta^{\mathcal{E}}}; \overline{u^{\mathcal{E}}} \vdash H; \overline{S_p}$ — Assumption

$\cdot; \Sigma; \Delta, \Delta'; \mathcal{E}; k' \cdot o'@s' \vdash o'.f'_i := k \cdot o : \exists x' : C_i.P' \otimes [o'.f_i/x]P \setminus \{o_i.f\} | k' \cdot o'@s'$

Assumption

$\mathsf{localFields}(C'') = \overline{f : C} \quad (o' : C') \in \Sigma \quad \mathsf{writes}(k')$

$\cdot; \Sigma; \Delta \vdash_{\mathcal{E}} k \cdot o : \exists x : C_i.P \quad \cdot; \Sigma; \Delta' \vdash_{\mathcal{E}} [o'.f_i/x']P'$

Inversion of typing rule.

$o \mapsto C(\ldots)@\mathsf{unpacked}(k') \in H$ — Heap condition 3

$k'_i \in S_p$ — Heap condition 4.d

$k_i \leq k_i$ — Heap condition 4.d

Only $\mathsf{not\text{-}active}(e)$ can be derrived.

We can step when $a = \circ$

$a; H; \langle e, S_p \rangle \rightarrow a'; H'; \langle e', S'_p \rangle; T'$

By rule E-ASSIGN-E

References

[1] Beckman, N., Bierhoff, K., Aldrich, J. Verifying Correct Usage of Atomic Blocks and Typestate. OOPSLA '08, Nashville, TN. October, 2008.

[2] Bierhoff, K., Aldrich, J. Modular Typestate Checking of Aliased Objects. OOPSLA ‘07, Montreal, Canada. October, 2007.

[3] Moore, K., Grossman, D. High-Level Small-Step Operational Semantics for Transactions. POPL ‘08, San Francisco, CA. January, 2008.