Explicit CN Soundness Proof

Dhruv Makwana

June 24, 2021

1 Weakening

If $\mathcal{C};\mathcal{L};\Phi;\mathcal{R} \sqsubseteq \mathcal{C}';\mathcal{L}';\Phi';\mathcal{R}'$ and $\mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash J$ then $\mathcal{C}';\mathcal{L}';\Phi';\mathcal{R}' \vdash J$.

Proof sketch: Induction over the typing judgements.

Assume: 1. $\mathcal{C};\mathcal{L};\Phi;\mathcal{R} \sqsubseteq \mathcal{C}';\mathcal{L}';\Phi';\mathcal{R}'$.
2. $\mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash J$.

Prove: $\mathcal{C}';\mathcal{L}';\Phi';\mathcal{R}' \vdash J$.

2 Substitution

2.1 Weakening for Substitution

Weakening for substitution: as above, but with $J = (\sigma) : (\mathcal{C}'';\mathcal{L}'';\Phi'';\mathcal{R}'')$.

Proof sketch: Induction over the substitution.

Assume: 1. $\mathcal{C};\mathcal{L};\Phi;\mathcal{R} \sqsubseteq \mathcal{C}';\mathcal{L}';\Phi';\mathcal{R}'$.
2. $\mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash (\sigma):(\mathcal{C}'';\mathcal{L}'';\Phi'';\mathcal{R}'')$.

Prove: $\mathcal{C}';\mathcal{L}';\Phi';\mathcal{R}' \vdash (\sigma):(\mathcal{C}'';\mathcal{L}'';\Phi'';\mathcal{R}'')$.

2.2 Substitution Lemma

If $\mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash (\sigma):(\mathcal{C}';\mathcal{L}';\Phi';\mathcal{R}')$ and $\mathcal{C}';\mathcal{L}';\Phi';\mathcal{R}' \vdash J$ then $\mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash \sigma(J)$.

Proof sketch: Induction over the typing judgements.

Assume: 1. $\mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash (\sigma):(\mathcal{C}';\mathcal{L}';\Phi';\mathcal{R}')$.
2. $\mathcal{C}';\mathcal{L}';\Phi';\mathcal{R}' \vdash J$.

Prove: $\mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash \sigma(J)$.

⟨1⟩1. Case: Ty_PVal_Var.
$\mathcal{C}';\mathcal{L}';\Phi' \vdash x \Rightarrow \beta$

⟨2⟩1. Have $x{:}\beta \in \mathcal{C}'$ (or $x{:}\beta \in \mathcal{L}'$).

⟨2⟩2. So $\exists pval.\ \mathcal{C};\mathcal{L};\Phi \vdash pval \Rightarrow \beta$ by Ty_Subs_Cons_{Comp,Log}.

⟨2⟩3. Since $pval = \sigma(x)$, we are done.

⟨1⟩2. Case: Ty_TPE_Let.
$\mathcal{C}'; \mathcal{L}'; \Phi' \vdash \texttt{let}\, ident\_or\_pattern = pexpr\, \texttt{in}\, tpexpr \Leftarrow y_2{:}\beta_2.\, term_2.$

⟨2⟩1. By induction,
1. $\mathcal{C}; \mathcal{L}; \Phi \vdash \sigma(pexpr) \Rightarrow y_1{:}\beta.\, \sigma(term_1)$
2. $\mathcal{C}, \mathcal{C}_1; \mathcal{L}, y_1{:}\beta; \Phi, term_1, \Phi' \vdash \sigma(tpexpr) \Leftarrow y_2{:}\beta.\, \sigma(term_2)$.

⟨2⟩2. $\mathcal{C}; \mathcal{L}; \Phi \vdash \sigma(\texttt{let}\, ident\_or\_pattern = pexpr\, \texttt{in}\, tpexpr) \Leftarrow y_2{:}\beta_2.\, \sigma(term_2)$ as required.

⟨1⟩3. Case: Ty_TVal_Log.
$\mathcal{C}'; \mathcal{L}'; \Phi'; \mathcal{R}' \vdash \texttt{done}\, pval,\ \overline{spine\_elem_i}^{\,i} \Leftarrow \exists\, y{:}\beta.\, ret.$

⟨2⟩1. By inversion and then induction,
1. $\mathcal{C}; \mathcal{L}; \Phi \vdash \sigma(pval) \Rightarrow \beta$
2. $\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \sigma(\texttt{done}\, \overline{spine\_elem_i}^{\,i}) \Leftarrow \sigma(pval/y, \cdot(ret))$.

⟨2⟩2. Therefore $\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \sigma(\texttt{done}\, pval,\ \overline{spine\_elem_i}^{\,i}) \Leftarrow \exists\, y{:}\beta.\, \sigma(ret)$.

⟨1⟩4. Case: Ty_Spine_Res.
$\mathcal{C}'; \mathcal{L}'; \Phi'; \mathcal{R}_1', \mathcal{R}_2 \vdash x = res\_term,\ \overline{x_i = spine\_elem_i}^{\,i} :: res \multimap arg \gg res\_term/x, \psi; ret$

⟨2⟩1. By inversion and then induction,
1. $\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R}_1 \vdash \sigma(res\_term) \Leftarrow \sigma(res)$.
2. $\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R}_2 \vdash\ \overline{x_i = \sigma(spine\_elem_i)}^{\,i} :: \sigma(res) \multimap \sigma(arg) \gg \sigma(\psi); \sigma(ret)$.

⟨2⟩2. Hence $\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R}_1, \mathcal{R}_2 \vdash x = \sigma(res\_term),\ \overline{x_i = \sigma(spine\_elem_i)}^{\,i} :: \sigma(res \multimap arg) \gg \sigma(res\_term/x, \psi); \sigma(ret)$ as required.

2.3 Identity Extension

If $\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash (\sigma){:}(\mathcal{C}'; \mathcal{L}'; \Phi'; \mathcal{R}')$ then $\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R}_1, \mathcal{R} \vdash (\sigma, \mathrm{id}){:}(\mathcal{C}, \mathcal{C}'; \mathcal{L}, \mathcal{L}'; \Phi, \Phi'; \mathcal{R}_1, \mathcal{R}')$.

Proof sketch: Induction over the substitution.

Assume: $\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash (\sigma){:}(\mathcal{C}'; \mathcal{L}'; \Phi'; \mathcal{R}')$.

Prove: $\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R}_1, \mathcal{R} \vdash (\sigma, \mathrm{id}){:}(\mathcal{C}, \mathcal{C}'; \mathcal{L}, \mathcal{L}'; \Phi, \Phi'; \mathcal{R}_1, \mathcal{R}')$.
⟨1⟩1. $\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R}_1 \vdash (\mathrm{id}){:}(\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R}_1)$.
Proof: By induction on each of $\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R}_1$.

⟨1⟩2. $\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R}_1, \mathcal{R} \vdash (\sigma, \mathrm{id}){:}(\mathcal{C}, \mathcal{C}'; \mathcal{L}, \mathcal{L}'; \Phi, \Phi'; \mathcal{R}_1, \mathcal{R}')$
Proof: By induction on σ with base case as above.

2.4 Usable Substitution Lemma

If $\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash (\sigma){:}(\mathcal{C}'; \mathcal{L}'; \Phi'; \mathcal{R}')$ and $\mathcal{C}, \mathcal{C}'; \mathcal{L}, \mathcal{L}'; \Phi, \Phi'; \mathcal{R}_1, \mathcal{R}' \vdash J$ then $\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R}_1, \mathcal{R} \vdash \sigma(J)$.

Proof sketch: Apply identity extension then substitution lemma.

Assume: 1. $\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash (\sigma){:}(\mathcal{C}'; \mathcal{L}'; \Phi'; \mathcal{R}')$.
2. $\mathcal{C}, \mathcal{C}'; \mathcal{L}, \mathcal{L}'; \Phi, \Phi'; \mathcal{R}_1, \mathcal{R}' \vdash J$.

Prove: $\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R}_1, \mathcal{R} \vdash \sigma(J)$.

$\langle 1 \rangle 1$. $\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash (\sigma, \mathrm{id}) : (\mathcal{C}, \mathcal{C}'; \mathcal{L}, \mathcal{L}'; \Phi, \Phi'; \mathcal{R}_1, \mathcal{R}')$.
PROOF: Apply identity extension to 1.

$\langle 1 \rangle 2$. $\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R}_1, \mathcal{R} \vdash (\sigma, \mathrm{id})(J)$.
PROOF: Apply substitution lemma to $\langle 1 \rangle 1$.

$\langle 1 \rangle 3$. $\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R}_1, \mathcal{R} \vdash \sigma(J)$.
PROOF: $\mathrm{id}(J) = J$.

3 Progress

If $\cdot; \cdot; \cdot; \mathcal{R} \vdash e \Leftrightarrow t$ then either value(e) or $\forall h : R.\ \exists e', h'.\ \langle h; e \rangle \longrightarrow \langle h'; e' \rangle$.

PROOF SKETCH: Induction over the typing rules.

ASSUME: $\cdot; \cdot; \cdot; \mathcal{R} \vdash e \Leftrightarrow t$.
PROVE: either value(e) or $\forall h : R.\ \exists e', h'.\ \langle h; e \rangle \longrightarrow \langle h'; e' \rangle$.

4 Framing

If $\langle h_1; e \rangle \longrightarrow \langle h_1'; e' \rangle$ and h_1, h_2 disjoint then $\langle h_1 + h_2; e \rangle \longrightarrow \langle h_1' + h_2; e' \rangle$.

PROOF SKETCH: Induction over the operational rules.

ASSUME: 1. $\langle h_1; e \rangle \longrightarrow \langle h_1'; e' \rangle$.
2. h_1, h_2 disjoint.

PROVE: $\langle h_1 + h_2; e \rangle \longrightarrow \langle h_1' + h_2; e' \rangle$.

5 Type Preservation

5.1 Ty_Spine_* and Decons_Arg_* construct same substitution and return type

If $\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \overline{x_i = \mathit{spine\_elem}_i}^{\,i} :: \mathit{arg} \gg \sigma; \mathit{ret}$ and $\overline{x_i = \mathit{spine\_elem}_i}^{\,i} :: \mathit{arg} \gg \sigma'; \mathit{ret}'$ then $\sigma = \sigma'$ and $\mathit{ret} = \mathit{ret}'$.

PROOF SKETCH: Induction over *arg*.

5.2 Pointed-to values have type β_τ

For $pt = \_ \overset{\checkmark}{\mapsto}_\tau \mathit{pval}$, if $\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash pt \Leftarrow pt$ then $\mathcal{C}; \mathcal{L}; \Phi \vdash \mathit{pval} \Rightarrow \beta_\tau$.

PROOF SKETCH: Induction over the typing judgements. Only TY_ACTION_STORE create such permissions, and its premise $\mathcal{C}; \mathcal{L}; \Phi \vdash \mathit{pval}_1 \Rightarrow \beta_\tau$ ensures the desired property. TY_ACTION_LOAD simply preserves the property.

5.3 Type Preservation Statement and Proof

If $\cdot; \cdot; \cdot; \mathcal{R} \vdash e \Leftrightarrow t$ then $\forall h : \mathcal{R}, e', h' : \mathcal{R}'.\ \langle h; e \rangle \longrightarrow \langle h'; e' \rangle \implies \cdot; \cdot; \cdot; \mathcal{R}' \vdash e' \Leftrightarrow t$.

PROOF SKETCH: Induction over the typing rules.

ASSUME: 1. $\cdot;\cdot;\cdot;\mathcal{R} \vdash e \Leftrightarrow t$
2. arbitrary $h : \mathcal{R}, e', h' : \mathcal{R}'$
3. $\langle h; e\rangle \longrightarrow \langle h'; e'\rangle$.

PROVE: $\cdot;\cdot;\cdot;\mathcal{R}' \vdash e' \Leftrightarrow t$.

$\langle 1\rangle 1$. CASE: TY_MEMOP_PTRVALIDFORDEREF.
LET: $pt = mem\_ptr \overset{\checkmark}{\mapsto}_\tau \_$.
$ret = \Sigma\, y{:}\texttt{bool}.\ y = \texttt{aligned}\,(\tau, mem\_ptr) \wedge pt \otimes \texttt{I}$.
ASSUME: 1. $\cdot;\cdot;\cdot;\mathcal{R} \vdash \texttt{ptrValidForDeref}\,(\tau, mem\_ptr, pt) \Rightarrow ret$.
2. $\langle \cdot + \{pt\}; \texttt{ptrValidForDeref}\,(\tau, mem\_ptr, pt)\rangle \longrightarrow \langle \cdot + \{pt\}; \texttt{done}\, bool\_value, pt\rangle$.
PROVE: $\cdot;\cdot;\cdot;\mathcal{R} \vdash \texttt{done}\, bool\_value, pt \Leftarrow ret$.

$\langle 2\rangle 1$. $\cdot;\cdot;\cdot;\mathcal{R} \vdash pt \Leftarrow pt$, by inversion on 1.

$\langle 2\rangle 2$. $R = \cdot, \_{:}pt$, by TY_RES_POINTSTO.

$\langle 2\rangle 3$. $bool\_value = \texttt{aligned}\,(\tau, mem\_ptr)$ by construction of $bool\_value$ (inversion on 2).

$\langle 2\rangle 4$. $\cdot;\cdot;\cdot \vdash bool\_value \Rightarrow \texttt{bool}$ by TY_PVAL_{TRUE,FALSE}.

$\langle 2\rangle 5$. By TY_TVAL_I, and then $\langle 2\rangle 2$ – $\langle 2\rangle 4$ with TY_TVAL_{RES,PHI,COMP} respectively, we are done.

$\langle 1\rangle 2$. CASE: TY_PE_ARRAY_SHIFT.
LET: $term = mem\_ptr +_{\text{ptr}} (mem\_int \times \text{size\_of}(\tau))$.
ASSUME: 1. $\cdot;\cdot;\cdot \vdash \texttt{array\_shift}\,(mem\_ptr, \tau, mem\_int) \Rightarrow y{:}\texttt{loc}.\ y = term$.
2. $\langle \texttt{array\_shift}\,(mem\_ptr, \tau, mem\_int)\rangle \longrightarrow \langle mem\_ptr'\rangle$.
PROVE: $\cdot;\cdot;\cdot \vdash mem\_ptr' \Rightarrow y{:}\texttt{loc}.\ y = term$.
PROOF: By TY_PVAL_OBJ_INT, TY_PVAL_OBJ, TY_PE_VAL and construction of mem_ptr' (inversion on 2).

$\langle 1\rangle 3$. CASE: TY_PE_MEMBER_SHIFT.
PROOF SKETCH: Similar to TY_ARRAY_SHIFT.

$\langle 1\rangle 4$. CASE: TY_PE_NOT.
ASSUME: 1. $\cdot;\cdot;\cdot \vdash \texttt{not}\,(bool\_value) \Rightarrow y{:}\texttt{bool}.\ y = \neg\, bool\_value$.
2. $\langle \texttt{not}\,(\texttt{True})\rangle \longrightarrow \langle \texttt{False}\rangle$ or $\langle \texttt{not}\,(\texttt{False})\rangle \longrightarrow \langle \texttt{True}\rangle$.
PROVE: $\cdot;\cdot;\cdot \vdash bool\_value' \Rightarrow y{:}\texttt{bool}.\ y = \neg\, bool\_value$.
PROOF: By TY_PVAL_{TRUE,FALSE}, TY_PE_VAL and 2.

$\langle 1\rangle 5$. CASE: TY_PE_ARITH_BINOP.
LET: $term = mem\_int_1\ binop_{arith}\ mem\_int_2$.
ASSUME: 1. $\cdot;\cdot;\cdot \vdash mem\_int_1\ binop_{arith}\ mem\_int_2 \Rightarrow y{:}\texttt{integer}.\ y = term$.
2. $\langle mem\_int_1\ binop_{arith}\ mem\_int_2\rangle \longrightarrow \langle mem\_int\rangle$.
PROVE: $\cdot;\cdot;\cdot \vdash mem\_int \Rightarrow y{:}\texttt{integer}.\ y = term$.
PROOF: By TY_PVAL_OBJ_INT, TY_PVAL_OBJ, TY_PE_VAL and construction of mem_int (inversion on 2).

$\langle 1\rangle 6$. CASE: TY_PE_{REL,BOOL}_BINOP.
PROOF SKETCH: Similar to TY_PE_ARITH_BINOP.

⟨1⟩7. Case: Ty_PE_Call.
Assume: 1. $\cdot;\cdot;\cdot \vdash name(\overline{pval_i}^{\,i}) \Rightarrow y{:}\beta.\ \sigma(term)$.
2. $\langle name(\overline{pval_i}^{\,i})\rangle \longrightarrow \langle \sigma'(tpexpr){:}(y{:}\beta'.\ \sigma'(term'))\rangle$.
Prove: $\cdot;\cdot;\cdot \vdash \sigma(tpexpr) \Leftarrow y{:}\beta.\ \sigma(term)$.

⟨2⟩1. $name{:}pure\_arg \equiv \overline{x_i}^{\,i} \mapsto tpexpr \in \texttt{Globals}$ by inversion (on either assumption).

⟨2⟩2. $\cdot;\cdot;\cdot;\cdot \vdash \overline{x_i = pval_i}^{\,i} :: pure\_arg \gg \sigma; \Sigma\, y{:}\beta.\ term \wedge \texttt{I}$ by inversion on 1.

⟨2⟩3. $\beta = \beta'$, $term = term'$ and $\sigma = \sigma'$ by induction on *pure_arg*.
Follows from lemma 5.1.

⟨2⟩4. $\cdot;\cdot;\cdot;\cdot \vdash (\sigma){:}(\mathcal{C};\cdot;\Phi;\cdot)$.
Proof: Constructing such a substitution requires $\overline{\cdot;\cdot;\cdot \vdash pval_i \Rightarrow \beta_i}^{\,i}$ for each $x_i{:}\beta_i \in \mathcal{C}$ which can be deduced from ⟨2⟩2.

⟨2⟩5. $\mathcal{C}'';\cdot;\Phi'' \vdash tpexpr \Leftarrow y{:}\beta''.term''$ where $\overline{x_i}^{\,i} :: pure\_arg \rightsquigarrow \mathcal{C}'';\cdot;\Phi'';\cdot \mid \Sigma\, y{:}\beta''.term'' \wedge \texttt{I}$ formalises the assumption that all global functions and labels are well-typed.

⟨2⟩6. $\mathcal{C} = \mathcal{C}''$, $\Phi = \Phi''$, $\beta = \beta''$ and $term = term''$.
Proof: By induction on *pure_arg*.

⟨2⟩7. Apply usable substitution lemma to ⟨2⟩4 and ⟨2⟩5 to finish proof.

⟨1⟩8. Case: Ty_Fill_In.
Assume: 1. $\cdot;\cdot;\cdot \vdash \texttt{assert\_undef}\,(\texttt{True},\ UB\_name) \Rightarrow y{:}\texttt{unit}.\ y = \texttt{unit}$.
2. $\langle \texttt{assert\_undef}\,(\texttt{True},\ UB\_name)\rangle \longrightarrow \langle \texttt{Unit}\rangle$.
Prove: $\cdot;\cdot;\cdot \vdash \texttt{Unit} \Rightarrow y{:}\texttt{unit}.\ y = \texttt{unit}$.
Proof: By Ty_PVal_Unit and Ty_PE_Val.

⟨1⟩9. Case: Ty_PE_Bool_To_Integer.
Let: $term = \texttt{if}\ bool\_value\ \texttt{then}\ 1\ \texttt{else}\ 0$.
Assume: 1. $\cdot;\cdot;\cdot \vdash \texttt{bool\_to\_integer}\,(bool\_value) \Rightarrow y{:}\texttt{integer}.\ y = term$.
2. $\langle \texttt{bool\_to\_integer}\,(\texttt{True})\rangle \longrightarrow \langle 1\rangle$ or $\langle \texttt{bool\_to\_integer}\,(\texttt{False})\rangle \longrightarrow \langle 0\rangle$.
Prove: $\cdot;\cdot;\cdot \vdash mem\_int \Rightarrow y{:}\texttt{integer}.\ y = term$
Proof: By cases on *bool_value*, then applying Ty_PVal_{True,False} and Ty_PE_Val.

⟨1⟩10. Case: Ty_PE_WrapI.
Proof sketch: Similar to Ty_PE_Bool_To_Integer, except by cases on $abbrev_2 \leq \text{max\_int}_\tau$, then applying Ty_PVal_Obj_Int, Ty_PVal_Obj and Ty_PE_Val.

⟨1⟩11. Case: Ty_TPE_If.
Proof: See Ty_Seq_TE_If for a more general case and proof.

⟨1⟩12. Case: Ty_TPE_Let.
Proof: See Ty_Seq_TE_Let for a more general case and proof.

⟨1⟩13. Case: Ty_TPE_LetT.
Proof: See Ty_Seq_TE_LetT for a more general case and proof.

⟨1⟩14. Case: Ty_TPE_Case.
Proof: See Ty_Seq_TE_Case for a more general case and proof.

⟨1⟩15. Case: Ty_Action_Create.

Let: $pt = mem\_ptr \overset{\times}{\mapsto}_\tau pval$.
$term = \texttt{representable}\,(\tau*, y_p) \wedge \texttt{alignedI}\,(mem\_int, y_p)$.
$ret = \Sigma\, y_p{:}\texttt{loc}.\ term \wedge \exists\, y{:}\beta_\tau.\ y_p \overset{\times}{\mapsto}_\tau y \otimes \texttt{I}$.

Assume: 1. $\cdot;\cdot;\cdot;\cdot \vdash \texttt{create}\,(mem\_int, \tau) \Rightarrow ret$.
2. $\langle \cdot; \texttt{create}\,(mem\_int, \tau)\rangle \longrightarrow \langle \cdot + \{pt\}; \texttt{done}\, mem\_ptr, pval, pt\rangle$.

Prove: $\cdot;\cdot;\cdot;\cdot, \_{:}pt \vdash \texttt{done}\, mem\_ptr, pval, pt \Leftarrow ret$.

⟨2⟩1. $\cdot;\cdot;\cdot \vdash mem\_ptr \Rightarrow \texttt{loc}$ by Ty_PVal_Obj_Int and Ty_PVal_Obj.

⟨2⟩2. $\texttt{smt}\,(\cdot \Rightarrow term)$ by construction of mem_ptr.

⟨2⟩3. $\cdot;\cdot;\cdot \vdash pval \Rightarrow \beta_\tau$ by construction of $pval$.

⟨2⟩4. $\cdot;\cdot;\cdot;\cdot, \_{:}pt \vdash pt \Leftarrow pt$ by Ty_Res_PointsTo.

⟨2⟩5. By Ty_TVal_I and then ⟨2⟩4 – ⟨2⟩1 with Ty_TVal_{Res,Log,Phi,Comp} respectively, we are done.

⟨1⟩16. Case: Ty_Action_Load.

Let: $pt = mem\_ptr \overset{\checkmark}{\mapsto}_\tau pval$.
$ret = \Sigma\, y{:}\beta_\tau.\ y = pval \wedge pt \otimes \texttt{I}$.

Assume: 1. $\cdot;\cdot;\cdot;\cdot, \_{:}pt \vdash \texttt{load}\,(\tau, mem\_ptr, \_, pt) \Rightarrow ret$.
2. $\langle \cdot + \{pt\}; \texttt{load}\,(\tau, mem\_ptr, \_, pt)\rangle \longrightarrow \langle \cdot + \{pt\}; \texttt{done}\, pval, pt\rangle$.

Prove: $\cdot;\cdot;\cdot;\cdot, \_{:}pt \vdash \texttt{done}\, pval, pt \Leftarrow ret$

⟨2⟩1. $\cdot;\cdot;\cdot;\mathcal{R} \vdash pt \Leftarrow pt$, by inversion on 1.

⟨2⟩2. $\texttt{smt}\,(\cdot \Rightarrow pval = pval)$ trivially.

⟨2⟩3. $\cdot;\cdot;\cdot \vdash pval \Rightarrow \beta_\tau$ by ⟨2⟩1 and lemma 5.2.

⟨2⟩4. By Ty_TVal_I and then ⟨2⟩1 – ⟨2⟩3 with Ty_TVal_{Res,Phi,Comp} respectively, we are done.

⟨1⟩17. Case: Ty_Action_Store.

Let: $pt = mem\_ptr \overset{\checkmark}{\mapsto}_\tau \_$.
$pt' = mem\_ptr \overset{\checkmark}{\mapsto}_\tau pval$.
$ret = \Sigma\, \_{:}\texttt{unit}.\ pt' \otimes \texttt{I}$.

Assume: 1. $\cdot;\cdot;\cdot;\cdot, \_{:}pt \vdash \texttt{store}\,(\_, \tau, pval_0, pval_1, \_, pt) \Rightarrow ret$.
2. $\langle \cdot + \{pt\}; \texttt{store}\,(\_, \tau, mem\_ptr, pval, \_, pt)\rangle \longrightarrow \langle \cdot + \{pt'\}; \texttt{done}\, \texttt{Unit}, pt'\rangle$.

Prove: $\cdot;\cdot;\cdot;\cdot, \_{:}pt' \vdash \texttt{done}\, \texttt{Unit}, pt' \Leftarrow ret$

⟨2⟩1. $\cdot;\cdot;\cdot \vdash \texttt{Unit} \Rightarrow \texttt{unit}$ by Ty_PVal_Unit.
.

⟨2⟩2. $\cdot;\cdot;\cdot;\cdot, \_{:}pt' \vdash pt' \Leftarrow pt'$ by Ty_Res_PointsTo.

⟨2⟩3. By Ty_TVal_I and ⟨2⟩1 and ⟨2⟩2 with Ty_TVal_{Res,Comp} respectively, we are done.

⟨1⟩18. Case: Ty_Action_Kill_Static.

Let: $pt = mem\_ptr \mapsto_\tau \_$.

Assume: 1. $\cdot;\cdot;\cdot;\cdot, \_{:}pt \vdash \texttt{kill}\,(\texttt{static}\,\tau, pval_0, pt) \Rightarrow \Sigma\, \_{:}\texttt{unit}.\ \texttt{I}$.

2. $\langle \cdot + \{pt\}; \texttt{kill}\,(\texttt{static}\,\tau, mem\_ptr, pt)\rangle \longrightarrow \langle h; \texttt{done}\,\texttt{Unit}\rangle$.

PROVE: $\cdot;\cdot;\cdot;\cdot \vdash \texttt{done}\,\texttt{Unit} \Leftarrow \Sigma\,\_{:}\texttt{unit}.\,\texttt{I}$

PROOF: By TY_TVAL_I, TY_PVAL_UNIT and then TY_TVAL_COMP.

⟨1⟩19. CASE: TY_FILL_IN.

LET:

ASSUME: 1. .
2. .

PROVE:

⟨1⟩20. CASE: TY_FILL_IN.

LET:

ASSUME: 1. .
2. .

PROVE:

⟨1⟩21. CASE: TY_FILL_IN.

LET:

ASSUME: 1. .
2. .

PROVE:

⟨1⟩22. CASE: TY_FILL_IN.

LET:

ASSUME: 1. .
2. .

PROVE:

⟨1⟩23. CASE: TY_FILL_IN.

LET:

ASSUME: 1. .
2. .

PROVE:

⟨1⟩24. CASE: TY_FILL_IN.

LET:

ASSUME: 1. .
2. .

PROVE:

⟨1⟩25. CASE: TY_FILL_IN.

LET:

ASSUME: 1. .
2. .

PROVE:

⟨1⟩26. CASE: TY_FILL_IN.

LET:

ASSUME: 1. .
2. .

PROVE:

6 Typing Judgements

$$
\begin{array}{lll}
\mathit{object\_value\_jtype} & ::= & \\
 & \mid & \mathcal{C}; \mathcal{L}; \Phi \vdash \mathit{object\_value} \Rightarrow \texttt{obj}\ \beta \\
 & & \\
\mathit{pval\_jtype} & ::= & \\
 & \mid & \mathcal{C}; \mathcal{L}; \Phi \vdash \mathit{pval} \Rightarrow \beta \\
 & & \\
\mathit{res\_jtype} & ::= & \\
 & \mid & \Phi \vdash \mathit{res} \equiv \mathit{res}' \\
 & \mid & \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \mathit{res\_term} \Leftarrow \mathit{res} \\
 & & \\
\mathit{spine\_jtype} & ::= & \\
 & \mid & \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \overline{x_i = \mathit{spine\_elem}_i}^{\,i} :: \mathit{arg} \gg \sigma; \mathit{ret} \\
 & & \\
\mathit{pexpr\_jtype} & ::= & \\
 & \mid & \mathcal{C}; \mathcal{L}; \Phi \vdash \mathit{pexpr} \Rightarrow \mathit{ident}{:}\beta.\, \mathit{term} \\
 & & \\
\mathit{tpval\_jtype} & ::= & \\
 & \mid & \mathcal{C}; \mathcal{L}; \Phi \vdash \mathit{tpval} \Leftarrow \mathit{ident}{:}\beta.\, \mathit{term} \\
 & & \\
\mathit{tpexpr\_jtype} & ::= & \\
 & \mid & \mathcal{C}; \mathcal{L}; \Phi \vdash \mathit{tpexpr} \Leftarrow \mathit{ident}{:}\beta.\, \mathit{term} \\
 & & \\
\mathit{action\_jtype} & ::= & \\
 & \mid & \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \mathit{mem\_action} \Rightarrow \mathit{ret} \\
 & & \\
\mathit{memop\_jtype} & ::= & \\
 & \mid & \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \mathit{mem\_op} \Rightarrow \mathit{ret} \\
 & & \\
\mathit{seq\_expr\_jtype} & ::= & \\
 & \mid & \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \mathit{seq\_expr} \Rightarrow \mathit{ret} \\
 & & \\
\mathit{is\_expr\_jtype} & ::= & \\
 & \mid & \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \mathit{is\_expr} \Rightarrow \mathit{ret} \\
 & & \\
\mathit{tval\_jtype} & ::= & \\
 & \mid & \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \mathit{tval} \Leftarrow \mathit{ret} \\
 & & \\
\mathit{texpr\_jtype} & ::= & \\
 & \mid & \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \mathit{seq\_texpr} \Leftarrow \mathit{ret} \\
 & \mid & \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \mathit{is\_texpr} \Leftarrow \mathit{ret} \\
 & \mid & \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \mathit{texpr} \Leftarrow \mathit{ret}
\end{array}
$$

7 Opsem Judgements

$$
\begin{array}{lll}
\mathit{pure\_opsem\_jtype} & ::= & \\
 & | & \langle \mathit{pexpr} \rangle \longrightarrow \langle \mathit{pexpr}' \rangle \\
 & | & \langle \mathit{pexpr} \rangle \longrightarrow \langle \mathit{tpexpr}{:}(y{:}\beta.\ \mathit{term}) \rangle \\
 & | & \langle \mathit{tpexpr} \rangle \longrightarrow \langle \mathit{tpexpr}' \rangle
\end{array}
$$

$$
\begin{array}{lll}
\mathit{opsem\_jtype} & ::= & \\
 & | & \langle \mathit{seq\_expr} \rangle \longrightarrow \langle \mathit{texpr}{:}\mathit{ret} \rangle \\
 & | & \langle h; \mathit{seq\_texpr} \rangle \longrightarrow \langle h'; \mathit{texpr} \rangle \\
 & | & \langle h; \mathit{mem\_op} \rangle \longrightarrow \langle h'; \mathit{tval} \rangle \\
 & | & \langle h; \mathit{mem\_action} \rangle \longrightarrow \langle h'; \mathit{tval} \rangle \\
 & | & \langle h; \mathit{is\_expr} \rangle \longrightarrow \langle h'; \mathit{is\_expr}' \rangle \\
 & | & \langle h; \mathit{is\_texpr} \rangle \longrightarrow \langle h'; \mathit{texpr} \rangle \\
 & | & \langle h; \mathit{texpr} \rangle \longrightarrow \langle h'; \mathit{texpr}' \rangle
\end{array}
$$