Decidability of Conversion for Type Theory in Type Theory

Citation for the original published paper (version of record):
Abel, A., Öhman, J., Vezzosi, A. (2018). Decidability of Conversion for Type Theory in Type Theory. Proceedings of the ACM on Programming Languages, 2(POPL): 23:1-23:29.
http://dx.doi.org/10.1145/3158111

Decidability of Conversion for Type Theory in Type Theory

ANDREAS ABEL, Gothenburg University, Sweden
JOAKIM ÖHMAN, IMDEA Software Institute, Spain
ANDREA VEZZOSI, Chalmers University of Technology, Sweden

Type theory should be able to handle its own meta-theory, both to justify its foundational claims and to obtain a verified implementation. At the core of a type checker for intensional type theory lies an algorithm to check equality of types, or in other words, to check whether two types are convertible. We have formalized in Agda a practical conversion checking algorithm for a dependent type theory with one universe à la Russell, natural numbers, and η-equality for Π types. We prove the algorithm correct via a Kripke logical relation parameterized by a suitable notion of equivalence of terms. We then instantiate the parameterized fundamental lemma twice: once to obtain canonicity and injectivity of type formers, and once again to prove the completeness of the algorithm. Our proof relies on inductive-recursive definitions, but not on the uniqueness of identity proofs. Thus, it is valid in variants of intensional Martin-Löf Type Theory as long as they support induction-recursion, for instance, Extensional, Observational, or Homotopy Type Theory.

CCS Concepts: • **Theory of computation → Type theory**; *Proof theory*;

Additional Key Words and Phrases: Dependent types, Logical relations, Formalization, Agda

ACM Reference Format:
Andreas Abel, Joakim Öhman, and Andrea Vezzosi. 2018. Decidability of Conversion for Type Theory in Type Theory. *Proc. ACM Program. Lang.* 2, POPL, Article 23 (January 2018), 29 pages. https://doi.org/10.1145/3158111

1 INTRODUCTION

A fundamental component of the implementation of a typed functional programming language is an algorithm that checks equality of types; even more so for dependently-typed languages where equality of types is non-trivial, as it depends on the equality of terms. In this paper, we consider a dependent type theory with one universe à la Russell, natural numbers, and η-equality for Π types. The algorithm we implement follows the structure of the one used by the dependently-typed language Agda [2017], and has been discussed and refined before in the literature [Abel et al. 2016; Abel and Scherer 2012; Coquand 1991; Harper and Pfenning 2005]. In short, the algorithm will reduce the types or terms under comparison to weak head normal form, compare the heads, and, if they match, recurse on the bodies. Additionally, when comparing terms there is an extra type-directed phase which takes care of η-equality: at function type, we apply the terms under comparison to a fresh variable and continue comparing the results. The type-directed phase could be easily extended to η-equality for other types, by comparing how their elements behave under their eliminators, like it is done in Agda for records, singleton types and others. The proof that the

Authors' addresses: Andreas Abel, Department of Computer Science and Engineering, Gothenburg University, Rännvägen 6b, Göteborg, 41296, Sweden, andreas.abel@gu.se; Joakim Öhman, IMDEA Software Institute, Campus Montegancedo s/n, Madrid, 28223, Spain, joakim.ohman@imdea.org; Andrea Vezzosi, Department of Computer Science and Engineering, Chalmers University of Technology, Rännvägen 6b, Göteborg, 41296, Sweden, vezzosi@chalmers.se.

2475-1421/2018/1-ART23
https://doi.org/10.1145/3158111

algorithm correctly implements equality of types is based on a Kripke logical relation and follows Abel and Scherer [2012] and Abel et al. [2016].

Our novel contribution is a *full formalization* of the algorithm and its proof of soundness, completeness, and termination, in Martin-Löf Type Theory [Martin-Löf 1975] with intensional equality, plus some well-understood extensions such as induction-recursion [Dybjer 2000]. As a mechanization language, we use Agda itself in its latest version, using the language variant without Streicher's Axiom K [1993].[1] Consequently, our proof is directly transferable to related foundations such as Homotopy Type Theory extended by induction-recursion.

The Agda formalization is quite sizeable: around 10.000 lines of code (500.000 characters). Especially the proof of the fundamental theorem of logical relations is substantial (5.000 lines). The congruence rule for the recursor for natural numbers alone requires a lemma that stretches over more than 500 lines. It is not that the proof is mathematically deep, once the right definition of the logical relation and the right formulation of the fundamental theorem are in place—it is just that a formalization requires us to get all the technicalities right. In research articles with pen and paper proofs only, the proof of the fundamental theorem is often skipped or reduced to the single sentence "proof by induction on the typing and equality derivations". Yet checking that each case of the induction goes through would require a reviewer many hours of disciplined technical reasoning. Written out, the proof would stretch over many pages.

In previous works [Abel et al. 2007; Abel and Scherer 2012; Harper and Pfenning 2005], two logical relations and two fundamental theorems are needed for the meta theory: one that entails soundness of the algorithmic equality, and one for completeness. While in pen-and-paper proofs we could get away with remarks like "proof analogous", a formalization would require us to do the proof exercise a second time; in our case, a truly intimidating task. Instead, we have been able to formulate a more abstract version of the fundamental theorem which we instantiate twice. The properties of judgemental equality which we obtain for the first instantiation are actually necessary to establish the preconditions for the second instantiation; a single, most general instantiation of the fundamental theorem is not possible to the best of our knowledge.

The abstract version of the fundamental theorem requires a logical relation parametrized on a generic notion of typed equality that is specified by 8 properties (see Section 3.1). We were able to extract these conditions by evolving the original, specific version of fundamental theorem and logical relation into the abstract one. Here, we have been harvesting the first fruit of our formalization: Once a proof is formalized, it can be safely *refactored* like a piece of software to make it more general. The mechanical proof checker ensures we are not introducing mistakes during the factorization. Our abstract formulation of the fundamental lemma is thus not only a necessity, but also an outcome of the formalization. We have been able to advance the meta-theory of dependent types by our formalization efforts.

The simpler proof technique of Harper and Pfenning [2005] which considers only the approximate shape of types by erasing the dependencies is not applicable in our case because our types might be determined by computations involving those dependencies, e.g., by recursion on natural numbers. An alternative technique to prove decidability of conversion is Normalization by Evaluation [Abel et al. 2007], however reducing terms to normal form before comparing them is often wasteful, so such a proof technique would not directly prove the correctness of a practical conversion checking algorithm. Highly related is the work of Barras [2010] which formalizes impredicative type theory in set theory which is in turn formalized in the type-theoretic proof assistent Coq [INRIA 2017]. Barras uses axiomatic inaccessible cardinals to model universes; we try here to cut out the set theory and formalize type theory directly in type theory. Altenkirch and Kaposi [2016; 2017] formalize type

[1] `agda --without-K` implements pattern matching without the K axiom.

theory in type theory using intrinsically well-typed object syntax via quotient inductive types in the meta-language. They prove normalization by evaluation in Agda. However, their formalization is less comprehensive than ours: their object type theory lacks inductive types, and their universe lacks function types.

Overview. Logical relations have been used for many purposes and with equally varied definitions, here we try to give an overview of the design principle used for ours. After specifying the syntax and rules of our type theory $\lambda^{\Pi U \mathbb{N}}$, in Section 2 we prove only a minimal amount of properties by direct induction on typing or equality judgements. In particular, we prove the weakening lemma, i.e., that all the judgments still hold if we apply well-typed renamings. The remaining properties are derived from the logical relation later. We believe this makes the proof more extensible and resilient to changes in the formulation of the typing rules. In Section 2 we also define a *typed* weak head reduction relation which we prove deterministic. Using a typed variant of reduction gives us the soundness of reduction immediately, which is otherwise established by the subject reduction theorem. Subject reduction relies on the injectivity of the function type constructor which is difficult to prove for a dependent type theory with universes. In fact, it is only a consequence of the logical relation argument. Further, typed reduction is more flexible than untyped reduction and could be equipped with type-directed reduction rules needed in extensions of type theory, for instance, by singleton types or strict equality.

In Section 3 we define a Kripke logical relation, i.e., for each judgment $\Gamma \vdash J$ we define a corresponding relation $\Gamma \Vdash J$ which exhibits the inductive structure of types and their normalization properties. Many expected consequences of the judgments are actually non-trivial to derive. The logical relation rationalizes their meaning by focusing on which observations are supposed to make sense for each term. So for example $\Gamma \Vdash t : T$ not only tells us that $\Gamma \vdash t : T$ but that the same judgment holds respectively for the weak head normal forms, a and A, of t and T, and that the possible observations of a also belong to the logical relation. What we mean by observation depends on the type: if A is the universe then a has to either be neutral or a type former whose subterms also belong in the logical relation, if A is the type of natural numbers then a must be neutral or either zero or $\mathrm{suc}\, t$ for a t in the logical relation, and finally if A is the type of dependent functions, then applying a to a term in the logical relation must produce another such term. Equality judgments are similarly refined by comparing how the weak head normal forms of the terms involved react to observations. In Section 4 we present the conversion algorithm and use the consequences of the fundamental theorem to prove its termination. We prove the properties necessary to instantiate the logical relation with the conversion algorithm and use the fundamental theorem to derive its completeness. With this we can derive the decidability of the conversion judgments, which proves the conversion algorithm's correctness.

In summary, our work makes the following contribution to the programming language and type theory:

- A complete formalization of the decidability of conversion for a dependent type theory with one inductive type and one universe.
- Meta-theory based on typed weak head reduction.
- A single inductive-recursive Kripke logical relation which can be instantiated twice to first prove soundness and then completeness of the conversion algorithm. As a condition of the definition, the logical relation is indexed by a semantic type derivation, but we show proof irrelevance for these derivations.

We have restricted our investigation to the minimal core of type theory which gives us large elimination, to expose the structure of the metatheoretical development in its pure form.[2] We

[2] In contrast, the development of Abel and Scherer [2012] is slightly veiled by in addition of an irrelevance modality.

believe that our development can serve as a model for the justification and meta-theoretical investigations of extensions of type theory.

This paper is best **read in a PDF viewer**, because definitions and lemmata in blue are clickable and will open the corresponding Agda code in a browser, which is available online.[3]

2 A CORE TYPE THEORY WITH ONE UNIVERSE

In this section, we introduce $\lambda^{\Pi\mathsf{U}\mathbb{N}}$, a dependent type theory with natural numbers and recursion, dependent function types, and one universe. Using the recursor into the universe, we can define types whose shape depends on a value, for instance, the type of functions of arity $n \in \mathbb{N}$. Such recursively defined types are sometimes called *large eliminations* [Werner 1992]; their presence makes the type theory *fully dependent* in the sense that value-dependencies cannot be erased from types when constructing a model.[4]

2.1 Syntax

The grammar in Fig. 1 describes the raw syntax of $\lambda^{\Pi\mathsf{U}\mathbb{N}}$ in de Bruijn style. Expressions $t \in \mathsf{Exp}$ may or may not be in weak head normal form (Whnf), which in turn may or may not be neutral (Ne). An expression in Whnf is an expression that cannot be further reduced by the weak head reduction rules which we will later present. Expressions which are not in Whnf can all be deterministically reduced by these rules. Neutral expressions have a variable in head position that blocks further reductions.

In the formalization, Whnf and neutral are formalized as predicates over expressions. This allows us to use a simple data structure for expressions and, thanks to Agda's dependent pattern matching, it is easy to inspect them.

$$\begin{array}{lllll l}
\mathbb{N} & \ni & x & & & \text{de Bruijn indices} \\
\mathsf{Exp} & \ni & t, u, v, A, B & ::= & \bar{t} \mid t\,u \mid \mathsf{natrec}\,A\,t\,u\,v & \text{expressions} \\
\mathsf{Whnf} & \ni & \bar{t} & ::= & n \mid \mathsf{suc}\,t \mid \mathsf{zero} \mid \lambda t \mid \mathsf{U} \mid \mathbb{N} \mid \Pi\,A\,B & \text{weak head normal forms} \\
\mathsf{Ne} & \ni & n, m, N, M & ::= & \mathsf{i}_x \mid n\,t \mid \mathsf{natrec}\,A\,t\,u\,n & \text{neutral expressions} \\
\mathsf{Cxt} & \ni & \Gamma, \Delta & ::= & \epsilon \mid \Gamma, A & \text{contexts} \\
\mathsf{Wk} & \ni & \rho & ::= & \mathsf{id} \mid \uparrow\rho \mid \Uparrow\rho \mid \rho \circ \rho' & \text{weakenings} \\
\mathsf{Subst} & \ni & \sigma & ::= & \rho \mid \uparrow\sigma \mid \Uparrow\sigma \mid \sigma \circ \sigma' \mid \sigma, t & \text{substitutions}
\end{array}$$

Fig. 1. Grammar of $\lambda^{\Pi\mathsf{U}\mathbb{N}}$.

Expressions consist of the functional expressions: function application $t\,u$, abstraction λt, dependent function type $\Pi\,A\,B$; as well as the natural number expressions: zero, successor $\mathsf{suc}\,t$, natural number type $\mathbb{N}$, natural number recursion $\mathsf{natrec}\,A\,t\,u\,v$; and variables i_x and universe type U. We use $\boxed{t \equiv u}$ to denote syntactical equality of terms, which corresponds to propositional equality in the Agda formalization. For variables, we use de Bruijn [1972] indices i_x with $x \in \mathbb{N}$. The following positions bind one de Bruijn index: the sole argument of λ, the second argument of Π and the first argument of natrec. Note that the formalization does not enforce well-scopedness of expressions, instead we rely on the typing judgments to implicitly guarantee well-scopedness. In practice this has allowed for some mistakes when formalizing the typing rules, which we had to go back and correct, so intrinsically well-scoped syntax might have been a better choice.

[3] https://mr-ohman.github.io/logrel-mltt/decofconv/

[4] An example for a not fully-dependent type theory would be the Calculus of Constructions [Coquand and Huet 1988] which can be erased to F^{ω} [Geuvers 1994].

Expressions encompass values and types. For instance, $\lambda(\lambda \mathsf{i}_0)$, which would be $\lambda A.\ \lambda x.\ x$ with variable names, is the polymorphic identity function of type $\Pi\, \mathsf{U}\, (\Pi\, \mathsf{i}_0\, \mathsf{i}_1)$, which is $\Pi(A{:}\mathsf{U}).\ \Pi(x{:}A).\ A$ with names. Some expressions serve both as an object (term) and a type. For instance, $\mathbb{N}$ is the type of zero but also an inhabitant of universe U.

Contexts Γ are snoc-lists of (type) expressions to record the types of the free variables of a term or its type. We number the entries in contexts from right to left. For instance, the 3-element context (ϵ, C, B, A), associates the variables i_0 and i_1 and i_2 with types A and B and C, respectively.

Weakenings ρ, if executed on a term t as $t[\rho]$, can raise free de Bruijn indices of term t. The identity weakening id does nothing, the shifting of a weakening $\uparrow\!\rho$ adds one to all indices, the lifting of a weakening $\Uparrow\!\rho$ is for traversing under binders, and composition $\rho \circ \rho'$ (pronounce: ρ *after* ρ') lets us execute first weakening ρ' and then ρ.

Substitutions σ, executed as $t[\sigma]$, replace the free de Bruijn indices of term t by new terms. The substitution action $t[\sigma]$ is defined by recursion on the term, in Fig. 2 we give the relevant rewrite rules that follow from that. For instance, substitution (σ, t) would replace variable i_0 by t and the others according to σ. We abbreviate the singleton substitution action $t[\mathsf{id}, u]$ by $t[u]$; it just replaces i_0 by u and leaves the other variables unchanged. Weakenings are implicitly coerced to substitutions; this defines $t[\rho]$. Weakenings and substitutions obey the usual laws, for instance, $\Uparrow\!\sigma \circ \uparrow\!\mathsf{id} = \uparrow\!\mathsf{id} \circ \sigma$.

$$
\begin{array}{rcl@{\qquad\qquad}lll}
\mathsf{U}[\sigma] & \Longrightarrow & \mathsf{U} & \mathsf{i}_x & [\mathsf{id}] & \Longrightarrow\ \mathsf{i}_x \\
(\Pi\, F\, G)[\sigma] & \Longrightarrow & \Pi\, F[\sigma]\, G[\Uparrow\!\sigma] & \mathsf{i}_x & [\uparrow\!\mathsf{id}] & \Longrightarrow\ \mathsf{i}_{x+1} \\
\mathbb{N}[\sigma] & \Longrightarrow & \mathbb{N} & \mathsf{i}_x & [\uparrow\!\sigma] & \Longrightarrow\ (\mathsf{i}_x[\sigma])[\uparrow\!\mathsf{id}] \\
(\lambda t)[\sigma] & \Longrightarrow & \lambda\, t[\Uparrow\!\sigma] & \mathsf{i}_0 & [\Uparrow\!\sigma] & \Longrightarrow\ \mathsf{i}_0 \\
(t\, u)[\sigma] & \Longrightarrow & t[\sigma]\, u[\sigma] & \mathsf{i}_{x+1} & [\Uparrow\!\sigma] & \Longrightarrow\ (\mathsf{i}_x[\sigma])[\uparrow\!\mathsf{id}] \\
\mathsf{zero}[\sigma] & \Longrightarrow & \mathsf{zero} & \mathsf{i}_0 & [\sigma, t] & \Longrightarrow\ t \\
(\mathsf{suc}\, t)[\sigma] & \Longrightarrow & \mathsf{suc}\, t[\sigma] & \mathsf{i}_{x+1} & [\sigma, t] & \Longrightarrow\ \mathsf{i}_x[\sigma] \\
(\mathsf{natrec}\, A\, t\, u\, a)[\sigma] & \Longrightarrow & \mathsf{natrec}\, A[\Uparrow\!\sigma]\, t[\sigma]\, u[\sigma]\, a[\sigma] & \mathsf{i}_x & [\sigma \circ \sigma'] & \Longrightarrow\ (\mathsf{i}_x[\sigma'])[\sigma]
\end{array}
$$

Fig. 2. Rewrite rules for substitutions.

For non-dependent function types, we introduce the arrow notation. We define it as a Π type with its second element weakened:

$$A \to B \triangleq \Pi\, A\, B[\uparrow\!\mathsf{id}]$$

Observe that $(A \to B)[\sigma] = (\Pi\, A\, B[\uparrow\!\mathsf{id}])[\sigma] \Longrightarrow \Pi\, A[\sigma]\, (B[\uparrow\!\mathsf{id}][\Uparrow\!\sigma]) = \Pi\, A[\sigma]\, (B[\sigma][\uparrow\!\mathsf{id}]) = A[\sigma] \to B[\sigma]$ as expected, according to the substitution laws.

The expression $\mathsf{natrec}\, \mathsf{U}\, A\, (\lambda\lambda(\mathbb{N} \to \mathsf{i}_0))\, n$, with names $\mathsf{natrec}\, \mathsf{U}\, A\, (\lambda m.\lambda B.\, \mathbb{N} \to B)\, n$, codes type $\mathbb{N}^n \to A$ which is short for $\mathbb{N} \to (\cdots \to (\mathbb{N} \to A) \ldots)$ with n occurrences of $\mathbb{N}$. It is an example of a large elimination of value n into universe U, producing a (small) type.

In the remainder of this paper, we reserve and exclusively use names n, m, N, M for neutral expressions, Γ, Δ for contexts, ρ for weakenings and σ for substitutions. Other names, for instance t, u, A, B, are used for expressions. Names with a bar, for instance $\bar{t}$, are used for expressions in Whnf. Note that symbols t and $\bar{t}$ are distinct, and $\bar{t}$ need not necessarily denote the Whnf of t. However, we will often use them in that way.

$$\boxed{\vdash \Gamma}$$

$$\frac{}{\vdash \epsilon} \qquad \frac{\vdash \Gamma \qquad \Gamma \vdash A}{\vdash \Gamma, A}$$

$$\boxed{\Gamma \vdash A}$$

$$\frac{\vdash \Gamma}{\Gamma \vdash \mathsf{U}} \qquad \frac{\vdash \Gamma}{\Gamma \vdash \mathbb{N}} \qquad \frac{\Gamma \vdash F \qquad \Gamma, F \vdash G}{\Gamma \vdash \Pi\, F\, G} \qquad \frac{\Gamma \vdash A : \mathsf{U}}{\Gamma \vdash A}$$

$$\boxed{i : A \in \Gamma}$$

$$\frac{}{\mathsf{i}_0 : A[\uparrow\mathsf{id}] \in \Gamma, A} \qquad \frac{\mathsf{i}_x : A \in \Gamma}{\mathsf{i}_{x+1} : A[\uparrow\mathsf{id}] \in \Gamma, B}$$

$$\boxed{\Gamma \vdash A = B}$$

$$\frac{\Gamma \vdash A}{\Gamma \vdash A = A} \qquad \frac{\Gamma \vdash A = B}{\Gamma \vdash B = A} \qquad \frac{\Gamma \vdash A = B \qquad \Gamma \vdash B = C}{\Gamma \vdash A = C}$$

$$\frac{\Gamma \vdash F \qquad \Gamma \vdash F = H \qquad \Gamma, F \vdash G = E}{\Gamma \vdash \Pi\, F\, G = \Pi\, H\, E} \qquad \frac{\Gamma \vdash A = B : \mathsf{U}}{\Gamma \vdash A = B}$$

$$\boxed{\Gamma \vdash t : A}$$

$$\frac{\vdash \Gamma}{\Gamma \vdash \mathbb{N} : \mathsf{U}} \qquad \frac{\Gamma \vdash F : \mathsf{U} \qquad \Gamma, F \vdash G : \mathsf{U}}{\Gamma \vdash \Pi\, F\, G : \mathsf{U}} \qquad \frac{\vdash \Gamma \qquad i : A \in \Gamma}{\Gamma \vdash i : A} \qquad \frac{\Gamma \vdash F \qquad \Gamma, F \vdash t : G}{\Gamma \vdash \lambda t : \Pi\, F\, G}$$

$$\frac{\Gamma \vdash g : \Pi\, F\, G \qquad \Gamma \vdash a : F}{\Gamma \vdash g\, a : G[a]} \qquad \frac{\vdash \Gamma}{\Gamma \vdash \mathsf{zero} : \mathbb{N}} \qquad \frac{\Gamma \vdash t : \mathbb{N}}{\Gamma \vdash \mathsf{suc}\, t : \mathbb{N}} \qquad \frac{\Gamma \vdash t : A \qquad \Gamma \vdash A = B}{\Gamma \vdash t : B}$$

$$\frac{\Gamma, \mathbb{N} \vdash G \qquad \Gamma \vdash z : G[\mathsf{zero}] \qquad \Gamma \vdash s : \Pi\, \mathbb{N}\, (G \longrightarrow G[\uparrow\mathsf{id}, \mathsf{suc}\, i_0]) \qquad \Gamma \vdash t : \mathbb{N}}{\Gamma \vdash \mathsf{natrec}\, G\, z\, s\, t : G[t]}$$

$$\boxed{\Gamma \vdash t = u : A}$$

$$\frac{\Gamma \vdash t : A}{\Gamma \vdash t = t : A} \qquad \frac{\Gamma \vdash t = u : A}{\Gamma \vdash u = t : A} \qquad \frac{\Gamma \vdash t_1 = t_2 : A \qquad \Gamma \vdash t_2 = t_3 : A}{\Gamma \vdash t_1 = t_3 : A} \qquad \frac{\Gamma \vdash t = u : A \qquad \Gamma \vdash A = B}{\Gamma \vdash t = u : B}$$

$$\frac{\Gamma \vdash F \qquad \Gamma \vdash F = H : \mathsf{U} \qquad \Gamma, F \vdash G = E : \mathsf{U}}{\Gamma \vdash \Pi\, F\, G = \Pi\, H\, E : \mathsf{U}} \qquad \frac{\Gamma \vdash f = g : \Pi\, F\, G \qquad \Gamma \vdash a = b : F}{\Gamma \vdash f\, a = g\, b : G[a]}$$

$$\frac{\Gamma \vdash F \qquad \Gamma \vdash f : \Pi\, F\, G \qquad \Gamma \vdash g : \Pi\, F\, G \qquad \Gamma, F \vdash f[\uparrow\mathsf{id}]\, \mathsf{i}_0 = g[\uparrow\mathsf{id}]\, \mathsf{i}_0 : G}{\Gamma \vdash f = g : \Pi\, F\, G} \qquad \frac{\Gamma \vdash t = u : \mathbb{N}}{\Gamma \vdash \mathsf{suc}\, t = \mathsf{suc}\, u : \mathbb{N}}$$

$$\frac{\Gamma \vdash F \qquad \Gamma, F \vdash t : G \qquad \Gamma \vdash a : F}{\Gamma \vdash (\lambda t)\, a = t[a] : G[a]} \qquad \frac{\Gamma, \mathbb{N} \vdash F \qquad \Gamma \vdash z : F[\mathsf{zero}] \qquad \Gamma \vdash s : \Pi\, \mathbb{N}\, (F \longrightarrow F[\uparrow\mathsf{id}, \mathsf{suc}\, i_0])}{\Gamma \vdash \mathsf{natrec}\, F\, z\, s\, \mathsf{zero} = z : F[\mathsf{zero}]}$$

$$\frac{\Gamma, \mathbb{N} \vdash F \qquad \Gamma \vdash z : F[\mathsf{zero}] \qquad \Gamma \vdash s : \Pi\, \mathbb{N}\, (F \longrightarrow F[\uparrow\mathsf{id}, \mathsf{suc}\, i_0]) \qquad \Gamma \vdash t : \mathbb{N}}{\Gamma \vdash \mathsf{natrec}\, F\, z\, s\, (\mathsf{suc}\, t) = (s\, t)\, (\mathsf{natrec}\, F\, z\, s\, t) : F[\mathsf{suc}\, t]}$$

$$\frac{\Gamma, \mathbb{N} \vdash F = F' \qquad \Gamma \vdash z = z' : F[\mathsf{zero}] \qquad \Gamma \vdash s = s' : \Pi\, \mathbb{N}\, (F \longrightarrow F[\uparrow\mathsf{id}, \mathsf{suc}\, i_0]) \qquad \Gamma \vdash t = t' : \mathbb{N}}{\Gamma \vdash \mathsf{natrec}\, F\, z\, s\, t = \mathsf{natrec}\, F'\, z'\, s'\, t' : F[t]}$$

Fig. 3. Inference rules of $\lambda^{\Pi\mathsf{U}\mathbb{N}}$.

$$\boxed{\Gamma \vdash A \longrightarrow B} \text{ and } \boxed{\Gamma \vdash t \longrightarrow u : A}$$

$$\frac{\Gamma \vdash F \qquad \Gamma, F \vdash t : G \qquad \Gamma \vdash a : F}{\Gamma \vdash (\lambda t)\, a \longrightarrow t[a] : G[a]} \qquad \frac{\Gamma \vdash f \longrightarrow g : \Pi\, F\, G \qquad \Gamma \vdash a : F}{\Gamma \vdash f\, a \longrightarrow g\, a : G[a]}$$

$$\frac{\Gamma, \mathbb{N} \vdash F \qquad \Gamma \vdash z : F[\mathsf{zero}] \qquad \Gamma \vdash s : \Pi\, \mathbb{N}\, (F \rightarrow F[\uparrow\mathsf{id}, \mathsf{suc}\; i_0])}{\Gamma \vdash \mathsf{natrec}\; F\, z\, s\; \mathsf{zero} \longrightarrow z : F[\mathsf{zero}]}$$

$$\frac{\Gamma, \mathbb{N} \vdash F \qquad \Gamma \vdash z : F[\mathsf{zero}] \qquad \Gamma \vdash s : \Pi\, \mathbb{N}\, (F \rightarrow F[\uparrow\mathsf{id}, \mathsf{suc}\; i_0]) \qquad \Gamma \vdash t : \mathbb{N}}{\Gamma \vdash \mathsf{natrec}\; F\, z\, s\, (\mathsf{suc}\; t) \longrightarrow (s\, t)\, (\mathsf{natrec}\; F\, z\, s\, t) : F[\mathsf{suc}\; t]}$$

$$\frac{\Gamma, \mathbb{N} \vdash F \qquad \Gamma \vdash z : F[\mathsf{zero}] \qquad \Gamma \vdash s : \Pi\, \mathbb{N}\, (F \rightarrow F[\uparrow\mathsf{id}, \mathsf{suc}\; i_0]) \qquad \Gamma \vdash t \longrightarrow u : \mathbb{N}}{\Gamma \vdash \mathsf{natrec}\; F\, z\, s\, t \longrightarrow \mathsf{natrec}\; F\, z\, s\, u : F[t]}$$

$$\frac{\Gamma \vdash A \longrightarrow B : \mathsf{U}}{\Gamma \vdash A \longrightarrow B} \qquad \frac{\Gamma \vdash t \longrightarrow u : A \qquad \Gamma \vdash A = B}{\Gamma \vdash t \longrightarrow u : B}$$

$$\boxed{\Gamma \vdash A \longrightarrow^* B} \text{ and } \boxed{\Gamma \vdash t \longrightarrow^* u : A}$$

$$\frac{\Gamma \vdash A}{\Gamma \vdash A \longrightarrow^* A} \qquad \frac{\Gamma \vdash A \longrightarrow B \quad \Gamma \vdash B \longrightarrow^* C}{\Gamma \vdash A \longrightarrow^* C} \qquad \frac{\Gamma \vdash t : A}{\Gamma \vdash t \longrightarrow^* t : A} \qquad \frac{\Gamma \vdash t \longrightarrow u : A \quad \Gamma \vdash u \longrightarrow^* a : A}{\Gamma \vdash t \longrightarrow^* a : A}$$

Fig. 4. Weak head reduction rules.

2.2 Rules and Semantics

In Fig. 3 we define the judgements $\boxed{\vdash \Gamma}$ for well-formed contexts, $\boxed{\Gamma \vdash A}$ for well-formed types, $\boxed{\Gamma \vdash A = B}$ for conversion of types, $\boxed{\Gamma \vdash t : A}$ for type membership and $\boxed{\Gamma \vdash t = u : A}$ for conversion of terms. These are all defined simultaneously: If we look at the following rules, we can see how the judgements depend on each other:

$$\frac{\Gamma \vdash A : \mathsf{U}}{\Gamma \vdash A} \qquad \frac{\Gamma \vdash A = B : \mathsf{U}}{\Gamma \vdash A = B} \qquad \frac{\Gamma \vdash t : A \qquad \Gamma \vdash A = B}{\Gamma \vdash t : B} \qquad \frac{\Gamma \vdash t = u : A \qquad \Gamma \vdash A = B}{\Gamma \vdash t = u : B}$$

The first two rules let us lift a term or a term equality to the type level if it is of type U. This shows us that the judgements for types, $\Gamma \vdash A$ and $\Gamma \vdash A = B$, depend on the judgements for terms $\Gamma \vdash t : A$ and $\Gamma \vdash t = u : A$, respectively. The last two rules are the conversion rules, which let us take a term or a term equality to another, equivalent type. Here we see that both $\Gamma \vdash t : A$ and $\Gamma \vdash t = u : A$ depend on $\Gamma \vdash A = B$. Additionally, with the reflexivity rules, we can see that the equality judgements refer to the typing judgements, thus, all the judgements depend on each other and need to be defined simultaneously.

In Fig. 4 we list the reduction rules of $\lambda^{\Pi \mathsf{U} \mathbb{N}}$. Judgement $\boxed{\Gamma \vdash t \longrightarrow u : A}$ performs a single weak head reduction step from t to u, and $\boxed{\Gamma \vdash t \longrightarrow^* u : A}$ is its reflexive-transitive closure. Our grammar for Whnf captures exactly the well-typed terms that cannot be reduced further with these judgements. Any well-typed term not in Whnf will in finitely many steps reduce to a term in Whnf, yet this fact requires proof by logical relation and will appear quite late in our meta-theoretic development, see the Weak Head Normalization Theorem (3.28). Weak head normalization allows us to find the canonical head constructor of an expression if there is one.

Note that the reduction rules are typed, which is one of the main technical innovations of Abel, Coquand, and Manna [2016]. We immediately get that reduction is included in conversion. In contrast, with untyped reduction, we would need a type preservation (aka subject reduction) theorem which requires function type injectivity (aka Π injectivity) which in turn needs proof by a logical relation [Abel et al. 2007; Abel and Scherer 2012; Goguen 1994].

Below, we prove some key properties of the rules.

Lemma 2.1 (Well-formed context). *If $\Gamma \vdash J$ then $\vdash \Gamma$ for any typing or conversion judgement J.*

Proof. By induction on the judgement. □

Lemma 2.2 (Reduction subsumed by equality).

(1) *If either $\Gamma \vdash A \longrightarrow B$ or $\Gamma \vdash A \longrightarrow^* B$ then $\Gamma \vdash A = B$.*
(2) *If either $\Gamma \vdash t \longrightarrow u : A$ or $\Gamma \vdash t \longrightarrow^* u : A$ then $\Gamma \vdash t = u : A$.*

Proof. By induction on the judgement. □

With typing, we can also show that the first element of a reduction is a well-formed type or term.

Lemma 2.3 (Subject typing).

(1) *If $\Gamma \vdash A \longrightarrow B$ then $\Gamma \vdash A$.*
(2) *If $\Gamma \vdash t \longrightarrow u : A$ then $\Gamma \vdash t : A$.*

Proof. By induction on the reduction and by the well-formedness of the context (2.1). □

The analogue of this lemma, to derive $\Gamma \vdash B$ from $\Gamma \vdash A \longrightarrow B$ and $\Gamma \vdash u : A$ from $\Gamma \vdash t \longrightarrow u : A$, will be proven as a consequence of the fundamental theorem (see Thm. 3.26).

Lemma 2.4 (Whnfs do not reduce). *We cannot step from a weak head normal form.*

(1) *$\Gamma \vdash \overline{A} \longrightarrow A'$ is impossible.*
(2) *$\Gamma \vdash \overline{t} \longrightarrow t' : B$ is impossible.*

Furthermore, a reduction sequence starting with a Whnf *goes nowhere:*

(3) *If $\Gamma \vdash \overline{A} \longrightarrow^* A'$ then $\overline{A} \equiv A'$.*
(4) *If $\Gamma \vdash \overline{t} \longrightarrow^* t' : C$ then $\overline{t} \equiv t'$.*

Proof. By induction on the reduction. □

Reduction is deterministic; each expression has at most one reduct.

Lemma 2.5 (Reduction is deterministic).

(1) *If $\Gamma \vdash A \longrightarrow B$ and $\Gamma \vdash A \longrightarrow B'$ then $B \equiv B'$.*
(2) *If $\Gamma \vdash t \longrightarrow u : A$ and $\Gamma \vdash t \longrightarrow u' : A$ then $u \equiv u'$.*
(3) *If $\Gamma \vdash A \longrightarrow^* \overline{A}$ and $\Gamma \vdash A \longrightarrow^* \overline{A'}$ then $\overline{A} \equiv \overline{A'}$.*
(4) *If $\Gamma \vdash t \longrightarrow^* \overline{t} : A$ and $\Gamma \vdash t \longrightarrow^* \overline{t'} : A$ then $\overline{t} \equiv \overline{t'}$.*

Proof. By induction on the reduction, using the fact that Whnfs do not reduce (2.4). □

We classify the weakenings ρ from Γ to Δ by judgement $\boxed{\rho : \Delta \leq \Gamma}$, inductively given by the rules to follow. If we apply a well-formed weakening $\rho : \Delta \leq \Gamma$ to a term t in Γ, we obtain a term $t[\rho]$ in Δ.

$$\frac{}{\mathsf{id} : \Gamma \leq \Gamma} \qquad \frac{\rho : \Delta \leq \Gamma}{\uparrow\!\rho : (\Delta, A) \leq \Gamma} \qquad \frac{\rho : \Delta \leq \Gamma}{\Uparrow\!\rho : (\Delta, A[\rho]) \leq (\Gamma, A)}$$

The rule for weakening composition is missing, because it is admissible.

Lemma 2.6 (Weakening composition). *If $\rho : \Delta' \leq \Delta$ and $\rho' : \Delta \leq \Gamma$ then $\rho \circ \rho' : \Delta' \leq \Gamma$.*

Proof. By induction on the structure of well-formed weakenings. □

In the remainder of this article, we shall implicitly assume well-formedness of both contexts, $\vdash \Delta$ and $\vdash \Gamma$, whenever we mention $\rho : \Delta \leq \Gamma$.

To prove the Weakening Lemma (2.7) we strengthened our typing rules slightly, leading to the hypotheses marked in grey. For instance, in the case of $\Gamma \vdash \lambda t : \Pi\, F\, G$, it is not enough to only have $\Gamma, F \vdash t : G$ since we also need $\Gamma \vdash F$. While we can extract $\Gamma \vdash F$ from $\Gamma, F \vdash t : G$ via the context $\vdash \Gamma, F$, it does not immediately follow that the extracted derivation of $\Gamma \vdash F$ is a smaller than the original derivation of $\Gamma \vdash \lambda t : \Pi\, F\, G$, which means that the use of the induction hypothesis would not be justified a priori. Hence, we have strengthened the premises of our rules by an additional hypothesis $\Gamma \vdash F$ whenever the context extension Γ, F appears in a hypothesis. A similar technique has been applied by Harper and Pfenning [2005].

Alternatively, we could use sized types in the metalanguage [Abel and Pientka 2016; Hughes et al. 1996; Sacchini 2013]. This would involve making our judgments sized, so that when we extract $\Gamma \vdash F$ from $\Gamma, F \vdash t : G$ via context $\vdash \Gamma, F$, the sizes would witness that $\Gamma \vdash F$ is smaller than $\vdash \Gamma, F$ which is in turn smaller than $\Gamma, F \vdash t : G$. Thus, we could justify the use of the induction hypothesis on $\Gamma \vdash F$.

Lemma 2.7 (Weakening). *Let $\rho : \Delta \leq \Gamma$. If $\Gamma \vdash J$ then $\Delta \vdash J[\rho]$ for any typing, conversion, or reduction judgement J.*

Proof. By induction on the judgement. □

Finally, we define well-formed substitutions and their equality as $\boxed{\Delta \vdash \sigma : \Gamma}$ and $\boxed{\Delta \vdash \sigma = \sigma' : \Gamma}$, inductively by the rules to follow.

$$\frac{}{\Delta \vdash \sigma : \epsilon} \qquad \frac{\Delta \vdash \sigma \circ {\uparrow}\mathrm{id} : \Gamma \qquad \Delta \vdash i_0[\sigma] : A[\sigma \circ {\uparrow}\mathrm{id}]}{\Delta \vdash \sigma : \Gamma, A}$$

$$\frac{}{\Delta \vdash \sigma = \sigma' : \epsilon} \qquad \frac{\Delta \vdash \sigma \circ {\uparrow}\mathrm{id} = \sigma' \circ {\uparrow}\mathrm{id} : \Gamma \qquad \Delta \vdash i_0[\sigma] = i_0[\sigma'] : A[\sigma \circ {\uparrow}\mathrm{id}]}{\Delta \vdash \sigma = \sigma' : \Gamma, A}$$

We may consider a non-empty substitution $\Delta \vdash \sigma : \Gamma, A$ as a pair $(\sigma \circ {\uparrow}\mathrm{id},\ i_0[\sigma])$ of a substitution $\sigma \circ {\uparrow}\mathrm{id}$ and a term $i_0[\sigma]$, where we call the second component the *head* of σ and the first component the *tail* of σ.

Similarly to well-formed weakenings, we shall henceforth assume that the contexts Γ and Δ are well-formed when we mention $\Delta \vdash \sigma : \Gamma$ or $\Delta \vdash \sigma = \sigma' : \Gamma$. Unlike well-formed weakenings, we do not need to immediately prove that well-formed substitutions can be applied to well-formed expressions to create new well-formed expressions. Instead, this will follow from the fundamental theorem (see Thm. 3.31).

3 KRIPKE LOGICAL RELATIONS

To prove decidability of $\lambda^{\Pi \cup \mathbb{N}}$'s judgemental equality, we will show that it is equivalent to a more structured equality relation, called *algorithmic equality* $\Gamma \vdash t \mathrel{\hat{\Longleftrightarrow}} u : A$. For one, algorithmic equality will have no rule for transitivity, because the transitivity rule is very non-deterministic: If searching for a derivation of $\Gamma \vdash t = v : A$ we have to find a u such that $\Gamma \vdash t = u : A$ and $\Gamma \vdash u = v : A$, it is not clear how to guarantee progress. Instead, transitivity for algorithmic equality will be admissible. Secondly, we will prove that to derive $\Gamma \vdash t = u : A$, it is sufficient to derive equality $\Gamma \vdash \bar{t} = \bar{u} : A$ of the weak head normal forms of t and u. This way, we can exploit the structure of objects during equality check; for instance, *injectivity of constructors*: $\Gamma \vdash \mathsf{suc}\, t = \mathsf{suc}\, t' : \mathbb{N}$ holds iff $\Gamma \vdash t = t' : \mathbb{N}$.

For $\lambda^{\Pi \mathsf{U} \mathbb{N}}$ and related type theories, the essential properties like weak head normalization and injectivity of constructors are not provable directly by induction on the typing and equality judgements. Instead a *logical relation* is needed which represents the structure of objects explicitely, such as

- *canonicity*: a closed term of type $\mathbb{N}$ reduces either to zero or $\mathsf{suc}\, t$, or
- *function type injectivity*: if two function types are equal, then their domains and codomains are equal.

In this section, we will construct a logical relation $\Gamma \Vdash_{\ell} t = u : A$, pronounced "$t$ and u are reducibly equal at type A (of level ℓ in context Γ)." This relation will be *Kripke* in the sense that it is closed under weakening. It will expose the inductive structure of objects such that we can prove desired properties like normalization, canonicity, and injectivity. Analogously, we will define reducibility predicates and relations for the other main judgements of $\lambda^{\Pi \mathsf{U} \mathbb{N}}$, namely $\Gamma \vdash A$ and $\Gamma \vdash A = B$ and $\Gamma \vdash t : A$.

Reducible equality $\Gamma \Vdash_{\ell} t = u : A$ will be a subrelation of judgemental equality $\Gamma \vdash t = u : A$, but we will later need a similar relation that is a subrelation of algorithmic equality. Thus, we define it as a subrelation of a *generic equality* relation $\Gamma \vdash t \cong u : A$ which can be instantiated for both purposes. To prove that judgemental equality is in turn a subrelation of reducible equality, i.e., reducible equality is complete, we have to introduce a third relation, *valid equality* $\Gamma \Vdash^{\mathsf{v}}_{\ell} t = u : A$. Fig. 5 summarizes these relations and their connections.

$$\begin{tikzcd}
& \Gamma \vdash t \mathrel{\hat{\Longleftrightarrow}} u : A \arrow[dl, "\text{Soundness (4.3)}"'] & \\
\Gamma \vdash t = u : A \arrow[d, "\text{Fundamental Theorem (3.21)}"'] & & \Gamma \vdash t \cong u : A \arrow[ul, "\text{Instance 2}"'] \arrow[ll, "\text{Subsumption}", "\text{(Property 1)}"'] \\
\Gamma \Vdash^{\mathsf{v}}_{\ell} t = u : A \arrow[rr, "\text{Identity}", "\text{substitution (3.17)}"'] & & \Gamma \Vdash_{\ell} t = u : A \arrow[u, "\text{Escape Lemma (3.2)}"']
\end{tikzcd}$$

Fig. 5. Proof outline.

3.1 Generic Equality

To prove decidability of conversion, we need to introduce two logical relations, one with the conversion judgements for equality and another with algorithmic equality (see Section 4.1). The first logical relation will be used to derive proofs which are necessary to prove the properties of algorithmic equality, one being decidability, while the second logical relation will be used to prove completeness of algorithmic equality. The proofs about these logical relations are quite large, and with the two relations being quite similar, we would like to be able to not duplicate our work. We therefore introduce a notion of generic equality which we use to parameterize the definition of our logical relation, such that it can be instantiated to get the two logical relations we need.

We here give a specification for a set of relations being a *generic equality*. A generic equality consists of three relations that satisfy the properties we list below: $\boxed{\Gamma \vdash A \cong B}$ for equality of types, $\boxed{\Gamma \vdash t \cong u : A}$ for equality of terms of a type and $\boxed{\Gamma \vdash t \sim u : A}$ for equality between neutral terms. In the formalization, this is implemented as a record type with one field for each of the three relations and one field for each property.

Property 1 (Subsumption).

(1) *If* $\Gamma \vdash t \sim u : A$ *then* $\Gamma \vdash t \cong u : A$. *(Neutral equality is included in generic equality.)*
(2) *If* $\Gamma \vdash A \cong B : \mathsf{U}$ *then* $\Gamma \vdash A \cong B$. *(Small types are included in large types.)*

Further, generic equality is a subrelation of judgmental equality.

(3) *If* $\Gamma \vdash A \cong B$ *then* $\Gamma \vdash A = B$.
(4) *If* $\Gamma \vdash t \cong u : A$ *then* $\Gamma \vdash t = u : A$.

Property 2 (Partial equivalence relation). *The three generic equality relations* $\Gamma \vdash \_ \cong \_$ *and* $\Gamma \vdash \_ \cong \_ : A$ *and* $\Gamma \vdash \_ \sim \_ : A$ *are symmetric and transitive.*

Property 3 (Conversion). *If* $\Gamma \vdash t \cong u : A$ *and* $\Gamma \vdash A = B$ *then* $\Gamma \vdash t \cong u : B$. *(Same for* $\sim$*.)*

Property 4 (Weakening). *Generic equality is closed under weakning, i.e., if* $\rho : \Delta \leq \Gamma$ *and* $\Gamma \vdash J$ *then* $\Delta \vdash J[\rho]$, *where* J *ranges over the three equality forms.*

Property 5 (Weak head expansion). $\Gamma \vdash \_ \cong \_$ *and* $\Gamma \vdash \_ \cong \_ : A$ *are closed under weak head expansion.*

(1) *If* $\Gamma \vdash A \longrightarrow^* \overline{A}$ *and* $\Gamma \vdash B \longrightarrow^* \overline{B}$ *and* $\Gamma \vdash \overline{A} \cong \overline{B}$ *then* $\Gamma \vdash A \cong B$.
(2) *If* $\Gamma \vdash A \longrightarrow^* \overline{B}$ *and* $\Gamma \vdash a \longrightarrow^* \overline{a} : B$ *and* $\Gamma \vdash b \longrightarrow^* \overline{b} : B$ *and* $\Gamma \vdash \overline{a} \cong \overline{b} : B$ *then* $\Gamma \vdash a \cong b : A$.

Property 6 (Type constructor congruence). *If* $\vdash \Gamma$ *then:*

(1) $\Gamma \vdash \mathsf{U} \cong \mathsf{U}$.
(2) $\Gamma \vdash \mathbb{N} \cong \mathbb{N}$ *and* $\Gamma \vdash \mathbb{N} \cong \mathbb{N} : \mathsf{U}$.
(3) *If* $\Gamma \vdash F \cong H$ *and* $\Gamma, F \vdash G \cong E$ *then* $\Gamma \vdash \Pi\, F\, G \cong \Pi\, H\, E$. *(And analogously for* $\Gamma \vdash \_ \cong \_ : \mathsf{U}$.*)*

Property 7 (Value constructor congruence and η).

(1) *If* $\vdash \Gamma$ *then* $\Gamma \vdash \mathsf{zero} \cong \mathsf{zero} : \mathbb{N}$.
(2) *If* $\Gamma \vdash t \cong u : \mathbb{N}$ *then* $\Gamma \vdash \mathsf{suc}\, t \cong \mathsf{suc}\, u : \mathbb{N}$.
(3) *If* $\Gamma \vdash F$ *and* $\Gamma \vdash f : \Pi\, F\, G$ *and* $\Gamma \vdash g : \Pi\, F\, G$ *and* $\Gamma, F \vdash f[\uparrow\mathsf{id}]\, \mathsf{i}_0 \cong g[\uparrow\mathsf{id}]\, \mathsf{i}_0 : G$ *then* $\Gamma \vdash f \cong g : \Pi\, F\, G$.

Property 8 (Congruence for neutrals).

(1) *If* $\Gamma \vdash i : A$ *then* $\Gamma \vdash i \sim i : A$.
(2) *If* $\Gamma \vdash f \sim g : \Pi\, F\, G$ *and* $\Gamma \vdash a \cong b : F$ *then* $\Gamma \vdash f\, a \sim g\, b : G[a]$.
(3) *If* $\Gamma, \mathbb{N} \vdash F \cong F'$ *and* $\Gamma \vdash z \cong z' : F[\mathsf{zero}]$ *and* $\Gamma \vdash s \cong s' : \Pi\, \mathbb{N}\, (F \longrightarrow F[\uparrow\mathsf{id}, \mathsf{suc}\, i_0])$ *and* $\Gamma \vdash n \sim n' : \mathbb{N}$ *then* $\Gamma \vdash \mathsf{natrec}\, F\, z\, s\, n \sim \mathsf{natrec}\, F'\, z'\, s'\, n' : F[n]$.

Now that we have defined all the necessary properties, we will introduce our first instance of generic equality, which is simply judgmental equality.

Instance 1 (Judgmental equality). *The following instantiation of generic equality satisfies all the required properties:*

$$\Gamma \vdash A \cong B \textit{ instantiated to } \Gamma \vdash A = B$$
$$\Gamma \vdash t \cong u : A \textit{ instantiated to } \Gamma \vdash t = u : A$$
$$\Gamma \vdash t \sim u : A \textit{ instantiated to } \Gamma \vdash t = u : A$$

Proof. Subsumption (Prop. 1) is obvious. The weakening property (Prop. 4) is satisfied by the weakening lemma (2.7) and weak head expansion (Prop. 5) is proven by the fact that reduction is subsumed by equality (2.2). The remaining properties are fulfilled by the inference rules of judgmental equality. □

Note that for definitional equality we do not need the distinction between $\Gamma \vdash t \cong u : A$ and $\Gamma \vdash t \sim u : A$. However, for the algorithmic equality it is required, therefore it is part of the specification.

3.2 A Logical Relation for Reducibility

In this section, we define *reducibility judgements* $\Gamma \Vdash_\ell J$ for types, type equality, terms, and term equality. These judgements take the form of logical predicates and relations [Friedman 1975]. They entail well-formedness, i.e., if $\Gamma \Vdash_\ell J$ then $\Gamma \vdash J$, see the Escape[5] Lemma (3.2). The opposite direction, $\Gamma \vdash J$ implies $\Gamma \Vdash_\ell J$ will be a consequence of the Fundamental Theorem of Logical Relations (3.21). A logical relation on well-typed open terms is necessarily *Kripke*, i.e., closed under weakening: if $\rho : \Delta \leq \Gamma$ and $\Gamma \Vdash_\ell J$ then $\Delta \Vdash_\ell J$. Otherwise, the Fundamental Theorem would fail for binders (λ and Π).

The index ℓ denotes the type *level* and ranges over $0, 1$. We shall define the reducibility judgements by induction on ℓ, i.e., first for small types ($\ell = 0$) and then for large types ($\ell = 1$), using the judgements for small types. Spelled out, the reducibility judgements are:

$\Gamma \Vdash_\ell A$	A is a reducible type of level ℓ in context Γ
$\Gamma \Vdash_\ell A = B$	A and B are reducibly equal types of level ℓ in context Γ
$\Gamma \Vdash_\ell t : A$	t is a reducible term of level-ℓ type A in context Γ
$\Gamma \Vdash_\ell t = u : A$	t and u are reducibly equal terms of level-ℓ type A in context Γ

These judgements will imply that all involved objects are *reducible* [Girard 1972] (in the sense of weak normalization), in particular, all involved objects have a weak head normal form. Further, the judgements do not distinguish between objects that have the same weak head normal form. This is achieved by defining the judgements on weak head normal forms and closing them under weak head expansion.

The qualifier *"logical"* means that objects are characterized by their behavior, e.g., non-neutral objects of type $\mathbb{N}$ should reduce to zero or $\mathsf{suc}\, t$ for a reducible t, and functions should yield a reducible result if applied to a reducible argument. As a first approximation, let us define $\Gamma \Vdash_\ell t : \Pi F G$ to hold if $\Gamma \vdash t \longrightarrow^* \underline{t} : \Pi F G$ and for all $\Gamma \Vdash_\ell a : F$ we have $\Gamma \Vdash_\ell t\, a : G[a]$. Note that formula $\Gamma \Vdash_\ell a : F$ occurs *negatively* in this definition, which has dire consequences: First, we will not be able to inductively prove weakening for this definition, thus, we have to build it into the definition. Hence, we require instead that for all $\rho : \Delta \leq \Gamma$ and $\Delta \Vdash_\ell a : F[\rho]$ we have $\Delta \Vdash_\ell t[\rho]\, a : G[\rho, a]$.

Secondly, the negative occurrence prevents us to define $\Gamma \Vdash_\ell t : A$ as inductive predicate. Instead, we have to define it by recursion on the type A, but not simply on the size of the type expression A, since this does not get smaller in the recursive calls, e.g., $\Delta \Vdash_\ell t[\rho]\, a : G[\rho, a]$. We define $\Gamma \Vdash_\ell t : A$ by recursion on the *derivation* $\mathcal{T}$ of $\Gamma \Vdash_\ell A$, written $\mathcal{T} :: \Gamma \Vdash_\ell A$, which in turn is an inductive predicate.

[5] The terminology *escaping the logical relation* was coined by Schürmann and Sarnat [2008].

Thus, the term reducibility judgement will depend on $\mathcal{T}$ and we write $\Gamma \Vdash_\ell t : A/_{\mathcal{T}}$. In turn, the rule for $\Gamma \Vdash_\ell \Pi\, F\, G$ will have to refer to reducible terms of type F, but we may assume $\Delta \Vdash_\ell a : F/_{\mathcal{U}}$ to be already defined since $\mathcal{U} :: \Gamma \Vdash_\ell F$ is a subderivation mentioned in the premise of the rule. This definition scheme of interleaving induction and recursion is called *induction-recursion* [Dybjer 2000]. While the dependence on derivation is necessary for the well-foundedness of the definition process, the exact shape of the derivation should be irrelevant for the reducibility judgements. We prove this *a posteriori* in Lemma 3.5.

We now to proceed to give the rules for the inductive predicate $\boxed{\Gamma \Vdash_\ell A}$, each rule followed by the clauses for the judgements that are simultaneously defined by recursion on its derivation $\mathcal{T}$: $\boxed{\Gamma \Vdash_\ell A = B/_{\mathcal{T}}}$ and $\boxed{\Gamma \Vdash_\ell t : A/_{\mathcal{T}}}$ and $\boxed{\Gamma \Vdash_\ell t = u : A/_{\mathcal{T}}}$. In the following, we often refer to the package of these four judgements as *the logical relation*. The logical relation is implemented under an external well-founded induction on ℓ, so that definitions for large types can make use of the relations for small ones. Most rules are identical for both levels, as most of the type formers belong to both small and large types—except for the universe U which is necessarily a large type.

Universe. U shall be a large reducible type.

$$\frac{\vdash \Gamma}{\Gamma \Vdash_\ell \mathsf{U}}\ \ell'{<}\ell$$

Since we have no reduction on the level of large types, and hence, nothing reduces to U, this rule is trivially closed under weak head expansion. The side condition $\ell'{<}\ell$ could be written as $\ell = 1$. But since we doing a well-founded induction on ℓ, our formulation is convenient, as it directly gives us the induction hypothesis for ℓ'.

For derivations $\mathcal{T}$ built with that rule we define:

- $\Gamma \Vdash_\ell \mathsf{U} = B/_{\mathcal{T}}$ iff $B \equiv \mathsf{U}$. *This means that only* U *is reducibly equal to itself.*
- $\Gamma \Vdash_\ell t : \mathsf{U}/_{\mathcal{T}}$ *(t is a reducible member of* U*)* iff the following hold:
  1. There exists some $\bar{t}$ such that $\Gamma \vdash t :\longrightarrow^*: \bar{t} : \mathsf{U}$ and $\Gamma \vdash \bar{t} \cong \bar{t} : \mathsf{U}$, *meaning t has a reflexive whnf.*
  2. $\Gamma \Vdash_{\ell'} t$, *which means that t is a reducible small type (already defined by induction hypothesis).*
- $\Gamma \Vdash_\ell t = u : \mathsf{U}/_{\mathcal{T}}$ *(t and u are reducibly equal members of* U*)* iff:
  1. There are $\bar{t}$ and $\bar{u}$ such that $\Gamma \vdash t :\longrightarrow^*: \bar{t} : \mathsf{U}$ and $\Gamma \vdash u :\longrightarrow^*: \bar{u} : \mathsf{U}$ and $\Gamma \vdash \bar{t} \cong \bar{u} : \mathsf{U}$. *This means that t and u have whnfs related by the generic equality.*
  2. $\mathcal{U} :: \Gamma \Vdash_{\ell'} t$ and $\Gamma \Vdash_{\ell'} u$ and $\Gamma \Vdash_{\ell'} t = u/_{\mathcal{U}}$, *meaning that t and u are reducibly equal small types.*

Neutrals. Any type that has a neutral whnf N shall be reducible. Since generic equality is not reflexive in general, we also require N to be reflexive.

$$\frac{\Gamma \vdash A :\longrightarrow^*: N \qquad \Gamma \vdash N \sim N : \mathsf{U}}{\Gamma \Vdash_\ell A}$$

Here, $\boxed{\Gamma \vdash A :\longrightarrow^*: N}$ is a short-hand for the conjunction of $\Gamma \vdash A$ and $\Gamma \vdash N$ and $\Gamma \vdash A \longrightarrow^* N$. For any relation $\circledR$, we will from now on use $\Gamma \vdash A :\circledR: B$ to signify the conjunction of $\Gamma \vdash A$ and $\Gamma \vdash B$ and $\Gamma \vdash A \circledR B$, and use $\Gamma \vdash t :\circledR: u : A$ to signify the conjunction of $\Gamma \vdash t : A$ and $\Gamma \vdash u : A$ and $\Gamma \vdash t \circledR u : A$. We introduce the extra well-formedness premises because we cannot yet prove $\Gamma \vdash A$ or $\Gamma \vdash B$ from most of our relations, for instance, neither from $\Gamma \vdash A = B$ nor from $\Gamma \vdash A \longrightarrow^* B$. The lacking property is known as *syntactic validity* [Harper and Pfenning 2005] or *presupposition* [Goguen 2000]. It is non-trivial to derive because of the asymmetry present in some of the inference rules, like the congruence rule for application or Π-types. We will obtain syntactic validity (Thm. 3.26) later as a consequence of the fundamental theorem.

Given a derivation $\mathcal{T} :: \Gamma \Vdash_\ell A$ built by the rule for neutral types we define:

- $\Gamma \Vdash_\ell A = B/_{\mathcal{T}}$ iff there is a neutral M such that $\Gamma \vdash B :\longrightarrow^*: M$ and $\Gamma \vdash N \sim M : \mathsf{U}$.
  Neutral types are reducibly equal to types that have an equal whnf up to generic neutral equality.
- $\Gamma \Vdash_\ell t : A/_{\mathcal{T}}$ iff there is a neutral n such that $\Gamma \vdash t :\longrightarrow^*: n : N$ and $\Gamma \vdash n \sim n : N$.
  Neutral types are inhabited by terms that have a neutral whnf.
- $\Gamma \Vdash_\ell t = u : A/_{\mathcal{T}}$ iff $\Gamma \vdash t :\longrightarrow^*: n : N$ and $\Gamma \vdash u :\longrightarrow^*: m : N$ and $\Gamma \vdash n \sim m : N$ for some neutrals n, m.
  At neutral type, objects are reducibly equal if they have the same whnf up to generic neutral equality.

Natural Numbers. $\mathbb{N}$ and its well-formed weak head expansions are reducible types.

$$\frac{\Gamma \vdash A :\longrightarrow^*: \mathbb{N}}{\Gamma \Vdash_\ell A}$$

For a derivation $\mathcal{T}$ built by that rule we define:

- $\Gamma \Vdash_\ell A = B/_{\mathcal{T}}$ iff $\Gamma \vdash B \longrightarrow^* \mathbb{N}$.
- $\Gamma \Vdash_\ell t : A/_{\mathcal{T}}$ iff $\boxed{\Gamma \Vdash \mathbb{N}\ t}$ which is defined to hold iff there exists a whnf $\bar{t}$ such that:
  (1) $\Gamma \vdash t :\longrightarrow^*: \bar{t} : \mathbb{N}$
  (2) $\Gamma \vdash \bar{t} \cong \bar{t} : \mathbb{N}$
  (3) $\Gamma \Vdash \mathbb{N}_{\mathsf{w}}\ \bar{t}$, which is inductively defined by the rules

$$\frac{}{\Gamma \Vdash \mathbb{N}_{\mathsf{w}}\ \mathsf{zero}} \qquad \frac{\Gamma \Vdash \mathbb{N}\ t}{\Gamma \Vdash \mathbb{N}_{\mathsf{w}}\ \mathsf{suc}\ t} \qquad \frac{\Gamma \vdash n : \mathbb{N} \qquad \Gamma \vdash n \sim n : \mathbb{N}}{\Gamma \Vdash \mathbb{N}_{\mathsf{w}}\ n}$$

  With that inductive definition, we model the different properties of the possible constructions of natural numbers. For the $\mathsf{suc}\ t$ case, we require that t is a reducible natural number, so that we can properly use natural recursion on the number. For the neutral case, we require that n is well-formed and neutrally reflexive. Finally, for the zero case, we have no additional requirements.
- $\Gamma \Vdash_\ell t = u : A/_{\mathcal{T}}$ iff $\boxed{\Gamma \Vdash \mathbb{N}\ t = u}$ which is defined to hold iff there exist whnfs $\bar{t}$ and $\bar{u}$ such that:
  (1) $\Gamma \vdash t :\longrightarrow^*: \bar{t} : \mathbb{N}$
  (2) $\Gamma \vdash u :\longrightarrow^*: \bar{u} : \mathbb{N}$
  (3) $\Gamma \vdash \bar{t} \cong \bar{u} : \mathbb{N}$
  (4) $\Gamma \Vdash \mathbb{N}_{\mathsf{w}}\ \bar{t} = \bar{u}$, which is analogously to $\Gamma \Vdash \mathbb{N}_{\mathsf{w}}\ \bar{t}$ defined inductively by the rules

$$\frac{}{\Gamma \Vdash \mathbb{N}_{\mathsf{w}}\ \mathsf{zero} = \mathsf{zero}} \qquad \frac{\Gamma \Vdash \mathbb{N}\ t = u}{\Gamma \Vdash \mathbb{N}_{\mathsf{w}}\ \mathsf{suc}\ t = \mathsf{suc}\ u} \qquad \frac{\Gamma \vdash n \sim m : \mathbb{N}}{\Gamma \Vdash \mathbb{N}_{\mathsf{w}}\ n = m}$$

Function Spaces. A type with whnf $\Pi\, F\, G$ is reducible if both F and G are reducible in a way detailed in the following rule:

$$\frac{\begin{array}{l}\Gamma \vdash A :\longrightarrow^*: \Pi\, F\, G \qquad \Gamma \vdash F \qquad \Gamma, F \vdash G \\ \mathcal{U} :: (\forall \rho : \Delta \leq \Gamma.\ \ \Delta \Vdash_\ell F[\rho]) \qquad \mathcal{T} :: (\forall \rho : \Delta \leq \Gamma.\ \ \Delta \Vdash_\ell a : F[\rho]/_{\mathcal{U}(\rho)} \implies \Delta \Vdash_\ell G[\rho, a]) \\ \forall \rho : \Delta \leq \Gamma,\ u :: \Delta \Vdash_\ell a : F[\rho]/_{\mathcal{U}(\rho)}. \\ \qquad \Delta \Vdash_\ell b : F[\rho]/_{\mathcal{U}(\rho)} \implies \Delta \Vdash_\ell a = b : F[\rho]/_{\mathcal{U}(\rho)} \implies \Delta \Vdash_\ell G[\rho, a] = G[\rho, b]/_{\mathcal{T}(\rho, u)}\end{array}}{\Gamma \Vdash_\ell A}$$

The notation "::" denotes the assignment of a type to a variable in the meta-theory. It is the same as Agda's elementhood relation ":" and can for instance be used in $\forall u :: \mathcal{U}.\ \mathcal{T}(u)$ which means "for all u of type $\mathcal{U}$, there is $\mathcal{T}(u)$." For weakening, we abbreviate $(\forall \rho :: \mathsf{Wk}.\ \ \rho : \Gamma \leq \Delta \implies P)$ to $(\forall \rho : \Gamma \leq \Delta.\ \ P)$.

For A to be a reducible Π-type, we first require A to reduce to $\Pi\, F\, G$, where F and G are well-formed. Secondly, F has to be reducible under any weakening ρ. Thirdly, for any reducible term a in $F[\rho]$, the type $G[\rho, a]$ has to be reducible. Finally, if a and b are reducibly equal of type $F[\rho]$, the types $G[\rho, a]$ and $G[\rho, b]$ are reducibly equal as well. The requirement of reducibility under weakening is needed for proving reducibility under binder cases like λ and Π, and the last two conditions model that G is a F-indexed family of types.

Given a derivation $\mathcal{T}$ built by the function space rule we define:

- $\Gamma \Vdash_\ell A = B/_{\mathcal{T}}$ iff there are F' and G' such that:
  (1) $\Gamma \vdash B \longrightarrow^* \Pi\, F'\, G'$
  (2) $\Gamma \vdash \Pi\, F\, G \cong \Pi\, F'\, G'$
  (3) $\forall \rho : \Delta \leq \Gamma.\ \ \Delta \Vdash_\ell F[\rho] = F'[\rho]/_{\mathcal{T}(\rho)}$
  (4) $\forall \rho : \Delta \leq \Gamma,\ u :: \Delta \Vdash_\ell a : F[\rho]/_{\mathcal{T}(\rho)}.\ \ \Delta \Vdash_\ell G[\rho, a] = G'[\rho, a]/_{\mathcal{U}(\rho,u)}$
  For B to be reducibly equal to A, the type B has to reduce to $\Pi\, F'\, G'$ which is equal in generic equality to the $\Pi\, F\, G$, the type A reduces to. We also require that under weakening ρ, the types $F[\rho]$ and $F'[\rho]$ have to be reducibly equal and for a of type $F[\rho]$, $G[\rho, a]$ and $G'[\rho, a]$ are reducibly equal.
- $\Gamma \Vdash_\ell t : A/_{\mathcal{T}}$ iff there is a $\bar{t}$ such that:
  (1) $\Gamma \vdash t :\longrightarrow^*: \bar{t} : \Pi\, F\, G$
  (2) $\Gamma \vdash \bar{t} \cong \bar{t} : \Pi\, F\, G$
  (3) $\forall \rho : \Delta \leq \Gamma,\ u :: \Delta \Vdash_\ell a : F[\rho]/_{\mathcal{T}(\rho)}.\ \ \Delta \Vdash_\ell \bar{t}[\rho]\, a : G[\rho, a]/_{\mathcal{U}(\rho,u)}$
  (4) $\forall \rho : \Delta \leq \Gamma,\ u :: \Delta \Vdash_\ell a : F[\rho]/_{\mathcal{T}(\rho)}.\ \ \Delta \Vdash_\ell b : F[\rho]/_{\mathcal{T}(\rho)} \implies \Delta \Vdash_\ell a = b : F[\rho]/_{\mathcal{T}(\rho)} \implies$
  $\Delta \Vdash_\ell \bar{t}[\rho]\, a = \bar{t}[\rho]\, b : G[\rho, a]/_{\mathcal{U}(\rho,u)}$
  The requirement 3 effectively says that under weakening ρ, given a term a of type $F[\rho]$, we can apply a to $\bar{t}[\rho]$ with type $G[\rho, a]$. Requirement 4 basically says the same thing, but for equality. Also note that the reducibility of b is not necessary to complete the definition, but will be helpful in some of the proofs.
- $\Gamma \Vdash_\ell t = u : A/_{\mathcal{T}}$ iff there is a $\bar{t}$ and a $\bar{u}$ such that:
  (1) $\Gamma \vdash t :\longrightarrow^*: \bar{t} : \Pi\, F\, G$
  (2) $\Gamma \vdash u :\longrightarrow^*: \bar{u} : \Pi\, F\, G$
  (3) $\Gamma \vdash \bar{t} \cong \bar{u} : \Pi\, F\, G$
  (4) $\Gamma \Vdash_\ell t : A/_{\mathcal{T}}$
  (5) $\Gamma \Vdash_\ell u : A/_{\mathcal{T}}$
  (6) $\forall \rho : \Delta \leq \Gamma,\ u :: \Delta \Vdash_\ell a : F[\rho]/_{\mathcal{T}(\rho)}.\ \ \Delta \Vdash_\ell \bar{t}[\rho]\, a = \bar{u}[\rho]\, a : G[\rho, a]/_{\mathcal{U}(\rho,u)}$
  In requirement 6, we say that under weakening ρ, given a term a of type $F[\rho]$, $\bar{t}[\rho]$ and $\bar{u}[\rho]$ are equal under application of a. This is necessary to prove congruence of application.

Embedding. Reducible small types can also be viewed as reducible large types.

$$\frac{\mathcal{U} :: \Gamma \Vdash_{\ell'} A}{\Gamma \Vdash_\ell A}\ \ell' < \ell$$

Given a derivation $\mathcal{T}$ built by that rule:

- $\Gamma \Vdash_\ell A = B/_{\mathcal{T}}$ iff $\Gamma \Vdash_{\ell'} A = B/_{\mathcal{U}}$.
- $\Gamma \Vdash_\ell t : A/_{\mathcal{T}}$ iff $\Gamma \Vdash_{\ell'} t : A/_{\mathcal{U}}$. *This allows type level embedding of reducible terms.*
- $\Gamma \Vdash_\ell t = u : A/_{\mathcal{T}}$ iff $\Gamma \Vdash_{\ell'} t = u : A/_{\mathcal{U}}$.

3.3 Properties of the Logical Relation

In this section, we establish some basic properties of the logical relation. We start by reflexivity, showing that reducible objects are well-formed, and logically related objects are generically equal.

Lemma 3.1 (Reflexivity). *Given* $\mathscr{T} :: \Gamma \Vdash_\ell A$ *then:*

(1) $\Gamma \Vdash_\ell A = A/_\mathscr{T}$.
(2) $\Gamma \Vdash_\ell t : A/_\mathscr{T}$ *then* $\Gamma \Vdash_\ell t = t : A/_\mathscr{T}$.

Proof. By induction on $\mathscr{T}$. □

Lemma 3.2 (Escape lemma). *Given* $\mathscr{T} :: \Gamma \Vdash_\ell A$ *then:*

(1) $\Gamma \vdash A$.
(2) *If* $\Gamma \Vdash_\ell A = B/_\mathscr{T}$ *then* $\Gamma \vdash A \cong B$.
(3) *If* $\Gamma \Vdash_\ell t : A/_\mathscr{T}$ *then* $\Gamma \vdash t : A$.
(4) *If* $\Gamma \Vdash_\ell t = u : A/_\mathscr{T}$ *then* $\Gamma \vdash t \cong u : A$.

Proof. By induction on $\mathscr{T}$ and reduction being subsumed by equality (2.2). □

For much of the following proofs, we will need to use induction on two or more derivations of type reducibility for types that are reducibly equal. As there are four different cases for typing derivations (ignoring the embedding case, as one can immediately recurse on its premise), with two derivations we would have 16 cases and with three derivation we would have 64 cases. However, it turns out that a lot of cases can be refuted by the reduction properties. We would like to avoid repeating this refutation process and instead only prove it once.

We introduce what we call the shape view, which is an inductively defined relation on two or more type reducibility derivations such that an instance of the view is only valid if the derivations have a compatible inductive structure, i.e. either they are built with the same inference rule or one of them is an embedding, for which we use the structure of the premise. For example, if we have two type reducibility derivations $\mathscr{T}$ and $\mathscr{U}$, and $\mathscr{T}$ is built using the natural number type rule:

$$\frac{\Gamma \vdash A :\longrightarrow^{*}: \mathbb{N}}{\Gamma \Vdash_\ell A}$$

Then for $\mathscr{U}$ to be related by the view with $\mathscr{T}$, it must be built by the same rule either directly or inside the embedding case. Pattern matching on proofs of the view then allows us to consider only the compatible cases and not have to worry everywhere about the ones where e.g. both $A \longrightarrow^{*} \Pi\, F\, G$ and $A \longrightarrow^{*} \mathbb{N}$.

Lemma 3.3 (Shape view construction). *Given* $\mathscr{T} :: \Gamma \Vdash_\ell A$ *and* $\mathscr{T}' :: \Gamma \Vdash_{\ell'} B$*, if* $\Gamma \Vdash_\ell A = B/_\mathscr{T}$ *then there is an shape view of* $\mathscr{T}$ *and* $\mathscr{T}'$.

Proof. By induction on $\mathscr{T}$ and $\mathscr{T}'$, as Whnfs do not reduce (2.4) and reduction is deterministic (2.5). □

Corollary 3.4 (Reflexive shape view construction). *Given* $\mathscr{T} :: \Gamma \Vdash_\ell A$ *and* $\mathscr{T}' :: \Gamma \Vdash_{\ell'} A$*, there is an shape view of* $\mathscr{T}$ *and* $\mathscr{T}'$.

Proof. Directly by reflexivity of reducible equality (3.1) and shape view construction (3.3). □

Lemma 3.5 (Irrelevance). *Given* $\mathscr{T} :: \Gamma \Vdash_\ell A$ *and* $\mathscr{T}' :: \Gamma \Vdash_{\ell'} A$ *then:*

(1) *If* $\Gamma \Vdash_\ell A = B/_\mathscr{T}$ *then* $\Gamma \Vdash_{\ell'} A = B/_{\mathscr{T}'}$.
(2) *If* $\Gamma \Vdash_\ell t : A/_\mathscr{T}$ *then* $\Gamma \Vdash_{\ell'} t : A/_{\mathscr{T}'}$.
(3) *If* $\Gamma \Vdash_\ell t = u : A/_\mathscr{T}$ *then* $\Gamma \Vdash_{\ell'} t = u : A/_{\mathscr{T}'}$.

Proof. By induction on the shape view of $\mathcal{T}$ and $\mathcal{T}'$ given by reflexive shape view construction (3.4) and by determinism of reduction (2.5). □

Even if $\mathcal{T}$ and $\mathcal{T}'$ above can actually differ, e.g. by containing different typing derivations, our irrelevance lemma shows that for the recursively defined judgements, the specific proof of type reducibility $\mathcal{T}$ does not matter. We can therefore use these judgements more freely as we no longer need to refer to a specific type reducibility derivation.

Based on this intuition, we will from here on drop the type derivation from the reducibility judgements and instead implicitly state that there exists such a derivation for a judgement. More formally, we let $\boxed{\Gamma \Vdash_\ell J : A}$ mean that $\Gamma \Vdash_\ell J \,:\, A/_{\mathcal{T}}$ holds for some $\mathcal{T} \,::\, \Gamma \Vdash_\ell A$. However, this definition is not practical in the implementation as often judgements will share the same type, and thus using only the above definition would introduce extra complications. Therefore, in the implementation, we use derivations explicitly.

Lemma 3.6 (Weakening). *For all judgements J of the logical relation, given $\rho : \Delta \leq \Gamma$ and $\Gamma \Vdash_\ell J$ then $\Delta \Vdash_\ell J[\rho]$.*

Proof. By induction on the derivation of type A and by weakening of well-formed expressions (2.7) and irrelevance (3.5). □

Lemma 3.7 (Conversion). *Given $\Gamma \Vdash_\ell A :=: B$ then:*

(1) *$\Gamma \Vdash_\ell t : A$ iff $\Gamma \Vdash_\ell t : B$.*
(2) *$\Gamma \Vdash_\ell t = u : A$ iff $\Gamma \Vdash_\ell t = u : B$.*

Proof. By induction on the shape view (3.3) of equal types A and B, using determinism of reduction (2.5) and irrelevance 3.5. The two directions of iff are proven simultaneously. □

Lemma 3.8 (Symmetry).

(1) *If $\Gamma \Vdash_\ell A :=: B$ then $\Gamma \Vdash_\ell B = A$.*
(2) *If $\Gamma \Vdash_\ell t = u : A$ then $\Gamma \Vdash_\ell u = t : A$.*

Proof. By induction on the shape view (3.3) of the equal types A and B, using determinism of reduction (2.5), irrelevance (3.5) and conversion for reducible equality (3.7). □

Lemma 3.9 (Transitivity).

(1) *If $\Gamma \Vdash_\ell A :=: A'$ and $\Gamma \Vdash_\ell A' :=: A''$ then $\Gamma \Vdash_\ell A = A''$.*
(2) *If $\Gamma \Vdash_\ell t = t' : A$ and $\Gamma \Vdash_\ell t' = t'' : A$ then $\Gamma \Vdash_\ell t = t'' : A$.*

Proof. By induction on the shape view (3.3) of type A, A' and A'', using determinism of reduction (2.5), irrelevance (3.5) and conversion of reducible equality (3.7). □

Lemma 3.10 (Neutrals are reducible). *Let $\Gamma \Vdash_\ell A$ and $\Gamma \vdash n : A$.*

(1) *If $\Gamma \vdash n :\sim: n : A$ then $\Gamma \Vdash_\ell n : A$.*
(2) *If $\Gamma \vdash n' : A$ and $\Gamma \vdash n :\sim: n' : A$ then $\Gamma \Vdash_\ell n = n' : A$.*

Proof. By induction on the derivation of type A and by well-formed weakening (2.7), reduction is subsumed by equality (2.2) and escape (3.2). □

Lemma 3.11 (Weak head expansion).

(1) *If $\Gamma \Vdash_\ell B$ and $\Gamma \vdash A \longrightarrow^* B$ then $\Gamma \Vdash_\ell A :=: B$.*
(2) *If $\Gamma \Vdash_\ell u : B$ and $\Gamma \vdash t \longrightarrow^* u : B$ then $\Gamma \Vdash_\ell t :=: u : B$.*

Proof. By induction on the reducibility of B, subject typing (2.3) and reflexivity (3.1). □

Lemma 3.12 (Application reducibility). *Given* $\Gamma \Vdash_\ell \Pi\, F\, G$ *and* $\Gamma \Vdash_\ell u : F$ *and* $\Gamma \Vdash_\ell G[u]$ *then:*

(1) *If* $\Gamma \Vdash_\ell t : \Pi\, F\, G$ *then* $\Gamma \Vdash_\ell t\, u : G[u]$.
(2) *If* $\Gamma \Vdash_\ell t = t' : \Pi\, F\, G$ *and* $\Gamma \Vdash_\ell u :=: u' : F$ *then* $\Gamma \Vdash_\ell t\, u = t'\, u' : G[u]$.

Proof. For case (1): By applying u instead of a in the following premise of the reducible term t:

$$\forall \rho : \Delta \le \Gamma.\;\; \Delta \Vdash_\ell a : F[\rho] \implies \Delta \Vdash_\ell \bar{t}[\rho]\, a : G[\rho, a]$$

Case (2) is solved similarly. This proof is accomplished with escape (3.2), irrelevance (3.5), conversion (3.7), symmetry (3.8), transitivity (3.9) and weak head expansion (3.11). □

3.4 Validity Judgements

In Section 3.2, we have introduced reducible objects $\Gamma \Vdash_\ell t : A$, which are guaranteed to be well-typed, $\Gamma \vdash t : A$, but we cannot prove the converse yet, i.e., that each well-typed term is reducible. In other words, reducible objects do not directly provide a model of $\lambda^{\Pi\mathsf{U}\mathbb{N}}$, in the sense that the interpretation function cannot be implemented by a naive induction on the derivation. The simplest counterexample is the typing rule for λ-abstraction:

$$\frac{\Gamma, F \vdash t : G}{\Gamma \vdash \lambda t : \Pi\, F\, G}$$

By definition of $\Gamma \Vdash_\ell \lambda t : \Pi\, F\, G$, we have to show, amongst others, that for any $\rho : \Delta \le \Gamma$ and any $\Delta \Vdash_\ell a : F$, we have $\Delta \Vdash_\ell (\lambda t)[\rho]\, a : G[\rho, a]$. Using weak head expansion, it is sufficient to show $\Delta \Vdash_\ell t[\rho, a] : G[\rho, a]$. However, our induction hypothesis $\Gamma, F \Vdash_\ell t : G$ is too weak to prove that. We would need to substitute with $\Delta \vdash (\rho, a) : (\Gamma, F)$, but reducible terms are not closed under substitution. The way out is by blunt force: we define *valid* objects $\Gamma \Vdash^{\mathsf{v}}_\ell t : A$ to be objects that are reducible under substitution with reducible objects. Those then model our syntax, i.e., we can prove the fundamental theorem (3.21) which states that $\Gamma \vdash t : A$ implies $\Gamma \Vdash^{\mathsf{v}}_\ell t : A$. Applying valid object t to the identity substitution (3.17), we obtain reducibility $\Gamma \Vdash_\ell t : A$ and all the good properties that follow from it.

For each of the reducibility judgments we then define their validity counterparts. We will make use of judgments for validity of contexts $\mathscr{G} :: \Vdash^{\mathsf{v}} \Gamma$, reducible substitutions $\mathscr{S} :: \Delta \Vdash^{\mathsf{s}} \sigma : \Gamma/_{\mathscr{G}}$ and their equality $\Delta \Vdash^{\mathsf{s}} \sigma = \sigma' : \Gamma/_{\mathscr{G}}/_{\mathscr{S}}$ which we will define just after through induction-recursion.

First we define valid types as those that are reducible under any reducible substitution and respect their equality. $\boxed{\Gamma \Vdash^{\mathsf{v}}_\ell A/_{\mathscr{G}}}$ iff for all $\mathscr{S} :: \Delta \Vdash^{\mathsf{s}} \sigma : \Gamma/_{\mathscr{G}}$ it holds that:

(1) $\mathscr{U} :: \Delta \Vdash_\ell A[\sigma]$
(2) $\Delta \Vdash^{\mathsf{s}} \sigma' : \Gamma/_{\mathscr{G}}$ and $\Delta \Vdash^{\mathsf{s}} \sigma = \sigma' : \Gamma/_{\mathscr{G}}/_{\mathscr{S}}$ imply $\Delta \Vdash_\ell A[\sigma] = A[\sigma']/_{\mathscr{U}}$

Note that we will use the above enumeration indexes as projections, e.g., given $\mathscr{T} :: \Gamma \Vdash^{\mathsf{v}}_\ell A/_{\mathscr{G}}$ the notation $\mathscr{T}(\mathscr{S}).1$ stands for the proof of that $A[\sigma]$ is reducible.

Given $\mathscr{T}$ as above we define valid type equality, valid terms and valid term equality:

$\boxed{\Gamma \Vdash^{\mathsf{v}}_\ell A = B/_{\mathscr{G}}/_{\mathscr{T}}}$ iff for all $\mathscr{S} :: \Delta \Vdash^{\mathsf{s}} \sigma : \Gamma/_{\mathscr{G}}$ we have $\Delta \Vdash_\ell A[\sigma] = B[\sigma]/_{\mathscr{T}(\mathscr{S}).1}$.

$\boxed{\Gamma \Vdash^{\mathsf{v}}_\ell t : A/_{\mathscr{G}}/_{\mathscr{T}}}$ iff for all $\mathscr{S} :: \Delta \Vdash^{\mathsf{s}} \sigma : \Gamma/_{\mathscr{G}}$ it holds that:

(1) $\Delta \Vdash_\ell t[\sigma] : A[\sigma]/_{\mathscr{T}(\mathscr{S}).1}$
(2) $\Delta \Vdash^{\mathsf{s}} \sigma' : \Gamma/_{\mathscr{G}}$ and $\Delta \Vdash^{\mathsf{s}} \sigma = \sigma' : \Gamma/_{\mathscr{G}}/_{\mathscr{S}}$ imply $\Delta \Vdash_\ell t[\sigma] = t[\sigma'] : A[\sigma]/_{\mathscr{T}(\mathscr{S}).1}$

$\boxed{\Gamma \Vdash^{\mathsf{v}}_\ell t = u : A/_{\mathscr{G}}/_{\mathscr{T}}}$ iff for all $\mathscr{S} :: \Delta \Vdash^{\mathsf{s}} \sigma : \Gamma/_{\mathscr{G}}$ we have $\Delta \Vdash_\ell t[\sigma] = u[\sigma] : A[\sigma]/_{\mathscr{T}(\mathscr{S}).1}$.

By induction-recursion, we define $\boxed{\Vdash^{\mathsf{v}} \Gamma}$ inductively by two rules, one for the empty context and one for context expansion, and simultaneously define $\boxed{\Delta \Vdash^{\mathsf{s}} \sigma : \Gamma/_{\mathscr{G}}}$ and $\boxed{\Delta \Vdash^{\mathsf{s}} \sigma = \sigma' : \Gamma/_{\mathscr{G}}/_{\mathscr{S}}}$ by recursion on the derivation $\mathcal{G} :: \Vdash^{\mathsf{v}} \Gamma$, the second depending on $\mathcal{S} :: \Delta \Vdash^{\mathsf{s}} \sigma : \Gamma/_{\mathcal{G}}$. For the following definitions, let Δ be a well-formed context.

Empty Context.

$$\frac{}{\Vdash^{\mathsf{v}} \epsilon}$$

For derivations $\mathcal{G}$ built with that rule we define that $\Delta \Vdash^{\mathsf{s}} \sigma : \epsilon/_{\mathcal{G}}$ holds unconditionally and for all $\mathcal{S} :: \Delta \Vdash^{\mathsf{s}} \sigma : \epsilon/_{\mathcal{G}}$ we have $\Delta \Vdash^{\mathsf{s}} \sigma = \sigma' : \epsilon/_{\mathcal{G}}/_{\mathcal{S}}$.

Extended Context.

$$\frac{\mathcal{G}' :: \Vdash^{\mathsf{v}} \Gamma \qquad \mathcal{T} :: \Gamma \Vdash^{\mathsf{v}}_{\ell} A/_{\mathcal{G}'}}{\Vdash^{\mathsf{v}} \Gamma, A}$$

For derivations $\mathcal{G}$ built with that rule we define:

- $\Delta \Vdash^{\mathsf{s}} \sigma : \Gamma, A/_{\mathcal{G}}$ iff
  (1) $\mathcal{S}' :: \Delta \Vdash^{\mathsf{s}} \sigma \circ \uparrow\mathrm{id} : \Gamma/_{\mathcal{G}'}$ *meaning that the tail of σ is reducible.*
  (2) $\Delta \Vdash_{\ell} i_0[\sigma] : A[\sigma \circ \uparrow\mathrm{id}]/_{\mathcal{T}(\mathcal{S}').1}$ *meaning that the head of σ is reducible.*
- For all $\mathcal{S} :: \Delta \Vdash^{\mathsf{s}} \sigma : \Gamma, A/_{\mathcal{G}}$ we have $\Delta \Vdash^{\mathsf{s}} \sigma = \sigma' : \Gamma, A/_{\mathcal{G}}/_{\mathcal{S}}$ iff
  (1) $\Delta \Vdash^{\mathsf{s}} \sigma \circ \uparrow\mathrm{id} = \sigma' \circ \uparrow\mathrm{id} : \Gamma/_{\mathcal{G}'}/_{\mathcal{S}.1}$
  (2) $\Delta \Vdash_{\ell} i_0[\sigma] = i_0[\sigma'] : A[\sigma \circ \uparrow\mathrm{id}]/_{\mathcal{T}(\mathcal{S}.1)}$
  Note that $\mathcal{S}.1$ stands for the proof that the tail of σ is reducible.

3.5 Properties of the Validity Judgements

In preparation for the fundamental theorem, we prove a few properties about validity and reducible substitutions.

As the validity judgements are defined using induction-recursion, we will—similarly to the logical relation—prove context and typing derivation irrelevance.

Lemma 3.13 (Irrelevance). *Let $\mathcal{G} :: \Vdash^{\mathsf{v}} \Gamma$ and $\mathcal{G}' :: \Vdash^{\mathsf{v}} \Gamma$.*

(1) *If $\mathcal{S} :: \Delta \Vdash^{\mathsf{s}} \sigma : \Gamma/_{\mathcal{G}}$ then there is a derivation $\mathcal{S}' :: \Delta \Vdash^{\mathsf{s}} \sigma : \Gamma/_{\mathcal{G}'}$.*
If further $\Delta \Vdash^{\mathsf{s}} \sigma = \sigma' : \Gamma/_{\mathcal{G}}/_{\mathcal{S}}$ then $\Delta \Vdash^{\mathsf{s}} \sigma = \sigma' : \Gamma/_{\mathcal{G}'}/_{\mathcal{S}'}$.
(2) *If $\mathcal{T} :: \Gamma \Vdash^{\mathsf{v}}_{\ell} A/_{\mathcal{G}}$ then there is a derivation $\mathcal{T}' :: \Gamma \Vdash^{\mathsf{v}}_{\ell} A/_{\mathcal{G}'}$. It also follows that:*
  (a) *If $\Gamma \Vdash^{\mathsf{v}}_{\ell} A = B/_{\mathcal{G}}/_{\mathcal{T}}$ then $\Gamma \Vdash^{\mathsf{v}}_{\ell} A = B/_{\mathcal{G}'}/_{\mathcal{T}'}$.*
  (b) *If $\Gamma \Vdash^{\mathsf{v}}_{\ell} t : A/_{\mathcal{G}}/_{\mathcal{T}}$ then $\Gamma \Vdash^{\mathsf{v}}_{\ell} t : A/_{\mathcal{G}'}/_{\mathcal{T}'}$.*
  (c) *If $\Gamma \Vdash^{\mathsf{v}}_{\ell} t = u : A/_{\mathcal{G}}/_{\mathcal{T}}$ then $\Gamma \Vdash^{\mathsf{v}}_{\ell} t = u : A/_{\mathcal{G}'}/_{\mathcal{T}'}$.*

Proof. By induction on $\mathcal{G}$ and $\mathcal{G}'$, using irrelevance for the reducibility judgements (3.5). □

Derivation irrelevance justifies that we from here on drop the context and type derivation arguments from the validity judgements, as we do for reducibility. Further, when we state $\Delta \Vdash^{\mathsf{s}} \sigma : \Gamma$ or $\Delta \Vdash^{\mathsf{s}} \sigma = \sigma' : \Gamma$ we presuppose $\Vdash^{\mathsf{v}} \Gamma$.

The escape lemma extends to reducible substitutions:

Lemma 3.14 (Escape for substitutions).

(1) *If $\Delta \Vdash^{\mathsf{s}} \sigma : \Gamma$ then $\Delta \vdash \sigma : \Gamma$.*
(2) *If $\Delta \Vdash^{\mathsf{s}} \sigma = \sigma' : \Gamma$ then $\Delta \vdash \sigma = \sigma' : \Gamma$.*

Proof. By induction on $\Vdash^{\mathsf{v}} \Gamma$, using the escape lemma (3.2). □

The following lemma is needed to make the fundamental theorem go through the binder cases (λ, Π), in particular, to establish the well-formedness and well-typedness conditions there. It relies on weakening for the reducibility judgements, and is the reason why our logical relation is *Kripke* in the first place.

Lemma 3.15 (Substitution weakening). *Given* $\rho : \Delta' \leq \Delta$ *and* $\Delta \Vdash^{\mathsf{s}} \sigma : \Gamma$ *then* $\Delta' \Vdash^{\mathsf{s}} \rho \circ \sigma : \Gamma$. *If further* $\Delta \Vdash^{\mathsf{s}} \sigma = \sigma' : \Gamma$ *then* $\Delta' \Vdash^{\mathsf{s}} \rho \circ \sigma = \rho \circ \sigma' : \Gamma$.

Proof. By induction on $\Vdash^{\mathsf{v}} \Gamma$, using weakening for the reducibility judgements (3.6). □

Lemma 3.16 (Substitution lifting). *Given* $\Gamma \Vdash^{\mathsf{v}}_{\ell} A/_{\mathscr{G}}$ *and* $\Delta \Vdash^{\mathsf{s}} \sigma : \Gamma$ *then* $\Delta, A[\sigma] \Vdash^{\mathsf{s}} \Uparrow\sigma : \Gamma, A$. *If further* $\Delta \Vdash^{\mathsf{s}} \sigma = \sigma' : \Gamma$ *then* $\Delta, A[\sigma] \Vdash^{\mathsf{s}} \Uparrow\sigma = \Uparrow\sigma' : \Gamma, A$.

Proof. By reducibility of neutrals (3.10) and substitution weakening (3.15). □

Lemma 3.17 (Identity substitution). *If* $\Vdash^{\mathsf{v}} \Gamma$ *then* $\vdash \Gamma$ *and* $\Gamma \Vdash^{\mathsf{s}} \mathsf{id} : \Gamma$.

Proof. By induction on the validity of Γ, escape (3.2), irrelevance (3.5) and substitution lifting (3.16). □

Lemma 3.18 (Substitution equivalence). $\Delta \Vdash^{\mathsf{s}} \_ = \_ : \Gamma$ *is an equivalence relation on valid substitutions.*

(1) *If* $\Delta \Vdash^{\mathsf{s}} \sigma : \Gamma$ *then* $\Delta \Vdash^{\mathsf{s}} \sigma = \sigma : \Gamma$.
(2) *If* $\Delta \Vdash^{\mathsf{s}} \sigma \mathrel{:=:} \sigma' : \Gamma$ *then* $\Delta \Vdash^{\mathsf{s}} \sigma' = \sigma : \Gamma$.
(3) *If* $\Delta \Vdash^{\mathsf{s}} \sigma \mathrel{:=:} \sigma' : \Gamma$ *and* $\Delta \Vdash^{\mathsf{s}} \sigma' \mathrel{:=:} \sigma'' : \Gamma$ *then* $\Delta \Vdash^{\mathsf{s}} \sigma = \sigma'' : \Gamma$.

Proof. By induction on $\Vdash^{\mathsf{v}} \Gamma$, using reflexivity (3.1), conversion (3.7), symmetry (3.8) and transitivity (3.9) of reducible equality. □

Corollary 3.19 (Reducibility of validity). *All valid types, terms and equalities are reducible.*

Proof. By applying the reducible identity substitution (3.17) to the valid object and using reducible irrelevance (3.5). □

While we can substitute valid objects with arbitrary reducible substitutions and get reducible objects, we do not directly get valid objects. This is because reducibility is not closed under substitution. Thus we will restrict ourselves to proving that substitution of a single term preserves validity.

Lemma 3.20 (Single substitution). *Given* $\Gamma \Vdash^{\mathsf{v}}_{\ell} F$*:*

(1) *If* $\Gamma \Vdash^{\mathsf{v}}_{\ell} t : F$ *and either* $\Gamma, F \Vdash^{\mathsf{v}}_{\ell} G$ *or* $\Gamma \Vdash^{\mathsf{v}}_{\ell} \Pi\, F\, G$ *then* $\Gamma \Vdash^{\mathsf{v}}_{\ell} G[t]$.
(2) *If* $\Gamma \Vdash^{\mathsf{v}}_{\ell} F \mathrel{:=:} F'$ *and* $\Gamma \Vdash^{\mathsf{v}}_{\ell} t \mathrel{:=:} t' : F$ *with* $\Gamma \Vdash^{\mathsf{v}}_{\ell} t' : F'$ *and either* $\Gamma, F \Vdash^{\mathsf{v}}_{\ell} G \mathrel{:=:} G'$ *with* $\Gamma, F' \Vdash^{\mathsf{v}}_{\ell} G'$ *or* $\Gamma \Vdash^{\mathsf{v}}_{\ell} \Pi\, F\, G \mathrel{:=:} \Pi\, F'\, G'$ *then* $\Gamma \Vdash^{\mathsf{v}}_{\ell} G[t] = G'[t']$.
(3) *If* $\Gamma \Vdash^{\mathsf{v}}_{\ell} t : F$ *and* $\Gamma, F \Vdash^{\mathsf{v}}_{\ell} G$ *and* $\Gamma, F \Vdash^{\mathsf{v}}_{\ell} f : G$ *then* $\Gamma \Vdash^{\mathsf{v}}_{\ell} f[t] : G[t]$.
(4) *If* $\Gamma, F \Vdash^{\mathsf{v}}_{\ell} u : F[\uparrow\mathsf{id}]$ *then* $\Gamma, F \Vdash^{\mathsf{v}}_{\ell} G[\uparrow\mathsf{id}, u]$.
(5) *If* $\Gamma, F \Vdash^{\mathsf{v}}_{\ell} u \mathrel{:=:} u' : F[\uparrow\mathsf{id}]$ *then* $\Gamma, F \Vdash^{\mathsf{v}}_{\ell} G[\uparrow\mathsf{id}, u] = G'[\uparrow\mathsf{id}, u']$.

Proof. We elaborate case (1): If $\Gamma, F \Vdash^{\mathsf{v}}_{\ell} G$, we deconstruct that validity judgement and reconstruct it such that when applying a substitution σ, we instead apply $(\sigma, t[\sigma])$. Otherwise, if $\Gamma \Vdash^{\mathsf{v}}_{\ell} \Pi\, F\, G$, we deconstruct the reducible object of the validity judgement and replace a with $t[\sigma]$ in the following premise of reducible Π:

$$\forall \rho : \Delta \leq \Gamma.\ \ \Delta \Vdash_{\ell} a : F[\rho] \implies \Delta \Vdash_{\ell} G[\rho, a]$$

All in all, to prove this lemma we need escape (3.2), irrelevance (3.5), conversion (3.7) and transitivity (3.9) of reducible equality, and reflexivity of substitution (3.18). □

With the above lemmas, we can now prove the fundamental theorem:

Theorem 3.21 (Fundamental theorem).

(1) *If* $\vdash \Gamma$ *then* $\Vdash^{\mathsf{v}} \Gamma$.
(2) *If* $\Gamma \vdash A$ *then* $\Gamma \Vdash^{\mathsf{v}}_{\ell} A$.
(3) *If* $\Gamma \vdash A = B$ *then* $\Gamma \Vdash^{\mathsf{v}}_{\ell} A :=: B$.
(4) *If* $\Gamma \vdash t : A$ *then* $\Gamma \Vdash^{\mathsf{v}}_{\ell} A$ *and* $\Gamma \Vdash^{\mathsf{v}}_{\ell} t : A$.
(5) *If* $\Gamma \vdash t = u : A$ *then* $\Gamma \Vdash^{\mathsf{v}}_{\ell} A$ *and* $\Gamma \Vdash^{\mathsf{v}}_{\ell} t :=: u : A$.

Proof. By induction on the well-formed judgements. The most salient cases are, Π and Π-congruence, variables, λ and η-equality, application and application-congruence, and finally natrec and natrec-congruence.

This is accomplished with escape (3.2), reducible equality being a equivalence relation (3.1 & 3.8 & 3.9), conversion (3.7), irrelevance of reducibility (3.5) and validity (3.13), reducibility of neutrals (3.10), weak head expansion (3.11), reducibility of application (3.12), substitution weakening (3.15) and lifting (3.16), reflexivity of substitutions (3.18) and single substitution (3.20). □

With the fundamental theorem, escape lemma (3.2) and reducibility of validity (3.19) we have effectively proven $\Gamma \vdash J$ iff $\Gamma \Vdash^{\mathsf{v}} J$ for the judgements J for types, terms and their respective equality. From this we can also prove $\Gamma \vdash J$ iff $\Gamma \Vdash J$:

Corollary 3.22 (Reducibility of well-formedness). *Any well-formed object is reducible.*

Proof. Well-formedness implies validity by the fundamental theorem (3.21) instantiated to judgemental equality (1). Further, validity implies reducibility (3.19). □

We will also prove a fundamental theorem for substitutions:

Theorem 3.23 (Fundamental theorem for substitutions). *If* $\Delta \vdash \sigma : \Gamma$ *for well-formed contexts* Γ *and* Δ *then* $\Delta \Vdash^{\mathsf{s}} \sigma : \Gamma$.

Proof. By induction on Γ and with the judgmental instance (1): by fundamental theorem (3.21) and reducible irrelevance (3.5), valid irrelevance (3.13) and identity substitution (3.17). □

3.6 Consequences of the Fundamental Theorem

We will here declare and prove some theorems we can now prove using the fundamental theorem with generic equality instance 1.

Theorem 3.24 (Canonicity). *Let* $\mathsf{suc}^0\, t \triangleq t$ *and* $\mathsf{suc}^{n+1}\, t \triangleq \mathsf{suc}\,(\mathsf{suc}^n\, t)$. *Given* $\epsilon \vdash t : \mathbb{N}$ *then there exists* n *such that* $\epsilon \vdash t = \mathsf{suc}^n\, \mathsf{zero} : \mathbb{N}$.

Proof. Since well-formed objects are reducible (3.22) we have $\epsilon \Vdash \mathbb{N}\, t$, on which we induct. □

Theorem 3.25 (Π-injectivity). *If* $\Gamma \vdash \Pi\, F\, G = \Pi\, H\, E$ *then* $\Gamma \vdash F = H$ *and* $\Gamma, F \vdash G = E$.

Proof. By well-formed objects being reducible (3.22) we have $\Gamma \Vdash_{\ell} \Pi\, F\, G = \Pi\, H\, E$, from which injectivity can be retrieved by escape (3.2), irrelevance (3.5) and reducibility of neutrals (3.10). □

Theorem 3.26 (Syntactic validity).

(1) *If* $\Gamma \vdash A = B$ *then* $\Gamma \vdash A$ *and* $\Gamma \vdash B$.
(2) *If* $\Gamma \vdash t : A$ *then* $\Gamma \vdash A$.
(3) *If* $\Gamma \vdash t = u : A$ *then* $\Gamma \vdash t : A$ *and* $\Gamma \vdash u : A$.
(4) *If* $\Gamma \vdash A \longrightarrow^* B$ *then* $\Gamma \vdash A$ *and* $\Gamma \vdash B$.
(5) *If* $\Gamma \vdash t \longrightarrow^* u : A$ *then* $\Gamma \vdash t : A$ *and* $\Gamma \vdash u : A$.

Proof. By the fundamental theorem (3.21) and by reductions being subsumed by equality (2.2) and escape (3.2). □

Corollary 3.27 (Syntactic Pi). *If* $\Gamma \vdash \Pi\, F\, G$ *then* $\Gamma \vdash F$ *and* $\Gamma, F \vdash G$.

Proof. By lemma Π-injectivity (3.25) and syntactic validity (3.26). □

Theorem 3.28 (Weak head normalization).

(1) *If* $\Gamma \vdash A$ *then there exists some* $\overline{A}$ *such that* $\Gamma \vdash A \longrightarrow^* \overline{A}$.
(2) *If* $\Gamma \vdash t : A$ *then there exists some* $\overline{t}$ *such that* $\Gamma \vdash t \longrightarrow^* \overline{t} : A$.

Proof. By well-formed objects being reducible (3.22) we get a reducible object for which we use induction and reduction being subsumed by equality (2.2). □

Theorem 3.29 (Syntactic equality).

(1) *If* $\Gamma \vdash \mathsf{U} = A$ *then* $\mathsf{U} \equiv A$.
(2) *If* $\Gamma \vdash \mathbb{N} = \overline{A}$ *then* $\mathbb{N} \equiv \overline{A}$.
(3) *If* $\Gamma \vdash N = \overline{A}$ *then there exists* M *such that* $M \equiv \overline{A}$.
(4) *If* $\Gamma \vdash \Pi\, F\, G = \overline{A}$ *then there exist* H *and* E *such that* $\Pi\, H\, E \equiv \overline{A}$.

Proof. By well-formed objects being reducible (3.22) and induction on the reducible equality with reducible irrelevance (3.5). □

Theorem 3.30 (Syntactic inequality). *For* $A, B \in \{\mathsf{U}, \mathbb{N}, N, \Pi\, F\, G\}$, *if* $A \not\equiv B$ *then* $\Gamma \vdash A \neq B$.

Proof. By well-formed objects being reducible (3.22) and showing that there cannot exist an instance of the shape view of types A and B using escape (3.2), shape view construction (3.3) and reducible irrelevance (3.5). □

Theorem 3.31 (Substitution).

(1) *If* $\Delta \vdash \sigma : \Gamma$ *and* $\Gamma \vdash A$ *then* $\Delta \vdash A[\sigma]$.
(2) *If* $\Delta \vdash \sigma = \sigma' : \Gamma$ *and* $\Gamma \vdash A = B$ *then* $\Delta \vdash A[\sigma] = B[\sigma']$.
(3) *If* $\Delta \vdash \sigma : \Gamma$ *and* $\Gamma \vdash t : A$ *then* $\Delta \vdash t[\sigma] : A[\sigma]$.
(4) *If* $\Delta \vdash \sigma = \sigma' : \Gamma$ *and* $\Gamma \vdash t = u : A$ *then* $\Delta \vdash t[\sigma] = u[\sigma'] : A[\sigma]$.

Proof. By the fundamental theorem (3.21) and fundamental theorem for substitutions (3.23), escape (3.2) and valid irrelevance (3.13). □

Theorem 3.32 (Substitution composition). *Given* $\Delta' \vdash \sigma : \Delta$ *and* $\Delta \vdash \sigma' : \Gamma$ *then* $\Delta' \vdash \sigma \circ \sigma' : \Gamma$

Proof. By induction on the well-formedness of Γ and σ, using the substitution theorem(3.31). □

Theorem 3.33 (Neutral type equality). *If* $\Gamma \vdash n : A$ *and* $\Gamma \vdash n : B$ *then* $\Gamma \vdash A = B$.

Proof. By induction on the syntactic structure of n and by Π-injectivity (3.25), syntactic validity (3.26) and substitution (3.31). □

Theorem 3.34 (Universe membership).

(1) *If* $\Gamma \vdash A$ *and* A *has no occurrence of* U *then* $\Gamma \vdash A : \mathsf{U}$.
(2) *If* $\Gamma \vdash A = B$ *and* A *and* B *has no occurrence of* U *then* $\Gamma \vdash A = B : \mathsf{U}$.

Proof. By induction on the judgements and by syntactic validity (3.26). □

Theorem 3.35 (Consistency). $\Gamma \vdash \mathsf{zero} = \mathsf{suc\ zero} : \mathbb{N}$ *is impossible.*

Proof. By well-formed objects being reducible (3.22), using induction on the reducible instance of the equality and using the fact that Whnf do not reduce (2.4) and reducible irrelevance (3.5). □

4 DECIDABILITY

To prove that our language has decidable conversion, we will now introduce an algorithm for conversion of types and terms. The algorithm is defined as a relation which we then show decidable and equivalent to the conversion judgements. In particular completeness is proven by constructing a generic equality instance for the algorithmic equality, so that we can apply our fundamental theorem.

4.1 Conversion Algorithm

Our conversion algorithm is defined inductively as seen in Fig. 6, with six different relations defined simultaneously.

We first have the relations $\Gamma \vdash n \longleftrightarrow m : A$ and $\Gamma \vdash n \overset{\wedge}{\longleftrightarrow} m : A$, which is the algorithm that determines equality between neutral terms, where the former relation enforces the type A to be in Whnf. Secondly we have the relations $\Gamma \vdash A \Longleftrightarrow B$ and $\Gamma \vdash A \overset{\wedge}{\Longleftrightarrow} B$, which determines equality between types, where the former of the two relations enforces the types to be in Whnf. Lastly, we have the relations $\Gamma \vdash t \Longleftrightarrow u : A$ and $\Gamma \vdash t \overset{\wedge}{\Longleftrightarrow} u : A$, which determines equality between terms. Similarly to above the former enforces Whnf of the type and the two terms.

Of note is the third rule of these two relations. We do not check that the type N is equal to the type M inferred by the neutral algorithm. Since we can derive this equality by Lemma 3.33, we can be economic and drop this premise, thus simplifying the relation.

4.2 Properties of the Conversion Algorithm

Our next goal is to prove decidability and construct a generic equality instance for the logical relation. It turns out that some of the properties necessary for instance validity is also necessary for proving decidability, thus we will begin proving those properties.

To prove these properties, we also need a notion of context equality, which we will denote as $\boxed{\vdash \Gamma = \Delta}$. We define it inductively:

$$\frac{}{\vdash \epsilon = \epsilon} \qquad\qquad \frac{\vdash \Gamma = \Delta \qquad \Gamma \vdash A = B}{\vdash \Gamma, A = \Delta, B}$$

Lemma 4.1 (Context conversion for typing judgements). *Given* $\vdash \Gamma = \Delta$*:*

(1) $\Delta \vdash \mathrm{id} : \Gamma$ *and contexts* Γ *and* Δ *are well-formed.*
(2) *If* $\Gamma \vdash J$ *then* $\Delta \vdash J$ *for the syntactic judgements* J *of types, type membership and their respective equality.*

Proof. By induction on the context equality and by substitution (3.31). □

Lemma 4.2 (Context conversion for reduction and algorithmic equality). *If* $\vdash \Gamma = \Delta$ *and* $\Gamma \vdash J$ *then* $\Delta \vdash J$ *for the syntactic judgements* J *of reduction and algorithmic equality.*

Proof. By induction on the judgements and context conversion for typing judgements (4.1). □

Lemma 4.3 (Soundness).

(1) *If either* $\Gamma \vdash A \overset{\wedge}{\Longleftrightarrow} B$ *or* $\Gamma \vdash A \Longleftrightarrow B$ *then* $\Gamma \vdash A = B$.
(2) *If either* $\Gamma \vdash t \overset{\wedge}{\longleftrightarrow} u : A$ *or* $\Gamma \vdash t \longleftrightarrow u : A$ *or* $\Gamma \vdash t \overset{\wedge}{\Longleftrightarrow} u : A$ *or* $\Gamma \vdash t \Longleftrightarrow u : A$ *then* $\Gamma \vdash t = u : A$.

$$\boxed{\Gamma \vdash n \longleftrightarrow m : A}$$

$$\frac{\Gamma \vdash \mathrm{i}_x : A \qquad x \equiv y}{\Gamma \vdash \mathrm{i}_x \mathrel{\hat{\longleftrightarrow}} \mathrm{i}_y : A} \qquad \frac{\Gamma \vdash n \longleftrightarrow m : \Pi\, F\, G \qquad \Gamma \vdash t \mathrel{\hat{\Longleftrightarrow}} u : F}{\Gamma \vdash n\, t \mathrel{\hat{\longleftrightarrow}} m\, u : G[t]}$$

$$\frac{\begin{array}{c}\Gamma, \mathbb{N} \vdash F \mathrel{\hat{\Longleftrightarrow}} G \qquad \Gamma \vdash z \mathrel{\hat{\Longleftrightarrow}} z' : F[\mathsf{zero}] \\ \Gamma \vdash s \mathrel{\hat{\Longleftrightarrow}} s' : \Pi\, \mathbb{N}\, (F \to F[\uparrow\mathrm{id}, \mathsf{suc}\, i_0]) \qquad \Gamma \vdash n \longleftrightarrow m : \mathbb{N}\end{array}}{\Gamma \vdash \mathsf{natrec}\, F\, z\, s\, n \mathrel{\hat{\longleftrightarrow}} \mathsf{natrec}\, G\, z'\, s'\, m : F[n]}$$

$$\frac{\Gamma \vdash A \longrightarrow^* \overline{A} \qquad \Gamma \vdash n \mathrel{\hat{\longleftrightarrow}} m : A}{\Gamma \vdash n \longleftrightarrow m : \overline{A}}$$

$$\boxed{\Gamma \vdash A \Longleftrightarrow B}$$

$$\frac{\Gamma \vdash A \longrightarrow^* \overline{A} \qquad \Gamma \vdash B \longrightarrow^* \overline{B} \qquad \Gamma \vdash \overline{A} \Longleftrightarrow \overline{B}}{\Gamma \vdash A \mathrel{\hat{\Longleftrightarrow}} B} \qquad \frac{\Gamma \vdash N \longleftrightarrow M : \mathsf{U}}{\Gamma \vdash N \Longleftrightarrow M}$$

$$\frac{\vdash \Gamma}{\Gamma \vdash \mathsf{U} \Longleftrightarrow \mathsf{U}} \qquad \frac{\vdash \Gamma}{\Gamma \vdash \mathbb{N} \Longleftrightarrow \mathbb{N}} \qquad \frac{\Gamma \vdash F \qquad \Gamma \vdash F \mathrel{\hat{\Longleftrightarrow}} H \qquad \Gamma, F \vdash G \mathrel{\hat{\Longleftrightarrow}} E}{\Gamma \vdash \Pi\, F\, G \Longleftrightarrow \Pi\, H\, E}$$

$$\boxed{\Gamma \vdash t \Longleftrightarrow u : A}$$

$$\frac{\Gamma \vdash A \longrightarrow^* \overline{A} \qquad \Gamma \vdash t \longrightarrow^* \overline{t} : \overline{A} \qquad \Gamma \vdash u \longrightarrow^* \overline{u} : \overline{A} \qquad \Gamma \vdash \overline{t} \Longleftrightarrow \overline{u} : \overline{A}}{\Gamma \vdash t \mathrel{\hat{\Longleftrightarrow}} u : A}$$

$$\frac{\Gamma \vdash n \longleftrightarrow m : \mathbb{N}}{\Gamma \vdash n \Longleftrightarrow m : \mathbb{N}} \qquad \frac{\Gamma \vdash n : N \qquad \Gamma \vdash m : N \qquad \Gamma \vdash n \longleftrightarrow m : M}{\Gamma \vdash n \Longleftrightarrow m : N}$$

$$\frac{\Gamma \vdash \overline{A} : \mathsf{U} \quad \Gamma \vdash \overline{B} : \mathsf{U} \quad \Gamma \vdash \overline{A} \Longleftrightarrow \overline{B}}{\Gamma \vdash \overline{A} \Longleftrightarrow \overline{B} : \mathsf{U}} \qquad \frac{\vdash \Gamma}{\Gamma \vdash \mathsf{zero} \Longleftrightarrow \mathsf{zero} : \mathbb{N}} \qquad \frac{\Gamma \vdash t \mathrel{\hat{\Longleftrightarrow}} u : \mathbb{N}}{\Gamma \vdash \mathsf{suc}\, t \Longleftrightarrow \mathsf{suc}\, u : \mathbb{N}}$$

$$\frac{\Gamma \vdash F \qquad \Gamma \vdash \overline{f} : F \qquad \Gamma \vdash \overline{g} : F \qquad \Gamma \vdash \overline{f}[\uparrow\mathrm{id}]\, \mathrm{i}_0 \mathrel{\hat{\Longleftrightarrow}} \overline{g}[\uparrow\mathrm{id}]\, \mathrm{i}_0 : G}{\Gamma \vdash \overline{f} \Longleftrightarrow \overline{g} : \Pi\, F\, G}$$

Fig. 6. Algorithm for conversion of neutrals, types and terms.

Proof. By induction on the judgements and by syntactic validity (3.26), universe membership (3.34), neutral type equality (3.33), and reduction being subsumed by equality (2.2). □

Lemma 4.4 (Conversion).
Given $\vdash \Gamma = \Delta$ *and* $\Gamma \vdash A = B$ *and* $\Gamma \vdash t \mathrel{\hat{\Longleftrightarrow}} u : A$ *then* $\Delta \vdash t \mathrel{\hat{\Longleftrightarrow}} u : B$.

Proof. By induction on the judgements and by Π-injectivity (3.25), syntactic validity (3.26), weak head normalization (3.28), reduction being subsumed by equality (2.2) and context conversion (4.1 and 4.2). □

We can now prove decidability for the algorithm.

Theorem 4.5 (Decidability of algorithmic equality). *Given* $\vdash \Gamma = \Delta$*:*

(1) *If* $\Gamma \vdash t \mathrel{\hat{\longleftrightarrow}} t : A$ *and* $\Delta \vdash u \mathrel{\hat{\longleftrightarrow}} u : B$ *then it is decidable that there exists* C *such that* $\Gamma \vdash t \mathrel{\hat{\longleftrightarrow}} u : C$.
(2) *If* $\Gamma \vdash A \mathrel{\hat{\Longleftrightarrow}} A$ *and* $\Delta \vdash B \mathrel{\hat{\Longleftrightarrow}} B$ *then* $\Gamma \vdash A \mathrel{\hat{\Longleftrightarrow}} B$ *is decidable.*
(3) *If* $\Gamma \vdash t \mathrel{\hat{\Longleftrightarrow}} t : A$ *and* $\Delta \vdash u \mathrel{\hat{\Longleftrightarrow}} u : A$ *then* $\Gamma \vdash t \mathrel{\hat{\Longleftrightarrow}} u : A$ *is decidable.*

Proof. By induction on the judgements and by Π-injectivity (3.25), syntactic validity (3.26), weak head normalization (3.28), strong equality (3.29), syntactic inequality (3.30), substitution (3.31), neutral type equality (3.33), determinism of reduction (2.5), soundness (4.3), context conversion for typing judgements (4.1) and conversion of algorithmic equality (4.4). □

Lemma 4.6 (Symmetry). *Given* $\vdash \Gamma = \Delta$*:*

(1) *If* $\Gamma \vdash t \mathrel{\hat{\longleftrightarrow}} u : A$ *then there exists* B *such that* $\Gamma \vdash A = B$ *and* $\Delta \vdash u \mathrel{\hat{\longleftrightarrow}} t : B$.
(2) *If* $\Gamma \vdash A \mathrel{\hat{\Longleftrightarrow}} B$ *then* $\Delta \vdash B \mathrel{\hat{\Longleftrightarrow}} A$.
(3) *If* $\Gamma \vdash t \mathrel{\hat{\Longleftrightarrow}} u : A$ *then* $\Delta \vdash u \mathrel{\hat{\Longleftrightarrow}} t : A$.

Proof. By induction on the judgements and by Π-injectivity (3.25), syntactic validity (3.26), weak head normalization (3.28), strong equality (3.29), substitution (3.31), context conversion (4.1 and 4.2), soundness (4.3) and conversion of algorithmic equality (4.4). □

Lemma 4.7 (Transitivity). *Given* $\vdash \Gamma = \Delta$*:*

(1) *If* $\Gamma \vdash t \mathrel{\hat{\longleftrightarrow}} u : A$ *and* $\Delta \vdash u \mathrel{\hat{\longleftrightarrow}} v : A$ *then there exists* B *such that* $\Gamma \vdash A = B$ *and* $\Gamma \vdash t \mathrel{\hat{\longleftrightarrow}} v : B$.
(2) *If* $\Gamma \vdash A \mathrel{\hat{\Longleftrightarrow}} B$ *and* $\Delta \vdash B \mathrel{\hat{\Longleftrightarrow}} C$ *then* $\Gamma \vdash A \mathrel{\hat{\Longleftrightarrow}} C$.
(3) *If* $\Gamma \vdash t \mathrel{\hat{\Longleftrightarrow}} u : A$ *and* $\Delta \vdash u \mathrel{\hat{\Longleftrightarrow}} v : A$ *then* $\Delta \vdash t \mathrel{\hat{\Longleftrightarrow}} v : A$.

Proof. By induction on the judgements and by Π-injectivity (3.25), syntactic inequality (3.30), substitution (3.31), neutral type equality (3.33), reduction being subsumed by equality 2.2, determinism of reduction 2.5, context conversion (4.1 and 4.2) and soundness (4.3). □

Lemma 4.8 (Weakening). *For all algorithmic equality judgements* J*, given* $\rho : \Delta \leq \Gamma$ *and* $\Gamma \vdash J$ *then* $\Delta \vdash J[\rho]$.

Proof. By induction on the judgements and well-formed weakening (2.7). □

Lemma 4.9 (Whnf Lifting).

(1) *If* $\Gamma \vdash A \Longleftrightarrow B$ *then* $\Gamma \vdash A \mathrel{\hat{\Longleftrightarrow}} B$.
(2) *If* $\Gamma \vdash t \Longleftrightarrow u : A$ *then* $\Gamma \vdash t \mathrel{\hat{\Longleftrightarrow}} u : A$.

Proof. By syntactic validity (3.26) and soundness (4.3). □

Lemma 4.10 (Neutral lifting). *If* $\Gamma \vdash t \mathrel{\hat{\longleftrightarrow}} u : A$ *then* $\Gamma \vdash t \mathrel{\hat{\Longleftrightarrow}} u : A$.

Proof. By well-formed objects being reducible (3.22), induction on the reducible type and then by syntactic validity (3.26), weak head normalization (3.28), reduction being subsumed by equality (2.2), determinism of reduction (2.5), reducible neutrals (3.10), and soundness (4.3). □

We can now construct our instance.

Instance 2 (Algorithmic equality instance). *The following instantiation of generic equality satisfies all the required properties for the logical relation:*

$$\Gamma \vdash A \cong B \textit{ instantiated to } \Gamma \vdash A \overset{\wedge}{\Longleftrightarrow} B$$

$$\Gamma \vdash t \cong u : A \textit{ instantiated to } \Gamma \vdash t \overset{\wedge}{\Longleftrightarrow} u : A$$

$$\Gamma \vdash t \sim u : A \textit{ instantiated to the pair } \Gamma \vdash A = B \textit{ and } \Gamma \vdash t \overset{\wedge}{\longleftrightarrow} u : B$$

Proof. We get the necessary properties from the definition of the algorithm and soundness (4.3), conversion (4.4), symmetry (4.6), transitivity (4.7), weakening (4.8), lifting (4.9) and neutral lifting (4.10) of algorithmic equality. □

With the instance of generic equality, we can now use the logical relation and the fundamental theorem to prove the completeness of algorithmic equality and finally decidability of conversion:

Theorem 4.11 (Completeness of algorithmic equality).

(1) *If* $\Gamma \vdash A = B$ *then* $\Gamma \vdash A \overset{\wedge}{\Longleftrightarrow} B$.
(2) *If* $\Gamma \vdash t = u : A$ *then* $\Gamma \vdash t \overset{\wedge}{\Longleftrightarrow} u : A$.

Proof. By the fundamental theorem (3.21) with algorithmic equality (2) and escape (3.2). □

Theorem 4.12 (Decidability of conversion).

(1) *If* $\Gamma \vdash A$ *and* $\Gamma \vdash B$ *then* $\Gamma \vdash A = B$ *is decidable.*
(2) *If* $\Gamma \vdash t : A$ *and* $\Gamma \vdash u : A$ *then* $\Gamma \vdash t = u : A$ *is decidable.*

Proof. By completeness (4.11), soundness (4.3) and decidability of algorithmic equality (4.5). □

5 CONCLUSION

We have fully formalized a substantial part of the meta-theory of a fragment of Martin-Löf type theory, using a limited set of features of Agda. Our formalization should be implementable in other type theories that support dependent types, universes, and induction-recursion.

However, the gap between expressive power of the formalized type theory (the object type theory) and the host type theory (the meta type theory) is fairly large: we require induction-recursion on the meta level to formalize just one inductive type and one universe on the object level. Yet the grand goal is *boot-strapping type theory*, meaning the implementation of type theory in a small extension of itself. (An extension in proof-theoretical strength is needed due to Gödel's incompleteness theorem.) A small extension would be, e.g., one more universe or one additional axiom. To advance towards the grand goal, we have to bring object and meta type theory closer together. Obviously, there are two directions: we can make the meta theory weaker, or the object theory stronger.

The main strength of the meta theory comes from induction-recursion [Dybjer 2000; Dybjer and Setzer 2001, 2003], which is a very powerful principle, proof-theoretically. Using iterated inductive-recursive definitions in the meta-theory with just one universe, we can model a countable hierarchy of universes of an object theory that lacks induction-recursion. Allowing more universes in the meta-theory, it is likely possible to encode our inductive-recursive logical relation using iterated inductive definitions only, placing the inductive definition of object-level universe n in meta-level universe n. Similar techniques have been used for the semantic modeling and auto-validation of impredicative type theories [Barras 2010; Werner 1994], using inaccessible cardinals which are analogous to universes. Eliminating induction-recursion would require reworking the central definition of the logical relation in our development, but could lead to a formalized boot-strapping of intensional predicative Martin-Löf Type Theory in itself (plus an extra universe and/or axiom).

Concerning the other direction, making the object theory stronger, this would amount to extending our logical relation to model induction-recursion. This is interesting future work, as it will enhance our understanding of induction-recursion from a syntactic perspective. Recent advances in understanding induction-recursion might aid this research [Ghani et al. 2015].

It remains interesting to investigate the relationship between the constructions in our proof and standard notions of models of type theory like Categories with Families (CwFs) by Dybjer [1995]. By contrast, Altenkirch and Kaposi [2016; 2017] directly define the syntax of type theory as an internal formulation of the initial CwF, i.e., quotienting terms by judgmental equality. In our case, however, we want to discuss reduction, which is only meaningful on terms that are not yet quotiented. A way to bridge this gap could be to formulate a weaker notion of CwFs, where some equations only have to hold up to some equivalence, and use it to restructure our proof, with the intent of providing a more abstract and generalizable presentation.

As for our contributions to proof methods, we have used proof-relevant induction-recursion to define a parameterized logical relation, which allows us to derive important properties like consistency, normalization, and Π-injectivity. We have managed to show that the *a priori* proof-relevant definitions are proof-irrelevant *a posteriori*, in the sense that the structure of the inductive derivations we recurse on does not matter in the end, only the judgement it derives (Lemma 3.5). Informally, this aspect is often glossed over; proof irrelevance is often assumed without officially being present in the meta-theory.

We have also parameterized this logical relation such that we only need to prove the fundamental theorem once, even though we have two slightly different instances. For the formalization, this means that we have saved a lot of work, as the parameterization does not add much code considering the size of the logical relation and the proof of the fundamental theorem, which add up to roughly 5000 lines of Agda code.

Our formalization comprises roughly 10.000 lines of Agda code (totaling 500.000 characters). It took 10 man-months of development work and type-checks on a contemporary laptop (16 GB memory, CPU speed 3.3GHz) in around 90 seconds. From this code we could extract the core of a type checker for fully-elaborated terms of a small fragment of the Agda language. It would neither be efficient nor could it do any type reconstruction, however, it could serve as a conceptual double checker of elaborated terms. In contrast, Agda itself consists roughly of 100.000 lines of Haskell code and features printer, parser, elaborator, termination and positivity analysis, pattern matcher and coverage checker, and many other components that are needed for a practically usable implementation of type theory. Verifying all these components with the same rigor as applied to our formalization seems clearly out of reach for an academic institution. However, a verified double checker for the full type theory behind Agda remains as a pursuable, grand challenge.

ACKNOWLEDGMENTS

We are grateful to the anonymous reviewers, whose comments greatly helped to improve the presentation of our research in this article. The idea of basing the Kripke logical relation on a typed version of weak head reduction came from our colleagues Thierry Coquand and Bassel Mannaa. We also thank Paolo Capriotti for stimulating conversations on the relationship between notions of models and normalization. The first author acknowledges support by the Swedish Research Council (Vetenskapsrådet) under Grant No. 621-2014-4864 *Termination Certificates for Dependently-Typed Programs and Proofs via Refinement Types*. Our research groups are part of the EU Cost Action CA15123 *The European research network on types for programming and verification (EUTypes)*.

REFERENCES

Andreas Abel, Thierry Coquand, and Peter Dybjer. 2007. Normalization by Evaluation for Martin-Löf Type Theory with Typed Equality Judgements. In *22nd IEEE Symposium on Logic in Computer Science (LICS 2007), 10-12 July 2007, Wroclaw, Poland, Proceedings*. IEEE Computer Society Press, 3–12. https://doi.org/10.1109/LICS.2007.33

Andreas Abel, Thierry Coquand, and Bassel Mannaa. 2016. On Decidability of Conversion in Type Theory. In *22nd International Conference on Types for Proofs and Programs, TYPES 2016, Novi Sad, Serbia, May 23-26, 2016, Book of Abstracts*, Silvia Ghilezan and Jelena Ivetic (Eds.). EasyChair.

Andreas Abel and Brigitte Pientka. 2016. Well-founded recursion with copatterns and sized types. *Journal of Functional Programming* 26 (2016), 61. https://doi.org/10.1017/S0956796816000022 ICFP 2013 special issue.

Andreas Abel and Gabriel Scherer. 2012. On Irrelevance and Algorithmic Equality in Predicative Type Theory. *Logical Methods in Computer Science* 8, 1:29 (2012), 1–36. https://doi.org/10.2168/LMCS-8(1:29)2012 TYPES'10 special issue.

AgdaTeam. 2017. The Agda Wiki. http://wiki.portal.chalmers.se/agda/pmwiki.php

Thorsten Altenkirch and Ambrus Kaposi. 2016. Type theory in type theory using quotient inductive types. In *Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2016, St. Petersburg, FL, USA, January 20 - 22, 2016*, Rastislav Bodík and Rupak Majumdar (Eds.). ACM Press, 18–29. https://doi.org/10.1145/2837614.2837638

Thorsten Altenkirch and Ambrus Kaposi. 2017. Normalisation by Evaluation for Type Theory, in Type Theory. *Logical Methods in Computer Science* 13(4:1) (2017), 1–26. https://doi.org/10.23638/LMCS-13(4:1)2017

Bruno Barras. 2010. Sets in Coq, Coq in Sets. *Journal of Formalized Reasoning* 3, 1 (2010), 29–48. https://doi.org/10.6092/issn.1972-5787/1695

Thierry Coquand. 1991. An Algorithm for Testing Conversion in Type Theory. In *Logical Frameworks*, Gérard Huet and Gordon Plotkin (Eds.). Cambridge University Press, 255–279. http://dl.acm.org/citation.cfm?id=120477.120486

Thierry Coquand and Gérard P. Huet. 1988. The Calculus of Constructions. *Information and Computation* 76, 2/3 (1988), 95–120. https://doi.org/10.1016/0890-5401(88)90005-3

N. G. de Bruijn. 1972. Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation, with application to the Church-Rosser theorem. *Indagationes Mathematicae* 75, 5 (1972), 381–392. https://doi.org/10.1016/1385-7258(72)90034-0

Peter Dybjer. 1995. Internal Type Theory. In *Types for Proofs and Programs, International Workshop TYPES'95, Torino, Italy, June 5-8, 1995, Selected Papers (Lecture Notes in Computer Science)*, Stefano Berardi and Mario Coppo (Eds.), Vol. 1158. Springer, 120–134. https://doi.org/10.1007/3-540-61780-9_66

Peter Dybjer. 2000. A General Formulation of Simultaneous Inductive-Recursive Definitions in Type Theory. *The Journal of Symbolic Logic* 65, 2 (2000), 525–549. https://doi.org/10.2307/2586554

Peter Dybjer and Anton Setzer. 2001. Indexed Induction-Recursion. In *Proof Theory in Computer Science, International Seminar, PTCS 2001, Dagstuhl Castle, Germany, October 7-12, 2001, Proceedings (Lecture Notes in Computer Science)*, Reinhard Kahle, Peter Schroeder-Heister, and Robert F. Stärk (Eds.), Vol. 2183. Springer, 93–113. https://doi.org/10.1007/3-540-45504-3_7

Peter Dybjer and Anton Setzer. 2003. Induction-recursion and initial algebras. *Annals of Pure and Applied Logic* 124, 1-3 (2003), 1–47. https://doi.org/10.1016/S0168-0072(02)00096-9

Harvey Friedman. 1975. Equality between functionals. In *Logic Colloquium (Lecture Notes in Mathematics)*, R. Parikh (Ed.), Vol. 453. Springer, 22–37. https://doi.org/10.1007/BFb0064870

Herman Geuvers. 1994. A short and flexible proof of Strong Normalization for the Calculus of Constructions. In *Types for Proofs and Programs, International Workshop TYPES'94, Båstad, Sweden, June 6-10, 1994, Selected Papers (Lecture Notes in Computer Science)*, Peter Dybjer, Bengt Nordström, and Jan M. Smith (Eds.), Vol. 996. Springer, 14–38. https://doi.org/10.1007/3-540-60579-7_2

Neil Ghani, Lorenzo Malatesta, and Fredrik Nordvall Forsberg. 2015. Positive Inductive-Recursive Definitions. 11, 1 (2015). https://doi.org/10.2168/LMCS-11(1:13)2015

Jean-Yves Girard. 1972. *Interprétation fonctionnelle et élimination des coupures dans l'arithmétique d'ordre supérieur*. Thèse de Doctorat d'État. Université de Paris VII.

Healfdene Goguen. 1994. *A Typed Operational Semantics for Type Theory*. Ph.D. Dissertation. University of Edinburgh. Available as LFCS Report ECS-LFCS-94-304.

Healfdene Goguen. 2000. A Kripke-Style Model for the Admissibility of Structural Rules. In *Types for Proofs and Programs, International Workshop, TYPES 2000, Durham, UK, December 8-12, 2000, Selected Papers (Lecture Notes in Computer Science)*, Paul Callaghan, Zhaohui Luo, James McKinna, and Robert Pollack (Eds.), Vol. 2277. Springer, 112–124. https://doi.org/10.1007/3-540-45842-5_8

Robert Harper and Frank Pfenning. 2005. On Equivalence and Canonical Forms in the LF Type Theory. *ACM Trans. Comput. Logic* 6, 1 (Jan. 2005), 61–101. https://doi.org/10.1145/1042038.1042041

John Hughes, Lars Pareto, and Amr Sabry. 1996. Proving the Correctness of Reactive Systems Using Sized Types. In *Conference Record of POPL'96: The 23rd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages,* *Papers Presented at the Symposium, St. Petersburg Beach, Florida, USA, January 21-24, 1996*, Hans-Juergen Boehm and Guy L. Steele Jr. (Eds.). ACM Press, 410–423. https://doi.org/10.1145/237721.240882

INRIA. 2017. *The Coq Proof Assistant Reference Manual* (version 8.7 ed.). INRIA. http://coq.inria.fr/

Per Martin-Löf. 1975. An Intuitionistic Theory of Types: Predicative Part. In *Logic Colloquium '73*, H. E. Rose and J. C. Shepherdson (Eds.). North-Holland, 73–118.

Jorge Luis Sacchini. 2013. Type-Based Productivity of Stream Definitions in the Calculus of Constructions. In *28th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2013, New Orleans, LA, USA, June 25-28, 2013*. IEEE Computer Society Press, 233–242. https://doi.org/10.1109/LICS.2013.29

Carsten Schürmann and Jeffrey Sarnat. 2008. Structural Logical Relations. In *Proceedings of the Twenty-Third Annual IEEE Symposium on Logic in Computer Science, LICS 2008, 24-27 June 2008, Pittsburgh, PA, USA*, Frank Pfenning (Ed.). IEEE Computer Society Press, 69–80. https://doi.org/10.1109/LICS.2008.44

Thomas Streicher. 1993. Investigations into Intensional Type Theory. Habilitation thesis, Ludwig-Maximilians-University Munich.

Benjamin Werner. 1992. A Normalization Proof for an Impredicative Type System with Large Eliminations over Integers. In *Proceedings of the 1992 Workshop on Types for Proofs and Programs, Båstad, Sweden, June 1992*, Bengt Nordström, Kent Petersson, and Gordon Plotkin (Eds.). 341–357. http://www.cs.chalmers.se/Cs/Research/Logic/Types/proc92.ps

Benjamin Werner. 1994. *Une Théorie des Constructiones Inductives*. Ph.D. Dissertation. Universite Paris 7.