Explicit CN Soundness Proof

Dhruv Makwana

June 28, 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 $\exists \mathit{pval}.\ \mathcal{C}; \mathcal{L}; \Phi \vdash \mathit{pval} \Rightarrow \beta$ by Ty_Subs_Cons_{Comp,Log}.

⟨2⟩3. Since $\mathit{pval} = \sigma(x)$, 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 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 $\mathit{pt} = \_ \overset{\checkmark}{\mapsto}_\tau \mathit{pval}$, 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

ASSUME: 1. $\overline{\mathit{ret\_pattern}_i = \mathit{spine\_elem}_i}^{\,i} \rightsquigarrow \sigma$.
2. $\cdot;\cdot;\cdot;\mathcal{R} \vdash \mathtt{done}\ \overline{\mathit{spine\_elem}_i}^{\,i} \Leftarrow \mathit{ret}$.

3. $\overline{\mathit{ret\_pattern}_i}^{\,i} : \mathit{ret} \rightsquigarrow \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R}'$.

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

PROOF SKETCH: Induction on *ret*; inversion on 1, 2 and 3 to conclude the empty substitution is well-typed and each addition preserves that property.

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: $\mathit{term} = \mathit{mem\_ptr} +_{\text{ptr}} (\mathit{mem\_int} \times \text{size\_of}(\tau))$.
ASSUME: 1. $\cdot;\cdot;\cdot \vdash \texttt{array\_shift}\,(\mathit{mem\_ptr}, \tau, \mathit{mem\_int}) \Rightarrow y{:}\texttt{loc}.\ y = \mathit{term}$.
2. $\langle \texttt{array\_shift}\,(\mathit{mem\_ptr}, \tau, \mathit{mem\_int})\rangle \longrightarrow \langle \mathit{mem\_ptr}'\rangle$.
PROVE: $\cdot;\cdot;\cdot \vdash \mathit{mem\_ptr}' \Rightarrow y{:}\texttt{loc}.\ y = \mathit{term}$.
PROOF: By TY_PVAL_OBJ_INT, TY_PVAL_OBJ, TY_PE_VAL and construction of $\mathit{mem\_ptr}'$ (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. $\cdot;\cdot;\cdot \vdash \texttt{not}\,(\mathit{bool\_value}) \Rightarrow y{:}\texttt{bool}.\ y = \neg\,\mathit{bool\_value}$.
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: $\cdot;\cdot;\cdot \vdash \mathit{bool\_value}' \Rightarrow y{:}\texttt{bool}.\ y = \neg\,\mathit{bool\_value}$.
PROOF: By TY_PVAL_{TRUE,FALSE}, TY_PE_VAL and 2.

⟨1⟩4. CASE: TY_PE_ARITH_BINOP.
LET: $\mathit{term} = \mathit{mem\_int}_1\ \mathit{binop}_{\mathit{arith}}\ \mathit{mem\_int}_2$.
ASSUME: 1. $\cdot;\cdot;\cdot \vdash \mathit{mem\_int}_1\ \mathit{binop}_{\mathit{arith}}\ \mathit{mem\_int}_2 \Rightarrow y{:}\texttt{integer}.\ y = \mathit{term}$.
2. $\langle \mathit{mem\_int}_1\ \mathit{binop}_{\mathit{arith}}\ \mathit{mem\_int}_2\rangle \longrightarrow \langle \mathit{mem\_int}\rangle$.
PROVE: $\cdot;\cdot;\cdot \vdash \mathit{mem\_int} \Rightarrow y{:}\texttt{integer}.\ y = \mathit{term}$.
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. $\cdot;\cdot;\cdot \vdash \texttt{assert\_undef}\,(\texttt{True},\ \mathit{UB\_name}) \Rightarrow y{:}\texttt{unit}.\ y = \texttt{unit}$.
2. $\langle \texttt{assert\_undef}\,(\texttt{True},\ \mathit{UB\_name})\rangle \longrightarrow \langle \texttt{Unit}\rangle$.

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: $\mathit{term} = \texttt{if}\, \mathit{bool\_value}\, \texttt{then}\, 1\, \texttt{else}\, 0$.
ASSUME: 1. $\cdot;\cdot;\cdot \vdash \texttt{bool\_to\_integer}\,(\mathit{bool\_value}) \Rightarrow y{:}\texttt{integer}.\ y = \mathit{term}$.
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: $\cdot;\cdot;\cdot \vdash \mathit{mem\_int} \Rightarrow y{:}\texttt{integer}.\ y = \mathit{term}$
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 $\mathit{abbrev}_2 \leq \mathrm{max\_int}_\tau$, 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: $\mathit{pt} = \mathit{mem\_ptr} \overset{\times}{\mapsto}_\tau \mathit{pval}$.
$\mathit{term} = \texttt{representable}\,(\tau*, y_p) \wedge \texttt{alignedI}\,(\mathit{mem\_int}, y_p)$.
$\mathit{ret} = \Sigma\, y_p{:}\texttt{loc}.\ \mathit{term} \wedge \exists\, y{:}\beta_\tau.\ y_p \overset{\times}{\mapsto}_\tau y \otimes \texttt{I}$.
ASSUME: 1. $\cdot;\cdot;\cdot;\cdot \vdash \texttt{create}\,(\mathit{mem\_int}, \tau) \Rightarrow \mathit{ret}$.
2. $\langle \cdot; \texttt{create}\,(\mathit{mem\_int}, \tau)\rangle \longrightarrow \langle \cdot + \{\mathit{pt}\}; \texttt{done}\, \mathit{mem\_ptr}, \mathit{pval}, \mathit{pt}\rangle$.
PROVE: $\cdot;\cdot;\cdot;\cdot, \_{:}\mathit{pt} \vdash \texttt{done}\, \mathit{mem\_ptr}, \mathit{pval}, \mathit{pt} \Leftarrow \mathit{ret}$.

⟨2⟩1. $\cdot;\cdot;\cdot \vdash \mathit{mem\_ptr} \Rightarrow \texttt{loc}$ by TY_PVAL_OBJ_INT and TY_PVAL_OBJ.

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

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

⟨2⟩4. $\cdot;\cdot;\cdot;\cdot, \_{:}\mathit{pt} \vdash \mathit{pt} \Leftarrow \mathit{pt}$ 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: $\mathit{pt} = \mathit{mem\_ptr} \overset{\checkmark}{\mapsto}_\tau \mathit{pval}$.
$\mathit{ret} = \Sigma\, y{:}\beta_\tau.\ y = \mathit{pval} \wedge \mathit{pt} \otimes \texttt{I}$.
ASSUME: 1. $\cdot;\cdot;\cdot;\cdot, \_{:}\mathit{pt} \vdash \texttt{load}\,(\tau, \mathit{mem\_ptr}, \_, \mathit{pt}) \Rightarrow \mathit{ret}$.
2. $\langle \cdot + \{\mathit{pt}\}; \texttt{load}\,(\tau, \mathit{mem\_ptr}, \_, \mathit{pt})\rangle \longrightarrow \langle \cdot + \{\mathit{pt}\}; \texttt{done}\, \mathit{pval}, \mathit{pt}\rangle$.
PROVE: $\cdot;\cdot;\cdot;\cdot, \_{:}\mathit{pt} \vdash \texttt{done}\, \mathit{pval}, \mathit{pt} \Leftarrow \mathit{ret}$

⟨2⟩1. $\cdot;\cdot;\cdot;\mathcal{R} \vdash pt \Leftarrow pt$, 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: $pt = \mathit{mem\_ptr} \overset{\checkmark}{\mapsto}_\tau \_$.

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

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

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

⟨2⟩4. $\cdot;\cdot;\cdot \vdash bool\_value \Rightarrow \texttt{bool}$ 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: $ret = \Sigma\, y{:}\texttt{bool}.\ y = \texttt{aligned}\,(\tau, mem\_ptr) \wedge \texttt{I}.$

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

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

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

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

⟨2⟩2. $\cdot;\cdot;\cdot \vdash bool\_value \Rightarrow \texttt{bool}$ 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. $\cdot;\cdot;\cdot;\mathcal{R} \vdash \texttt{ccall}\,(\tau, pval, \overline{spine\_elem_i}^{\,i}) \Rightarrow \sigma(ret).$

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

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

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

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

⟨2⟩3. $\sigma = \sigma'$ and $ret = ret'$ 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 $\mathcal{C};\mathcal{L};\Phi \vdash pval_i \Rightarrow \beta_i$ for each $x_i{:}\beta_i \in \mathcal{C}$ or $x_i{:}\beta_i \in \mathcal{L}$ and $\mathcal{C};\mathcal{L};\Phi;\mathcal{R}' \vdash res\_term_i \Leftarrow res_i$ for each $\_ : res_i \in \mathcal{R}'$ which can be deduced from ⟨2⟩2.

⟨2⟩5. $\mathcal{C}'';\mathcal{L}'';\Phi'';\mathcal{R}'' \vdash texpr \Leftarrow ret''$ where $\overline{x_i}^{\,i} :: arg \rightsquigarrow \mathcal{C}'';\mathcal{L}'';\Phi'';\mathcal{R}'' \mid ret''$ 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 $ret = ret''$.

PROOF: By induction on *arg*.

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

⟨1⟩25. CASE: TY_SEQ_E_PROC.
PROOF: Similar to TY_SEQ_E_CCALL.

⟨1⟩26. CASE: TY_IS_E_MEMO.
PROOF: By induction on TY_MEMOP* cases.

⟨1⟩27. CASE: TY_IS_E_{NEG_}ACTION.
PROOF: By induction on TY_ACTION* cases.

⟨1⟩28. CASE: TY_SEQ_TE_LETP.
PROOF SKETCH: Only covering case $\langle pexpr\rangle \longrightarrow \langle pexpr'\rangle$ here.
See TY_SEQ_TE_LET for a more general version and proof for the remaining $\langle pexpr\rangle \longrightarrow \langle tpexpr{:}(y{:}\beta.\ term)\rangle$ case.
ASSUME: 1. $\cdot;\cdot;\cdot \vdash \texttt{let}\, ident\_or\_pattern = pexpr\, \texttt{in}\, tpexpr \Leftarrow y_2{:}\beta_2.\ term_2$.
2. $\langle \texttt{let}\, ident\_or\_pattern = pexpr\, \texttt{in}\, tpexpr\rangle \longrightarrow \langle \texttt{let}\, ident\_or\_pattern = pexpr'\, \texttt{in}\, tpexpr\rangle$.
PROVE: $\cdot;\cdot;\cdot \vdash \texttt{let}\, ident\_or\_pattern = pexpr'\, \texttt{in}\, tpexpr \Leftarrow y_2{:}\beta_2.\ term_2$.

⟨2⟩1. 1. $\cdot;\cdot;\cdot \vdash pexpr \Rightarrow y{:}\beta.\ term$.
2. $y\, \texttt{as}\, ident\_or\_pattern{:}\beta \rightsquigarrow \mathcal{C}_1; \Phi_1$.
3. $\mathcal{C}_1; \cdot, y{:}\beta; \cdot, term, \Phi_1; \mathcal{R} \vdash texpr \Leftarrow ret$.
PROOF: Invert assumption 1.

⟨2⟩2. $\langle pexpr\rangle \longrightarrow \langle pexpr'\rangle$.
PROOF: Invert assumption 2.

⟨2⟩3. $\cdot;\cdot;\cdot \vdash pexpr' \Rightarrow y{:}\beta.\ term$.
PROOF: By induction on ⟨2⟩1.1 and ⟨2⟩2.

⟨2⟩4. $\cdot;\cdot;\cdot \vdash \texttt{let}\, ident\_or\_pattern = pexpr'\, \texttt{in}\, tpexpr \Leftarrow y_2{:}\beta_2.\ term_2$.
PROOF: By TY_SEQ_TE_LETP using ⟨2⟩1.2,3 and ⟨2⟩3.

⟨1⟩29. CASE: TY_SEQ_TE_LETPT.
PROOF: See TY_SEQ_TE_LETT for a more general case and proof.

⟨1⟩30. CASE: TY_SEQ_TE_LET.
LET:
ASSUME: 1. $\cdot;\cdot;\cdot;\mathcal{R}', \mathcal{R} \vdash \texttt{let}\, \overline{ret\_pattern_i}^{\,i} = seq\_expr\, \texttt{in}\, texpr \Leftarrow ret_2$.
2. $\langle h; \texttt{let}\, \overline{ret\_pattern_i}^{\,i} = seq\_expr\, \texttt{in}\, texpr_2\rangle \longrightarrow \langle h; \texttt{let}\, \overline{ret\_pattern_i}^{\,i}{:}ret_1 = texpr_1\, \texttt{in}\, texpr_2\rangle$.
PROVE: $\cdot;\cdot;\cdot;\mathcal{R}', \mathcal{R} \vdash \texttt{let}\, \overline{ret\_pattern_i}^{\,i}{:}ret_1 = texpr_1\, \texttt{in}\, texpr_2 \Leftarrow ret_2$.

⟨1⟩31. CASE: TY_SEQ_TE_LETT.
LET:
ASSUME: 1. .
2. .
PROVE: .

6 Typing Judgements

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

7 Opsem Judgements

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

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