Explicit CN Soundness Proof

Dhruv Makwana

July 5, 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 STRATEGY: 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 STRATEGY: 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 STRATEGY: 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.
$\mathcal{C}'; \mathcal{L}'; \Phi' \vdash \texttt{let}\, \mathit{ident\_or\_pattern} = \mathit{pexpr}\, \texttt{in}\, \mathit{tpexpr} \Leftarrow y_2{:}\beta_2.\ \mathit{term}_2$.

⟨2⟩1. By induction,
1. $\mathcal{C}; \mathcal{L}; \Phi \vdash \sigma(\mathit{pexpr}) \Rightarrow y_1{:}\beta.\ \sigma(\mathit{term}_1)$
2. $\mathcal{C}, \mathcal{C}_1; \mathcal{L}, y_1{:}\beta; \Phi, \mathit{term}_1, \Phi' \vdash \sigma(\mathit{tpexpr}) \Leftarrow y_2{:}\beta.\ \sigma(\mathit{term}_2)$.

⟨2⟩2. $\mathcal{C}; \mathcal{L}; \Phi \vdash \sigma(\texttt{let}\, \mathit{ident\_or\_pattern} = \mathit{pexpr}\, \texttt{in}\, \mathit{tpexpr}) \Leftarrow y_2{:}\beta_2.\ \sigma(\mathit{term}_2)$ as required.

⟨1⟩3. CASE: TY_TVAL_LOG.
$\mathcal{C}'; \mathcal{L}'; \Phi'; \mathcal{R}' \vdash \texttt{done}\, \mathit{pval},\ \overline{\mathit{spine\_elem}_i}^{\,i} \Leftarrow \exists\, y{:}\beta.\ \mathit{ret}$.

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

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

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

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

⟨2⟩2. Hence $\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R}_1, \mathcal{R}_2 \vdash x = \sigma(\mathit{res\_term}),\ \overline{x_i = \sigma(\mathit{spine\_elem}_i)}^{\,i} :: \sigma(\mathit{res} \multimap \mathit{arg}) \gg \sigma(\mathit{res\_term}/x, \psi); \sigma(\mathit{ret})$ 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 Let-friendly 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 (2.2) 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 $\mathrm{value}(e)$ or $\forall h : R.\ \exists e', h'.\ \langle h; e \rangle \longrightarrow \langle h'; e' \rangle$.

Proof Strategy: Induction over the typing rules.

Assume: $\cdot;\cdot;\cdot;\mathcal{R} \vdash e \Leftrightarrow t$.
Prove: either $\mathrm{value}(e)$ 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 Strategy: 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 $\mathit{pt} = \_ \overset{\checkmark}{\mapsto}_\tau \mathit{pval}$, if $\mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash \mathit{pt} \Leftarrow \mathit{pt}$ 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 Deconstructing a pattern leads to a well-typed substitution

First, computational part.

Assume: 1. $\cdot;\cdot;\cdot \vdash \mathit{pval} \Rightarrow \beta_1$.
2. $\mathit{ident\_or\_pattern}{:}\beta \rightsquigarrow \mathcal{C}\ \mathtt{with}\ \mathit{term}$.

3. $ident\_or\_pattern = pval \rightsquigarrow \sigma$.

PROVE: $\cdot;\cdot;\cdot;\cdot \vdash (\sigma){:}(\mathcal{C};\cdot;\cdot;\cdot)$.

PROOF SKETCH: By induction over 2.

⟨1⟩1. CASE: TY_PAT_SYM_OR_PATTERN_SYM and TY_PAT_COMP_SYM_ANNOT.
$\sigma = pval/x, \cdot$ and $\mathcal{C} = \cdot, x{:}\beta$.
PROOF: By TY_SUBS_CONS_COMP and 1 and TY_SUBS_CONS_PHI.

⟨1⟩2. CASE: TY_PAT_NO_SYM_ANNOT and TY_PAT_COMP_NIL.
σ and $\mathcal{C}$ are empty.
PROOF: By TY_SUBS_EMPTY, we are done.

⟨1⟩3. CASE: TY_PAT_COMP_{SPECIFIED,CONS,TUPLE,ARRAY}.
PROOF: By induction (and concatenating well-typed substitutions).

Now, resource part.

ASSUME: 1. $\cdot;\cdot;\cdot;\mathcal{R} \vdash res\_term \Leftarrow res$.
2. $res\_pattern{:}res \rightsquigarrow \mathcal{L};\Phi;\mathcal{R}'$.
3. $res\_pattern = res\_term \rightsquigarrow \sigma$.

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

PROOF SKETCH: By induction over 2.

⟨1⟩1. CASE: TY_PAT_RES_EMPTY.
$res\_pattern = res\_term = res = \texttt{emp}$. $\sigma, \mathcal{L}, \Phi, \mathcal{R}, \mathcal{R}'$ are all empty.
PROOF: By TY_SUBS_EMPTY, we are done.

⟨1⟩2. CASE: TY_PAT_RES_POINTSTO.
$res\_pattern = res\_term = res = pt$. $\sigma = \cdot$, $\mathcal{L} = \cdot$, $\Phi = \cdot$, $\mathcal{R} = \mathcal{R}' = \cdot, pt$.
PROOF: By TY_SUBS_CONS_RES_ANON.

⟨1⟩3. CASE: TY_PAT_RES_VAR.
$res\_pattern = r$, $\sigma = res\_term/x, \cdot$, $\mathcal{L} = \cdot$, $\Phi = \cdot$, $\mathcal{R}' = \cdot, x{:}res$.
PROOF: By TY_SUBS_CONS_RES_NAMED.

⟨1⟩4. CASE: TY_PAT_RES_SEPCONJ.
PROOF: By induction (and concatenating well-typed substitutions).

⟨1⟩5. CASE: TY_PAT_RES_CONJ.
PROOF: By induction and TY_SUBS_CONS_PHI.

⟨1⟩6. CASE: TY_PAT_RES_PACK.
$res\_pattern = \texttt{pack}\,(x, res\_pattern')$, $res\_term = \texttt{pack}\,(pval, res\_term')$, $res = \exists\, x{:}\beta.\ res'$.
$\sigma = pval/x, \sigma'$, $\mathcal{L} = \mathcal{L}', x{:}\beta$, $\mathcal{R} = \mathcal{R}'$.
PROOF: By induction and TY_SUBS_CONS_LOG.

Now, full proof.

ASSUME: 1. $\overline{ret\_pattern_i = spine\_elem_i}^{\,i} \rightsquigarrow \sigma$.
2. $\cdot;\cdot;\cdot;\mathcal{R} \vdash \texttt{done}\ \overline{spine\_elem_i}^{\,i} \Leftarrow ret$.
3. $\overline{ret\_pattern_i}^{\,i}{:}ret \rightsquigarrow \mathcal{C};\mathcal{L};\Phi;\mathcal{R}'$.

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

PROOF SKETCH: Induction on 3. Base case by TY_SUBS_EMPTY. TY_RET_PAT_{COMP,RES} by induction, well-typed computational / resource substitutions and concatenating well-typed substitutions. TY_RET_PAT_{LOG,PHI} by induction and TY_SUBS_CONS_{LOG,PHI}.

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

⟨1⟩1. 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).

⟨1⟩2. CASE: TY_PE_MEMBER_SHIFT.
PROOF SKETCH: Similar to TY_ARRAY_SHIFT.

⟨1⟩3. 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.

⟨1⟩4. 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).

⟨1⟩5. CASE: TY_PE_{REL,BOOL}_BINOP.
PROOF SKETCH: Similar to TY_PE_ARITH_BINOP.

⟨1⟩6. CASE: TY_PE_CALL.
PROOF: See TY_SEQ_E_CALL for a more general case and proof.

⟨1⟩7. CASE: TY_PE_ASSERT_UNDEF.
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⟩8. 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⟩9. 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⟩10. CASE: TY_TPE_IF.
PROOF: See TY_SEQ_TE_IF for a more general case and proof.

⟨1⟩11. CASE: TY_TPE_LET.
PROOF: See TY_SEQ_TE_LET for a more general case and proof.

⟨1⟩12. CASE: TY_TPE_LETT.
PROOF: See TY_SEQ_TE_LETT for a more general case and proof.

⟨1⟩13. CASE: TY_TPE_CASE.
PROOF: See TY_SEQ_TE_CASE for a more general case and proof.

⟨1⟩14. 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⟩15. 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. $\cdot;\cdot;\cdot;\cdot, pt \vdash pt \Leftarrow pt$, by inversion on 1.

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

⟨2⟩3. $\cdot; \cdot; \cdot \vdash \mathit{pval} \Rightarrow \beta_\tau$ 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⟩16. Case: Ty_Action_Store.
Let: $pt = \mathit{mem\_ptr} \overset{\checkmark}{\mapsto}_\tau \_$.
$pt' = \mathit{mem\_ptr} \overset{\checkmark}{\mapsto}_\tau \mathit{pval}$.
$\mathit{ret} = \Sigma\, \_{:}\texttt{unit}.\ pt' \otimes \texttt{I}$.
Assume: 1. $\cdot; \cdot; \cdot; \cdot, pt \vdash \texttt{store}\,(\_, \tau, \mathit{pval}_0, \mathit{pval}_1, \_, pt) \Rightarrow \mathit{ret}$.
2. $\langle \cdot + \{pt\}; \texttt{store}\,(\_, \tau, \mathit{mem\_ptr}, \mathit{pval}, \_, pt)\rangle \longrightarrow \langle \cdot + \{pt'\}; \texttt{done}\,\texttt{Unit}, pt'\rangle$.
Prove: $\cdot; \cdot; \cdot; \cdot, pt' \vdash \texttt{done}\,\texttt{Unit}, pt' \Leftarrow \mathit{ret}$.

⟨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⟩17. Case: Ty_Action_Kill_Static.
Let: $pt = \mathit{mem\_ptr} \mapsto_\tau \_$.
Assume: 1. $\cdot; \cdot; \cdot; \cdot, pt \vdash \texttt{kill}\,(\texttt{static}\,\tau, \mathit{pval}_0, pt) \Rightarrow \Sigma\, \_{:}\texttt{unit}.\ \texttt{I}$.
2. $\langle \cdot + \{pt\}; \texttt{kill}\,(\texttt{static}\,\tau, \mathit{mem\_ptr}, pt)\rangle \longrightarrow \langle h; \texttt{done}\,\texttt{Unit}\rangle$.
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⟩18. Case: Ty_Memop_Rel_Binop.
Proof: Similar Ty_PE_Rel_Binop, except with Ty_TVal_{I,Phi,Comp} at the end.

⟨1⟩19. Case: Ty_Memop_IntFromPtr.
Let: $\mathit{ret} = \Sigma\, y{:}\texttt{integer}.\ y = \texttt{cast\_ptr\_to\_int}\,\mathit{mem\_ptr} \wedge \texttt{I}$.
Assume: 1. $\cdot; \cdot; \cdot; \cdot \vdash \texttt{intFromPtr}\,(\tau_1, \tau_2, \mathit{mem\_ptr}) \Rightarrow \mathit{ret}$.
2. $\langle \cdot; \texttt{intFromPtr}\,(\tau_1, \tau_2, \mathit{mem\_ptr})\rangle \longrightarrow \langle \cdot; \texttt{done}\,\mathit{mem\_int}\rangle$.
Prove: $\cdot; \cdot; \cdot; \cdot \vdash \texttt{done}\,\mathit{mem\_int} \Leftarrow \mathit{ret}$

⟨2⟩1. $\texttt{smt}\,(\cdot \Rightarrow \mathit{mem\_int} = \texttt{cast\_ptr\_to\_int}\,\mathit{mem\_ptr})$ by construction of $\mathit{mem\_int}$ (inversion on 2).

⟨2⟩2. $\cdot; \cdot; \cdot \vdash \mathit{mem\_int} \Rightarrow \texttt{integer}$ by Ty_PVal_Obj_Int and Ty_PVal_Obj.

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

⟨1⟩20. Case: Ty_Memop_PtrFromInt.
Proof: Similar to Ty_Memop_IntFromPtr, swapping base types `integer` and `loc`.

⟨1⟩21. Case: Ty_Memop_PtrValidForDeref.
Let: $pt = \mathit{mem\_ptr} \overset{\checkmark}{\mapsto}_\tau \_$.
$\mathit{ret} = \Sigma\, y{:}\texttt{bool}.\ y = \texttt{aligned}\,(\tau, \mathit{mem\_ptr}) \wedge pt \otimes \texttt{I}$.
Assume: 1. $\cdot; \cdot; \cdot; \mathcal{R} \vdash \texttt{ptrValidForDeref}\,(\tau, \mathit{mem\_ptr}, pt) \Rightarrow \mathit{ret}$.

2. $\langle \cdot + \{pt\}; \texttt{ptrValidForDeref}\,(\tau, \mathit{mem\_ptr}, pt)\rangle \longrightarrow \langle \cdot + \{pt\}; \texttt{done}\, \mathit{bool\_value}, pt\rangle$.

PROVE: $\cdot; \cdot; \cdot; \mathcal{R} \vdash \texttt{done}\, \mathit{bool\_value}, pt \Leftarrow \mathit{ret}$.

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

⟨2⟩2. $R = \cdot, pt$, by TY_RES_POINTSTO.

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

⟨2⟩4. $\cdot; \cdot; \cdot \vdash \mathit{bool\_value} \Rightarrow \texttt{bool}$ by TY_PVAL_{TRUE,FALSE}.

⟨2⟩5. By TY_TVAL_I, and then ⟨2⟩2 – ⟨2⟩4 with TY_TVAL_{RES,PHI,COMP} respectively, we are done.

⟨1⟩22. CASE: TY_MEMOP_PTRWELLALIGNED.

LET: $\mathit{ret} = \Sigma\, y{:}\texttt{bool}.\ y = \texttt{aligned}\,(\tau, \mathit{mem\_ptr}) \wedge \texttt{I}$.

ASSUME: 1. $\cdot; \cdot; \cdot; \cdot \vdash \texttt{ptrWellAligned}\,(\tau, \mathit{mem\_ptr}) \Rightarrow \mathit{ret}$.

2. $\langle \cdot; \texttt{ptrWellAligned}\,(\tau, \mathit{mem\_ptr})\rangle \longrightarrow \langle \cdot; \texttt{done}\, \mathit{bool\_value}\rangle$.

PROVE: $\cdot; \cdot; \cdot; \cdot \vdash \texttt{done}\, \mathit{bool\_value} \Rightarrow \mathit{ret}$.

⟨2⟩1. $\texttt{smt}\,(\cdot \Rightarrow \mathit{bool\_value} = \texttt{aligned}\,(\tau, \mathit{mem\_ptr}))$ by construction of $\mathit{bool\_value}$ (inversion on 2).

⟨2⟩2. $\cdot; \cdot; \cdot \vdash \mathit{bool\_value} \Rightarrow \texttt{bool}$ by TY_PVAL_{TRUE,FALSE}.

⟨2⟩3. By TY_TVAL_I and ⟨2⟩1 and ⟨2⟩2 with TY_TVAL_{PHI,COMP} respectively, we are done.

⟨1⟩23. CASE: TY_MEMOP_PTRARRAYSHIFT.

PROOF: Similiar to TY_PE_ARRAY_SHIFT, except with TY_TVAL_{I,PHI,COMP} at the end.

⟨1⟩24. CASE: TY_SEQ_E_CCALL.

ASSUME: 1. $\cdot; \cdot; \cdot; \mathcal{R} \vdash \texttt{ccall}\,(\tau, \mathit{pval}, \overline{\mathit{spine\_elem}_i}^{\,i}) \Rightarrow \sigma(\mathit{ret})$.

2. $\langle h; \texttt{ccall}\,(\tau, \mathit{pval}, \overline{\mathit{spine\_elem}_i}^{\,i})\rangle \longrightarrow \langle h; \sigma'(\mathit{texpr}){:}\sigma'(\mathit{ret})\rangle$.

PROVE: $\cdot; \cdot; \cdot; \mathcal{R} \vdash \sigma(\mathit{texpr}) \Leftarrow \sigma(\mathit{ret})$

⟨2⟩1. $\mathit{pval}{:}\mathit{arg} \equiv \overline{x_i}^{\,i} \mapsto \mathit{texpr} \in \texttt{Globals}$ by inversion (on either assumption).

⟨2⟩2. $\cdot; \cdot; \cdot; \mathcal{R} \vdash \overline{x_i = \mathit{spine\_elem}_i}^{\,i} :: \mathit{arg} \gg \sigma; \mathit{ret}$ by inversion on 1.

⟨2⟩3. $\sigma = \sigma'$ and $\mathit{ret} = \mathit{ret}'$ by induction on arg.

PROOF: Follows from lemma 5.1.

⟨2⟩4. LET: $\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R}'$ be the the type of substitution σ: $\cdot; \cdot; \cdot; \mathcal{R} \vdash (\sigma){:}(\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R}')$.

PROOF: Constructing such a substitution requires $\mathcal{C}; \mathcal{L}; \Phi \vdash \mathit{pval}_i \Rightarrow \beta_i$ for each $x_i{:}\beta_i \in \mathcal{C}$ or $x_i{:}\beta_i \in \mathcal{L}$ and $\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R}' \vdash \mathit{res\_term}_i \Leftarrow \mathit{res}_i$ for each $\mathit{res}_i \in \mathcal{R}'$ which can be deduced from ⟨2⟩2.

⟨2⟩5. $\mathcal{C}''; \mathcal{L}''; \Phi''; \mathcal{R}'' \vdash \mathit{texpr} \Leftarrow \mathit{ret}''$ where $\overline{x_i}^{\,i} :: \mathit{arg} \rightsquigarrow \mathcal{C}''; \mathcal{L}''; \Phi''; \mathcal{R}'' \mid \mathit{ret}''$ formalises the assumption that all global functions and labels are well-typed.

⟨2⟩6. $\mathcal{C} = \mathcal{C}''$, $\Phi = \Phi''$, $\mathcal{L} = \mathcal{L}''$, $\mathcal{R}' = \mathcal{R}''$ and $\mathit{ret} = \mathit{ret}''$.

PROOF: By induction on arg.

$\langle 2 \rangle 7$. Apply substitution lemma (2.2) to $\langle 2 \rangle 4$ and $\langle 2 \rangle 5$ to finish proof.

$\langle 1 \rangle 25$. CASE: TY_SEQ_E_PROC.
PROOF: Similar to TY_SEQ_E_CCALL.

$\langle 1 \rangle 26$. CASE: TY_IS_E_MEMOP.
PROOF: By induction on TY_MEMOP* cases.

$\langle 1 \rangle 27$. CASE: TY_IS_E_{NEG_}ACTION.
PROOF: By induction on TY_ACTION* cases.

$\langle 1 \rangle 28$. CASE: TY_SEQ_TE_LETP.
PROOF SKETCH: Only covering case $\langle \mathit{pexpr} \rangle \longrightarrow \langle \mathit{pexpr}' \rangle$ here.
See TY_SEQ_TE_LET for a more general version and proof for the remaining $\langle \mathit{pexpr} \rangle \longrightarrow \langle \mathit{tpexpr}{:}(y{:}\beta.\, \mathit{term}) \rangle$ case.
ASSUME: 1. $\cdot;\cdot;\cdot \vdash \texttt{let}\, \mathit{ident\_or\_pattern} = \mathit{pexpr}\, \texttt{in}\, \mathit{tpexpr} \Leftarrow y_2{:}\beta_2.\, \mathit{term}_2$.
2. $\langle \texttt{let}\, \mathit{ident\_or\_pattern} = \mathit{pexpr}\, \texttt{in}\, \mathit{tpexpr} \rangle \longrightarrow \langle \texttt{let}\, \mathit{ident\_or\_pattern} = \mathit{pexpr}'\, \texttt{in}\, \mathit{tpexpr} \rangle$.
PROVE: $\cdot;\cdot;\cdot \vdash \texttt{let}\, \mathit{ident\_or\_pattern} = \mathit{pexpr}'\, \texttt{in}\, \mathit{tpexpr} \Leftarrow y_2{:}\beta_2.\, \mathit{term}_2$.

$\langle 2 \rangle 1$. 1. $\cdot;\cdot;\cdot \vdash \mathit{pexpr} \Rightarrow y{:}\beta.\, \mathit{term}$.
2. $\mathit{ident\_or\_pattern}{:}\beta \rightsquigarrow \mathcal{C}_1\, \texttt{with}\, \mathit{term}_1$.
3. $\mathcal{C}_1;\cdot;\cdot, \mathit{term}_1/y, \cdot(\mathit{term}), \Phi_1; \mathcal{R} \vdash \mathit{texpr} \Leftarrow \mathit{ret}$.
PROOF: Invert assumption 1.

$\langle 2 \rangle 2$. $\langle \mathit{pexpr} \rangle \longrightarrow \langle \mathit{pexpr}' \rangle$.
PROOF: Invert assumption 2.

$\langle 2 \rangle 3$. $\cdot;\cdot;\cdot \vdash \mathit{pexpr}' \Rightarrow y{:}\beta.\, \mathit{term}$.
PROOF: By induction on $\langle 2 \rangle 1.1$ and $\langle 2 \rangle 2$.

$\langle 2 \rangle 4$. $\cdot;\cdot;\cdot \vdash \texttt{let}\, \mathit{ident\_or\_pattern} = \mathit{pexpr}'\, \texttt{in}\, \mathit{tpexpr} \Leftarrow y_2{:}\beta_2.\, \mathit{term}_2$.
PROOF: By TY_SEQ_TE_LETP using $\langle 2 \rangle 1.2,3$ and $\langle 2 \rangle 3$.

$\langle 1 \rangle 29$. CASE: TY_SEQ_TE_LETPT.
PROOF: See TY_SEQ_TE_LETT for a more general case and proof.

$\langle 1 \rangle 30$. CASE: TY_SEQ_TE_LET.
ASSUME: 1. $\cdot;\cdot;\cdot;\mathcal{R}', \mathcal{R} \vdash \texttt{let}\, \overline{\mathit{ret\_pattern}_i}^{\,i} = \mathit{seq\_expr}\, \texttt{in}\, \mathit{texpr}_2 \Leftarrow \mathit{ret}_2$.
2. $\langle h; \texttt{let}\, \overline{\mathit{ret\_pattern}_i}^{\,i} = \mathit{seq\_expr}\, \texttt{in}\, \mathit{texpr}_2 \rangle \longrightarrow \langle h; \texttt{let}\, \overline{\mathit{ret\_pattern}_i}^{\,i}{:}\mathit{ret}_1' = \mathit{texpr}_1\, \texttt{in}\, \mathit{texpr}_2 \rangle$.
PROVE: $\cdot;\cdot;\cdot;\mathcal{R}', \mathcal{R} \vdash \texttt{let}\, \overline{\mathit{ret\_pattern}_i}^{\,i}{:}\mathit{ret}_1 = \mathit{texpr}_1\, \texttt{in}\, \mathit{texpr}_2 \Leftarrow \mathit{ret}_2$.

$\langle 2 \rangle 1$. 1. $\cdot;\cdot;\cdot;\mathcal{R}' \vdash \mathit{seq\_expr} \Rightarrow \mathit{ret}_1$.
2. $\overline{\mathit{ret\_pattern}_i}^{\,i}{:}\mathit{ret}_1 \rightsquigarrow \mathcal{C}_1; \mathcal{L}_1; \Phi_1; \mathcal{R}_1$.
3. $\mathcal{C}_1; \mathcal{L}_1; \Phi_1; \mathcal{R}, \mathcal{R}_1 \vdash \mathit{texpr} \Leftarrow \mathit{ret}_2$.
PROOF: By inversion on 1.

$\langle 2 \rangle 2$. $\langle h; \mathit{seq\_expr} \rangle \longrightarrow \langle h; \mathit{texpr}_1{:}\mathit{ret}_1' \rangle$.
PROOF: By inversion on 2.

$\langle 2 \rangle 3$. $\cdot;\cdot;\cdot;\mathcal{R}' \vdash \mathit{texpr}_1 \Leftarrow \mathit{ret}_1$.
PROOF: By induction on $\langle 2 \rangle 1.1$ and $\langle 2 \rangle 2$.

⟨2⟩4. $ret_1 = ret_1'$.
PROOF: By cases TY_SEQ_E_{CCALL,PCALL}.

⟨2⟩5. By TY_SEQ_TE_LET with ⟨2⟩1.2,3 and ⟨2⟩3, we are done.

⟨1⟩31. CASE: TY_SEQ_TE_LETT.
NOTE: $h : \mathcal{R}', \mathcal{R}$ and $h : \mathcal{R}_1, \mathcal{R}$.

ASSUME: 1. $\cdot; \cdot; \cdot; \mathcal{R}', \mathcal{R} \vdash \texttt{let}\, \overline{ret\_pattern_i}^{\,i} : ret_1 = \texttt{done}\, \overline{spine\_elem_i}^{\,i}\, \texttt{in}\, texpr_2 \Leftarrow ret_2$.
2. $\langle h; \texttt{let}\, \overline{ret\_pattern_i}^{\,i} : ret_1 = \texttt{done}\, \overline{spine\_elem_i}^{\,i}\, \texttt{in}\, texpr \rangle \longrightarrow \langle h; \sigma(texpr_2) \rangle$.
PROVE: $\cdot; \cdot; \cdot; \mathcal{R}', \mathcal{R} \vdash \sigma(texpr_2) \Leftarrow \sigma(ret_2)$.

⟨2⟩1. 1. $\cdot; \cdot; \cdot; \mathcal{R}' \vdash \texttt{done}\, \overline{spine\_elem_i}^{\,i} \Leftarrow ret_1$.
2. $\overline{ret\_pattern_i}^{\,i} : ret_1 \rightsquigarrow \mathcal{C}_1; \mathcal{L}_1; \Phi_1; \mathcal{R}_1$.
3. $\mathcal{C}_1; \mathcal{L}_1; \Phi_1; \mathcal{R}_1, \mathcal{R} \vdash texpr_2 \Leftarrow ret_2$.
PROOF: By inversion on 1.

⟨2⟩2. $\overline{ret\_pattern_i = spine\_elem_i}^{\,i} \rightsquigarrow \sigma$.
PROOF: By inversion on 2.

⟨2⟩3. $\cdot; \cdot; \cdot; \mathcal{R}' \vdash (\sigma) : (\mathcal{C}_1; \mathcal{L}_1; \Phi_1; \mathcal{R}_1)$.
PROOF: By ⟨2⟩1.1,2 and ⟨2⟩2 using lemma 5.3.

⟨2⟩4. By ⟨2⟩1.3 and ⟨2⟩3 and lemma 2.4, we are done.

⟨1⟩32. CASE: TY_SEQ_TE_LETT.
ASSUME: 1. $\cdot; \cdot; \cdot; \mathcal{R}', \mathcal{R} \vdash \texttt{let}\, \overline{ret\_pattern_i}^{\,i} : ret_1 = texpr_1\, \texttt{in}\, texpr_2 \Leftarrow ret_2$.
2. $\langle h; \texttt{let}\, \overline{ret\_pattern_i}^{\,i} : ret = texpr_1\, \texttt{in}\, texpr_2 \rangle \longrightarrow \langle h'; \texttt{let}\, \overline{ret\_pattern_i}^{\,i} : ret = texpr_1'\, \texttt{in}\, texpr_2 \rangle$.
PROVE: $\cdot; \cdot; \cdot; \mathcal{R}'', \mathcal{R} \vdash \texttt{let}\, \overline{ret\_pattern_i}^{\,i} : ret_1 = texpr_1'\, \texttt{in}\, texpr_2 \Leftarrow ret_2$.

⟨2⟩1. 1. $\cdot; \cdot; \cdot; \mathcal{R}' \vdash texpr_1 \Leftarrow ret_1$.
2. $\overline{ret\_pattern_i}^{\,i} : ret_1 \rightsquigarrow \mathcal{C}_1; \mathcal{L}_1; \Phi_1; \mathcal{R}_1$.
3. $\mathcal{C}_1; \mathcal{L}_1; \Phi_1; \mathcal{R}_1, \mathcal{R} \vdash texpr_2 \Leftarrow ret_2$.
PROOF: By inversion on 1.

⟨2⟩2. $\langle h; texpr_1 \rangle \longrightarrow \langle h'; texpr_1' \rangle$.
PROOF: By inversion on 2.

⟨2⟩3. $\cdot; \cdot; \cdot; \mathcal{R}'' \vdash texpr_1' \Leftarrow ret_1$.
PROOF: By induction on ⟨2⟩1.1 and ⟨2⟩2.

⟨2⟩4. By ⟨2⟩3, ⟨1⟩32.2,3 using TY_SEQ_TE_LETT, we are done.

⟨1⟩33. CASE: TY_SEQ_TE_CASE.
ASSUME: 1. $\cdot; \cdot; \cdot; \mathcal{R} \vdash \texttt{case}\, pval\, \texttt{of}\, \overline{\mid pattern_i \Rightarrow texpr_i}^{\,i}\, \texttt{end} \Leftarrow ret$.
2. $\langle h; \texttt{case}\, pval\, \texttt{of}\, \overline{\mid pattern_i \Rightarrow texpr_i}^{\,i}\, \texttt{end} \rangle \longrightarrow \langle h; \sigma_j(texpr_j) \rangle$.
PROVE: $\cdot; \cdot; \cdot; \mathcal{R} \vdash \sigma_j(texpr_j) \Leftarrow ret$.

⟨2⟩1. 1. $\cdot; \cdot; \cdot \vdash pval \Rightarrow \beta_1$.
2. $\overline{pattern_i : \beta_1 \rightsquigarrow \mathcal{C}_i\, \texttt{with}\, term_i}^{\,i}$.
3. $\overline{\mathcal{C}_i; \cdot; \cdot, term_i = pval; \mathcal{R} \vdash texpr_i \Leftarrow ret}^{\,i}$.
PROOF: By inversion on 1.

⟨2⟩2. 1. $pattern_j = pval \rightsquigarrow \sigma_j$.
2. $\forall\, i < j.\ \texttt{not}\,(pattern_i = pval \rightsquigarrow \sigma_i)$.
PROOF: By inversion on 2.

⟨2⟩3. $\cdot;\cdot;\cdot;\cdot \vdash (\sigma_j){:}(\mathcal{C}_i;\cdot;\cdot;\cdot)$.
PROOF: By lemma 5.3.

⟨2⟩4. $\cdot;\cdot;\cdot;\mathcal{R} \vdash (\sigma_j){:}(\mathcal{C}_i;\cdot;\cdot, term_j = pval_j;\mathcal{R})$.
PROOF: By ⟨2⟩3, TY_SUBS_CONS_PHI and TY_SUBS_CONS_RES*.

⟨2⟩5. By ⟨2⟩1.3 and 2.2, we are done.

⟨1⟩34. CASE: TY_SEQ_TE_IF.
Only covering `True` case, `False` is almost identical.
ASSUME: 1. $\cdot;\cdot;\cdot;\mathcal{R} \vdash \texttt{if}\,\texttt{True}\,\texttt{then}\, texpr_1\,\texttt{else}\, texpr_2 \Leftarrow ret$.
2. $\langle h; \texttt{if}\,\texttt{True}\,\texttt{then}\, texpr_1\,\texttt{else}\, texpr_2\rangle \longrightarrow \langle h; texpr_1\rangle$.
PROVE: $\cdot;\cdot;\cdot;\mathcal{R} \vdash texpr_1 \Leftarrow ret$.
PROOF: Invert 1, note $\cdot;\cdot;\cdot;\mathcal{R} \vdash (\mathrm{id}){:}(\cdot;\cdot;\cdot, \texttt{true} = \texttt{true};\mathcal{R})$ and then apply substitution lemma (2.2).

⟨1⟩35. CASE: TY_SEQ_TE_RUN.
PROOF SKETCH: Similar to case TY_SEQ_E_{CCALL,PCALL}.

⟨1⟩36. CASE: TY_SEQ_TE_BOUND.
PROOF: By inversion on the typing rule.

⟨1⟩37. CASE: TY_IS_TE_LETS.
PROOF SKETCH: Similar to TY_SEQ_TE_LETT.

6 Typing Judgements

object_value_jtype	::=	
	|	$\mathcal{C}; \mathcal{L}; \Phi \vdash \mathit{object\_value} \Rightarrow \texttt{obj}\,\beta$
pval_jtype	::=	
	|	$\mathcal{C}; \mathcal{L}; \Phi \vdash \mathit{pval} \Rightarrow \beta$
res_jtype	::=	
	|	$\Phi \vdash \mathit{res} \equiv \mathit{res}'$
	|	$\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \mathit{res\_term} \Leftarrow \mathit{res}$
spine_jtype	::=	
	|	$\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \overline{x_i = \mathit{spine\_elem}_i}^{\,i} :: \mathit{arg} \gg \sigma; \mathit{ret}$
pexpr_jtype	::=	
	|	$\mathcal{C}; \mathcal{L}; \Phi \vdash \mathit{pexpr} \Rightarrow \mathit{ident}{:}\beta.\, \mathit{term}$
tpval_jtype	::=	
	|	$\mathcal{C}; \mathcal{L}; \Phi \vdash \mathit{tpval} \Leftarrow \mathit{ident}{:}\beta.\, \mathit{term}$
tpexpr_jtype	::=	
	|	$\mathcal{C}; \mathcal{L}; \Phi \vdash \mathit{tpexpr} \Leftarrow \mathit{ident}{:}\beta.\, \mathit{term}$
action_jtype	::=	
	|	$\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \mathit{mem\_action} \Rightarrow \mathit{ret}$
memop_jtype	::=	
	|	$\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \mathit{mem\_op} \Rightarrow \mathit{ret}$
seq_expr_jtype	::=	
	|	$\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \mathit{seq\_expr} \Rightarrow \mathit{ret}$
is_expr_jtype	::=	
	|	$\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \mathit{is\_expr} \Rightarrow \mathit{ret}$
tval_jtype	::=	
	|	$\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \mathit{tval} \Leftarrow \mathit{ret}$
texpr_jtype	::=	
	|	$\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \mathit{seq\_texpr} \Leftarrow \mathit{ret}$
	|	$\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \mathit{is\_texpr} \Leftarrow \mathit{ret}$
	|	$\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \mathit{texpr} \Leftarrow \mathit{ret}$

7 Opsem Judgements

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

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