Modularly Composing Typed Language Fragments

Supplemental Material

1. Internal Language

We assume the statics of the IL are specified in the standard way by judgements for type formation $\Delta \vdash \tau$, typing context formation $\Delta \vdash \Gamma$ and type assignment $\Delta\ \Gamma \vdash \iota : \tau^{+}$. The internal dynamics are specified as a structural operational semantics with a stepping judgement $\iota \mapsto \iota^{+}$ and a value judgement $\iota\ \texttt{val}$. The multi-step judgement $\iota \mapsto^{*} \iota^{+}$ is the reflexive, transitive closure of the stepping judgement and the evaluation judgement $\iota \Downarrow \iota'$ is defined iff $\iota \mapsto^{*} \iota'$ and $\iota'\ \texttt{val}$. Both the static and dynamic semantics of the IL can be found in any standard textbook covering typed lambda calculi (we directly follow [1]), so we assume familiarity and give the lemmas in this section without proof.

We use $\mathcal{L}\{\rightharpoonup \forall\, \mu\, 1 \times +\}$, the syntax for which is shown in Figure 1, as representative of any intermediate language for a typed functional language. In fact, our intention is not to prescribe a particular choice of IL, so we will here only review the key metatheoretic properties that the IL must possess. Each choice of IL is technically a distinct dialect of @λ, but for the broad class of ILs that enjoy these properties, the metatheory in the remainder of the supplement should follow without trouble.

Lemma 1 (Internal Type Assignment)**.** *If $\Delta \vdash \Gamma$ and $\Delta\ \Gamma \vdash \iota : \tau$ then $\Delta \vdash \tau$.*

Internal type safety follows the standard methodology of proving preservation and progress lemmas.

Lemma 2 (Internal Progress)**.** *If $\emptyset\ \emptyset \vdash \iota : \tau$ then either $\iota\ \texttt{val}$ or $\iota \mapsto \iota'$.*

Lemma 3 (Internal Preservation)**.** *If $\emptyset\ \emptyset \vdash \iota : \tau$ and $\iota \mapsto \iota'$ then $\emptyset\ \emptyset \vdash \iota' : \tau$.*

We assume substitution and simultaneous n-ary substitutions, δ and γ, have the standard semantics, and that typing and type formation contexts, Δ and Γ, obey standard properties of weakening, exchange and contraction. We assume that the names of variables and type variables are unimportant, so that α-equivalent terms, types and contexts can be identified implicitly throughout this work. In particular, we need the definitions of substitution validity shown in Figures 2 and 3 and the following lemmas.

internal types

$$\tau \quad ::= \quad \tau \rightharpoonup \tau \mid \alpha \mid \forall(\alpha.\tau) \mid t \mid \mu(t.\tau) \mid 1 \mid \tau \times \tau \mid \tau + \tau$$

internal terms

$$\begin{array}{lll} \iota & ::= & x \mid \lambda[\tau](x.\iota) \mid \iota(\iota) \mid \mathsf{fix}[\tau](x.\iota) \mid \Lambda(\alpha.\iota) \mid \iota\texttt{[}\tau\texttt{]} \\ & \mid & \mathsf{fold}[t.\tau](\iota) \mid \mathsf{unfold}(\iota) \mid () \mid (\iota, \iota) \mid \mathsf{fst}(\iota) \mid \mathsf{snd}(\iota) \\ & \mid & \mathsf{inl}[\tau](\iota) \mid \mathsf{inr}[\tau](\iota) \mid \mathsf{case}(\iota; x.\iota; x.\iota) \end{array}$$

internal typing contexts $\Gamma ::= \emptyset \mid \Gamma, x : \tau$

internal type formation contexts $\Delta ::= \emptyset \mid \Delta, \alpha \mid \Delta, t$

Figure 1. Syntax of $\mathcal{L}\{\rightharpoonup \forall\, \mu\, 1 \times +\}$, our internal language (IL). Metavariable x ranges over term variables and α and t both range over type variables.

$\boxed{\Delta \vdash \delta : \Delta}$ $\quad \delta ::= \emptyset \mid \delta, \tau/\alpha$

$$\text{(tysub-emp)}\ \frac{}{\Delta \vdash \emptyset : \emptyset} \qquad \text{(tysub-ext)}\ \frac{\Delta \vdash \delta : \Delta' \qquad \Delta \vdash \tau}{\Delta \vdash \delta, \tau/\alpha : \Delta', \alpha}$$

Figure 2. Internal Type Substitution Validity

$\boxed{\Delta\ \Gamma \vdash \gamma : \Gamma}$ $\quad \gamma ::= \emptyset \mid \gamma, \iota/x$

$$\text{(tmsub-emp)}\ \frac{}{\Delta\ \Gamma \vdash \emptyset : \emptyset} \qquad \text{(tmsub-ext)}\ \frac{\Delta\ \Gamma \vdash \gamma : \Gamma' \qquad \Delta \vdash \tau \qquad \Delta\ \Gamma \vdash \iota : \tau}{\Delta\ \Gamma \vdash \gamma, \iota/x : \Gamma', x : \tau}$$

Figure 3. Internal Term Substitution Validity

Lemma 4 (Internal Type Substitution on Types)**.** *If $\Delta \vdash \delta : \Delta'$ and $\Delta\Delta' \vdash \tau$ then $\Delta \vdash [\delta]\tau$.*

Lemma 5 (Internal Type Substitutions on Typing Contexts)**.** *If $\Delta \vdash \delta : \Delta'$ and $\Delta\Delta' \vdash \Gamma$ then $\Delta \vdash [\delta]\Gamma$.*

Lemma 6 (Internal Type Substitutions on Terms)**.** *If $\Delta \vdash \delta : \Delta'$ and $\Delta\Delta' \vdash \tau$ and $\Delta\Delta' \vdash \Gamma$ and $\Delta\Delta'\ \Gamma \vdash \iota : \tau$ then $\Delta\ [\delta]\Gamma \vdash [\delta]\iota : [\delta]\tau$.*

$\boxed{\vdash \Phi}$

(tcc-emp)

$$\frac{}{\vdash \cdot}$$

(tcc-ext)

$$\frac{\vdash \Phi \qquad \textsc{tc} \notin \text{dom}(\Phi) \qquad \emptyset \vdash \kappa_{\text{tyidx}}\ \mathsf{eq} \qquad \emptyset\ \emptyset \vdash^0_\Phi \sigma_{\text{schema}} :: \kappa_{\text{tyidx}} \to \mathsf{ITy} \qquad \vdash_{\Phi, \mathsf{tycon}\ \textsc{tc}\ \{\mathsf{trans} = \sigma_{\text{schema}}\ \mathsf{in}\ \omega\} \sim \mathsf{tcsig}[\kappa_{\text{tyidx}}]\ \{\chi\}}\ \omega \sim \mathsf{tcsig}[\kappa_{\text{tyidx}}]\ \{\chi\}}{\vdash \Phi, \mathsf{tycon}\ \textsc{tc}\ \{\mathsf{trans} = \sigma_{\text{schema}}\ \mathsf{in}\ \omega\} \sim \mathsf{tcsig}[\kappa_{\text{tyidx}}]\ \{\chi\}}$$

Figure 4. Tycon context well-definedness.

$\boxed{\vdash_\Phi \omega \sim \psi}$

(ocstruct-intro)

$$\frac{\mathsf{intro}[\kappa_{\text{tmidx}}] \in \chi \qquad \emptyset \vdash \kappa_{\text{tmidx}} \qquad \emptyset\ \emptyset \vdash^0_\Phi \sigma_{\text{def}} :: \kappa_{\text{tyidx}} \to \kappa_{\text{tmidx}} \to \mathsf{List}[\mathsf{Arg}] \to \mathsf{ITm}}{\vdash_\Phi \mathsf{ana\ intro} = \sigma_{\text{def}} \sim \mathsf{tcsig}[\kappa_{\text{tyidx}}]\ \{\chi\}}$$

(ocstruct-targ)

$$\frac{\vdash_\Phi \omega \sim \mathsf{tcsig}[\kappa_{\text{tyidx}}]\ \{\chi\} \qquad \mathbf{op} \notin \text{dom}(\chi) \qquad \emptyset \vdash \kappa_{\text{tmidx}} \qquad \emptyset\ \emptyset \vdash^0_\Phi \sigma_{\text{def}} :: \kappa_{\text{tyidx}} \to \kappa_{\text{tmidx}} \to \mathsf{List}[\mathsf{Arg}] \to (\mathsf{Ty} \times \mathsf{ITm})}{\vdash_\Phi \omega; \mathsf{syn}\ \mathbf{op} = \sigma_{\text{def}} \sim \mathsf{tcsig}[\kappa_{\text{tyidx}}]\ \{\chi, \mathbf{op}[\kappa_{\text{tmidx}}]\}}$$

Figure 5. Checking opcon structures against tycon signatures.

Lemma 7 (Internal Term Substitutions)**.** *If* $\Delta \vdash \Gamma$ *and* $\Delta \vdash \Gamma'$ *and* $\Delta\ \Gamma \vdash \gamma : \Gamma'$ *and* $\Delta\ \Gamma\Gamma' \vdash \iota : \tau$ *then* $\Delta\ \Gamma \vdash [\gamma]\iota : \tau$*.*

2. Tycon Contexts

2.1 Tycon Context Well-Definedness

2.2 Equality Kinds

Need an equational theory for SL to state equality kind property, but not important for other metatheory.

2.3 Full Examples

3. Static Language

3.1 Kind Formation

3.2 Kinding Context Formation

3.3 Kinding

3.4 Dynamic Semantics

3.5 Kind Safety

4. Types

4.1 Type Translations

4.2 Typing Context Translations

Unicity The rules are structured so that if a term is well-typed, both its type and translation are unique.

could move this whole thing to supple

Theorem 1 (Unicity)**.** *If* $\vdash \Phi$ *and* $\vdash_\Phi \Upsilon \leadsto \Gamma$ *and* $\vdash_\Phi \sigma \leadsto \tau$ *and* $\vdash_\Phi \sigma' \leadsto \tau'$ *and* $\Upsilon \vdash_\Phi e \Leftarrow \sigma \leadsto \iota$ *and* $\Upsilon \vdash_\Phi e \Leftarrow \sigma' \leadsto \iota'$ *then* $\sigma = \sigma'$ *and* $\tau = \tau'$ *and* $\iota = \iota'$*.*

5. External Language

5.1 Additional Desugarings

5.2 Typing

5.3 Proof of Regular String Soundness Tycon Invariant

References

[1] R. Harper. *Practical Foundations for Programming Languages*. Cambridge University Press, 2012.

A. Appendix

(s-ty-step)
$$\frac{\sigma \mapsto_{\mathcal{A}} \sigma'}{c\langle\sigma\rangle \mapsto_{\mathcal{A}} c\langle\sigma'\rangle}$$

(s-ty-err)
$$\frac{\sigma\ \mathtt{err}_{\mathcal{A}}}{c\langle\sigma\rangle\ \mathtt{err}_{\mathcal{A}}}$$

(s-ty-v)
$$\frac{\sigma\ \mathtt{val}_{\mathcal{A}}}{c\langle\sigma\rangle\ \mathtt{val}_{\mathcal{A}}}$$

(s-otherty-v)
$$\frac{}{\mathsf{otherty}[m;\tau]\ \mathtt{val}_{\mathcal{A}}}$$

(s-tycase-step)
$$\frac{\sigma \mapsto_{\mathcal{A}} \sigma'}{\mathsf{tycase}[c](\sigma;\boldsymbol{x}.\sigma_1;\sigma_2) \mapsto_{\mathcal{A}} \mathsf{tycase}[c](\sigma';x.\sigma_1;\sigma_2)}$$

(s-tycase-err)
$$\frac{\sigma\ \mathtt{err}_{\mathcal{A}}}{\mathsf{tycase}[c](\sigma;\boldsymbol{x}.\sigma_1;\sigma_2)\ \mathtt{err}_{\mathcal{A}}}$$

(s-tycase-match)
$$\frac{c\langle\sigma\rangle\ \mathtt{val}_{\mathcal{A}}}{\mathsf{tycase}[c](c\langle\sigma\rangle;\boldsymbol{x}.\sigma_1;\sigma_2) \mapsto_{\mathcal{A}} [\sigma/x]\sigma_1}$$

(s-tycase-fail)
$$\frac{\sigma \neq c\langle\sigma'\rangle}{\mathsf{tycase}[c](\sigma;\boldsymbol{x}.\sigma_1;\sigma_2) \mapsto_{\mathcal{A}} \sigma_2}$$

(keq-k)
$$\frac{\boldsymbol{k} \in \boldsymbol{\Delta}}{\boldsymbol{\Delta} \vdash \boldsymbol{k}\ \mathsf{eq}}$$

(keq-ind)
$$\frac{\boldsymbol{\Delta}, \boldsymbol{k} \vdash \kappa\ \mathsf{eq}}{\boldsymbol{\Delta} \vdash \mu_{\text{ind}}(\boldsymbol{k}.\kappa)\ \mathsf{eq}}$$

(keq-unit)
$$\frac{}{\boldsymbol{\Delta} \vdash 1\ \mathsf{eq}}$$

(keq-prod)
$$\frac{\boldsymbol{\Delta} \vdash \kappa_1\ \mathsf{eq} \quad \boldsymbol{\Delta} \vdash \kappa_2\ \mathsf{eq}}{\boldsymbol{\Delta} \vdash \kappa_1 \times \kappa_2\ \mathsf{eq}}$$

(keq-sum)
$$\frac{\boldsymbol{\Delta} \vdash \kappa_1\ \mathsf{eq} \quad \boldsymbol{\Delta} \vdash \kappa_2\ \mathsf{eq}}{\boldsymbol{\Delta} \vdash \kappa_1 + \kappa_2\ \mathsf{eq}}$$

(keq-ty)
$$\frac{}{\boldsymbol{\Delta} \vdash \mathsf{Ty}\ \mathsf{eq}}$$

(k-ity-alpha)
$$\frac{}{\boldsymbol{\Delta}\ \boldsymbol{\Gamma} \vdash^n_\Phi \blacktriangleright(\alpha) :: \mathsf{ITy}}$$

(s-ity-lam-step-1)
$$\frac{\blacktriangleright(\hat{\tau}_1) \mapsto_{\mathcal{A}} \blacktriangleright(\hat{\tau}_1')}{\blacktriangleright(\hat{\tau}_1 \times \hat{\tau}_2) \mapsto_{\mathcal{A}} \blacktriangleright(\hat{\tau}_1' \times \hat{\tau}_2)}$$

(s-ity-lam-step-2)
$$\frac{\blacktriangleright(\hat{\tau}_1)\ \mathtt{val}_{\mathcal{A}} \qquad \blacktriangleright(\hat{\tau}_2) \mapsto_{\mathcal{A}} \blacktriangleright(\hat{\tau}_2')}{\blacktriangleright(\hat{\tau}_1 \times \hat{\tau}_2) \mapsto_{\mathcal{A}} \blacktriangleright(\hat{\tau}_1 \times \hat{\tau}_2')}$$

(s-ity-lam-err-1)
$$\frac{\blacktriangleright(\hat{\tau}_1)\ \mathtt{err}_{\mathcal{A}}}{\blacktriangleright(\hat{\tau}_1 \times \hat{\tau}_2)\ \mathtt{err}_{\mathcal{A}}}$$

(s-ity-lam-err-2)
$$\frac{\blacktriangleright(\hat{\tau}_2)\ \mathtt{err}_{\mathcal{A}}}{\blacktriangleright(\hat{\tau}_1 \times \hat{\tau}_2)\ \mathtt{err}_{\mathcal{A}}}$$

(s-ity-lam-v)
$$\frac{\blacktriangleright(\hat{\tau}_1)\ \mathtt{val}_{\mathcal{A}} \qquad \blacktriangleright(\hat{\tau}_2)\ \mathtt{val}_{\mathcal{A}}}{\blacktriangleright(\hat{\tau}_1 \times \hat{\tau}_2)\ \mathtt{val}_{\mathcal{A}}}$$

(s-ity-alpha-v)
$$\frac{}{\blacktriangleright(\alpha)\ \mathtt{val}_{\mathcal{A}}}$$

(k-tycase-parr)
$$\frac{\begin{array}{c}\boldsymbol{\Delta}\ \boldsymbol{\Gamma} \vdash^n_\Phi \sigma :: \mathsf{Ty} \\ \boldsymbol{\Delta}\ \boldsymbol{\Gamma}, \boldsymbol{x} :: \mathsf{Ty} \times \mathsf{Ty} \vdash^n_\Phi \sigma_1 :: \kappa \qquad \boldsymbol{\Delta}\ \boldsymbol{\Gamma} \vdash^n_\Phi \sigma_2 :: \kappa\end{array}}{\boldsymbol{\Delta}\ \boldsymbol{\Gamma} \vdash^n_\Phi \mathsf{tycase}[\rightharpoonup](\sigma;\boldsymbol{x}.\sigma_1;\sigma_2) :: \kappa}$$

(s-ity-unquote-step)
$$\frac{\sigma \mapsto_{\mathcal{A}} \sigma'}{\blacktriangleright(\blacktriangleleft(\sigma)) \mapsto_{\mathcal{A}} \blacktriangleright(\blacktriangleleft(\sigma'))}$$

(s-ity-unquote-err)
$$\frac{\sigma\ \mathtt{err}_{\mathcal{A}}}{\blacktriangleright(\blacktriangleleft(\sigma))\ \mathtt{err}_{\mathcal{A}}}$$

(s-ity-trans-step)
$$\frac{\sigma \mapsto_{\mathcal{A}} \sigma'}{\blacktriangleright(\mathsf{trans}(\sigma)) \mapsto_{\mathcal{A}} \blacktriangleright(\mathsf{trans}(\sigma'))}$$

(s-ity-trans-err)
$$\frac{\sigma\ \mathtt{err}_{\mathcal{A}}}{\blacktriangleright(\mathsf{trans}(\sigma))\ \mathtt{err}_{\mathcal{A}}}$$

This judgement is defined by the following straightforward rules:

(tstore-emp)
$$\frac{}{\emptyset \rightsquigarrow \emptyset : \emptyset}$$

(tstore-ext)
$$\frac{\mathcal{D} \rightsquigarrow \delta : \Delta}{(\mathcal{D}, \sigma \leftrightarrow \tau/\alpha) \rightsquigarrow (\delta, \tau/\alpha) : (\Delta, \alpha)}$$

Description	**Concrete Form**	**Desugared Form**
sequences	$(e_1, \ldots, e_n)$ or $[e_1, \ldots, e_n]$	$\mathsf{intro}[()](e_1; \ldots; e_n)$
labeled sequences	$\{\mathtt{lbl}_1 = e_1, \ldots, \mathtt{lbl}_n = e_n\}$	$\mathsf{intro}[[\mathtt{lbl}_1, \ldots, \mathtt{lbl}_n]](e_1; \ldots; e_n)$
label application	$\mathtt{lbl}\langle e_1, \ldots, e_n\rangle$	$\mathsf{intro}[\mathtt{lbl}](e_1, \ldots, e_n)$
numerals	n	$\mathsf{intro}[n](\cdot)$
labeled numerals	$n\mathtt{lbl}$	$\mathsf{intro}[(n, \mathtt{lbl})](\cdot)$
strings	"s"	$\mathsf{intro}[\texttt{"s"}](\cdot)$

(s-itm-var-v)
$$\frac{}{\triangleright(x)\ \mathtt{val}_{\mathcal{A}}}$$

(s-itm-lam-step-1)
$$\frac{\blacktriangleright(\hat{\tau}) \mapsto_{\mathcal{A}} \blacktriangleright(\hat{\tau}')}{\triangleright(\lambda[\hat{\tau}](x.\hat{\iota})) \mapsto_{\mathcal{A}} \triangleright(\lambda[\hat{\tau}'](x.\hat{\iota}))}$$

(s-itm-lam-step-2)
$$\frac{\blacktriangleright(\hat{\tau})\ \mathtt{val}_{\mathcal{A}} \qquad \triangleright(\hat{\iota}) \mapsto_{\mathcal{A}} \triangleright(\hat{\iota}')}{\triangleright(\lambda[\hat{\tau}](x.\hat{\iota})) \mapsto_{\mathcal{A}} \triangleright(\lambda[\hat{\tau}](x.\hat{\iota}'))}$$

(s-itm-lam-err-1)
$$\frac{\blacktriangleright(\hat{\tau})\ \mathtt{err}_{\mathcal{A}}}{\triangleright(\lambda[\hat{\tau}](x.\hat{\iota}))\ \mathtt{err}_{\mathcal{A}}}$$

(s-itm-lam-err-2)
$$\frac{\triangleright(\hat{\iota})\ \mathtt{err}_{\mathcal{A}}}{\triangleright(\lambda[\hat{\tau}](x.\hat{\iota}))\ \mathtt{err}_{\mathcal{A}}}$$

(s-itm-lam-v)
$$\frac{\blacktriangleright(\hat{\tau})\ \mathtt{val}_{\mathcal{A}} \qquad \triangleright(\hat{\iota})\ \mathtt{val}_{\mathcal{A}}}{\triangleright(\lambda[\hat{\tau}](x.\hat{\iota}))\ \mathtt{val}_{\mathcal{A}}}$$

(k-itm-unquote)
$$\frac{\mathbf{\Delta}\ \mathbf{\Gamma} \vdash^n_{\Phi} \sigma :: \mathsf{ITm}}{\mathbf{\Delta}\ \mathbf{\Gamma} \vdash^n_{\Phi} \triangleright(\triangleleft(\sigma)) :: \mathsf{ITm}}$$

(s-itm-unquote-step)
$$\frac{\sigma \mapsto_{\mathcal{A}} \sigma'}{\triangleright(\triangleleft(\sigma)) \mapsto_{\mathcal{A}} \triangleright(\triangleleft(\sigma'))}$$

(s-itm-unquote-err)
$$\frac{\sigma\ \mathtt{err}_{\mathcal{A}}}{\triangleright(\triangleleft(\sigma))\ \mathtt{err}_{\mathcal{A}}}$$

(s-itm-unquote-elim)
$$\frac{\triangleright(\hat{\iota})\ \mathtt{val}_{\mathcal{A}}}{\triangleright(\triangleleft(\triangleright(\hat{\iota}))) \mapsto_{\mathcal{A}} \triangleright(\hat{\iota})}$$

(s-ana-step)
$$\frac{\sigma \mapsto_{\mathcal{A}} \sigma'}{\mathsf{ana}[n](\sigma) \mapsto_{\mathcal{A}} \mathsf{ana}[n](\sigma')}$$

(s-ana-err)
$$\frac{\sigma\ \mathtt{err}_{\mathcal{A}}}{\mathsf{ana}[n](\sigma)\ \mathtt{err}_{\mathcal{A}}}$$

(s-itm-anatrans-step)
$$\frac{\sigma \mapsto_{\mathcal{A}} \sigma'}{\triangleright(\mathsf{anatrans}[n](\sigma)) \mapsto_{\mathcal{A}} \triangleright(\mathsf{anatrans}[n](\sigma'))}$$

(s-itm-anatrans-err)
$$\frac{\sigma\ \mathtt{err}_{\mathcal{A}}}{\triangleright(\mathsf{anatrans}[n](\sigma))\ \mathtt{err}_{\mathcal{A}}}$$

, as specified by the judgement $\mathcal{G} \rightsquigarrow \gamma : \Gamma$ defined by the following rules:

(ttrs-emp)
$$\frac{}{\emptyset \rightsquigarrow \emptyset : \emptyset}$$

(ttrs-ext)
$$\frac{\mathcal{G} \rightsquigarrow \gamma : \Gamma}{(\mathcal{G}, n : \sigma \rightsquigarrow \iota/x : \tau) \rightsquigarrow (\gamma, \iota/x) : (\Gamma, x : \tau)}$$

(k-itm-lam)
$$\frac{\mathbf{\Delta}\ \mathbf{\Gamma} \vdash^n_{\Phi} \blacktriangleright(\hat{\tau}) :: \mathsf{ITy} \qquad \mathbf{\Delta}\ \mathbf{\Gamma} \vdash^n_{\Phi} \triangleright(\hat{\iota}) :: \mathsf{ITm}}{\mathbf{\Delta}\ \mathbf{\Gamma} \vdash^n_{\Phi} \triangleright(\lambda[\hat{\tau}](x.\hat{\iota})) :: \mathsf{ITm}}$$

(k-raise)
$$\frac{\mathbf{\Delta} \vdash \kappa}{\mathbf{\Delta}\ \mathbf{\Gamma} \vdash^n_{\Phi} \mathsf{raise}[\kappa] :: \kappa}$$

(s-raise)
$$\frac{}{\mathsf{raise}[\kappa]\ \mathtt{err}_{\mathcal{A}}}$$

(s-syn-success)
$$\frac{\mathsf{nth}[n](\overline{e}) = e \qquad \Upsilon \vdash_{\Phi} e \Rightarrow \sigma \rightsquigarrow \iota}{\mathsf{syn}[n] \mapsto_{\overline{e};\Upsilon;\Phi} (\sigma, \triangleright(\mathsf{syntrans}[n]))}$$

(s-syn-fail)
$$\frac{\mathsf{nth}[n](\overline{e}) = e \qquad [\Upsilon \vdash_{\Phi} e \not\Rightarrow]}{\mathsf{syn}[n]\ \mathtt{err}_{\overline{e};\Upsilon;\Phi}}$$

(k-itm-syntrans)
$$\frac{n' < n}{\mathbf{\Delta}\ \mathbf{\Gamma} \vdash^n_{\Phi} \triangleright(\mathsf{syntrans}[n']) :: \mathsf{ITm}}$$

, e.g. for lambdas:

(abs-lam)
$$\frac{\hat{\tau} \parallel \mathcal{D} \hookrightarrow^{\text{TC}}_{\Phi} \tau \parallel \mathcal{D}' \qquad \hat{\iota} \parallel \mathcal{D}'\ \mathcal{G} \hookrightarrow^{\text{TC}}_{\overline{e};\Upsilon;\Phi} \iota \parallel \mathcal{D}''\ \mathcal{G}'}{\lambda[\hat{\tau}](x.\hat{\iota}) \parallel \mathcal{D}\ \mathcal{G} \hookrightarrow^{\text{TC}}_{\overline{e};\Upsilon;\Phi} \lambda[\tau](x.\iota) \parallel \mathcal{G}'\ \mathcal{D}''}$$

(abs-anatrans-stored)
$$\frac{n : \sigma \rightsquigarrow \iota/x : \tau \in \mathcal{G}}{\mathsf{anatrans}[n](\sigma) \parallel \mathcal{G}\ \mathcal{D} \hookrightarrow^{\text{TC}}_{\mathcal{A}} x \parallel \mathcal{G}\ \mathcal{D}}$$

(abs-syntrans-stored)
$$\frac{n : \sigma \rightsquigarrow \iota/x : \tau \in \mathcal{G}}{\mathsf{syntrans}[n] \parallel \mathcal{G}\ \mathcal{D} \hookrightarrow^{\text{TC}}_{\mathcal{A}} x \parallel \mathcal{G}\ \mathcal{D}}$$

(k-itm-anatrans)
$$\frac{n' < n \qquad \mathbf{\Delta}\ \mathbf{\Gamma} \vdash^n_{\Phi} \sigma :: \mathsf{Ty}}{\mathbf{\Delta}\ \mathbf{\Gamma} \vdash^n_{\Phi} \triangleright(\mathsf{anatrans}[n'](\sigma)) :: \mathsf{ITm}}$$

(abs-syntrans-new)
$$\frac{\begin{array}{c} n \notin \mathrm{dom}(\mathcal{G}) \qquad \mathsf{nth}[n](\overline{e}) = e \\ \Upsilon \vdash_{\Phi} e \Rightarrow \sigma \rightsquigarrow \iota \qquad \mathsf{trans}(\sigma) \parallel \mathcal{D} \hookrightarrow^{\text{TC}}_{\Phi} \tau \parallel \mathcal{D}' \qquad (x \text{ fresh}) \end{array}}{\mathsf{syntrans}[n] \parallel \mathcal{G}\ \mathcal{D} \hookrightarrow^{\text{TC}}_{\overline{e};\Upsilon;\Phi} x \parallel \mathcal{G}, n : \sigma \rightsquigarrow \iota/x : \tau\ \mathcal{D}'}$$

(etctx-emp)
$$\frac{}{\vdash_{\Phi} \emptyset \rightsquigarrow \emptyset}$$

(etctx-ext)
$$\frac{\vdash_{\Phi} \Upsilon \rightsquigarrow \Gamma \qquad \sigma\ \mathtt{type}_{\Phi} \qquad \vdash_{\Phi} \sigma \rightsquigarrow \tau}{\vdash_{\Phi} \Upsilon, x \Rightarrow \sigma \rightsquigarrow \Gamma, x : \tau}$$

Description	Concrete Form	Desugared Form
index projection	$e_{\text{targ}}\#n$	$\mathsf{targ}[\mathbf{idx}; n](e_{\text{targ}}; \cdot)$
label projection	$e_{\text{targ}}\#\mathtt{lbl}$	$\mathsf{targ}[\mathbf{prj}; \mathtt{lbl}](e_{\text{targ}}; \cdot)$
explicit invocation	$e_{\text{targ}}\cdot\mathbf{op}[\sigma_{\text{tmidx}}](\overline{e})$	$\mathsf{targ}[\mathbf{op}; \sigma_{\text{tmidx}}](e_{\text{targ}}; \overline{e})$
	$e_{\text{targ}}\cdot\mathbf{op}(\overline{e})$	$\mathsf{targ}[\mathbf{op}; ()](e_{\text{targ}}; \overline{e})$
	$e_{\text{targ}}\cdot\mathbf{op}(\mathtt{lbl}_1 = e_1, \ldots, \mathtt{lbl}_n = e_n)$	$\mathsf{targ}[\mathbf{op}; [\mathtt{lbl}_1, \ldots, \mathtt{lbl}_n]](e_{\text{targ}}; e_1; \ldots; e_n)$
labeled case analysis	$e_{\text{targ}}\cdot\mathbf{case}\,\{ \mid \sigma_1\langle x_1, \ldots, x_k\rangle \Rightarrow e_1 \mid \ldots \mid \sigma_n\langle x_1, \ldots, x_k\rangle \Rightarrow e_n\}$	$\mathsf{targ}[\mathbf{case}; [\sigma_1, \ldots, \sigma_n]](e_{\text{targ}}; \lambda(x_1.\ldots\lambda(x_k.e_1)); \ldots; \lambda(x_1.\ldots\lambda(x_k.e_n)))$

For example,

(abs-prod)
$$\frac{\hat{\tau}_1 \parallel \mathcal{D} \hookrightarrow^{\text{TC}}_{\Phi} \tau_1 \parallel \mathcal{D}' \qquad \hat{\tau}_2 \parallel \mathcal{D}' \hookrightarrow^{\text{TC}}_{\Phi} \tau_2 \parallel \mathcal{D}''}{\hat{\tau}_1 \times \hat{\tau}_2 \parallel \mathcal{D} \hookrightarrow^{\text{TC}}_{\Phi} \tau_1 \times \tau_2 \parallel \mathcal{D}''}$$

The argument interfaces that populate the list provided to opcon definitions is derived from the argument list by the judgement $\mathsf{args}(\overline{e}) =_n \sigma_{\text{args}}$, defined as follows:

(args-z)
$$\frac{}{\mathsf{args}(\cdot) =_0 \boldsymbol{nil}[\mathsf{Arg}]}$$

(args-s)
$$\frac{\mathsf{args}(\overline{e}) =_n \sigma}{\mathsf{args}(\overline{e}; e) =_{n+1} \boldsymbol{rcons}[\mathsf{Arg}]\ \sigma\ (\lambda \boldsymbol{ty}{::}\mathsf{Ty}.\mathsf{ana}[n](\boldsymbol{ty}), \lambda\_{::}1.\mathsf{syn}[n])}$$

We assume that the definitions of the standard helper functions $\boldsymbol{nil} :: \forall(\boldsymbol{\alpha}.\mathsf{List}[\boldsymbol{\alpha}])$ and $\boldsymbol{rcons} :: \forall(\boldsymbol{\alpha}.\mathsf{List}[\boldsymbol{\alpha}] \to \boldsymbol{\alpha} \to \mathsf{List}[\boldsymbol{\alpha}])$, which adds an item to the end of a list, have been substituted into these rules. The result is that the nth element of the argument interface list simply wraps the static terms $\mathsf{ana}[n](\sigma)$ and $\mathsf{syn}[n]$.