On Parametric Polymorphism and Irrelevance in Martin-Löf Type Theory

Andreas Abel
Project PI.R2, INRIA Rocquencourt and PPS, Paris
andreas.abel@ifi.lmu.de

Abstract

We devise a typed equality judgement for a predicative version of Miquel's Implicit Calculus and complete it with a calculus for explicit substitutions. The resulting theory IITT, Implicit Intensional Type Theory*, is shown consistent by a partial equivalence model. We further present a bidirectional type checking and extraction algorithm and briefly sketch the integration of another notion of irrelevance, Awodey and Bauer's bracket types. This work is aimed at providing a solid an practical foundation for extraction of efficient programs from type theory.*

Keywords: Explicit Substitutions, Program Extraction, Implicit Quantification, PER Model, Typed Equality.

1 Introduction

Dependently typed programming languages such as Agda [18] and Coq [11] allow the programmer to express in one language programs, their types, rich invariants, and even proofs of these invariants. Besides code executed at runtime, dependently typed programs contain some to excessively much code needed only to pass the type checker, which is a the same time the verifier of the proofs woven into the program. After type checking, this static code needs to be discarded as much as possible to extract an efficient program.

To determine code and data irrelevant at runtime, some heuristics have been proposed, e.g., the erasure of index arguments in the constructors of inductive families [10]. The Coq program extractor [12] discards all proofs meaning all pieces of code whose type is a proposition, i. e., lives in universe `Prop`. In general, it is undecidable which piece of code is dead [9], and only the programmer himself knows. Therefore, I advocate means to explicitly state which part of the code is static (i. e., runtime irrelevant), and they should be more flexible than the universe-based approach. I will now introduce a few such means to the reader—the main body of this article is then mostly on advancing Miquel's approach.

1.1 Approaches to Irrelevance

Pfenning's logical framework with proof irrelevance [19] defines a judgement $\Gamma \vdash M \div A$, meaning that term M is a proof of type A, and proof irrelevance is realized by the rule:

$$\frac{\Gamma \vdash M, M' \div A}{\Gamma \vdash M = M' \div A}$$

Consequently, Pfenning allows proof hypotheses, with $\Gamma, x \div A \vdash M : B$ meaning that x is irrelevant both in term M and its type B, which is ensured by the *missing* typing rule $\Gamma, x \div A \vdash x : A$. Pfenning's system is a bit restrictive, it forbids proof hypotheses even in programs terms which do not have computational content anyway, e.g., in inhabitants of the empty type like $f : A \to 0, x \div A \vdash f\, x : 0$.

Awodey and Bauer [6] have a more semantic criterion of irrelevance. The introduce and eliminate proof types, called *bracket types* $[A]$ by the following rules.

$$\frac{\Gamma \vdash M : A}{\Gamma \vdash [M] : [A]}$$

$$\frac{\Gamma \vdash N : [A] \qquad \Gamma \vdash B \qquad \Gamma, x{:}A \vdash M : B \qquad \Gamma, x{:}A, y{:}A \vdash M = M[y/x] : B}{\Gamma \vdash M \;\mathsf{where}\; [x] \leftarrow N : B}$$

A proof N can be used in a term M if M does not depend on its value, expressed by the equality premise in the elimination rule. By this extensional perspective, the term $f : A \to 0, x : [A] \vdash f\, z \;\mathsf{where}\; [z] \leftarrow x : 0$ is well typed, since $f\, x = f\, y : 0$ for arbitrary x, y. However, the type B in the elimination rule may still not depend on the value x of proof N. This limitation is overcome by Miquel.

Miquel [16] has intersection types $\forall x : A.\, B$ similar to Pfenning's $\Pi x \div A.\, B$ and Awodey and Bauer's $\Pi x : [A].\, B$. However, they are much more powerful because B may now depend on x in a non-parametric way. For example, the constructor cons for vectors can be given type

$$\mathsf{cons} : \forall A{:}\mathsf{Type}.\, \forall n{:}\mathsf{Nat}.\, A \to \mathsf{Vec}\, A\, n \to \mathsf{Vec}\, A\, (n+1)$$

This means that the number n, denoting the length of the vector, is irrelevant for the construction of a vector at runtime, and can be erased during compilation. However, n is relevant for the type itself, it determines the shape of the vector type. Using proposition $m < n$, the type of a safe projection function for vectors can be expressed as

$$\begin{array}{rcl}\mathsf{lookup} & : & \forall A\!:\!\mathsf{Type}.\,\forall n\!:\!\mathsf{Nat}.\,\Pi m\!:\!\mathsf{Nat}.\,\forall p\!:\!m<n.\\ & & \Pi v\!:\!\mathsf{Vec}\,A\,n.\,A,\end{array}$$

marking arguments A, n, p as static, requiring only the computationally relevant m and v at runtime. Miquel's Implicit Calculus of Constructions (ICC) does not even store irrelevant terms, he uses a Curry-style representation with the typing rules:

$$\frac{\Gamma, x\!:\!A \vdash M : B}{\Gamma \vdash M : \forall x\!:\!A.\,B}\; x \notin \mathsf{FV}(M)$$

$$\frac{\Gamma \vdash M : \forall x\!:\!A.\,B \qquad \Gamma \vdash N : A}{\Gamma \vdash M : B[N/x]}$$

$$\frac{\Gamma \vdash M : A \qquad \Gamma \vdash B}{\Gamma \vdash M : B}\; A =_{\beta\eta} B$$

To accommodate impredicativity, countably many predicative universes, Curry-style, and contravariant subtyping, Miquel constructs an impressive model in coherence spaces, solving a recursive domain equation over a uncountable carrier, hosting countably many inaccessible cardinals.

*ICC**. Since typing is undecidable in Miquel's ICC, Barras and Bernardo [7] design a Church-style term language ICC*, which is justified by well-typed erasure into ICC. They show completeness for ICC without subtyping, meaning that every ICC term M has a decoration N in ICC* whose erasure N^* is equal to M. The core rules of ICC* are:

$$\frac{\Gamma, x\!:\!A \vdash M : B}{\Gamma \vdash \lambda[x\!:\!A].\,M : \forall x\!:\!A.\,B}\; x \notin \mathsf{FV}(M^*)$$

$$\frac{\Gamma \vdash M : \forall x\!:\!A.\,B \qquad \Gamma \vdash N : A}{\Gamma \vdash M[N] : B[N/x]}$$

$$\frac{\Gamma \vdash M : A \qquad \Gamma \vdash B}{\Gamma \vdash M : B}\; A^* =_{\beta\eta} B^*$$

The soundness of ICC* wrt. ICC is immediate by the formulation of the rules. However, the use of untyped conversion has well-known drawbacks: It cannot be extended to η for Σ types, since surjective pairing destroys confluence of untyped reduction. Extensional equality for the unit and empty type cannot even be stated in the absence of type information. It is therefore desirable to move to a typed conversion relation à la Pfenning.

1.2 This Work: IITT

Type conversion (aka judgmental equality) for ICC* faces the problem that types may depend on irrelevant terms. The crux is the congruence rule for application:

$$\frac{\Gamma \vdash M = M' : \forall x\!:\!A.\,B \qquad \Gamma \vdash N : A \qquad \Gamma \vdash N' : A}{\Gamma \vdash M[N] = M'[N'] : B[N/x]}$$

Since N and N' are unrelated, the rule produces an ill-typed right hand side $\Gamma \vdash M'[N'] : B[N/x]$ in ICC*. Consider for example the function $\mathsf{length} : \forall A : \mathsf{Type}.\,\forall n : \mathsf{Nat}.\,\mathsf{Vec}\,A\,n \to \mathsf{Nat}$, the rule allows us to derive

$$\mathsf{length}[\mathsf{String}][3] = \mathsf{length}[\mathsf{Nat}][5] : \mathsf{Vec}\ \mathsf{String}\ 3$$

which does not look sound. But is it? Since the arguments to length are erased and only serve to help type checking, this should still be fine. In fact, the congruence rule is sound by forcing type correctness of the seemingly ill-typed right hand side. This is realized by the unexpected typing rule

$$\frac{\Gamma \vdash M' : \forall x\!:\!A.\,B \qquad \Gamma \vdash N : A}{\Gamma \vdash M'[N'] : B[N/x]}.$$

This rule is obviously sound by erasure into ICC, but paradoxical since N' is in no connection to N, provides no help for the type checker, and may even be ill-typed. [1] In fact, this rule is as undecidable as Miquel's ICC, and it should not be considered in the specification of the well-typed terms we want to be able to type check. Yet it serves us in the specification of *well-typed* applications and paves the way for a typed equality for implicit quantification.

Contributions and overview. In the remainder of this paper, we present IITT, Implicit Intensional Type Theory, a predicative type theory á la Martin-Löf with implicit quantification.

In the next section, we present the typed equality in detail, and in Section 3 we add a non-standard explicit substitution calculus. Consistency of IITT is shown via a PER model in Section 4 before we proceed to bidirectional type checking and extraction in Section 5. In Section 6 we discuss the integration of bracket types into IITT.

2 Typed Equality for Implicit Quantification

IITT is a core dependently typed language with countably many predicative universes Set_k, three base types 0, 1,

[1] While I have never seen this rule in syntax—it is certainly a sound rule for Church-style System F—it is present in untyped reducibility semantics, e.g.

$$[\![\forall X B]\!] = \{M \mid \forall A, \mathcal{A}.\ M[A] \in [\![B]\!]_{X \mapsto \mathcal{A}}\}$$

with no connection between the syntactic type A and the reducibility candidate $\mathcal{A}$ (see for instance [8]).

and Nat with their introduction and elimination principles given by constants, and dependent ($\Pi^{\cdot}$) and polymorphic ($\Pi^{\div}$) function type. We represent bound variables by de Bruijn indices v_i ($i \in \mathbb{N}$).

$$
\begin{array}{lll}
\mathsf{Sort} & \ni s & ::= \mathsf{Set}_k \ \ (k \in \mathbb{N}) \\
\mathsf{Const} & \ni c & ::= s \mid 1 \mid \langle\rangle \mid 0 \mid \mathsf{abort}^2 \\
 & & \quad \mid \mathsf{Nat} \mid \mathsf{zero} \mid \mathsf{succ}^1 \mid \mathsf{natrec}^4 \\
\mathsf{Ann} & \ni \star & ::= \div \mid \cdot \\
\mathsf{Exp} & \ni t, u, T, U & ::= c \mid \Pi^\star U\, T \\
 & & \quad \mid \mathsf{v}_i \mid \lambda t \mid t\, u \\
\mathsf{Cxt} & \ni \Gamma, \Delta & ::= \diamond \mid \Gamma.^\star T
\end{array}
$$

The arity $\mathsf{ar}(c)$ of a constant c is 0 except where denoted otherwise by superscript, e.g., $\mathsf{ar}(\mathsf{abort}) = 2$.

The *irrelevance* annotation $^{\div}T$ shall mean that we consider the inhabitants of T irrelevant in the expression they are embedded. In general, we drop the *relevance* annotation $^{\cdot}T$, writing just T instead.

The notation $\Pi^\star U\, T$ subsumes the dependent function type $\Pi^{\cdot} U\, T$, short $\Pi\, U\, T$, and the polymorphic type $\Pi^{\div} U\, T$, also written $\forall\, U\, T$. The informal syntax $\Pi x\!:\!A.\, B$ is sugar for $\Pi\, A\, \lambda x B$ which de Bruijn-translates to $\Pi\, U\, \lambda T$ in our formal syntax. If T does not depend on v_0, we write $U \to T$ for $\Pi\, U\, \lambda T$.

At this point, we stress that despite the notational similarity of $\Pi^{\div} U\, T$ to Pfenning's proof quantification $\Pi x \div A.\, B$, it corresponds to Miquel's implicit quantification $\forall x\!:\!A.\, B$. This is witnessed by its formation rule (see below).

Resurrection $\Gamma^\oplus$ erases all markers $^{\div}(-)$ from Γ, making every variable relevant [19]. The generalization $\Gamma^\star$ shall mean $\Gamma^\oplus$ if $\star = \div$, and just Γ otherwise.

Judgements are the usual ones:

$\Gamma \vdash$	context Γ is well-formed
$\Gamma \vdash t : T$	expression t has type T
$\Gamma \vdash t = t' : T$	expressions t and t' are equal of type T

A crucial invariant of the last two judgements is $\Gamma^\oplus \vdash T : s$ for some sort. This means that in types T all variables are relevant and can be used, even those that are declared irrelevant and unusable for term t in the context Γ.

Rudimentary subtyping. In this work, we do not consider contravariant subtyping as Miquel does [15]. For subsumption, we only honor subsorting, $\mathsf{Set}_k \le \mathsf{Set}_l$ for $l \le k$, and type equality. Let $\Gamma \vdash T \le T'$ mean that there exists a sort s such that either $\Gamma \vdash T = T' : s$ or $T \equiv s_1$ and $T' \equiv s_2$ with $s_1 \le s_2 < s$.

Signature. Types for the constants are given by the judgement $\vdash_\Sigma c : T$ with the axioms below. Note that abort and natrec have a universe-polymorphic type, so it is really an axiom *scheme*.

$$
\begin{array}{ll}
\vdash_\Sigma \mathsf{Set}_k : \mathsf{Set}_{k+1} & \\
\vdash_\Sigma 1 : \mathsf{Set}_0 & \vdash_\Sigma \langle\rangle : 1 \\
\vdash_\Sigma 0 : \mathsf{Set}_0 & \vdash_\Sigma \mathsf{abort} : \forall\, \mathsf{Set}_k\, \lambda.\forall\, 0\, \lambda \mathsf{v}_1 \\
\vdash_\Sigma \mathsf{Nat} : \mathsf{Set}_0 & \vdash_\Sigma \mathsf{zero} : \mathsf{Nat} \\
\vdash_\Sigma \mathsf{succ} : \mathsf{Nat} \to \mathsf{Nat} & \vdash_\Sigma \mathsf{natrec} : \mathsf{Natrec}
\end{array}
$$

Herein, $\mathsf{Natrec} = \forall\, (\mathsf{Nat} \to \mathsf{Set}_k)\, \lambda.(\mathsf{v}_0\, \mathsf{zero}) \to (\Pi\, \mathsf{Nat}\, \lambda.(\mathsf{v}_2\, \mathsf{v}_0) \to \mathsf{v}_3\, (\mathsf{succ}\, \mathsf{v}_1)) \to \Pi\, \mathsf{Nat}\, \mathsf{v}_3$. In informal syntax, the type of abort is $\forall A : \mathsf{Set}_k.\, \forall\_ : 0.\, A$ and the one of natrec is $\forall C : \mathsf{Nat} \to \mathsf{Set}_k.\, C\, \mathsf{zero} \to (\Pi n : \mathsf{Nat}.\, C\, n \to C\, (\mathsf{succ}\, n)) \to \Pi n\!:\!\mathsf{Nat}.\, C\, n$.

Equations $\Delta \vdash_\Sigma t = t' : T$. The computational behavior of primitive recursion natrec is given by the following equations. Let herein $\Delta = C \div \mathsf{Nat} \to \mathsf{Set}_k\ .\ t_z : C\, \mathsf{zero}\ .\ t_s : \Pi n\!:\!\mathsf{Nat}.\, C\, n \to C\, (\mathsf{succ}\, n)$.

$$
\begin{array}{l}
\Delta \vdash_\Sigma \mathsf{natrec}_C\, t_z\, t_s\, \mathsf{zero} = t_z : C\, \mathsf{zero} \\
\Delta\ .\ n : \mathsf{Nat} \vdash_\Sigma \mathsf{natrec}_C\, t_z\, t_s\, (\mathsf{succ}\, n) = \\
\qquad\qquad t_s\, n\, (\mathsf{natrec}_C\, t_z\, t_s\, n) : C\, (\mathsf{succ}\, n)
\end{array}
$$

Context well-formedness. The judgement $\Gamma \vdash$ accepts only contexts that do not contain erased variable declarations $^{\div}T$. This does not mean that contexts with such declarations are not well-formed. It just means that this judgement is only invoked for resurrected contexts.

$$
\frac{}{\diamond \vdash} \qquad \frac{\Gamma \vdash \qquad \Gamma \vdash T : s}{\Gamma.T \vdash}
$$

Typing.

$$
\frac{\Gamma^\oplus \vdash \qquad \vdash_\Sigma c : T}{\Gamma \vdash c : T}
$$

$$
\frac{\Gamma \vdash U : s_1 \qquad \Gamma \vdash T : U \to s_2}{\Gamma \vdash \Pi^\star U\, T : s_3}\ s_1, s_2 \le s_3
$$

$$
\frac{\Gamma^\oplus \vdash \qquad \Gamma(i) = {}^{\cdot}T}{\Gamma \vdash \mathsf{v}_i : T \uparrow^{i+1}} \qquad \frac{\Gamma.^\star U \vdash t : T}{\Gamma \vdash \lambda t : \Pi^\star U\, \lambda T}
$$

$$
\frac{\Gamma \vdash t : \Pi^\star U\, T \qquad \Gamma^\star \vdash u : U}{\Gamma \vdash t\, u : T\, u}
$$

$$
\textsc{app2}\ \frac{\Gamma \vdash t : \forall\, U\, T \qquad \Gamma^\oplus \vdash u : U}{\Gamma \vdash t\, u' : T\, u}
$$

$$
\frac{\Gamma \vdash t : T \qquad \Gamma^\oplus \vdash T \le T'}{\Gamma \vdash t : T'}
$$

The typing rules are inspired by Pfenning [19], like him, we do not have a rule for $\Gamma.\div T \vdash \mathsf{v}_0 : T\uparrow$. However, there are subtle differences of great impact. Our calculus, as Miquel's [15], has for $\Gamma \vdash t : T$ the presupposition $\Gamma^\oplus \vdash T : s$ for some sort s, whereas Pfenning's has $\Gamma \vdash T : s$. This means he cannot type $\mathsf{cons} : \forall n : \mathsf{Nat}.\, A \to \mathsf{Vec}\, A\, n \to \mathsf{Vec}\, A\, (\mathsf{succ}\, n)$ since n is not irrelevant in $\mathsf{Vec}\, A\, n$. In our calculus, types are always irrelevant, which corresponds to the intuition that they are not present at run-time.

The second rule for implicit application APP2 destroys decidability of type checking, since u and u' are not connected, and u' might even be ill-typed. The rule is needed for internal soundness properties of the calculus (Theorem 2), it is omitted in the specification of type-checkable terms. Defining the judgement $\Gamma \vdash t \preccurlyeq u \star U$ by

$$\begin{array}{lcl} \Gamma \vdash t \preccurlyeq u : U & \iff & \Gamma \vdash u : U \text{ and } \Gamma \vdash t = u : U \\ \Gamma \vdash t \preccurlyeq u \div U & \iff & \Gamma^\oplus \vdash u : U \end{array}$$

we get the admissible rule of application

$$\frac{\Gamma \vdash t : \Pi^\star U\, T \qquad \Gamma \vdash u' \preccurlyeq u \star U}{\Gamma \vdash t\, u' : T\, u}.$$

Equality for expressions without substitutions is given by the following rules, plus reflexivity, symmetry, transitivity, and subsumption.

Computation. Substitution of u for index 0 in t is written $t\,\langle u\rangle$.

$$\frac{\Gamma.^\star U \vdash t : T \qquad \Gamma \vdash u' \preccurlyeq u \star U}{\Gamma \vdash (\lambda t)\, u' = t\,\langle u'\rangle : T\,\langle u\rangle} \qquad \frac{\Gamma \vdash_\Sigma t = t' : T}{\Gamma \vdash t = t' : T}$$

Extensionality.

$$\frac{\Gamma \vdash t : \Pi^\star U\, T}{\Gamma \vdash t = \lambda.\, (t\uparrow)\, \mathsf{v}_0 : \Pi^\star U\, T} \qquad \frac{\Gamma \vdash t, t' : T}{\Gamma \vdash t = t' : T}\ T \in \{0, 1\}$$

Compatibility.

$$\frac{\Gamma^\oplus \vdash \qquad \vdash_\Sigma c : T}{\Gamma \vdash c = c : T}$$

$$\frac{\Gamma \vdash U = U' : s_1 \qquad \Gamma \vdash T = T' : U \to s_2}{\Gamma \vdash \Pi^\star U\, T = \Pi^\star U'\, T' : s_3}\ s_1, s_2 \le s_3$$

$$\frac{\Gamma^\oplus \vdash T : s}{\Gamma.T \vdash \mathsf{v}_0 = \mathsf{v}_0 : T\uparrow} \qquad \frac{\Gamma.^\star U \vdash t = t' : T}{\Gamma \vdash \lambda t = \lambda t' : \Pi^\star U\, \lambda T}$$

$$\frac{\Gamma \vdash t = t' : \Pi^\star U\, T \qquad \Gamma \vdash u \preccurlyeq u' \star U}{\Gamma \vdash t\, u = t'\, u' : T\, u'}$$

The last rule is the crucial one. In case of a dependent function type, we get the instance

$$\frac{\Gamma \vdash t = t' : \Pi\, U\, T \qquad \Gamma \vdash u = u' : U}{\Gamma \vdash t\, u = t'\, u' : T\, u'}.$$

which is equivalent to the usual compatibility rule for application. In case of polymorphic type, we obtain

$$\frac{\Gamma \vdash t = t' : \forall\, U\, T \qquad \Gamma^\oplus \vdash u' : U}{\Gamma \vdash t\, u = t'\, u' : T\, u'}.$$

Using symmetry and transitivity, we can derive the fully general

$$\frac{\Gamma \vdash t_1 = t_2 : \forall\, U\, T \qquad \Gamma^\oplus \vdash u' : U}{\Gamma \vdash t_1\, u_1 = t_1\, u_2 : T\, u'}.$$

When is this rule needed in its full generality, meaning for terms u_1, u_2 which are arbitrary or at least not of the same type? Suppose we have a term $t : \forall n : \mathsf{Nat}.\, \forall v : \mathsf{Vec}\, n.\, C\, n\, v$. Then we should be able to derive $t\,(\mathsf{succ}\,(\mathsf{succ}\,\mathsf{zero}))\; bla' = t\,(\mathsf{succ}\,\mathsf{zero})\; bla : C\,\mathsf{zero}\,\mathsf{nil}$, since after erasure of implicit arguments we have just $t = t : C\,\mathsf{zero}\,\mathsf{nil}$ which is certainly an instance of the polymorphic typing of t. Hence, a more modest rule like

$$\frac{\Gamma \vdash t_1 = t_2 : \forall\, U\, T \qquad \Gamma^\oplus \vdash u_1, u_2, u' : U}{\Gamma \vdash t_1\, u_1 = t_1\, u_2 : T\, u'}$$

which avoids arbitrary untyped terms u_1, u_2 is not sufficient; the terms bla' and bla do not have a common type U.

3 A Calculus of Explicit Substitutions

In this section, we present a calculus of explicit substitutions [13, 1] for IITT. Explicit substitutions are important for implementing IITT, especially in the presence of meta variables [17]. On the theoretical side, explicit substitution calculi are incarnations of PER models (see next section).

Extension of syntax.

$$\begin{array}{lll} \mathsf{Exp} \ni t, u, T, U & ::= \cdots \mid t\,\sigma & \text{explicit substitution} \\ \mathsf{Subst} \ni \sigma, \tau & ::= \mathsf{id} \mid \sigma\,\tau & \text{identity and composition} \\ & \mid \uparrow \mid \langle \sigma, t\rangle & \text{lifting and extension} \end{array}$$

In the presence of explicit substitutions, we only need the 0th variable v_0. The ith variable is now represented by the i-fold application of the lifting substitution $\uparrow$ to the expression v_0. We still use the expression v_i, but now as a shorthand for the expression $\mathsf{v}_0\uparrow^i$. The notation $\langle t\rangle$ shall now abbreviate the singleton substitution $\langle \mathsf{id}, t\rangle$.

Judgements. Usually [13], the judgements for substitutions are $\Gamma \vdash \sigma : \Delta$ and $\Gamma \vdash \sigma = \sigma' : \Delta$, and then there is a rule like

$$\frac{\Gamma \vdash \sigma : \Delta \qquad \Delta \vdash t : T}{\Gamma \vdash t\,\sigma : T\,\sigma}.$$

However, with irrelevant assumptions that can nevertheless be used in types, this rule is unsound. We have $\vdash [\Omega/x] : (x \div \mathsf{Nat})$ and $x \div \mathsf{Nat} \vdash t\,x : T\,x$ if $t : \forall x\!:\!\mathsf{Nat}.\, T\,x$. After substitution, $\vdash t\,\Omega : T\,\Omega$, and the type $T\,\Omega$ is ill-formed. Inside types we need to substitute with only well-typed terms, yet inside terms we need to be able to instantiate irrelevant variables with arbitrary, possibly junk terms. This dilemma is solved by considering a rule

$$\frac{\Gamma \vdash \sigma \preccurlyeq \tau : \Delta \qquad \Delta \vdash t : T}{\Gamma \vdash t\,\sigma : T\,\tau}$$

with a judgement

$$\Gamma \vdash \sigma \preccurlyeq \tau : \Delta$$

which relates a term-side substitution σ to a type-side substitution τ. The substitutions must coincide at relevant variables, but might differ for irrelevant variables. This is illustrated by the substitution extension rule:

$$\frac{\Gamma \vdash \sigma \preccurlyeq \tau : \Delta \qquad \Delta^{\oplus} \vdash T : s \qquad \Gamma \vdash t \preccurlyeq u \star T\,\tau}{\Gamma \vdash \langle \sigma, t\rangle \preccurlyeq \langle \tau, u\rangle : \Delta.\!\star T}$$

In case $\star = :$ of a relevant variable the third assumption is $\Gamma \vdash t = u : T\,\tau$. The term t must be well-typed and equal to u. In the case of an irrelevant variable we only ask for $\Gamma^{\oplus} \vdash u : T\,\tau$. Only the value u substituted into types must be well-typed and may of course use even irrelevant variables. The value t substituted into terms can be arbitrary. In the example above we would have substitutions $\vdash [\Omega/x] \preccurlyeq [\mathsf{zero}/x] : (x \div \mathsf{Nat})$ and get the sound $t\,\Omega : T\,\mathsf{zero}$ after substitution.

Accordingly, we refine the substitution equality judgement to

$$\Gamma \vdash \sigma = \sigma' \preccurlyeq \tau : \Delta.$$

The rules of the substitution calculus are given in Fig. 1. Once the need for a separate type-side substitution τ has been understood, the rules are standard. Only conversion is new and deserves some explanation:

$$\frac{\Gamma \vdash \sigma \preccurlyeq \tau : \Delta \qquad \Gamma^{\oplus} \vdash \tau = \tau' \preccurlyeq \tau' : \Delta^{\oplus}}{\Gamma \vdash \sigma \preccurlyeq \tau' : \Delta}$$

A type-side substitution τ always maps a completely relevant context $\Delta^{\oplus}$ to a completely relevant context $\Gamma^{\oplus}$. Hence, even if τ and τ' appear on the term-side in the second hypothesis, because of resurrection they are equal type-side substitutions.

3.1 Internal Soundness

We write $\Gamma \vdash J$ for any judgement where J stands for the (possibly empty) statement right of the turnstile $\vdash$. We also use the forms $\Gamma \vdash L : R$ and $\Gamma \vdash L \preccurlyeq M : R$ to denote a set of judgements, where L could be just an expression or substitution, or an equation between expressions or substitutions.

Lemma 1 (Resurrection as weakening)**.**

1. *If* $\Gamma \vdash J$ *then* $\Gamma^{\oplus} \vdash J$.
2. *If* $\Gamma \vdash L : \Delta^{\oplus}$ *then* $\Gamma \vdash L : \Delta$.

Theorem 2 (Syntactic Validity)**.**

1. *If* $\Gamma.\Gamma' \vdash J$ *then* $\Gamma^{\oplus} \vdash$.
2. *If* $\Gamma \vdash_{(\Sigma)} L : T$ *then* $\Gamma^{\oplus} \vdash T : s$ *for some sort* s.
3. *If* $\Gamma \vdash_{(\Sigma)} t = t' : T$ *then* $\Gamma \vdash t : T$ *and* $\Gamma \vdash t' : T$.
4. *If* $\Gamma \vdash L \preccurlyeq \tau : \Delta$ *then* $\Delta^{\oplus} \vdash$ *and* $\Gamma^{\oplus} \vdash \tau \preccurlyeq \tau : \Delta^{\oplus}$.
5. *If* $\Gamma \vdash \sigma = \sigma' \preccurlyeq \tau : \Delta$ *then* $\Gamma \vdash \sigma \preccurlyeq \tau : \Delta$ *and* $\Gamma \vdash \sigma' \preccurlyeq \tau : \Delta$.

Rule APP2 is needed to show proposition 3 for the application congruence rule.

4 Semantics

In this section, we give a PER model for IITT and use an instance involving only closed terms to show consistency of the calculus. Normalization could be shown by another instance allowing open terms, however, this is beyond the scope of this paper. Since we are defining a PER model for typed equality, we do not need extensionality of the underlying λ-model, which in our case are closures and weak head normal forms. Extensionality is handled at the level of PERs. This is in contrast to Miquel's [14] semantics for ICC with untyped equality where the interpretation of λ-terms must be semi-extensional. Our semantics, in contrast to Miquel's impressive piece of art, is relievingly simple, mostly due to the absence of impredicativity.

4.1 Evaluation

In the following, we present a call-by-name semantics for terms and substitutions. A term t in an environment η constitutes a closure $t\,\eta \in \mathsf{Clos}$, where an environment $\eta \in \mathsf{Env}$ is a list of closures. Weak head evaluation $\alpha \searrow a$ takes a closure $\alpha \in \mathsf{Clos}$ to a value $a \in \mathsf{Val}$. Values $\mathsf{Val} \subseteq \mathsf{Clos} \subseteq \mathsf{Exp}$ are expressions in weak head normal form. Evaluation of a function λt in an environment η results in a function closure $(\lambda t)\,\eta$.

$$\begin{array}{lll} \mathsf{Val} \ni a, b, f, A, B, F & ::= \Pi\,A\,F \mid \forall\,A\,F \mid (\lambda t)\,\eta \mid c \mid \mathsf{succ}\,\alpha & \\ & \quad \mid c\,\vec{\alpha}\ \ (|\vec{\alpha}| < \mathsf{ar}(c)) & \\ \mathsf{Clos} \ni \alpha, \beta & ::= t\,\eta \mid a \mid \mathsf{natrec}_{\beta}\,\alpha_z\,\alpha_s\,\alpha_n & \\ \mathsf{Env} \ni \eta, \rho & ::= \mathsf{id} \mid \langle \eta, \alpha\rangle & \end{array}$$

$\boxed{\Gamma \vdash \sigma \preccurlyeq \tau}$

$$\frac{\Gamma \vdash \sigma \preccurlyeq \tau : \Delta \qquad \Delta \vdash t : T}{\Gamma \vdash t\,\sigma : T\,\tau} \qquad \frac{\Gamma \vdash \sigma \preccurlyeq \tau : \Delta \qquad \Delta^{\oplus} \vdash T : s \qquad \Gamma \vdash t \preccurlyeq u \star T\,\tau}{\Gamma \vdash \langle \sigma, t\rangle \preccurlyeq \langle \tau, u\rangle : \Delta.^{\star}T}$$

$$\frac{\Gamma^{\oplus} \vdash}{\Gamma \vdash \mathsf{id} \preccurlyeq \mathsf{id} : \Gamma} \qquad \frac{\Gamma_1 \vdash \sigma' \preccurlyeq \tau' : \Gamma_2 \qquad \Gamma_2 \vdash \sigma \preccurlyeq \tau : \Gamma_3}{\Gamma_1 \vdash \sigma\,\sigma' \preccurlyeq \tau\,\tau' : \Gamma_3} \qquad \frac{\Gamma^{\oplus} \vdash T : s}{\Gamma.^{\star}T \vdash \uparrow \preccurlyeq \uparrow : \Gamma}$$

$$\frac{\Gamma \vdash \sigma \preccurlyeq \tau : \Delta \qquad \Gamma^{\oplus} \vdash \tau = \tau' \preccurlyeq \tau' : \Delta^{\oplus}}{\Gamma \vdash \sigma \preccurlyeq \tau' : \Delta}$$

$\boxed{\Gamma \vdash \sigma = \sigma' \preccurlyeq \tau : \Delta}$

$$\frac{\Gamma \vdash \sigma \preccurlyeq \tau : \Delta}{\Gamma \vdash \mathsf{id}\,\sigma = \sigma \preccurlyeq \tau : \Delta} \qquad \frac{\Gamma \vdash \sigma \preccurlyeq \tau : \Delta}{\Gamma \vdash \sigma\,\mathsf{id} = \sigma \preccurlyeq \tau : \Delta} \qquad \frac{\Gamma_i \vdash \sigma_i \preccurlyeq \tau_i : \Gamma_{i+1} \quad \text{for } i = 1..3}{\Gamma_1 \vdash (\sigma_3\,\sigma_2)\,\sigma_1 = \sigma_3\,(\sigma_2\,\sigma_1) \preccurlyeq \tau_3\,(\tau_2\,\tau_1) : \Gamma_4}$$

$$\frac{\Gamma \vdash \sigma \preccurlyeq \tau : \Delta \qquad \Delta^{\oplus} \vdash T : s \qquad \Gamma^{\oplus} \vdash u \star T\,\tau}{\Gamma \vdash \uparrow \langle \sigma, t\rangle = \sigma \preccurlyeq \tau : \Delta}$$

$$\frac{\Gamma' \vdash \sigma' \preccurlyeq \tau' : \Gamma \qquad \Gamma \vdash \sigma \preccurlyeq \tau : \Delta \qquad \Delta^{\oplus} \vdash T : s \qquad \Gamma \vdash u \preccurlyeq t \star T\,\tau}{\Gamma' \vdash \langle \sigma, u\rangle\,\sigma' = \langle \sigma\,\sigma', u\,\sigma'\rangle \preccurlyeq \langle \tau\,\tau', t\,\tau'\rangle : \Delta.^{\star}T}$$

$$\frac{\Gamma^{\oplus} \vdash T : s}{\Gamma.^{\star}T \vdash \mathsf{id} = \langle \uparrow, \mathsf{v}_0\rangle \preccurlyeq \mathsf{id} : \Gamma.^{\star}T}$$

$$\frac{\Gamma^{\oplus} \vdash}{\Gamma \vdash \mathsf{id} = \mathsf{id} \preccurlyeq \mathsf{id} : \Gamma} \qquad \frac{\Gamma_1 \vdash \sigma_1' = \sigma_2' \preccurlyeq \tau' : \Gamma_2 \qquad \Gamma_2 \vdash \sigma_1 = \sigma_2 \preccurlyeq \tau : \Gamma_3}{\Gamma_1 \vdash \sigma_1\,\sigma_1' = \sigma_2\,\sigma_2' \preccurlyeq \tau\,\tau' : \Gamma_3}$$

$$\frac{\Gamma^{\oplus} \vdash T : s}{\Gamma.^{\star}T \vdash \uparrow = \uparrow \preccurlyeq \uparrow : \Gamma} \qquad \frac{\Gamma \vdash \sigma = \sigma' \preccurlyeq \tau : \Delta \qquad \Delta^{\oplus} \vdash T : s \qquad \Gamma \vdash t \preccurlyeq t' \star T\,\tau}{\Gamma \vdash \langle \sigma, t\rangle = \langle \sigma', t'\rangle \preccurlyeq \langle \tau, t'\rangle : \Delta.^{\star}T}$$

$$\frac{\Gamma \vdash \sigma \preccurlyeq \tau : \Delta}{\Gamma \vdash \sigma = \sigma \preccurlyeq \tau : \Delta} \qquad \frac{\Gamma \vdash \sigma = \sigma' \preccurlyeq \tau : \Delta}{\Gamma \vdash \sigma' = \sigma \preccurlyeq \tau : \Delta} \qquad \frac{\Gamma \vdash \sigma_1 = \sigma_2 \preccurlyeq \tau : \Delta \qquad \Gamma \vdash \sigma_2 = \sigma_3 \preccurlyeq \tau : \Delta}{\Gamma \vdash \sigma_1 = \sigma_3 \preccurlyeq \tau : \Delta}$$

$$\frac{\Gamma \vdash \sigma = \sigma' \preccurlyeq \tau : \Delta \qquad \Gamma^{\oplus} \vdash \tau = \tau' \preccurlyeq \tau' : \Delta^{\oplus}}{\Gamma \vdash \sigma = \sigma' \preccurlyeq \tau' : \Delta}$$

$\boxed{\Gamma \vdash t = t' : T}$

$$\frac{\Gamma \vdash \sigma = \sigma' \preccurlyeq \tau : \Delta \qquad \Delta \vdash t = t' : T}{\Gamma \vdash t\,\sigma = t'\,\sigma' : T\,\tau}$$

$$\frac{\Gamma \vdash \sigma \preccurlyeq \tau : \Delta \qquad \Delta^{\oplus} \vdash T : s \qquad \Gamma \vdash t : T\,\tau}{\Gamma \vdash \mathsf{v}_0\,\langle \sigma, t\rangle = t : T\,\tau} \qquad \frac{\Gamma \vdash \sigma \preccurlyeq \tau : \Delta \qquad \vdash_{\Sigma} c : T}{\Gamma \vdash c\,\sigma = c : T} \qquad \frac{\Gamma \vdash t : T}{\Gamma \vdash t\,\mathsf{id} = t : T}$$

$$\frac{\Gamma_1 \vdash \sigma' \preccurlyeq \tau' : \Gamma_2 \qquad \Gamma_2 \vdash \sigma \preccurlyeq \tau : \Gamma_3 \qquad \Gamma_3 \vdash t : T}{\Gamma_1 \vdash (t\,\sigma)\,\sigma' = t\,(\sigma\,\sigma') : T\,(\tau\,\tau')} \qquad \frac{\Gamma \vdash \sigma \preccurlyeq \tau : \Delta \qquad \Delta \vdash U : s_1 \qquad \Delta \vdash T : U \to s_2}{\Gamma \vdash (\Pi^{\star} U\,T)\,\sigma = \Pi^{\star}(U\,\sigma)\,(T\,\sigma) : s_3}\ s_1, s_2 \le s_3$$

$$\frac{\Gamma \vdash \sigma \preccurlyeq \tau : \Delta \qquad \Delta.^{\star}U \vdash t : T}{\Gamma \vdash (\lambda t)\,\sigma = \lambda.\, t\,\langle \sigma\uparrow, \mathsf{v}_0\rangle : \Pi^{\star}(U\,\tau)\,(\lambda.\, T\,\langle \tau\uparrow, \mathsf{v}_0\rangle)} \qquad \frac{\Gamma \vdash \sigma \preccurlyeq \tau : \Delta \qquad \Delta \vdash t : \Pi^{\star} U\,T \qquad \Delta \vdash u \preccurlyeq u' \star U}{\Gamma \vdash (t\,u)\,\sigma = (t\,\sigma)\,(u\,\sigma) : (T\,\tau)\,(u'\,\tau)}$$

Figure 1. Explicit substitution calculus.

Evaluation of substitutions. We define the total function $(\!|\sigma|\!)_\eta = \eta'$ by induction on σ.

$$\begin{array}{lcl} (\!|\mathsf{id}|\!)_\eta & = & \eta \\ (\!|\sigma\,\tau|\!)_\eta & = & (\!|\sigma|\!)_{(\!|\tau|\!)_\eta} \end{array} \qquad \begin{array}{lcl} (\!|\langle\sigma, t\rangle|\!)_\eta & = & \langle (\!|\sigma|\!)_\eta, t\,\eta\rangle \\ (\!|\uparrow|\!)_{\langle\eta,\alpha\rangle} & = & \eta \\ (\!|\uparrow|\!)_{\mathsf{id}} & = & \mathsf{id} \end{array}$$

The last equation is only there to make the definition total, it will never be invoked for well-scoped substitutions.

We introduce the judgements $\alpha \searrow a$ "closure α evaluates to a" and $f \mathbin{@} \alpha \searrow b$ "value f applied to closure α evaluates to b" inductively by the following rules.

$$\frac{}{a \searrow a} \qquad \frac{}{c\,\eta \searrow c} \qquad \frac{U\,\eta \searrow A \qquad T\,\eta \searrow F}{(\Pi^\star U\,T)\,\eta \searrow \Pi^\star A\,F}$$

$$\frac{\alpha \searrow a}{\mathsf{v}_0\,\langle\eta,\alpha\rangle \searrow a} \qquad \frac{}{(\lambda t)\,\eta \searrow (\lambda t)\,\eta}$$

$$\frac{t\,\eta \searrow f \qquad f \mathbin{@} (u\,\eta) \searrow b}{(t\,u)\,\eta \searrow b} \qquad \frac{t\,(\!|\sigma|\!)_\eta \searrow a}{(t\,\sigma)\,\eta \searrow a}$$

$$\frac{t\,\langle\eta,\alpha\rangle \searrow b}{(\lambda t)\,\eta \mathbin{@} \alpha \searrow b} \qquad \frac{}{c\,\vec{\alpha} \mathbin{@} \alpha \searrow c\,\vec{\alpha}\,\alpha}$$

$$\frac{\mathsf{natrec}_\beta\,\alpha_z\,\alpha_s\,\alpha_n \searrow b}{\mathsf{natrec}_\beta\,\alpha_z\,\alpha_s \mathbin{@} \alpha_n \searrow b} \qquad \frac{\alpha_n \searrow \mathsf{zero} \qquad \alpha_z \searrow b}{\mathsf{natrec}_\beta\,\alpha_z\,\alpha_s\,\alpha_n \searrow b}$$

$$\frac{\alpha_n \searrow \mathsf{succ}\,\alpha \qquad \alpha_s \searrow f_s \qquad f_s \mathbin{@} \alpha \searrow f \qquad f \mathbin{@} (\mathsf{natrec}_\beta\,\alpha_z\,f_s\,\alpha) \searrow b}{\mathsf{natrec}_\beta\,\alpha_z\,\alpha_s\,\alpha_n \searrow b}$$

Evaluation is deterministic, thus, in case $f \mathbin{@} \alpha \searrow b$, we write $f \cdot \alpha$ for (the unique) b. Also, we write $(\!|t|\!)_\eta$ for b in case $t\,\eta \searrow b$.

While we have now given a quite concrete λ-model, we can present it more abstractly as a partial applicative structure. Although we do not go into detail, we like to stress that the formulation of PER models in the next section relies only on the abstract properties.

4.2 PER Models

In this section, we axiomatize PER models over the partial syntactical applicative structure given in the last section and prove the soundness of IITT in all PER models.

We denote by $\mathcal{A}, \mathcal{B} \in \mathsf{Per}$ *partial equivalence relations* (PERs) over the set of values Val, thus, $\mathcal{A}$ is a subset of Val with an equivalence relation written $a = a' \in \mathcal{A}$. A *family* $\mathcal{F} \in \mathcal{A} \to S$ respects the equivalence, i. e., $\mathcal{F}(a) = \mathcal{F}(a')$ for all $a = a' \in \mathcal{A}$.

We write $f \cdot \alpha = f' \cdot \alpha' \in \mathcal{B}$ if there are $b = b' \in \mathcal{B}$ such that $f \mathbin{@} \alpha \searrow b$ and $f' \mathbin{@} \alpha' \searrow b'$. Given $\mathcal{A} \in \mathsf{Per}$ we define the closure extension $\overline{\mathcal{A}}$ and the PER $[\mathcal{A}]$ ("smash $\mathcal{A}$"), and the PERs $\Pi\,\mathcal{A}\,\mathcal{F}$ and $\forall\,\mathcal{A}\,\mathcal{F}$ for $\mathcal{F} \in \overline{\mathcal{A}} \to \mathsf{Per}$, by

$$\begin{array}{lcl} \alpha = \alpha' \in \overline{\mathcal{A}} & \iff & \alpha \searrow a \text{ and } \alpha' \searrow a' \\ & & \quad \text{and } a = a' \in \mathcal{A} \\ a = a' \in [\mathcal{A}] & \iff & a \in \mathcal{A} \text{ and } a' \in \mathcal{A} \\ f = f' \in \Pi\,\mathcal{A}\,\mathcal{F} & \iff & f \cdot \alpha = f' \cdot \alpha' \in \mathcal{F}(\alpha) \\ & & \quad \text{for all } \alpha = \alpha' \in \overline{\mathcal{A}} \\ f = f' \in \forall\,\mathcal{A}\,\mathcal{F} & \iff & f \cdot \beta = f' \cdot \beta' \in \mathcal{F}(\alpha) \\ & & \quad \text{for all } \alpha \in \overline{\mathcal{A}} \text{ and } \beta, \beta' \in \mathsf{Clos} \end{array} .$$

Smashing $\mathcal{A}$ trivializes the equality structure on $\mathcal{A}$ but preserves the carrier.

Let us write $\Pi\alpha \in \mathcal{A}.\ \mathcal{F}(\alpha)$ for $\Pi\,\mathcal{A}\,(\alpha \mapsto \mathcal{F}(\alpha))$ and $\forall\alpha \in \mathcal{A}.\ \mathcal{F}(\alpha)$ for $\forall\,\mathcal{A}\,(\alpha \mapsto \mathcal{F}(\alpha))$. We abbreviate $\Pi\_ \in \mathcal{A}.\ \mathcal{B}$ further to $\mathcal{A} \to \mathcal{B}$.

Universes. A pair $(\mathcal{U} \in \mathsf{Per}, \mathcal{El} \in \mathcal{U} \to \mathsf{Per})$ is called a *universe* if it is closed under dependent and polymorphic function space, i. e., if $A = A' \in \mathcal{U}$ and $F \cdot \alpha = F' \cdot \alpha' \in \mathcal{U}$ for all $\alpha = \alpha' \in \overline{\mathcal{El}(A)}$, then

1. $\Pi\,A\,F = \Pi\,A'\,F' \in \mathcal{U}$ with $\mathcal{El}(\Pi\,A\,F) = \Pi\alpha \in \mathcal{El}(A).\ \mathcal{El}(F \cdot \alpha)$ and
2. $\forall\,A\,F = \forall\,A'\,F' \in \mathcal{U}$ with $\mathcal{El}(\forall\,A\,F) = \forall\alpha \in \mathcal{El}(A).\ \mathcal{El}(F \cdot \alpha)$.

Note that the "formation rules" for Π and $\forall$ coincide, i. e., membership of $\Pi\,A\,F$ and $\forall\,A\,F$ in $\mathcal{U}$ depends on the same conditions. In particular, F is not required to be constant in the formation of $\forall\,A\,F$, as it would be in $\Pi\,[A]\,F$. Thus, polymorphism gives us more than bracket types.

A sequence $(\mathcal{S}et_k, \mathcal{El}_k)$ is a *cumulative hierarchy of universes à la Russell* if for all $k \in \mathbb{N}$

1. $\mathcal{S}et_k \subseteq \mathcal{S}et_{k+1}$ and $\mathcal{El}_k(A) = \mathcal{El}_{k+1}(A)$ for all $A \in \mathcal{S}et_k$, and
2. $\mathsf{Set}_k \in \mathcal{S}et_{k+1}$ and $\mathcal{El}_{k+1}(\mathsf{Set}_k) = \mathcal{S}et_k$.

The first condition guarantees the existence of a supremum $(\mathcal{S}et_\omega, \mathcal{El}_\omega)$ with $\mathcal{S}et_k \subseteq \mathcal{S}et_\omega$ and $\mathcal{El}_k(A) = \mathcal{El}_\omega(A)$ for all $k \in \mathbb{N}$ and $A \in \mathcal{S}et_k$. We may simply write $\mathcal{El}$ for $\mathcal{El}_\omega$.

Base types for a universe $(\mathcal{U}, \mathcal{El})$. A PER $\mathcal{U}nit$ is called a *unit type* if $\mathcal{U}nit = [\mathcal{U}nit]$ and $\langle\rangle \in \mathcal{U}nit$. A PER $\mathcal{E}mpty$ is called an *empty type* if $\mathcal{E}mpty = [\mathcal{E}mpty]$ and $\mathsf{abort} \in \forall A \in \mathcal{U}.\ \mathcal{E}mpty \to \mathcal{El}(A)$. A PER $\mathcal{N}at$ is called a *natural number type* if $\mathsf{zero} \in \mathcal{N}at$, $\mathsf{succ} \in \mathcal{N}at \to \mathcal{N}at$, and $\mathsf{natrec} \in \forall F \in \mathcal{N}at \to \mathcal{U}.\ \mathcal{El}(F \cdot \mathsf{zero}) \to (\Pi a \in \mathcal{N}at.\ \mathcal{El}(F \cdot a) \to \mathcal{El}(F \cdot (\mathsf{succ}\,a))) \to \Pi a \in \mathcal{N}at.\ \mathcal{El}(F \cdot a)$. We say the universe *has the base types* if $\mathcal{El}(1)$ is a unit type, $\mathcal{El}(0)$ is an empty type, and $\mathcal{El}(\mathsf{Nat})$ is a natural number type.

Definition 3 (PER model for IITT)**.** *A* PER model for IITT *is a cumulative hierarchy* $(\mathcal{S}et_k, \mathcal{E}l_k)$ *of universes that have the base types.*

For the definition of PER models we have just used the application operation on values and closures. To show soundness of IITT we also need to utilize the partial interpretation function $(\!|\_|\!)_\_$ and the computation laws.

Context. We write $(\!|t|\!)_\eta = (\!|t'|\!)_{\eta'} \in \mathcal{A}$ if there are $a = a' \in \mathcal{A}$ such that $t\,\eta \searrow a$ and $t'\,\eta' \searrow a'$. Also, Let $[\![T]\!]_\eta = \mathcal{E}l(\!|T|\!)_\eta$.

By induction on Γ we simultaneously define the proposition $\Gamma \Vdash$ and the PER $\rho = \rho' \in \Gamma$.

$$\diamond \Vdash \qquad \Gamma.^\star T \Vdash \iff \Gamma \Vdash$$
$$\text{and } (\!|T|\!)_\rho = (\!|T|\!)_{\rho'} \in \mathcal{S}et_\omega \text{ for all } \rho = \rho' \in \Gamma$$
$$\mathsf{id} = \mathsf{id} \in \diamond \qquad \langle \rho, \alpha\rangle = \langle \rho', \alpha'\rangle \in \Gamma.^\star T \iff$$
$$\rho = \rho' \in \Gamma \text{ and } \alpha = \alpha' \in \overline{[\![T]\!]}_\rho$$

Note that irrelevance markers are ignored by theses definitions. Thus, we have $\Gamma \Vdash$ iff $\Gamma^\oplus \Vdash$ and $\rho = \rho' \in \Gamma$ iff $\rho = \rho' \in \Gamma^\oplus$.

Environment approximation. By induction on Γ we define the proposition $\eta \preccurlyeq \rho \in \Gamma$.

$$\mathsf{id} \preccurlyeq \mathsf{id} \in \diamond$$
$$\langle \eta, \alpha\rangle \preccurlyeq \langle \rho, \beta\rangle \in \Gamma.T \iff$$
$$\eta \preccurlyeq \rho \in \Gamma \text{ and } \alpha = \beta \in \overline{[\![T]\!]}_\rho$$
$$\langle \eta, \alpha\rangle \preccurlyeq \langle \rho, \beta\rangle \in \Gamma.\dot{-}T \iff$$
$$\eta \preccurlyeq \rho \in \Gamma \text{ and } \alpha \in \mathsf{Clos} \text{ and } \beta \in \overline{[\![T]\!]}_\rho$$

Note that $\eta \preccurlyeq \rho \in \Gamma$ implies $\rho = \rho \in \Gamma$ and $\eta \preccurlyeq \rho \in \Gamma^\oplus$ implies $\eta = \rho \in \Gamma$.

Lemma 4. *Let* $\Gamma \Vdash$ *and* $\rho = \rho' \in \Gamma$*. Then* $\rho \preccurlyeq \rho' \in \Gamma$ *and* $\eta \preccurlyeq \rho' \in \Gamma$ *for all* $\eta \preccurlyeq \rho \in \Gamma$*.*

We write $\eta, \eta' \preccurlyeq \rho \in \Gamma$ iff $\eta \preccurlyeq \rho \in \Gamma$ and $\eta' \preccurlyeq \rho \in \Gamma$. Observe that $\_, \_ \preccurlyeq \rho \in \Gamma$ is a PER, trivially.

Judgements. The semantic counterparts $\Gamma \Vdash J$ of the remaining syntactic judgements $\Gamma \vdash J$ are defined as follows.

$$\Gamma \Vdash T \quad :\iff \quad \Gamma^\oplus \Vdash \text{ and}$$
$$\text{either } T = s \text{ or } \Gamma \Vdash T : s \text{ for some } s$$
$$\Gamma \Vdash t : T \quad :\iff \quad \Gamma \Vdash t = t : T$$
$$\Gamma \Vdash t = t' : T \quad :\iff \quad \Gamma \Vdash T \text{ and}$$
$$\forall \eta, \eta' \preccurlyeq \rho \in \Gamma.\ (\!|t|\!)_\eta = (\!|t'|\!)_{\eta'} \in [\![T]\!]_\rho$$
$$\Gamma \Vdash \sigma \preccurlyeq \tau : \Delta \quad :\iff \quad \Gamma \Vdash \sigma = \sigma \preccurlyeq \tau : \Delta$$
$$\Gamma \Vdash \sigma = \sigma' \preccurlyeq \tau : \Delta \quad :\iff \quad \Gamma^\oplus \Vdash \text{ and } \Delta^\oplus \Vdash \text{ and}$$
$$\forall \eta, \eta' \preccurlyeq \rho \in \Gamma.\ (\!|\sigma|\!)_\eta, (\!|\sigma'|\!)_{\eta'} \preccurlyeq (\!|\tau|\!)_\rho \in \Delta$$

Lemma 5 (Resurrection as weakening)**.**

1. *If* $\Gamma \Vdash J$ *then* $\Gamma^\oplus \Vdash J$*.*
2. *If* $\Gamma \Vdash \sigma = \sigma' \preccurlyeq \tau : \Delta^\oplus$ *then* $\Gamma \Vdash \sigma = \sigma' \preccurlyeq \tau : \Delta$*.*

Theorem 6 (Soundness of IITT)**.** *If* $\Gamma \vdash J$ *then* $\Gamma \Vdash J$*.*

Proof. By induction on $\Gamma \vdash J$. Each case follows quite directly from the induction hypotheses. □

4.3 Closed Model and Consistency

In the following, we construct the least PER model over the set of closed values Val, to show consistency of IITT.

Define PERs $\mathcal{E}mpty = \emptyset$ and $\mathcal{U}nit = \{(\langle\rangle, \langle\rangle)\}$ and let the PER $\mathcal{N}at$ be inductively given by:

$$\frac{}{\mathsf{zero} = \mathsf{zero} \in \mathcal{N}at} \qquad \frac{\alpha = \alpha' \in \mathcal{N}at}{\mathsf{succ}\,\alpha = \mathsf{succ}\,\alpha' \in \mathcal{N}at}$$

We define each of the universes $(\mathcal{S}et_k, \mathcal{E}l_k)$ inductive-recursively, and the whole sequence by induction on $k \in \mathbb{N}$. The clauses are:

$$\frac{}{0 = 0 \in \mathcal{S}et_0} \qquad \mathcal{E}l_0(0) = \mathcal{E}mpty$$

$$\frac{}{1 = 1 \in \mathcal{S}et_0} \qquad \mathcal{E}l_0(1) = \mathcal{U}nit$$

$$\frac{}{\mathsf{Nat} = \mathsf{Nat} \in \mathcal{S}et_0} \qquad \mathcal{E}l_0(\mathsf{Nat}) = \mathcal{N}at$$

$$\frac{A = A' \in \mathcal{S}et_k \qquad F \cdot \alpha = F' \cdot \alpha' \in \mathcal{S}et_k \text{ for all } \alpha = \alpha' \in \overline{\mathcal{E}l_k(A)}}{\Pi^\star A\,F = \Pi^\star A'\,F' \in \mathcal{S}et_k}$$
$$\mathcal{E}l_k(\Pi^\star A\,F) = \Pi^\star \alpha \in \mathcal{E}l_k(A).\ \mathcal{E}l_k(F \cdot \alpha)$$

$$\frac{A = A' \in \mathcal{S}et_k}{A = A' \in \mathcal{S}et_{k+1}} \qquad \mathcal{E}l_{k+1}(A) = \mathcal{E}l_k(A)$$

$$\frac{}{\mathsf{Set}_k = \mathsf{Set}_k \in \mathcal{S}et_{k+1}} \qquad \mathcal{E}l_{k+1}(\mathsf{Set}_k) = \mathcal{S}et_k$$

Then $\mathcal{S}et_\omega = \bigcup_{k\in\mathbb{N}} \mathcal{S}et_k$ and $\mathcal{E}l_\omega = \bigcup_{k\in\mathbb{N}} \mathcal{E}l_k$.

Lemma 7 (Closed PER model)**.** *The sequence* $(\mathcal{S}et_k, \mathcal{E}l_k)$ *is a PER model of IITT.*

Proof. Clearly, $\mathcal{U}nit$ is a unit type. The requirement $\mathsf{abort} \in \forall A \in \mathcal{S}et_k.\ \forall \beta \in \mathcal{E}mpty.\ \mathcal{E}l_k(A)$ follows from $\mathcal{E}mpty = \emptyset$. The well-definedness of natrec follows by induction on $\mathcal{N}at$ using the laws of weak head evaluation. □

Theorem 8 (Consistency)**.** $\not\vdash t : 0$.

Proof. By Thm. 6 we have $(\!|t|\!)_{\mathsf{id}} \in \mathcal{E}mpty$ in contradiction to the definition of $\mathcal{E}mpty$. □

5 Type Checking and Extraction

In this section, we specify a procedure that type-checks input $t : T$ and extracts an untyped lambda term M from t in which all implicit abstractions and applications have been removed. Our bidirectional type-checker [20, 5] does not require type annotation at λs but can therefore only check terms in β-normal form (which is in reality not a severe restriction in practice). Implicit application can only be type checked in the form

$$\frac{\Gamma \vdash t : \forall\, U\, T \qquad \Gamma^{\oplus} \vdash u : U}{\Gamma \vdash t\, u : T\, u}$$

and not in the form $t\, u'$ for an arbitrary u' as permitted by rule APP2. After all, the sole purpose of the argument u in implicit application is to aid the type checker, and it will be discarded afterwards.

Since implicit abstraction $\lambda t : \forall\, U\, T$ will be erased as well, it is convenient to extract *named* terms rather than de Bruijn terms. This saves us shifting de Bruijn indices; if necessary the named term can be converted back to de Bruijn after extraction is complete. Named terms are given by the usual grammar

$$M, N ::= c \mid x \mid \lambda x M \mid M\, N$$

where we consider new arities for abort^0 and natrec^3 since the implicit arguments have been dropped.

Bidirectional type-checking and extraction is given by the two judgements:

$\Delta \vdash t \rightrightarrows A\, [\rightsquigarrow M]$ in Δ, the inferred type of t is A
$\Delta \vdash t \leftrightarrows A\, [\rightsquigarrow M]$ in Δ, term t checks against type A

Both judgements have the extraction M of t as an optional output. It is convenient to keep types A in evaluated form during the process of type checking. Consequently, Δ is a context of type *values*. To save us de Bruijn index shifting, we endorse a *locally nameless* approach to values [21]. Free variables are represented by de Bruijn levels x_j while restricting indices v_i to bound variables. We write x_Δ for $\mathsf{x}_{|\Delta|}$. We extend the set of values a by de Bruijn levels x_i and neutral values e:

$$\begin{array}{lcl} a & ::= & \cdots \mid e \\ e & ::= & \mathsf{x}_i \mid e\, \alpha \mid \mathsf{abort}_\beta\, e \mid \mathsf{natrec}_\beta\, \alpha_z\, \alpha_s\, e \end{array}$$

For evaluation of terms needed during type checking, we define the identity environment id_Δ which replaces the $|\Delta|$ free de Bruijn indices by de Bruijn levels as follows:

$$\begin{array}{lcl} \mathsf{id}_\diamond & = & \mathsf{id} \\ \mathsf{id}_{\Delta.\star A} & = & \langle \mathsf{id}_\Delta, \mathsf{x}_\Delta \rangle \end{array}$$

Rules for inferring the level of a type $\Delta \vdash T \rightrightarrows \mathsf{Set}_k$. Checking well-formedness of types involves inference of their universe level. We only accept Π (and $\forall$)-types in the form $\Pi\, U\, \lambda T$ which is not a restriction since, coming from actual input $\Pi x : U.\, T$, they are always in this form.

$$\frac{}{\Delta \vdash \mathsf{Set}_k \rightrightarrows \mathsf{Set}_{k+1}} \qquad \frac{}{\Delta \vdash c \rightrightarrows \mathsf{Set}_0}\ c \in \{0, 1, \mathsf{Nat}\}$$

$$\frac{\Delta \vdash U \rightrightarrows \mathsf{Set}_i \qquad \Delta.^\star(\!|U|\!)_{\mathsf{id}_\Delta} \vdash T \rightrightarrows \mathsf{Set}_j}{\Delta \vdash \Pi^\star U\, \lambda T \rightrightarrows \mathsf{Set}_{\max(i,j)}}$$

We need not extract types since they have no computational content. If they extraction should be demanded nevertheless, we return a dummy value:

$$\frac{\Delta \vdash T \rightrightarrows \mathsf{Set}_k}{\Delta \vdash T \rightrightarrows \mathsf{Set}_k \rightsquigarrow \langle\rangle}$$

This is necessary, e. g., if we extract $\mathsf{natrec}_{\lambda.\mathsf{natrec}_{\mathsf{Set}_1}\,\mathsf{Nat}\,(\lambda\lambda\mathsf{Set}_0)\,\mathsf{v}_0}\,\mathsf{zero}\,(\lambda\lambda\mathsf{Nat})$, a function that returns value zero on input zero and type Nat on other inputs. The extracted term is then $\mathsf{natrec}\,\mathsf{zero}\,(\lambda\lambda\langle\rangle)$.

Rules of type inference $\Delta \vdash t \rightrightarrows A \rightsquigarrow M$. The type of a applicative term starting with a variable or a constant can be inferred. Extraction in context Δ turns a de Bruijn index v_i into a name of a de Bruijn level $\mathsf{x}_{|\Delta|-1-i}$, which can later be abstracted over.

$$\frac{}{\Delta \vdash \mathsf{v}_i \rightrightarrows \Delta(i) \rightsquigarrow \mathsf{x}_{|\Delta|-1-i}}$$

$$\frac{\Delta \vdash t \rightrightarrows \Pi\, A\, F \rightsquigarrow M \qquad \Delta \vdash u \leftrightarrows A \rightsquigarrow N}{\Delta \vdash t\, u \rightrightarrows F \cdot (u\, \mathsf{id}_\Delta) \rightsquigarrow M\, N}$$

$$\frac{\Delta \vdash t \rightrightarrows \forall\, A\, F \rightsquigarrow M \qquad \Delta^{\oplus} \vdash u \leftrightarrows A}{\Delta \vdash t\, u \rightrightarrows F \cdot (u\, \mathsf{id}_\Delta) \rightsquigarrow M}$$

$$\frac{\vdash_\Sigma c : T}{\Delta \vdash c \rightrightarrows T \rightsquigarrow c}\ c \in \{\langle\rangle, \mathsf{zero}, \mathsf{succ}\}$$

$$\frac{\Delta^{\oplus} \vdash T \rightrightarrows \mathsf{Set}_k}{\Delta \vdash \mathsf{abort}_T \rightrightarrows (\!|\forall\, 0\, \lambda T|\!)_{\mathsf{id}_\Delta} \rightsquigarrow \mathsf{abort}}$$

$$\frac{\Delta^{\oplus}.\mathsf{Nat} \vdash C \rightrightarrows \mathsf{Set}_k}{\Delta \vdash \mathsf{natrec}_{\lambda C} \rightrightarrows (\!|T_C|\!)_{\mathsf{id}_\Delta} \rightsquigarrow \mathsf{natrec}}$$

In the last rule $T_C = C\,\langle\mathsf{zero}\rangle \to (\Pi\,\mathsf{Nat}\,\lambda.\, C\,\langle\mathsf{v}_0\rangle \to C\,\langle\mathsf{succ}\,\mathsf{v}_1\rangle) \to \Pi\,\mathsf{Nat}\,\lambda.\, C\,\langle\mathsf{v}_0\rangle$. The rules for abort and natrec not subsumed under the rule for constants c as to account for universe polymorphism (they can be used at any level k).

Rules of type checking $\Delta \vdash t \leftleftarrows A \rightsquigarrow M$. Only λ-abstractions cannot be inferred, they need a type to be checked against as to provide the type of the abstracted variable. Note that extraction discards the implicit abstraction (case $\forall\, A\, F$). When asked to check non-abstractions t, their type A is inferred and then compared to the ascribed type B.

$$\frac{\Delta.A \vdash t \leftleftarrows F \cdot \mathsf{x}_\Delta \rightsquigarrow M}{\Delta \vdash \lambda t \leftleftarrows \Pi\, A\, F \rightsquigarrow \lambda \mathsf{x}_\Delta M}$$

$$\frac{\Delta.\dot{\div}A \vdash t \leftleftarrows F \cdot \mathsf{x}_\Delta \rightsquigarrow M}{\Delta \vdash \lambda t \leftleftarrows \forall\, A\, F \rightsquigarrow M}$$

$$\frac{\Delta \vdash t \rightrightarrows A \rightsquigarrow M \qquad \Delta \vdash A \leq B}{\Delta \vdash t \leftleftarrows B \rightsquigarrow M}$$

This completes the specification of the type checker. The implementation of algorithmic subtyping $\Delta \vdash A \leq B$ is left for future work. We favor the solution which computes the normal forms of A and B with normalization by evaluation [4] and then employs structural comparison.

6 On Bracket Types

Awodey and Bauer's bracket type former $[A]$ is another means to mark the terms of type A as irrelevant. Practically, it can be used when one wants to consider type A as a proposition. It is important to note that bracket types and implicit quantification, although they look similar, do not subsume each other. In type $\Pi x : [A].\, B$, type B cannot depend on x, as opposed to Miquel's $\forall x : A.\, B$. On the other hand, bracket types can also appear in positive positions as in the type $\Pi x : \mathsf{Nat}.\, [x \geq 0]$, thus they are useful for manipulating proofs.

Bracket types can be readily added to IITT with the following rules:

$$\frac{\Gamma \vdash A : \mathsf{Set}_k}{\Gamma \vdash [A] : \mathsf{Set}_k} \qquad \frac{\Gamma \vdash M : A}{\Gamma \vdash M : [A]}$$

$$\frac{\Gamma \vdash N : [A] \qquad \Gamma \vdash B \qquad \Gamma, x{:}A \vdash M : B \qquad \Gamma, x{:}A, y{:}A \vdash M = M[y/x] : B}{\Gamma \vdash M \;\mathsf{where}\; x \leftarrow N : B}$$

The PER model can be easily extended using the semantics $a = a' \in [\mathcal{A}] \iff a, a' \in \mathcal{A}$. Type directed extraction replaces all inhabitants of bracket types by $\langle\rangle$ and simplifies $(M \;\mathsf{where}\; x \leftarrow N)$ to M.

7 Conclusions

We have presented IITT with a typed equality judgement for implicit quantification with a surprising twist in the application rule. Typed equality has two advantages: It allows equality axioms which rely on typing, especially η-laws for types with at most one constructor, such as Σ-, unit, and empty types. A second advantage is the presentation of models, which requires no technical tricks as opposed to models for untyped equality.

MiniAgda [3] has a prototypical implementation of implicit quantification à la IITT, which dispenses with irrelevance markers on the term level (as opposed to Barras' ICC*).

To complete this work, we aim to implement and verify normalization by evaluation for IITT. We also like to investigate how to integrate the Werner's universe-based notion of irrelevance [22] which allows to eliminate identity proofs into types. Preliminary investigations have been carried out [2], but many open questions remain.

Acknowledgments. Thanks to Bruno Barras and Bruno Bernardo for discussions on implicit quantification. Thanks also to Hugo Herbelin, Brigitte Pientka, Matthieu Sozeau, and Lionel Vaux for their interest in this work.

References

[1] M. Abadi, L. Cardelli, P.-L. Curien, and J.-J. Lévy. Explicit substitutions. *JFP*, 1(4):375–416, 1991.

[2] A. Abel. Extensional normalization in the logical framework with proof irrelevant equality. In *Wksh. on Normalization by Evaluation*, 2009.

[3] A. Abel. Miniagda, a toy implementation of type theory. http://www2.tcs.ifi.lmu.de/~abel/miniagda/, 2010.

[4] A. Abel, T. Coquand, and P. Dybjer. Normalization by evaluation for Martin-Löf Type Theory with typed equality judgements. In *LICS'07*, pages 3–12. IEEE CS Press, 2007.

[5] A. Abel, T. Coquand, and P. Dybjer. Verifying a semantic $\beta\eta$-conversion test for Martin-Löf type theory. In *MPC'08*, volume 5133 of *LNCS*, pages 29–56. Springer, 2008.

[6] S. Awodey and A. Bauer. Propositions as [Types]. *JLC*, 14(4):447–471, 2004.

[7] B. Barras and B. Bernardo. The implicit calculus of constructions as a programming language with dependent types. In *FoSSaCS*, volume 4962 of *LNCS*, pages 365–379. Springer, 2008.

[8] G. Barthe, B. Grégoire, and C. Riba. Type-based termination with sized products. In *CSL'09*, volume 5213 of *LNCS*, pages 493–507. Springer, 2008.

[9] S. Berardi, M. Coppo, F. Damiani, and P. Giannini. Type-based useless-code elimination for functional programs. In *SAIG'00*, volume 1924 of *LNCS*, pages 172–189. Springer, 2000.

[10] E. Brady, C. McBride, and J. McKinna. Inductive families need not store their indices. In *TYPES'03*, volume 3085 of *LNCS*, pages 115–129. Springer, 2004.

[11] INRIA. *The Coq Proof Assistant Reference Manual*. INRIA, version 8.2 edition, 2008. http://coq.inria.fr/.

[12] P. Letouzey. Extraction in coq: An overview. In *CiE'08*, volume 5028 of *LNCS*, pages 359–369. Springer, 2008.

[13] P. Martin-Löf. Substitution calculus. Unpublished notes from a lecture in Göteborg, 1992.
[14] A. Miquel. A model for impredicative type systems, universes, intersection types and subtyping. In *LICS'00*, pages 18–29, 2000.
[15] A. Miquel. The implicit calculus of constructions. In *TLCA'01*, volume 2044 of *LNCS*, pages 344–359. Springer, 2001.
[16] A. Miquel. *Le Calcul des Constructions implicite: syntaxe et sémantique.* PhD thesis, Université Paris 7, 2001.
[17] A. Nanevski, F. Pfenning, and B. Pientka. Contextual modal type theory. *ACM Trans. Comput. Log.*, 9(3), 2008.
[18] U. Norell. *Towards a Practical Programming Language Based on Dependent Type Theory*. PhD thesis, Chalmers, Göteborg, Sweden, 2007.
[19] F. Pfenning. Intensionality, extensionality, and proof irrelevance in modal type theory. In *LICS'01*. IEEE CS Press, 2001.
[20] B. C. Pierce and D. N. Turner. Local type inference. In *POPL'98*, San Diego, California, 1998.
[21] R. Pollack. Closure under alpha-conversion. In *TYPES'93*, volume 806 of *LNCS*, pages 313–332. Springer, 1994.
[22] B. Werner. On the strength of proof-irrelevant type theories. *LMCS*, 4(3), 2008.

A Proofs

Theorem (2 Syntactic Validity)**.**

1. *If* $\Gamma.\Gamma' \vdash J$ *then* $\Gamma^{\oplus} \vdash$.
2. *If* $\Gamma \vdash_{(\Sigma)} L : T$ *then* $\Gamma^{\oplus} \vdash T : s$ *for some sort* s.
3. *If* $\Gamma \vdash_{(\Sigma)} t = t' : T$ *then* $\Gamma \vdash t : T$ *and* $\Gamma \vdash t' : T$.
4. *If* $\Gamma \vdash L \preccurlyeq \tau : \Delta$ *then* $\Delta^{\oplus} \vdash$ *and* $\Gamma^{\oplus} \vdash \tau \preccurlyeq \tau : \Delta^{\oplus}$.
5. *If* $\Gamma \vdash \sigma = \sigma' \preccurlyeq \tau : \Delta$ *then* $\Gamma \vdash \sigma \preccurlyeq \tau : \Delta$ *and* $\Gamma \vdash \sigma' \preccurlyeq \tau : \Delta$.

Proof. Simultaneously by induction on the derivation. Let us look at proposition 4. The first goal $\Delta^{\oplus} \vdash$ is trivial, we focus on $\Gamma^{\oplus} \vdash \tau \preccurlyeq \tau : \Delta^{\oplus}$ for $L = \sigma$ a substitution.

Case

$$\frac{\Gamma \vdash \sigma \preccurlyeq \tau : \Delta \qquad \Delta^{\oplus} \vdash T : s \qquad \Gamma \vdash t' \preccurlyeq t \star T\,\tau}{\Gamma \vdash \langle \sigma, t' \rangle \preccurlyeq \langle \tau, t \rangle : \Delta.\star T}$$

We have $\Gamma^{\oplus} \vdash \tau \preccurlyeq \tau : \Delta^{\oplus}$ by induction hypothesis, and since resurrection is idempotent, $\Delta^{\oplus\oplus} \vdash T : s$ by assumption. In case $\star = \div$, we have $\Gamma^{\oplus} \vdash t : T\,\tau$ by assumption, otherwise it follows from the assumption $\Gamma \vdash t : T\,\tau$ by Lemma 1. Together, we can infer $\Gamma^{\oplus} \vdash \langle \tau, t \rangle : \Delta^{\oplus}.T$.

Case

$$\frac{\Gamma^{\oplus} \vdash}{\Gamma \vdash \mathsf{id} \preccurlyeq \mathsf{id} : \Gamma}$$

Since $\Gamma^{\oplus\oplus} \vdash$ by assumption, $\Gamma^{\oplus} \vdash \mathsf{id} \preccurlyeq \mathsf{id} : \Gamma^{\oplus}$.

Case

$$\frac{\Gamma_1 \vdash \sigma' \preccurlyeq \tau' : \Gamma_2 \qquad \Gamma_2 \vdash \sigma \preccurlyeq \tau : \Gamma_3}{\Gamma_1 \vdash \sigma\,\sigma' \preccurlyeq \tau\,\tau' : \Gamma_3}$$

By composition of the induction hypotheses.

Case

$$\frac{\Gamma^{\oplus} \vdash T : s}{\Gamma.\star T \vdash \uparrow \preccurlyeq \uparrow : \Gamma}$$

By assumption $\Gamma^{\oplus\oplus} \vdash T : s$, hence, $\Gamma^{\oplus}.T \vdash \uparrow \preccurlyeq \uparrow : \Gamma^{\oplus}$.

Case

$$\frac{\Gamma \vdash \sigma \preccurlyeq \tau : \Delta \qquad \Gamma^{\oplus} \vdash \tau = \tau' \preccurlyeq \tau' : \Delta^{\oplus}}{\Gamma \vdash \sigma \preccurlyeq \tau' : \Delta}$$

By induction hypothesis 5, $\Gamma^{\oplus} \vdash \tau' \preccurlyeq \tau' : \Delta^{\oplus}$.

Next, consider proposition 2.

Case

$$\frac{\Gamma \vdash \sigma \preccurlyeq \tau : \Delta \qquad \Delta \vdash t : T}{\Gamma \vdash t\,\sigma : T\,\tau}$$

By induction hypothesis $\Delta^{\oplus} \vdash T : s$ for some sort s, and $\Gamma^{\oplus} \vdash \tau \preccurlyeq \tau : \Delta^{\oplus}$, from which we obtain $\Gamma^{\oplus} \vdash T\,\tau : s\,\tau$ entailing the goal $\Gamma^{\oplus} \vdash T\,\tau : s$.

For proposition 5, consider the cases:

Case conversion.

$$\frac{\Gamma \vdash \sigma = \sigma' \preccurlyeq \tau : \Delta \qquad \Gamma^{\oplus} \vdash \tau = \tau' \preccurlyeq \tau' : \Delta^{\oplus}}{\Gamma \vdash \sigma = \sigma' \preccurlyeq \tau' : \Delta}$$

By induction hypothesis, $\Gamma \vdash \sigma \preccurlyeq \tau : \Delta$ which entails $\Gamma \vdash \sigma \preccurlyeq \tau' : \Delta$ by conversion.

Case left identity.

$$\frac{\Gamma \vdash \sigma \preccurlyeq \tau : \Delta}{\Gamma \vdash \mathsf{id}\,\sigma = \sigma \preccurlyeq \tau : \Delta}$$

We have $\Gamma \vdash \mathsf{id}\,\sigma \preccurlyeq \mathsf{id}\,\tau : \Delta$, and since by induction hypothesis $\Gamma^{\oplus} \vdash \tau \preccurlyeq \tau : \Delta^{\oplus}$, we can use $\Gamma^{\oplus} \vdash \mathsf{id}\,\tau = \tau \preccurlyeq \tau : \Delta^{\oplus}$ to convert our first inference to the goal $\Gamma \vdash \mathsf{id}\,\sigma \preccurlyeq \tau : \Delta$.

Case second projection.

$$\frac{\Gamma \vdash \sigma \preccurlyeq \tau : \Delta \qquad \Delta^{\oplus} \vdash T : s \qquad \Gamma^{\oplus} \vdash u \star T\,\tau}{\Gamma \vdash \uparrow \langle \sigma, t \rangle = \sigma \preccurlyeq \tau : \Delta}$$

First, $\Gamma \vdash \langle \sigma, t \rangle \preccurlyeq \langle \tau, u \rangle : \Delta.^{\star}T$, and since $\Delta.^{\star}T \vdash \uparrow \preccurlyeq \uparrow : \Delta$, $\Gamma \vdash \uparrow \langle \sigma, t \rangle \preccurlyeq \uparrow \langle \tau, u \rangle : \Delta$. Since by induction hypothesis $\Gamma^{\oplus} \vdash \tau \preccurlyeq \tau : \Delta^{\oplus}$, another instance of the above rule yields $\Gamma^{\oplus} \vdash \uparrow \langle \tau, u \rangle = \tau \preccurlyeq \tau : \Delta^{\oplus}$. Thus we use conversion to arrive at our goal $\Gamma \vdash \uparrow \langle \sigma, t \rangle \preccurlyeq \tau : \Delta$.

Finally let us look at proposition 3.

Case

$$\frac{\Gamma \vdash \sigma \preccurlyeq \tau : \Delta \qquad \Delta.^{\star}U \vdash t : T}{\Gamma \vdash (\lambda t)\,\sigma = \lambda.\, t\,\langle \sigma \uparrow, \mathsf{v}_0 \rangle : \Pi^{\star}(U\,\tau)\,(\lambda.\, T\,\langle \tau \uparrow, \mathsf{v}_0 \rangle)}$$

Let us first establish some general facts.

$\Gamma^{\oplus} \vdash \tau \preccurlyeq \tau : \Delta^{\oplus}$	by ind.hyp.
$\Delta^{\oplus} \vdash U : s$	by ind.hyp.
$\Gamma^{\oplus} \vdash U\,\tau : s\,\tau$	by substitution
$\vdash_{\Sigma} s : s'$	by sort abundance
$\Gamma^{\oplus} \vdash s\,\tau = s : s'$	
$\Gamma^{\oplus} \vdash U\,\tau : s$	by conversion
$(\Gamma.^{\star}U\,\tau)^{\oplus} \vdash$	

The first goal $\Gamma \vdash (\lambda t)\,\sigma : \Pi^{\star}(U\,\tau)\,\lambda.\, T\,\langle \tau \uparrow, \mathsf{v}_0 \rangle$ is then derived as follows:

$\Delta \vdash \lambda t : \Pi^{\star} U\,\lambda T$	by abstraction
$\Gamma \vdash (\lambda t)\,\sigma : (\Pi^{\star} U\,\lambda T)\,\tau$	by substitution
$\Gamma^{\oplus} \vdash U\,\tau = U\,\tau : s$	by reflexivity
$\Delta^{\oplus}.U \vdash T : s''$	by ind.hyp.
$\Gamma^{\oplus} \vdash (\lambda T)\,\tau = \lambda.\, T\,\langle \tau \uparrow, \mathsf{v}_0 \rangle : \Pi\,(U\,\tau)\,(\lambda.\, s''\,\langle \tau \uparrow, \mathsf{v}_0 \rangle)$	by this rule
$\Gamma^{\oplus} \vdash \Pi\,(U\,\tau)\,(\lambda.\, s''\,\langle \tau \uparrow, \mathsf{v}_0 \rangle) = \Pi\,(U\,\tau)\,(\lambda s'') : s'''$	by boring derivation
$\Gamma^{\oplus} \vdash (\lambda T)\,\tau = \lambda.\, T\,\langle \tau \uparrow, \mathsf{v}_0 \rangle : \Pi\,(U\,\tau)\,(\lambda s'')$	by conversion
$\Gamma^{\oplus} \vdash \Pi^{\star}(U\,\tau)\,((\lambda T)\,\tau) = \Pi^{\star}(U\,\tau)\,(\lambda.\, T\,\langle \tau \uparrow, \mathsf{v}_0 \rangle) : \max(s, s'')$	by compatibility
$\Gamma^{\oplus} \vdash (\Pi^{\star} U\,\lambda T)\,\tau = \Pi^{\star}(U\,\tau)\,(\lambda.\, T\,\langle \tau \uparrow, \mathsf{v}_0 \rangle) : \max(s, s'')$	by propagation
$\Gamma \vdash (\lambda t)\,\sigma : \Pi^{\star}(U\,\tau)\,(\lambda.\, T\,\langle \tau \uparrow, \mathsf{v}_0 \rangle)$	by conversion

The second goal $\Gamma \vdash \lambda.\, t\,\langle \sigma \uparrow, \mathsf{v}_0 \rangle : \Pi^{\star}(U\,\tau)\,\lambda.\, T\,\langle \tau \uparrow, \mathsf{v}_0 \rangle$ follows from the facts by this sequence of inferences: as

follows:

$\Gamma.^\star U\,\tau \vdash \uparrow \preccurlyeq \uparrow : \Gamma$
$\Gamma.^\star U\,\tau \vdash \sigma\uparrow \preccurlyeq \tau\uparrow : \Delta$ by composition
$\Gamma.^\star U\,\tau \vdash \mathsf{v}_0 \preccurlyeq \mathsf{v}_0 \star U\,\tau\uparrow$ by variable rule
$\Gamma.^\star U\,\tau \vdash \langle \sigma\uparrow, \mathsf{v}_0\rangle \preccurlyeq \langle \tau\uparrow, \mathsf{v}_0\rangle : \Delta.^\star U$ by pairing
$\Gamma.^\star U\,\tau \vdash t\,\langle \sigma\uparrow, \mathsf{v}_0\rangle : T\,\langle \tau\uparrow, \mathsf{v}_0\rangle$ by substitution
$\Gamma \vdash \lambda.\, t\,\langle \sigma\uparrow, \mathsf{v}_0\rangle : \Pi^\star(U\,\tau)\,\lambda.\, T\,\langle \tau\uparrow, \mathsf{v}_0\rangle$ by abstraction

□

Theorem (6 Soundness of IITT)**.** *If* $\Gamma \vdash J$ *then* $\Gamma \Vdash J$.

Proof. By induction on $\Gamma \vdash J$. Each case follows quite directly from the induction hypotheses.

Case

$$\frac{\Gamma \vdash \sigma \preccurlyeq \tau : \Delta \qquad \Delta^\oplus \vdash T : s \qquad \Gamma^\oplus \vdash t : T\,\tau}{\Gamma \vdash \langle \sigma, t'\rangle \preccurlyeq \langle \tau, t\rangle : \Delta.\dot{\div} T}$$

$\Gamma^\oplus \Vdash$ goal 1, by ind.hyp.1
$\Delta^\oplus \Vdash T : s$ by ind.hyp.2
$(\Delta.\dot{\div} T)^\oplus = \Delta^\oplus.T \Vdash$ goal 2
$\eta, \eta' \preccurlyeq \rho \in \Gamma$ assumption
$(\!|\sigma|\!)_\eta, (\!|\sigma|\!)_{\eta'} \preccurlyeq (\!|\tau|\!)_\rho \in \Delta$ by ind.hyp.1
$\rho, \rho \preccurlyeq \rho \in \Gamma^\oplus$
$(\!|t|\!)_\rho = (\!|t|\!)_\rho \in [\![T\,\tau]\!]_\rho$ by ind.hyp.3
$[\![T\,\tau]\!]_\rho = [\![T]\!]_{(\!|\tau|\!)_\rho}$ by ind.hyp.2
$t\,\rho \in \overline{[\![T]\!]}_{(\!|\tau|\!)_\rho}$
$t'\,\eta, t'\,\eta' \in \mathsf{Clos}$
$(\!|\langle \sigma, t'\rangle|\!)_\eta, (\!|\langle \sigma, t'\rangle|\!)_{\eta'} \preccurlyeq (\!|\langle \tau, t\rangle|\!)_\rho \in \Delta.\dot{\div} T$ goal 3

Case

$$\frac{\Gamma \vdash \sigma = \sigma' \preccurlyeq \tau : \Delta \qquad \Delta^\oplus \vdash T : s \qquad \Gamma^\oplus \vdash t' : T\,\tau}{\Gamma \vdash \langle \sigma, t\rangle = \langle \sigma', t'\rangle \preccurlyeq \langle \tau, t'\rangle : \Delta.\dot{\div} T}$$

$\Gamma^{\oplus} \Vdash$ goal 1, by ind.hyp.1

$\Delta^{\oplus} \Vdash T : s$ by ind.hyp.2

$(\Delta.\div T)^{\oplus} = \Delta^{\oplus}.T \Vdash$ goal 2

$\eta, \eta' \preccurlyeq \rho \in \Gamma$ assumption

$(\!|\sigma|\!)_\eta, (\!|\sigma'|\!)_{\eta'} \preccurlyeq (\!|\tau|\!)_\rho \in \Delta$ by ind.hyp.1

$\rho \preccurlyeq \rho \in \Gamma^{\oplus}$

$(\!|t'|\!)_\rho \in [\![T\,\tau]\!]_\rho$ by ind.hyp.3

$[\![T\,\tau]\!]_\eta = [\![T]\!]_{(\!|\tau|\!)_\eta}$ by ind.hyp.2

$t'\,\rho \in \overline{[\![T]\!]}_{(\!|\tau|\!)_\rho}$

$t\,\eta, t'\,\eta' \in \mathsf{Clos}$

$(\!|\langle\sigma, t\rangle|\!)_\eta, (\!|\langle\sigma', t'\rangle|\!)_{\eta'} \preccurlyeq (\!|\langle\tau, t'\rangle|\!)_\rho \in \Delta.\div T$ goal 3

Case

$$\frac{\Gamma \vdash \sigma = \sigma' \preccurlyeq \tau : \Delta \qquad \Gamma^{\oplus} \vdash \tau = \tau' \preccurlyeq \tau' : \Delta^{\oplus}}{\Gamma \vdash \sigma = \sigma' \preccurlyeq \tau' : \Delta}$$

$\eta, \eta' \preccurlyeq \rho \in \Gamma$ assumption

$(\!|\sigma|\!)_\eta, (\!|\sigma'|\!)_{\eta'} \preccurlyeq (\!|\tau|\!)_\rho \in \Delta$ by ind.hyp.1

$\rho \preccurlyeq \rho \in \Gamma^{\oplus}$ by Lemma 4

$(\!|\tau|\!)_\rho \preccurlyeq (\!|\tau'|\!)_\rho \in \Delta^{\oplus}$ by ind.hyp.2

$(\!|\sigma|\!)_\eta, (\!|\sigma'|\!)_{\eta'} \preccurlyeq (\!|\tau'|\!)_\rho \in \Delta$ by Lemma 4

Case

$$\frac{\Gamma.\div U \vdash t = t' : T}{\Gamma \vdash \lambda t = \lambda t' : \Pi \div U\ \lambda T}$$

$\eta, \eta' \preccurlyeq \rho \in \Gamma$ assumption

$\beta, \beta' \in \mathsf{Clos}, a \in [\![U]\!]_\rho$ assumption

$\langle\eta, \beta\rangle, \langle\eta', \beta'\rangle \preccurlyeq \langle\rho, a\rangle \in \Gamma.\div U$ by def.

$(\!|t|\!)_{\langle\eta,\beta\rangle} = (\!|t'|\!)_{\langle\eta',\beta'\rangle} \in [\![T]\!]_{\langle\rho,a\rangle}$ by ind.hyp.

$(\!|\lambda t|\!)_\eta \cdot \beta = (\!|\lambda t'|\!)_{\eta'} \cdot \beta' \in \mathcal{E}l((\!|\lambda T|\!)_\rho \cdot a)$ by def.

$(\!|\lambda t|\!)_\eta = (\!|\lambda t'|\!)_{\eta'} \in \forall a \in [\![U]\!]_\rho.\ \mathcal{E}l((\!|\lambda T|\!)_\rho \cdot a)$

$(\!|\lambda t|\!)_\eta = (\!|\lambda t'|\!)_{\eta'} \in [\![\Pi \div U\ \lambda T]\!]_\rho$

Case

$$\frac{\Gamma \vdash t_1 = t_2 : \forall\, U\, T \qquad \Gamma^{\oplus} \vdash u : U}{\Gamma \vdash t_1\, u_1 = t_2\, u_2 : T\, u}$$

$\eta_1, \eta_2 \preccurlyeq \rho \in \Gamma$ assumption
$(\!|t_1|\!)_{\eta_1} = (\!|t_2|\!)_{\eta_2} \in [\![\forall\, U\, T]\!]_\rho$ by ind.hyp.1
$\rho \preccurlyeq \rho \in \Gamma^{\oplus}$ by Lemma 4
$(\!|u|\!)_\rho \in [\![U]\!]_\rho$ by ind.hyp.2
$u_1\, \eta_1, u_2\, \eta_2 \in \mathsf{Clos}$
$(\!|t_1\, u_1|\!)_{\eta_1} = (\!|t_2\, u_2|\!)_{\eta_2} \in [\![T\, u]\!]_\rho$ goal

Case

$$\frac{\Gamma \vdash \sigma = \sigma' \preccurlyeq \tau : \Delta \qquad \Delta \vdash t = t' : T}{\Gamma \vdash t\,\sigma = t'\,\sigma' : T\,\tau}$$

$\eta, \eta' \preccurlyeq \rho \in \Gamma$ assumption
$(\!|\sigma|\!)_\eta, (\!|\sigma'|\!)_{\eta'} \preccurlyeq (\!|\tau|\!)_\rho \in \Delta$ by ind.hyp.1
$(\!|t|\!)_{(\!|\sigma|\!)_\eta} = (\!|t'|\!)_{(\!|\sigma'|\!)_{\eta'}} \in [\![T]\!]_{(\!|\tau|\!)_\rho}$ by ind.hyp.2
$\forall u, a. \quad (\!|u|\!)_{(\!|\sigma|\!)_\rho} \searrow a \iff (\!|u\,\sigma|\!)_\rho \searrow a$ by def.
$(\!|t\,\sigma|\!)_\eta = (\!|t'\,\sigma'|\!)_{\eta'} \in [\![T\,\tau]\!]_\rho$

□