Syntax and Semantics of Cedille

Aaron Stump
Computer Science
The University of Iowa
aaron-stump@uiowa.edu

1 Introduction

The type theory of Cedille is called the Calculus of Dependent Lambda Eliminations (CDLE). This document presents the version of CDLE as of June 1, 2018. We have made many changes from the first paper on CDLE [11], mostly in the form of dropping constructs we discovered (to our surprise) could be derived [12]. I have also omitted *lifting* – a technique for large eliminations with lambda encodings – in this document's version of CDLE. Some uses of lifting can be simulated other ways within the system, though the limits of this are still under investigation. We also include a construct δ, for deriving a contradiction from a proof that lambda-encoded true equals lambda-encoded false. This also compensates somewhat for the lack of lifting.

At a high level, CDLE is an extrinsic (i.e., Curry-style) type theory extending the Calculus of Constructions with three additional constructs, which allow deriving induction principles within the theory, for inductive datatypes. The goal is to support usual idioms of dependently typed programming and proving as in Agda or similar tools, but using pure lambda encodings for all data, and requiring a much smaller core theory.

The current Cedille implementation of CDLE extends the system described below with a number of features intended to make programming in the system more convenient and with less redundancy. These features all compile away to a slightly simplified version of the theory presented in this document, called Cedille Core, described here: `https://github.com/astump/cedille-core-spec`.

2 Classification Rules

The classification rules are given in Figures 1, 2, and 3. For brevity, we take these figures as implicitly specifying the syntax of kinds κ, types T, and annotated terms t; these may use term variables x and type variables X, which we assume come from distinct sets. So terms and types are syntactically distinguished. The typing rules (Figure 3) are bidirectional [10], while the kinding and superkinding rules (Figure 2 and 1) are only synthesizing. We write $\Leftrightarrow$ to range over $\{\Leftarrow, \Rightarrow\}$. We follow the syntax of our implementation Cedille, which distinguishes application of a term or type e to a type $(e \cdot T)$, from application to a term $(e\ t)$, and application to an erased term argument $(e\ \text{-}t)$. The rules are intended, with a few points of nondeterminism, to be read bottom-up (in a standard way; cf. [9]) as an algorithm for computing a classifier from a context and an expression $(\Rightarrow)$ or checking an expression against a classifier in context $(\Leftarrow)$.

The classification rules refer to an erasure function, defined in Figure 4. The type theory is *extrinsic* (aka, Curry-style), and hence we only consider erasures $|t|$ of terms when testing for $\beta\eta$-equivalence. This is done by the conversion relation $T \cong T'$, whose central rules are given in Figure 5. That figure omits the various congruence rules needed to equate bigger expressions by equating subexpressions. The main ideas of

$$\frac{}{\Gamma \vdash \star} \qquad \frac{\Gamma \vdash T \Rightarrow \star \quad \Gamma, x : T \vdash \kappa}{\Gamma \vdash \Pi\, x{:}T.\, \kappa} \qquad \frac{\Gamma \vdash \kappa' \quad \Gamma, X : \kappa' \vdash \kappa}{\Gamma \vdash \Pi\, X{:}\kappa'.\, \kappa}$$

Figure 1: Rules for checking that a kind is well-formed ($\Gamma \vdash \kappa$)

$$\frac{(X : \kappa) \in \Gamma}{\Gamma \vdash X \Rightarrow \kappa} \qquad \frac{\Gamma \vdash \kappa \quad \Gamma, X : \kappa \vdash T \Rightarrow \star}{\Gamma \vdash \forall\, X{:}\kappa.\, T \Rightarrow \star}$$

$$\frac{\Gamma \vdash T \Rightarrow \star \quad \Gamma, x : T \vdash T' \Rightarrow \star}{\Gamma \vdash \forall\, x{:}T.\, T' \Rightarrow \star} \qquad \frac{\Gamma \vdash T \Rightarrow \star \quad \Gamma, x : T \vdash T' \Rightarrow \star}{\Gamma \vdash \Pi\, x{:}T.\, T' \Rightarrow \star}$$

$$\frac{\Gamma \vdash T \Rightarrow \star \quad \Gamma, x : T \vdash T' \Rightarrow \kappa}{\Gamma \vdash \lambda\, x{:}T.\, T' \Rightarrow \Pi\, x{:}T.\, \kappa} \qquad \frac{\Gamma \vdash \kappa \quad \Gamma, X : \kappa \vdash T' \Rightarrow \kappa'}{\Gamma \vdash \lambda\, X{:}\kappa.\, T' \Rightarrow \Pi\, X{:}\kappa.\, \kappa'}$$

$$\frac{\Gamma \vdash T \Rightarrow \Pi\, x{:}T'.\, \kappa \quad \Gamma \vdash t \Leftarrow T'}{\Gamma \vdash T\ t \Rightarrow [t/x]\kappa} \qquad \frac{\Gamma \vdash T \Rightarrow \Pi\, X{:}\kappa'.\, \kappa \quad \Gamma \vdash T' \Rightarrow \kappa' \quad \kappa \cong \kappa'}{\Gamma \vdash T \cdot T' \Rightarrow [T'/X]\kappa}$$

$$\frac{\Gamma \vdash T \Rightarrow \star \quad \Gamma, x : T \vdash T' \Rightarrow \star}{\Gamma \vdash \iota\, x{:}T.\, T' \Rightarrow \star} \qquad \frac{FV(t\ t') \subseteq dom(\Gamma)}{\Gamma \vdash \{t \simeq t'\} : \star}$$

Figure 2: Rules for synthesizing a kind for a type ($\Gamma \vdash T \Rightarrow \kappa$)

conversion shown in the figure are to use β-equivalence at the type level, and $\beta\eta$-equivalence of erased terms at the term level.

2.1 Overview of the constructs

CDLE has as a subsystem the extrinsic Calculus of Constructions (CC). We have dependent types $\Pi\, x{:}T.\, T'$ and kinds $\Pi\, x{:}T.\, \kappa$, as well as term- and type-level quantification over (possibly higher-kinded) types $\forall\, X{:}\kappa.\, T$ and $\Pi\, X{:}\kappa.\, \kappa'$. We use $\forall$ when the corresponding argument will be erased, and Π when it will be retained. Since we do not erase term or type arguments from type-level applications, we thus write $\Pi\, X{:}\kappa.\, \kappa'$ instead of $\forall\, X : \kappa.\, \kappa'$. We write λ to correspond to Π and Λ to correspond to $\forall$. As noted above, application to a type is denoted with center dot ($\cdot$).

To Curry-style CC, CDLE adds: implicit products, introduced orginially by Miquel [7]; a primitive equality type $\{t \simeq t'\}$; and dependent intersection types $\iota\, x{:}T.\, T'$, introduced by Kopylov [5]. Implicit products are used for erased arguments to functions, found also in systems like Agda (cf. [8]). Dependent intersections are a rather exotic construct allowing us to assign type $\iota\, x{:}T'.\, T$ to erased term t when we can assign T' to t, and also assign $[t/x]T$ to t. For an annotated introduction form, we write $[t, t']$, where t checks against type T', t' checks against $[t/x]T$, and t and t' have identical (i.e., α-equivalent) erasures. Dependent intersections thus enable a controlled form of self-reference in the type. Previous work showed how to use this to derive induction for Church-encoded natural numbers [12]. We will see below further uses of this construct.

The typing rules include conversion checks in a few places; e.g., as standardly, when switching from checking to synthesizing mode. Two rules near the top of Figure 3 state that one may (nondeterministically) β-reduce the type one is synthesizing or checking, before proceeding. This allows reduction to head-normal form, to match the form of type required by other rules. Finally, we include the construct $\chi\ T$ - t to change the synthesized or checked type T' to T, if $T \cong T'$. This may be necessary to get the type into a specific form for purposes of rewriting with the ρ construct.

$$\frac{(x : t) \in \Gamma}{\Gamma \vdash x \Rightarrow T} \qquad \frac{\Gamma \vdash t \Leftarrow T \quad T' \rightsquigarrow_\beta^* T}{\Gamma \vdash t \Leftarrow T'}$$

$$\frac{\Gamma \vdash t \Rightarrow T \quad T \rightsquigarrow_\beta^* T'}{\Gamma \vdash t \Rightarrow T'} \qquad \frac{\Gamma \vdash t \Rightarrow T' \quad T' \cong T}{\Gamma \vdash t \Leftarrow T}$$

$$\frac{\Gamma, x : T \vdash t \Leftarrow T'}{\Gamma \vdash \lambda\, x.\, t \Leftarrow \Pi\, x{:}T.\, T'} \qquad \frac{\Gamma \vdash t \Rightarrow \Pi\, x{:}T'.\, T \quad \Gamma \vdash t' \Leftarrow T'}{\Gamma \vdash t\ t' \Rightarrow [t'/x]T}$$

$$\frac{\Gamma, X : \kappa \vdash t \Leftarrow T}{\Gamma \vdash \Lambda\, X.\, t \Leftarrow \forall\, X{:}\kappa.\, T} \qquad \frac{\Gamma \vdash t \Rightarrow \forall\, X{:}\kappa.\, T \quad \Gamma \vdash T' \Leftarrow \kappa}{\Gamma \vdash t \cdot T' \Rightarrow [T'/X]T}$$

$$\frac{\Gamma, x : T' \vdash t \Leftarrow T \quad x \notin FV(|t|)}{\Gamma \vdash \Lambda\, x.\, t \Leftarrow \forall\, x{:}T'.\, T} \qquad \frac{\Gamma \vdash t \Rightarrow \forall\, x{:}T'.\, T \quad \Gamma \vdash t' \Leftarrow T'}{\Gamma \vdash t\ \text{-}t' \Rightarrow [t'/x]T}$$

$$\frac{\Gamma \vdash t \Leftarrow T \quad \Gamma \vdash t' \Leftarrow [t/x]T' \quad |t| =_{\beta\eta} |t'|}{\Gamma \vdash [t, t'] \Leftarrow \iota\, x{:}T.\, T'} \qquad \frac{\Gamma \vdash t \Rightarrow \iota\, x{:}T.\, T'}{\Gamma \vdash t.1 \Rightarrow T}$$

$$\frac{\Gamma \vdash t \Rightarrow \iota\, x{:}T.\, T'}{\Gamma \vdash t.2 \Rightarrow [t.1/x]T'} \qquad \frac{\Gamma \vdash FV(t) \subseteq dom(\Gamma)}{\Gamma \vdash \beta\{t'\} \Leftarrow \{t \simeq t\}}$$

$$\frac{\Gamma \vdash t \Leftarrow \{\lambda\, x.\, \lambda\, y.\, x \simeq \lambda\, x.\, \lambda\, y.\, y\}}{\Gamma \vdash \delta\ t \Leftarrow T} \qquad \frac{\Gamma \vdash t' \Rightarrow \{t_1 \simeq t_2\} \quad \Gamma \vdash t \Leftrightarrow [t_1/x]T}{\Gamma \vdash \rho\ t'\ \text{-}\ t \Leftrightarrow [t_2/x]T}$$

$$\frac{\Gamma \vdash T \Leftarrow \star \quad \Gamma \vdash t \Leftarrow T \quad T \cong T'}{\Gamma \vdash \chi\ T\ \text{-}\ t \Leftarrow T'} \qquad \frac{\Gamma \vdash T \Leftarrow \star \quad \Gamma \vdash t \Rightarrow T' \quad T \cong T'}{\Gamma \vdash \chi\ T\ \text{-}\ t \Rightarrow T}$$

$$\frac{\Gamma \vdash t \Rightarrow \{t' \simeq t''\} \quad \Gamma \vdash t' \Leftrightarrow T}{\Gamma \vdash \phi\ t\ \text{-}\ t'\{t''\} \Leftrightarrow T}$$

Figure 3: Rules for checking a term against a well-kinded type ($\Gamma \vdash t \Leftarrow T$) and synthesizing a type for a term ($\Gamma \vdash t \Rightarrow T$)

$$\begin{array}{lcllcl}
|x| & = & x & |\lambda\, x{:}t'.\, t| & = & \lambda\, x.\, |t| \\
|t\ t'| & = & |t|\ |t'| & |t \cdot T| & = & |t| \\
|\Lambda\, x.\, t't| & = & |t| & |t\ \text{-}t'| & = & |t| \\
|[t, t']| & = & |t| & |t.1| & = & |t| \\
|t.2| & = & |t| & |\beta\{t\}| & = & |t| \\
|\delta\ t| & = & |t| & |\rho\ t\ \text{-}\ t'| & = & |t'| \\
|\phi\ t\ \text{-}\ t'\ \{t''\}| & = & |t''| & |\chi\ T\ \text{-}\ t'| & = & |t'|
\end{array}$$

Figure 4: Erasure for annotated terms

$$\frac{T \rightsquigarrow^*_\beta T_1 \quad T' \rightsquigarrow^*_\beta T_2 \quad T_1 \cong^t T_2}{T \cong T'} \qquad \frac{T \cong^t T'}{T \cong T'}$$

$$\frac{T \cong^t T' \quad |t| =_{\beta\eta} |t'|}{T\ t \cong^t T\ t'} \qquad \frac{|t_1| =_{\beta\eta} |t'_1| \quad |t_2| =_{\beta\eta} |t'_2|}{\{t_1 \simeq t_2\} \cong^t \{t'_1 \simeq t'_2\}}$$

Figure 5: Non-congruence rules for conversion

Finally, we have modified the rules for equality types $\{t \simeq t'\}$ so that we require nothing of t and t' except that the set $dom(\Gamma)$ of variables declared by Γ includes their free variables $FV(t\ t')$. Further modifications over the version of CDLE in [12] are:

- To prove $\{t \simeq t\}$, one now writes $\beta\{t'\}$, with the critical idea that $|\beta\{t\}|$ erases to $|t|$. We call this the **Kleene trick**, because it goes back to Kleene's numeric realizability, which accepts any number n as a realizer of a true equation. Here, we accept any closed term t as a realizer of $\{t \simeq t\}$. This means that in Cedille, any such term – even otherwise untypable terms, non-normalizing terms, etc. – have type $\{t \simeq t\}$ for any term t.
- The ρ construct allows one to rewrite occurrences of t_1 to t_2 in the synthesized or checked type, where t_1 and t_2 are provably equal. In the Cedille implementation, we rewrite all matching occurrences. This may be compared to `rewrite` in Agda, except that it may be applied anywhere, not just as part of pattern matching [6].
- We adopt a strong form of Nuprl's **direct computation rules** [1]: If we have a term t' of type T and a proof t that $\{t' \simeq t''\}$, then we may conclude that t'' has type T by writing the annotated term $\phi\ t$ - $t'\{t''\}$, which erases to t''.
- Where the previous version of CDLE uses β-equivalence for (erased) terms, we here adopt $\beta\eta$-equivalence. This allows us to observe in many cases that retyping functions are actually $\beta\eta$-equivalent to $\lambda\, x.\, x$. While $\beta\eta$-equivalence takes more work to incorporate into intrinsic type theory [4], it raises no difficulties for our extrinsic one.
- In this version, we add an explicit axiom δ saying that Church-encoded boolean *true* is different from *false*. In the first version of CDLE, such an axiom was derivable from *lifting*, a construct allowing simply typable terms to be lifted to the type level [11]. We omit lifting in this new version of CDLE, because while sound, lifting as defined in that previous work is complicated and appears to be incomplete. Developing a new form of lifting remains to future work.

The equality type remains **intensional**: we equate terms iff they are $\beta\eta$-equal.

2.2 Semantics and metatheory

Figure 6 gives a realizability semantics for types and kinds, following the semantics given in the previous papers on CDLE [12, 11]. Details of this semantics are presented further in Section 2.3 below. Using the semantics and the definition in Figure 7 of $[\![\Gamma]\!]$, we can prove the following theorem:

Theorem 1 (Soundness)**.** *Suppose* $(\sigma, \rho) \in [\![\Gamma]\!]$*. Then we have:*

1. *If* $\Gamma \vdash \kappa$*, then* $[\![\kappa]\!]_{\sigma,\rho}$ *is defined.*
2. *If* $\Gamma \vdash T \Rightarrow \kappa$*, then* $[\![T]\!]_{\sigma,\rho} \in [\![\kappa]\!]_{\sigma,\rho}$*.*
3. *If* $\Gamma \vdash t \Rightarrow T$ *then* $[\sigma|t|]_{c\beta\eta} \in [\![T]\!]_{\sigma,\rho} \in \mathcal{R}$*.*
4. *If* $\Gamma \vdash t \Leftarrow T$ *and* $[\![T]\!]_{\sigma,\rho} \in \mathcal{R}$*, then* $[\sigma|t|]_{c\beta\eta} \in [\![T]\!]_{\sigma,\rho} \in \mathcal{R}$*.*

$$
\begin{array}{lcl}
[\![X]\!]_{\sigma,\rho} & = & \rho(X) \\
[\![\Pi x : T_1.T_2]\!]_{\sigma,\rho} & = & [\{\lambda x.t \mid \forall E \in [\![T_1]\!]_{\sigma,\rho}. \\
 & & \quad [[\zeta(E)/x]t]_{c\beta\eta} \in [\![T_2]\!]_{\sigma[x\mapsto\zeta(E)],\rho} \ \wedge \ t = |t|\}]_{c\beta\eta} \\
[\![\forall X : \kappa.T]\!]_{\sigma,\rho} & = & \cap\{[\![T]\!]_{\sigma,\rho[X\mapsto S]} |\ S \in [\![\kappa]\!]_{\sigma,\rho}\} \\
[\![\forall x : T.T']\!]_{\sigma,\rho} & = & \cap_\star\{[\![T']\!]_{\sigma[x\mapsto\zeta(E)],\rho} \mid E \in [\![T]\!]_{\sigma,\rho}\} \\
[\![\iota x : T.T']\!]_{\sigma,\rho} & = & \{E \in [\![T]\!]_{\sigma,\rho} |\ E \in [\![T']\!]_{\sigma[x\mapsto\zeta(E)],\rho}\} \\
[\![\lambda X : \kappa.T]\!]_{\sigma,\rho} & = & (S \in [\![\kappa]\!]_{\sigma,\rho} \mapsto [\![T]\!]_{\sigma,\rho[X\mapsto S]}) \\
[\![\lambda x : T.T']\!]_{\sigma,\rho} & = & (E \in [\![T]\!]_{\sigma,\rho} \mapsto [\![T']\!]_{\sigma[x\mapsto\zeta(E)],\rho}) \\
[\![T\ T']\!]_{\sigma,\rho} & = & [\![T]\!]_{\sigma,\rho}([\![T']\!]_{\sigma,\rho}) \\
[\![T\ t]\!]_{\sigma,\rho} & = & [\![T]\!]_{\sigma,\rho}([\sigma|t|]_{c\beta\eta}) \\
[\![t \simeq t']\!]_{\sigma,\rho} & = & [\{t'' \mid \sigma|t| =_{\beta\eta} \sigma|t'| \ \wedge \ t'' = |t''|\}]_{c\beta\eta} \\
 & & \quad \text{if } FV(t\ t') \subseteq dom(\sigma) \\
[\![\star]\!]_{\sigma,\rho} & = & \mathcal{R} \\
[\![\Pi x : T.\kappa]\!]_{\sigma,\rho} & = & (E \in [\![T]\!]_{\sigma,\rho} \to [\![\kappa]\!]_{\sigma[x\mapsto\zeta(E)],\rho}), \\
 & & \quad \text{if } [\![T]\!]_{\sigma,\rho} \in \mathcal{R} \\
[\![\Pi x : \kappa.\kappa']\!]_{\sigma,\rho} & = & (S \in [\![\kappa]\!]_{\sigma,\rho} \to [\![\kappa]\!]_{\sigma,\rho[X\mapsto S]}) \\
\cap_\star X & = & \begin{cases} \cap X, \text{ if } X \neq \emptyset \\ [\mathcal{L}]_{c\beta\eta}, \text{ otherwise} \end{cases}
\end{array}
$$

Figure 6: Semantics for types and kinds

5. *If $T \cong T'$ or $T \cong^t T'$ and $[\![T]\!]_{\sigma,\rho}$ and $[\![T']\!]_{\sigma,\rho}$ are both defined, then they are equal.*

An easy corollary, by the semantics of $\forall$-types, is then:
Theorem 2 (Logical consistency)**.** *There is no term t such that $\vdash t : \forall\, X{:}\star.\, X$.*

It may worry some readers that we have:
Observation 3. *There are typable terms t which fail to normalize.*

Defining `Top` to be $\{\lambda\, x.\, x \simeq \lambda\, x.\, x\}$, we may assign `Top` to any closed term `t`, including non-normalizing ones. In our annotated syntax, we write `{t}`. Even without this, the presence of δ in combination with ϕ allows us to type non-normalizing terms assuming an erased argument x of type $\{tt \simeq ff\}$ for Church-encoded booleans *tt* and *ff*. For example, $\delta\ x$ has type `{ x . x  x . x x}`, and with ϕ we can use this to type Ω by changing the typed term `id  True id`, where `True` is `X :  . X  X`. But failure of normalization does not impinge on Theorem 2. Extensional Martin-Löf type theory (MLTT) is also non-normalizing, for a very similar reason, but fact does not contradict its logical soundness [3]. In CDLE, the guarantees one gets about the behavior of terms are expressed almost entirely in their types. If the types are weak, then not much is guaranteed; but stronger types can guarantee properties like normalization.

Given the lack of normalization, several checks in the typing rules – for things like $t =_{\beta\eta} t'$ – are formally undecidable. In practice, we simply impose a bound on the number of steps of reduction, and thus restore formal decidability (we are checking "typable within a given budget"). In practice, the same is done for Coq and Agda, where type checking is decidable but, in general, infeasible (since one may write astronomically slow terminating functions).

Finally, in line with ideas recently advocated by Dreyer, we do not concern ourselves with syntactic type preservation [2], noting instead that by construction, semantic types $[\![T]\!]_{\sigma,\rho}$ are preserved by $\beta\eta$-reduction:
Theorem 4 (Semantic type preservation)**.** *If $t \rightsquigarrow_{\beta\eta} t'$ and $t \in [\![T]\!]_{\sigma,\rho}$, then $t' \in [\![T]\!]_{\sigma,\rho}$.*

Confluence of $\beta\eta$-reduction for (erased) terms is nothing other than confluence of untyped lambda calculus. This is because, as easily verified by inspecting Figure 4, the erasure function maps annotated terms t to terms $|t|$ of pure untyped lambda calculus.

$$
\begin{array}{lcl}
(\sigma \uplus [x \mapsto t], \rho) \in [\![\Gamma, x : T]\!] & \Leftrightarrow & (\sigma, \rho) \in [\![\Gamma]\!] \ \wedge \ [t]_{c\beta\eta} \in [\![T]\!]_{\sigma,\rho} \in \mathcal{R} \ \wedge \ t = |t| \\
(\sigma, \rho \uplus [X \mapsto S]) \in [\![\Gamma, X : \kappa]\!] & \Leftrightarrow & (\sigma, \rho) \in [\![\Gamma]\!] \ \wedge \ S \in [\![\kappa]\!]_{\sigma,\rho} \\
(\emptyset, \emptyset) \in [\![\cdot]\!] & &
\end{array}
$$

Figure 7: Semantics of typing contexts Γ

2.3 Some details about the semantics and the proof of Theorem 1

Following the development in [11], we work with set-theoretic partial functions for the semantics of higher-kinded types. Types are interpreted as $\beta\eta$-closed sets of closed terms. Let $\mathcal{L}$ be the set of closed terms of pure lambda calculus (differently from [11], we include all terms at this point, even non-normalizing ones). We write $=_{c\beta\eta}$ for standard $\beta\eta$-equivalence of pure lambda calculus, restricted to closed terms; and $[t]_{c\beta\eta}$ for $\{t' \mid t =_{c\beta\eta} t'\}$. This is extended to sets S of terms by writing $[S]_{c\beta\eta}$ for $\{[t]_{c\beta\eta} \mid t \in S\}$. In a few places we write $nf(t)$ for the (unique) $\beta\eta$-normal form of term t, if it has one. If (in our meta-language) we affirm a statement involving application of a partial function, then it is to be understood that that application is defined.

Definition 5 (Reducibility candidates)**.** $\mathcal{R} := \{[S]_{c\beta\eta} \mid S \subseteq \mathcal{L}\}$.

Throughout the development we find it convenient to use a **choice function** ζ. Given any nonempty set E of terms, ζ returns some element of E. Note that if $a \in A \in \mathcal{R}$, then a is a nonempty set of terms of pure lambda calculus; it can also happen that $A \in \mathcal{R}$ is empty. The proof of Theorem 1 (see appendix) is then a straightforward adaptation of [11].

Acknowledgments. This work was partially supported by the US NSF support under award 1524519, and US DoD support under award FA9550-16-1-0082 (MURI program).

References

[1] Robert L. Constable, Stuart F. Allen, Mark Bromley, Rance Cleaveland, J. F. Cremer, R. W. Harper, Douglas J. Howe, Todd B. Knoblock, N. P. Mendler, Prakash Panangaden, James T. Sasaki, and Scott F. Smith. *Implementing mathematics with the Nuprl proof development system.* Prentice Hall, 1986.

[2] Derek Dreyer. The Type Soundness Theorem That You Really Want to Prove (and Now You Can). Milner Award Lecture, delivered at Principles of Programming Languages (POPL), 2018.

[3] Peter Dybjer and Erik Palmgren. Intuitionistic Type Theory. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy.* Metaphysics Research Lab, Stanford University, winter 2016 edition, 2016.

[4] Herman Geuvers. The Church-Rosser Property for beta-eta-reduction in Typed lambda-Calculi. In *Proceedings of the Seventh Annual Symposium on Logic in Computer Science (LICS '92), Santa Cruz, California, USA, June 22-25, 1992*, pages 453–460. IEEE Computer Society, 1992.

[5] Alexei Kopylov. Dependent intersection: A new way of defining records in type theory. In *18th IEEE Symposium on Logic in Computer Science (LICS)*, pages 86–95, 2003.

[6] The Agda development team. *Agda*, 2018. Version 2.5.4.

[7] Alexandre Miquel. The Implicit Calculus of Constructions Extending Pure Type Systems with an Intersection Type Binder and Subtyping. In Samson Abramsky, editor, *Typed Lambda Calculi and Applications*, volume 2044 of *Lecture Notes in Computer Science*, pages 344–359. Springer, 2001.

[8] Nathan Mishra-Linger and Tim Sheard. Erasure and Polymorphism in Pure Type Systems. In Roberto M. Amadio, editor, *Foundations of Software Science and Computational Structures, 11th In-*

ternational Conference, FOSSACS 2008, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2008, Budapest, Hungary, March 29 - April 6, 2008. Proceedings, volume 4962 of *Lecture Notes in Computer Science*, pages 350–364. Springer, 2008.

[9] Simon Peyton Jones, Dimitrios Vytiniotis, Stephanie Weirich, and Mark Shields. Practical Type Inference for Arbitrary-rank Types. *J. Funct. Program.*, 17(1):1–82, January 2007.

[10] Benjamin C. Pierce and David N. Turner. Local type inference. *ACM Trans. Program. Lang. Syst.*, 22(1):1–44, 2000.

[11] Aaron Stump. The Calculus of Dependent Lambda Eliminations. *J. Funct. Program.*, 27:e14, 2017.

[12] Aaron Stump. From Realizability to Induction via Dependent Intersection, 2018. in press.

A Proof of Theorem 1

First a few lemmas (easy proofs omitted):
Lemma 6. *$[\![\kappa]\!]_{\sigma,\rho}$ is nonempty if defined.*
Lemma 7. *If E is nonempty, then $[\zeta(E)]_{c\beta\eta} = E$*
Lemma 8. *The set $\mathcal{R}$ ordered by subset forms a complete lattice, with greatest element $[\mathcal{L}]_{c\beta\eta}$ and greatest lower bound of a nonempty set of elements given by intersection. Also, $\emptyset$ is the least element.*
Lemma 9 (Term substitution and interpretation)**.** *If $t' =_{c\beta\eta} \sigma|t|$, then:*

- $[\![T]\!]_{\sigma[x\mapsto t'],\rho} = [\![[t/x]T]\!]_{\sigma,\rho}$
- $[\![\kappa]\!]_{\sigma[x\mapsto t'],\rho} = [\![[t/x]\kappa]\!]_{\sigma,\rho}$

Lemma 10 (Type substitution and interpretation)**.**

- $[\![T]\!]_{\sigma,\rho[X\mapsto[\![T']\!]_{\sigma,\rho}]} = [\![[T'/X]T]\!]_{\sigma,\rho}$
- $[\![\kappa]\!]_{\sigma,\rho[X\mapsto[\![T']\!]_{\sigma,\rho}]} = [\![[T'/X]\kappa]\!]_{\sigma,\rho}$

Lemma 11. *If $T \rightsquigarrow^*_\beta T'$ and $[\![T]\!]_{\sigma,\rho}$ is defined, then $[\![T']\!]_{\sigma,\rho}$ is also defined and equals $[\![T]\!]_{\sigma,\rho}$.*

Proof. This follows by induction on the reduction derivation, making use of the previous substitution lemmas. □

Soundness (Theorem 1). The following proof is adapted from [11]. It proceeds by mutual induction on the assumed typing, kinding, or superkinding derivation, for each part of the lemma. We prove the parts successively.

A.1 Proof of part (1)

Case:

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

$[\![\star]\!]_{\sigma,\rho}$ is just $\mathcal{R}$, which is defined.

Case:

$$\frac{\Gamma \vdash T \Rightarrow \star \quad \Gamma, x : T \vdash \kappa}{\Gamma \vdash \Pi\, x{:}T.\, \kappa}$$

By the IH, $[\![T]\!]_{\sigma,\rho} \in \mathcal{R}$, and so $[\![\Pi x : T.\, \kappa]\!]_{\sigma,\rho}$ is $(E \in [\![T]\!]_{\sigma,\rho} \to [\![\kappa]\!]_{\sigma[x\mapsto\zeta(E)],\rho})$. The latter quantity is defined if for all $E \in [\![T]\!]_{\sigma,\rho}$, $[\![\kappa]\!]_{\sigma[x\mapsto\zeta(E)],\rho})$ is, too. Since $[\![T]\!]_{\sigma,\rho} \in \mathcal{R}$, every element E of $[\![T]\!]_{\sigma,\rho}$ is nonempty, as noted above, so $\zeta(E)$ is defined. We may apply the IH to the second premise, since $(\sigma[x \mapsto \zeta(E)], \rho) \in [\![\Gamma, x : T]\!]$,

because $E \in [\![T]\!]_{\sigma,\rho}$ (by assumption) and $[\zeta(E)]_{c\beta\eta} = E$. This gives definedness of the semantics of the Π-kind.

Case:

$$\frac{\Gamma \vdash \kappa' \quad \Gamma, X : \kappa' \vdash \kappa}{\Gamma \vdash \Pi\, X : \kappa'.\, \kappa}$$

We must show ($S \in [\![\kappa]\!]_{\sigma,\rho} \to [\![\kappa]\!]_{\sigma,\rho[X \mapsto S]}$) is defined. This is true if $[\![\kappa]\!]_{\sigma,\rho}$ is defined, which is the case by the IH applied to the first premise; and if for all $S \in [\![\kappa]\!]_{\sigma,\rho}$, $[\![\kappa]\!]_{\sigma,\rho[X \mapsto S]}$ is defined. The latter is true by the IH applied to the second premise.

A.2 Proof of part (2)

Case:

$$\frac{(X : \kappa) \in \Gamma}{\Gamma \vdash X \Rightarrow \kappa}$$

From the definition of $[\![\Gamma]\!]$, we obtain $\rho(x) \in [\![\kappa]\!]_{\sigma,\rho}$.

Case:

$$\frac{\Gamma \vdash T \Rightarrow \star \quad \Gamma, x : T \vdash T' \Rightarrow \star}{\Gamma \vdash \Pi\, x : T.\, T' \Rightarrow \star}$$

We must show $[\![\Pi x : T.T']\!]_{\sigma,\rho} \in \mathcal{R}$. The semantics defines $[\![\Pi x : T.T']\!]_{\sigma,\rho}$ to be $[A]_{c\beta\eta}$ for a certain A, where if A is defined, then $A \subseteq \mathcal{L}$. So it suffices to shown definedness. By the IH for the first premise, $[\![T]\!]_{\sigma,\rho} \in \mathcal{R}$. This means that if $E \in [\![T]\!]_{\sigma,\rho}$, $\zeta(E)$ is defined. We can then apply the IH to the second premise, since $\sigma[x \mapsto \zeta(E)] \in [\![\Gamma, x : T]\!]$, to obtain definedness of $[\![T']\!]_{\sigma[x \mapsto \zeta(E),\rho}$.

Case:

$$\frac{\Gamma \vdash T \Rightarrow \star \quad \Gamma, x : T \vdash T' \Rightarrow \star}{\Gamma \vdash \forall\, x : T.\, T' \Rightarrow \star}$$

By the IH for the second premise, $[\![T_2]\!]_{\sigma[x \mapsto \zeta(E)],\rho} \in \mathcal{R}$, for every $E \in [\![T_1]\!]_{\sigma,\rho}$ where $[\![T_1]\!]_{\sigma,\rho} \in \mathcal{R}$. By the IH for the first premise, we indeed have $[\![T_1]\!]_{\sigma,\rho} \in \mathcal{R}$. So if $[\![T_1]\!]_{\sigma,\rho}$ is non-empty, then the intersection of all the sets $[\![T_2]\!]_{\sigma[x \mapsto \zeta(E)],\rho}$ where $E \in [\![T_1]\!]_{\sigma,\rho}$ is a reducibility candidate, since each of those sets is. By the semantics of $\forall$-types quantifying over terms, this is sufficient. If $[\![T_1]\!]_{\sigma,\rho}$ is empty, then the interpretation of the $\forall$-type is $[\mathcal{L}]_{c\beta\eta}$ by the definition of $\cap_\star$, and this is in $\mathcal{R}$.

Case:

$$\frac{\Gamma \vdash \kappa \quad \Gamma, X : \kappa \vdash T \Rightarrow \star}{\Gamma \vdash \forall\, X : \kappa.\, T \Rightarrow \star}$$

Similarly to the previous case: by the IH for the second premise, $[\![T_2]\!]_{\sigma,\rho[X \mapsto S} \in \mathcal{R}$, for every $S \in [\![\kappa]\!]_{\sigma,\rho}$. By the IH part for the first premise, $[\![\kappa]\!]_{\sigma,\rho}$ is defined. So the intersection of all the sets $[\![T_2]\!]_{\sigma,\rho[X \mapsto S]}$ where $S \in [\![\kappa]\!]_{\sigma,\rho}$ is a reducibility candidate, since each of those sets is. The intersection is nonempty, since $[\![\kappa]\!]_{\sigma,\rho}$ is (as stated in a lemma above). By the semantics of $\forall$-types quantifying over types, this is sufficient.

Case:

$$\frac{\Gamma \vdash T \Rightarrow \star \quad \Gamma, x : T \vdash T' \Rightarrow \star}{\Gamma \vdash \iota\, x : T.\, T' \Rightarrow \star}$$

The set $[\![\iota x : T.T']\!]_{\sigma,\rho}$ is explicitly defined to be a subset of $[\![T]\!]_{\sigma,\rho}$, which is in $\mathcal{R}$, by the IH applied to the first premise. Since for any $A \subseteq \mathcal{L}$, $[A]_{c\beta\eta}$ is in $\mathcal{R}$, to show that $[\![\iota x : T.T']\!]_{\sigma,\rho}$ is also in $\mathcal{R}$ it suffices to show

definedness of $[\![T']\!]_{\sigma[x\mapsto\zeta(E)],\rho}\}$ (which is used in the predicate picking out the particular subset of $[\![T]\!]_{\sigma,\rho}$), for $E \in [\![T]\!]_{\sigma,\rho}$. For such E, $\zeta(E)$ is defined (since $[\![T]\!]_{\sigma,\rho} \in \mathcal{R}$ and hence $E \in [\![T]\!]_{\sigma,\rho}$ is nonempty) and in E, so $\sigma[x \mapsto \zeta(E)] \in [\![\Gamma, x : T]\!]$. So by the IH for the second premise, $[\![T']\!]_{\sigma[x\mapsto\zeta(E),\rho]}$ is defined.

Case:

$$\frac{\Gamma \vdash T \Rightarrow \star \quad \Gamma, x : T \vdash T' \Rightarrow \kappa}{\Gamma \vdash \lambda\, x{:}T.\, T' \Rightarrow \Pi\, x{:}T.\, \kappa}$$

By the semantics, $[\![\lambda x : T.T']\!]_{\sigma,\rho}$ is $(E \in [\![T]\!]_{\sigma,\rho} \mapsto [\![T']\!]_{\sigma[x\mapsto\zeta(E)],\rho})$. We must show that this (meta-level) function is in $[\![\Pi x : T.\kappa]\!]_{\sigma,\rho}$. By the semantics of kinds, the latter quantity, if defined, is $(E \in [\![T]\!]_{\sigma,\rho} \to_{c\beta\eta} [\![\kappa]\!]_{\sigma[x\mapsto\zeta(E)],\rho})$. By the IH for the first premise, $[\![T]\!]_{\sigma,\rho} \in \mathcal{R}$. So we must just show that for any $E \in [\![T]\!]_{\sigma,\rho}$, $[\![T']\!]_{\sigma[x\mapsto\zeta(E)],\rho} \in [\![\kappa]\!]_{\sigma[x\mapsto\zeta(E)],\rho}$. But this follows by the IH for the second premise.

Case:

$$\frac{\Gamma \vdash \kappa \quad \Gamma, X : \kappa \vdash T' \Rightarrow \kappa'}{\Gamma \vdash \lambda\, X{:}\kappa.\, T' \Rightarrow \Pi\, X{:}\kappa.\, \kappa'}$$

This case is an easier version of the previous one. It suffices to assume an arbitrary $S \in [\![\kappa]\!]_{\sigma,\rho}$ and show $[\![T']\!]_{\sigma,\rho[X\mapsto S]} \in [\![\kappa']\!]_{\sigma,\rho[X\mapsto S]}$. But this follows by the IH applied to the second premise. And we have definedness of $[\![\kappa]\!]_{\sigma,\rho}$ by the IH for the first premise.

Case:

$$\frac{\Gamma \vdash T \Rightarrow \Pi\, x{:}T'.\, \kappa \quad \Gamma \vdash t \Leftarrow T'}{\Gamma \vdash T\ t \Rightarrow [t/x]\kappa}$$

By the IH for the first premise, $[\![T]\!]_{\sigma,\rho} \in [\![\Pi x : T'.\kappa]\!]_{\sigma,\rho}$. By the semantics of Π-kinds, this means that $[\![T]\!]_{\sigma,\rho}$ is a function which given any $E \in [\![T']\!]_{\sigma,\rho}$, will produce a result in $[\![\kappa]\!]_{\sigma[x\mapsto\zeta(E)],\rho}$. By the semantics of type applications, $[\![T\ t]\!]_{\sigma,\rho}$ is equal to $[\![T]\!]_{\sigma,\rho}([\sigma|t|]_{c\beta\eta})$. This is defined, since $[\sigma|t|]_{c\beta\eta} \in [\![T']\!]_{\sigma,\rho}$, by the IH for the second premise; note that $[\![T']\!]_{\sigma,\rho}$ is defined since otherwise $[\![\Pi x : T'.\kappa]\!]_{\sigma,\rho}$ would not be defined. The result of applying the function is thus indeed in $[\![[t/x]\kappa]\!]_{\sigma,\rho}$, since by Lemma 9, this equals $[\![\kappa]\!]_{\sigma[x\mapsto\zeta([\sigma|t|]_{c\beta\eta})],\rho}$ (the codomain of the function being applied).

Case:

$$\frac{\Gamma \vdash T \Rightarrow \Pi\, X{:}\kappa'.\, \kappa \quad \Gamma \vdash T' \Rightarrow \kappa' \quad \kappa \cong \kappa'}{\Gamma \vdash T \cdot T' \Rightarrow [T'/X]\kappa}$$

By the IH applied to the first premise, $[\![T]\!]_{\sigma,\rho} \in [\![\Pi\, X : \kappa'.\, \kappa]\!]_{\sigma,\rho}$. By the semantics of Π-kinds, this means that for any $S \in [\![\kappa']\!]_{\sigma,\rho}$, $[\![T]\!]_{\sigma,\rho}\ S$ is in $[\![\kappa]\!]_{\sigma,\rho[X\mapsto S]}$. By the IH for the second premise, we have $[\![T']\!][\![\kappa']\!]_{\sigma,\rho}$, and by the IH for the third premise, we have $[\![\kappa]\!]_{\sigma,\rho} = [\![\kappa']\!]_{\sigma,\rho}$. So we get $[\![T]\!]_{\sigma,\rho}([\![T']\!]_{\sigma,\rho}) \in [\![\kappa]\!]_{\sigma,\rho[X\mapsto[\![T']\!]_{\sigma,\rho}]}$, which suffices by Lemma 10.

Case:

$$\frac{FV(t\ t') \subseteq dom(\Gamma)}{\Gamma \vdash \{t \simeq t'\} : \star}$$

Either $\sigma|t| =_{c\beta\eta} \sigma|t'|$ or not. Either way, the interpretation is defined and in $\mathcal{R}$, since $FV(t\ t') \subseteq dom(\sigma)$ (as an easy consequence of $(\sigma, \rho) \in [\![\Gamma]\!]$).

A.3 Proof of parts (3) and (4)

Case:

$$\frac{(x : t) \in \Gamma}{\Gamma \vdash x \Rightarrow T}$$

This follows from the definition of $[\![\Gamma]\!]$.

Case:

$$\frac{\Gamma \vdash t \Leftarrow T \quad T' \rightsquigarrow^*_\beta T}{\Gamma \vdash t \Leftarrow T'}$$

We are assuming $[\![T']\!]_{\sigma,\rho}$ is defined, since this is a checking judgment. The desired result then follows from Lemma 11.

Case:

$$\frac{\Gamma \vdash t \Rightarrow T \quad T \rightsquigarrow^*_\beta T'}{\Gamma \vdash t \Rightarrow T'}$$

This also follows from Lemma 11 and the induction hypothesis for the first premise, which implies $[\![T]\!]_{\sigma,\rho} \in \mathcal{R}$ (and hence defined).

Case:

$$\frac{\Gamma \vdash t \Rightarrow T' \quad T' \cong T}{\Gamma \vdash t \Leftarrow T}$$

By the IH applied to the first premise, we have $[\sigma t]_{c\beta\eta} \in [\![T']\!]_{\sigma,\rho} \in \mathcal{R}$. By assumption, $[\![T]\!]_{\sigma,\rho} \in \mathcal{R}$, and so by the IH applied to the second premise, we have $[\sigma t]_{c\beta\eta} \in [\![T']\!]_{\sigma,\rho} = [\![T]\!]_{\sigma,\rho}$.

Case:

$$\frac{\Gamma, x : T \vdash t \Leftarrow T'}{\Gamma \vdash \lambda\, x.\, t \Leftarrow \Pi\, x{:}T.\, T'}$$

To show $[\sigma\lambda x.t]_{c\beta\eta} \in [\![\Pi x : T.T']\!]_{\sigma,\rho}$ (noting that the latter is defined and in $\mathcal{R}$ by assumption), it suffices to assume an arbitrary $E \in [\![T]\!]_{\sigma,\rho}$, and show $[[\zeta(E)/x]\sigma t]_{c\beta\eta} \in [\![T']\!]_{\sigma[x\mapsto\zeta(E)],\rho}$. By the IH, we have $[\sigma[x \mapsto \zeta(E)]t]_{c\beta\eta} \in [\![T']\!]_{\sigma[x\mapsto\zeta(E)],\rho}$. But $[\sigma[x \mapsto \zeta(E)]t]_{c\beta\eta} = [[\zeta(E)/x]\sigma t]_{c\beta\eta}$, so this is sufficient.

Case:

$$\frac{\Gamma \vdash t \Rightarrow \Pi\, x{:}T'.\, T \quad \Gamma \vdash t' \Leftarrow T'}{\Gamma \vdash t\ t' \Rightarrow [t'/x]T}$$

By the IH applied to the first premise, $[\sigma t]_{c\beta\eta} \in [\![\Pi x : T'.T]\!]_{\sigma,\rho} \in \mathcal{R}$. This means that there exists a λ-abstraction $\lambda x.\hat{t}$ such that $\lambda x.\hat{t} =_{c\beta\eta} \sigma t$, by the semantics of Π-types. Furthermore, for any $E \in [\![T']\!]_{\sigma,\rho}$, $[[\zeta(E)/x]\hat{t}]_{c\beta\eta} \in [\![T]\!]_{\sigma[x\mapsto\zeta(E)],\rho}$. By the IH applied to the second premise, $[\sigma t']_{c\beta\eta} \in [\![T']\!]_{\sigma,\rho}$, so we can instantiate the quantifier in the previous formula to obtain

$$[[\zeta([\sigma t']_{c\beta\eta})/x]\hat{t}]_{c\beta\eta} \in [\![T]\!]_{\sigma[x\mapsto\zeta([\sigma t']_{c\beta\eta})],\rho}$$

By Lemma 9, this is equivalent to

$$[[\zeta([\sigma t']_{c\beta\eta})/x]\hat{t}]_{c\beta\eta} \in [\![[t'/x]T_2]\!]_{\sigma,\rho}$$

Since $\sigma(t\ t') =_{c\beta\eta} (\lambda x.\hat{t})\ \sigma t' =_{c\beta\eta} [[\zeta([\sigma t']_{c\beta\eta})/x]\hat{t}$, this is sufficient.

Case:

$$\frac{\Gamma, X : \kappa \vdash t \Leftarrow T}{\Gamma \vdash \Lambda\, X.\, t \Leftarrow \forall\, X{:}\kappa.\, T}$$

By the IH, $[\sigma|t|]_{c\beta\eta} \in [\![T]\!]_{\sigma,\rho[X\mapsto S]}$, for all $S \in [\![\kappa]\!]_{\sigma,\rho}$. This is sufficient to prove $[\sigma|\Lambda\, X.\, t|]_{c\beta\eta} \in [\![\forall X : \kappa.T]\!]_{\sigma,\rho}$, by the semantics of $\forall$-types and definition of erasure.

Case:

$$\frac{\Gamma \vdash t \Rightarrow \forall\, X{:}\kappa.\, T \quad \Gamma \vdash T' \Leftarrow \kappa}{\Gamma \vdash t \cdot T' \Rightarrow [T'/X]T}$$

By the semantics of $\forall$-types and the IH applied to the first premise, we have $[\sigma|t|]_{c\beta\eta} \in [\![T]\!]_{\sigma,\rho[X\mapsto S]}$, for all $S \in [\![\kappa]\!]_{\sigma,\rho}$. Since $[\![T']\!]_{\sigma,\rho} \in [\![\kappa]\!]_{\sigma,\rho}$ by the IH applied to the second premise, we can derive $[\sigma t]_{c\beta\eta} \in [\![T]\!]_{\sigma,\rho[X\mapsto[\![T']\!]_{\sigma,\rho}]}$. By Lemma 10, this is equivalent to the required $[\sigma|t|]_{c\beta\eta} \in [\![[T'/X]T]\!]_{\sigma,\rho}$, using also the definition of erasure.

Case:

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

By the IH applied to the first premise, we have $[\sigma[x \mapsto \zeta(E)]t]_{c\beta\eta} \in [\![T']\!]_{\sigma[x\mapsto\zeta(E)],\rho}$, for any $E \in [\![T]\!]_{\sigma,\rho}$. This is because $[\![T]\!]_{\sigma,\rho} \in \mathcal{R}$, since $[\![\forall\, x{:}T'.\, T]\!]_{\sigma,\rho}$ is in $\mathcal{R}$ and hence defined, by assumption. Since $x \notin FV(t)$, we know $[[\sigma[x \mapsto \zeta(E)]t]_{c\beta\eta} = [\sigma t]_{c\beta\eta}$. By the semantics of $\forall$-types and definition of erasure, this suffices to show the desired conclusion.

Case:

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

The result follows easily by the IH applied to the premises, the semantics of $\forall$-types, definition of erasure, and Lemma 9.

Case:

$$\frac{\Gamma \vdash t \Leftarrow T \quad \Gamma \vdash t' \Leftarrow [t/x]T' \quad |t| =_{\beta\eta} |t'|}{\Gamma \vdash [t, t'] \Leftarrow \iota\, x{:}T.\, T'}$$

By the IH, we have $[\sigma|t|]_{c\beta\eta} \in [\![T]\!]_{\sigma,\rho}$ and $[\sigma|t|]_{c\beta\eta} \in [\![[t/x]T']\!]_{\sigma,\rho}$. By Lemma 9, the latter is equivalent to $[\sigma t]_{c\beta\eta} \in [\![T']\!]_{\sigma[x\mapsto\zeta([\sigma t]_{c\beta\eta}),\rho}$. These two facts about $[\sigma t]_{c\beta\eta}$ are sufficient, by the semantics of ι-types, for the desired conclusion, using also the fact (from the third premise) that $\sigma|t| =_{c\beta\eta} \sigma|t'|$.

Case:

$$\frac{\Gamma \vdash t \Rightarrow \iota\, x{:}T.\, T'}{\Gamma \vdash t.1 \Rightarrow T}$$

The desired conclusion follows easily from the IH and the semantics of ι-types.

Case:

$$\frac{\Gamma \vdash t \Rightarrow \iota\, x{:}T.\, T'}{\Gamma \vdash t.2 \Rightarrow [t.1/x]T'}$$

Similar to the previous case, using Lemma 9.

Case:

$$\frac{\Gamma \vdash FV(t) \subseteq dom(\Gamma)}{\Gamma \vdash \beta\{t'\} \Leftarrow \{t \simeq t\}}$$

$[\sigma|t'|]_{c\beta\eta} \in [\![\{t \simeq t\}]\!]_{\sigma,\rho}$ follows directly from the semantics of equality types.

Case:

$$\frac{\Gamma \vdash t \Rightarrow \{\lambda\, x.\, \lambda\, y.\, x \simeq \lambda\, x.\, \lambda\, y.\, y\}}{\Gamma \vdash \delta\ t \Leftarrow T}$$

By the semantics of equality types, $[\sigma|t'|]_{c\beta\eta}$ cannot be in the interpretation of the equation in the premise, since the two terms in question are closed and not $\beta\eta$-equal. By the IH applied to the first premise, however, $[\sigma|t'|]_{c\beta\eta}$ is in the interpretation of that equation. This is a contradiction.

Case:

$$\frac{\Gamma \vdash t' \Rightarrow t_1 \simeq t_2 \quad \Gamma \vdash t \Leftrightarrow [t_1/x]T}{\Gamma \vdash \rho\ t'\ \text{-}\ t \Leftrightarrow [t_2/x]T}$$

By the IH applied to the first premise, $\sigma|t_1| =_{\beta\eta} \sigma|t_2|$. The result then follows by the IH applied to the second premise, and Lemma 9.

Case:

$$\frac{\Gamma \vdash T \Leftarrow \star \quad \Gamma \vdash t \Leftarrow T \quad T \cong T'}{\Gamma \vdash \chi\ T\ \text{-}\ t \Leftarrow T'}$$

Using the IH for the first premise and the assumption that $[\![T']\!]_{\sigma,\rho}$ is in $\mathcal{R}$ and hence defined, we can apply the IH to the third premise to get $[\![T]\!]_{\sigma,\rho} = [\![T']\!]_{\sigma,\rho}$. Using this and the IH for second premise, we get the desired conclusion, using also the definition of erasure.

Case:

$$\frac{\Gamma \vdash T \Leftarrow \star \quad \Gamma \vdash t \Rightarrow T' \quad T \cong T'}{\Gamma \vdash \chi\ T\ \text{-}\ t \Rightarrow T}$$

By the IH applied to the second premise, we have $[\sigma|t|]_{c\beta\eta} \in [\![T']\!]_{\sigma,\rho} \in \mathcal{R}$. Using definedness of $[\![T']\!]_{\sigma,\rho}$ and the IH applied to the first premise, we can apply the IH to the third premise to get $[\![T]\!]_{\sigma,\rho} = [\![T']\!]_{\sigma,\rho}$, from which the desired conclusion follows by definition of erasure.

Case:

$$\frac{\Gamma \vdash t \Rightarrow \{t' \simeq t''\} \quad \Gamma \vdash t' \Leftrightarrow T}{\Gamma \vdash \phi\ t\ \text{-}\ t'\{t''\} \Leftrightarrow T}$$

By the IH for the first premise, $\sigma|t'| =_{c\beta\eta} \sigma|t''|$. By the IH for the second premise, $[\sigma|t'|]_{c\beta\eta} \in [\![T]\!]_{\sigma,\rho}$. This suffices for the desired conclusion, using also the definition of erasure ($|\phi\ t\ \text{-}\ t'\{t''\}| = |t''|$).

Proof of part (5)

Case:

$$\frac{T \rightsquigarrow^*_\beta T_1 \quad T' \rightsquigarrow^*_\beta T_2 \quad T_1 \cong^t T_2}{T \cong T'}$$

By Lemma 11, we have

$$\begin{array}{rcl} [\![T]\!]_{\sigma,\rho} & = & [\![T_1]\!]_{\sigma,\rho} \\ [\![T']\!]_{\sigma,\rho} & = & [\![T_2]\!]_{\sigma,\rho} \end{array}$$

By the IH for the third premise, we have $[\![T_1]\!]_{\sigma,\rho} = [\![T_2]\!]_{\sigma,\rho}$, which suffices.

Case:

$$\frac{T \cong^t T'}{T \cong T'}$$

By the IH.

Case:

$$\frac{T \cong^t T' \quad |t| =_{\beta\eta} |t'|}{T\ t \cong^t T'\ t'}$$

By the semantics, $[\![T\ t]\!]_{\sigma,\rho} = [\![T]\!]_{\sigma,\rho}([\sigma|t|]_{c\beta\eta})$. By the second premise and the IH for the first premise, this equals $[\![T']\!]_{\sigma,\rho}([\sigma|t'|]_{c\beta\eta})$, as required.

Case:

$$\frac{|t_1| =_{\beta\eta} |t_1'| \quad |t_2| =_{\beta\eta} |t_2'|}{\{t_1 \simeq t_2\} \cong^t \{t_1' \simeq t_2'\}}$$

This follows easily from the premises and the semantics of equality types.

□