On Extensionality of $\lambda*$

Andrew Polonsky

Laboratoire PPS, Université Paris-Diderot – Paris 7, Paris, France
andrew.polonsky@gmail.com

Abstract

We prove an extensionality theorem for the "type-in-type" dependent type theory with Σ-types. We suggest that in type theory the notion of extensional equality be identified with the logical equivalence relation defined by induction on type structure.

1998 ACM Subject Classification F.4.1 Mathematical Logic

Keywords and phrases Extensionality, logical relations, type theory, lambda calculus, reflection

Digital Object Identifier 10.4230/LIPIcs.TYPES.2014.221

1 Introduction

Our goal is to find a formulation of type theory with extensional equality in such a way that the essential features of type theory – including canonicity, normalization, and decidability of type checking – remain valid.

This paper is the first in a series where we will pursue this goal using a syntactic methodology. Specifically, we will define the extensional equality type by induction on type structure, so that, for example, the equality on Π- and Σ-types is definitionally equal to

$$f \simeq_{\Pi x:A.B(x)} f' = \prod_{xx':A} \prod_{x^*:x\simeq_A x'} fx \simeq_{B(x^*)} f'x' \tag{1}$$

$$(a,b) \simeq_{\Sigma x:A.B(x)} (a',b') = \sum_{x^*:a\simeq_A a'} b \simeq_{B(x^*)} b' \tag{2}$$

The view that the notion of extensional equality in type theory should be identified with the equivalence relation so defined has extensive presence in the literature, going back at least to Tait [9] and Altenkirch [1].

This position is to be contrasted with another prominent view, which takes extensional equality to mean the adjunction to Martin-Löf intensional identity type $\mathit{Id}_A(x,y)$ of the propositional reflection rule

$$\frac{p : \mathit{Id}_A(s,t)}{s = t : A}$$

This inference rule commits treason against some fundamental design principles of type theory, resulting in the failure of normalization and related pathologies.

We therefore begin with a moment of "full disclosure":

Extensionality Thesis. The extensional equality of type theory is the logical equivalence relation between elements of the term model defined by induction on type structure.

Before we lay out in greater detail our program to convert this philosophical thesis into a mathematical definition, we invite the reader to consider what the final result may be expected to look like.

20th International Conference on Types for Proofs and Programs (TYPES 2014).
Editors: Hugo Herbelin, Pierre Letouzey, and Matthieu Sozeau; pp. 221–250
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

First and foremost – we seek a canonical way to associate to every type A, and any two terms of type A, a (possibly new) type of *extensional equalities* between the two terms. In other words, given expressions A, a, a', we wish to define, or construct, a new expression $a \simeq_A a'$ validating the following rule:

$$\frac{A : * \qquad a : A \qquad a' : A}{a \simeq_A a' : *}$$

(In this paper, $*$ denotes the universe of types.)

So far, we have redisplayed the formation rule for the intensional identity type with a new symbol. The difference here is that, in general, *we do not require $a \simeq_A a'$ to be a new type constructor.* We expect that the $\simeq_A$-symbol may well behave like a defined function, so that it *reduces* to previously defined types, according to the top symbol of A. This is the situation encountered in (1) and (2).

At the same time, we want to be able to iterate the $\simeq$-operation, so that we can form the types $a \simeq_A a'$, $p \simeq_{a \simeq_A a'} p'$, $\alpha \simeq_{p \simeq_{a \simeq_A a'} p'} \alpha'$, etc.

What properties should such a "globular type family" satisfy?

Intuitively, the key properties of equality are:

1. Equality is preserved by every construction of the language.
2. Equality forms a (higher-dimensional) equivalence relation.

The first property, congruence, may be schematically rendered as:

$$\frac{\Gamma, x : A \vdash t(x) : B \qquad \Gamma \vdash a^* : a \simeq_A a'}{\Gamma \vdash \text{“}t(a^*)\text{”} : t(a) \simeq_B t(a')}$$

Note that in presence of type dependency, even the statement of the above rule becomes hardly trivial, since the type $B = B(x)$ of $t(x)$ might itself depend on x.

The second property states that $\simeq_A$ is reflexive, symmetric, transitive on every "level", and that adjacent levels interact correctly. Altogether, these data may be captured neatly by the *Kan filling condition* from homotopy theory, stating that the above globular structure of equalities forms a weak ω-groupoid.

All of the needed properties above are encapsulated in five axioms isolated by Coquand [5]. Accordingly, the type $a \simeq_A a'$ must admit the following operations:

$(a : A)$	$\mathtt{r}(a)$	:	$a \simeq_A a$
$(x : A \vdash B(x) : \mathbf{Type})$	$\mathtt{transp}$	:	$B(a) \to (a \simeq_A a') \to B(a')$
$(b : B(a))$	$\mathtt{Jcomp}$	:	$\mathtt{transp}\, b\, \mathtt{r}(a) \simeq_{B(a)} b$
$(a : A)$	π_a	:	$\mathtt{isContr}(\Sigma x{:}A. a \simeq_A x)$
$(A, B(x), f, g)$	$\mathtt{FE}$	:	$(\Pi x{:}A)\;\; fx \simeq_{B(x)} gx \;\;\to\;\; f \simeq_{\Pi x:A.B(x)} g$

And now we have obtained a well-defined mathematical problem:
To define an extension of type theory with a new type $a \simeq_A a'$, so that all of Coquand's axioms are satisfied, and the usual metatheoretic properties of type theory remain valid.

Equality from logical relations

The point of departure of all syntactic approaches to extensional equality is to define an equivalence relation on the term model of type theory by induction, where the relation associated to each type constructor is given inductively by the logical condition for that type.

The *logical relation principle* then allows one to derive that every term of type theory preserves this relation, so that the relation is indeed a congruence.

As a simple example, let us consider the universe of simple types.

$$\begin{aligned}&\mathsf{Inductive}\ \mathcal{U} : * \ := \\ &\qquad\qquad\mid \circledcirc : \mathcal{U} \\ &\qquad\qquad\mid \ominus : \mathcal{U} \to \mathcal{U} \to \mathcal{U} \\ &\qquad\qquad\mid \otimes : \mathcal{U} \to \mathcal{U} \to \mathcal{U}\end{aligned}$$

To each element $A : \mathcal{U}$, we can associate the type of *terms of type* A, by defining a $*$-valued *decoding function* $\mathcal{T} : \mathcal{U} \to *$ by recursion over $\mathcal{U}$:

$$\begin{aligned}\mathcal{T}(\circledcirc) &= N \\ \mathcal{T}(\ominus AB) &= \mathcal{T}A \to \mathcal{T}B \\ \mathcal{T}(\otimes AB) &= \mathcal{T}A \times \mathcal{T}B\end{aligned}$$

We think of $\mathcal{T}A$ as giving the interpretation of object types $A : \mathcal{U}$ on the metalevel.

In just the same way, we may define a family of relations $\{R_A : \mathcal{T}A \to \mathcal{T}A \to *\}$ by recursion over $A : \mathcal{U}$:

$$\begin{aligned}R_{\circledcirc} mn &= \mathcal{Id}_N(m,n) \\ R_{\ominus AB} ff' &= \Pi x{:}\mathcal{T}A \Pi x'{:}\mathcal{T}A.\ R_A xx' \to R_B(fx)(f'x') \\ R_{\otimes AB} pp' &= R_A(\pi_1 p)(\pi_1 p') \ \times \ R_B(\pi_2 p)(\pi_2 p')\end{aligned}$$

By induction on A, we can verify that R_A is indeed an equivalence relation. Moreover, in contrast to the $\mathcal{Id}_A$-relation, R_A actually validates the rule of function extensionality – pretty much just by definition!

Thus we are confronted with a possibility that the intensional character of type theory may be quashed simply by declaring the logical equivalence relation to be the “right” notion of equality, and the $\mathcal{Id}$-type to be no good, ugly, bad.

All that remains is to extend the previous construction to dependent type theory, and make extensional equality part of our language.

Alas, while the logical relations argument can indeed be extended to the dependent setting – and we will treat this procedure in some detail – the additional requirement is oxymoronic as stated.

Logical relations are an external concept – they are valued in types of the meta-level. Their very construction requires the language to be a “closed” object, since it takes place in a metatheory where we have an interpretation of the given system $(\mathcal{U}, \mathcal{T})$ of types.

One might imagine some kind of a reflection procedure, whereby the values of the relation are reincarnated as elements of $\mathcal{U}$ again.

But what should that mean, exactly? If we ever add new types into $\mathcal{U}$, we change the domain of the relation being reflected. Is there a universe $\mathcal{U}$ which is “closed under” its own logical relation?

Among our contributions here is to clarify what indeed it could all mean, how one might go about constructing such a $\mathcal{U}$, and what one can do with the result. Toward the end of the paper, we produce a candidate universe containing the extensional equality relation for all *closed* types. Verification of the logical relation principle (congruence) for this universe will be treated in subsequent work.

Taking the long view, the promise of a syntactic foundation of extensional equality will be fulfilled whenever the following objectives are attained.

1. Find a type theory in which the universe $\mathcal{U}$ of types reflects the canonical logical relation defined by induction on type structure.
2. Prove extensionality theorem, that every term preserves this relation. Conclude that every ground term is equal to itself (reflexivity). Obtain extensional equality as the relation induced by the reflexive equality of a type with itself.
3. Extend the reflection mechanism to open terms.
4. Add operators to witness the Kan filling condition.
5. Treat the remaining axioms of Coquand.

Contents of the paper

Here we will deliver on the first goal in the above sequence. Starting from the simply typed lambda calculus, we proceed in a systematic way to extend our theory until we arrive at a universe which already contains the extensional relation inside its type hierarchy.

The paper can thus be read as a kind of "bootstrapping procedure" for the plan above. We show from first principles, how to define, by following a set pattern, a minimal type system containing the extensional equality type for all closed types. Further properties of extensional equality, including extension to open terms, can be built up using this system as a base.

An intermediate stage in our development is the treatment of logical relations for dependent types. For this purpose, we choose to work with the inconsistent pure type system $\lambda*$. In our view, this is by far and away the simplest formulation of dependent type theory, and tuning out the "universe management noise" allows us to see more clearly the computational relationships between the types and extensional equalities. [1]

After finding the candidate universe, we apply the usual method of stratification to remove the inconsistency inherited from $\lambda*$. We conjecture that the stratified system is strongly normalizing.

One can also read this paper without regard to the program above, as presenting a generalization of the extensionality theorem of Tait [9] to the dependent setting.

While logical relations for dependent types have been treated by various authors ([2], [4], [8]), our presentation contains some novel features. Specifically, we use a generalization of Dybjer's indexed inductive-recursive definitions to encapsulate the data pertaining to (heterogeneous) dependent relations associated to equalities between types. Our treatment of universes also differs from e.g. parametricity theory in that we instantiate the relation on the universe with the least congruence between types.

In particular, in our approach, every relation between types is necessarily an equivalence in the sense of homotopy type theory.

2 The simply typed case

We begin by stating the extensionality theorem for the simply typed λ-calculus.

The syntax of simple types and typed terms is as follows:

$$\mathbb{T} = o \mid \mathbb{T} \to \mathbb{T} \mid \mathbb{T} \times \mathbb{T}$$
$$\Lambda = x \mid \lambda x{:}\mathbb{T}.\Lambda \mid \Lambda\Lambda \mid (\Lambda, \Lambda) \mid \pi_1\Lambda \mid \pi_2\Lambda$$

[1] It was also attractive to work in a system where types and terms are treated completely homogeneously, if only as a "sanity check" that we are not making ad hoc definitions based on the "kind" of the object.

A model of $\lambda_{\rightarrow}$ consists of a family of sets $\{X_A \mid A \in \mathbb{T}\}$ where

$$X_{A\rightarrow B} \subseteq X_A^{X_B}$$
$$X_{A\times B} \subseteq X_A \times X_B$$

are such that $X_{A\rightarrow B}$ is closed under abstraction of terms of type B over variables of type A, and $X_{A\times B}$ is closed under pairs of definable elements of X_A and X_B.

The interpretation of types is given by

$$[\![A]\!] = X_A$$

The interpretation of terms is parametrized by an environment $\rho = \{\rho_A : V_A \rightarrow X_A\}$, assigning elements of the domain to the free variables of the term.

Let Env be the set of such collections of functions.

A term $t : A$ is interpreted as a map $[\![t]\!] : \mathsf{Env} \rightarrow [\![A]\!]$. We write $[\![t]\!]_\rho$ for $[\![t]\!](\rho)$. The definition of $[\![t]\!]_\rho$ is given by induction:

$$\begin{aligned}
[\![x : A]\!]_\rho &= \rho_A(x)\\
[\![st]\!]_\rho &= [\![s]\!]_\rho [\![t]\!]_\rho\\
[\![\lambda x{:}A.t]\!]_\rho &= (a \mapsto [\![t]\!]_{\rho, x:=a})\\
[\![(s,t)]\!]_\rho &= ([\![s]\!]_\rho, [\![t]\!]_\rho)\\
[\![\pi_i t]\!]_\rho &= a_i, \text{ where } [\![t]\!]_\rho = (a_1, a_2) \in [\![A_1 \times A_2]\!]
\end{aligned}$$

A relation $R = \{R_A \mid R_A \subseteq [\![A]\!] \times [\![A]\!], A \in \mathbb{T}\}$ is said to be *logical* if

$$\begin{aligned}
R_{A\rightarrow B} f f' &\iff \forall a a'{:}X_A . R_A a a' \Rightarrow R_B (fa)(f'a')\\
R_{A\times B}(a,b)(a',b') &\iff R_A a a' \wedge R_B b b'
\end{aligned}$$

Here and throughout, (two-sided) double arrows represent logical implication (equivalence) on the meta-level.

▶ **Theorem 1** (Extensionality Theorem). *Let R be logical. Suppose that t is a typed term:*

$$x_1 : A_1, \ldots, x_n : A_n \vdash t : T$$

and let there be given

$$a_1, a_1' \in [\![A_1]\!], \ldots, a_n, a_n' \in [\![A_n]\!]$$

Then

$$R_{A_1} a_1 a_1', \ldots, R_{A_n} a_n a_n' \implies R_T [\![t]\!]_{\vec{x}:=\vec{a}} [\![t]\!]_{\vec{x}:=\vec{a}'}$$

In other words, every typed λ-term induces a function which maps related elements to related elements. As a corollary, we get that a closed term $t \in \Lambda^0(A)$ is R_A-related to itself.

The proof of the above theorem proceeds by induction on the structure of derivation that $t : T$. For illustration, we treat the abstraction case $t = \lambda x{:}A.t'$. Suppose we have

$$\frac{x_1 : A_1, \ldots, x_n : A_n, x : A \vdash t' : B}{x_1 : A_1, \ldots, x_n : A_n \vdash \lambda x{:}A.t' : A \rightarrow B}\ \text{Abs}$$

Let $(a_1,\dots,a_n),(a'_1,\dots,a'_n) \in [\![A_1]\!] \times \dots \times [\![A_n]\!]$ be such that $R_{A_i} a_i a'_i$. Assume $a, a' \in [\![A]\!]$ are given, and suppose that $R_A a a'$.

By induction hypothesis,

$$R_B [\![t']\!]_{x_1\dots x_n x := a_1 \dots a_n a} [\![t']\!]_{x_1\dots x_n x := a'_1 \dots a'_n a'}$$

which can be rewritten as

$$R_B [\![\lambda x.t']\!]_{x_1\dots x_n := a_1 \dots a_n}(a) [\![\lambda x.t']\!]_{x_1\dots x_n := a'_1 \dots a'_n}(a')$$

Since a, a' were arbitrary, and $R_{A\to B}$ is logical, it follows that

$$R_{A\to B} [\![\lambda x.t']\!]_{x_1\dots x_n := a_1 \dots a_n} [\![\lambda x.t']\!]_{x_1\dots x_n := a'_1 \dots a'_n}$$

The other cases are treated similarly.

We remark that the structure of the proof that $R_T tt$ recapitulates rather precisely the structure of t itself. In particular, the theorem is completely constructive. Anticipating upcoming development, consider a constructive reading of the theorem's statement:

From a proof a_1^* that $R_{A_1} a_1 a'_1$
  and a proof a_2^* that $R_{A_2} a_2 a'_2$
  ...
  and a proof a_n^* that $R_{A_n} a_n a'_n$
Get a proof $t(a_1^*,\dots,a_n^*)$
  that $R_T t(a_1,\dots,a_n) t(a'_1,\dots,a'_n)$

This motivates us to think of the above extensionality property as an operation which, given terms which relate elements in the context, substitutes these connections into t to get a relation between the corresponding instances of t.

In this interpretation, the proof that a closed term t is related to itself

$$\mathsf{r}(t) : R_T tt$$

has specific computational content. Furthermore, the algorithm associated to this proof has the same structure as t itself.

Given a relation R_o on X_o, we can extend it to the full structure by *defining* $R_{A\to B}$, $R_{A\times B}$ to be such as to satisfy the logical conditions; the resulting family then satisfies the theorem by construction.

3 The dependent case

To make matters simple, we work with the pure type system (PTS) formulation of dependent type theory with "type-in-type" extended by Σ-types. This system is denoted as $\lambda *$. It has a universal type $*$, the type of all types. This allows us to unify into one the three classical judgement forms of dependent type theory:

$$\Gamma \vdash A \ \ \mathbf{Type}$$
$$\Gamma \vdash a : A$$
$$\Gamma \vdash B : (A)\mathbf{Type}$$

The judgment $\Gamma \vdash A$ **Type** is replaced by $\Gamma \vdash A : *$. Similarly, $\Gamma \vdash (A)B$ **Type** is replaced by $\Gamma, x{:}A \vdash B : *$. Thus types and terms of type $*$ are completely identified.

The syntax of $\lambda *$ is given in Figure 1.

$$A, t \quad ::= \quad * \mid x \mid \Pi x{:}A.B \mid \Sigma x{:}A.B \mid \lambda x{:}A.t \mid st \mid (s,t) \mid \pi_1 t \mid \pi_2 t$$

$$\frac{}{\vdash * : *}$$

$$\frac{\Gamma \vdash A : *}{\Gamma, x : A \vdash x : A}$$

$$\frac{\Gamma \vdash M : A \qquad \Gamma \vdash B : *}{\Gamma, y : B \vdash M : A}$$

$$\frac{\Gamma \vdash A : * \qquad \Gamma, x : A \vdash B : *}{\begin{array}{c}\Gamma \vdash \Pi x{:}A.B : * \\ \Gamma \vdash \Sigma x{:}A.B : *\end{array}}$$

$$\frac{\Gamma \vdash A : * \qquad \Gamma, x : A \vdash B : * \qquad \Gamma, x : A \vdash b : B}{\Gamma \vdash \lambda x{:}A.b : \Pi x{:}A.B}$$

$$\frac{\Gamma \vdash A : * \quad \Gamma, x : A \vdash B : * \qquad \Gamma \vdash f : \Pi x{:}A.B \qquad \Gamma \vdash a : A}{\Gamma \vdash fa : B[a/x]}$$

$$\frac{\Gamma \vdash A : * \quad \Gamma, x : A \vdash B : * \qquad \Gamma \vdash a : A \qquad \Gamma \vdash b : B[a/x]}{\Gamma \vdash (a,b) : \Sigma x{:}A.B}$$

$$\frac{\Gamma \vdash A : * \qquad \Gamma, x : A \vdash B : * \qquad \Gamma \vdash p : \Sigma x{:}A.B}{\begin{array}{c}\Gamma \vdash \pi_1 p : A \\ \Gamma \vdash \pi_2 p : B[\pi_1 p/x]\end{array}}$$

$$\frac{\Gamma \vdash M : A \qquad \Gamma \vdash B : * \qquad A = B}{\Gamma \vdash M : B}$$

$$\begin{array}{rcl}(\lambda x{:}A.s)t & \longrightarrow & s[t/x] \\ \pi_1(s,t) & \longrightarrow & s \\ \pi_2(s,t) & \longrightarrow & t\end{array}$$

Figure 1 The system $\lambda *$.

Our choice of rules differs from the standard definition of $\lambda *$ as a PTS in that the introduction and elimination rules (for both Π- and Σ-types) come with information on the well-formedness of the type arguments A, B. This however does not change the set of derivable sequents.
The proof of this essentially appears already in Barendregt [3], see Corollary 5.2.14(1,2) and Lemma 5.2.13(3). The argument there is equally applicable to Σ as to Π. (And in $\lambda *$, it becomes even easier.)

Notation. The parentheses following the matrix of the Π, Σ, and λ constructors are not part of the syntax, and merely pronounce the fact that the term may depend on the variables in question. In general, when we write $t = t(x_1, \dots, x_n)$, we do not commit to having displayed all the free variables of t; it is never mandatory to display a free variable.

The purpose of this notation is merely to reduce clutter in anticipation of substitution of t by an instance of (some of the) variables. Our general notation for capture-avoiding substitution of a free variable x in t by a term a is

$$t[a/x]$$

In particular, if $t = t(x_1, \dots, x_n)$, then

$$t[a_1/x_1]\cdots[a_n/x_n] = t(a_1, \dots, a_n)$$

In the following development, we shall consider the open term model of the above type theory, using the same theory as our meta-level. To simplify notation, we write $[\![A]\!]$ simply as A. As well, if $t(x_1, \dots, x_n) : T(x_1, \dots, x_n)$, then $[\![t]\!]_{x_1,\dots,x_n:=a_1,\dots,a_n}$ is denoted as $t(a_1, \dots, a_n)$.

From now on, we work in $\lambda*$.

The only axiom of this type system has the form $* : *$, asserting that the universe of types $*$ is itself a type. Its intuitive meaning is

> The collection of structures, which types are interpreted by, forms the same kind of structure.

In particular, if types are interpreted by types-with-relation $R_A : A \to A \to *$, then this interpretation must also include a relation on the universe of types

$$R_* : * \to * \to *$$

But how should this relation interact with objects inhabiting related types? Is there a logical condition for the universe? What is a dependent logical relation?

To answer these questions, we engage in a series of thought experiments.

First, we stipulate that, dependent logical relation, whatever it finally turns out to be, must at the very least be such that the congruence (extensionality) rule is always valid.

Let us then consider how the previous extensionality theorem could be extended to the dependent case. Suppose we have terms

$$x{:}A \vdash B(x) : * \tag{3}$$

$$x{:}A \vdash b(x) : B(x) \tag{4}$$

If we are now given $a : A$, $a' : A$, we want to conclude that

$$R_A a a' \to R_B b(a) b(a') \tag{5}$$

However, the two terms $b(a)$ and $b(a')$ have different types! Evidently, we need more structure to formulate extensionality of dependent maps. But where should this structure come from?

Looking again at (3), let us first consider extensionality of the term $B(x)$:
From any witness a^* of the hypothesis of (5), it must be possible to construct a witness $B(a^*)$ to the relation $R_* B(a) B(a')$.

Thinking of $B(a^*)$ as encoding some kind of correspondence between types, we can imagine the relation $b(a^*)$ between $b(a)$ and $b(a')$ to be "lying over" the relation $B(a^*)$ between $B(a)$ and $B(a')$.

This suggests the following principle:

Every witness $e : R_* AB$ to the fact that A and B are related elements of the universe induces a relation

$$\sim e : A \to B \to *$$

between elements of corresponding types.

Notation. Put

$$\begin{array}{rcll} a \simeq_A a' & := & R_A a a' & a, a' : A \\ A \simeq B & := & R_* AB & A, B : * \\ a \sim_e b & := & \sim eab & a{:}A, b{:}B, e : A \simeq B \end{array}$$

We call $e : A \simeq B$ a *type equality* between A and B, and $a \sim_e b$ a *dependent equality*, or *heterogeneous equality induced by* e.

Notice that $(A \simeq B) = (A \simeq_* B)$.

The considerations thus far have yielded that

$$\frac{x{:}A \vdash B(x) : * \qquad \vdash a^* : a \simeq_A a'}{B(a^*) : B(a) \simeq B(a')} \qquad \frac{x{:}A \vdash b(x) : B(x) \qquad \vdash a^* : a \simeq_A a'}{b(a^*) : b(a) \sim_{B(a^*)} b(a')}$$

Now suppose $x \notin B(x) = B$. Then, for $a^* : a \simeq_A a'$, we have

$$B() = B(a^*) : B(a) \simeq B(a')$$

$$B() : B \simeq B$$

So $B(a^*)$, for $x \notin B(x)$, gives us a type equality of B with itself. We call such equality the *identity on* B; it corresponds to the identity equivalence in homotopy type theory.

The relation $\sim_{B()}: B \to B \to *$ induced by the identity equivalence we call the *extensional equality on type* B, and we define

$$a \simeq_A a' \quad := \quad a \sim_{A()} a'$$

In particular, since $x \notin * : *$, we have

$$*() : * \simeq *$$

$$*() : * \sim_{*()} *$$

$$A \simeq_* B \quad = \quad A \sim_{*()} B \quad = \quad A \simeq A$$

so that extensional equality on the universe is type equality and is the relation induced by the identity equivalence of the universe with itself.

With these definitions, we can make sense of the "logical conditions"

$$R_{\Pi x:A.B(x)} f f' = \prod_{a:A} \prod_{a':A} \Pi a^* : R_A a a'. (fa) \sim_{B(a^*)} (f'a') \tag{6}$$

$$R_{\Sigma x:A.B(x)} pp' = \Sigma a^* : R_A\,(\pi_1 p)\,(\pi_1 p'). (\pi_2 p) \sim_{B(a^*)} (\pi_2 p') \tag{7}$$

However, as the considerations illustrate, we may also just dispense with the relation $(R_A) = (\simeq_A)$ altogether and assume as primitive only the following two objects:

- A binary relation on the universe:

$$\frac{A : * \qquad B : *}{A \simeq B : *}$$

- For all types A, B, and for every proof $e : A \simeq B$ that they are related, a binary relation between A and B:

$$\frac{e : A \simeq B \qquad a : A \qquad b : B}{a \sim_e b : *}$$

The question of what it means for a pair $(\simeq, \sim_e)$ to be logical will be answered in Section 5. It will allow us to prove the generalization of extensionality theorem of the following shape:

▶ **Theorem 2.** *Let $(\simeq, \sim_e)$ be logical. For every term t typed in the context*

$$x_1 : A_1, \cdots, x_n : A_n(x_1, \ldots, x_{n-1}) \vdash t(x_1, \ldots, x_n) : T(x_1, \ldots, x_n)$$

and for any sequence of coordinate-wise related instances

$$a_1 : A_1, \cdots, a_n : A_n(a_1, \ldots, a_{n-1})$$
$$a'_1 : A_1, \cdots, a'_n : A_n(a'_1, \ldots, a'_{n-1})$$
$$a^*_1 : a_1 \sim_{A_1()} a'_1, \cdots, a^*_n : a_n \sim_{A_n(a^*_1, \ldots, a^*_{n-1})} a'_n$$

there is a witness $t(a^*_1, \ldots, a^*_n)$ to the fact that

$$t(a_1, \ldots, a_n) \sim_{T(a^*_1, \ldots, a^*_n)} t(a'_1, \ldots, a'_n).$$

As it appears now, the statement is perhaps unparsable, since the conclusion already makes reference to the result of substituting $a^*_1, \ldots, a^*_n$ into $T(\vec{x})$, which requires the extensionality of the judgement $\Gamma \vdash T(\vec{x}) : *$ to be known beforehand. The proof of this latter fact will again depend on extensionality of subterms appearing in T.

Also, the definitions of $\sim$ and $\simeq$ themselves, will evidently depend on being able to perform the "cell substitution" given by the theorem (as in the case of the equality for Π- and Σ-types).

In order to cut through all this circularity, and to formulate all the necessary concepts precisely, we first move to represent the type universe of $\lambda*$ in a minimal extension of the system relevant for this purpose. The above theorem will be stated for the result of reflecting the meta-level into this universe. The next step is to mutually define the type of equivalences between two elements of this universe, and the corresponding relations induced by such equivalences. The inter-dependency between these concepts is resolved using an indexed inductive–recursive definition of Dybjer and Setzer [6], and this allows us to state the above theorem for the (reflected) universe. Finally, we prove the theorem by induction on the structure of derivations.

4 $\lambda*$ in $\lambda*$

To motivate the inductive–recursive definition of the universe of $\lambda*$, let us recall the representation of simply typed lambda calculus.

```
Inductive 𝒰 : * :=
        | ⊚ : 𝒰
        | ⊖ : 𝒰 → 𝒰 → 𝒰
        | ⊗ : 𝒰 → 𝒰 → 𝒰
```

$$\begin{aligned}\mathcal{T} : \mathcal{U} &\to * \\ \mathcal{T}(\circledcirc) &= A_0 \\ \mathcal{T}(\ominus AB) &= \mathcal{T}A \to \mathcal{T}B \\ \mathcal{T}(\otimes AB) &= \mathcal{T}A \times \mathcal{T}B\end{aligned}$$

It is straightforward to define, for every simple type $A \in \mathbb{T}$, a corresponding term $\overline{A}$ of type $\mathcal{U}$. Furthermore, for every valid $\lambda_{\to\times}$ judgment $\Gamma \vdash M : A$, there is a term $\overline{M}$ of type $\mathcal{T}\overline{A}$ and a derivation of $\overline{\Gamma} \vdash \overline{M} : \mathcal{T}\overline{A}$, where contexts are translated as

$$\overline{x_1 : A_1, \cdots, x_n : A_n} = x_1 : \mathcal{T}\overline{A_1}, \cdots, x_n : \mathcal{T}\overline{A_n}$$

Next, consider the family R_A, for $A : \mathcal{U}$, defined on page 223. Clearly, this family gives a logical relation on the interpretation of $\lambda_{\to\times}$ in $(\mathcal{U}, \mathcal{T})$. Thus we may verify the following

▶ **Proposition 3.** *Suppose $x_1 : A_1, \ldots, x_n : A_n \vdash_{\lambda_{\to\times}} M : A$.*
Consider the context consisting of variables $x_1, x_1' : T\overline{A_1}, \ldots, x_n, x_n' : T\overline{A_n}$ and

$$x_1^* : R_{A_1} x_1 x_1', \ldots, x_n^* : R_{A_n} x_n x_n',$$

There is a term $M^* = M^*(x_1, x_1', x_1^*, \ldots, x_n, x_n', x_n^*)$ such that

$$\{x_1, x_1', x_1^*, \ldots, x_n, x_n', x_n^*\} \vdash M^* : R_A\, \overline{M}\, \overline{M'}$$

where $M' = M[x_1'/x_1, \ldots, x_n'/x_n]$.

(The proof proceeds exactly as before, replacing set-theoretic concepts by their type-theoretic counterparts.)

In trying to repeat the above proposition for dependent type theory, we run into a problem already when trying to represent the universe of types. As becomes evident, the collection of types must be defined *simultaneously* with the decoding function:

$$\begin{aligned}&\mathsf{Inductive}\ \mathcal{U} : * \ := \\ &\qquad \mid \circledcirc : \mathcal{U} \\ &\qquad \mid \textcircled{Π} : \Pi A : \mathcal{U}.(\mathcal{T}A \to \mathcal{U}) \to \mathcal{U} \\ &\qquad \mid \textcircled{Σ} : \Pi A : \mathcal{U}.(\mathcal{T}A \to \mathcal{U}) \to \mathcal{U} \\ &\ \ \mathsf{with}\ \ \mathcal{T} : \mathcal{U} \to * \ := \\ &\qquad \mathcal{T}(\circledcirc) = A_0 \\ &\qquad \mathcal{T}(\textcircled{Π}AB) = \Pi a : \mathcal{T}A.\mathcal{T}[Ba] \\ &\qquad \mathcal{T}(\textcircled{Σ}AB) = \Sigma a : \mathcal{T}A.\mathcal{T}[Ba]\end{aligned}$$

The full theory of such inductive-recursive definitions, together with a model construction, is treated in detail by Dybjer and Setzer [6]. Ghani et al. [7] give a modern presentation, considerably generalizing the concept.

Reflection of λ∗

The inductive-recursive definition of the universe $\mathcal{U}$ of λ∗-types is as follows:

$$
\begin{aligned}
&\mathsf{Inductive}\ \mathcal{U} : * \ :=\\
&\qquad \mid \textcircled{Π} : \Pi A : \mathcal{U}.(\mathcal{T}A \to \mathcal{U}) \to \mathcal{U}\\
&\qquad \mid \textcircled{Σ} : \Pi A : \mathcal{U}.(\mathcal{T}A \to \mathcal{U}) \to \mathcal{U}\\
&\qquad \mid \circledast : \mathcal{U}\\
&\quad \mathsf{with}\ \ \mathcal{T} : \mathcal{U} \to * \ :=\\
&\qquad \mathcal{T}(\textcircled{Π}AB) = \Pi a : \mathcal{T}A.\mathcal{T}[Ba]\\
&\qquad \mathcal{T}(\textcircled{Σ}AB) = \Sigma a : \mathcal{T}A.\mathcal{T}[Ba]\\
&\qquad \mathcal{T}(\circledast) = \mathcal{U}
\end{aligned}
$$

Let $\lambda{*}\mathcal{U}$ be λ∗ augmented with the above datatype. Notice that every derivation in λ∗ is also a derivation in $\lambda{*}\mathcal{U}$.

▶ **Definition 4.** We define a map $\overline{(\cdot)}$ from the raw terms of λ∗ to the raw terms of $\lambda{*}\mathcal{U}$ as follows:

$$
\begin{aligned}
\overline{*} &= \circledast\\
\overline{x} &= x\\
\overline{\Pi x{:}A.B} &= \textcircled{Π}\overline{A}(\lambda x{:}\mathcal{T}\overline{A}.\overline{B})\\
\overline{\Sigma x{:}A.B} &= \textcircled{Σ}\overline{A}(\lambda x{:}\mathcal{T}\overline{A}.\overline{B})\\
\overline{\lambda x{:}A.t} &= \lambda x{:}\overline{A}.\overline{t}\\
\overline{st} &= \overline{s}\,\overline{t}\\
\overline{(s,t)} &= (\overline{s},\overline{t})\\
\overline{\pi_i s} &= \pi_i\overline{s}
\end{aligned}
$$

▶ **Definition 5.** Let $\Gamma = \{x_1 : A_1, \dots, x_n : A_n\}$ be a context in λ∗. We define[2]

$$\overline{\Gamma} := \{x_1 : \mathcal{T}\overline{A_1}, \dots, x_n : \mathcal{T}\overline{A_n}\}$$

▶ **Lemma 6** (Substitution Lemma)**.** *Let M, N be λ∗-terms. Then*

$$\overline{M[N/x]} = \overline{M}[\overline{N}/x]$$

Proof. This is manifest from the fact that $\overline{\cdot}$ is defined completely compositionally. ◀

▶ **Corollary 7.** *Let M, N be λ∗-terms. Then*

$$M = N \quad\Longrightarrow\quad \overline{M} = \overline{N}$$

[2] Note that we cannot yet conclude that this definition yields a valid context: so far the $\overline{A_i}$ are just raw terms, and we have not checked that

$$x_1 : \mathcal{T}\overline{A_1}, \dots, x_i : \mathcal{T}\overline{A_i} \vdash \overline{A_{i+1}} : \mathcal{U}$$

for $0 \le i < n$. As a matter of fact, this will follow from the theorem we are about to prove, but for now we just treat $\overline{\Gamma}$ as a "raw context".
(Indeed, the typing rules of λ∗, like all PTSs, do not include context hygiene, because it is enforced implicitly via the hypotheses of context-extending rules.)

▶ **Theorem 8** (Reflection of $*$ into $\mathcal{U}$)**.**

$$\Gamma \vdash_{\lambda *} M : A \implies \overline{\Gamma} \vdash_{\lambda * \mathcal{U}} \overline{M} : \mathcal{T}\overline{A}$$

Proof. The translation is done by induction on $\Gamma \vdash M : A$.
Axiom. $\vdash_{\lambda *} * : *$. Then $\overline{\Gamma} = \Gamma = \langle\rangle$. Also $\overline{A} = \overline{M} = \circledast$. The conversion rule gives

$$\frac{\circledast : \mathcal{U} \qquad \mathcal{T}\circledast : * \qquad \mathcal{U} = \mathcal{T}\circledast}{\circledast : \mathcal{T}\circledast}$$

Thus indeed $\vdash_{\mathcal{U}} \overline{M} : \mathcal{T}\overline{A}$.
Variable. Suppose the derivation ends with

$$\frac{\Gamma \vdash A : *}{\Gamma, x : A \vdash x : A}$$

By induction hypothesis, we have

$$\overline{\Gamma} \vdash \overline{A} : \mathcal{T}\overline{*}$$

Since $\overline{*} = \circledast$, we have $\Gamma \vdash \overline{A} : \mathcal{T}\circledast$. Then $\overline{\Gamma} \vdash \overline{A} : \mathcal{U}$, and $\overline{\Gamma} \vdash \mathcal{T}\overline{A} : *$.
By the variable rule, we have

$$\overline{\Gamma}, x : \mathcal{T}\overline{A} \vdash x : \mathcal{T}\overline{A}$$

Weakening. Let the derivation end with

$$\frac{\Gamma \vdash M : A \qquad \Gamma \vdash B : *}{\Gamma, y : B \vdash M : A}$$

By induction hypothesis, we have

$$\overline{\Gamma} \vdash \overline{M} : \mathcal{T}\overline{A}$$
$$\overline{\Gamma} \vdash \overline{B} : \mathcal{T}\circledast$$

That is, $\overline{\Gamma}$ yields $\vdash \overline{B} : \mathcal{U}$. Then $\mathcal{T}\overline{B} : *$. By weakening,

$$\overline{\Gamma}, y : \mathcal{T}\overline{B} \vdash \overline{M} : \mathcal{T}\overline{A}$$

Π-formation. Given

$$\frac{\Gamma \vdash A : * \qquad \Gamma, x : A \vdash B : *}{\Gamma \vdash \Pi x{:}A.B : *}$$

the induction hypotheses yield

$$\overline{\Gamma} \vdash \overline{A} : \mathcal{T}\overline{*} \tag{8}$$
$$\overline{\Gamma}, x : \mathcal{T}\overline{A} \vdash \overline{B} : \mathcal{T}\overline{*} \tag{9}$$

Since $\overline{A}, \overline{B} : \mathcal{T}\overline{*} = \mathcal{T}\circledast = \mathcal{U}$, we have $\overline{\Gamma} \vdash \mathcal{T}\overline{A} : *$ as well as $\Gamma, x : \mathcal{T}\overline{A} \vdash \mathcal{T}\overline{B} : *$.
By the Π-introduction rule, (9) yields

$$\overline{\Gamma} \vdash \lambda x{:}\mathcal{T}\overline{A}.\overline{B} : \mathcal{T}\overline{A} \to \mathcal{U}$$

whence Π-elimination together with (8) yields

$$\overline{\Gamma} \vdash \textcircled{\scriptsize\Pi}\overline{A}(\lambda x{:}\mathcal{T}\overline{A}.\overline{B}) : \mathcal{U}$$

That is,

$$\overline{\Gamma} \vdash \overline{\Pi x{:}A.B} : \mathcal{T}\overline{*}$$

Σ-formation. Treated in an analogous fashion.
Π-introduction. Suppose the derivation is of the form

$$\frac{\Gamma \vdash A : * \qquad \Gamma, x : A \vdash B : * \qquad \Gamma, x : A \vdash b : B}{\Gamma \vdash \lambda x{:}A.b : \Pi x{:}A.B}$$

The induction hypotheses give us

$$\begin{aligned} \overline{\Gamma} &\vdash \overline{A} : \mathcal{T}* \\ \overline{\Gamma}, x : \mathcal{T}\overline{A} &\vdash \overline{B} : \mathcal{T}* \\ \overline{\Gamma}, x : \mathcal{T}\overline{A} &\vdash \overline{b} : \mathcal{T}\overline{B} \end{aligned} \tag{10}$$

As in the previous case, we actually have

$$\begin{aligned} \overline{\Gamma} &\vdash \overline{A} : \mathcal{U} & \overline{\Gamma} &\vdash \mathcal{T}\overline{A} : * \\ \overline{\Gamma}, x : \mathcal{T}\overline{A} &\vdash \overline{B} : \mathcal{U} & \overline{\Gamma}, x : \mathcal{T}\overline{A} &\vdash \mathcal{T}\overline{B} : * \\ \overline{\Gamma} &\vdash \overline{\Pi x{:}A.B} : \mathcal{U} & \overline{\Gamma} &\vdash \mathcal{T}[\overline{\Pi x{:}A.B}] : * \end{aligned}$$

By Π-introduction on (10), we have

$$\overline{\Gamma} \vdash \lambda x{:}\mathcal{T}\overline{A}.\overline{b} : \Pi x{:}\mathcal{T}\overline{A}.\mathcal{T}\overline{B}$$

But we also find that

$$\Pi x{:}\mathcal{T}\overline{A}.\mathcal{T}\overline{B} = \mathcal{T}[\textcircled{Π}\overline{A}(\lambda x{:}\mathcal{T}\overline{A}.\overline{B})] = \mathcal{T}\overline{\Pi x{:}A.B} \tag{11}$$

and so conclude that

$$\overline{\Gamma} \vdash \overline{\lambda x{:}A.b} : \mathcal{T}\overline{\Pi x{:}A.B}$$

Π-elimination. Suppose we are given

$$\frac{\Gamma \vdash A : * \quad \Gamma, x : A \vdash B : * \qquad \Gamma \vdash f : \Pi x{:}A.B \qquad \Gamma \vdash a : A}{\Gamma \vdash f a : B[a/x]}$$

The induction hypothesis yield, on the one hand, that

$$\begin{aligned} \overline{\Gamma} &\vdash \overline{A} : \mathcal{U} & \overline{\Gamma} &\vdash \mathcal{T}\overline{A} : * \\ \overline{\Gamma}, x : \mathcal{T}\overline{A} &\vdash \overline{B} : \mathcal{U} & \overline{\Gamma}, x : \mathcal{T}\overline{A} &\vdash \mathcal{T}\overline{B} : * \\ \overline{\Gamma} &\vdash \overline{\Pi x{:}A.B} : \mathcal{U} & \overline{\Gamma} &\vdash \mathcal{T}[\overline{\Pi x{:}A.B}] : * \end{aligned}$$

and on the other hand, that

$$\begin{aligned} &\overline{\Gamma} \vdash \overline{f} : \mathcal{T}\overline{\Pi x{:}A.B} \\ &\overline{\Gamma} \vdash \overline{a} : \mathcal{T}\overline{A} \end{aligned}$$

Since $\overline{f}$ by conversion in (11) has type $\Pi x{:}\mathcal{T}\overline{A}.\mathcal{T}\overline{B}$, we may write

$$\overline{\Gamma} \vdash \overline{f}\overline{a} : \mathcal{T}\overline{B}[\overline{a}/x]$$

By Lemma 6, the type in the above judgment is equal to $\mathcal{T}\overline{B[a/x]}$.
Σ-introduction. When we are at

$$\frac{\Gamma \vdash A : * \quad \Gamma, x : A \vdash B : * \qquad \Gamma \vdash a : A \qquad \Gamma \vdash b : B[a/x]}{\Gamma \vdash (a, b) : \Sigma x{:}A.B}$$

the induction hypotheses give as before that

$$\begin{aligned} \overline{\Gamma} &\vdash \mathcal{T}\overline{A} : * \\ \overline{\Gamma}, x : \mathcal{T}\overline{A} &\vdash \mathcal{T}\overline{B} : * \end{aligned}$$

and, in addition, we also have

$$\begin{aligned} &\overline{\Gamma} \vdash \overline{a} : \mathcal{T}\overline{A} \\ &\overline{\Gamma} \vdash \overline{b} : \mathcal{T}\overline{B[a/x]} \end{aligned}$$

Recall that

$$\begin{aligned} \overline{\Sigma x{:}A.B} &= \text{\textcircled{$\scriptstyle\Sigma$}}\overline{A}(\lambda x{:}\mathcal{T}\overline{A}.\overline{B}) \\ \mathcal{T}\overline{\Sigma x{:}A.B} &= \Sigma x{:}\mathcal{T}\overline{A}.\mathcal{T}\overline{B} \end{aligned}$$

By Lemma 6, $\mathcal{T}\overline{B[a/x]} = \mathcal{T}\overline{B}[\overline{a}/x]$. Hence $b : \mathcal{T}\overline{B}[\overline{a}/x]$.
By Σ-introduction, we now obtain

$$\overline{\Gamma} \vdash (\overline{a}, \overline{b}) : \Sigma x{:}\mathcal{T}\overline{A}.\mathcal{T}\overline{B}$$

In other words, $\overline{\Gamma} \vdash \overline{(a,b)} : \mathcal{T}\overline{\Sigma x{:}A.B}$.

Σ-elimination. Let there be derived

$$\frac{\Gamma \vdash A : * \qquad \Gamma, x : A \vdash B : * \qquad \Gamma \vdash p : \Sigma x{:}A.B}{\begin{array}{c}\Gamma \vdash \pi_1 p : A \\ \Gamma \vdash \pi_2 p : B[\pi_1 p/x]\end{array}}$$

Assume we have

$$\begin{aligned} \overline{\Gamma} &\vdash \overline{A} : \mathcal{U} \\ \overline{\Gamma}, x : \mathcal{T}\overline{A} &\vdash \overline{B} : \mathcal{U} \\ \overline{\Gamma} &\vdash \overline{p} : \mathcal{T}\overline{\Sigma x{:}A.B} \end{aligned}$$

We have just seen that $\mathcal{T}\overline{\Sigma x{:}A.B} = \Sigma x{:}\mathcal{T}\overline{A}.\mathcal{T}\overline{B}$. Thus

$$\pi_1\overline{p} : \mathcal{T}\overline{A}$$

$$\pi_2\overline{p} : [\mathcal{T}\overline{B}][\pi_1\overline{p}/x]$$

The subjects of these judgements can be rewritten as $\overline{\pi_i p}$.
Also

$$[\mathcal{T}\overline{B}][\pi_1\overline{p}/x] = [\mathcal{T}\overline{B}][\overline{\pi_1 p}/x] = \mathcal{T}[\overline{B}[\overline{\pi_1 p}/x]] = \mathcal{T}[\overline{B[\pi_1 p/x]}]$$

Thus we have

$$\begin{aligned} &\overline{\Gamma} \vdash \overline{\pi_1 p} : \mathcal{T}\overline{A} \\ &\overline{\Gamma} \vdash \overline{\pi_2 p} : \mathcal{T}\overline{B[\pi_1 p/x]} \end{aligned}$$

Conversion. Suppose we come across

$$\frac{\Gamma \vdash M : A \qquad \Gamma \vdash B : * \qquad A = B}{\Gamma \vdash M : B}$$

By induction hypothesis, we have

$$\overline{\Gamma} \vdash \overline{M} : \mathcal{T}\overline{A}$$
$$\overline{\Gamma} \vdash \overline{B} : \mathcal{U}$$

By Lemma 7, we have

$$\overline{A} = \overline{B}$$

But clearly that implies that

$$\mathcal{T}\overline{A} = \mathcal{T}\overline{B}$$

It is likewise clear that

$$\overline{\Gamma} \vdash \mathcal{T}\overline{B} : *$$

By the conversion rule, we comprehend

$$\overline{\Gamma} \vdash \overline{M} : \mathcal{T}\overline{B}$$

This completes the proof of the theorem. ◀

5 Extensionality of λ*

We work in $\lambda{*}\mathcal{U}$.

In order to precisely state extensionality theorem using the reflection of types into terms of type $\mathcal{U}$, we must give answers to the following questions:

- What does it mean for two types $A, B : \mathcal{U}$ to be equal?
- What does it mean for two elements to be related by a type equality?

To answer these questions, we continue our reasoning as in Section 3.

There we arrived at the signature

$$\simeq : \mathcal{U} \to \mathcal{U} \to *, \qquad \sim_e : \mathcal{T}A \to \mathcal{T}B \to * \quad \text{for all } e : A \simeq B$$

that a dependent logical relation should possess. But what should be the logical conditions associated to type constructors?

Well, keeping in mind the interpretation of $\simeq$ as type equality, we should at the very least require that $\simeq$ is a congruence with respect to type structure – in other words, that every type constructor preserves type equality.

For example, if we are given type equalities $A^* : A \simeq A'$, $B^* : B \simeq B'$, we should be able to construct terms

$$\times^* A^* B^* \quad : \quad (A \times B) \simeq (A' \times B') \tag{12}$$
$$\to^* A^* B^* \quad : \quad (A \to B) \simeq (A' \to B') \tag{13}$$

Notice that these inferences are valid regardless of what kind of semantics we have in mind for $\simeq$: isomorphism, weak equivalence, exact/strict equality, or even logical equivalence. But it should be a notion of equality, not an arbitrary relation on the universe.

It is likewise clear how the relations induced by (12) and (13) should be given in terms of those for A^* and B^*:

$$\begin{aligned}(a,b) \sim_{\times^* A^* B^*} (a',b') &= (a \sim_{A^*} a') \times (b \sim_{B^*} b') \\ f \sim_{\to^* A^* B^*} f' &= \prod_{x:A} \prod_{x':A'} x \sim_{A^*} x' \to fx \sim_{B^*} f'x'\end{aligned}$$

We invite the reader to the fun exercise of generalizing the above relation laws to the dependent case, to obtain the *dependent* logicality conditions for the Π- and Σ-types.

Concerning the type constructor $*$, being a nullary constructor, it must also preserve equality. Since it has no arguments, this simply means that it is equal to itself:

$$*^* : * \simeq *$$

The logical condition for the universe must produce, given $A, B : *$, a new type $A \sim_{*^*} B$. What should this type be?

The answers to these questions are displayed on the bottom of this page, where we define the dependent logical relation

$$(\mathcal{E}q(A,B) : *, \mathcal{R}el_{\{A,B\}}(e : \mathcal{E}q(A,B)) : \mathcal{T}A \to \mathcal{T}B \to *)$$

on the universe $(\mathcal{U}, \mathcal{T})$ of reflected $\lambda*$-types.

The key difficulty introduced by type dependency is that the definitions of $\mathcal{E}q(A,B)$ and $\mathcal{R}el_{\{A,B\}}(e)$ cannot be disentangled from one another: they must be defined simultaneously. This naturally suggests that their interdependency could be captured using a variant of the inductive–recursive (IR) definitions.

Upon reflecting on this possibility, it shall become manifest that the concepts cannot be defined uniformly in A and B either; rather, the A and B must take part in the recursive construction of both the set $\mathcal{E}q(A,B)$ as well as the map $\mathcal{R}el_{A,B}(e)$. Thus, the arguments A and B are to be treated as *indices*, so that we are dealing with an *indexed inductive–recursive definition* (IIRD).

We are now in the position to answer the questions posed above. The notion of equivalence of types A and B and the notion of elements of the corresponding types being related over an equivalence are both defined simultaneously by indexed induction–recursion. The definition appears below.

$$\begin{aligned}
&\textsf{Inductive}\ \mathcal{E}q : \mathcal{U} \to \mathcal{U} \to * \quad := \\
&\qquad \mid \mathsf{r}(\circledast) \quad : \quad \mathcal{E}q \circledast \circledast \\
&\qquad \mid \textcircled{Π}^* \{AA' : \mathcal{U}\}\{B : \mathcal{T}A \to \mathcal{U}\}\{B' : \mathcal{T}A' \to \mathcal{U}\} \\
&\qquad\qquad (A^* : \mathcal{E}q AA')(B^* : \Pi a{:}\mathcal{T}A \Pi a'{:}\mathcal{T}A' \Pi a^* : \mathcal{R}el\, A^* aa'. \mathcal{E}q(Ba)(B'a')) \\
&\qquad\qquad\qquad : \quad \mathcal{E}q(\textcircled{Π} AB)(\textcircled{Π} A'B') \\
&\qquad \mid \textcircled{Σ}^* \{AA' : \mathcal{U}\}\{B : \mathcal{T}A \to \mathcal{U}\}\{B' : \mathcal{T}A' \to \mathcal{U}\} \\
&\qquad\qquad (A^* : \mathcal{E}q AA')(B^* : \Pi a{:}\mathcal{T}A \Pi a'{:}\mathcal{T}A' \Pi a^* : \mathcal{R}el\, A^* aa'. \mathcal{E}q(Ba)(B'a')) \\
&\qquad\qquad\qquad : \quad \mathcal{E}q(\textcircled{Σ} AB)(\textcircled{Σ} A'B') \\
&\ \ \textsf{with}\quad \mathcal{R}el : \Pi\{A\}\{B\} : \mathcal{U}.\ \mathcal{E}q AB \to \mathcal{T}A \to \mathcal{T}B \to * \\
&\qquad \mathcal{R}el\,(\mathsf{r}(\circledast))AB \quad = \quad \mathcal{E}q AB \\
&\qquad \mathcal{R}el\,(\textcircled{Π}^* A^* B^*) f f' \quad = \quad \Pi x{:}\mathcal{T}A \Pi x'{:}\mathcal{T}A' \Pi x^* : \mathcal{R}el\, A^* xx'. \\
&\qquad\qquad\qquad\qquad\qquad \mathcal{R}el\,(B^* xx'x^*)(fx)(f'x') \\
&\qquad \mathcal{R}el\,(\textcircled{Σ}^* A^* B^*) pp' \quad = \quad \Sigma x^* : \mathcal{R}el\, A^* (\pi_1 p)(\pi_1 p'). \\
&\qquad\qquad\qquad\qquad\qquad \mathcal{R}el\,(B^* (\pi_1 p)(\pi_1 p') x^*)(\pi_2 p)(\pi_2 p')
\end{aligned}$$

We denote the system obtained by extending $\lambda{*}\mathcal{U}$ with the above IIRD by $\lambda{*}\mathcal{UE}$.

We remark that $\lambda{*}\mathcal{U}$ is a subsystem of $\lambda{*}\mathcal{UE}$ in the sense that every term of $\lambda{*}\mathcal{U}$ is a term of $\lambda{*}\mathcal{UE}$, and every derivation in $\lambda{*}\mathcal{U}$ is also a derivation in $\lambda{*}\mathcal{UE}$.

▶ **Definition 9.** Let $(-)' : \mathcal{Terms}(\lambda{*}\mathcal{U}) \to \mathcal{Terms}(\lambda{*}\mathcal{U})$ be the operation of marking every variable with an apostrophe.

The next operation computes the substitution into a term t of a “universal path” in the context of t. The context of t^* is three times larger than t: not only do we add the apostrophized variables, but also the starred variables which witness that x and x' are related.

▶ **Definition 10.**

$$(-)^* : \mathcal{Terms}(\lambda{*}\mathcal{U}) \to \mathcal{Terms}(\lambda{*}\mathcal{UE})$$

$$\begin{aligned}
(x)^* &= x^* \\
\circledast^* &= \mathsf{r}(\circledast) \\
(\textcircled{Π}AB)^* &= \textcircled{Π}^* A^* B^* \\
(\textcircled{Σ}AB)^* &= \textcircled{Σ}^* A^* B^* \\
(\lambda x{:}A.b)^* &= \lambda x{:}A\ \lambda x'{:}A'\ \lambda x^* : \mathcal{Rel}\, A^* x x'.b^* \\
(fa)^* &= f^* a a' a^* \\
(a,b)^* &= (a^*, b^*) \\
(\pi_1 p)^* &= \pi_1 p^* \\
(\pi_2 p)^* &= \pi_2 p^*
\end{aligned}$$

▶ **Theorem 11.** $(M[N/x])' = M'[N'/x']$

▶ **Theorem 12.** $(M[N/x])^* = M^*[N/x, N'/x', N^*/x^*]$

Proof.
Axiom. $(\circledast[N/x])^* = (\circledast)^* = \mathsf{r}(\circledast) = \mathsf{r}(\circledast)[N/x, N'/x', N^*/x^*]$
Variable.

$$(y[N/x])^* = \begin{cases} (x[N/x])^* = N^* = x^*[N/x, N'/x', N^*/x^*] & y = x \\ (y[N/x])^* = y^* = y^*[N/x, N'/x', N^*/x^*] & y \neq x \end{cases}$$

Product.

$$\begin{aligned}
(\textcircled{Π}AB\ [N/x])^* &= (\textcircled{Π}A[N/x]B[N/x])^* \\
&= \textcircled{Π}^* (A[N/x])^* (B[N/x])^* \\
&= \textcircled{Π}^* A^*[N/x, N'/x', N^*/x^*] B^*[N/x, N'/x', N^*/x^*] \\
&= (\textcircled{Π}^* A^* B^*)[N/x, N'/x', N^*/x^*]
\end{aligned}$$

Sum.

$$\begin{aligned}
(\textcircled{Σ}AB\ [N/x])^* &= (\textcircled{Σ}A[N/x]B[N/x])^* \\
&= \textcircled{Σ}^* (A[N/x])^* (B[N/x])^* \\
&= \textcircled{Σ}^* A^*[N/x, N'/x', N^*/x^*] B^*[N/x, N'/x', N^*/x^*] \\
&= (\textcircled{Σ}^* A^* B^*)[N/x, N'/x', N^*/x^*]
\end{aligned}$$

Abstraction. We remark that the variables can always be chosen so as not to interfere.

$$\begin{aligned}((\lambda y{:}A.b)[N/x])^* &= (\lambda y : A[N/x].b[N/x])^*\\ &= \lambda y{:}A[N/x]\ \lambda y'{:}(A[N/x])'\ \lambda y^* : \mathcal{R}el\,(A[N/x])^* yy'.(b[N/x])^*\\ &= \lambda y{:}A[N/x]\ \lambda y'{:}A'[N'/x']\ \lambda y^* : \mathcal{R}el\,A^*[N/x, N'/x', N^*/x^*]yy'.\\ &\qquad\qquad b^*[N/x, N'/x', N^*/x^*]\\ &= \lambda y{:}A[N, N', N^*/x, x', x^*]\ \lambda y'{:}A'[N, N', N^*/x, x', x^*]\\ &\qquad \lambda y^* : (\mathcal{R}el\,A^* yy')[N, N', N^*/x, x', x^*].b^*[N, N', N^*/x, x', x^*]\\ &= (\lambda y{:}A\ \lambda y'{:}A'\ \lambda y^* : \mathcal{R}el\,A^* yy'.b^*)[N, N', N^*/x, x', x^*]\\ &= (\lambda y : A.b)^*[N/x, N'/x', N^*/x^*]\end{aligned}$$

Application.

$$\begin{aligned}(st[N/x])^* &= (s[N/x]t[N/x])^*\\ &= (s[N/x])^*(t[N/x])(t[N/x])'(t[N/x])^*\\ &= (s^*[N, N', N^*/x, x', x^*]t[N/x]t'[N'/x]t^*[N, N', N^*/x, x', x^*]\\ &= (s^* tt't^*)[N, N', N^*/x, x', x^*]\\ &= (st)^*[N, N', N^*/x, x', x^*]\end{aligned}$$

Pairing.

$$\begin{aligned}((s,t)[N/x])^* &= (s[N/x], t[N/x])^*\\ &= ((s[N/x])^*, (t[N/x])^*)\\ &= (s^*[N/x, N'/x', N^*/x^*], t^*[N/x, N'/x', N^*/x^*])\\ &= (s^*, t^*)[N, N', N^*/x, x', x^*]\\ &= (s,t)^*[N, N', N^*/x, x', x^*]\end{aligned}$$

Projection.

$$\begin{aligned}((\pi_i t)[N/x])^* &= (\pi_i t[N/x])^*\\ &= \pi_i(t[N/x])^*\\ &= \pi_i(t^*[N, N', N^*/x, x', x^*])\\ &= \pi_i t^*[N, N', N^*/x, x', x^*]\\ &= (\pi_i t)^*[N, N', N^*/x, x', x^*]\end{aligned}$$ ◀

▶ **Corollary 13.** *Suppose $M = N$. Then $M^* = N^*$.*

Proof. Assume $M = (\lambda x{:}A.s)t$ and $N = s[t/x]$. We have

$$\begin{aligned}M^* = ((\lambda x{:}A.s)t)^* &= (\lambda x{:}A.s)^* tt't^*\\ &= (\lambda x{:}A\ \lambda x'{:}A'\ \lambda x^* : \mathcal{R}el\,A^* xx'.s^*)tt't^*\\ &= s^*[t/x][t'/x'][t^*/x^*]\\ &= (s[t/x])^* = N^*\end{aligned}$$

where the last equality is by the previous proposition.

Now suppose that $M = \pi_i(t_1, t_2)$, and $N = t_i$. Then

$$M^* = \pi_i(t_1^*, t_2^*) = t_i^* = N^*\ .$$

◀

▶ **Definition 14.** A $\lambda{*}\mathscr{U}$-context Γ is said to be a *$\mathscr{U}$-context* if Γ is of the form

$$x_1 : \mathscr{T}A_1, \dots, x_n : \mathscr{T}A_n(x_1, \dots, x_{n-1})$$

and for $0 \le i < n$, it holds that

$$x_1 : \mathscr{T}A_1, \dots, x_i : \mathscr{T}A_i(x_1, \dots, x_{i-1}) \vdash A_{i+1}(x_1, \dots, x_i) : \mathscr{U}$$

If Γ is a $\mathscr{U}$-context, and $\Gamma \vdash A : \mathscr{U}$, we call A a *$\mathscr{U}$-type in* Γ.

▶ **Definition 15.** Given a $\mathscr{U}$-context $\Gamma = \{x_1{:}\mathscr{T}A_1, \dots x_n{:}\mathscr{T}A_n\}$, put

$$\Gamma^* = \begin{cases} x_1 : \mathscr{T}A_1, \ \dots, \ x_n : \mathscr{T}A_n, \\ x_1' : \mathscr{T}A_1', \ \dots, \ x_n' : \mathscr{T}A_n', \\ x_1^* : \mathscr{Rel}\,A_1^* x_1 x_1', \ \dots, \ x_n^* : \mathscr{Rel}\,A_n^* x_n x_n' \end{cases}$$

Let Γ' be obtained from Γ by apostrophizing every variable, including those occurring in their declared types. Obviously, we can have

▶ **Theorem 16.** $\Gamma \vdash M : A \implies \Gamma' \vdash M' : A'$.

▶ **Theorem 17** (Extensionality of $\lambda{*}$)**.**

$$\Gamma \vdash_{\lambda*} M : A \quad \implies \quad \overline{\Gamma}^* \vdash_{\lambda*\mathscr{UE}} \overline{M}^* : \mathscr{Rel}\,\overline{A}^*\,\overline{M}\,\overline{M}' \tag{14}$$

Proof. We proceed by induction on the derivation.

We will work directly with the images $\overline{M}$ in $\lambda{*}\mathscr{U}$, confusing M with $\overline{M}$. The distinction is mostly irrelevant, but we need it e.g. in the conversion rule, where the type to be converted to has to be in the image of $\overline{\cdot}$.

Axiom. Suppose $\Gamma \vdash \circledast : \mathscr{T}\circledast$. We have

$$\circledast^* = \mathsf{r}(\circledast) : \mathscr{Eq}\circledast\circledast = \mathscr{Rel}\,\mathsf{r}(\circledast)\circledast\circledast = \mathscr{Rel}\circledast^*\circledast\circledast'$$

where $\mathsf{r}(\circledast) : \mathscr{Eq}\circledast\circledast$ in any context.

By conversion rule, $\Gamma^* \vdash \circledast^* : \mathscr{Rel}\circledast^*\circledast\circledast'$.

Variable. Suppose we have a derivation tree with root

$$\frac{\Gamma \vdash A : \mathscr{T}\circledast}{\Gamma, x : \mathscr{T}A \vdash x : \mathscr{T}A}$$

(Notice that the hypothesis says that A is a $\mathscr{U}$-type in Γ.)

By the previous proposition, $\Gamma' \vdash A' : \mathscr{T}\circledast$.

By induction hypothesis, $\Gamma^* \vdash A^* : \mathscr{Rel}\circledast^* AA'$.

Since $\mathscr{Rel}\circledast^* AA' = \mathscr{Eq}AA'$, we have $\Gamma^* \vdash A^* : \mathscr{Eq}AA'$ by conversion.

Yet Γ^* also yields that $\mathscr{T}A : *$ and $\mathscr{T}A' : *$, and thus we may form the context $\Gamma^*, x : \mathscr{T}A, x' : \mathscr{T}A' \vdash$. In this context, we may derive that

$$\Gamma^*, x : \mathscr{T}A, x' : \mathscr{T}A' \vdash \mathscr{Rel}\,A^* x x' : *$$

using the typing rule for the $\mathscr{Rel}$ constructor.

By the variable rule, we have

$$\Gamma^*, x : \mathscr{T}A, x' : \mathscr{T}A', x^* : \mathscr{Rel}\,A^* x x' \vdash x^* : \mathscr{Rel}\,A^* x x'$$

The context in the above judgement is $(\Gamma, x : \mathscr{T}A)^*$. The subject is $(x)^*$. The type predicate is as displayed in (14).

Weakening. Suppose they give you

$$\frac{\Gamma \vdash M : \mathcal{T}A \qquad \Gamma \vdash B : \mathcal{T}\circledast}{\Gamma, y : \mathcal{T}B \vdash M : \mathcal{T}A}$$

The induction hypotheses give that

$$\Gamma^* \vdash M^* : \mathscr{Rel}\, A^* M M'$$
$$\Gamma^* \vdash B^* : \mathscr{Rel}\, \circledast^* B B'$$

As before, we may conclude that $B, B' : \mathscr{U}$ in Γ^*, that $B^* : \mathscr{Eq} B B'$, and that $\Gamma^*, y : \mathcal{T}B, y' : \mathcal{T}B'$ is a valid context.
Then $\mathscr{Rel}\, B^* y y' : *$, and by weakening we get

$$(\Gamma, y : \mathcal{T}B)^* \vdash M^* : \mathscr{Rel}\, A^* M M'$$

Formation. Consider the typing

$$\frac{\Gamma \vdash A : \mathcal{T}\circledast \qquad \Gamma, x{:}\mathcal{T}A \vdash B : \mathcal{T}\circledast}{\Gamma \vdash \textcircled{Π} A (\lambda x{:}\mathcal{T}A.B) : \circledast}$$

By induction, $\Gamma^* \vdash A^* : \mathscr{Rel}\, \circledast^* A A'$.
By conversion, this gives $\Gamma^* \vdash A^* : \mathscr{Eq} A A'$.
We also have $(\Gamma, x{:}\mathcal{T}A)^* \vdash B^* : \mathscr{Rel}\, \circledast^* B B'$.
That gives $\Gamma^*, x{:}\mathcal{T}A, x' : \mathcal{T}A', x^* : \mathscr{Rel}\, A^* x x' \vdash B^* : \mathscr{Eq} B B'$.
Using the abstraction rule, we derive

$$\Gamma^* \vdash \lambda x{:}\mathcal{T}A\; \lambda x'{:}\mathcal{T}A'\; \lambda x^* : \mathscr{Rel}\, A^* x x'.B^*$$
$$: \Pi x{:}\mathcal{T}A\; \Pi x'{:}\mathcal{T}A\;\; \Pi x^* : \mathscr{Rel}\, A^* x x'.\mathscr{Eq} B B'$$

which can be rewritten as

$$\Gamma^* \vdash (\lambda x : \mathcal{T}A.B)^* : \Pi x{:}\mathcal{T}A\; \Pi x'{:}\mathcal{T}A\;\; \Pi x^* : \mathscr{Rel}\, A^* x x'.\, \mathscr{Eq} B B'$$

Using the $\textcircled{$\Pi*-constructor, we may derive

$$\Gamma^* \vdash \textcircled{Π}^* A^* (\lambda x : \mathcal{T}A.B)^* : \mathscr{Eq}(\textcircled{Π} A B)(\textcircled{Π} A' B')$$

The subject of the above judgment is equal to

$$(\textcircled{Π} A (\lambda x : \mathcal{T}A.B))^*$$

while the type is convertible to $\mathscr{Rel}\, \mathsf{r}(\circledast)(\textcircled{Π} A B)(\textcircled{Π} A' B')$. Putting these together using the conversion rule yields

$$\Gamma^* \vdash (\textcircled{Π} A (\lambda x : \mathcal{T}A.B))^* : \mathscr{Rel}\, \circledast^* (\textcircled{Π} A B)(\textcircled{Π} A B)'$$

being of the required form.
By replacing Π with Σ, $\textcircled{$\Pi$}$ with $\textcircled{$\Sigma$}$, and $\textcircled{$\Pi* with $\textcircled{$\Sigma*, we may derive from the same hypotheses that

$$\Gamma^* \vdash (\textcircled{Σ} A (\lambda x : \mathcal{T}A.B))^* : \mathscr{Rel}\, \circledast^* (\textcircled{Σ} A B)(\textcircled{Σ} A B)'$$

Abstraction. If we have to do

$$\frac{\Gamma \vdash A : \mathcal{T}\circledast \qquad \Gamma, x : \mathcal{T}A \vdash B : \mathcal{T}\circledast \qquad \Gamma, x : \mathcal{T}A \vdash b : \mathcal{T}B}{\Gamma \vdash \lambda x{:}\mathcal{T}A.b \;:\; \mathcal{T}(\textcircled{Π} A (\lambda x{:}\mathcal{T}A.B))}$$

the induction hypotheses yield, with conversion, that

$$\begin{aligned}\Gamma^* &\vdash A^* : \mathcal{E}q A A' \\ \Gamma^*, x : \mathcal{T}A, x' : \mathcal{T}A', x^* : \mathcal{R}el\, A^* x x' &\vdash B^* : \mathcal{E}q B B' \\ \Gamma^*, x : \mathcal{T}A, x' : \mathcal{T}A', x^* : \mathcal{R}el\, A^* x x' &\vdash b^* : \mathcal{R}el\, B^* b b'\end{aligned}$$

Since A, A' are $\mathcal{U}$-types in Γ^*, and $\mathcal{R}el\, A^* x x' : *$, we can apply the abstraction rule three times in a row to see that the context

$$(\Gamma, x : \mathcal{T}A)^* \;=\; \Gamma^*, x : \mathcal{T}A, x' : \mathcal{T}A', x^* : \mathcal{R}el\, A^* x x'$$

yields typing judgment

$$\vdash \lambda x{:}\mathcal{T}A\; \lambda x'{:}\mathcal{T}A'\; \lambda x^* : \mathcal{R}el\, A^* x x'.b^* \;:\; \Pi x{:}\mathcal{T}A\; \Pi x'{:}\mathcal{T}A'\; \Pi x^* : \mathcal{R}el\, A^* x x'.\mathcal{R}el\, B^* b b'$$

The subject of this judgment is equal to $(\lambda x : \mathcal{T}A.b)^*$.
The type predicate may be converted as

$$\begin{aligned}&\Pi x{:}\mathcal{T}A\; \Pi x'{:}\mathcal{T}A'\; \Pi x^* : \mathcal{R}el\, A^* x x'. \quad \mathcal{R}el\, B^* b b' \\ &= \Pi x{:}\mathcal{T}A\; \Pi x'{:}\mathcal{T}A'\; \Pi x^* : \mathcal{R}el\, A^* x x'. \quad \mathcal{R}el\, B^* ((\lambda x{:}\mathcal{T}A.b)x)((\lambda x'{:}\mathcal{T}A'.b')x') \\ &= \Pi x{:}\mathcal{T}A\; \Pi x'{:}\mathcal{T}A'\; \Pi x^* : \mathcal{R}el\, A^* x x'. \\ &\qquad\qquad \mathcal{R}el\, ((\lambda x : \mathcal{T}A.B)^* x x' x^*)((\lambda x{:}\mathcal{T}A.b)x)((\lambda x'{:}\mathcal{T}A'.b')x') \\ &\quad = \mathcal{R}el\, (\textcircled{\Pi}^* A^* (\lambda x : \mathcal{T}A.B)^*)(\lambda x{:}\mathcal{T}A.b)(\lambda x{:}\mathcal{T}A.b)' \\ &\quad = \mathcal{R}el\, (\textcircled{\Pi} A (\lambda x : \mathcal{T}A.B))^* (\lambda x{:}\mathcal{T}A.b)(\lambda x{:}\mathcal{T}A.b)'\end{aligned}$$

which is of the form (14), as desired.

Application. If the derivation ends with

$$\frac{\begin{array}{l}\Gamma \qquad\quad \vdash A : \mathcal{T}\circledast \\ \Gamma, x{:}\mathcal{T}A \vdash B : \mathcal{T}\circledast \qquad \Gamma \vdash f : \mathcal{T}(\textcircled{\Pi} A(\lambda x{:}\mathcal{T}A.B)) \qquad \Gamma \vdash a : \mathcal{T}A\end{array}}{\Gamma \vdash f a : B[a/x]}$$

We thus have that that A (A') and B (B') are $\mathcal{U}$-types in Γ (Γ') and $\Gamma, x{:}\mathcal{T}A$ ($\Gamma', x' : \mathcal{T}A'$), respectively.
The induction hypotheses give us

$$\begin{aligned}\Gamma^* &\vdash A^* : \mathcal{E}q A A' \\ (\Gamma, x : \mathcal{T}A)^* &\vdash B^* : \mathcal{E}q B B' \\ \Gamma^* &\vdash f^* : \mathcal{R}el\, (\textcircled{\Pi} A (\lambda x{:}\mathcal{T}A.B))^* f f' \\ \Gamma^* &\vdash a^* : \mathcal{R}el\, A^* a a'\end{aligned}$$

We may rewrite the type of f^* as

$$\begin{aligned}&\mathcal{R}el\, (\textcircled{\Pi} A (\lambda x{:}\mathcal{T}A.B))^* f f' \\ &= \Pi x{:}\mathcal{T}A\; \Pi x'{:}\mathcal{T}A'\; \Pi x^* : \mathcal{R}el\, A^* x x'. \\ &\qquad \mathcal{R}el\, ((\lambda x{:}\mathcal{T}A.B)^* x x' x^*)(f x)(f' x') \\ &= \Pi x{:}\mathcal{T}A\; \Pi x'{:}\mathcal{T}A'\; \Pi x^* : \mathcal{R}el\, A^* x x'. \\ &\qquad \mathcal{R}el\, ((\lambda x{:}\mathcal{T}A\; \lambda x'{:}\mathcal{T}A'\; \lambda x^* : \mathcal{R}el\, A^* x x'.\; \mathcal{R}el\, B^*) x x' x^*)(f x)(f' x') \\ &= \Pi x{:}\mathcal{T}A\; \Pi x'{:}\mathcal{T}A'\; \Pi x^* : \mathcal{R}el\, A^* x x'.\mathcal{R}el\, B^* (f x)(f' x')\end{aligned}$$

Working in Γ^*, we now apply f^* to a, a', a^* (which types are $\mathcal{T}A, \mathcal{T}A', \mathcal{R}el\,A^*aa'$, respectively), in order to obtain

$$f^*aa'a^* : \mathcal{R}el\,B^*(fx)(f'x')[a/x, a'/x', a^*/x^*],$$

where we have used the hypotheses on A^* and B^* in validating application typing rule. Since the sets of primed, starred, and vanilla variables are disjoint, and every variable in f' is primed, while every variable in f vanilla, we may rewrite the above as

$$f^*aa'a^* : \mathcal{R}el\,B^*[a/x, a'/x', a^*/x^*](fa)(f'a')$$

By the substitution lemma,

$$B^*[a/x, a'/x', a^*/x^*] = (B[a/x])^*$$

We may thus rewrite the above judgment as

$$\Gamma^* \vdash (fa)^* : \mathcal{R}el\,B[a/x]^*(fa)(fa)'$$

as required.

Pairing. Given a derivation

$$\dfrac{\begin{array}{lll} \Gamma \qquad\quad \vdash A : \mathcal{T}\otimes & & \\ \Gamma, x{:}\mathcal{T}A \vdash B : \mathcal{T}\otimes & \Gamma \vdash a : \mathcal{T}A & \Gamma \vdash b : \mathcal{T}B[a/x] \end{array}}{\Gamma \vdash (a,b) : \mathcal{T}(\textcircled{\Sigma}A(\lambda x : \mathcal{T}A.B))}$$

we have

$$\begin{array}{l} \Gamma^* \vdash a^* : \mathcal{R}el\,A^*aa' \\ \Gamma^* \vdash b^* : \mathcal{R}el\,B[a/x]^*bb' \end{array}$$

We also have

$$\begin{array}{l} \mathcal{R}el\,(\textcircled{\Sigma}A(\lambda x{:}\mathcal{T}A.B))^*(a,b)(a',b') \\ \;= \mathcal{R}el\,(\textcircled{\Sigma}^*A^*(\lambda x : \mathcal{T}A.B)^*)(a,b)(a',b') \\ \;= \Sigma a^* : \mathcal{R}el\,A^*\pi_1(a,b)\pi_1(a',b'). \\ \qquad \mathcal{R}el\,((\lambda x{:}\mathcal{T}A.B)^*\pi_1(a,b)\pi_1(a',b')a^*)\pi_2(a,b)\pi_2(a',b') \\ \;= \Sigma a^* : \mathcal{R}el\,A^*aa'.\mathcal{R}el\,((\lambda x{:}\mathcal{T}A.B)^*aa'a^*)bb' \\ \;= \Sigma a^* : \mathcal{R}el\,A^*aa'.\mathcal{R}el\,(B^*[a, a', a^*/x, x', x^*])bb' \\ \;= \Sigma a^* : \mathcal{R}el\,A^*aa'.\mathcal{R}el\,B[a/x]^*bb' \end{array}$$

Using the pairing rule, we see that (a^*, b^*) can be given the type derived above. So by conversion, we find

$$\Gamma^* \vdash (a,b)^* : \mathcal{R}el\,(\textcircled{\Sigma}A(\lambda x{:}\mathcal{T}A.B))^*(a,b)(a,b)'$$

as required.

Projections. Given

$$\dfrac{\begin{array}{ll} \Gamma \qquad\quad \vdash A : \mathcal{T}\otimes & \\ \Gamma, x{:}\mathcal{T}A \vdash B : \mathcal{T}\otimes & \Gamma \vdash p : \mathcal{T}(\textcircled{\Sigma}A(\lambda x{:}\mathcal{T}A.B)) \end{array}}{\begin{array}{c} \Gamma \vdash \pi_1 p : \mathcal{T}A \\ \Gamma \vdash \pi_2 p : \mathcal{T}B[\pi_1 p/x] \end{array}}$$

we get, by induction hypothesis, that

$$\Gamma^* \vdash p^* : \mathcal{Rel}\,(\textcircled{\Sigma} A(\lambda x{:}\mathcal{T}A.B))^* pp'$$

By the same computation as the previous case, we see that that the type of p^* above is convertible to

$$\Sigma a^* : \mathcal{Rel}\,A^*(\pi_1 p)(\pi_1 p').\mathcal{Rel}\,B^*[\pi_1 p, \pi_1 p', a^*/x, x', x^*](\pi_2 p)(\pi_2 p')$$

But then we have

$$\pi_1 p^* : \mathcal{Rel}\,A^* \pi_1 p \pi_1 p'$$

$$\pi_2 p^* : \mathcal{Rel}\,B^*[\pi_1 p, \pi_1 p', \pi_1 p^*/x, x', x^*](\pi_2 p)(\pi_2 p')$$

The first judgment above already has the form required. As for the second, we use the substitution lemma to rewrite it as

$$(\pi_2 p)^* : \mathcal{Rel}\,B[\pi_1 p/x]^*(\pi_2 p)(\pi_2 p)'$$

and this too obeys the form of (14).

Conversion. Suppose

$$\frac{\Gamma \vdash M : A \qquad \Gamma \vdash B : \mathcal{T}\circledast \qquad A = B}{\Gamma \vdash M : B}$$

Since the source derivation is assumed to come from $\lambda{*}$, we may apply the induction hypothesis and obtain

$$\Gamma \vdash M^* : \mathcal{Rel}\,A^* M M'$$

The fact that $A = B$, entails, for deep typographical reasons, that $A' = B'$.
Hence by conversion, we have that $M : B$ as well as $M' : B'$.
But we also have that $B : \mathcal{U}$, so that $B^* : \mathcal{Rel}\,\circledast^* BB'$, or equivalently $B^* : \mathcal{Eq}BB'$.
These facts yield that $\mathcal{Rel}\,B^* M M' : *$.
By Proposition 13, $\mathcal{Rel}\,A^* M M' = \mathcal{Rel}\,B^* M M'$.
$\Gamma^* \vdash M^* : \mathcal{Rel}\,B^* M M'$ ◀

6 Internalization

Let us summarize the results of the previous two sections.

1. There exists a translation

$$\bar{\cdot} \quad : \quad \mathcal{Terms}(\lambda{*}) \to \mathcal{Terms}(\lambda{*}\mathcal{U})$$

such that

$$\Gamma \vdash_{\lambda *} M : A \quad \Longrightarrow \quad \overline{\Gamma} \vdash_{\lambda * \mathcal{U}} \overline{M} : \mathcal{T}\overline{A}$$

2. There exists a translation

$$(\cdot)^* \quad : \quad \mathcal{Terms}(\lambda{*}\mathcal{U}) \to \mathcal{Terms}(\lambda{*}\mathcal{U}\mathcal{E})$$

such that

$$\Gamma \vdash_{\lambda *} M : A \quad \Longrightarrow \quad \overline{\Gamma}^* \vdash_{\lambda * \mathcal{U}\mathcal{E}} \overline{M}^* : \mathcal{Rel}\,\overline{A}^* \overline{M}\,\overline{M}'$$

As we see, the final transformation

$$M \longmapsto \overline{M}^*$$

does not yet give us what we seek, because it maps a term in system $\lambda *$ to a term in a larger system. For example, the type of $\overline{M}^*$ is, in general, not a type of $\lambda *$, since it may contain references to $\mathcal{U}, \mathcal{T}, \textcircled{\Pi}^*$, etc.

What we are after is an operation that can be iterated. At the very least, it should map terms from one system to the same system. Preferably, the type of the result should also belong to the same universe as the type of the input term.

How can we obtain a universe closed under the $(\mathcal{E}q, \mathcal{R}el)$-family?

Easy: just add the codes corresponding to these types into the grammar of the universe.

$$
\begin{array}{lll}
\textsf{Inductive } \mathcal{U} : * & := & \\
\quad \mid \circledast & : & \mathcal{U} \\
\quad \mid \textcircled{\Pi} & : & \Pi A : \mathcal{U}.\, (\mathcal{T}A \to \mathcal{U}) \to \mathcal{U} \\
\quad \mid \textcircled{\Sigma} & : & \Pi A : \mathcal{U}.\, (\mathcal{T}A \to \mathcal{U}) \to \mathcal{U} \\
\quad \mid \textcircled{\simeq} & : & \mathcal{U} \to \mathcal{U} \to \mathcal{U} \\
\quad \mid \textcircled{\sim} & : & \Pi\{AB\} : \mathcal{U}.\, \mathcal{T}[\textcircled{\simeq}AB] \to \mathcal{T}A \to \mathcal{T}B \to \mathcal{U}
\end{array}
$$

The decodings of the new symbols are

$$
\begin{aligned}
\mathcal{T}(\textcircled{\simeq}AB) &= \mathcal{E}qAB \\
\mathcal{T}(\textcircled{\sim}ABeab) &= \mathcal{R}el\,eab
\end{aligned}
$$

Now, denoting by $\lambda * \mathcal{U}^{\mathcal{E}-}$ the extension of $\lambda *$ with the above universe, we can still define the operation $\overline{M}$ and $(M)^*$, but the problem is that the type of the resulting term

$$\mathcal{R}el\,\overline{A}^*\,\overline{M}\,\overline{M}' \quad : \quad *$$

belongs to the universe of $\lambda * \mathcal{U}^{\mathcal{E}-}$, and we do not have a reflection from terms of $\lambda * \mathcal{U}^{\mathcal{E}-}$ into $\mathcal{U}$ (since $\mathcal{U}$ does not have an internal universe).

Is it ever possible to define a universe where $(\cdot)^*$ can be iterated?

Surely, one cannot know for sure until one has tried every possible avenue. With the benefit of that knowledge, let us try to expand the realm of inductive-recursive definitions, to allow the decoding function of the IIRD type $\mathcal{E}qAB$, which is itself the decoding of a constructor of a lower index type $\mathcal{U}$ being defined simultaneously with it ($\mathcal{T}(\textcircled{\simeq}AB) = \mathcal{E}qAB$), to be valued back in the type $\mathcal{U}$! In essence, after evaluating the relation $\mathcal{R}el_{\{A,B\}}(e) : \mathcal{T}A \to \mathcal{T}B \to *$ coded by $e : \mathcal{E}q(A,B)$, we immediately reflect it back into $\mathcal{U}$! We denote $\lambda *$ extended with such a universe by $\lambda * \mathcal{U}^{\mathcal{E}}$.

The two IIRD definitions of the internal universes of $\lambda * \mathcal{U}^{\mathcal{E}-}$ and $\lambda * \mathcal{U}^{\mathcal{E}}$, are given in Figures 2 and 3, and may be compared side-by-side. It shall be seen that the only difference is in the $\mathcal{R}el$ map, and in the value of $\mathcal{T}$ at $\textcircled{\sim}$.

Observe that, this time, for $M : \mathcal{T}A$, the result of applying the $(\cdot)^*$-operator stays in $\mathcal{U}$! Indeed, we have

$$
\begin{aligned}
\Gamma \vdash_{\lambda *} M : A &\Longrightarrow \overline{\Gamma} \vdash_{\lambda * \mathcal{U}} \overline{M} : \mathcal{T}\overline{A} \\
&\Longrightarrow \overline{\Gamma}^* \vdash_{\lambda * \mathcal{U}^{\mathcal{E}}} \overline{M}^* : \mathcal{T}(\mathcal{R}el\,\overline{A}^*\,\overline{M}\,\overline{M}')
\end{aligned}
$$

and we have a code in $\mathcal{U}$ for the type of $\overline{M}^*$. This allows us to iterate the $(\cdot)^*$-operator as many times as we like.

$$
\begin{array}{lll}
\text{Inductive } \mathcal{U} : * \ := & & \\
\quad \mid \circledast & : & \mathcal{U} \\
\quad \mid \textcircled{Π} & : & \Pi A : \mathcal{U}.\ (\mathcal{T}A \to \mathcal{U}) \to \mathcal{U} \\
\quad \mid \textcircled{Σ} & : & \Pi A : \mathcal{U}.\ (\mathcal{T}A \to \mathcal{U}) \to \mathcal{U} \\
\quad \mid \textcircled{$\simeq$} & : & \mathcal{U} \to \mathcal{U} \to \mathcal{U} \\
\quad \mid \textcircled{$\sim$} & : & \Pi\{AB\} : \mathcal{U}.\ \mathcal{T}[\textcircled{$\simeq$}AB] \to \mathcal{T}A \to \mathcal{T}B \to \mathcal{U} \\
\text{with } \mathcal{T} : \mathcal{U} \to * \ := & & \\
\quad \mathcal{T}(\circledast) & = & \mathcal{U} \\
\quad \mathcal{T}(\textcircled{Π}AB) & = & \Pi a : \mathcal{T}A.\ \mathcal{T}[Ba] \\
\quad \mathcal{T}(\textcircled{Σ}AB) & = & \Sigma a : \mathcal{T}A.\ \mathcal{T}[Ba] \\
\quad \mathcal{T}(\textcircled{$\simeq$}AB) & = & \mathcal{E}q AB \\
\quad \mathcal{T}(\textcircled{$\sim$}eab) & = & \mathcal{R}el\, eab \\
\text{and } \mathcal{E}q : \mathcal{U} \to \mathcal{U} \to * \ := & & \\
\quad \mid \mathsf{r}(\circledast) & : & \mathcal{E}q \circledast\circledast \\
\quad \mid \textcircled{Π}^* & : & \Pi\{A\}\{A'\}\ \Pi A^* : \mathcal{E}q AA' \\
 & & \Pi\{B\}\{B'\}\ \Pi B^* : (\Pi aa'a^*.\ \mathcal{E}q(Ba)(B'a')).\quad \mathcal{E}q(\textcircled{Π}AB)(\textcircled{Π}A'B') \\
\quad \mid \textcircled{Σ}^* & : & \Pi\{A\}\{A'\}\ \Pi A^* : \mathcal{E}q AA' \\
 & & \Pi\{B\}\{B'\}\ \Pi B^* : (\Pi aa'a^*.\ \mathcal{E}q(Ba)(B'a')).\quad \mathcal{E}q(\textcircled{Σ}AB)(\textcircled{Σ}A'B') \\
\quad \mid \textcircled{$\simeq$}^* & : & \Pi\{AA'\}A^*\{BB'\}B^*.\quad \mathcal{E}q(\textcircled{$\simeq$}AB)(\textcircled{$\simeq$}A'B') \\
\quad \mid \textcircled{$\sim$}^* & : & \Pi AA'A^*BB'B^*ee'e^*aa'a^*bb'b^*.\quad \mathcal{E}q(\textcircled{$\sim$}eab)(\textcircled{$\sim$}e'a'b') \\
\text{with } \mathcal{R}el\,\{AB : \mathcal{U}\} : \mathcal{E}q AB \to \mathcal{T}A \to \mathcal{T}B \to * \ := & & \\
\quad \mid \mathcal{R}el\,(\mathsf{r}(\circledast))AB & = & \mathcal{E}q AB \\
\quad \mid \mathcal{R}el\,(\textcircled{Π}^* A^* B^*)ff' & = & \Pi x{:}\mathcal{T}A\Pi x'{:}\mathcal{T}A'\Pi x^* : \mathcal{R}el\, A^* xx'. \\
 & & \qquad \mathcal{R}el\,(B^* xx'x^*)(fx)(f'x') \\
\quad \mid \mathcal{R}el\,(\textcircled{Σ}^* A^* B^*)pp' & = & \Sigma x^* : \mathcal{R}el\, A^*(\pi_1 p)(\pi_1 p'). \\
 & & \qquad \mathcal{R}el\,(B^*(\pi_1 p)(\pi_1 p')x^*)(\pi_2 p)(\pi_2 p') \\
\quad \mid \mathcal{R}el\,(\textcircled{$\simeq$}^* AA'A^*BB'B^*)ee' & = & \Pi aa'a^*\ \Pi bb'b^*.\ \mathcal{E}q(\textcircled{$\sim$}eab)(\textcircled{$\sim$}e'a'b') \\
\quad \mid \mathcal{R}el\,(\textcircled{$\sim$}^* AA'A^*BB'B^*ee'e^*aa'a^*bb'b^*)\gamma\gamma' & = & \mathcal{R}el(e^*aa'a^*bb'b^*)\gamma\gamma'
\end{array}
$$

Figure 2 The universe of $\lambda * \mathcal{U}^{\mathcal{E}-}$.

$$
\begin{array}{lll}
\textsf{Inductive}\ \mathcal{U} : * \ := & & \\
\quad \mid \circledast & : & \mathcal{U} \\
\quad \mid \textcircled{$\scriptstyle\Pi$} & : & \Pi A : \mathcal{U}.\, (\mathcal{T}A \to \mathcal{U}) \to \mathcal{U} \\
\quad \mid \textcircled{$\scriptstyle\Sigma$} & : & \Pi A : \mathcal{U}.\, (\mathcal{T}A \to \mathcal{U}) \to \mathcal{U} \\
\quad \mid \textcircled{$\scriptstyle\simeq$} & : & \mathcal{U} \to \mathcal{U} \to \mathcal{U} \\
\quad \mid \textcircled{$\scriptstyle\sim$} & : & \Pi\{AB\} : \mathcal{U}.\, \mathcal{T}[\textcircled{$\scriptstyle\simeq$} AB] \to \mathcal{T}A \to \mathcal{T}B \to \mathcal{U} \\
\textsf{with}\ \ \mathcal{T} : \mathcal{U} \to * \ := & & \\
\quad \mathcal{T}(\circledast) & = & \mathcal{U} \\
\quad \mathcal{T}(\textcircled{$\scriptstyle\Pi$} AB) & = & \Pi a : \mathcal{T}A.\, \mathcal{T}[Ba] \\
\quad \mathcal{T}(\textcircled{$\scriptstyle\Sigma$} AB) & = & \Sigma a : \mathcal{T}A.\, \mathcal{T}[Ba] \\
\quad \mathcal{T}(\textcircled{$\scriptstyle\simeq$} AB) & = & \mathcal{E}q AB \\
\quad \mathcal{T}(\textcircled{$\scriptstyle\sim$} eab) & = & \mathcal{T}(\mathcal{R}el\, eab) \\
\textsf{and}\ \ \mathcal{E}q : \mathcal{U} \to \mathcal{U} \to * \ := & & \\
\quad \mid \mathsf{r}(\circledast) & : & \mathcal{E}q \circledast\circledast \\
\quad \mid \textcircled{$\scriptstyle\Pi$}^* & : & \Pi\{A\}\{A'\}\, \Pi A^* : \mathcal{E}q AA' \\
 & & \quad \Pi\{B\}\{B'\}\, \Pi B^* : (\Pi a a' a^* .\, \mathcal{E}q(Ba)(B'a')).\quad \mathcal{E}q(\textcircled{$\scriptstyle\Pi$} AB)(\textcircled{$\scriptstyle\Pi$} A'B') \\
\quad \mid \textcircled{$\scriptstyle\Sigma$}^* & : & \Pi\{A\}\{A'\}\, \Pi A^* : \mathcal{E}q AA' \\
 & & \quad \Pi\{B\}\{B'\}\, \Pi B^* : (\Pi a a' a^* .\, \mathcal{E}q(Ba)(B'a')).\quad \mathcal{E}q(\textcircled{$\scriptstyle\Sigma$} AB)(\textcircled{$\scriptstyle\Sigma$} A'B') \\
\quad \mid \textcircled{$\scriptstyle\simeq$}^* & : & \Pi\{AA'\}A^*\{BB'\}B^*.\quad \mathcal{E}q(\textcircled{$\scriptstyle\simeq$} AB)(\textcircled{$\scriptstyle\simeq$} A'B') \\
\quad \mid \textcircled{$\scriptstyle\sim$}^* & : & \Pi AA'A^*BB'B^*ee'e^*aa'a^*bb'b^*.\quad \mathcal{E}q(\textcircled{$\scriptstyle\sim$} eab)(\textcircled{$\scriptstyle\sim$} e'a'b') \\
\textsf{with}\ \ \mathcal{R}el\,\{AB : \mathcal{U}\} : \mathcal{E}q AB \to \mathcal{T}A \to \mathcal{T}B \to \mathcal{U} \ := & & \\
\quad \mid \mathcal{R}el\,(\mathsf{r}(\circledast))AB & = & \textcircled{$\scriptstyle\simeq$} AB \\
\quad \mid \mathcal{R}el\,(\textcircled{$\scriptstyle\Pi$}^* A^* B^*)ff' & = & \textcircled{$\scriptstyle\Pi$} x{:}A\ \textcircled{$\scriptstyle\Pi$} x'{:}A'\ \textcircled{$\scriptstyle\Pi$} x^* : \mathcal{R}el\, A^* xx'. \\
 & & \qquad \textcircled{$\scriptstyle\sim$}\,(B^* xx'x^*)(fx)(f'x') \\
\quad \mid \mathcal{R}el\,(\textcircled{$\scriptstyle\Sigma$}^* A^* B^*)pp' & = & \textcircled{$\scriptstyle\Sigma$} x^* : \mathcal{R}el\, A^*(\pi_1 p)(\pi_1 p'). \\
 & & \qquad \textcircled{$\scriptstyle\sim$}\,(B^*(\pi_1 p)(\pi_1 p')x^*)(\pi_2 p)(\pi_2 p') \\
\quad \mid \mathcal{R}el\,(\textcircled{$\scriptstyle\simeq$}^* AA'A^*BB'B^*)ee' & = & \textcircled{$\scriptstyle\Pi$} aa'a^*\ \textcircled{$\scriptstyle\Pi$} bb'b^*.\ \textcircled{$\scriptstyle\simeq$}(\textcircled{$\scriptstyle\sim$} eab)(\textcircled{$\scriptstyle\sim$} e'a'b') \\
\quad \mid \mathcal{R}el\,(\textcircled{$\scriptstyle\sim$}^* AA'A^*BB'B^*ee'e^*aa'a^*bb'b^*)\gamma\gamma' & = & \textcircled{$\scriptstyle\sim$}(e^*aa'a^*bb'b^*)\gamma\gamma'
\end{array}
$$

Figure 3 The universe of $\lambda{*}\mathcal{U}^{\mathcal{E}}$.

▶ Remark. One could naturally wonder whether our use of the expanded induction-recursion format is really necessary, whether it is consistent, and how much logical strength it consumes. Since the logical relation encapsulates some universal properties of the base system, we expect that closing off a universe under its logical relation would require a significant use of logical power. At the same time, we believe that our particular definition could be justified using sufficiently large universes (such as Palmgren's higher-order universes, or Setzer's Mahlo universes), considering the rather obvious inductive structure evident in the form of our definition.

In any case, the above issues do not really concern us here, because we are now ready to define our candidate type system internalizing extensional equality for closed types, and this system is almost certainly consistent. If we ultimately get where we were going, then it's not as important how we got there . . .

7 The wormhole

Having defined the universe containing (a reflection of) $\lambda*$ and closed under its own logical relation, we are now ready to . . . go there!

We define $\lambda\simeq$, a *dependent type theory with type equality.*

The system is obtained by *unquoting* the internal universe $\mathcal{U}$ of $\lambda*\mathcal{U}^{\mathcal{I}}$.

That is, rather than defining a reflection of meta-level into the object level, we expand our meta-level by *descending into the object level.*

Once we are inside $\mathcal{U}$ we forget that $*$ ever existed,

$$* \quad := \quad \mathcal{U}$$

The syntax of our new environment, as it appears to us, is given in Figure 4. It is almost the same as $\lambda*$ (and we only display the new typing rules), the only differences are:

1. A new type constructor, *type equality*, is present.
2. The universes are stratified to remove inconsistency.

The new type $A \simeq B$ has four introduction rules, stating that every type constructor preserves type equality – including type equality.

It has a single elimination rule which, given an element $e : A \simeq B$, returns a relation

$$\sim e : A \to B \to *$$

For every combination of an introduction rule with the elimination rule, it has a computation rule stating that the relation induced by one of the congruence constructors is given inductively by the logical condition associated to the corresponding type constructor.

▶ **Theorem 18.** *There exists an operation* $(\cdot)^* : \mathcal{T}\!erms(\lambda\simeq) \to \mathcal{T}\!erms(\lambda\simeq)$ *such that*

$$\Gamma \vdash_{\lambda\simeq} M : A \implies \Gamma^* \vdash_{\lambda\simeq} M^* : M \sim_{A^*} M'$$

In particular, if $\vdash A : *$ is a closed type, then there exists a closed term

$$\mathsf{r}(A) : A \simeq A$$

We write $a \simeq_A a' := a \sim_{\mathsf{r}(A)} a'$.
If $\vdash a : A$ is a closed term, then there exists closed

$$\mathsf{r}(a) : a \simeq_A a$$

The study of system $\lambda\simeq$ will be continued in future work. A proof of the above theorem may be found in a draft at `http://arxiv.org/abs/1401.1148`.

$$\begin{array}{rcl} A, t, e & ::= & *_n \;\mid x \mid \Pi x{:}A.B \mid \Sigma x{:}A.B \mid A \simeq B \mid a \sim_e b \\ & & \mid \lambda x{:}A.t \mid st \mid (s,t) \mid \pi_1 t \mid \pi_2 t \\ & & \mid *_n^* \mid \Pi^*[x,x',x^*]{:}A^*.B^* \mid \Sigma^*[x,x',x^*]{:}A^*.B^* \mid \simeq^* A^* B^* \end{array}$$

$$\frac{}{\Gamma \vdash *_n : *_{n+1}} \qquad \frac{\Gamma \vdash A : *_n}{\Gamma \vdash A : *_{n+1}}$$

$$\frac{\Gamma \vdash A : *_n \qquad \Gamma \vdash B : *_n}{\Gamma \vdash A \simeq B : *_n}$$

$$\frac{\Gamma \vdash A : *_n \quad \Gamma \vdash B : *_n \qquad \Gamma \vdash e : A \simeq B}{\Gamma \vdash \;\sim e : A \to B \to *_n}\;(\simeq\text{-Elim})$$

$$(a : A)(b : B) \qquad\qquad a \sim_e b \;:=\; \sim eab$$

$$\frac{}{\Gamma \vdash *_n^* : *_n \simeq *_n}$$

$$\frac{\left\{\begin{array}{l} \Gamma \vdash A : * \\ \Gamma \vdash A' : * \\ \Gamma \vdash A^* : A \simeq A' \end{array}\right\} \quad \left\{\begin{array}{r} \Gamma, x : A \;\; \vdash B : * \\ \Gamma, x' : A' \vdash B' : * \\ \Gamma, x{:}A, x'{:}A', x^* : x{\sim_{A^*}}x' \vdash B^* : B \simeq B' \end{array}\right\}}{\begin{array}{c} \Gamma \vdash \Pi^*[x,x',x^*] : A^*.\, B^* : \Pi x{:}A.B \simeq \Pi x'{:}A'.B' \\ \Gamma \vdash \Sigma^*[x,x',x^*] : A^*.\, B^* : \Sigma x{:}A.B \simeq \Sigma x'{:}A'.B' \end{array}}$$

$$\frac{\left\{\begin{array}{l} \Gamma \vdash A : * \\ \Gamma \vdash A' : * \\ \Gamma \vdash A^* : A \simeq A' \end{array}\right\} \quad \left\{\begin{array}{l} \Gamma \vdash B : * \\ \Gamma \vdash B' : * \\ \Gamma \vdash B^* : B \simeq B' \end{array}\right\}}{\Gamma \vdash \simeq^* A^* B^* : (A \simeq B) \simeq (A' \simeq B')}$$

$$\begin{array}{rcl} A \sim_{*^*} B & \longrightarrow & A \simeq B \\ f \sim_{\Pi^*[x,x',x^*]:A^*.B^*} f' & \longrightarrow & \Pi a{:}A \Pi a'{:}A' \Pi a^* : a \sim_{A^*} a'.\; fx \sim_{B^*[a,a',a^*/x,x',x^*]} f'x' \\ p \sim_{\Sigma^*[x,x',x^*]:A^*.B^*} p' & \longrightarrow & \Sigma a^* : \pi_1 p \sim_{A^*} \pi_1 p'.\; \pi_2 p \sim_{B^*[\pi_1 p, \pi_1 p', a^*/x,x',x^*]} \pi_2 p' \\ e \sim_{\simeq^* A^* B^*} e' & \longrightarrow & \Pi a{:}A \Pi a'{:}A' \Pi a^* : a \sim_{A^*} a' \\ & & \Pi b{:}B \Pi b'{:}B'\, \Pi b^* : b \sim_{B^*} b'.\; (a \sim_e b) \simeq (a' \sim_{e'} b') \end{array}$$

Figure 4 $\lambda\simeq$.

Acknowledgments. I would like to thank Thierry Coquand, Marc Bezem, Henk Barendregt, Jan Willem Klop, Mike Nahas, and Robbert Krebbers for insightful and inspiring discussions about extensionality and related topics. I also thank the anonymous referees for their many suggestions that led to major improvements in this paper.

This research was partially supported by French ANR project COQUAS (12 JS02 006 01).

References

1 Thorsten Altenkirch. Extensional equality in intensional type theory. In *LICS*, pages 412–420. IEEE Computer Society, 1999.

2 Robert Atkey, Neil Ghani, and Patricia Johann. A relationally parametric model of dependent type theory. In *Proceedings of the 41st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL'14, pages 503–515, New York, NY, USA, 2014. ACM.

3 H. P. Barendregt. Lambda calculi with types. In S. Abramsky, Dov M. Gabbay, and S. E. Maibaum, editors, *Handbook of Logic in Computer Science (Vol. 2)*, pages 117–309. Oxford University Press, Inc., New York, NY, USA, 1992.

4 Jean-Philippe Bernardy and Marc Lasson. Realizability and parametricity in pure type systems. In Martin Hofmann, editor, *Foundations of Software Science and Computational Structures*, volume 6604 of *Lecture Notes in Computer Science*, pages 108–122. Springer Berlin Heidelberg, 2011.

5 Thierry Coquand. Equality and dependent type theory, 2011.
`http://www.cse.chalmers.se/~coquand/equality.pdf`.

6 Peter Dybjer and Anton Setzer. Indexed induction-recursion. In Reinhard Kahle, Peter Schroeder-Heister, and Robert Stärk, editors, *Proof Theory in Computer Science*, volume 2183 of *Lecture Notes in Computer Science*, pages 93–113. Springer Berlin Heidelberg, 2001.

7 Neil Ghani, Lorenzo Malatesta, Fredrik Nordvall Forsberg, and Anton Setzer. Fibred data types. In *LICS*, pages 243–252. IEEE Computer Society, 2013.

8 F. Rabe and K. Sojakova. Logical Relations for a Logical Framework. *ACM Transactions on Computational Logic*, 2013. to appear.

9 William W. Tait. Extensional equality in the classical theory of types. In Werner Depauli-Schimanovich, Eckehart Köhler, and Friedrich Stadler, editors, *The Foundational Debate*, volume 3 of *Vienna Circle Institute Yearbook [1995]*, pages 219–234. Springer Netherlands, 1995.