Elaborating course-of-values induction in Cedille

Christopher Jenkins, Denis Firsov, Aaron Stump, Larry Diehl, Colin McDonald

October 15, 2020

Contents

0 Figures and Definitions

0.1 Syntax

The reader is refered to Stump [Stu18] for more information about the standard fragment of Cedille that does not include datatypes. In Figures 1 and 2, the new language constructs introduced by the datatype system are $\boxed{\text{boxed}}$.

a, u, x, y, z	term variables
X, Y, Z, R	type variables
$\boxed{c}$	constructors
$\boxed{D}$	datatypes

Figure 1: Identifiers

p, q	::=	x	variables
		$\lambda\, x.\, p$	functions
		$p\ p'$	applications
		$\boxed{c}$	constructors
		$\boxed{\mu\ u.\ p\ \{\ c_i\ \overline{a_i} \to p_i\}_{i=1..n}}$	recursive definitions
		$\boxed{\sigma\ p\ \{c_i\ \overline{a_i} \to p_i\}_{i=1..n}}$	case analysis
$\overline{a}$	::=	$\boxed{\emptyset \mid a\ \overline{a}}$	

Figure 2: Untyped terms

0.2 CDLE judgments and meta-theory

Our notational convention is that judgments with a hooked arrow $\hookrightarrow$ are for the surface language Cedille (which contains CDLE) as a sub-language, and judgments without are for pure CDLE contstructs (as listed in [Stu18]). The CDLE judgemnts are

- $\Gamma \vdash K$ meaning under typing context Γ the kind K is well formed
- $\Gamma \vdash T : K$ meaning under typing context Γ type T has kind K
- $\Gamma \vdash t : T$ meaning under typing context Γ term t has type T.

We list two of CDLE's meta-theoretic results, *logical consistency* and *call-by-name normalization of closed functions.*

Proposition 1 ([Stu18])**.** *There t such that $\emptyset \vdash t : \forall\, X : \star.\, X \to X$.*

Proposition 2 ([Stu18])**.** *Suppose $\Gamma \vdash t : T$, t is closed, and there exists a closed t' that erases to $\lambda\, x.\, x$ and whose type is $T \to \Pi\, x : T_1.\, T_2$ for some T_1, T_2. Then $|t|$ is call-by-name normalizing.*

In Theorem 2, the condition on t that it may be "re-typed" to a function excludes non-terminating terms introduced by the Kleene trick, and the condition that t be closed relates to the derivability of "zero-cost type coercions" in CDLE (Cast, Figure 7), which can cause non-termination in inconsistent contexts [AC19].

Kinds K	$::=$	$\star$	the kind of types that classify terms
		$\Pi\, X{:}K.\, K'$	product over types
		$\Pi\, x{:}T.\, K$	product over terms
Types S, T, P	$::=$	X	type variables
		$\Pi\, x{:}S.\, T$	product over terms
		$\forall\, x{:}S.\, T$	implicit product over terms
		$\forall\, X{:}K.\, T$	implicit product over types
		$\lambda\, x{:}S.\, T$	term-to-type function
		$\lambda\, X{:}K.\, T$	type-to-type function
		$T\ t$	type-to-term application
		$T\ \cdot S$	type-to-type application
		$\iota\, x{:}T.\, T'$	dependent intersection
		$\{p_1 \simeq p_2\}$	equality of untyped terms
		$\boxed{D}$	datatypes
Terms s, t	$::=$	x	variables
		$\lambda\, x.\, t$	term abstraction
		$\Lambda\, x.\, t$	erased term abstraction
		$\Lambda\, X.\, t$	type abstraction
		$t\ s$	term application
		$t\ \text{-}s$	erased term application
		$t \cdot T$	type application
		$[t, s]$	intro dependent intersection
		$t.1$	dep. intersection left projection
		$t.2$	dep. intersection right projection
		β	reflexivity of equality
		$\rho\ t\ @x.T\ -s$	rewrite by equality
		$\varphi\ t - t_1\ \{t_2\}$	cast by equality
		$\delta\ T - t$	anything by absurd equality
		$\boxed{c}$	data constructors
		$\boxed{\mu\ x.\ s\ @P\ \{\ c_i\ \overline{a_i} \to t_i\}_{i=1..n}}$	recursive def. over datatype
		$\boxed{\sigma\texttt{<}s_1\texttt{>}\ s_2\ @P\ \{\ c_i\ \overline{a_i} \to t_i\}_{i=1..n}}$	case analysis over datatype
Argument Sequence $\overline{s}$	$::=$	$\emptyset \mid s\ \overline{s} \mid \cdot S\ \overline{s} \mid \text{-}s\ \overline{s}$	for constructor patterns and applications
Variable Sequence $\overline{a}$	$::=$	$\emptyset \mid a\ \overline{a} \mid \cdot X\ \overline{a} \mid \text{-}a\ \overline{a}$	

Figure 3: Syntax for Cedille kinds, types, terms

$$\text{Typing contexts } \Gamma \quad ::= \quad \emptyset \mid \Gamma, x{:}T \mid \Gamma, X{:}K \mid \Gamma, \text{IndEl}[D, R, \Delta, m, L, \Theta, \mathcal{E}]$$

Figure 4: Contexts

0.3 Erasure

$$
\begin{array}{lcl}
|x| & = & x \\
|\star| & = & \star \\
|\{t \simeq t'\}|| & = & \{|t| \simeq |t'|\} \\
|\beta| & = & \lambda\, x.\, x \\
|\delta\ T - t| & = & |t| \\
|\varphi\ t\ \text{-}\ t'\ \{t''\}| & = & |t''| \\
|\rho\ t'\ @\ x.T\ \text{-}\ t| & = & |t| \\
|\iota\, x\!:\!T.\, T'| & = & \iota\, x\!:\!|T|.\, |T'| \\
|[t, t']| & = & |t| \\
|t.1| & = & |t| \\
|t.2| & = & |t| \\
|\Pi\, x\!:\!T.\, T'| & = & \Pi\, x\!:\!|T|.\, |T'| \\
|\lambda\, x.\, t| & = & \lambda\, x.\, |t| \\
|t\ t'| & = & |t|\ |t'| \\
|\forall\, x\!:\!T.\, T'| & = & \forall\, x\!:\!|T|.\, |T'| \\
|\Lambda\, x\!:\!T.\, t| & = & |t| \\
|t\ \text{-}t'| & = & |t| \\
|\forall\, X\!:\!K.\, T| & = & \forall\, X\!:\!|K|.\, |T| \\
|\Lambda\, X\!:\!K.\, t| & = & |t| \\
|t\ \cdot T| & = & |t| \\
|\Pi\, x\!:\!T.\, K| & = & \Pi\, x\!:\!|T|.\, |K| \\
|\lambda\, x\!:\!T.\, T'| & = & \lambda\, x\!:\!|T|.\, |T'| \\
|T\ t| & = & |T|\ |t| \\
|\Pi\, X\!:\!K.\, K'| & = & \Pi\, X\!:\!|K|.\, |K'| \\
|\lambda\, X\!:\!K.\, T| & = & \lambda\, X\!:\!|K|.\, |T| \\
|T \cdot T'| & = & |T| \cdot |T'| \\
|c| & = & c \\
|\mu\ x.\ t\ @P\ \{\ c_i\ \overline{a_i} \to t_i\}_{i=1..n}| & = & \mu\ x.\ |t|\ \{\ c_i\ \overline{|a_i|} \to |t_i|\}_{i=1..n} \\
|\sigma\texttt{<}s\texttt{>}\ t\ @P\ \{\ c_i\ \overline{a_i} \to t_i\}_{i=1..n}| & = & \sigma\ |t|\ \{c_i\ \overline{|a_i|} \to |t_i|\}_{i=1..n}
\end{array}
$$

$$
\begin{array}{lcl}
|\varnothing| & = & \varnothing \\
|s\ \overline{s}| & = & |s|\ |\overline{s}| \\
|\text{-}s\ \overline{s}| & = & |\overline{s}| \\
|\cdot S\ \overline{s}| & = & |\overline{s}|
\end{array}
$$

Figure 5: Erasure for annotated terms

0.4 Vector notation

Notation $\#\overline{s}$ denotes the number of terms in the sequence $\overline{s}$, and similarly for $\#\overline{p}$ for untyped terms and $\#\overline{a}$ for variables. Variable sequences are assumed to contain no repeating variables.

Notation $\{c_i\ \overline{a_i} \to p_i\}_{i=1\ldots n}$ denotes a case tree with n branches, headed by n distinct constructors $(c_i)_{i=1\ldots n}$ and n variable sequences $(\overline{a_i})_{i=1\ldots n}$.

Notation $\overline{[s/a]}$ denotes the point-wise simultaneous and capture avoiding substitution of each expression in $\overline{s}$ with each variable in $\overline{a}$. These will always be checked to have the correct length and, in typed settings, that $\overline{s}$ can be classified by the telescope given by $\overline{a}$.

0.5 Other notation

We write $FV(t)$ (resp. $FV(T)$) to denote the set of free variables of term t (resp. type T). We write $DV(\Gamma)$ to denote the set of declared variables contained in context Γ (this applies also to constructor contexts, where $FV(\Delta)$ indicates the set of constructor identifiers).

0.6 Operational Semantics

The operational semantics for Cedille is defined on untyped terms extracted from erasure. We add to $\beta\eta$-conversion reduction rules for μ and σ and an axiom that identifiers of the form $\texttt{to/}D$ are convertible with $\lambda\, x.\, x$. (Identifiers with `/` are not legal for names for user defined terms, only for automatically generated names). For this proof appendix[1], constructors are convertible precisely when they have the same identifier.

$$\frac{1 \le j \le n \quad \#\overline{p} = \#\overline{a_j}}{\sigma\ (c_j\ \overline{p})\ \{c_i\ \overline{a_i} \to t_i\}_{i=1..n} \rightsquigarrow \overline{[p/a_j]}t_j} \qquad \frac{1 \le j \le n \quad \#\overline{p} = \#\overline{a_j} \quad q = \lambda\, x.\, \mu\ ih.\ x\ \{\ c_i\ \overline{a_i} \to q_i\}_{i=1..n}}{\mu\ y.\ (c_j\ \overline{p})\ \{\ c_i\ \overline{a_i} \to q_i\}_{i=1..n} \rightsquigarrow \overline{[p/a_j]}[q/y]q_j}$$

$$\frac{}{\texttt{to/}D = \lambda\, x.\, x}$$

Figure 6: Conversion rules for μ, σ, and identifiers of the form $\texttt{to/}D$

The convertibility relation $=_{\beta\eta\mu}$ is the reflexive transitive congruence closure of $\beta\eta$-reduction additionally equipped with the reduction rules of Figure 6. Constructors are convertible when they are identical. The convertibility relation for terms is lifted to types similarly to Stump [Stu18], with datatype identifiers also convertible when they are identical.

[1] Cedille actually does something different, but that's for a separate paper on zero-cost reuse between datatypes in the surface language, c.f. [DFS18].

0.7 CDLE derivations

For simplicity, we treat all CDLE typing constructs presented axiomatically as globally available, i.e., they do not need to appear in the local typing context to be used. All such types and terms are closed and could be replaced in proofs with their direct definitions.

$$\frac{\Gamma \vdash S : \star \quad \Gamma \vdash T : \star}{\Gamma \vdash \mathsf{Cast} \cdot S \cdot T : \star} \qquad \frac{\Gamma \vdash t : \mathsf{Cast} \cdot S \cdot T}{\Gamma \vdash \mathsf{elimCast}\ \text{-}t : S \to T}$$

$$\frac{\Gamma \vdash t_1 : S \to T \quad \Gamma \vdash t_2 : \Pi\, x : S.\, \{f\ x \simeq x\}}{\Gamma \vdash \mathsf{intrCast}\ \text{-}t_1\ \text{-}t_2 : \mathsf{Cast} \cdot S \cdot T}$$

$$\frac{\Gamma \vdash S : \star}{\Gamma \vdash \mathsf{castRefl} \cdot S : \mathsf{Cast} \cdot S \cdot S} \qquad \frac{\Gamma \vdash t_1 : \mathsf{Cast} \cdot S \cdot T \quad \Gamma \vdash t_2 : \mathsf{Cast} \cdot T \cdot U}{\Gamma \vdash \mathsf{castTrans}\ \text{-}t_1\ \text{-}t_2 : \mathsf{Cast} \cdot S \cdot U}$$

$$\frac{\Gamma \vdash F : \star \to \star}{\Gamma \vdash \mathsf{Mono} \cdot F : \star} \qquad \frac{\Gamma \vdash t_1 : \mathsf{Mono} \cdot F \quad \Gamma \vdash t_2 : \mathsf{Cast} \cdot S \cdot T}{\Gamma \vdash \mathsf{elimMono}\ \text{-}t_1\ \text{-}t_2 : F \cdot S \to F \cdot T}$$

$$\frac{\Gamma \vdash t : \forall\, X : \star.\, \forall\, Y : \star.\, \mathsf{Cast} \cdot X \cdot Y \Rightarrow \mathsf{Cast} \cdot (F \cdot X) \cdot (F \cdot Y)}{\Gamma \vdash \mathsf{intrMono}\ \text{-}t : \mathsf{Mono} \cdot F}$$

$$|\mathsf{elimCast}\ \text{-}t| \quad = \quad \lambda\, x.\, x, \qquad |\mathsf{elimMono}\ \text{-}t_1\ \text{-}t_2| \quad = \quad \lambda\, x.\, x$$

Figure 7: Axiomatic presentation of inclusions and positivity

Proposition 3 (Firsov et al. [FBS18])**.** *For $F : \star \to \star$ and $m : \mathsf{Mono} \cdot F$:*

- *For all term t_1 and t_2, $|\mathsf{ind}\ t_1\ (\mathsf{in}\ t_2)| =_{\beta\eta} |t_1\ (\mathsf{ind}\ t_1)\ t_2|$*
- *For all terms t, $|\mathsf{out}\ (\mathsf{in}\ t)| =_{\beta\eta} |t|$.*
- *If t is a term of type μF then $\{\mathsf{in}\ \text{-}m\ (\mathsf{out}\ \text{-}m\ t) \simeq t\}$ is provable.*

For our termination guarantee of elaborations of datatype expressions, we must record an additional fact: the impredicative encoding of datatypes can be included into a function type.

Lemma 1. *For all closed definitions of $F : \star \to \star$ and $m : \mathtt{Mono} \cdot F$, there exists some closed term t' of type $\mu F \to \Pi\, x : T_1.\, T_2$ for some T_1 and T_2 such that $|t'| =_{\beta\eta} \lambda\, x.\, x$*

Proof. Consulting Firsov et al. [FBS18], one sees that μF is formed of a dependent intersection type whose first component is

$$\forall\, X : \star.\, (\forall\, R : \star.\,)(R \to X) \to F \cdot R \to X$$

It is then clear that $\lambda\, x.\, x.1 \cdot (\forall\, X : \star.\, X \to X)$ has a type of the desired form and erases to $\lambda\, x.\, x$. □

Theorem 1 (Characterization)**.** *For all $F : \star \to \star$ and $m : \mathsf{Mono} \cdot F$:*

- *for all terms t_1 and t_2, $|\mathsf{indCV}\ t_1\ (\mathsf{inCV}\ t_2)| =_{\beta\eta} |t_1\ \mathsf{outCV}\ (\mathsf{indCV}\ t_1)\ t_2|$;*
- *for all terms t, $|\mathsf{outCV}\ (\mathsf{inCV}\ t)| =_{\beta\eta} |t|$*
- *for all terms t of type $\mu(\mathsf{CV} \cdot F)$, $\{\mathsf{inCV}\ \text{-}m\ (\mathsf{outCV}\ \text{-}mt) \simeq t\}$ is provable.*

Proof. Given by resp. `indCVHom`, `lambekCV1`, and `lambekCV2`. □

```
module util.Cast .

CastCod : Π A: ⋆. Π B: ⋆. A → ⋆ = λ A: ⋆. λ B: ⋆. λ a: A. Σ b: B. { b ≃ a } .

Cast : Π A: ⋆. Π B: ⋆. ⋆ = λ A: ⋆. λ B: ⋆. Π a: A. CastCod ·A ·B a .

intrCast : ∀ A: ⋆. ∀ B: ⋆. ∀ f: A → B. (Π x: A. {f x ≃ x}) ⇒ Cast ·A ·B
= Λ A. Λ B. Λ f. Λ eq. λ a. (φ (eq a) - (f a) {|a|}, β) .

elimCast : ∀ A: ⋆. ∀ B: ⋆. Cast ·A ·B ⇒ A → B =
  Λ A. Λ B. Λ c. λ a. φ (π2 (c a)) - (π1 (c a)) {|a|}.

_ : { elimCast ≃ λ x. x } = β .

castRefl : ∀ A: ⋆. Cast ·A ·A = Λ A. intrCast -(λ x. x) -(λ x. β) .

castTrans : ∀ A: ⋆. ∀ B: ⋆. ∀ C: ⋆. Cast ·A ·B ⇒ Cast ·B ·C ⇒ Cast ·A ·C
= Λ A. Λ B. Λ C. Λ c1. Λ c2. intrCast -(λ x. elimCast -c2 (elimCast -c1 x)) -λ x. β .

Mono : (⋆ → ⋆) → ⋆ = λ F: ⋆ → ⋆. ∀ X: ⋆. ∀ Y: ⋆. Cast ·X ·Y ⇒ Cast ·(F ·X) ·(F ·Y).

intrMono : ∀ F: ⋆ → ⋆. (∀ X: ⋆. ∀ Y: ⋆. Cast ·X ·Y ⇒ Cast ·(F ·X) ·(F ·Y)) ⇒ Mono ·F
= Λ F. Λ f. Λ X. Λ Y. Λ c. intrCast -(elimCast -(f -c)) -λ x. β .

elimMono : ∀ F: ⋆ → ⋆. Mono ·F ⇒ ∀ X: ⋆. ∀ Y: ⋆. Cast ·X ·Y ⇒ F ·X → F ·Y
= Λ F. Λ mono. Λ X. Λ Y. Λ c. elimCast -(mono -c) .

_ : { elimMono ≃ λ x. x } = β .
```

Figure 8: CDLE derivation of inclusions and positivity

$$\frac{\Gamma \vdash F : \star \to \star}{\Gamma \vdash \mu F : \star} \qquad \frac{\Gamma \vdash t_1 : \mathsf{Mono} \cdot F \quad \Gamma \vdash t_2 : F \cdot \mu F}{\Gamma \vdash \mathsf{in}\ \text{-}t_1\ t_2 : \mu F} \qquad \frac{\Gamma \vdash t_1 : \mathsf{Mono} \cdot F \quad \Gamma \vdash t_2 : \mu F}{\Gamma \vdash \mathsf{out}\ \text{-}t_1\ t_2 : F \cdot \mu F}$$

$$\begin{aligned} &\mathsf{PrfAlg} : \Pi\, F : \star \to \star.\, \mathsf{Mono} \cdot F \to (\mu F \to \star) \to \star \\ &\mathsf{PrfAlg} \cdot F\ t \cdot P = \quad \forall\, R : \star.\, \forall\, c : \mathsf{Cast} \cdot R \cdot \mu F.\, (\Pi\, x : R.\, P\ (\mathsf{elimCast}\ \text{-}c\ x)) \\ &\qquad\qquad \to \Pi\, xs : F \cdot R.\, P\ (\mathsf{in}\ \text{-}t\ (\mathsf{elimMono}\ \text{-}t\ \text{-}c\ xs)) \end{aligned}$$

$$\frac{\Gamma \vdash t_1 : \mathsf{Mono} \cdot F \quad \Gamma \vdash t_2 : \mathsf{PrfAlg} \cdot F\ t_1 \cdot P}{\Gamma \vdash \mathsf{ind}\ \text{-}t_1\ t_2 : \Pi\, x : \mu F.\, P\ x} \qquad \begin{aligned} |\mathsf{out}\ (\mathsf{in}\ t)| &=_{\beta\eta} |t| \\ |\mathsf{ind}\ t_1\ (\mathsf{in}\ t_2)| &=_{\beta\eta} |t_1\ (\mathsf{ind}\ t_1)\ t_2| \end{aligned}$$

Figure 9: Interface for encodings of inductive types provided by [FBS18]

$$\frac{\Gamma \vdash H : \star \to \star \quad \Gamma \vdash F : \star \to \star}{\Gamma \vdash \mathsf{RExt} \cdot H \cdot F : \star} \qquad \frac{\Gamma \vdash S : \star \quad \Gamma \vdash s : H \cdot S \quad \Gamma \vdash t : F \cdot S}{\Gamma \vdash \mathsf{pack} \cdot S \text{ -}s\ t : \mathsf{RExt} \cdot H \cdot F}$$

$$\frac{\Gamma \vdash P : \mathsf{RExt} \cdot H \cdot F \to \star \quad \Gamma \vdash t : \forall\, X : \star .\, \forall\, x : H \cdot X .\, \Pi\, y : F \cdot X .\, P\ (\mathsf{pack} \cdot X \text{ -}x\ y)}{\Gamma \vdash \mathsf{unpack} \cdot P\ t : \Pi\, z : \mathsf{RExt} \cdot H \cdot F .\, P\ z}$$

$$|\mathsf{pack} \cdot S \text{ -}s\ t| \;=\; (\lambda\, x .\, \lambda\, f .\, f\ x)\ |t|, \quad |\mathsf{unpack} \cdot P\ t| \;=\; (\lambda\, f .\, \lambda\, x .\, x\ f)\ |t|$$

Figure 10: Axiomatic presentation of implicily restricted existential types

```
module util.RExt .

RExtC : (⋆ → ⋆) → (⋆ → ⋆) → ⋆
= λ H: ⋆ → ⋆. λ F: ⋆ → ⋆.
  ∀ X: ⋆. (∀ R: ⋆. H ·R ⇒ F ·R → X) → X .

packC : ∀ H: ⋆ → ⋆. ∀ F: ⋆ → ⋆. ∀ R: ⋆. H ·R ⇒ F ·R → RExtC ·H ·F
= Λ H. Λ F. Λ R. Λ r. λ xs. Λ X. λ f. f -r xs .

unpackC : ∀ H: ⋆ → ⋆. ∀ F: ⋆ → ⋆. ∀ X: ⋆. (∀ R: ⋆. H ·R ⇒ F ·R → X) → RExtC ·H ·F → X
= Λ H. Λ F. Λ X. λ f. λ x. x f .

RExtI : Π H: ⋆ → ⋆. Π F: ⋆ → ⋆. RExtC ·H ·F → ⋆
= λ H: ⋆ → ⋆. λ F: ⋆ → ⋆. λ x: RExtC ·H ·F.
  ∀ P: RExtC ·H ·F → ⋆.
  (∀ R: ⋆. ∀ r: H ·R. Π xs: F ·R. P (packC -r xs)) → P x .

RExt : (⋆ → ⋆) → (⋆ → ⋆) → ⋆
= λ H: ⋆ → ⋆. λ F: ⋆ → ⋆. ι x: RExtC ·H ·F. RExtI ·H ·F x .

pack : ∀ H: ⋆ → ⋆. ∀ F: ⋆ → ⋆. ∀ R: ⋆. H ·R ⇒ F ·R → RExt ·H ·F
= Λ H. Λ F. Λ R. Λ r. λ xs. [ packC -r xs , Λ P. λ f. f -r xs ] .

_ : { pack ≃ λ xs. λ f. f xs } = β .

unpack : ∀ H: ⋆ → ⋆. ∀ F: ⋆ → ⋆. ∀ P: RExt ·H ·F → ⋆.
  (∀ R: ⋆. ∀ r: H ·R. Π xs: F ·R. P (pack -r xs)) → Π x: RExt ·H ·F. P x
= Λ H. Λ F. Λ P. λ f. λ x.
  x.2 ·(λ x: RExtC ·H ·F. ∀ y: RExt ·H ·F. ∀ eq:  y ≃ x . P (φ eq - y |x|))
    (Λ R. Λ r. λ xs. Λ y. Λ eq. f -r xs)
    -x -β'' .

_ : { unpack ≃ λ f. λ x. x f } = β .
```

Figure 11: CDLE derivation of implicitly restricted existential types

$$\mathsf{Top} = \{\lambda\, x.\, x \simeq \lambda\, x.\, x\} \qquad \frac{\Gamma \vdash S : \star \quad \Gamma \vdash t : \mathsf{Top}}{\Gamma \vdash \mathsf{View} \cdot S\ t}$$

$$\frac{\Gamma \vdash t : \mathsf{Top} \quad \Gamma \vdash t_1 : S \quad \Gamma \vdash t_2 : \{t_1 \simeq t\}}{\Gamma \vdash \mathsf{intrView}\ t\ \text{-}t_1\ \text{-}t_2 : \mathsf{View} \cdot S\ t} \qquad \frac{\Gamma \vdash t_1 : \mathsf{Top} \quad \Gamma \vdash t_2 : \mathsf{View} \cdot S\ t_1}{\Gamma \vdash \mathsf{elimView}\ t_1\ \text{-}t_2 : S}$$

$$|\mathsf{intrView}\ t\ \text{-}t_1\ \text{-}t_2| \;=\; (\lambda\, x.\, x)\ |t|, \qquad |\mathsf{elimView}\ t_1\ \text{-}t_2| \;=\; (\lambda\, x.\, x)\ |t_1|$$

Figure 12: Axiomatic presentation of singleton types

```
module util.view.

Top : ⋆ =  λ x. x ≃ λ x. x  .

View : Π A: ⋆. Top → ⋆ = λ A: ⋆. λ x: Top. ι a: A. { a ≃ x } .

intrView : Π x: Top. ∀ A: ⋆. ∀ a: A. { a ≃ x } ⇒ View ·A x
= λ x. Λ A. Λ a. Λ eq. [ φ eq - a {|x|} , β{|x|} ] .

_ : { intrView ≃ λ x. x } = β .

elimView : Π x: Top. ∀ A: ⋆. View ·A x ⇒ A
= λ x. Λ A. Λ v. φ v.2 - v.1 {|x|}.

_ : { elimView ≃ λ x. x } = β .
```

Figure 13: CDLE derivation of singleton types

```
import util.Cast .
import util.View .

module cv.CVF (F: ⋆ → ⋆) {mono: Mono ·F} .

import encoding.FixIndM . {- Comment: [FSB18] -}

import util.RExt .
import util.Sigma .
import util.Top .

outCVU : Top = β{| λ x. unpack (λ xs. xs) (out x) |} .

RCV : ⋆ → ⋆ → ⋆
= λ X: ⋆. λ R: ⋆. (Cast ·R ·X) × View ·(R → F ·R) outCVU .

CV : ⋆ → ⋆ = λ X: ⋆. RExt ·(RCV ·X) ·F .

mCV : Mono ·CV = intrMono -(Λ X. Λ Y. Λ c.
  intrCast -(unpack Λ R. Λ r. λ xs.
        pack -((castTrans -(π1 r) -c), π2 r) xs)
    -(unpack Λ R. Λ r. λ xs. β)) .

outCV : μ CV → F ·(μ CV) = λ x.
  unpack (Λ R. Λ r. λ xs. elimMono -m -(π1 r) xs) (out -mCV x) .

inCV : F ·(μ CV) → μ CV = λ xs. in -mCV
    (pack -(castRefl, intrView outCVU -outCV -β) xs) .

PrfAlgCV : (μ CV → ⋆) → ⋆ = λ P: μ CV → ⋆.
  ∀ R: ⋆. ∀ c: Cast ·R ·(μ CV). View ·(R → F ·R) outCVU →
  (Π x: R. P (elimCast -c x)) →
  Π xs: F ·R. P (inCV (elimMono -m -c xs)) .

indCV : ∀ P: μ CV → ⋆. PrfAlgCV ·P → Π x: μ CV. P x = Λ P. λ alg.
  ind -mCV Λ R. Λ c. λ ih. unpack Λ S. Λ r. λ xs.
      alg ·S -(castTrans -(π1 r) -c)
        (intrView outCVU -(elimView outCVU -(π2 r)) -β)
        (λ x. ih (elimCast -(proj1 r) x)) xs .

lambekCV1 : ∀ x: F ·(μ CV). { outCV (inCV x) ≃ x } = Λ x. β .
lambekCV2 : ∀ x: μ CV. { inCV (outCV x) ≃ x } = Λ x.
  [pf : { inCV (outCV x) ≃ x }
    = indCV ·(λ y: μ CV. { inCV (outCV y) ≃ y })
        (Λ R. Λ c. λ o. λ ih. λ xs. β) x]
- ρ pf - β .

indCVHom : ∀ P: μ CV → ⋆. ∀ alg: PrfAlgCV ·P. ∀ xs: F ·(μ CV).
  { indCV alg (inCV xs) ≃ alg outCV (indCV alg) xs }
= Λ P. Λ alg. Λ xs. β .
```

Figure 14: Generic encoding of course-of-values datatypes in CDLE

```
import util.Cast .
import util.View .
import util.Sigma .
import util.RExt .

module cv.MuSigma (F: ⋆ → ⋆) m: Mono ·F .

import cv.CVF ·F -m .
import encoding.FixIndM ·CVF -monoCVF .
{- Comment: [FSB18] -}

Case : (μ CV → ⋆) → Π R: ⋆. RCV ·(μ CV) ·R → ⋆
= λ P: μ CV → ⋆. λ R: ⋆. λ is: RCV ·(μ CV) ·R.
  Π xs: F ·R. P (inCV (elimMono -m -(π1 is) xs)) .

sigma : ∀ P: μ CV → ⋆. ∀ R: ⋆. ∀ is: RCV ·(μ CV) ·R.
  Π x: R. Case ·P ·R is → P (elimCast -(π1 is) x)
= Λ P. Λ R. Λ is. λ x. λ f.
  ρ ς (lambekCV2 -(elimCast -(π1 is) x)) - f (elimView outCVU -(π2 is) x) .

_ : ∀ P: μ CV → ⋆. ∀ R: ⋆. ∀ is: RCV ·(μ CV) ·R.
  ∀ f: Case ·P ·R is. ∀ xs: F ·R. { sigma (inCV xs) f ≃ f xs }
= Λ P. Λ R. Λ is. Λ f. Λ xs. β .

AlgMu : (μ CV → ⋆) → ⋆ = λ P: μ CV → ⋆. ∀ R: ⋆. ∀ is: RCV ·(μ CV) ·R.
  (Π x: R. P (elimCast -(π1 is) x)) → Case ·P ·R is .

mu : ∀ P: μ CV → ⋆. Π x: μ CV. AlgMu ·P → P x
= Λ P. λ x. λ f. ind ·P
    (Λ R. Λ c. λ ih.
     unpack ·(RCV ·R) ·F ·(λ x: CVF ·R. P (in (elimMono -monoCVF -c x)))
       Λ S. Λ r. λ xs.
       f ·S
         -(pair (castTrans -(π1 r) -c) (π2 r))
         (λ x. ih (elimCast -(π1 r) x)) xs) x .

muComp : ∀ t1: Top. ∀ t2: Top. { mu (inCV t1) t2 ≃ t2 (λ x. mu x t2) t1 }
= Λ t1. Λ t2. β .

sigComp : ∀ t1: Top. ∀ t2: Top. { sigma (inCV t1) t2 ≃ t2 t1 }
= Λ t1. Λ t2. β .

sigExt : ∀ x: IndCV. { sigma x inCV ≃ x } = Λ x. ρ (lambekCV2 -x) - β .
```

Figure 15: CDLE derivation of sigma and mu

Theorem 2 (Characteriation (`mu`, `sigma`))**.**

- *For all terms* t_1 *and* t_2, $|\mathsf{mu}\ (\mathsf{inCV}\ t_1)\ t_2| =_{\beta\eta} |t_2\ (\lambda\, x.\, \mathsf{mu}\ x\ t_2)\ t_1|$, *where* $x \notin FV(t_2)$
- *For all terms* t_1 *and* t_2, $|\mathsf{sigma}\ (\mathsf{inCV}\ t_1)\ t_2| =_{\beta\eta} |t_2\ t_1|$
- *For all terms* t *of type* $\mu\mathsf{CV}$, $\{\mathsf{sigma}\ t\ \mathsf{inCV} \simeq t\}$ *is provable.*

Proof. These proofs are listed in Figure 15 as muComp, sigComp, and sigExt. Notice that the computation rules hold for all Top-typed terms □

0.8 Elaboration Rules

Notation 1. *Let $\overline{a}$ be a variable sequence, $\overline{s}$ be an expression sequence, and $\overline{A}$ be a classifier sequence (i.e., types and kinds).*

- *We write $\overline{a:A}$ to denote a telescope.*
  For example if $\overline{a} = x_1,\ \text{-}x_2,\ \cdot X$ and $\overline{A} = T_1, T_2, K$ then
  $$\overline{a:A} = x_1 : T_1, \text{-}x_2 : T_2, \cdot X : K$$
- *We write $\overset{\Pi}{\forall}\ \overline{a:A}.$ for erasure-respecting quantification in types.*
  Let $\overline{a}$ and $\overline{A}$ be as above, then
  $$\overset{\Pi}{\forall}\ \overline{a:A}.\, D = \Pi\, x_1 : T_1.\, \forall\, x_2 : T_2.\, \forall\, X : K.\, D$$
- *We write $\overset{\lambda}{\Lambda}\ \overline{a:A}.$ to indicate erasure-respecting term level abstraction.*
  Let $\overline{a}$ and $\overline{A}$ be as above, then
  $$\overset{\lambda}{\Lambda}\ \overline{a:A}.\, t = \lambda\, x_1 : T_1.\, \Lambda\, x_2 : T_2.\, \Lambda\, X : K.\, t$$
- *We write $t\ \overline{s}$ to represent erasure- and level- respecting application of t to $\overline{s}$*
  Let $\overline{s}$ be as $\overline{a}$ above, then
  $$t\ \overline{s} = t\ x_1\ \text{-}x_2\ \cdot X$$

Definition 1 (Datatype declaration)**.** *A* datatype declaration *is a triple, written $Ind[D, R, \Delta]$, where:*

- *D (the datatype name) is a unique identifier supplied by the user;*
- *R is a freshly generated type variable; and*
- *Δ is a context associating the $\#\Delta$ unique constructor identifiers $(c_i)_{i=1\ldots\#\Delta}$ to their type signatures*
  - *if $c \in DV(\Delta)$ is a constructor declared in Δ, we require that the type associated to c in Δ, $\Delta(c)$, is of the form $\overset{\Pi}{\forall}\ \overline{a : A}.\, D$*
  - *every occurrence of D in the user-supplied constructor argument types $\overline{a:A}[D/R]$ is in Δ replaced by the fresh variable R.*

We write $Ind[D, R, \Delta]$ wf if this declaration satisfies the above criteria.

Example 1. *Natural numbers are declared in Cedille as*

```
data Nat : ⋆ = zero : Nat | succ : Nat → Nat .
```

would be represented by

$$Ind[\mathtt{Nat}, R,\ \begin{array}{lcl} \mathtt{zero} & : & \mathtt{Nat} \\ \mathtt{suc} & : & R \to \mathtt{Nat} \end{array}\ ]$$

Restricted existentials for specific H and F can be declared in Cedille as

```
data RExt : ⋆ = pack : ∀ X: ⋆. H ·X ⇒ F ·X → RExt .
```

and would be represented by

$$Ind[\mathtt{RExt}, R,\ \mathtt{pack}\ :\ \forall\, X : \star.\, H \cdot X \Rightarrow F \cdot X \to \mathtt{RExt}\ ]$$

The elaboration IndEl$[D, R, \Delta, m, L, \Theta, \mathcal{E}]$ of a delcaration IndEl$[D, R, \Delta$,,c,a,r]ries the same information, as well as

- a monotonicity witness m that the elaborated signature of datatype D is positive
- a predicate transformer L used for deriving the induction principle for the particular datatype D
- additional global declarations given in Θ for termination checking

- a map $\mathcal{E}$ associating all exported definitions to their elaborations in CDLE

We require that the type of the constructor argument is well-kinded and that R occurs only positively within it. We describe the algorithm for checking these requirements by a collection of judgments whose inference rules are mutually inductively defined.

We remark that some inference rules have premises in the form $(\Gamma \vdash \overset{\Pi}{\forall}\, \overline{a_i : A_i}.\, T : \star \hookrightarrow \_)_{i=1..\#\Delta}$, accompanied by a premise $(c_i : \overset{\Pi}{\forall}\, \overline{a_i : A_i}.\, D \in \Delta)_{i=1..\#\Delta}$; the first indicates a family of derivations of the parenthesized judgment indexed by the ith constructor of Δ and its constructor argument telescope $\overline{a_i : A_i}$, and the second names these telescopes explicitly, and also carries with it the assumption that $FV(\Delta) = \{c_i \mid i = 1 \ldots \Delta\}$. A premise of the form $c_j : \overset{\Pi}{\forall}\, \overline{a_j : A_j}.\, D \in \Delta$ indicates not only that the constructor c_j is declared in Δ with the given classifier, but that it is the jth constructor, i.e., $1 \leq j \leq \#\Delta$. *Italics* indicates meta-variables, `teletype` font indicates code literals (except in meta-variables denoting generated names like `Is/`D), and $^{\text{normal text superscript}}$ denotes labels for meta-variables.

We use the following naming convention for expressions elaborated from datatypes and their constructors: D^{F} for the usual impredicative encoding of a signature functor of D; D^{FI} for the intersected version supporting functor induction; and D^{FIX} for the least fixpoint of the inductive functor.

(a) $\boxed{\Gamma \vdash K \hookrightarrow K'}$ Kind elaboration

$$\frac{}{\Gamma \vdash \star \hookrightarrow \star} \qquad \frac{\Gamma \vdash K_1 \hookrightarrow K_1' \quad \Gamma, X : K_1 \vdash K_2 \hookrightarrow K_2'}{\Gamma \vdash \Pi\, X : K_1.\, K_2 \hookrightarrow \Pi\, X : K_1'.\, K_2'} \qquad \frac{\Gamma \vdash T : K_2 \hookrightarrow T' \quad \Gamma, x : T \vdash K_1 \hookrightarrow K_1'}{\Gamma \vdash \Pi\, x : T.\, K_1 \hookrightarrow \Pi\, x : T'.\, K_1'}$$

(b) $\boxed{\Gamma \vdash T : K \hookrightarrow T'}$ Type Elaboration (congruence rules)

$$\frac{FV(p_1\ p_2) \subseteq dom(\Gamma) \quad \Gamma \vdash p_1 \hookrightarrow p_1' \quad \Gamma \vdash p_2 \hookrightarrow p_2'}{\Gamma \vdash \{p_1 \simeq p_2\} : \star \hookrightarrow \{p_1' \simeq p_2'\}} \qquad \frac{\Gamma \vdash T_1 : \star \hookrightarrow T_1' \quad \Gamma, x : T_1 \vdash T_2 : \star \hookrightarrow T_2'}{\Gamma \vdash \iota\, x : T_1.\, T_2 : \star \hookrightarrow \iota\, x : T_1'.\, T_2'}$$

$$\frac{\Gamma \vdash T_1 : \star \hookrightarrow T_1' \quad \Gamma, x : T_1 \vdash T_2 : \star T_2'}{\Gamma \vdash \forall\, x : T_1.\, T_2 : \star \hookrightarrow \forall\, x : T_1'.\, T_2'} \qquad \frac{\Gamma \vdash T_1 : \star \hookrightarrow T_1' \quad \Gamma, x : T_1 \vdash T_2 : \star \hookrightarrow T_2'}{\Gamma \vdash \Pi\, x : T_1.\, T_2 : \star \hookrightarrow \Pi\, x : T_1'.\, T_2'}$$

$$\frac{\Gamma \vdash K \hookrightarrow K' \quad \Gamma, X : K \vdash T : \star \hookrightarrow T'}{\Gamma \vdash \forall\, X : K.\, T : \star \hookrightarrow \forall\, X : K'.\, T'} \qquad \frac{\Gamma \vdash S : K_1 \hookrightarrow S' \quad \Gamma, x : S \vdash T : K_2 \hookrightarrow T'}{\Gamma \vdash \lambda\, x : S.\, T : \Pi\, x : S.\, K_2 \hookrightarrow \lambda\, x : S'.\, T'}$$

$$\frac{\Gamma \vdash K_1 \hookrightarrow K_1' \quad \Gamma, X : K_1 \vdash T : K_2 \hookrightarrow T'}{\Gamma \vdash \lambda\, X : K_1.\, T : \Pi\, X : K_1.\, K_2 \hookrightarrow \lambda\, X : K_1'.\, T'} \qquad \frac{\Gamma \vdash T_1 : \Pi\, X : K_2.\, K_1 \hookrightarrow T_1' \quad \Gamma \vdash T_2 : K_2 \hookrightarrow T_2'}{\Gamma \vdash T_1 \cdot T_2 : [T_2/X]K_1 \hookrightarrow T_1' \cdot T_2'}$$

$$\frac{\Gamma \vdash T : \Pi\, x : S.\, K \hookrightarrow T' \quad \Gamma \vdash t : S \hookrightarrow t'}{\Gamma \vdash T\ t : [t/x]K \hookrightarrow T'\ t'} \qquad \frac{}{\Gamma \vdash X : \Gamma(X) \hookrightarrow X}$$

Figure 16: Type and kind elaboration

(a) $\boxed{\vdash \Gamma \hookrightarrow \Gamma'}$ Elaboration of contexts

$$\frac{}{\vdash \emptyset \hookrightarrow \emptyset} \qquad \frac{\vdash \Gamma \hookrightarrow \Gamma' \quad \Gamma \vdash T : \star \hookrightarrow T'}{\vdash \Gamma, x{:}T \hookrightarrow \Gamma', x{:}T'}$$

$$\frac{\vdash \Gamma \hookrightarrow \Gamma' \quad \Gamma \vdash K \hookrightarrow K'}{\vdash \Gamma, X{:}K \hookrightarrow \Gamma', X{:}K'} \qquad \frac{\vdash \Gamma \hookrightarrow \Gamma'}{\vdash \Gamma, \mathrm{IndEl}[D, R, \Delta, m, L, \Theta, \mathcal{E}] \hookrightarrow \Gamma'}$$

(b) $\boxed{\Gamma\,;\, (\overline{a{:}A}) \vdash \overline{s} : (\overline{a{:}B}) \hookrightarrow \overline{s'}}$ Elaboration of type-coerced constructor argument telescope

$$\frac{}{\Gamma\,;\, \varnothing \vdash \varnothing : \varnothing \hookrightarrow \varnothing} \qquad \frac{\Gamma, x{:}S \vdash s : T \hookrightarrow s' \quad \Gamma, x{:}S\,;\, (\overline{a{:}A}) \vdash \overline{s} : ([s/x]\overline{a{:}B}) \hookrightarrow \overline{s'}}{\Gamma\,;\, (x{:}S, \overline{a{:}A}) \vdash s, \overline{s} : (x{:}T, \overline{a{:}B}) \hookrightarrow s', \overline{s'}}$$

$$\frac{\Gamma, x{:}S \vdash s : T \hookrightarrow S' \quad \Gamma, x{:}S\,;\, (\overline{a{:}A}) \vdash \overline{s} : ([s/x]\overline{a{:}B}) \hookrightarrow \overline{s'}}{\Gamma\,;\, (\text{-}x{:}S, \overline{a{:}A}) \vdash \text{-}s, \overline{s} : (\text{-}x{:}T, \overline{a{:}B}) \hookrightarrow \text{-}s', \overline{s'}}$$

$$\frac{\Gamma, X{:}K_1 \vdash S : K_2 \hookrightarrow S' \quad \Gamma, X{:}K_1\,;\, (\overline{a{:}A}) \vdash \overline{s} : ([S/X]\overline{a{:}B}) \hookrightarrow \overline{s'}}{\Gamma\,;\, (\cdot X{:}K_1, \overline{a{:}A}) \vdash \cdot S, \overline{s} : (\cdot X{:}K_2, \overline{a{:}B}) \hookrightarrow \cdot S', \overline{s'}}$$

Figure 17: Context and telescope elaboration

$$\frac{}{\Gamma \vdash x : \Gamma(x) \hookrightarrow x} \qquad \frac{\Gamma \vdash S : \star \hookrightarrow S' \quad \Gamma, x{:}S \vdash t : T \hookrightarrow t'}{\Gamma \vdash \lambda\, x.\, t : \Pi\, x{:}S.\, T \hookrightarrow \lambda\, x.\, t'}$$

$$\frac{\Gamma \vdash S : \star \hookrightarrow S' \quad x \notin FV(|t|) \quad \Gamma, x{:}S \vdash t : T \hookrightarrow t'}{\Gamma \vdash \Lambda\, x.\, t : \forall\, x{:}S.\, T \hookrightarrow \Lambda\, x.\, t'} \qquad \frac{\Gamma \vdash K \hookrightarrow K' \quad X \notin FV(|t|) \quad \Gamma, X{:}K \vdash t : T \hookrightarrow t'}{\Gamma \vdash \Lambda\, X.\, t : \forall\, X{:}K.\, T \hookrightarrow \Lambda\, X.\, t'}$$

$$\frac{\Gamma \vdash t : \Pi\, x{:}S.\, T \hookrightarrow t' \quad \Gamma \vdash s : S \hookrightarrow s'}{\Gamma \vdash t\ s : [s/x]T \hookrightarrow t'\ s'} \qquad \frac{\Gamma \vdash t : \forall\, X{:}K.\, T \hookrightarrow t' \quad \Gamma \vdash S : K \hookrightarrow S'}{\Gamma \vdash t \cdot S : [S/X]T \hookrightarrow t' \cdot S'}$$

$$\frac{\Gamma \vdash t : \forall\, x{:}S.\, T \hookrightarrow t' \quad \Gamma \vdash s : S \hookrightarrow s'}{\Gamma \vdash t\ \text{-}s : [s/x]T \hookrightarrow t'\ \text{-}s'} \qquad \frac{\Gamma \vdash t : S \hookrightarrow t' \quad S \cong T \quad \Gamma \vdash T : \star \hookrightarrow T'}{\Gamma \vdash t : T \hookrightarrow t'}$$

$$\frac{\Gamma \vdash t_1 : T_1 \hookrightarrow t_1' \quad \Gamma \vdash t_2 : [t_1/x]T_2 \hookrightarrow t_2' \quad t_1' \cong t_2'}{\Gamma \vdash [t_1, t_2] : \iota\, x{:}T_1.\, T_2 \hookrightarrow [t_1', t_2']}$$

$$\frac{\Gamma \vdash t : \iota\, x{:}T_1.\, T_2}{\Gamma \vdash t.1 : T_1 \hookrightarrow t'.1} \qquad \frac{\Gamma \vdash t : \iota\, x{:}T_1.\, T_2 \hookrightarrow t'}{\Gamma \vdash t.2 : [t.1/x]T_2 \hookrightarrow t'.2}$$

$$\frac{\Gamma \vdash \{t_1 \simeq t_1\} : \star \hookrightarrow \{p_1' \simeq p_1'\} \quad \Gamma \vdash |t_2| \hookrightarrow p_2'}{\Gamma \vdash \beta\langle t_1\rangle\{t_2\} : \{t_1 \simeq t_1\} \hookrightarrow \beta\langle p_1'\rangle\{p_2'\}} \qquad \frac{\begin{array}{cc}\Gamma \vdash s : \{|t_1| \simeq |t_2|\} \hookrightarrow s' & \Gamma \vdash t : [t_1/x]T_1 \hookrightarrow t' \\ T_1 \cong T_2 & \Gamma \vdash [t_2/x]T_2 : \star \hookrightarrow [t_2'/x]T_2'\end{array}}{\Gamma \vdash \rho\ s\ @\ x.T_2\ \text{-}\ t : [t_2/x]T_2 \hookrightarrow \rho\ s'\ @\ x.T_2'\ \text{-}\ t'}$$

$$\frac{\Gamma \vdash s : \{|t_1| \simeq |t_2|\} \hookrightarrow s' \quad \Gamma \vdash t_1 : T \hookrightarrow t_1' \quad \Gamma \vdash t_2 \hookrightarrow t_2'}{\Gamma \vdash \varphi\ s - t_1\ \{t_2\} : T \hookrightarrow \varphi\ s' - t_1'\ \{t_2'\}} \qquad \frac{\Gamma \vdash t : \{\lambda\, x.\, \lambda\, y.\, x \simeq \lambda\, x.\, \lambda\, y.\, x\} \hookrightarrow t' \quad \Gamma \vdash T : \star \hookrightarrow T'}{\Gamma \vdash \delta\ T - t : T \hookrightarrow \delta\ T' - t'}$$

Figure 18: $\boxed{\Gamma \vdash t : T \hookrightarrow t'}$ Term elaboration (congruence rules)

$$\frac{x \in DV(x)}{\Gamma \vdash x \hookrightarrow x} \qquad \frac{\Gamma, x : \{\lambda\, x.\, x \simeq \lambda\, x.\, x\} \vdash p \hookrightarrow p'}{\Gamma \vdash \lambda\, x.\, p \hookrightarrow \lambda\, x.\, p'} \qquad \frac{\Gamma \vdash p_1 \hookrightarrow p_1' \quad \Gamma \vdash p_2 \hookrightarrow p_2'}{\Gamma \vdash p_1\ p_2 \hookrightarrow p_1'\ p_2}$$

Figure 19: $\boxed{\Gamma \vdash p \hookrightarrow p'}$ Post-erasure term elaboration (congruence rules)

$$\frac{(c_i : \overset{\Pi}{\forall}\, \overline{a_i : A_i}.\, D \in \Delta)_{i=1..\#\Delta} \quad (\Gamma, R : \star, X : \star \vdash \overset{\Pi}{\forall}\, \overline{a_i : A_i}.\, X : \star \hookrightarrow \overset{\Pi}{\forall}\, \overline{a_i : A_i'}.\, X)_{i=1..\#\Delta}}{\Gamma \vdash \mathrm{Ind}[D, R, \Delta] \overset{\mathrm{F}}{\hookrightarrow} \lambda\, R.\, \forall\, X : \star.\, (\Pi\, x_i : \overset{\Pi}{\forall}\, \overline{a_i : A_i'}.\, X)_{i=1..\#\Delta}.\, X}\ \mathrm{F}$$

$$\frac{c_j : \overset{\Pi}{\forall}\, \overline{a_j : A_j}.\, D \in \Delta}{\Gamma \vdash (\mathrm{Ind}[D, R, \Delta], j) \overset{\mathrm{cF}}{\hookrightarrow} \Lambda\, R.\, \overset{\lambda}{\Lambda}\, \overline{a_j}.\, \Lambda\, X.\, \lambda\, x_{i=1..\#\Delta}.\, x_j\ \overline{a_j}}\ \mathrm{cF}$$

$$\frac{\begin{array}{c}\Gamma \vdash \mathrm{Ind}[D, R, \Delta] \overset{\mathrm{F}}{\hookrightarrow} D^{\mathrm{F}} \qquad (\Gamma \vdash (\mathrm{Ind}[D, R, \Delta], i) \overset{\mathrm{cF}}{\hookrightarrow}\ c_i^{\mathrm{F}})_{i=1..\#\Delta} \\ (c_i : \overset{\Pi}{\forall}\, \overline{a_i : A_i}.\, D \in \Delta)_{i=1..\#\Delta} \quad (\Gamma, R : \star, X : \star \vdash \overset{\Pi}{\forall}\, \overline{a_i : A_i}.\, X : \star \hookrightarrow \overset{\Pi}{\forall}\, \overline{a_i : A_i'}.\, X)_{i=1..\#\Delta}\end{array}}{\Gamma \vdash \mathrm{Ind}[D, R, \Delta] \overset{\mathrm{FI}}{\hookrightarrow} \lambda\, R.\, \iota\, x : D^{\mathrm{F}} \cdot R.\, \forall\, X : D^{\mathrm{F}} \cdot R \to \star.\, (\Pi\, x_i : \overset{\Pi}{\forall}\, \overline{a_i : A_i'}.\, X\ (c_i^{\mathrm{F}}\ \overline{a_i}))_{i=1..\#\Delta}.\, X\ x}\ \mathrm{FI}$$

$$\frac{\Gamma \vdash (\mathrm{Ind}[D, R, \Delta], j) \overset{\mathrm{cF}}{\hookrightarrow} c_j^F \quad c_j : \overset{\Pi}{\forall}\, \overline{a_j : A_j}.\, D \in \Delta}{\Gamma \vdash (\mathrm{Ind}[D, R, \Delta], j) \overset{\mathrm{cFI}}{\hookrightarrow} \Lambda\, R.\, \overset{\lambda}{\Lambda}\, \overline{a_j}.\, [c_j^{\mathrm{F}} \cdot R\ \overline{a_j}, \Lambda\, X.\, \lambda\, x_{i=1..\#\Delta}.\, x_j\ \overline{a_j}]}\ \mathrm{cFI}$$

$$\frac{\Gamma \vdash \mathrm{Ind}[D, R, \Delta] \overset{\mathrm{FI}}{\hookrightarrow}\ D^{\mathrm{FI}}}{\Gamma \vdash \mathrm{Ind}[D, R, \Delta] \overset{\mathrm{FIX}}{\hookrightarrow}\ \mu(\mathtt{CV} \cdot D^{\mathrm{FI}})}\ \mathrm{FIX}$$

$$\frac{\begin{array}{c}\Gamma \vdash \mathrm{Ind}[D, R, \Delta] \overset{\mathrm{FIX}}{\hookrightarrow} \mu(\mathtt{CV} \cdot D^{\mathrm{FI}}) \quad c_j : \overset{\Pi}{\forall}\, \overline{a_j : A_j}.\, D \in \Delta \\ \Gamma \vdash (\mathrm{Ind}[D, R, \Delta], j) \overset{\mathrm{cFI}}{\hookrightarrow}\ c_j^{\mathrm{FI}} \qquad \Gamma \vdash D^{\mathrm{FI}} \overset{+}{\hookrightarrow} m\end{array}}{\Gamma \vdash (\mathrm{Ind}[D, R, \Delta], j) \overset{\mathrm{cFIX}}{\hookrightarrow} \overset{\lambda}{\Lambda}\, \overline{a_j}.\ \mathtt{inCV}\ \text{-}m\ (c_j^{\mathrm{FI}}\ \overline{a_j})}\ \mathrm{cFIX}$$

$$\frac{\Gamma \vdash \mathrm{Ind}[D, R, \Delta] \overset{\mathrm{F}}{\hookrightarrow} D^{\mathrm{F}} \quad \Gamma \vdash \mathrm{Ind}[D, R, \Delta] \overset{\mathrm{FIX}}{\hookrightarrow} \mu(\mathtt{CV} \cdot D^{\mathrm{FI}}) \quad \Gamma \vdash D^{\mathrm{FI}} \overset{+}{\hookrightarrow} m}{\begin{array}{l}\Gamma \vdash \mathrm{Ind}[D, R, \Delta] \overset{\mathrm{Lift}}{\hookrightarrow} \lambda\, P : \mu(\mathtt{CV} \cdot D^{\mathrm{FI}}) \to \star.\, \lambda\, R : \star. \\ \quad \lambda\, r : \mathtt{RCV} \cdot D^{\mathrm{FI}} \cdot \mu(\mathtt{CV} \cdot D^{\mathrm{FI}}) \cdot R.\, \lambda\, x : D^{\mathrm{F}} \cdot R.\, \forall\, y : \mathtt{View} \cdot (D^{\mathrm{FI}} \cdot R)\ \beta\{x\}. \\ \quad\quad P\ (\mathtt{inCV}\ \text{-}m\ (\mathtt{elimMono}\ \text{-}m\ \text{-}(\pi_1\ r)\ (\mathtt{elimView}\ \beta\{x\}\ \text{-}y)))\end{array}}\ \textsc{Lift}$$

$$\frac{\begin{array}{c}\Gamma \vdash \mathrm{Ind}[D, R, \Delta]\ wf \quad \Gamma \vdash \mathrm{Ind}[D, R, \Delta] \overset{\mathrm{FIX}}{\hookrightarrow} \mu(\mathtt{CV} \cdot D^{\mathrm{FI}}) \\ (\Gamma \vdash (\mathrm{Ind}[D, R, \Delta], i) \overset{\mathrm{cFIX}}{\hookrightarrow} c_i^{\mathrm{FIX}})_{i=1..\#\Delta} \quad \Gamma \vdash D^{\mathrm{FI}} \overset{+}{\hookrightarrow} m \quad \Gamma \vdash \mathrm{Ind}[D, R, \Delta] \overset{\mathrm{Lift}}{\hookrightarrow} L\end{array}}{\Gamma \vdash \mathrm{Ind}[D, R, \Delta] \dashv \Gamma, \mathrm{IndEl}[D, R, \Delta, m, L, \Theta, \mathcal{E}]}$$

where
$$\begin{array}{rcl}\Theta & = & \mathtt{Is/}D : \star \to \star, \mathtt{is/}D : \mathtt{Is/}D \cdot D,\ \mathtt{to/}D = \lambda\, x.\, x : \forall\, R : \star.\, \forall\, is : \mathtt{Is/}D \cdot R.\, R \to D \\ \mathcal{E} & = & \left\{ \begin{array}{c} D \mapsto \mu(\mathtt{CV} \cdot D^{\mathrm{FI}}), \quad (c_i \mapsto c_i^{\mathrm{FIX}})_{i=1...\#\Delta} \quad \mathtt{Is/}D \mapsto \mathtt{RCV} \cdot D^{\mathrm{FI}} \cdot \mu(CV \cdot D^{\mathrm{FI}}), \\ \mathtt{is/}D \mapsto (\mathtt{castRefl} \cdot \mu(\mathtt{CV} \cdot D^{\mathrm{FI}}), \mathtt{intrView}\ \mathtt{outCVU}\ \text{-}(\mathtt{outCV}\ \text{-}m)\ \text{-}\beta) \\ \mathtt{to/}D \mapsto \Lambda\, R.\, \Lambda\, x.\, \mathtt{elimCast}\ \text{-}(\pi_1\ x) \end{array} \right\}\end{array}$$

Figure 20: Elaboration of datatype declarations

$$\frac{\Gamma, R_1 : \star, R_2 : \star, z : \texttt{Cast} \cdot R_1 \cdot R_2 \ ; \ \texttt{elimCast -}z \vdash T[R_1/R] \leqslant T[R_2/R] \hookrightarrow s}{\Gamma \vdash \lambda\, R : \star.\, T \overset{+}{\hookrightarrow} \texttt{intrMono -}(\Lambda\, R_1.\, \Lambda\, R_2.\, \lambda\, z.\, \texttt{intrCast -}s\ \texttt{-}(\lambda\, y.\, \beta\{\lambda\, x.\, x\}))}$$

Figure 21: Positivity checking

$$\frac{\Gamma \vdash T_1 \cong T_2}{\Gamma; s \vdash T_1 \leqslant T_2 \hookrightarrow \ \lambda\, x.\, x} \qquad \frac{\Gamma \vdash s : S \to T \hookrightarrow \_}{\Gamma; s \vdash S \leqslant T \hookrightarrow s}$$

$$\frac{\Gamma; s' \vdash S_2 \leqslant S_1 \hookrightarrow s \quad \Gamma, y : S_2; s' \vdash T_1[(s\ y)/x] \leqslant T_2[y/x] \hookrightarrow t}{\Gamma; s' \vdash \Pi\, x : S_1.\, T_1 \leqslant \Pi\, x : S_2.\, T_2 \hookrightarrow \lambda\, f.\, \lambda\, x.\, t[x/y]\ (f\ (s\ x))}$$

$$\frac{\Gamma; s' \vdash S_2 \leqslant S_1 \hookrightarrow s \quad \Gamma, y : S_2; s' \vdash T_1[(s\ y)/x] \leqslant T_2[y/x] \hookrightarrow t}{\Gamma; s' \vdash \forall\, x : S_1.\, T_1 \leqslant \forall\, x : S_2.\, T_2 \hookrightarrow \lambda\, f.\, \Lambda\, x.\, t[x/y]\ (f\ \texttt{-}(s\ x))}$$

$$\frac{\Gamma; s' \vdash S_1 \leqslant S_2 \hookrightarrow s \quad \Gamma, y : S_1; s' \vdash T_1[y/x] \leqslant T_2[(s\ y)/x] \hookrightarrow t}{\Gamma; s' \vdash \iota\, x : S_1.\, T_1 \leqslant \iota\, x : S_2.\, T_2 \hookrightarrow \lambda\, u.\, [s\ u.1, t[u.1/y]\ u.2]}$$

$$\frac{\Gamma; s' \vdash K_2 \leqslant K_1 \hookrightarrow S \quad \Gamma, Y : K_2; s' \vdash T_1[(S \cdot Y)/X] \leqslant T_2[Y/X] \hookrightarrow s}{\Gamma; s' \vdash \forall\, X : K_1.\, T_1 \leqslant \forall\, X : K_2.\, T_2 \hookrightarrow \lambda\, f.\, \Lambda\, X.\, s[X/Y]\ (f \cdot (S \cdot X))}$$

Figure 22: Subtyping with type inclusion

$$\frac{}{\Gamma; s \vdash \star \ \leqslant \ \star \hookrightarrow \lambda\, X.\, X} \qquad \frac{\Gamma; s' \vdash S_2 \leqslant S_1 \hookrightarrow s \quad \Gamma, y : S_2; s' \vdash K_1[(s\ y)/x] \leqslant K_2[y/x] \hookrightarrow S}{\Gamma; s' \vdash \Pi\, x : S_1.\, K_1 \leqslant \Pi\, x : S_2.\, K_2 \hookrightarrow \lambda\, P.\, \lambda\, x.\, S[x/y] \cdot (P\ (s\ x))}$$

$$\frac{\Gamma; s \vdash K_1' \leqslant \ K_1 \hookrightarrow S_1 \quad \Gamma, Y : K_1'; s \vdash K_2[(S_1 \cdot Y)/X] \ \leqslant \ K_2'[Y/X] \hookrightarrow S_2}{\Gamma; s \vdash \Pi\, X : K_1.\, K_2 \leqslant \ \Pi\, X : K_1'.\, K_2' \hookrightarrow \lambda\, P.\, \lambda\, X.\, S_2[X/Y] \cdot (P \cdot (S_1 \cdot X))}$$

Figure 23: Subkinding with type inclusion

$$\frac{}{\Gamma; s' \vdash \emptyset \hookrightarrow \emptyset} \qquad \frac{\Gamma; s' \vdash A \leqslant A' \hookrightarrow s \quad \Gamma, y : A; s' \vdash \overline{a : [y/x]A_1} \leqslant \overline{a : [(s\ y)/x]A_2} \hookrightarrow \overline{s}}{\Gamma; s' \vdash (x : A, \overline{a : A_1}) \leqslant (x : A', \overline{a : A_2}) \hookrightarrow (s\ x), \overline{[x/y]s}}$$

$$\frac{\Gamma; s' \vdash K \leqslant K' \hookrightarrow S \quad \Gamma, Y : K; s \vdash \overline{a : [Y/X]A_1} \leqslant \overline{a : [(S \cdot Y)/X]A_2} \hookrightarrow \overline{s}}{\Gamma; s' \vdash (X : K, \overline{a : A_1}) \leqslant (X : K', \overline{x : A_2}) \hookrightarrow (S \cdot X), \overline{[X/Y]s}}$$

Figure 24: Telescope coercion

$$\frac{\begin{array}{c}\mathrm{IndEl}[D,R,\Delta,m,L,\Theta,\mathcal{E}]\in\Gamma\quad \Gamma\vdash is:\texttt{Is/}D\cdot S\hookrightarrow is'\\ (c_i:\overset{\Pi}{\forall}\overline{a_i:A_i}.\,D\in\Delta)_{i=1\ldots\#\Delta}\\ (\Gamma\,;\,\texttt{to/}D\ \texttt{-}is\vdash([S/R]\overline{a_i:A_i})\leqslant([D/R]\overline{a_i:A_i})\hookrightarrow\overline{s_i})_{i=1\ldots\#\Delta}\\ (\Gamma\vdash\overset{\lambda}{\Lambda}\,\overline{a_i}.\,t_i:\overset{\Pi}{\forall}\,[S/R]\overline{a_i:A_i}.\,P\ (c_i\ \overline{s_i})\hookrightarrow\overset{\lambda}{\Lambda}\,\overline{a_i}.\,t_i')_{i=1\ldots\#\Delta}\end{array}}{\Gamma\vdash\{c_i\ \overline{a_i}\to t_i\}_{i=1\ldots\#\Delta}:(P,is)\hookrightarrow(\overset{\lambda}{\Lambda}\,\overline{a_i}.\,t_i')_{i=1\ldots\#\Delta}}\ \textsc{Case}$$

$$\frac{\begin{array}{c}\mathrm{IndEl}[D,R,\Delta,m,L,\Theta,\mathcal{E}]\in\Gamma\quad \Gamma\vdash P:D\to\star\hookrightarrow P'\quad \Gamma\vdash t:D\hookrightarrow t'\\ \Gamma'=_{\mathbf{df}}\Gamma,\texttt{Type/}ih:\star,\texttt{isType/}ih:\texttt{Is/}D\cdot\texttt{Type/}ih,ih:\mathit{\Pi}\ y{:}\texttt{Type/}ih.\,P\ (\texttt{to/}D\ \texttt{-isType/}ih\ y)\\ \Gamma'\vdash\{c_i\ \overline{a_i}\to t_i\}_{i=1\ldots\#\Delta}:(P,\texttt{isType/}ih)\hookrightarrow(\overset{\lambda}{\Lambda}\,\overline{a_i}.\,t_i)_{i=1\ldots\#\Delta}\quad (z\notin FV(t_i))_{i=1\ldots\#\Delta}\end{array}}{\begin{array}{c}\Gamma\vdash\mu\ ih.\ t\ @P\ \{\ c_i\ \overline{a_i}\to t_i\}_{i=1..\#\Delta}:P\ t\hookrightarrow\\ \texttt{mu}\ \texttt{-}m\ t'\cdot P'\ \mathit{\Lambda}\,\texttt{Type/}ih.\,\mathit{\Lambda}\,\texttt{isType/}ih.\,\lambda\,ih.\,\lambda\,x.\\ x.2\cdot(L\cdot P'\cdot\texttt{Type/}ih\ \texttt{isType/}ih)\ (\overset{\lambda}{\Lambda}\,\overline{a_i}.\,\mathit{\Lambda}\,z.\,t_i')_{i=1..\#\Delta}\ \texttt{-(intrView}\ \beta\{x.1\}\ \texttt{-}x\ \texttt{-}\beta)\end{array}}$$

$$\frac{\begin{array}{c}\mathrm{IndEl}[D,R,\Delta,m,L,\Theta,\mathcal{E}]\in\Gamma\quad \Gamma\vdash is:\texttt{Is/}D\cdot T\hookrightarrow is'\\ \Gamma\vdash t:T\hookrightarrow t'\quad \Gamma\vdash T:\star\hookrightarrow T'\quad \Gamma\vdash P:D\to\star\hookrightarrow P'\\ \Gamma\vdash\{c_i\ \overline{a_i}\to t_i\}_{i=1\ldots\#\Delta}:(P,is)\hookrightarrow(\overset{\lambda}{\Lambda}\,\overline{a_i}.\,t_i')_{i=1\ldots\#\Delta}\quad (z\notin FV(t_i'))_{i=1\ldots\#\Delta}\end{array}}{\begin{array}{c}\sigma\texttt{<}is\texttt{>}\ t\ @P\ \{\ c_i\ \overline{a_i}\to t_i\}_{i=1..\#\Delta}:P\ (\texttt{to/}D\ \texttt{-}is\ t)\hookrightarrow\texttt{sigma}\ \texttt{-}m\ \texttt{-}is'\ t'\cdot P'\\ \lambda\,x.\ \ x.2\cdot(L\cdot P'\cdot T'\ is')\ (\overset{\lambda}{\Lambda}\,\overline{a_i}.\,\mathit{\Lambda}\,z.\,t_i')_{i=1..\#\Delta}\ \texttt{-(intrView}\ \beta\{x.1\}\ \texttt{-}x\ \texttt{-}\beta)\end{array}}$$

$$\frac{\mathrm{IndEl}[D,R,\Delta,m,L,\Theta,\mathcal{E}]\in\Gamma}{\Gamma\vdash D:\star\hookrightarrow\mathcal{E}(D)}\ \textsc{D}\qquad \frac{\mathrm{IndEl}[D,R,\Delta,m,L,\Theta,\mathcal{E}]\in\Gamma\quad c_j:\overset{\Pi}{\forall}\overline{a_j:A_j}.\,D\in\Delta}{\Gamma\vdash c_j:\overset{\Pi}{\forall}\,[D/R]\overline{a_j:A_j}.\,D\hookrightarrow\mathcal{E}(c_j)}\ \textsc{C}$$

$$\frac{\mathrm{IndEl}[D,R,\Delta,m,L,\Theta,\mathcal{E}]\in\Gamma}{\Gamma\vdash\texttt{Is/}D:D\to\star\hookrightarrow\mathcal{E}(\texttt{Is/}D)}\ \textsc{Is}\qquad \frac{\mathrm{IndEl}[D,R,\Delta,m,L,\Theta,\mathcal{E}]\in\Gamma}{\Gamma\vdash\texttt{is/}D:\texttt{Is/}D\cdot D\hookrightarrow\mathcal{E}(\texttt{is/}D)}\ \textsc{is}$$

$$\frac{\mathrm{IndEl}[D,R,\Delta,m,L,\Theta,\mathcal{E}]\in\Gamma}{\Gamma\vdash\texttt{to/}D:\forall\,R{:}\star.\,\texttt{Is/}D\cdot R\Rightarrow R\to D}\ \textsc{to}$$

Figure 25: Elaboration of terms

$$\frac{\mathrm{IndEl}[D, R, \Delta, m, L, \Theta, \mathcal{E}] \in \Gamma \quad DV(\Delta) = \{c_i \mid i = 1 \ldots n\} \quad \Gamma \vdash p \hookrightarrow p' \quad (\Gamma \vdash q_i \hookrightarrow q_i')_{i=1..n}}{\Gamma \vdash \mu\ ih.\ p\ \{\ c_i\ \overline{a_i} \to q_i\}_{i=1..n} \hookrightarrow |\texttt{mu}|\ p'\ (\lambda\, ih.\, \lambda\, x.\, x\ (\lambda\, \overline{a_i}.\, q_i')_{i=1..n})}$$

$$\frac{\mathrm{IndEl}[D, R, \Delta, m, L, \Theta, \mathcal{E}] \in \Gamma \quad DV(\Delta) = \{c_i \mid i = 1 \ldots n\} \quad \Gamma \vdash p \hookrightarrow p' \quad (\Gamma \vdash q_i \hookrightarrow q_i')_{i=1..n}}{\Gamma \vdash \sigma\ p\ \{c_i\ \overline{a_i} \to q_i\}_{i=1..n} \hookrightarrow |\texttt{sigma}|\ p'\ (\lambda\, x.\, x\ (\lambda\, \overline{a_i}.\, q_i')_{i=1..n})}$$

$$\frac{\mathrm{IndEl}[D, R, \Delta, m, L, \Theta, \mathcal{E}] \in \Gamma}{\Gamma \vdash \texttt{is/}D \hookrightarrow |\mathcal{E}(\texttt{is/}D)|} \qquad \frac{\mathrm{IndEl}[D, R, \Delta, m, L, \Theta, \mathcal{E}] \in \Gamma \quad c \in DV(\Delta)}{\Gamma \vdash c \hookrightarrow |\mathcal{E}(c)|}$$

$$\frac{\mathrm{IndEl}[D, R, \Delta, m, L, \Theta, \mathcal{E}] \in \Gamma}{\Gamma \vdash \texttt{to/}D \hookrightarrow |\mathcal{E}(\texttt{to/}D)|}$$

Figure 26: $\boxed{\Gamma \vdash p \hookrightarrow p'}$ Elaboration of untyped (post-erasure) expressions

Lemma 2. *If* $\Gamma \vdash t \hookrightarrow t'$ *then* $\Gamma \vdash |t| \hookrightarrow |t'|$

Proof. By a straightforward induction on the assumed derivation. □

Lemma 3.

- *If* $\Gamma \vdash t : T \hookrightarrow t_1'$ *and* $\Gamma \vdash t : T \hookrightarrow t_2'$ *then* $t_1 = t_2$ *(denoting syntactic equality)*
- *If* $\Gamma \vdash T : K \hookrightarrow T_1'$ *and* $\Gamma \vdash T : K \hookrightarrow T_2'$ *then* $T_1' = T_2'$
- *If* $\Gamma \vdash K \hookrightarrow K_1'$ *and* $\Gamma \vdash K \hookrightarrow K_2'$ *then* $K_1' = K_2'$

Proof. By a straightforward mutual induction on the assumed typing derivation. There is no overlap between inference rules. □

1 Datatype Declarations

1.1 Positivity Checker

Theorem 3 (Soundness of positivity checker)**.**

1. *If* $\Gamma; s \vdash S \leqslant T \hookrightarrow s'$ *then* $\Gamma \vdash s' : S \to T \hookrightarrow \_$ *and* $|s'| =_{\beta\eta} \lambda\, x.\, x$
2. *If* $\Gamma; s \vdash K_1 \leqslant K_2 \hookrightarrow S$ *then* $\Gamma \vdash S : K_1 \to K_2 \hookrightarrow \_$
3. *If* $\Gamma \vdash F \overset{+}{\hookrightarrow} m$ *then* $\Gamma \vdash m : \mathit{Mono} \cdot F \hookrightarrow \_$
4. [2] *If* $\Gamma; s' \vdash \overline{a : A} \leqslant \overline{a : A'} \hookrightarrow \overline{s}$ *then* $\Gamma; (\overline{a : A}) \vdash \overline{s} : (\overline{a : A'}) \hookrightarrow \_$ *and* $|\overline{s}| \cong |\overline{a}|$

Proof. By a straightforward mutual induction on the assumed derivation. A few example cases are given

Part 1. Case

$$\frac{\Gamma \vdash T_1 \cong T_2 \quad \Gamma \vdash T_1 : \star \hookrightarrow \_ \quad \Gamma \vdash T_2 : \star \hookrightarrow \_}{\Gamma; s \vdash T_1 \leqslant T_2 \hookrightarrow \ \lambda\, x.\, x}$$

Appeal to convertibility and function rules of Figure 18

Part 1. Case

$$\frac{\Gamma \vdash s : S \to T \hookrightarrow \_ \quad |s| \cong \lambda\, x.\, x}{\Gamma; s \vdash S \leqslant T \hookrightarrow s}$$

From the premises

Part 1. Case

$$\frac{\Gamma; s' \vdash S_2 \leqslant S_1 \hookrightarrow s \quad \Gamma, y : S_2; s' \vdash [(s\ y)/x]T_1 \leqslant [y/x]T_2 \hookrightarrow t}{\Gamma; s' \vdash \Pi\, x : S_1.\, T_1 \leqslant \Pi\, x : S_2.\, T_2 \hookrightarrow \lambda f.\, \lambda\, x.\, [x/y]t\ (f\ (s\ x))}$$

- By the IH, s has type $S_2 \to S_1$ and reducible with $\lambda\, x.\, x$
- By the IH, (under context extended by $y : S_2$), t has type $[(s\ y)/x]T_1 \to [y/x]T_2$ and reducible with $\lambda\, x.\, x$
- Assume f has type $\Pi\, x : S_1.\, T_1$ and x type S_2
- $f\ (s\ x)$ has type $[(s\ x)/x]T_1$ and reducible with $f\ x$
- $[x/y]t$ of this has type T_2, reducible to the same
- η-contract and conclude

Part 1. Case

$$\frac{\Gamma; s' \vdash S_2 \leqslant S_1 \hookrightarrow s \quad \Gamma, y : S_2; s' \vdash [(s\ y)/x]T_1 \leqslant [y/x]T_2 \hookrightarrow t}{\Gamma; s' \vdash \forall\, x : S_1.\, T_1 \leqslant \forall\, x : S_2.\, T_2 \hookrightarrow \lambda f.\, \Lambda\, x.\, [x/y]t\ (f\ \text{-}(s\ x))}$$

Similar to above

[2] c.f. Figure 17b

Part 1. Case

$$\frac{\Gamma; s' \vdash K_2 \leqslant K_1 \hookrightarrow S \quad \Gamma, Y : K_2; s' \vdash [(S \cdot Y)/X]T_1 \leqslant [Y/X]T_2 \hookrightarrow s}{\Gamma; s' \vdash \forall\, X : K_1.\, T_1 \leqslant \forall\, X : K_2.\, T_2 \hookrightarrow \lambda f.\, \Lambda X.\, [X/Y]s\ (f \cdot (S \cdot X))}$$

- By mutual induction on 2., S has kind $K_2 \to K_1$
- By IH, s (under context extended by Y of kind K_2) has type $[S \cdot Y/X]T_1 \to [Y/X]T_2$, and convertible with $\lambda\, x.\, x$
- Assume $f : \forall\, X : K_1.\, T_1$, $X : K_2$
- $f \cdot (S \cdot X)$ has kind $[S \cdot X/X]T_1$, convertible (after erasure!) with f
- $[X/Y]s$ of this has type T_2, convertible with the same
- Erase to get conversion with $\lambda f.\, f$

Part 1. Case

$$\frac{\Gamma; s' \vdash S_1 \leqslant S_2 \hookrightarrow s \quad \Gamma, y : S_1; s' \vdash [y/x]T_1 \leqslant [(s\ y)/x]T_2 \hookrightarrow t}{\Gamma; s' \vdash \iota\, x : S_1.\, T_1 \leqslant \iota\, x : S_2.\, T_2 \hookrightarrow \lambda\, u.\, [s\ u.1, [u.1/y]t\ u.2]}$$

By assumption $\Gamma \vdash s : S_1 \to S_2$ and $s =_{\beta\eta} \lambda\, x.\, x$, and $\Gamma, y : S_1 \vdash t : [y/x]T_1 \to [(s\ y)/x]T_2$ and $t =_{\beta\eta} \lambda\, x.\, x$.

Thus, the two components of the intersection in the body of the elaborated term have type S_2 and $[(s\ u.1)/x]T_2$ (and this type is convertible with $[u.1/x]T_2$), and are convertible with $u.1$ and $u.2$. Thus the entire expression is convertible with $\lambda\, u.\, u$ and can be assigned type $\iota\, x : S_1.\, T_1 \to \iota\, x : S_2.\, T_2$

Part 2. Case

$$\frac{}{\Gamma; s \vdash \star\ \leqslant\ \star \hookrightarrow \lambda\, X.\, X}$$

Immediate

Part 2. Case

$$\frac{\Gamma; s' \vdash S_2 \leqslant S_1 \hookrightarrow s \quad \Gamma, y : S_2; s' \vdash [(s\ y)/x]K_1 \leqslant [y/x]K_2 \hookrightarrow S}{\Gamma; s' \vdash \Pi\, x : S_1.\, K_1 \leqslant \Pi\, x : S_2.\, K_2 \hookrightarrow \lambda\, P.\, \lambda\, x.\, [x/y]S \cdot (P\ (s\ x))}$$

- By mutual induction on 1., $s : S_2 \to S_1$ and convertible with $\lambda\, x.\, x$
- By IH, (under context extended by $y : S_2$), S has kind $[(s\ y)/x]K_1 \to [y/x]K_2$
- Assume $P : \Pi\, x : S_1.\, K_1$, $x : S_2$
- $P\ (s\ x)$ has kind $[s\ x/x]K_1$
- $[x/y]S$ of this has kind K_2

Part 2. Case

$$\frac{\Gamma; s \vdash K_1' \leqslant\ K_1 \hookrightarrow S_1 \quad \Gamma, Y : K_1'; s \vdash [(S_1 \cdot Y)/X]K_2\ \leqslant\ [Y/X]K_2' \hookrightarrow S_2}{\Gamma; s \vdash \Pi\, X : K_1.\, K_2 \leqslant\ \Pi\, X : K_1'.\, K_2' \hookrightarrow \lambda\, P.\, \lambda\, X : [X/Y]S_2 \cdot (P \cdot (S_1 \cdot X)).}$$

By assumption $\Gamma \vdash S_1 : K_1' \to K_1$ and $S_1 =_{\beta\eta} \lambda\, X.\, X$,
and $\Gamma, Y : K_1' \vdash S_2 : [(S_1 \cdot Y)/X]K_2\ \to [Y/X]K_2'$ and $S_2 =_{\beta\eta} \lambda\, X.\, X$. It is clear then that the elaborated type in the conclusion has kind $\Pi\, X : K_1.\, K_2 \to \Pi\, X : K_1'.\, K_2'$, and reduces and η-contracts to $\lambda\, P.\, P$

Part 3. Case

$$\frac{\Gamma, R_1 : \star, R_2 : \star, z : \texttt{Cast} \cdot R_1 \cdot R_2\ ;\ \texttt{elimCast}\ \text{-}z \vdash T[R_1/R] \leqslant T[R_2/R] \hookrightarrow s}{\Gamma \vdash \lambda\, R : \star.\, T \overset{+}{\hookrightarrow} \texttt{intrMono}\ \text{-}(\Lambda\, R_1.\, \Lambda\, R_2.\, \lambda\, z.\, \texttt{intrCast}\ \text{-}s\ \text{-}(\lambda\, y.\, \beta\{\lambda\, x.\, x\}))}$$

We appeal to the proof of 1. (noting that the premises of the "base" rule will be satisfid as $\texttt{elimCast}\ \text{-}z$ has the required type and erasure). Then, t has type $F \cdot R_1 \to F \cdot R_2$, $\lambda\_.\, \beta$ is a proof that that $\Pi\, R_1 : \star.\, \{t\ r \simeq r\}$, and thus the whole elaborated expression has type $\texttt{Mono} \cdot F$.

Part 4. Case

$$\overline{\Gamma; s' \vdash \emptyset \hookrightarrow \emptyset}$$

Trivial

Part 4. Case

$$\frac{\Gamma; s' \vdash A \leqslant A' \hookrightarrow s \quad \Gamma, y : A; s' \vdash \overline{a : [y/x]A_1} \leqslant \overline{a : [(s\ y)/x]A_2} \hookrightarrow \overline{s}}{\Gamma; s' \vdash x : A, \overline{a : A_1} \leqslant x : A', \overline{a : A_2} \hookrightarrow (s\ x), \overline{[x/y]s}}$$

So, these are pretty nasty. The difficulty is mostly notational, not theoretical – that is, describing the coercion, and typing, of telescope of constructor arguments.

- Appealing to 1., we have s has type $A \to A'$ and erases to $\lambda\, x.\, x$

  This means $(s\ x)$ has type A' and erases to x

- Appealing to the inductive hypothesis, we have $\Gamma, y : A\,;\, (\overline{a : A_1}) \vdash \overline{s} : (\overline{a : [s\ y/x]A_2}) \hookrightarrow \_$ (Figure 17b) and $\overline{s}$ converts with $\overline{a}$

- Conclude $\Gamma\,;\, (x : A, \overline{a : A_1}) \vdash (s\ x), \overline{s} : (x : A', \overline{a : A_2}) \hookrightarrow \_$

Part 4. Case

$$\frac{\Gamma, X : K_1 \vdash S : K_2 \hookrightarrow S' \quad \Gamma, X : K_1\,;\, (\overline{a : A}) \vdash \overline{s} : ([S/X]\overline{a : B}) \hookrightarrow \overline{s'}}{\Gamma\,;\, (X : K_1, \overline{a : A}) \vdash S, \overline{s} : (X : K_2, \overline{a : B}) \hookrightarrow S', \overline{s'}}$$

Similar to above □

1.1.1 Additional Proofs

Lemma 4. *(Reconstruction of coercions from datatype elaboration) Assuming some* Γ, Γ', *IndEl*$[D, R, \Delta, m, L, \Theta, \mathcal{E}]$, U, U' *and term is such that*

- $\vdash \Gamma \hookrightarrow \Gamma'$
- *IndEl*$[D, R, \Delta, m, L, \Theta, \mathcal{E}] \in \Gamma, \mathcal{E}(D) = \mu(\texttt{CV} \cdot D^{FI})$
- $\Gamma \vdash U : \star \hookrightarrow U'$ *and* $\Gamma \vdash is : \texttt{Is/}D \cdot U \hookrightarrow is'$, *and* $\Gamma' \vdash U' : \star$

we have the following:

- *If* $\Gamma; \texttt{to/}D$ *-is* $\vdash S \leqslant T \hookrightarrow s$, *where* $\Gamma \vdash S : \star \hookrightarrow S'$, $\Gamma \vdash T : \star \hookrightarrow T'$, $\Gamma' \vdash S' : \star$, *and* $\Gamma' \vdash T' : \star$
  then there is some s' *such that* $\Gamma \vdash s : S \to T \hookrightarrow s'$ *and* $\Gamma'; \mathcal{E}(\texttt{to/}D)$ *-is'* $\vdash S' \leqslant T' \hookrightarrow s'$
- *If* $\Gamma; \texttt{to/}D$ *-is* $\vdash K_1 \leqslant K_2 \hookrightarrow S$, *where* $\Gamma \vdash K_1 \hookrightarrow K_1'$, $\Gamma \vdash K_2 \hookrightarrow K_2'$, $\Gamma' \vdash K_1'$, *and* $\Gamma' \vdash K_2'$
  then there is some S' *such that* $\Gamma \vdash S : K_1 \to K_2 \hookrightarrow S'$
  and $\Gamma'; \mathcal{E}(\texttt{to/}D)$ *-m -is'* $\vdash K_1' \leqslant K_2' \hookrightarrow S'$,

That is, that constructor-argument coercions produced from base assumption `to/`D *for types and kinds in the surface language elaborate to coercions produced by base assumption* $\mathcal{E}(\texttt{to/}D)$ *for their elaborations.*

Proof. By induction on the assumed derivation of the coercion. Corollary 3.1 of this lemma is used by Theorem 6. Some interesting cases are given below

Case
$$\frac{\Gamma \vdash \texttt{to/}D\ \textit{-is} : U \to D \quad |\texttt{to/}D\ \textit{-is}| = \lambda\, x.\, x}{\Gamma; \texttt{to/}D\ \textit{-is} \vdash U \leqslant D \hookrightarrow \texttt{to/}D\ \textit{-is}}$$

The elaboration of `to/`D *-is* is $\mathcal{E}(\texttt{to/}D) = \Lambda\, R.\, \Lambda\, x.\, \texttt{elimCast}\ \text{-}(\pi_1\ x)$ and it has the desired type and erasure under the elaborated context Γ' by assumption.

$$\frac{\Gamma' \vdash \mathcal{E}(\texttt{to/}D)\ \textit{-is'} : S' \to T' \quad |\mathcal{E}(\texttt{to/}D)\ \textit{-is'}| = \lambda\, x.\, x}{\Gamma'; \mathcal{E}(\texttt{to/}D)\ \textit{-is'} \vdash S' \leqslant T' \hookrightarrow \mathcal{E}(\texttt{to/}D)\ \textit{-is'}}$$

Case
$$\frac{\Gamma; \texttt{to/}D\ \textit{-is} \vdash S_2 \leqslant S_1 \hookrightarrow s \quad \Gamma, y : S_2; \texttt{to/}D\ \textit{-is} \vdash [(s\ y)/x]T_1 \leqslant [y/x]T_2 \hookrightarrow t}{\Gamma; \texttt{to/}D\ \textit{-is} \vdash \Pi\, x : S_1.\, T_1 \leqslant \Pi\, x : S_2.\, T_2 \hookrightarrow \lambda\, f.\, \lambda\, y.\, t\ (f\ (s\ y))}$$

Our assumptions are:

1. $\vdash \Gamma \hookrightarrow \Gamma'$
2. $\text{IndEl}[D, R, \Delta, m, L, \Theta, \mathcal{E}] \in \Gamma, \mathcal{E}(D) = \mu(\texttt{CV} \cdot D^{\text{FI}})$
3. $\Gamma \vdash U : \star \hookrightarrow U'$ and $\Gamma \vdash is : \texttt{Is/}D \cdot U \hookrightarrow is'$, and $\Gamma' \vdash U' : \star$
4. $\Gamma \vdash \Pi\, x : S_1.\, T_1 : \star \hookrightarrow \Pi\, x : S_1'.\, T_1'$ (inversion of elaboration for types)
5. $\Gamma \vdash \Pi\, x : S_2.\, T_2 : \star \hookrightarrow \Pi\, x : S_2'.\, T_2'$ (inversion of elaboration for types)
6. $\Gamma' \vdash \Pi\, x : S_2'.\, T_2' : \star$
   With this and assumptions 1. and 4., we know $\vdash \Gamma, y : S_2 \hookrightarrow \Gamma', y : S_2'$ and $\Gamma', y : S_2' \vdash [y/x]T_2' \vdash \star$
7. $\Gamma' \vdash \Pi\, x : S_1'.\, T_1' : \star$
   With this and assumptions 1., 5., and 6., we know that $\Gamma', y : S_2' \vdash [s'\ y/x]T_2' \vdash \star$ for any $s : S_2' \to S_1'$

Invoke the induction hypothesis on s and t to get their elaborations s' and t', noting that for the latter the elaborated context is $\vdash \Gamma, y : S_2 \hookrightarrow \Gamma', y : S_2'$ and that (from the IH) $s' : S_2' \to S_1'$ under the elaborated context. Then, the coercion we want is $\lambda\, f.\, \lambda\, y.\, t'\ (f\ (s'\ y))$, and this is obviously the elaboration of $\lambda\, f.\, \lambda\, y.\, t\ (f\ (s\ y))$

Case $$\frac{\Gamma; \texttt{to/}D\ \textit{-is} \vdash K_3 \leqslant\ K_1 \hookrightarrow S_1 \quad \Gamma, Y : K_3; \texttt{to/}D\ \textit{-is} \vdash [(S_1 \cdot Y)/X]K_2\ \leqslant\ [Y/X]K_4 \hookrightarrow S_2}{\Gamma; \texttt{to/}D\ \textit{-is} \vdash \Pi\, X : K_1.\, K_2 \leqslant\ \Pi\, X : K_3.\, K_4 \hookrightarrow \lambda\, P.\, \lambda\, X : [X/Y]S_2 \cdot (P \cdot (S_1 \cdot X)).}$$

This follows by a similar argument to the case above. Invoke the induction hypothesis on S_1 and S_2 to get their elaborations S_1' and T_2', noting that for the latter the elaborated context is $\Gamma', y : K_3' \vdash \Gamma, Y : K_3 \hookrightarrow \Gamma', y : K_3'$ (where we know K_3 is the elaboration of K_3 by assumption and by inversion of the elaboration rules). Then, the coercion we want is $\lambda\, P.\, \lambda\, X.\, [X/Y]S_2' \cdot (P \cdot (S_1' \cdot X))$, which is obviously the elaboration of $\lambda\, P.\, \lambda\, X : [X/Y]S_2 \cdot (P \cdot (S_1 \cdot X))$.

□

Corollary 3.1. *Lemma 4 also holds when extended to a sequence of coercions $\overline{s}$ for telescope $\overset{\Pi}{\forall}\ \overline{a : A_1}.\ \leqslant \overset{\Pi}{\forall}\ \overline{a : A_2}$.*

Proof. By a straighforward induction over the rules of Figure 17b, appealing to Lemma 4 for each non-empty telescope. □

1.2 Datatype and Constructor Elaboration

Theorem 4 (Soundness of elaboration of declarations)**.** *For all well-formed contexts* Γ*, and well-formed datatype declarations* $\Gamma \vdash \mathit{Ind}[D, R, \Delta]$ *wf, if*

- $\Gamma \vdash \mathit{Ind}[D, R, \Delta] \dashv \Gamma, \mathit{IndEl}[D, R, \Delta, m, L, \Theta, \mathcal{E}]$ *and* $(c_i : \overset{\Pi}{\forall}\, \overline{a_i : A_i}.\, D \in \Delta)_{i=1..\#\Delta}$
- *and* $\vdash \Gamma \hookrightarrow \Gamma'$ *for some* Γ' *implies* Γ' *is well formed*
- *and* $(\Gamma, X{:}\star, R{:}\star \vdash \overset{\Pi}{\forall}\, \overline{a_i : A_i}.\, X : \star \hookrightarrow \overset{\Pi}{\forall}\, \overline{a_i : A_i'}.\, X$ *for some* $(\overline{a_i : A_i})_{i=1...\#\Delta}$
  implies $(\Gamma', R : \star, X : \star \vdash \overset{\Pi}{\forall}\, \overline{a_i : A_i'}.\, X : \star)_{i=1..\#\Delta}$

we have that:

- $\Gamma' \vdash \mathcal{E}(D) : \star$ *and* $(\Gamma' \vdash \mathcal{E}(c_i) : \overset{\Pi}{\forall}\, \overline{a_i : [\mathcal{E}(D)/R]A_i'}.\, \mathcal{E}(D))_{i=1..\#\Delta}$
- $\Gamma' \vdash \mathcal{E}(\mathtt{Is/}D) : \star \to \star$ *and* $\Gamma' \vdash \mathcal{E}(\mathtt{is/}D) : \mathcal{E}(\mathtt{Is/}D) \cdot \mathcal{E}(D)$
- $\Gamma' \vdash \mathcal{E}(\mathtt{to/}D) : \forall\, R{:}\star.\, \forall\, \mathit{is} : \mathcal{E}(\mathtt{Is/}D) \cdot R.\, R \to \mathcal{E}(D)$ *and* $|\mathcal{E}(\mathtt{to/}D)| =_{\beta\eta} \lambda\, x.\, x$
- $\Gamma \vdash m : \mathtt{Mono} \cdot D^{FI}$*, where* $\Gamma \vdash \mathit{Ind}[D, R, \Delta] \overset{FI}{\hookrightarrow} D^{FI}$
- $\Gamma \vdash L : (\mathcal{E}(D) \to \star) \to \mathit{\Pi}\, R{:}\star.\, \mathcal{E}(\mathtt{Is/}D) \cdot R \to D^F \cdot R \to \star$*,*
  where D^{FI} *is as above and* $\Gamma \vdash \mathit{Ind}[D, R, \Delta] \overset{F}{\hookrightarrow} D^F$

Proof. The proof proceeds by traversing the derivation of the premises of rule [Data] and by mutual induction on the soundness of elaboration of terms and types (Theorem 6). We will write [**Rule**]**Premise : Classifier** to help guide the reader, and author(s). Throughout the proof $\overline{a_i : A_i'}$ will refer to the elaboration of the construct c_i's argument telescope $\overline{a_i : A_i}$. We will also abbreviate $\mu(\mathtt{CV} \cdot D^{\mathrm{FI}})$ to D^{FIX}.

We start at the root: the unlabeled judgment for elaborating datatype declarations.

[] $\Gamma' \vdash D^{\mathbf{FIX}} : \star$
The rule for deriving this premise is FIX. In that rule, we must show the single premise

- [FIX] $\Gamma' \vdash D^{\mathrm{FI}} : \star \to \star$

[**FIX**] $\Gamma' \vdash D^{\mathbf{FI}} : \star \to \star$
The rule for deriving this premise is FI. There, we must show

- [FI] $\Gamma' \vdash D^{\mathrm{F}} : \star \to \star$
  Shown later in the proof
- [FI] $(\Gamma' \vdash c_i^{\mathrm{F}} : \forall\, R{:}\star.\, \overset{\Pi}{\forall}\, \overline{a_i : A_i'}.\, D^{\mathrm{F}} \cdot R)_{i=1..\#\Delta}$
  Shown later in the proof

For now assuming we have the above, to ensure that $D^{\mathrm{FI}} = \lambda\, R.\, \iota\, x : D^{\mathrm{F}} \cdot R.\, \forall\, X : D^{\mathrm{F}} \cdot R \to \star.\, \mathit{\Pi}\, x_i : (\overset{\Pi}{\forall}\, \overline{a_i : A_i'}.\, X\ (c_i^{\mathrm{F}}\ \overline{a_i}))_{i=1..\#\Delta}.\, X\ x$ has kind $\star \to \star$, we show the two components of the intersection have kind $\star$ under a context extended by $R{:}\star$.

- We see that $D^{\mathrm{F}} \cdot R$ has kind $\star$ from the temporary assumption.
- We show the second component $\forall\, X : D^{\mathrm{F}} \cdot R \to \star.\, \mathit{\Pi}\, x_i : (\overset{\Pi}{\forall}\, \overline{a_i : A_i'}.\, X\ (c_i^{\mathrm{F}}\ \overline{a_i}))_{i=1..\#\Delta}.\, X\ x$ has kind $\star$ under a context extended by $x : D^{\mathrm{F}} \cdot R$ by
  - taking the premise $(\Gamma, R : \star, X : \star \vdash \overset{\Pi}{\forall}\, \overline{a_i : A_i}.\, X : \star \hookrightarrow \overset{\Pi}{\forall}\, \overline{a_i : A_i'}.\, X)_{i=1..\#\Delta}$ and combining it with the assumed implication that this means $(\Gamma', R{:}\star, X{:}\star \vdash \overset{\Pi}{\forall}\, \overline{a_i : A_i'}.\, X : \star)_{i=1..\#\Delta}$
  - from this we have $\Gamma', R : \star, X : D^{\mathrm{FI}} \cdot R \to \star \vdash \overset{\Pi}{\forall}\, \overline{a_i : A_i'}.\, X\ (c_i^{\mathrm{F}}\ \overline{a_i}) : \star)_{i=1..\#\Delta}$ by an easy inductive argument, using our temporary assumption that the c_i^{F} have the desired type
  - and we finally have $(\Gamma', R{:}\star \vdash \forall\, X : D^{\mathrm{FI}} \cdot R \to \star.\, \overset{\Pi}{\forall}\, \overline{a_i : A_i'}.\, X\ (c_i^{\mathrm{F}}\ \overline{a_i}) : \star)_{i=1...\#\Delta}$

Thus the elaborated D^{FI} has kind $\star \to \star$

[FI] $\Gamma' \vdash D^{\mathbf{F}} : \star \to \star$
The rule for deriving this premise is F.

- We have that $D^{\mathrm{F}} = \lambda\, R.\, \forall\, X{:}\star.\, \Pi\, x_i{:}(\overset{\Pi}{\forall}\, \overline{a_i : A_i'}.\, X)_{i=1..\#\Delta}.\, X$ has the desired kind $\star \to \star$
  since we have a premise $(\Gamma, X{:}\star, R{:}\star \vdash \overset{\Pi}{\forall}\, \overline{a_i : A_i}.\, X : \star \hookrightarrow \overset{\Pi}{\forall}\, \overline{a_i : A_i'}.\, X)_{i=1...\#\Delta}$,
  and by assumption this implies $(\Gamma', R{:}\star, X{:}\star \vdash \overset{\Pi}{\forall}\, \overline{a_i : A_i'}.\, X : \star)_{i=1...\#\Delta}$

[FI] $(\Gamma' \vdash c_i^{\mathbf{F}} : \forall\, R{:}\star.\, \overset{\Pi}{\forall}\, \overline{a_i : A_i'}.\, D^{\mathbf{F}} \cdot R)_{i=1..\#\Delta}$
The rule for deriving these premises is cF. We consider each constructor $c_j \in \Delta$.

- We wish to show $\lambda\, X.\, x_{i=1...\#\Delta} x_j\ \overline{a_j}$ has the type $D^{\mathrm{F}} \cdot R$ under a context extended by $R{:}\star$ and $\overline{a_j : A_j'}$.
  From [FI]$\Gamma' \vdash D^{\mathrm{F}} : \star \to \star$ above, we have that $D^{\mathrm{F}} \cdot R = \forall\, X{:}\star.\, \Pi\, x_i{:}(\overset{\Pi}{\forall}\, \overline{a_i : A_i'}.\, X)_{i=1..\#\Delta}.\, X$ has kind $\star$.
  We further extend the context by $X{:}\star$ and $(x_i{:}\overset{\Pi}{\forall}\, \overline{a_i : A_i'}.\, X)_{i=1...\#\Delta}$, and wish to show that $x_j\ \overline{a_j}$ has type X, which it clearly does.

[] $(\Gamma' \vdash c_i^{\mathbf{FIX}} : \overset{\Pi}{\forall}\, \overline{a_i : [D^{\mathbf{FIX}}/R]A_i'}.\, D^{\mathbf{FIX}})_{i=1..\#\Delta}$
We signpost that we are now at the next premise of the root rule for elaborating datatype declarations. The rule for deriving this premise is cFIX. For each $j = 1 \ldots \#\Delta$, it suffices to show the following for the premises:

- [cFIX] $\Gamma' \vdash D^{\mathrm{FIX}} : \star$
  This is satisfied by [] $\Gamma \vdash \mathrm{Ind}[D, R, \Delta] \overset{\mathrm{FIX}}{\hookrightarrow} D^{\mathrm{FIX}} : \star$ above (the provisional assumptions we made there have been discharged above).
- [cFIX] $\Gamma' \vdash c_j^{\mathrm{FI}} : \forall\, R{:}\star.\, \overset{\Pi}{\forall}\, \overline{a_j : A_j'}.\, D^{\mathrm{FI}} \cdot R$
  We show this further below, and for this case temporarily assume this has been shown.
- [cFIX] $\Gamma' \vdash m : \mathtt{Mono} \cdot D^{\mathrm{FI}}$
  Shown later in the proof, temporarily assumed.

We must now show that $\mathtt{inCV}\ \text{-}m\ (c_j^{\mathrm{FI}}\ \overline{a_j})$ has type $D^{\mathrm{FIX}} = \mu(\mathtt{CV} \cdot D^{\mathrm{FI}})$ under a context extended by $\overline{a_j : [D^{\mathrm{FIX}}/R]A_j'}$

- by the typing of `inCV`, we have $\mathtt{inCV}\ \text{-}m$ has type $F \cdot D^{\mathrm{FIX}} \to D^{\mathrm{FIX}}$
  so it suffices to show $c_j^{\mathrm{FI}}\ \overline{a_j}$ has type $F \cdot D^{\mathrm{FIX}}$
- instantiating the type argument R of c_j^{FI} to D^{FIX}, we see that this is indeed the case

[cFIX] $\Gamma' \vdash m : \mathtt{Mono} \cdot D^{\mathbf{FI}}$ This comes from Theorem 3. We satisfy the requirements:

- Γ' is well formed by assumption
- we established above [FIX]$\Gamma' \vdash D^{\mathrm{FI}} : \star \to \star$

[cFIX] $\Gamma \vdash\ c_j^{\mathbf{FI}} : \forall\, R{:}\star.\, \overset{\Pi}{\forall}\, \overline{a_j : A_j'}.\, D^{\mathbf{FI}} \cdot R$
The rule for deriving such premises is cFI. In that rule, we must show, for each instance j, the following for the premises

- [cFI] $\Gamma \vdash c_j^{\mathrm{F}} : \forall\, R{:}\star.\, \overset{\Pi}{\forall}\, \overline{a_j : A_j'}.\, D^{\mathrm{F}} \cdot R$
  Proved above in case [FI] $(\Gamma \vdash c_i^{\mathrm{F}} : \forall\, R{:}\star.\, \overset{\Pi}{\forall}\, \overline{a_i : A_i'}.\, D^{\mathrm{F}} \cdot R)_{i=1..\#\Delta}$

We wish to show that $c_j^{\mathrm{FI}} = \Lambda\, R.\, \overset{\lambda}{\Lambda}\, \overline{a_j}.\, [c_j^{\mathrm{F}}\ \overline{a_j}, \Lambda\, X.\, \lambda\, x_{i=1..\#\Delta}.\, x_j\ \overline{a_j}]$ has the expected type $\forall\, R : \star.\, \overset{\Pi}{\forall}\, \overline{a_j : A_j'}.\, D^{\mathrm{FI}} \cdot R$ where $D^{\mathrm{FI}} = \lambda\, R.\, \iota\, x{:}D^{\mathrm{F}} \cdot R.\, \forall\, X{:}D^{\mathrm{F}} \cdot R \to \star.\, (\Pi\, x_i{:}\overset{\Pi}{\forall}\, \overline{a_i : A_i'}.\, X\ (c_i^{\mathrm{F}}\ \overline{a_i}))_{i=1..\#\Delta}.\, X\ x$

- extend the context by $R{:}\star$ and $\overline{a_j : A_j'}$

- for the first component of the intersection, we know that $|c_j^{\mathrm{F}}\ \overline{a_j}| =_{\beta\eta} \lambda\, x_{i=1\ldots\#\Delta}.\, x_j\ |\overline{a_j}|$, and from above that $c_j^{\mathrm{F}}\ \overline{a_j}$ has type $D^{\mathrm{F}} \cdot R$
- for the second component, we see immediately that it is convertible with the first component, so it remains to show that it has type $\forall\, X : D^{\mathrm{F}} \cdot R \to \star.\, (\Pi\, x : \overset{\Pi}{\forall}\, \overline{a_i : A_i}.\, X\ (c_i^{\mathrm{F}}\ \overline{a_i}))_{i=1\ldots\#\Delta}.\, X\ (c_j^{\mathrm{F}}\ \overline{a_j})$
  - extend the context by $X : D^{\mathrm{F}} \cdot R$ and $(x_i : \Pi\ \overline{a_i : A_i}.\, X\ (c_i^{\mathrm{F}}\ \overline{a_i}))_{i=1\ldots\#\Delta}$
    (we know that the resulting context is well formed)
  - we see then that x_j has type $\Pi\ \overline{a_j : A'_j}.\, X\ (c_j^{\mathrm{F}}\ \overline{a_j})$
    and so $x_j\ \overline{a_j}$ has type $X\ (c_j^{\mathrm{F}}\ \overline{a_j})$ as required

[] $\Gamma' \vdash D^{\mathbf{FI}} : \mathtt{Mono} \cdot D^{\mathbf{FI}}$
This is a repeat of [cFIX] $\Gamma' \vdash m : \mathtt{Mono} \cdot D^{\mathrm{FI}}$

[] $\Gamma' \vdash L : (D^{\mathbf{FIX}} \to \star) \to \Pi\, R : \star.\, \mathtt{RCV} \cdot D^{\mathbf{FI}} \cdot D^{\mathbf{FIX}} \to D^{\mathbf{F}} \cdot R \to \star$ The rule for this premise is Lift sc, which gives us the definition of L.

- We extend the context by $P : D^{\mathrm{FIX}} \to \star$, $R : \star$, $r : \mathtt{RCV} \cdot D^{\mathrm{FI}} \cdot D^{\mathrm{FIX}} \cdot R$, and $x : D^{\mathrm{F}} \cdot R$.
  We have from above that D^{FI} and D^{F} are well kinded and have the approriate kind.
- Further extend the context by $y : \mathtt{View} \cdot (D^{\mathrm{FI}} \cdot R)\ \beta\{x\}$
- It suffices to show that $\mathtt{inCV}\ \text{-}m\ (\mathtt{elimMono}\ \text{-}m\ \text{-}(\pi_1\ r)\ (\mathtt{elimView}\ \beta\{x\}\ \text{-}y))$ has type $D^{\mathrm{FI}} \cdot D^{\mathrm{FIX}}$
  - we established in previous cases that m has type $\mathtt{Mono} \cdot D^{\mathrm{FI}}$
  - since r has type $\mathtt{RCV} \cdot D^{\mathrm{FI}} \cdot D^{\mathrm{FIX}} \cdot R = (\mathtt{Cast} \cdot R \cdot D^{\mathrm{FIX}}) \times (\mathtt{View} \cdot (R \to D^{\mathrm{FI}} \cdot R))\ \mathtt{outCVU}$, we have $\pi_1\ r$ has type $\mathtt{Cast} \cdot R \cdot D^{\mathrm{FIX}}$
  - with $\mathtt{elimMono}\ \text{-}m\ \text{-}(\pi_1\ r)$ we have a function (convertible with $\lambda\, x.\, x$) of type $D^{\mathrm{FI}} \cdot R \to D^{\mathrm{FI}} \cdot D^{\mathrm{FIX}}$
  - from the typing rules for `View` in Figure 13, we have $\mathtt{elimView}\ \beta\{x\}\ \text{-}y$ has type $D^{\mathrm{FI}} \cdot R$.

[] $\Gamma' \vdash \mathcal{E}(\mathtt{Is}/D) : \star \to \star$
We have $\mathcal{E}\mathtt{Is}/D = \mathtt{RCV} \cdot D^{\mathrm{FI}} \cdot D^{\mathrm{FIX}}$. Since we have from earlier in the proof $\Gamma' \vdash D^{\mathrm{FI}} : \star \to \star$ and $\Gamma' \vdash D^{\mathrm{FIX}} : \star$, we have the desired result.

[] $\Gamma' \vdash \mathcal{E}(\mathtt{is}/D) : \mathcal{E}(\mathtt{Is}/D) \cdot D^{\mathbf{FIX}}$
We have $\mathcal{E}(\mathtt{is}/D) = (\mathtt{castRefl} \cdot D^{\mathrm{FIX}}, \mathtt{intrView}\ \mathtt{outCVU}\ \text{-}(\mathtt{outCV}\ \text{-}m)\ \text{-}\beta)$ It is clear $\mathtt{castRefl} \cdot D^{\mathrm{FIX}}$ has type $\mathtt{Cast} \cdot D^{\mathrm{FIX}} \cdot D^{\mathrm{FIX}}$. Since we have from earlier that $\Gamma' \vdash m : \mathtt{Mono} \cdot D^{\mathrm{FI}}$, we know that $\mathtt{outCV}\ \text{-}m$ has type $D^{\mathrm{FIX}} \to D^{\mathrm{FI}} \cdot D^{\mathrm{FIX}}$, and since this is clearly convertible with `outCVU`, we have that the exression $\mathtt{intrView}\ \mathtt{outCVU}\ \text{-}(\mathtt{outCV}\ \text{-}m)\ \text{-}\beta$ had type $\mathtt{View} \cdot (D^{\mathrm{FIX}} \to D^{\mathrm{FI}} \cdot D^{\mathrm{FIX}})\ \mathtt{outCVU}$.

This gives us that the entire expression has type $\mathcal{E}(\mathtt{Is}/D) \cdot D^{\mathrm{FIX}} = \mathtt{RCV} \cdot D^{\mathrm{FI}} \cdot D^{\mathrm{FIX}} \cdot D^{\mathrm{FIX}}$.

[] $\Gamma' \vdash \mathcal{E}(\mathtt{to}/D) : \forall\, R : \star.\, \forall\, is : \mathcal{E}(\mathtt{Is}/D) \cdot R.\, R \to D^{\mathbf{FIX}}$ **and** $|\mathcal{E}(\mathtt{to}/D)| =_{\beta\eta} \lambda\, x.\, x$
We have that $\mathcal{E}(\mathtt{to}/D) = \Lambda\, R.\, \lambda\, x.\, \mathtt{elimCast}\ \text{-}(\pi_1\ x)$, and the desired result follows from the definition of $\mathcal{E}(\mathtt{Is}/D)$ and the erasure of `elimCast`. □

2 Functions over Dataypes

2.1 Value preservation

Lemma 5 (Soundness: Value-preservation of μ and σ)**.** *The elaborated lambda expresions of μ and σ expressions are confluent with the elaborations of the terms they step to, i.e.:*

- *If* $\Gamma \vdash \mu\ ih.\ (c_j\ \overline{p})\ \{\ c_i\ \overline{a} \to q_i\}_{i=1..n} \hookrightarrow p_1'$, *and*
  - $\mu\ ih.\ (c_j\ \overline{p})\ \{\ c_i\ \overline{a} \to q_i\}_{i=1..n} \rightsquigarrow p_2$, *and*
  - $\Gamma \vdash p_2 \hookrightarrow p_2'$,

  then $p_1' =_{\beta\eta} p_2'$
- *If* $\Gamma \vdash \sigma\ (c_j\ \overline{p})\ \{c_i\ \overline{a_i} \to q_i\}_{i=1..n} \hookrightarrow p_1'$, *and*
  - $\sigma\ (c_j\ \overline{p})\ \{c_i\ \overline{a_i} \to q_i\}_{i=1..n} \rightsquigarrow p_2$, *and*
  - $\Gamma \vdash p_2 \hookrightarrow p_2'$

  then $p_1' =_{\beta\eta} p_2'$

Proof.

Case μ: By elaboration of untyped μ-expressions in Figure 26, we know that our μ-expression elaborates as

$$p_1' = |\mathtt{mu}|\ (|c_j^{\mathrm{FIX}}|\ \overline{p'})\ (\lambda\, ih.\, \lambda\, x.\, x\ (\lambda\, \overline{a_i}.\, q_i')_{i=1..n})$$

and by inversion of reduction rule for μ we have the form of the single step reduction of the μ-expressions with know the shape of p_2

$$\frac{1 \leq j \leq n \quad \#\overline{p} = \#\overline{a_j} \quad r = \lambda\, x.\, \mu\ ih.\ x\ \{\ c_i\ \overline{a_i} \to q_i\}_{i=1..n}}{\mu\ ih.\ (c_j\ \overline{p})\ \{\ c_i\ \overline{a_i} \to q_i\}_{i=1..n} \rightsquigarrow \overline{[p/a_j]}[r/ih]q_j}$$

Now, the elaboration p_2' of p_2 has the form

$$\Gamma \vdash \overline{[p/a_j]}[r/ih]t_j \hookrightarrow \overline{[p'/a_j]}[r'/ih]{q_j}'$$

where r is as in the premise of the reduction rule. Appealing again to the rules for elaboration of untyped μ expressions in Figure 26, we see that the elaboration r' of r is.

$$r' = \lambda\, x.\, |\mathtt{mu}|\ x\ (\lambda\, ih.\, \lambda\, x.\, x\ (\lambda\, \overline{a_i}.\, q_i')_{i=1..n})$$

Now, recall that $|c_j^{\mathrm{FIX}}|\ \overline{p}$ is by the rules of Figure 20 convertible with $\mathtt{inCV}\ (|c^{\mathrm{FI}}|\ \overline{p})$. By $\mathtt{muComp}$ in Figure 15, we have

$$p_1' =_{\beta\eta} (\lambda\, ih.\, \lambda\, x.\, x\ (\lambda\, \overline{a_i}.\, q_i')_{i=1\ldots n})\ r'\ (|c_j^{\mathrm{FI}}|\ \overline{p})$$

which further simplifies to

$$p_1' =_{\beta\eta} |c_j^{\mathrm{FI}}|\ \overline{p}\ (\lambda\, \overline{a_i}.\, [r'/ih]q_i')_{i=1\ldots n}$$

(the same r' that occurs in the elaboration of p_2 shows up in the equivalence given by $\mathtt{muComp}$ applied to p_1').

Referring to the rules for the elaboration of c_j^{FI}, we see that it's body is formed by dependent intersection, so its erasure is equal to the erasure of the first component of that intersection, i.e., $|c_j^{\mathrm{F}}|$. We further have

$$p_1' =_{\beta\eta} (\lambda\, x_{i=1\ldots n}.\, x_j\ \overline{p})\ (\lambda\, \overline{a_i}.\, [r'/ih]q_i')_{i=1\ldots n}$$

which simplifies to

$$p_1' =_{\beta\eta} (\lambda\, \overline{a_j}.\, [r'/ih]q_j')\ \overline{p}$$

Since by assumption $\#\overline{p} = \#\overline{a_j}$, we finally have

$$p_1' =_{\beta\eta} \overline{[p/a_j]}[r'/ih]q_j' = p_2'$$

Case σ: Similar to the case for μ. By the untyped elaboration rule for σ expressions in Figure 26, we know that

$$p_1' = |\texttt{sigma}|\ (|c_j^{\text{FIX}}|\ \overline{p})\ (\lambda\, x.\, x\ (\lambda\, \overline{a_i}.\, q_i')_{i=1..n})$$

where each $\overline{p'}$ and q_i' are the elaboration of $\overline{p}$, q_i

By inversion of the reduction rule for σ we have the form of the expression that our σ expression steps to, p_2, is forced by the inference rule

$$\frac{1 \leq j \leq n \quad \#\overline{p} = \#\overline{a_j}}{\sigma\ (c_j\ \overline{p})\ \{c_i\ \overline{a_i} \to q_i\}_{i=1..n} \rightsquigarrow \overline{[p/a_j]}q_j}$$

The elaboration of *this*, p_2', thus has the form

$$p_2' = \overline{[p/a_j]}q_j$$

Just as we noted in the case for μ, we observe that $|c_j^{\text{FIX}}|\ \overline{p}$ reduces to $\texttt{inCV}\ \lambda\, x_{i=1\ldots n}.\, x_j\ \overline{p}$. Now, by `sigComp` in Figure 15, we have

$$p_1' =_{\beta\eta} (\lambda\, x.\, x\ (\lambda\, \overline{a_i}.\, q_i)_{i=1\ldots n})\ \lambda\, x_{i=1\ldots n}.\, x_j\ \overline{p}$$

which reduces to

$$p_1' =_{\beta\eta} (\lambda\, x_{i=1\ldots n}.\, x_j\ \overline{p})\ (\lambda\, \overline{a_i}.\, q_i)_{i=1\ldots n}$$

which reduces to

$$p_1' =_{\beta\eta} (\lambda\, \overline{a_j}.\, q_j)\ \overline{p}$$

which finally reduces to

$$p_1' =_{\beta\eta} \overline{[p/a_j]}q_j = p_2'$$

□

Corollary 4.1 (Generalization of Lemma 5)**.**

- *If* $\Gamma \vdash p_1 \hookrightarrow p_1'$, $\Gamma \vdash p_2 \hookrightarrow p_2'$, *and* $p_1 =_{\beta\eta\mu} p_2$, *then* $p_1' =_{\beta\eta} p_2'$

Proof. By a straightforward induction on the assumed derivations. The extension of the convertibility relation in the surface language are the μ and σ reduction rules and the axiom for identifiers of the form $\texttt{to/}D$, so we appeal to Lemma 5 and the observation that if $\text{IndEl}[D, R, \Delta, m, L, \Theta, \mathcal{E}] \in \Gamma$ then $|\mathcal{E}(\texttt{to/}D)| = \lambda\, x.\, x$. □

Theorem 5 (Value preservation)**.** *If* $\Gamma \vdash t_1 : T_1 \hookrightarrow t_1'$ *and* $\Gamma \vdash t_2 : T_2 \hookrightarrow t_2'$ *and* $|t_1| =_{\beta\eta\mu} |t_2|$ *then* $|t_1'| =_{\beta\eta} |t_2'|$.

Proof. Appeal to Lemma 2 to obtain that $\Gamma \vdash |t_1| \hookrightarrow |t_1'|$ and $\Gamma \vdash |t_2| \hookrightarrow |t_2'|$, then appeal to Corollary 4.1 to finish. □

2.2 Type Preservation

Theorem 6 (Type preservation)**.** *If* $\vdash \Gamma \hookrightarrow \Gamma'$ *then:*

- *If* $\Gamma \vdash K \hookrightarrow K'$ *then* $\Gamma' \vdash K'$
- *If* $\Gamma \vdash T : K \hookrightarrow T'$ *then for some* K', $\Gamma' \vdash K'$ *and* $\Gamma' \vdash T' : K'$
- *If* $\Gamma \vdash t : T \hookrightarrow t'$ *then for some* T', $\Gamma \vdash T : \star \hookrightarrow T'$ *and* $\Gamma' \vdash t' : T'$

Proof. By mutual induction on the assumed derivation.

Term: $\Gamma \vdash t : T \hookrightarrow t'\ [\sigma]$ By inversion, the assumed typing derivation is

$$\frac{\begin{array}{c}\mathrm{IndEl}[D, R, \Delta, m, L, \Theta, \mathcal{E}] \in \Gamma \quad \Gamma \vdash is : \mathtt{Is}/D \cdot T \hookrightarrow is' \\ \Gamma \vdash t : T \hookrightarrow t' \quad \Gamma \vdash T : \star \hookrightarrow T' \quad \Gamma \vdash P : D \to \star \hookrightarrow P' \\ \Gamma \vdash \{c_i\ \overline{a_i} \to t_i\}_{i=1\ldots\#\Delta} : (P, is) \hookrightarrow (\overset{\lambda}{\Lambda}\, \overline{a_i}.\, t'_i)_{i=1\ldots\#\Delta} \quad (z \notin FV(t'_i))_{i=1\ldots\#\Delta}\end{array}}{\begin{array}{c}\sigma\texttt{<}is\texttt{>}\ t\ @P\ \{\ c_i\ \overline{a_i} \to t_i\}_{i=1..\#\Delta} : P\ (\mathtt{to}/D\ \texttt{-}is\ t) \hookrightarrow \mathtt{sigma}\ \texttt{-}m\ \texttt{-}is'\ t' \cdot P' \\ \lambda\, x.\ x.2 \cdot (L \cdot P' \cdot T'\ is')\ (\overset{\lambda}{\Lambda}\, \overline{a_i}.\, \Lambda\, z.\, t'_i)_{i=1..\#\Delta}\ \texttt{-(}\mathtt{intrView}\ \beta\{x.1\}\ \texttt{-}x\ \texttt{-}\beta\texttt{)}\end{array}}$$

and we may further invert the judgement for case branches:

$$\frac{\begin{array}{c}\mathrm{IndEl}[D, R, \Delta, m, L, \Theta, \mathcal{E}] \in \Gamma \quad \Gamma \vdash is : \mathtt{Is}/D \cdot S \hookrightarrow is' \\ (c_i : \overset{\Pi}{\forall}\, \overline{a_i : A_i}.\, D \in \Delta)_{i=1\ldots\#\Delta} \\ (\Gamma\ ;\ \mathtt{to}/D\ \texttt{-}is \vdash ([S/R]\overline{a_i : A_i}) \leqslant ([D/R]\overline{a_i : A_i}) \hookrightarrow \overline{s_i})_{i=1\ldots\#\Delta} \\ (\Gamma \vdash \overset{\lambda}{\Lambda}\, \overline{a_i}.\, t_i : \overset{\Pi}{\forall}\, [S/R]\overline{a_i : A_i}.\, P\ (c_i\ \overline{s_i}) \hookrightarrow \overset{\lambda}{\Lambda}\, \overline{a_i}.\, t'_i)_{i=1\ldots\#\Delta}\end{array}}{\Gamma \vdash \{c_i\ \overline{a_i} \to t_i\}_{i=1\ldots\#\Delta} : (P, is) \hookrightarrow (\overset{\lambda}{\Lambda}\, \overline{a_i}.\, t'_i)_{i=1\ldots\#\Delta}}\ \textsc{Case}$$

By use of the inductive hypothesis (IH) on the premises of these rules:

- from $\Gamma \vdash T : \star \hookrightarrow T'$

  we have $\Gamma' \vdash T' : \star$ by mutual induction on type elaboration, and by inversion of kind elaboration ($\Gamma \vdash \star \hookrightarrow \star$)

- from $\Gamma \vdash t : T \hookrightarrow t'$

  we have $\Gamma \vdash t' : T'$ by induction (and by Lemma 3).

- from $\Gamma \vdash is : \mathtt{Is}/D \cdot T \hookrightarrow is'$

  we have $\Gamma' \vdash is' : \mathtt{RCV} \cdot D^{\mathrm{FI}} \cdot \mu(\mathtt{CV} \cdot D^{\mathrm{FI}}) \cdot T'$ by induction and by inversion on type elaboration rules. Here, D^{FI} is the inductive signature produced by elaboration of datatype declarations.

- from $\Gamma \vdash P : D \to \star \hookrightarrow P'$

  we have $\Gamma' \vdash P' : \mu(\mathtt{CV} \cdot D^{\mathrm{FI}}) \to \star$ by mutual induction and by inversion of kind elaboration

- from $(c_i : \overset{\Pi}{\forall}\, \overline{a_i : A_i} : D.\ \in \Delta)_{i=1\ldots\#\Delta}$

  we have the family of typing derivations $(\Gamma' \vdash c_i^{\mathrm{FIX}} : \overset{\Pi}{\forall}\, \overline{a_i : [\mu(\mathtt{CV} \cdot D^{\mathrm{FI}})/R]A'_i}.\, \mu(\mathtt{CV} \cdot D^{\mathrm{FI}}))_{i=1\ldots\#\Delta}$,

  where $(\Gamma \vdash \overset{\Pi}{\forall}\, \overline{a_i : [D/R]A_i}.\, D : \star \hookrightarrow \overset{\Pi}{\forall}\, \overline{a_i : [\mu(\mathtt{CV} \cdot D^{\mathrm{FI}})/R]A'_i}.\, \mu(\mathtt{CV} \cdot D^{\mathrm{FI}}))_{i=1\ldots\#\Delta}$

  This is from Theorem 4, whose assumptions are satisfied:

  - $\mathrm{IndEl}[D, R, \Delta, m, L, \Theta, \mathcal{E}] \in \Gamma$, which can only occur after elaborating a datatype declaration (Figure 20) (we implicitly using weakening here)
  - we have that $\Gamma, X : \star, R : \star \vdash \overset{\Pi}{\forall}\, \overline{a_i : A_i}.\, X \hookrightarrow \overset{\Pi}{\forall}\, \overline{a_i : A'_i}.\, X$ implies that $\Gamma', R : \star, X : \star \vdash \overset{\Pi}{\forall}\, \overline{a_i : A'_i} : X.\ : \star$ for $i = 1 \ldots \#\Delta$.

    This is by induction (note that before we weaken the context to Γ the antecedent is a sub-derivation, since we have elaborated datatype D) and inversion of type elaboration.

- from $(\Gamma; \texttt{to/}D\ \texttt{-}is \vdash [T/R]\overline{a_i : A_i} \leqslant [D/R]\overline{a_i : A_i} \hookrightarrow \overline{s_i})_{i=1..\#\Delta}$
  - we have $(\Gamma \vdash [T/R](\overline{a_i : A_i}) \vdash \overline{s_i} : [D/R](\overline{a_i : A_i}) \hookrightarrow \overline{s_i'})_{i=1..\#\Delta}$ and $|\overline{s_i}| =_{\beta\eta\mu} |\overline{a_i}|$ by Theorem 3, so by Theorem 5 we have $|\overline{s_i'}| =_{\beta\eta} |\overline{a_i}|$.
  - and have $\Gamma'; \mathcal{E}(\texttt{to/}D)\ \texttt{-}is' \vdash [T'/R](\overline{a_i : A_i'}) \leqslant [\mu(\texttt{CV} \cdot D^{\mathrm{FI}})/R](\overline{a_i : A_i'}) \hookrightarrow \overline{s_i'}$ by Corollary 3.1 and Theorem 3.
  - and finally $\Gamma \vdash [D/R]^{\Pi}_{\forall}\ \overline{a_i : A_i}.\ P\ (c_i\ \overline{s_i}) : \star \hookrightarrow [\mu(\texttt{CV} \cdot D^{\mathrm{FI}})/R]^{\Pi}_{\forall}\ \overline{a_i : A_i'}.\ P'\ (\mathcal{E}(c_i)\ \overline{s_i'})$
    the derivation of which we can easily obtain from the pieces elaborated already
- from the family of derivations $(\Gamma \vdash \overset{\lambda}{\Lambda}\ \overline{a_i}.\ t_i : [T/R]^{\Pi}_{\forall}\ \overline{a_i : A_i}.\ P\ (c_i\ \overline{s_i}) \hookrightarrow \overset{\lambda}{\Lambda}\ \overline{a_i}.\ t_i')_{i=1..\#\Delta}$
  we have $(\Gamma' \vdash \overset{\lambda}{\Lambda}\ \overline{a_i}.\ t_i' : \overset{\Pi}{\forall}\ \overline{a_i : A_i'}[T'/R].\ P'\ (\mathcal{E}(c_i)\ \overline{s_i'}))_{i=1..\#\Delta}$ by induction, and by inversion of type elaboration

It is clear that $P'\ (\mathcal{E}(\texttt{to/}D)\ \texttt{-}is'\ t')$ is the elaboration of $P\ (\texttt{to/}D\ \texttt{-}is\ t)$ and has kind $\star$ under Γ'. The goal is thus to show

$$\Gamma' \vdash \texttt{sigma}\ \texttt{-}is'\ t'\ \cdot P'\ (\lambda\, x.\ x.2 \cdot (L \cdot P' \cdot T'\ is')\ (\overset{\lambda}{\Lambda}\ \overline{a_i}.\ \Lambda\, z.\ t_i')_{i=1..\#\Delta}\ \texttt{-}(\texttt{intrView}\ \beta\{x.1\}\ \texttt{-}x\ \texttt{-}\beta)) : P'\ (\mathcal{E}(\texttt{to/}D)\ \texttt{-}is'\ t')$$

which we do by walking the sequence of arguments to `sigma`, checking types as we go

- $\Gamma' \vdash is' : \mathcal{E}(\texttt{Is/}D) \cdot T'$, established above
- $\Gamma' \vdash t' : T'$, established above
- $\Gamma' \vdash P' : \mathcal{E}(D) \to \star$, established above
- $\Gamma' \vdash \lambda\, x.\ x.2 \cdot (L \cdot P' \cdot T'\ is')\ (\overset{\lambda}{\Lambda}\ \overline{a_i}.\ \Lambda\, z.\ t_i')_{i=1..\#\Delta}\ \texttt{-}(\texttt{intrView}\ \beta\{x.1\}\ \texttt{-}x\ \texttt{-}\beta) : \Pi\ x : D^{\mathrm{FI}} \cdot T'.\ P\ (\texttt{inCV}\ \texttt{-}m\ (\texttt{elimMono}\ \texttt{-}m\ \texttt{-}(\pi_1\ is')\ x))$
  If we can show this, then the type of the entire `sigma` expression is $P'\ (\mathcal{E}(\texttt{to/}D)\ \texttt{-}is'\ t')$ as required
  - we observe that the type we are trying to assign this term is well kinded
  - extend the typing context with $x : D^{\mathrm{FI}} \cdot T'$
  - the head of the application, $x.2$, has type $\forall\, X : D^{\mathrm{F}} \cdot T' \to \star.\ (\Pi\ x_i : \overset{\Pi}{\forall}\ \overline{a_i : A_i'}[T'/R].\ X\ (c_i^{\mathrm{F}}\ \overline{a_i}))_{i=1\ldots\#\Delta}.\ X\ x.1$
  - by Theorem 4 and weakening, we have $\Gamma, x : D^{\mathrm{FI}} \cdot T' \vdash L : (\mathcal{E}(D) \to \star) \to \Pi\ R{:}\star.\, \mathcal{E}(Is/\mathrm{D}) \cdot R \to D^{\mathrm{F}} \cdot R \to \star$
    we can quickly see then that $L \cdot P' \cdot T'\ is'$ has type $D^{\mathrm{FI}} \cdot R \to \star$ under the current typing context
  - we must now check that the remaining arguments can be given to a function of type

$$(\Pi\ x_i : \overset{\Pi}{\forall}\ \overline{a_i : A_i}.\ L' \cdot P' \cdot T'\ is'\ (c^{\mathrm{F}}\ \overline{a_i}))_{i=1\ldots\#\Delta}.\ L \cdot P' \cdot T'\ is'\ x.1$$

    For each $i = 1 \ldots \#\Delta$
    - we have from above that $\Gamma' \vdash \overset{\lambda}{\Lambda}\ \overline{a_i}.\ t_i' : \overset{\Pi}{\forall}\ \overline{a_i : A_i}[T'/R].\ P'(\mathcal{E}(c_i)\ \overline{s_i'})$
    - the goal is to show
$$\Gamma' \vdash \overset{\lambda}{\Lambda}\ \overline{a_i}.\ \Lambda\, z.\ t_i' : \overset{\Pi}{\forall}\ \overline{a_i : A_i'}[T'/R].\ L \cdot P' \cdot T'\ is'\ (c_i^{\mathrm{F}}\ \overline{a_i})$$
    - extend the context by $\overline{a_i : A_i'}[T'/R]$ and $z : \texttt{View} \cdot (D^{\mathrm{FI}} \cdot T')\ \beta\{c_i^{\mathrm{F}}\ \overline{a_i'}\}$
    - now the goal is to show t_i' has type
$$P\ (\texttt{inCV}\ \texttt{-}m\ (\texttt{elimMono}\ \texttt{-}m\ \texttt{-}(\pi_1\ is')\ (\texttt{elimView}\ \beta\{c_i^{\mathrm{F}}\ \overline{a_i}\}\ \texttt{-}z)))$$
    - by weakening, conversion and inversion of typing derivations, we know that under the current context t_i' has type
$$P\ (\texttt{inCV}\ \texttt{-}m\ (c_i^{\mathrm{FI}}\ \overline{s_i'}))$$
    - and these two types are convertible: $\overline{s_i'}$ is convertible with $\overline{a_i}$, c^{FI} is convertible with c^{F}, and by the erasure rules for `View` and `Mono` these type coercions disappear
  - the last expression must be an argument of type $\texttt{View} \cdot (D^{\mathrm{FI}}\ \cdot T')\ \beta\{x.1\}$.
    Since x has this this type and is convertible with $x.1$, the introduction form $\texttt{intrView}\ \beta\{x.1\}\ \texttt{-}x\ \texttt{-}\beta$ is well typed.

Term: $\Gamma \vdash t : T \hookrightarrow t' \ [\mu]$ By inversion, our assumed typing derivation is

$$\frac{\begin{array}{c}\mathrm{IndEl}[D, R, \Delta, m, L, \Theta, \mathcal{E}] \in \Gamma \quad \Gamma \vdash P : D \to \star \hookrightarrow P' \quad \Gamma \vdash t : D \hookrightarrow t' \\ \Gamma' =_{\mathbf{df}} \Gamma, \mathtt{Type/}ih : \star, \mathtt{isType/}ih : \mathtt{Is/}D \cdot \mathtt{Type/}ih, ih : \Pi\ y{:}\mathtt{Type/}ih.\ P\ (\mathtt{to/}D\ \mathtt{-isType/}ih\ y) \\ \Gamma' \vdash \{c_i\ \overline{a_i} \to t_i\}_{i=1\ldots\#\Delta} : (P, \mathtt{isType/}ih) \hookrightarrow (\overline{\lambda\!\!\lambda\ a_i}.\ t_i)_{i=1\ldots\#\Delta} \quad (z \notin FV(t_i))_{i=1\ldots\#\Delta}\end{array}}{\begin{array}{c}\Gamma \vdash \mu\ ih.\ t\ @P\ \{\ c_i\ \overline{a_i} \to t_i\}_{i=1..\#\Delta} : P\ t \hookrightarrow \\ \mathtt{mu}\ \text{-}m\ t' \cdot P'\ \Lambda\,\mathtt{Type/}ih.\ \Lambda\,\mathtt{isType/}ih.\ \lambda\, ih.\ \lambda\, x. \\ x.2 \cdot (L \cdot P' \cdot \mathtt{Type/}ih\ \mathtt{isType/}ih)\ (\overline{\lambda\!\!\lambda\ a_i}.\ \Lambda\, z.\ t_i')_{i=1..\#\Delta}\ \text{-}(\mathtt{intrView}\ \beta\{x.1\}\ \text{-}x\ \text{-}\beta)\end{array}}$$

and we may further invert the the judgment Case for case branches:

$$\frac{\begin{array}{c}\mathrm{IndEl}[D, R, \Delta, m, L, \Theta, \mathcal{E}] \in \Gamma \quad \Gamma \vdash is : \mathtt{Is/}D \cdot S \hookrightarrow is' \\ (c_i : \overset{\Pi}{\forall}\ \overline{a_i : A_i}.\ D \in \Delta)_{i=1\ldots\#\Delta} \\ (\Gamma\,;\, \mathtt{to/}D\ \text{-}is \vdash ([S/R]\overline{a_i : A_i}) \leqslant ([D/R]\overline{a_i : A_i}) \hookrightarrow \overline{s_i})_{i=1\ldots\#\Delta} \\ (\Gamma \vdash \overline{\lambda\!\!\lambda\ a_i}.\ t_i : \overset{\Pi}{\forall}\ [S/R]\overline{a_i : A_i}.\ P\ (c_i\ \overline{s_i}) \hookrightarrow \overline{\lambda\!\!\lambda\ a_i}.\ t_i')_{i=1\ldots\#\Delta}\end{array}}{\Gamma \vdash \{c_i\ \overline{a_i} \to t_i\}_{i=1\ldots\#\Delta} : (P, is) \hookrightarrow (\overline{\lambda\!\!\lambda\ a_i}.\ t_i')_{i=1\ldots\#\Delta}}\ \textsc{Case}$$

and we have also $\vdash \Gamma \hookrightarrow \Gamma''$ (to avoid a name clash with the extended context Γ' in premise of the assumed derivation).

The proof proceeds similarly for the cases for σ above. By use of the inductive hypothesis on the premsises of these rules:

- $\Gamma \vdash T : \star \hookrightarrow T'$ yields $\Gamma'' \vdash T' : \star$ by mutual induction on type elaboration and by inversion of kind elaboration
- $\Gamma \vdash t : T \hookrightarrow t$ yields $\Gamma'' \vdash t' : T'$ by induction and by Lemma 3.
- $\Gamma \vdash P : D \to \star \hookrightarrow P'$ yeilds $\Gamma'' \vdash P' : \mathcal{E}(D) \to \star$ by mutual induction and inversion of kind elaboration.
- the family $(c_i : \overset{\Pi}{\forall}\ \overline{a_i : A_i}.D \in \Delta)_{i=1..\#\Delta}$
  yields $(\Gamma'' \vdash \mathcal{E}(c_i) : [\mathcal{E}(D)/R]\overset{\Pi}{\forall}\ \overline{a_i : A_i'}.\ \mathcal{E}(D))_{i=1..\#\Delta}$ by Theorem 4, whose assumptions we satisfy below
  - we obtain $\Gamma \vdash \mathrm{Ind}[D, R, \Delta] \dashv \Gamma, \mathrm{IndEl}[D, R, \Delta, m, L, \Theta, E]$ from the premise $\mathrm{IndEl}[D, R, \Delta, m, L, \Theta, \mathcal{E}] \in \Gamma$ and context weakening.
  - we have that $(\Gamma, X : \star, R : \star \vdash \overset{\Pi}{\forall}\ \overline{a_i : A_i}.\ X : \star \hookrightarrow \overset{\Pi}{\forall}\ \overline{a_i : A_i'}.\ X$ implies $\Gamma'', R : \star, X : \star \vdash \overset{\Pi}{\forall}\ \overline{a_i : A_i'}.\ X : \star)_{i=1..\#\Delta}$ as an instance of our induction hypothesis (note that before we weaken the context to Γ the antecedent is a sub-derivation, since we have elaborated type D) and by inversion of type elaboration

  and can also construct $(\Gamma \vdash c_i : [D/R]\overset{\Pi}{\forall}\ \overline{a_i : A_i}.\ D)_{i=1..\#\Delta} \hookrightarrow \mathcal{E}(c_i)$ directly (Figure 25)
- also from Theorem 4 we have $\Gamma'' \vdash m : \mathtt{Mono} \cdot D^{\mathrm{FI}}$ and $\Gamma'' \vdash L : (\mathcal{E}(D) \to \star) \to \Pi\ R{:}\star.\, \mathcal{E}(\mathtt{Is/}D) \cdot R \to D^{\mathrm{F}} \cdot R \to \star$
- from the assumption $\vdash \Gamma \hookrightarrow \Gamma''$, we can construct a derivation of $\vdash \Gamma' \hookrightarrow \Gamma'', \mathtt{Type/}ih : \star, \mathtt{isType/}ih : \mathcal{E}(\mathtt{Type/}ih) \cdot \mathtt{Type/}ih, ih : \Pi\ y{:}\mathtt{Type/}ih.\ P'\ (\mathcal{E}(\mathtt{to/}D\ \mathtt{-isType/}ih\ y)$ using of Theorem 4. Call the resulting elaborated context Γ'''
- the family of premises $(\Gamma'; \mathtt{to/}D\ \mathtt{-isType/}ih \vdash (\overline{a_i : A_i})[\mathtt{Type/}ih/R] \leqslant (\overline{a_i : A_i})[D/R] \hookrightarrow \overline{s_i})_{i=1..\#\Delta}$
  - yields $(\Gamma'; (\overline{a_i : A_i})[\mathtt{Type/}ih/R] \vdash \overline{s_i} : (\overline{a_i : A_i})[D/R] \hookrightarrow \overline{s_i'})_{i=1..\#\Delta}$ and $|\overline{s_i}| \cong |\overline{a_i}|$ by Theorem 3
    (by value-preservation, Theorem 5, we have $|\overline{s_i'}| \cong |\overline{a_i}|$)
  - which yields $\Gamma'''; \mathcal{E}(\mathtt{to/}D)\ \mathtt{-isType/}ih \vdash (\overline{a_i : A_i'})[\mathtt{Type/}ih/R] \leqslant (\overline{a_i : A_i'})[D^{\mathrm{FIX}}/R] \hookrightarrow \overline{s_i'}$ by Corollary 3.1 and Theorem 3
  - and we have $(\Gamma' \vdash [D/R]\overset{\Pi}{\forall}\ \overline{a_i : A_i}.\ P\ (c_i\ \overline{s_i}) : \star \hookrightarrow [D^{\mathrm{FIX}}/R]\overset{\Pi}{\forall}\ \overline{a_i : A_i'}.\ P'\ (\mathcal{E}(c_i)\ \overline{s_i'}))$
    easily obtained from the pieces we have elaborated already
- premise $(\Gamma' \vdash \overline{\lambda\!\!\lambda\ a_i}.\ t_i : [\mathtt{Type/}ih/R]\overset{\Pi}{\forall}\ \overline{a_i : A_i}.\ P\ (c_i\ \overline{s_i}) \hookrightarrow \overline{\lambda\!\!\lambda\ a_i}.\ t_i')_{i=1..\#\Delta}$
  yields $(\Gamma''' \vdash \overline{\lambda\!\!\lambda\ a_i}.\ t_i' : [\mathtt{Type/}ih/R]\overset{\Pi}{\forall}\ \overline{a_i : A_i'}.\ P'\ (\mathcal{E}(c_i)\ \overline{s_i'}))_{i=1..\#\Delta}$ by induction and by inversion of type elaboration.

The goal

$\Gamma'' \vdash \texttt{mu}\ \texttt{-}m\ t' \cdot P'\ (\Lambda\, \texttt{Type/}ih.\ \Lambda\, \texttt{isType/}ih.\ \lambda\, ih.\ \lambda\, x.x.2 \cdot (L \cdot P' \cdot \texttt{Type/}ih\ \texttt{isType/}ih\ (\overset{\lambda}{\Lambda}\ \overline{a_i}.\ \Lambda\, z.\, t'_i)_{i=1..\#\Delta}\ \texttt{-}(\texttt{intrView}\ \beta\{x.1\}\ \texttt{-}x\ \texttt{-}\beta)) : P'\ t'$

We walk the sequence of arguments in this expression, checking types as we go

- we have $\Gamma'' \vdash D^{\mathrm{FI}} : \star \to \star$ from Theorem 4
- $\Gamma'' \vdash m : \texttt{Mono} \cdot D^{\mathrm{FI}}$ by the same
- $\Gamma'' \vdash t' : D^{\mathrm{FIX}}$ from above
- $\Gamma'' \vdash P' : D^{\mathrm{FIX}} \to \star$ from above
- $\Gamma'' \vdash \Lambda\, \texttt{Type/}ih.\ \Lambda\, \texttt{isType/}ih.\ \lambda\, ih.\ \lambda\, x.\, x.2 \cdot (L \cdot P' \cdot \texttt{Type/}ih\ \texttt{isType/}ih)\ (\overset{\lambda}{\Lambda}\ \overline{a_i}.\ \Lambda\, z.\, t'_i)\ \texttt{-}(\texttt{intrView}\ \beta\{x.1\}\ \texttt{-}x\ \texttt{-}\beta) : \texttt{AlgMu} \cdot P'$

  If we can show this, then the entire `mu` expression has the type $P'\ t'$ as required.

  - we observe that the type we are trying to assign this term is well kinded
  - extend the typing context with $\texttt{Type/}ih : \star$, $\texttt{isType/}ih : \mathcal{E}(\texttt{Is/}D) \cdot \texttt{Type/}ih$, and $ih : \Pi\ y : D^{\mathrm{FI}} \cdot \texttt{Type/}ih.\ P\ (\texttt{elimCast}\ \texttt{-}(\pi_1\ \texttt{isType/}ih)\ y)$ and we get precisely Γ'''.
  - extend the typing context with $x : D^{\mathrm{FI}} \cdot \texttt{Type/}ih$
  - the head of the application, $x.2$, has type

    $$\forall\, X : D^{\mathrm{F}} \cdot \texttt{Type/}ih \to \star.\, [\texttt{Type/}ih/R](\Pi\ x_i : \overset{\Pi}{\forall}\ \overline{a_i : A'_i}.\ X\ (c_i^{\mathrm{F}}\ \overline{a_i}))_{i=1..\#\Delta}.\ X\ x.1$$

    by Theorem 4
  - check that $L \cdot P' \cdot \texttt{Type/}ih\ \texttt{isType/}ih$ has kind $D^{\mathrm{F}} \cdot \texttt{Type/}ih \to \star$
    This follows similarly to how we proceeded in the case of σ
  - We must now check the remaining arguments can be given to a function of type

    $$[\texttt{Type/}ih/R](\Pi\ x_i : \overset{\Pi}{\forall}\ \overline{a_i : A'_i}.\ (L \cdot P' \cdot \texttt{Type/}ih\ \texttt{isType/}ih\ (c_i^{\mathrm{F}}\ \overline{a_i})))_{i=1..\#\Delta}.\ L \cdot P' \cdot \texttt{Type/}ih\ \texttt{isType/}ih\ x.1$$

  - For each case $i = 1..\#\Delta$:
    - we have from above that $\Gamma''' \vdash \overset{\lambda}{\Lambda}\ \overline{a_i}.\, t'_i : \overset{\Pi}{\forall}\ \overline{a_i : A'_i}[\texttt{Type/}ih/R].\ P'\ (\mathcal{E}(c_i)\ \overline{s'_i})$
    - the goal is to show

      $$\Gamma''' \vdash \overset{\lambda}{\Lambda}\ \overline{a_i}.\ \Lambda\, z.\, t'_i : \overset{\Pi}{\forall}\ \overline{a_i : A'_i}[\texttt{Type/}ih/R].\ L \cdot P' \cdot \texttt{Type/}ih\ \texttt{isType/}ih\ (c_i^{\mathrm{F}}\ \overline{a_i})$$

    - extend the context by $\overline{a_i : A_i}[\texttt{Type/}ih/R]$ and $z : \texttt{View} \cdot (D^{\mathrm{FI}}) \cdot \texttt{Type/}ih\ \beta\{c_i^{\mathrm{F}}\ \overline{a_i}\}$
    - now the goal is to show t'_i has type

      $$P'\ (\texttt{inCV}\ \texttt{-}m\ (\texttt{elimMono}\ \texttt{-}m\ \texttt{-}(\pi_1\ \texttt{isType/}ih)\ (\texttt{elimView}\ \beta\{c_i^{\mathrm{F}}\ \overline{a_i}\}\ \texttt{-}z)))$$

    - by weakening, conversion, and inversion of typing derivations, we know that t'_i has type

      $$P'\ (\texttt{inCV}\ \texttt{-}m\ (c_i^{\mathrm{FI}}\ \overline{s'_i}))$$

      and these two types are convertible (see the case for σ).
  - the last expression is an argument of type $\texttt{View} \cdot (D^{\mathrm{FI}} \cdot \texttt{Type/}ih)\ \beta\{x.1\}$, by a similar argument as for the σ case.

Term: [c] [to] [is] By appeal to Theorem 4, appealing to the inductive hypothesis to show the elaborations of constructor argument types are well-kinded

Type: [D] [Is] By appeal to Theorem 4, appealing to the inductive hypothesis to show the elaborations of constructor argument types are well-kinded

Kind: Case

$$\frac{}{\Gamma \vdash \star \hookrightarrow \star}$$

Immediate

Kind: Case

$$\frac{\Gamma \vdash K_1 \hookrightarrow K_1' \quad \Gamma, X : K_1 \vdash K_2 \hookrightarrow K_2'}{\Gamma \vdash \Pi\, X : K_1.\, K_2 \hookrightarrow \Pi\, X : K_1'.\, K_2'}$$

- By the IH, $\Gamma' \vdash K_1'$
- By the IH, $\Gamma', X : K_1' \vdash K_2'$
- Conclude $\Gamma' \vdash \Pi\, X : K_1'.\, K_2'$

Kind: Case

$$\frac{\Gamma \vdash T : K_2 \hookrightarrow T' \quad \Gamma, x : T \vdash K_1 \hookrightarrow K_1'}{\Gamma \vdash \Pi\, x : T.\, K_1 \hookrightarrow \Pi\, x : T'.\, K_1'}$$

- By mutual induction, $\Gamma' \vdash T' : K_2'$, for some K_2' where $\Gamma' \vdash K_2'$
- By IH, $\Gamma', x : T' \vdash K_1'$
- Conclude $\Gamma' \vdash \Pi\, x : T'.\, K_1'$

Type: Case

$$\frac{FV(p_1\ p_2) \subseteq dom(\Gamma) \quad \Gamma \vdash p_1 \hookrightarrow p_1' \quad \Gamma \vdash p_2 \hookrightarrow p_2'}{\Gamma \vdash \{p_1 \simeq p_2\} : \star \hookrightarrow \{p_1' \simeq p_2'\}}$$

- Appeal to Lemma 6 to get $FV(p_1'\ p_2') \subseteq dom(\Gamma')$
- Conclude $\Gamma' \vdash \{p_1' \simeq p_2'\} : \star$

Type: Case

$$\frac{\Gamma \vdash T_1 : \star \hookrightarrow T_1' \quad \Gamma, x : T_1 \vdash T_2 : \star \hookrightarrow T_2'}{\Gamma \vdash \iota\, x : T_1.\, T_2 : \star \hookrightarrow \iota\, x : T_1'.\, T_2'}$$

- By IH, $\Gamma' \vdash T_1' : \star$
- By IH, $\Gamma', x : T_1' \vdash T_2' : \star$
- Conclude $\Gamma' \vdash \iota\, x : T_1'.\, T_2' : \star$

Type: Case

$$\frac{\Gamma \vdash T_1 : \star \hookrightarrow T_1' \quad \Gamma, x : T_1 \vdash T_2 : \star T_2'}{\Gamma \vdash \forall\, x : T_1.\, T_2 : \star \hookrightarrow \forall\, x : T_1'.\, T_2'}$$

Similar to above

Type: Case

$$\frac{\Gamma \vdash T_1 : \star \hookrightarrow T_1' \quad \Gamma, x : T_1 \vdash T_2 : \star \hookrightarrow T_2'}{\Gamma \vdash \Pi\, x : T_1.\, T_2 : \star \hookrightarrow \Pi\, x : T_1'.\, T_2'}$$

Similar to above

Type: Case

$$\frac{\Gamma \vdash K \hookrightarrow K' \quad \Gamma, X : K \vdash T : \star \hookrightarrow T'}{\Gamma \vdash \forall\, X : K.\, T : \star \hookrightarrow \forall\, X : K'.\, T'}$$

- By mutual induction, $\Gamma' \vdash K'$
- By IH, $\Gamma', X : K' \vdash T' : \star$
- Conclude $\Gamma' \vdash \forall\, X : K'.\, T' : \star$

Type: Case

$$\frac{\Gamma \vdash S : K_1 \hookrightarrow S' \quad \Gamma, x : S \vdash T : K_2 \hookrightarrow T'}{\Gamma \vdash \lambda\, x : S.\, T : \Pi\, x : S.\, K_2 \hookrightarrow \lambda\, x : S'.\, T'}$$

- By IH, $\Gamma' \vdash S' : K_1'$ for some $K_1{}'$ where $\Gamma' \vdash K_1'$
- By IH, $\Gamma', x : S' \vdash T' : K_2'$ for some K_2' where $\Gamma', X : S' \vdash K_2' : \star$
- Conclude $\Gamma' \vdash \lambda\, x : S'.\, T' : \Pi\, x : S'.\, K_2'$

Type: Case

$$\frac{\Gamma \vdash K_1 \hookrightarrow K_1' \quad \Gamma, X : K_1 \vdash T : K_2 \hookrightarrow T'}{\Gamma \vdash \lambda\, X : K_1.\, T : \Pi\, X : K_1.\, K_2 \hookrightarrow \lambda\, X : K_1'.\, T'}$$

Similar to above, appealing to mutual induction for the kind K_1 of the λ-bound variable X

Type: Case

$$\frac{\Gamma \vdash T_1 : \Pi\, X : K_2.\, K_1 \hookrightarrow T_1' \quad \Gamma \vdash T_2 : K_2 \hookrightarrow T_2'}{\Gamma \vdash T_1 \cdot T_2 : [T_2/X]K_1 \hookrightarrow T_1' \cdot T_2'}$$

- By IH, $\Gamma' \vdash T_1' : \Pi\, K_2' : K_1.$ for some K_2', K_1' s.t. $\Gamma' \vdash \Pi\, X : K_2'.\, K_1$ (inversion of kind elaboration)
- By IH, $\Gamma' \vdash T_2' : K_2'$
- Conclude $\Gamma' \vdash T_1' \cdot T_2' : [T_2'/X]K_1'$

Type: Case

$$\frac{\Gamma \vdash T : \Pi\, x : S.\, K \hookrightarrow T' \quad \Gamma \vdash t : S \hookrightarrow t'}{\Gamma \vdash T\ t : [t/x]K \hookrightarrow T'\ t'}$$

Similar to above, appealing to mutual induction for the elaboration of term argument t

Type: Case

$$\frac{}{\Gamma \vdash X : \Gamma(X) \hookrightarrow X}$$

Immediate (Lemma 6 and Figure 17a)

Term: Case

$$\frac{}{\Gamma \vdash x : \Gamma(x) \hookrightarrow x}$$

Immediate (Lemma 6 and Figure 17a)

Term: Case

$$\frac{\Gamma \vdash S : \star \hookrightarrow S' \quad \Gamma, x : S \vdash t : T \hookrightarrow t'}{\Gamma \vdash \lambda\, x.\, t : \Pi\, x : S.\, T \hookrightarrow \lambda\, x.\, t'}$$

- By mutual induction, $\Gamma' \vdash S' : \star$
- By IH, $\Gamma', x : S' \vdash t' : T'$ for some T' s.t. $\Gamma', x : S' \vdash T' : \star$
- Conclude $\Gamma' \vdash \lambda\, x.\, t' : \Pi\, x : S'.\, T'$

Term: Case

$$\frac{\Gamma \vdash K \hookrightarrow K' \quad X \notin FV(|t|) \quad \Gamma, X : K \vdash t : T \hookrightarrow t'}{\Gamma \vdash \Lambda\, X.\, t : \forall\, X : K.\, T \hookrightarrow \Lambda\, X.\, t'}$$

- By mutual induction, $\Gamma' \vdash K'$
- By IH, $\Gamma', X : K' \vdash t' : T'$ for some T' such that $\Gamma' \vdash T' : \star$
- By Lemma 6, $X \notin FV(|t'|)$
- Conclude $\Gamma' \vdash \Lambda\, X.\, t' : \forall\, X : K'.\, T'$

Term: Case

$$\frac{x \notin FV(|t|) \quad \Gamma, x : S \vdash t : T}{\Gamma \vdash \Lambda\, x.\, t : \forall\, x : S.\, T}$$

Similar to above, invoking mutual induction for elaboration of type S of the Λ-bound variable

Term: Case

$$\frac{\Gamma \vdash t : \Pi\, x : S.\, T \hookrightarrow t' \quad \Gamma \vdash s : S \hookrightarrow s'}{\Gamma \vdash t\ s : [s/x]T \hookrightarrow t'\ s'}$$

- By IH, $\Gamma' \vdash t' : \Pi\, x : S'.\, T'$ for some S' and T' s.t. $\Gamma' \vdash \Pi\, x : S'.\, T' : \star$ (inversion of type elaboration)
- By IH $\Gamma' \vdash s' : S'$
- Conclude $\Gamma' \vdash t'\ s' : [s'/x]T'$

Term: Case

$$\frac{\Gamma \vdash t : \forall\, X : K.\, T \hookrightarrow t' \quad \Gamma \vdash S : K \hookrightarrow S'}{\Gamma \vdash t \cdot S : [S/X]T \hookrightarrow t' \cdot S'}$$

Similar to above, invoking mutual induction for the the type argument S

Term: Case

$$\frac{\Gamma \vdash t : \forall\, x : S.\, T \hookrightarrow t' \quad \Gamma \vdash s : S \hookrightarrow s'}{\Gamma \vdash t\ \text{-}s : [s/x]T \hookrightarrow t'\ \text{-}s'}$$

Similar to above

Term: Case

$$\frac{\Gamma \vdash t : S \hookrightarrow t' \quad S \cong T \quad \Gamma \vdash T : \star \hookrightarrow T'}{\Gamma \vdash t : T \hookrightarrow t'}$$

By the IH, invoking Corollary 4.1 and referencing the type-convertibility relation of [Stu18]

Term: Case

$$\frac{\Gamma \vdash t_1 : T_1 \hookrightarrow t_1' \quad \Gamma \vdash t_2 : [t_1/x]T_2 \hookrightarrow t_2' \quad t_1' \cong t_2'}{\Gamma \vdash [t_1, t_2] : \iota\, x : T_1.\, T_2 \hookrightarrow [t_1', t_2']}$$

- By IH, $\Gamma' \vdash t_1' : T_1'$ for some T_1' s.t $\Gamma' \vdash T_1' : \star$
- By IH, $\Gamma' \vdash t_2' : [t_1'/x]T_2'$ for some T_2' s.t. $\Gamma', x : T_1' \vdash T_2' : \star$
- By Corollary 4.1, $t_1' \cong t_2'$
- Conclude $\Gamma' \vdash [t_1', t_2'] : \iota\, x : T_1'.\, T_2'$

Term: Case

$$\frac{\Gamma \vdash t : \iota\, x{:}T_1.\, T_2 \hookrightarrow t'}{\Gamma \vdash t.1 : T_1 \hookrightarrow t'.1}$$

- By IH, $\Gamma' \vdash t' : \iota\, x{:}T_1'.\, T_2'$ for some T_1', T_2' s.t. $\Gamma' \vdash \iota\, x{:}T_1'.\, T_2' : \star$ (inversion of type elaboration rules)
- Conclude $\Gamma' \vdash t.1' : T_1'$

Term: Case

$$\frac{\Gamma \vdash t : \iota\, x{:}T_1.\, T_2 \hookrightarrow t'}{\Gamma \vdash t.2 : [t.1/x]T_2 \hookrightarrow t'.2}$$

- By IH, $\Gamma' \vdash t' : \iota\, x{:}T_1'.\, T_2'$ for some T_1', T_2' s.t. $\Gamma' \vdash \iota\, x{:}T_1'.\, T_2' : \star$ (inversion of type elaboration rules)
- Conclude $\Gamma' \vdash t'.2 : [t'.1/x]T_2'$

Term: Case

$$\frac{\Gamma \vdash \{|t| \simeq |t|\} : \star \hookrightarrow \{p \simeq p\}}{\Gamma \vdash \beta : \{|t| \simeq |t|\} \hookrightarrow \beta}$$

- By mutual induction $\Gamma' \vdash \{p \simeq p\} : \star$
- Conclude $\Gamma' \vdash \beta : \{p \simeq p\}$

Term: Case

$$\frac{\begin{array}{cc}\Gamma \vdash s : \{|t_1| \simeq |t_2|\} \hookrightarrow s' & \Gamma \vdash t : [t_1/x]T_1 \hookrightarrow t' \\ T_1 \cong T_2 & \Gamma \vdash [t_2/x]T_2 : \star \hookrightarrow [t_2'/x]T_2'\end{array}}{\Gamma \vdash \rho\ s\ @\ x.T_2\ \text{-}\ t : [t_2/x]T_2 \hookrightarrow \rho\ s'\ @\ x.T_2'\ \text{-}\ t'}$$

Since you have made it this far: there's actually quite a subtle issue in [Stu18] concerning ρ, namely that rewriting by untyped equalities may produce a result type that is not re-kindable. This does not effect the denotational semantics (nor soundness proof) for Cedille. We have separately developed kind-preserving rewrite rules for the unguided (no @ $x.T_2$) ρ; the rule here is a specificational one that is consistent with the original rules but requires additional burden on the programmer to provide the correctly-rewritten T_2 that satisfies the premises (being convertible with T_1 modulo erasure and being well-kinded when instantiating t_2 for x)

- By IH, $\Gamma' \vdash s' : \{|t_1'| \simeq |t_2'|\}$ where some t_1' and t_2' s.t. $\Gamma' \vdash \{|t_1'| \simeq |t_2'|\} : \star$ (inversion of type elaboration rules)
- By IH, $\Gamma' \vdash t' : [t_1'/x]T_1'$ for some T_1' such that $\Gamma' \vdash [t_1'/x]T_1' : \star$
- By IH, $\Gamma' \vdash [t_2'/x]T_2' : \star$ for some T_2' s.t. $\Gamma' \vdash [t_2'/x]T_2' : \star$
- By Lemma 4.1 and reference to the type convertibility rules of [Stu18], $T_1' \cong T_2'$
- Conclude $\Gamma' \vdash \rho\ s'\ @\ x.T_2'\ -t' : [t_2'/x]T_2'$

Term: Case

$$\frac{\Gamma \vdash s : \{|t_1| \simeq |t_2|\} \hookrightarrow s' \quad \Gamma \vdash t_1 : T \hookrightarrow t_1' \quad \Gamma \vdash t_2 \hookrightarrow t_2'}{\Gamma \vdash \varphi\ s - t_1\ \{t_2\} : T \hookrightarrow \varphi\ s' - t_1'\ \{t_2'\}}$$

- By IH, $\Gamma' \vdash s' : \{|t_1'| \simeq |t_2'|\}$ for some t_1', t_2' s.t. $\Gamma' \vdash \{|t_1'| \simeq |t_2'|\} : \star$ (inversion of type elaboration rules)
- By IH, $\Gamma' \vdash t_1' : T'$ for some T' s.t. $\Gamma' \vdash T' : \star$
- Conclude $\Gamma' \vdash \varphi\ s' - t_1'\ \{t_2\} : T'$

Term: Case

$$\frac{\Gamma \vdash t : \{\lambda\, x.\, \lambda\, y.\, x \simeq \lambda\, x.\, \lambda\, y.\, x\} \hookrightarrow t' \quad \Gamma \vdash T : \star \hookrightarrow T'}{\Gamma \vdash \delta\ T - t : T \hookrightarrow \delta\ T' - t'}$$

Note that we are considering the restricted form of δ. Proving sound elaboration for the full Böhm-out algorithm, implemented by the Cedille tool, is beyond the scope of our contributions.

- By IH, $\Gamma' \vdash t' : \{\lambda\, x.\, \lambda\, y.\, x \simeq \lambda\, x.\, \lambda\, y.\, y\}$
- By mutual induction, $\Gamma' \vdash T' : \star$
- Conclude $\Gamma' \vdash \delta\ T' - t' : T'$

□

2.3 Termination Guarantee

Theorem 7 (Normalization guarantee)**.** *Suppose* $\Gamma \vdash t : D \hookrightarrow t'$ *and* $\mathit{IndEl}[D, R, \Delta, m, L, \Theta, \mathcal{E}] \in \Gamma$*. Suppose further that* t *is closed. Then* $|t'|$ *is call-by-name normalizing.*

Proof. By Theorem 6, there is a type T such that $\Gamma \vdash D : \star \hookrightarrow T$ and $\Gamma \vdash t' : T'$. By inversion on type elaboration, T is $\mathcal{E}(D)$, which is of the form $\mu(\mathtt{CV} \cdot D^{\mathrm{FI}})$. By Lemma 1, there is an identity function from this type to a Π-type. So, by Theorem 4 of [Stu18], $|t'|$ is call-by-name normalizing. □

2.4 Additional Proofs

Lemma 6.

- *If* $\Gamma \vdash K \hookrightarrow K'$ *then* $FV(K) = FV(K')$
- *If* $\Gamma \vdash T : K \hookrightarrow T'$ *then* $FV(T) = FV(T')$
- *If* $\Gamma \vdash t : T \hookrightarrow t'$ *then* $FV(t) = FV(t')$
- *If* $\vdash \Gamma \hookrightarrow \Gamma'$ *then* $DV(\Gamma) = DV(\Gamma')$

Proof. By a straightforward mutual induction on the assumed derivations. □

Lemma 6 is needed for type-checking the term elaborations of implicit products and kind-checking the elaborations of equality types.

References

[AC19] Andreas Abel and Thierry Coquand. Failure of normalization in impredicative type theory with proof-irrelevant propositional equality, 2019.

[DFS18] Larry Diehl, Denis Firsov, and Aaron Stump. Generic zero-cost reuse for dependent types. *Proc. ACM Program. Lang.*, 2(ICFP):104:1–104:30, jul 2018.

[FBS18] Denis Firsov, Richard Blair, and Aaron Stump. Efficient mendler-style lambda-encodings in cedille. In Jeremy Avigad and Assia Mahboubi, editors, *Interactive Theorem Proving*, pages 235–252, Cham, 2018. Springer International Publishing.

[Stu18] Aaron Stump. Syntax and semantics of Cedille. `https://github.com/cedille/cedille/blob/master/docs/semantics/paper.pdf`, 2018.