Explicit CN Soundness Proof

Dhruv Makwana

July 6, 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: $\mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash \sigma(J)$.

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

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

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

$\langle 2\rangle$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 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

3.1 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*.

3.2 Progress Statement and Proof

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

$\langle 1 \rangle 1.$ CASE: TY_PVAL_OBJ*, TY_PVAL*, TY_PE_VAL, TY_TPVAL*, TY_TVAL*, TY_SEQ_TE_TVAL.
PROOF: All these judgements/rules give types to syntactic values; and there are no operational rules corresponding to them (see Section 7).

$\langle 1 \rangle 2.$ CASE: TY_PE_ARRAY_SHIFT.
PROOF: By inversion on $\cdot; \cdot; \cdot \vdash \mathit{pval}_1 \Rightarrow \texttt{loc}$, pval_1 must be a *mem_ptr* (TY_PVAL_OBJ_PTR). Similarly pval_2 must be a *mem_int*, so rule OP_PE_PE_ARRAYSHIFT applies.

$\langle 1 \rangle 3.$ CASE: TY_PE_MEMBER_SHIFT.
PROOF: *pval* must be a *mem_ptr* so OP_PE_PE_MEMBERSHIFT.

$\langle 1 \rangle 4.$ CASE: TY_PE_NOT.
PROOF: *pval* must be a *bool_value* so OP_PE_PE_NOT_{TRUE,FALSE}.

$\langle 1 \rangle 5.$ CASE: TY_PE_{ARITH,REL}_BINOP.
PROOF: pval_1 and pval_2 must be *mem_int*s so OP_PE_PE_{ARITH,REL}_BINOP respectively.

$\langle 1 \rangle 6.$ CASE: TY_PE_BOOL_BINOP.
PROOF: pval_1 and pval_2 must be *bool_value*s so OP_PE_PE_BOOL_BINOP.

$\langle 1 \rangle 7.$ CASE: TY_PE_CALL.
PROOF: By inversion we have $\mathit{name}{:}\mathit{pure\_arg} \equiv \overline{x_i}^{\,i} \mapsto \mathit{tpexpr} \in \texttt{Globals}$ and $\cdot; \cdot; \cdot; \cdot \vdash \overline{x_i = \mathit{pval}_i}^{\,i} :: \mathit{pure\_arg} \gg \sigma; \Sigma\, y{:}\beta.\ \mathit{term} \wedge \texttt{I}$, with the latter implying $\overline{x_i = \mathit{pval}_i}^{\,i} :: \mathit{pure\_arg} \gg \sigma; \Sigma\, y{:}\beta.\ \mathit{term} \wedge \texttt{I}$ (lemma 3.1. Thus it can step with OP_PE_TPE_CALL.

⟨1⟩8. CASE: TY_PE_ASSERT_UNDEF.
PROOF: *pval* must be a *bool_value* and smt $(\Phi \Rightarrow pval)$. If it is `False`, then by the latter, we have an inconsistent constraints context, meaning the code is unreachable. If it is `True`, we may step with OP_PE_PE_ASSERT_UNDEF.

⟨1⟩9. CASE: TY_PE_BOOL_TO_INTEGER.
PROOF: *pval* must be a *bool_value* and so OP_PE_PE_BOOL_TO_INTEGER_{TRUE,FALSE}.

⟨1⟩10. CASE: TY_PE_WRAPI.
PROOF: *pval* must be a *mem_int* and so OP_PE_PE_WRAPI.

⟨1⟩11. CASE: TY_TPE_{IF,LET,LETT,CASE}.
PROOF: See TY_SEQ_TE_{IF,LET,LETT,CASE} cases for more general cases and proofs.

⟨1⟩12. CASE: TY_ACTION_CREATE.
PROOF: *pval* must be a *mem_int* and h must be $\cdot$, so OP_ACTION_TVAL_CREATE (*mem_ptr* and $pval{:}\beta_\tau$ are free in the premises and so can be constructed to satisfy the requirements).

⟨1⟩13. CASE: TY_ACTION_LOAD.
PROOF: $pval_0$ must be a *mem_ptr* and $h = \cdot + \{pval_1 \overset{\checkmark}{\mapsto}_\tau pval_2\}$, so OP_ACTION_TVAL_LOAD.

⟨1⟩14. CASE: TY_ACTION_STORE.
PROOF: $pval_0$ and $pval_2$ must be the same *mem_ptr*, so OP_ACTION_TVAL_STORE.

⟨1⟩15. CASE: TY_ACTION_KILL_STATIC.
PROOF: $pval_0$ and $pval_1$ must be the same *mem_ptr*, so OP_ACTION_TVAL_KILL_STATIC.

⟨1⟩16. CASE: TY_MEMOP_REL_BINOP.
PROOF: Similar to TY_PE_{ARITH,REL}_BINOP.

⟨1⟩17. CASE: TY_MEMOP_INTFROMPTR.
PROOF: *pval* must be a *mem_ptr* so OP_MEMOP_TVAL_REL_INTFROMPTR.

⟨1⟩18. CASE: TY_MEMOP_PTRFROMINT.
PROOF: *pval* must be a *mem_int* so OP_MEMOP_TVAL_REL_PTRFROMINT.

⟨1⟩19. CASE: TY_MEMOP_PTRVALIDFORDEREF.
PROOF: *pval* must be a *mem_ptr* and h must be $\cdot + \{mem_ptr \overset{\checkmark}{\mapsto}_\tau _\}$so it can take a step with OP_MEMOP_TVAL_REL_PTRVALIDFORDEREF.

⟨1⟩20. CASE: TY_MEMOP_PTRWELLALIGNED.
PROOF: *pval* must be a *mem_ptr* and so OP_MEMOP_TVAL_PTRWELLALIGNED.

⟨1⟩21. CASE: TY_MEMOP_PTRARRAYSHIFT.
PROOF: $pval_1$ must be a *mem_ptr* and $pval_2$ must be a *mem_int* and so OP_MEMOP_TVAL_PTRARRAYS

⟨1⟩22. CASE: TY_SEQ_E_CCALL.
PROOF: By inversion we have *pval* must be a *mem_ptr*, and $mem\_ptr{:}arg \equiv \overline{x_i}^{\,i} \mapsto$

$texpr \in \texttt{Globals}$ and $\cdot;\cdot;\cdot;\cdot \vdash \overline{x_i = spine\_elem_i}^{\,i} :: arg \gg \sigma; ret$, with the latter implying $\overline{x_i = spine\_elem_i}^{\,i} :: arg \gg \sigma; ret$ (lemma 3.1. Thus it can step with OP_SE_TE_CCALL.

⟨1⟩23. CASE: TY_SEQ_E_PROC.
PROOF: Similar to TY_SEQ_E_CCALL.

⟨1⟩24. CASE: TY_IS_E_MEMOP.
PROOF: By induction, if *mem_op* is unreachable, then the whole expression is so. Memops are not values. Only stepping cases applies, so OP_ISE_ISE_MEMOP.

⟨1⟩25. CASE: TY_IS_E_{NEG_}ACTION.
PROOF: By induction, if *mem_action* is unreachable, then the whole expression is so. Actions are not values. Only stepping case applies, so OP_ISE_ISE_{NEG_}ACTION.

⟨1⟩26. CASE: TY_SEQ_TE_{LETP,LETPT}.
PROOF: See TY_SEQ_TE_{LET,LETT} for more general cases and proofs.

⟨1⟩27. CASE: TY_SEQ_TE_LET.
PROOF: By induction, since *seq_expr* is not value, if it is unreachable, the whole expression is so. If it takes a step, then OP_STE_TE_LET_LETT.

⟨1⟩28. CASE: TY_SEQ_TE_LETT.
PROOF: By induction, if *texpr* is unreachable, so is the whole expression. If if it a *tval* then OP_STE_TE_LETT_SUB. If if takes a step, then OP_STE_TE_LETT_LETT.

⟨1⟩29. CASE: TY_SEQ_TE_CASE.
PROOF: We have to show that assuming the case-expression is well-typed, then there is at least one pattern against which *pval* will match.

⟨1⟩30. CASE: TY_SEQ_TE_IF.
PROOF: *pval* must be a *bool_value* and so OP_STE_TE_IF_{TRUE,FALSE}.

⟨1⟩31. CASE: TY_SEQ_TE_RUN.
PROOF: Similar to TY_SEQ_E_CCALL.

⟨1⟩32. CASE: TY_SEQ_TE_BOUND.
PROOF: By OP_STE_TE_BOUND.

⟨1⟩33. CASE: TY_IS_TE_LETS.
PROOF: Similar to TY_SEQ_TE_LETT.

4 Framing

If $\langle h; e\rangle \longrightarrow \langle h'; e'\rangle$ and $\exists h_1, h_2.\ \text{disjoint}(h_1, h_2) \wedge h = h_1 + h_2 \wedge \langle h_1; e\rangle \longrightarrow \langle h_1'; e'\rangle$ then $h' = h_1' + h_2$.

ASSUME: 1. $\langle h; e\rangle \longrightarrow \langle h'; e'\rangle$,
2. $h = h_1 + h_2$ where h_1, h_2 disjoint,
3. and $\langle h_1; e\rangle \longrightarrow \langle h_1'; e'\rangle$.
PROVE: $h' = h_1' + h_2$.

PROOF SKETCH:Induction over the operational rules. Only covering ones which modify the heap; rest are trivially true.

⟨1⟩1. CASE: OP_ACTION_TVAL_CREATE
PROOF: Because *mem_ptr* is fresh.

⟨1⟩2. CASE: OP_ACTION_TVAL_{STORE,KILL}.
PROOF: By assumption of disjointness, $mem\_ptr \in h_1$ implies $mem\_ptr \notin h_2$.

5 Type Preservation

5.1 Pointed-to values have type β_τ

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

PROOF SKETCH: Induction over the typing judgements. Only TY_ACTION_STORE create such permissions, and its premise $\mathcal{C}; \mathcal{L}; \Phi \vdash pval_1 \Rightarrow \beta_\tau$ ensures the desired property. TY_ACTION_LOAD simply preserves the property.

5.2 Deconstructing a pattern leads to a well-typed substitution

First, computational part.

ASSUME: 1. $\cdot; \cdot; \cdot \vdash pval \Rightarrow \beta_1$.
2. $ident\_or\_pattern{:}\beta \leadsto \mathcal{C}\,\texttt{with}\,term$.
3. $ident\_or\_pattern = pval \leadsto \sigma$.
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. $res\_pattern{:}res \leadsto \mathcal{L}; \Phi; \mathcal{R}'$.
3. $res\_pattern = res\_term \leadsto \sigma$.
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}.

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

⟨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. $\cdot;\cdot;\cdot \vdash \texttt{not}\,(\mathit{bool\_value}) \Rightarrow y{:}\texttt{bool}.\ y = \neg\, \mathit{bool\_value}$.
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: $pt = mem\_ptr \overset{\times}{\mapsto}_\tau pval$.
$term = \mathtt{representable}\,(\tau*, y_p) \wedge \mathtt{alignedI}\,(mem\_int, y_p)$.
$ret = \Sigma\, y_p{:}\mathtt{loc}.\ term \wedge \exists\, y{:}\beta_\tau.\ y_p \overset{\times}{\mapsto}_\tau y \otimes \mathtt{I}$.
ASSUME: 1. $\cdot;\cdot;\cdot;\cdot \vdash \mathtt{create}\,(mem\_int, \tau) \Rightarrow ret$.
2. $\langle \cdot; \mathtt{create}\,(mem\_int, \tau)\rangle \longrightarrow \langle \cdot + \{pt\}; \mathtt{done}\, mem\_ptr, pval, pt\rangle$.
PROVE: $\cdot;\cdot;\cdot;\cdot, pt \vdash \mathtt{done}\, mem\_ptr, pval, pt \Leftarrow ret$.

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

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

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

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

⟨2⟩3. $\cdot;\cdot;\cdot \vdash pval \Rightarrow \beta_\tau$ by ⟨2⟩1 and lemma 5.1.

⟨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: $pt = mem\_ptr \overset{\checkmark}{\mapsto}_\tau \_$.
$pt' = mem\_ptr \overset{\checkmark}{\mapsto}_\tau pval$.
$ret = \Sigma\, \_{:}\mathtt{unit}.\ pt' \otimes \mathtt{I}$.
ASSUME: 1. $\cdot;\cdot;\cdot;\cdot, pt \vdash \mathtt{store}\,(\_, \tau, pval_0, pval_1, \_, pt) \Rightarrow ret$.
2. $\langle \cdot + \{pt\}; \mathtt{store}\,(\_, \tau, mem\_ptr, pval, \_, pt)\rangle \longrightarrow \langle \cdot + \{pt'\}; \mathtt{done}\, \mathtt{Unit}, pt'\rangle$.
PROVE: $\cdot;\cdot;\cdot;\cdot, pt' \vdash \mathtt{done}\, \mathtt{Unit}, pt' \Leftarrow ret$.

⟨2⟩1. $\cdot;\cdot;\cdot \vdash \mathtt{Unit} \Rightarrow \mathtt{unit}$ 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: $pt = mem\_ptr \mapsto_\tau \_$.
ASSUME: 1. $\cdot;\cdot;\cdot;\cdot, pt \vdash \mathtt{kill}\,(\mathtt{static}\,\tau, pval_0, pt) \Rightarrow \Sigma\, \_{:}\mathtt{unit}.\ \mathtt{I}$.
2. $\langle \cdot + \{pt\}; \mathtt{kill}\,(\mathtt{static}\,\tau, mem\_ptr, pt)\rangle \longrightarrow \langle h; \mathtt{done}\, \mathtt{Unit}\rangle$.
PROVE: $\cdot;\cdot;\cdot;\cdot \vdash \mathtt{done}\, \mathtt{Unit} \Leftarrow \Sigma\, \_{:}\mathtt{unit}.\ \mathtt{I}$

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: $ret = \Sigma\, y{:}\mathtt{integer}.\ y = \mathtt{cast\_ptr\_to\_int}\, \mathit{mem\_ptr} \wedge \mathtt{I}$.
ASSUME: 1. $\cdot;\cdot;\cdot;\cdot \vdash \mathtt{intFromPtr}\,(\tau_1, \tau_2, \mathit{mem\_ptr}) \Rightarrow \mathit{ret}$.
2. $\langle \cdot; \mathtt{intFromPtr}\,(\tau_1, \tau_2, \mathit{mem\_ptr})\rangle \longrightarrow \langle \cdot; \mathtt{done}\, \mathit{mem\_int}\rangle$.
PROVE: $\cdot;\cdot;\cdot;\cdot \vdash \mathtt{done}\, \mathit{mem\_int} \Leftarrow \mathit{ret}$

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

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

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

⟨2⟩4. $\cdot;\cdot;\cdot \vdash \mathit{bool\_value} \Rightarrow \mathtt{bool}$ 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: $ret = \Sigma\, y{:}\mathtt{bool}.\ y = \mathtt{aligned}\,(\tau, \mathit{mem\_ptr}) \wedge \mathtt{I}$.
ASSUME: 1. $\cdot;\cdot;\cdot;\cdot \vdash \mathtt{ptrWellAligned}\,(\tau, \mathit{mem\_ptr}) \Rightarrow \mathit{ret}$.
2. $\langle \cdot; \mathtt{ptrWellAligned}\,(\tau, \mathit{mem\_ptr})\rangle \longrightarrow \langle \cdot; \mathtt{done}\, \mathit{bool\_value}\rangle$.
PROVE: $\cdot;\cdot;\cdot;\cdot \vdash \mathtt{done}\, \mathit{bool\_value} \Rightarrow \mathit{ret}$.

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

⟨2⟩2. $\cdot;\cdot;\cdot \vdash \mathit{bool\_value} \Rightarrow \mathtt{bool}$ 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 3.1.

⟨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.

⟨2⟩7. Apply substitution lemma (2.2) 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_MEMOP.
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 $\langle \mathit{pexpr}\rangle \longrightarrow \langle \mathit{pexpr}'\rangle$ 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: $\cdot;\cdot;\cdot \vdash \texttt{let}\, \mathit{ident\_or\_pattern} = \mathit{pexpr}'\, \texttt{in}\, \mathit{tpexpr} \Leftarrow y_2{:}\beta_2.\, \mathit{term}_2$.

⟨2⟩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.

⟨2⟩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 $\langle 2\rangle 2$.

$\langle 2\rangle 4.$ $\mathit{ret}_1 = \mathit{ret}_1'$.
PROOF: By cases TY_SEQ_E_{CCALL,PCALL}.

$\langle 2\rangle 5.$ By TY_SEQ_TE_LET with $\langle 2\rangle 1.2{,}3$ and $\langle 2\rangle 3$, we are done.

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

ASSUME: 1. $\cdot;\cdot;\cdot;\mathcal{R}',\mathcal{R} \vdash \texttt{let}\ \overline{\mathit{ret\_pattern}_i}^{\,i}{:}\mathit{ret}_1 = \texttt{done}\ \overline{\mathit{spine\_elem}_i}^{\,i}\ \texttt{in}\ \mathit{texpr}_2 \Leftarrow \mathit{ret}_2$.
2. $\langle h; \texttt{let}\ \overline{\mathit{ret\_pattern}_i}^{\,i}{:}\mathit{ret}_1 = \texttt{done}\ \overline{\mathit{spine\_elem}_i}^{\,i}\ \texttt{in}\ \mathit{texpr}\rangle \longrightarrow \langle h; \sigma(\mathit{texpr}_2)\rangle$.
PROVE: $\cdot;\cdot;\cdot;\mathcal{R}',\mathcal{R} \vdash \sigma(\mathit{texpr}_2) \Leftarrow \sigma(\mathit{ret}_2)$.

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

$\langle 2\rangle 2.$ $\overline{\mathit{ret\_pattern}_i = \mathit{spine\_elem}_i}^{\,i} \rightsquigarrow \sigma$.
PROOF: By inversion on 2.

$\langle 2\rangle 3.$ $\cdot;\cdot;\cdot;\mathcal{R}' \vdash (\sigma){:}(\mathcal{C}_1;\mathcal{L}_1;\Phi_1;\mathcal{R}_1)$.
PROOF: By $\langle 2\rangle 1.1{,}2$ and $\langle 2\rangle 2$ using lemma 5.2.

$\langle 2\rangle 4.$ By $\langle 2\rangle 1.3$ and $\langle 2\rangle 3$ and lemma 2.4, we are done.

$\langle 1\rangle 32.$ CASE: TY_SEQ_TE_LETT.
ASSUME: 1. $\cdot;\cdot;\cdot;\mathcal{R}',\mathcal{R} \vdash \texttt{let}\ \overline{\mathit{ret\_pattern}_i}^{\,i}{:}\mathit{ret}_1 = \mathit{texpr}_1\ \texttt{in}\ \mathit{texpr}_2 \Leftarrow \mathit{ret}_2$.
2. $\langle h; \texttt{let}\ \overline{\mathit{ret\_pattern}_i}^{\,i}{:}\mathit{ret} = \mathit{texpr}_1\ \texttt{in}\ \mathit{texpr}_2\rangle \longrightarrow \langle h'; \texttt{let}\ \overline{\mathit{ret\_pattern}_i}^{\,i}{:}\mathit{ret} =$

$texpr'_1$ in $texpr_2\rangle$.
PROVE: $\cdot;\cdot;\cdot;\mathcal{R}'',\mathcal{R} \vdash \mathtt{let}\ \overline{ret\_pattern_i}^{\,i} : ret_1 = texpr'_1\ \mathtt{in}\ texpr_2 \Leftarrow ret_2$.

$\langle 2\rangle 1$. 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.

$\langle 2\rangle 2$. $\langle h; texpr_1\rangle \longrightarrow \langle h'; texpr'_1\rangle$.
PROOF: By inversion on 2.

$\langle 2\rangle 3$. $\cdot;\cdot;\cdot;\mathcal{R}'' \vdash texpr'_1 \Leftarrow ret_1$.
PROOF: By induction on $\langle 2\rangle 1.1$ and $\langle 2\rangle 2$.

$\langle 2\rangle 4$. By $\langle 2\rangle 3$, $\langle 2\rangle 1.2,3$ using TY_SEQ_TE_LETT, we are done.

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

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

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

$\langle 2\rangle 3$. $\cdot;\cdot;\cdot;\cdot \vdash (\sigma_j){:}(\mathcal{C}_i;\cdot;\cdot;\cdot)$.
PROOF: By lemma 5.2.

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

$\langle 2\rangle 5$. By $\langle 1\rangle 32.3$ and 2.2, we are done.

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

$\langle 1\rangle 35$. CASE: TY_SEQ_TE_RUN.
PROOF SKETCH: Similar to case TY_SEQ_E_{CCALL,PCALL}.

$\langle 1\rangle 36$. CASE: TY_SEQ_TE_BOUND.
PROOF: By inversion on the typing rule.

$\langle 1\rangle 37$. CASE: TY_IS_TE_LETS.

PROOF SKETCH: Similar to TY_SEQ_TE_LETT.

6 Typing Judgements

$$
\begin{array}{lll}
\textit{object\_value\_jtype} & ::= & \\
& | & \mathcal{C}; \mathcal{L}; \Phi \vdash \textit{object\_value} \Rightarrow \texttt{obj}\, \beta \\[1em]
\textit{pval\_jtype} & ::= & \\
& | & \mathcal{C}; \mathcal{L}; \Phi \vdash \textit{pval} \Rightarrow \beta \\[1em]
\textit{res\_jtype} & ::= & \\
& | & \Phi \vdash \textit{res} \ \equiv \ \textit{res}' \\
& | & \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \textit{res\_term} \Leftarrow \textit{res} \\[1em]
\textit{spine\_jtype} & ::= & \\
& | & \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \ \overline{x_i = \textit{spine\_elem}_i}^{\,i} \ :: \textit{arg} \gg \sigma; \textit{ret} \\[1em]
\textit{pexpr\_jtype} & ::= & \\
& | & \mathcal{C}; \mathcal{L}; \Phi \vdash \textit{pexpr} \Rightarrow \textit{ident}{:}\beta.\, \textit{term} \\[1em]
\textit{tpval\_jtype} & ::= & \\
& | & \mathcal{C}; \mathcal{L}; \Phi \vdash \textit{tpval} \Leftarrow \textit{ident}{:}\beta.\, \textit{term} \\[1em]
\textit{tpexpr\_jtype} & ::= & \\
& | & \mathcal{C}; \mathcal{L}; \Phi \vdash \textit{tpexpr} \Leftarrow \textit{ident}{:}\beta.\, \textit{term} \\[1em]
\textit{action\_jtype} & ::= & \\
& | & \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \textit{mem\_action} \Rightarrow \textit{ret} \\[1em]
\textit{memop\_jtype} & ::= & \\
& | & \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \textit{mem\_op} \Rightarrow \textit{ret} \\[1em]
\textit{seq\_expr\_jtype} & ::= & \\
& | & \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \textit{seq\_expr} \Rightarrow \textit{ret} \\[1em]
\textit{is\_expr\_jtype} & ::= & \\
& | & \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \textit{is\_expr} \Rightarrow \textit{ret} \\[1em]
\textit{tval\_jtype} & ::= & \\
& | & \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \textit{tval} \Leftarrow \textit{ret} \\[1em]
\textit{texpr\_jtype} & ::= & \\
& | & \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \textit{seq\_texpr} \Leftarrow \textit{ret} \\
& | & \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \textit{is\_texpr} \Leftarrow \textit{ret} \\
& | & \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \textit{texpr} \Leftarrow \textit{ret}
\end{array}
$$

7 Opsem Judgements

$$
\begin{array}{llll}
\mathit{pure\_opsem\_jtype} & ::= & & \\
& | & \langle \mathit{pexpr} \rangle \longrightarrow \langle \mathit{pexpr}' \rangle & \\
& | & \langle \mathit{pexpr} \rangle \longrightarrow \langle \mathit{tpexpr}{:}(y{:}\beta.\, \mathit{term}) \rangle & \\
& | & \langle \mathit{tpexpr} \rangle \longrightarrow \langle \mathit{tpexpr}' \rangle & \\
\\
\mathit{opsem\_jtype} & ::= & & \\
& | & \langle h; \mathit{seq\_expr} \rangle \longrightarrow \langle h'; \mathit{texpr}{:}\mathit{ret} \rangle & \\
& | & \langle h; \mathit{seq\_texpr} \rangle \longrightarrow \langle h'; \mathit{texpr} \rangle & \\
& | & \langle h; \mathit{mem\_op} \rangle \longrightarrow \langle h'; \mathit{tval} \rangle & \\
& | & \langle h; \mathit{mem\_action} \rangle \longrightarrow \langle h'; \mathit{tval} \rangle & \\
& | & \langle h; \mathit{is\_expr} \rangle \longrightarrow \langle h'; \mathit{is\_expr}' \rangle & \\
& | & \langle h; \mathit{is\_texpr} \rangle \longrightarrow \langle h'; \mathit{texpr} \rangle & \\
& | & \langle h; \mathit{texpr} \rangle \longrightarrow \langle h'; \mathit{texpr}' \rangle &
\end{array}
$$