ON IRRELEVANCE AND ALGORITHMIC EQUALITY IN PREDICATIVE TYPE THEORY

ANDREAS ABEL AND GABRIEL SCHERER

Department of Computer Science, Ludwig-Maximilians-University Munich
e-mail address: andreas.abel@ifi.lmu.de

Gallium team, INRIA Paris-Rocquencourt
e-mail address: gabriel.scherer@gmail.com

Abstract. Dependently typed programs contain an excessive amount of static terms which are necessary to please the type checker but irrelevant for computation. To obtain reasonable performance of not only the compiled program but also the type checker such static terms need to be erased as early as possible, preferably immediately after type checking. To this end, Pfenning's type theory with irrelevant quantification, that models a distinction between static and dynamic code, is extended to universes and large eliminations. Normalization, consistency, and decidability are obtained via a universal Kripke model based on algorithmic equality.

1. Introduction and Related Work

Dependently typed programming languages such as Agda [BDN09], Coq [INR10], and Epigram [MM04] allow the programmer to express in one language programs, their types, rich invariants, and even proofs of these invariants. Besides code executed at run-time, dependently typed programs contain much code needed only to please the type checker, which is a the same time the verifier of the proofs woven into the program.

Program extraction takes type-checked terms and discards parts that are irrelevant for execution. Augustsson's dependently typed functional language Cayenne [Aug99] erases *types* using a universe-based analysis. Coq's extraction procedure has been designed by Paulin-Mohring and Werner [PMW93] and Letouzey [Let02] and discards not only types but also proofs. The erasure rests on Coq's universe-based separation between propositional (Prop) and computational parts (Set/Type). The rigid Prop/Set distinction has the drawback of code duplication: A structure which is sometimes used statically and sometimes dynamically needs to be coded twice, once in Prop and once in Set.

An alternative to the fixed Prop/Set-distinction is to let the usage context decide whether a term is a proof or a program. Besides whole-program analyses such as data

1998 ACM Subject Classification: F.4.1.

Key words and phrases: dependent types, proof irrelevance, typed algorithm equality, logical relation, universal Kripke model.

Revision and extension of FoSSaCS 2011 conference publication.

LOGICAL METHODS
IN COMPUTER SCIENCE
DOI:10.2168/LMCS-???

flow, some type-based analyses have been put forward. One of them is Pfenning's modal type theory of *Intensionality, Extensionality, and Proof Irrelevance* [Pfe01], later pursued by Reed [Ree03], which introduces functions with irrelevant arguments that play the role of proofs.[1] Not only can these arguments be erased during extraction, they can also be disregarded in type conversion tests during type checking. This relieves the user of unnecessary proof burden (proving that two proofs are equal). Furthermore, proofs can not only be discarded during program extraction but directly after type checking, since they will never be looked at again during type checking subsequent definitions.

In principle, we have to distinguish "post mortem" program extraction, let us call it *external erasure*, and proof disposal during type checking, let us call it *internal erasure*. External erasure deals with closed expressions, programs, whereas internal erasure deals with open expressions that can have free variables. Such free variables might be assumed proofs of (possibly false) equations and block type casts, or (possibly false) proofs of well-foundedness and prevent recursive functions from unfolding indefinitely. For type checking to not go wrong or loop, those proofs can only be externally erased, thus, the Prop/Set distinction is not for internal erasure. In Pfenning's type theory, proofs can never block computations even in open expressions (other than computations on proofs), thus, internal erasure is sound.

Miquel's Implicit Calculus of Constructions (ICC) [Miq01a] goes further than Pfenning and considers also *parametric* arguments as irrelevant. These are arguments which are irrelevant for function execution but relevant during type conversion checking. Such arguments may only be erased in function application but not in the associated type instantiation. Barras and Bernardo [BB08] and Mishra-Linger and Sheard [MLS08] have build decidable type systems on top of ICC, but both have not fully integrated inductive types and types defined by recursion (large eliminations). Barras and Bernardo, as Miquel, have inductive types only in the form of their impredicative encodings, Mishra-Linger [ML08] gives introduction and elimination principles for inductive types by example, but does not show normalization or consistency.

While Pfenning's type theory uses typed equality, ICC and its successors interpret typed expressions as untyped λ-terms up to untyped equality. In our experience, the implicit quantification of ICC, which allows irrelevant function arguments to appear unrestricted in the codomain type of the function, is incompatible with type-directed equality. Examples are given in Section 2.3. Therefore, we have chosen to scale Pfenning's notion of proof irrelevance up to inductive types, and integrated it into Agda.

In this article, we start with the "extensionality and proof irrelevance" fragment of Pfenning's type theory in Reed's version [Ree02, Ree03]. We extend it by a hierarchy of predicative universes, yielding *Irrelevant Intensional Type Theory* IITT (Sec. 2). After specifying a type-directed equality algorithm (Sec. 3), we construct a Kripke model for IITT (Sec. 4). It allows us to prove normalization, subject reduction, and consistency, in one go (Sec. 5). A second Kripke logical relation yields correctness of algorithmic equality and decidability of IITT (Sec. 6). Our models are ready for data types, large eliminations, types with extensionality principles, and internal erasure (Sec. 7).

[1] Awodey and Bauer [AB04] give a categorical treatment of proof irrelevance which is very similar to Pfenning and Reed's. However, they work in the setting of Extensional Type Theory with undecidable type checking, we could not directly use their results for this work.

Contribution and Related Work. We consider the design of our meta-theoretic argument as technical novelty, although it heavily relies on previous works to which we owe our inspiration. Allen [All87] describes a logical relation for Martin-Löf type theory with a countable universe hierarchy. The seminal work of Coquand [Coq91] describes an untyped equality check for the Logical Framework and justifies it by a logical relation for dependent types that establishes subject reduction, normalization, completeness of algorithmic equality, and injectivity of function types in one go. However, his approach cannot be easily extended to a *typed* algorithmic equality, due to problems with transitivity.

Goguen introduces *Typed Operational Semantics* [Gog94] to construct a *Kripke* logical relation that simultaneously proves normalization, subject reduction, and confluence for a variant of the *Calculus of Inductive Constructions*. From his results one can derive an equality check based on reduction to normal form. Goguen also shows how to derive syntactic properties, such as closure of typing and equality under substitution, by a Kripke-logical relation [Gog00].

Harper and Pfenning [HP05] popularize a type-directed equality check for the Logical Framework that scales to extensionality for unit types. They prove completeness of algorithmic equality by a Kripke model on *simple types* which are obtained by erasure from the dependent types. Erasure is necessary since algorithmic equality cannot be shown transitive before it is proven sound; yet soundness hinges on subject reduction which rests on function type injectivity which in turn is obtained from completeness of algorithmic equality—a vicious cycle. While erasure breaks the cycle, it also prevents types to be defined by recursion on values (so-called *large eliminations*), a common feature of proof assistants like Agda, Coq, and Epigram.

Normalization by evaluation (NbE) has been successfully used to obtain a type-directed equality check based on evaluation in the context of dependent types with large eliminations [ACD07]. In previous work [ACD08], the first author applied NbE to justify a variant of Harper and Pfenning's algorithmic equality *without erasure*. However, the meta-theoretic argument is long-winded, and there is an essential gap in the proof of transitivity of the Kripke logical relation.

In this work, we explore a novel approach to justify type-directed algorithmic equality for dependent types with predicative universes. First, we show its soundness by a Kripke model built on top of definitional equality. The Kripke logical relation yields normalization, subject reduction, and type constructor injectivity, which also imply logical consistency of IITT. Further, it proves syntactic properties such as closure under substitution, following Goguen's lead [Gog00]. The semantic proof of such syntactic properties relieves us from the deep lemma dependencies and abundant traps of syntactic meta-theory of dependent types [HP05, AC07]. Soundness of algorithmic equality entails transitivity (which is the stumbling stone), paving the way to show completeness of algorithmic equality by a second Kripke logical relation, much in the spirit of Coquand [Coq91] and Harper and Pfenning [HP05].

This article is a revised and extended version of paper *Irrelevance in Type Theory with a Heterogeneous Equality Judgement* presented at the conference FoSSaCS 2011 [Abe11]. Unfortunately, the conference version has inherited the above-mentioned gap [ACD08] in the proof of transitivity of the Kripke logical relation. This is fixed in the present article by an auxiliary Kripke model (Section 4). Further, we have dropped the heterogeneous approach to equality in favor of a standard homogeneous one. Heterogeneous equality is not necessary for the style of irrelevance we are embracing here.

2. Irrelevant Intensional Type Theory

In this section, we present *Irrelevant Intensional Type Theory* IITT which features two of Pfenning's function spaces [Pfe01], the ordinary "extensional" $(x:U) \to T$ and the proof irrelevant $(x \div U) \to T$. The main idea is that the argument of a $(x \div U) \to T$ function is counted as a proof and can neither be returned nor eliminated on, it can only be passed as argument to another proof irrelevant function or data constructor. Technically, this is realized by annotating variables as relevant, $x:U$, or irrelevant, $x \div U$, in the typing context, to confine occurrences of irrelevant variables to irrelevant arguments.

Expression and context syntax. We distinguish between relevant ($t\,u$ or simply $t\,u$) and irrelevant application ($t \mathbin{\dot{\div}} u$). Accordingly, we have relevant $(\lambda x : U.\, T)$ and irrelevant abstraction $(\lambda x \div U.\, T)$. Our choice of typed abstraction is not fundamental; a bidirectional type-checking algorithm [Coq96] can reconstruct type and relevance annotations at abstractions and applications.

$\mathsf{Var} \ni$	x, y, X, Y		
$\mathsf{Sort} \ni$	s	$::= \mathsf{Set}_k \ (k \in \mathbb{N})$	universes
$\mathsf{Ann} \ni$	$\star$	$::= \div \mid :$	annotation: irrelevant, relevant
$\mathsf{Exp} \ni$	t, u, T, U	$::= s \mid (x \star U) \overset{s,s'}{\to} T$	sort, (ir)relevant function type
		$\mid x \mid \lambda x \star U.\, t \mid t \mathbin{^\star} u$	lambda-calculus
$\mathsf{Cxt} \ni$	Γ, Δ	$::= \diamond \mid \Gamma.\, x \star T$	empty, (ir)relevant extension

Expressions are considered modulo α-equality, we write $t \equiv t'$ when we want to stress that t and t' identical (up to α). Similarly, we consider variables bound in a context to be distinct, and when opening a term binder we will implicitly use α-conversion to add a fresh variable in the context.

For technical reasons, namely, to prove transitivity (Lemma 12) of the Kripke logical relation in Section 4, we explicitly annotate function types $(x \star U) \overset{s,s'}{\to} T$ with the sorts s of domain U and s' of codomain T. We may omit the annotation if it is inessential or determined by the context of discourse. In case T does not mention x, we may write $U \to T$ for $(x:U) \to T$.

Sorts. IITT is a pure type system (PTS) with infinite hierarchy of predicative universes $\mathsf{Set}_0 : \mathsf{Set}_1 : \ldots$. The universes are not cumulative. We have the PTS axioms $\mathsf{Axiom} = \{(\mathsf{Set}_i, \mathsf{Set}_{i+1}) \mid i \in \mathbb{N}\}$ and the rules $\mathsf{Rule} = \{(\mathsf{Set}_i, \mathsf{Set}_j, \mathsf{Set}_{\max(i,j)}) \mid i, j \in \mathbb{N}\}$. As is customary, we will write the side condition $(s, s') \in \mathsf{Axiom}$ just as (s, s') and likewise $(s_1, s_2, s_3) \in \mathsf{Rule}$ just as (s_1, s_2, s_3). IITT is a full and functional PTS, which means that for all s_1, s_2 there is exactly one s_3 such that (s_1, s_2, s_3). There is no subtyping, so that types—and thus, sorts—are unique up to equality. A proof of sort unicity might relieve us from the sort annotation in function types, however, we obtain sort discrimination too late in our technical development (Lemma 36).

Substitutions. Substitutions σ are maps from variables to expressions. We require that the domain $\mathsf{dom}(\sigma) = \{x \mid \sigma(x) \neq x\}$ is finite. We write id for the identity substitution and $[u/x]$ for the singleton substitution σ such that $\sigma(x) := u$ and $\sigma(y) := y$ for $y \neq x$. Substitution extension $(\sigma, u/x)$ is formally defined as $\sigma \uplus [u/x]$. Capture avoiding parallel substitution of σ in t is written as juxtaposition $t\sigma$.

Contexts. Contexts Γ feature two kinds of bindings, relevant $(x:U)$ and irrelevant $(x \div U)$ ones. The intuition, implemented by the typing rules below, is that only relevant variables are in scope in an expression. *Resurrection* $\Gamma^{\div}$ turns all irrelevant bindings $(x \div T)$ into the corresponding relevant ones $(x:T)$ [Pfe01]. It is the tool to make irrelevant variables, also called proof variables, available in proofs. The generalization $\Gamma^{\star}$ shall mean $\Gamma^{\div}$ if $\star = \div$, and just Γ otherwise. We write $\Gamma.\Delta$ for the concatenation of Γ and Δ; herein, we suppose $\mathsf{dom}(\Gamma) \cap \mathsf{dom}(\Delta) = \emptyset$.

Primitive judgements of IITT. The following three judgements are mutually inductively defined by the rules given below and in Figure 1.

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

Derived judgements. To simplify notation, we introduce the following four abbreviations:

$$\begin{array}{lll} \Gamma \vdash t \div T & \text{iff} & \Gamma^{\div} \vdash t : T, \\ \Gamma \vdash t = t' \div T & \text{iff} & \Gamma \vdash t \div T \text{ and } \Gamma \vdash t' \div T, \\ \Gamma \vdash T & \text{iff} & \Gamma \vdash T : s \text{ for some } s, \\ \Gamma \vdash T = T' & \text{iff} & \Gamma \vdash T = T' : s \text{ for some } s. \end{array}$$

$\Gamma \vdash t \star T$ may mean $\Gamma \vdash t : T$ or $\Gamma \vdash t \div T$, depending on the value of placeholder $\star$; same for $\Gamma \vdash t = t' \star T$. We sometimes write $\Gamma \vdash t, t' \star T$ to abbreviate the conjunction of $\Gamma \vdash t \star T$ and $\Gamma \vdash t' \star T$. The notation $\Gamma \vdash T, T'$ is to be understood similarly.

2.1. **Rules.** Our rules for well-typed terms $\Gamma \vdash t : T$ extend Reed's rules [Ree02] to PTS style. There are only 6 rules; we shall introduce them one-by-one.

Variable rule. Only relevant variables can be extracted from the context.

$$\frac{\vdash \Gamma \qquad (x:U) \in \Gamma}{\Gamma \vdash x : U}$$

There is no variable rule for irrelevant bindings $(x \div U) \in \Gamma$, in particular, the judgement $x \div U \vdash x : U$ is not derivable. This essentially forbids proofs to appear in relevant positions.

Abstraction rule. Relevant and irrelevant functions are introduced analogously.

$$\frac{\Gamma.\, x \star U \vdash t : T \qquad \Gamma \vdash (x \star U) \overset{s,s'}{\to} T}{\Gamma \vdash \lambda x \star U.\, t : (x \star U) \overset{s,s'}{\to} T}$$

To check a relevant function $\lambda x{:}U.\, t$, we introduce a relevant binding $x:U$ into the context and continue checking the function body t. In case of an irrelevant function $\lambda x \div U.\, t$, we proceed with an irrelevant binding $x \div U$. This means that an irrelevant function cannot computationally depend on its argument—it is essentially a constant function. In particular, $\lambda x \div U.\, x$ is never well-typed.

As a side condition, we also need to check that the introduced function type $(x \star U) \overset{s,s'}{\to} T$ is well-sorted; the rule is given below.

Application rule.

$$\frac{\Gamma \vdash t : (x \star U) \to T \qquad \Gamma \vdash u \star U}{\Gamma \vdash t \,^{\star} u : T[u/x]}$$

This rule uses our overloaded notations for bindings $\star$, that can be specialized into two different instances for relevant and irrelevant applications.

For relevant functions, we get the ordinary dependently-typed application rule:

$$\frac{\Gamma \vdash t : (x{:}U) \to T \qquad \Gamma \vdash u : U}{\Gamma \vdash t\, u : T[u/x]}$$

When applying an irrelevant function, we resurrect the context before checking the function argument.

$$\frac{\Gamma \vdash t : (x \div U) \to T \qquad \Gamma^{\div} \vdash u : U}{\Gamma \vdash t \,^{\div} u : T[u/x]}$$

This means that irrelevant variables become relevant and can be used in u. The intuition is that the application $t \,^{\div} u$ does not computationally depend on u, thus, u may refer to any variable, even the "forbidden ones". One may think of u as a proof which may refer to both ordinary and proof variables.

For example, let $\Gamma = f : (y \div U) \to U$. Then the irrelevant η-expansion $\lambda x \div U.\, f \,^{\div} x$ is well-typed in Γ, with the following derivation:

$$\dfrac{\dfrac{\dfrac{}{\Gamma.\, x \div U \vdash f : (y \div U) \to U} \quad \dfrac{}{\Gamma.\, x : U \vdash x : U}}{\Gamma.\, x \div U \vdash f \,^{\div} x : U}}{\Gamma \vdash \lambda x \div U.\, f \,^{\div} x : (x \div U) \to U}$$

Observe how the status of x changes for irrelevant to relevant when we check the argument of f.

Sorting rules. These are the "Axioms" and the "Rules" of PTSs to form types.

$$\frac{\vdash \Gamma}{\Gamma \vdash s : s'}(s, s') \qquad \frac{\Gamma \vdash U : s_1 \qquad \Gamma.\, x \star U \vdash T : s_2}{\Gamma \vdash (x \star U) \overset{s_1,s_2}{\to} T : s_3}\ (s_1, s_2, s_3)$$

The rule for irrelevant function type formation follows Reed [Ree02].

$$\frac{\Gamma \vdash U : s_1 \qquad \Gamma.\, x \div U \vdash T : s_2}{\Gamma \vdash (x \div U) \overset{s_1,s_2}{\to} T : s_3}\ (s_1, s_2, s_3)$$

It states that the codomain of an irrelevant function cannot depend relevantly on the function argument. This fact is crucial for the construction of our semantics in Section 4. Note that it rules out *polymorphism* in the sense of Barras and Bernado's *Implicit Calculus of Constructions* ICC* [BB08] and Mishra-Linger and Sheard's *Erasure Pure Type Systems* EPTS [MLS08]; the type $(X \div \mathsf{Set}_0) \to (x : X) \to X$ is ill-formed in IITT, but not in ICC* or EPTS. In EPTS, there is the following rule:

$$\frac{\Gamma \vdash U : s_1 \qquad \Gamma.\, x{:}U \vdash T : s_2}{\Gamma \vdash (x \div U) \overset{s_1,s_2}{\to} T : s_3}\ (s_1, s_2, s_3)$$

It allows the codomain T of an irrelevant function to arbitrarily depend on the function argument x. This is fine in an erasure semantics, but incompatible with our typed semantics in the presence of large eliminations; we will detail the issues in examples 3 and 8.

Another variant is Pfenning's rule for irrelevant function type formation [Pfe01].

$$\frac{\Gamma \vdash U \div s_1 \qquad \Gamma.\, x \div U \vdash T : s_2}{\Gamma \vdash (x \div U) \overset{s_1,s_2}{\to} T : s_3}\ (s_1, s_2, s_3)$$

It allows the *domain* of an irrelevant function to make use of irrelevant variables in scope. It does not give polymorphism, e. g., $(X \div \mathsf{Set}_0) \to (x : X) \to X$ is still ill-formed. However, $(X \div \mathsf{Set}_0) \to (x \div X) \to X$ would be well-formed. It is unclear how the equality rule for irrelevant function types would look like—it is not given by Pfenning [Pfe01]. The rule

$$\frac{\Gamma \vdash U = U' \div s_1 \qquad \Gamma.\, x \div U \vdash T = T' : s_2}{\Gamma \vdash (x \div U) \overset{s_1,s_2}{\to} T = (x \div U') \overset{s_1,s_2}{\to} T' : s_3}\ (s_1, s_2, s_3)$$

would mean that any two irrelevant function types are equal as long as their codomains are equal—their domains are irrelevant. This is not compatible with our typed semantics and seems a bit problematic in general.[2]

Type conversion rule. We have *typed* conversion, thus, strictly speaking, IITT is not a PTS, but a *Pure Type System with Judgemental Equality* [Ada06].

$$\frac{\Gamma \vdash t : T \qquad \Gamma \vdash T = T'}{\Gamma' \vdash t : T'}$$

Equality. Figure 1 recapitulates the typing rules and lists the rules to derive context well-formedness $\vdash \Gamma$ and equality $\Gamma \vdash t = t' : T$. Equality is the least congruence over the β- and η-axioms. Since equality is typed we can extend IITT to include an extensional unit type (Section 7). Let us inspect the congruence rule for application:

$$\frac{\Gamma \vdash t = t' : (x \star U) \to T \qquad \Gamma \vdash u = u' \star U}{\Gamma \vdash t \star u = t' \star u' : T[u/x]}$$

In case of relevant functions ($\star = :$) we obtain the usual dependently-typed application rule of equality. Otherwise, we get:

$$\frac{\Gamma \vdash t = t' : (x \div U) \to T \qquad \Gamma^{\div} \vdash u : U \qquad \Gamma^{\div} \vdash u' : U}{\Gamma \vdash t \div u = t' \div u' : T[u/x]}$$

Note that the arguments u and u' to the irrelevant functions need to be well-typed but not related to each other. This makes precise the intuition that t and t' are constant functions.

2.2. **Simple properties of IITT.** In the following, we prove two basic invariants of derivable IITT-judgements: The context is always well-formed, and judgements remain derivable under well-formed context extensions (weakening).

Lemma 1 (Context well-formedness)**.**

(1) If $\vdash \Gamma.\, x : U.\, \Gamma'$ then $\Gamma \vdash U$.
(2) If $\Gamma \vdash t : T$ or $\Gamma \vdash t = t' : T$ then $\vdash \Gamma$.

Proof. By a simple induction on the derivations. □

[2] Maybe this is the reason why Reed [Ree02] differs from Pfenning.

Context well-formedness. $\boxed{\vdash \Gamma}$

$$\frac{}{\vdash \diamond} \qquad \frac{\vdash \Gamma \qquad \Gamma \vdash T}{\vdash \Gamma.\, x{\star}T}$$

Typing. $\boxed{\Gamma \vdash t : T}$

$$\frac{\vdash \Gamma}{\Gamma \vdash s : s'}(s, s') \qquad \frac{\Gamma \vdash U : s_1 \qquad \Gamma.\, x{\star}U \vdash T : s_2}{\Gamma \vdash (x{\star}U) \overset{s_1,s_2}{\to} T : s_3}\ (s_1, s_2, s_3)$$

$$\frac{\vdash \Gamma \qquad (x{:}U) \in \Gamma}{\Gamma \vdash x : U} \qquad \frac{\Gamma.\, x{\star}U \vdash t : T \qquad \Gamma \vdash (x{\star}U) \overset{s,s'}{\to} T}{\Gamma \vdash \lambda x{\star}U.\, t : (x{\star}U) \overset{s,s'}{\to} T}$$

$$\frac{\Gamma \vdash t : (x{\star}U) \to T \qquad \Gamma \vdash u \star U}{\Gamma \vdash t\,{}^{\star}u : T[u/x]} \qquad \frac{\Gamma \vdash t : T \qquad \Gamma \vdash T = T'}{\Gamma' \vdash t : T'}$$

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

Computation (β) and extensionality (η).

$$\frac{\Gamma.\, x{\star}U \vdash t : T \qquad \Gamma \vdash u \star U}{\Gamma \vdash (\lambda x{\star}U.\, t)\,{}^{\star}u = t[u/x] : T[u/x]} \qquad \frac{\Gamma \vdash t : (x{\star}U) \overset{s,s'}{\to} T}{\Gamma \vdash t = \lambda x{\star}U.\, t\,{}^{\star}x : (x{\star}U) \overset{s,s'}{\to} T}$$

Equivalence rules.

$$\frac{\Gamma \vdash t : T}{\Gamma \vdash t = t : T} \qquad \frac{\Gamma \vdash t = t' : T}{\Gamma \vdash t' = t : T} \qquad \frac{\Gamma \vdash t_1 = t_2 : T \qquad \Gamma \vdash t_2 = t_3 : T}{\Gamma \vdash t_1 = t_3 : T}$$

Compatibility rules.

$$\frac{\Gamma \vdash U = U' : s_1 \qquad \Gamma.\, x{\star}U \vdash T = T' : s_2}{\Gamma \vdash (x{\star}U) \overset{s_1,s_2}{\to} T = (x{\star}U') \overset{s_1,s_2}{\to} T' : s_3}\ (s_1, s_2, s_3)$$

$$\frac{\Gamma \vdash U = U' : s_1 \qquad \Gamma.\, x{\star}U \vdash T : s_2 \qquad \Gamma.\, x{\star}U \vdash t = t' : T}{\Gamma \vdash \lambda x{\star}U.\, t = \lambda x{\star}U'.\, t' : (x{\star}U) \overset{s_1,s_2}{\to} T}$$

$$\frac{\Gamma \vdash t = t' : (x{\star}U) \to T \qquad \Gamma \vdash u = u' \star U}{\Gamma \vdash t\,{}^{\star}u = t'\,{}^{\star}u' : T[u/x]}$$

Conversion rule.

$$\frac{\Gamma \vdash t = t' : T \qquad \Gamma \vdash T = T'}{\Gamma \vdash t = t' : T'}$$

Figure 1: Rules of IITT

It should be noted that we only prove the most basic well-formedness statements here. One would expect that $\Gamma \vdash t : T$ or $\Gamma \vdash t = t' : T$ also implies $\Gamma \vdash T$, or that $\Gamma \vdash t = t' : T$ implies $\Gamma \vdash t : T$. This is true—and we will refer to these implications as *syntactic validity*—but this cannot be proven without treatment of substitution, due to the typing rule for application, which requires substitution in the type, and due to the equality rule for a β-redex, which uses substitution in both term and type. Therefore, syntactic validity is delayed until Section 4 (Corollary 26), where substitution will be handled by semantic, rather than syntactic, methods.

Weakening. We can weaken a context Γ by adding bindings or making irrelevant bindings relevant. Formally, we have an order on binding annotations, which is the order induced by $: \leq \div$, and we define weakening by monotonic extension.

A well-formed context $\vdash \Delta$ *extends* a well-formed context $\vdash \Gamma$, written $\Delta \leq \Gamma$, if and only if:

$$\forall x \in \mathsf{dom}(\Gamma), \quad (x \star_1 U) \in \Gamma \implies (x \star_2 U) \in \Delta \text{ with } \star_1 \leq \star_2.$$

Note that this allows to insert new bindings or relax existing ones at any position in Γ, not just at the end.

Lemma 2 (Weakening)**.** Let $\Delta \leq \Gamma$.

(1) If $\vdash \Gamma.\Gamma'$ and $\mathsf{dom}(\Delta) \cap \mathsf{dom}(\Gamma') = \emptyset$ then $\vdash \Delta.\Gamma'$.
(2) If $\Gamma \vdash t : T$ then $\Delta \vdash t : T$.
(3) If $\Gamma \vdash t = t' : T'$ then $\Delta \vdash t = t' : T$.

Proof. Simultaneously by induction on the derivation. Let us look at some cases:

Case

$$\frac{\vdash \Gamma}{\Gamma \vdash s = s : s'}\ (s, s')$$

By assumption $\vdash \Delta$, thus $\Delta \vdash s = s : s'$.

Case

$$\frac{(x{:}U) \in \Gamma \qquad \vdash \Gamma}{\Gamma \vdash x = x : U}$$

Since $\Delta \leq \Gamma$ we have $(x{:}U) \in \Delta$, thus $\Delta \vdash x = x : U$.

Case

$$\frac{\Gamma \vdash U = U' : s_1 \qquad \Gamma.\, x{\star}U \vdash T : s_2 \qquad \Gamma.\, x{\star}U \vdash t = t' : T}{\Gamma \vdash \lambda x{\star}U.\, t = \lambda x{\star}U'.\, t' : (x{\star}U) \overset{s_1,s_2}{\to} T}$$

W. l. o. g., $x \notin \mathsf{dom}(\Delta)$. By (1) and definition of context weakening, $\Delta \leq \Gamma$ implies $\Delta.\, x{\star}U \leq \Gamma.\, x{\star}U$, so all premises can be appropriately weakened by induction hypothesis.

□

2.3. **Examples.**

Example 3 (Relevance of types)**.** [3] We can extend IITT by a unit type 1 with extensionality principle.

$$\frac{\vdash \Gamma}{\Gamma \vdash 1 : \mathsf{Set}_i} \qquad \frac{\vdash \Gamma}{\Gamma \vdash () : 1} \qquad \frac{\Gamma \vdash t : 1 \qquad \Gamma \vdash t' : 1}{\Gamma \vdash t = t' : 1}$$

Typed equality allows us to equate all inhabitants of the unit type. As a consequence, the Church numerals over the unit type all coincide, e. g.,

$$\begin{aligned}&\Gamma \vdash \lambda f{:}1 \to 1.\, \lambda x{:}1.\, x\\ &\quad = \lambda f{:}1 \to 1.\, \lambda x{:}1.\, f\, x \;:\; (1 \to 1) \to 1 \to 1.\end{aligned}$$

In systems with untyped equality, like ICC* and EPTS, these terms erase to untyped Church-numerals $\lambda f \lambda x.x$ and $\lambda f \lambda x.\, f\, x$ and are necessarily distinguished.

If we trade the unit type for Bool or any other type with more than one inhabitant, the two terms become different in IITT. This means that in IITT, types are relevant, and we need to reject irrelevant quantification over types like in $(X \div \mathsf{Set}_0) \to (X \to X) \to X \to X$. In IITT, the polymorphic types of Church numerals are $(X : \mathsf{Set}_i) \to (X \to X) \to X \to X$.

Example 4 (Σ-types)**.** IITT can be readily extended by weak Σ-types.

$$\frac{\Gamma \vdash U : s_1 \qquad \Gamma.\, x{\star}U \vdash T : s_2}{\Gamma \vdash (x{\star}U) \times T : s_3}\ (s_1, s_2, s_3)$$

$$\frac{\Gamma \vdash u \star U \qquad \Gamma \vdash t : T[u/x] \qquad \Gamma \vdash (x{\star}U) \times T}{\Gamma \vdash (u, t) : (x{\star}U) \times T}$$

$$\frac{\Gamma \vdash p : (x{\star}U) \times T \qquad \Gamma.\, x{\star}U.\, y{:}T \vdash v : V}{\Gamma \vdash \mathsf{let}\ (x, y) = p\ \mathsf{in}\ v : V}$$

$$\frac{\Gamma \vdash u \star U \qquad \Gamma \vdash t : T[u/x] \qquad \Gamma.\, x{\star}U.\, y{:}T \vdash v : V \qquad \Gamma \vdash (x{\star}U) \times T}{\Gamma \vdash (\mathsf{let}\ (x, y) = (u, t)\ \mathsf{in}\ v) = v[u/x][t/y] : V}$$

Additional laws for equality could be considered, like commuting conversions, or the identity $(\mathsf{let}\ (x, y) = p\ \mathsf{in}\ (x, y)) = p$. The relevant form $(x : U) \times T$ admits a strong version with projections fst and snd and full extensionality $p = (\mathsf{fst}\, p, \mathsf{snd}\, p) : (x : U) \times T$. However, strong irrelevant Σ-types $(x \div U) \times T$ are problematic because of the first projection:

$$\frac{\Gamma \vdash p : (x \div U) \times T}{\Gamma \vdash \mathsf{fst}\, p \div U}$$

With our definition of $\Gamma \vdash u \div U$ as $\Gamma^{\div} \vdash u : U$, this rule is misbehaved: it allows us get hold of an irrelevant value in a relevant context. We could define a closed function $\pi_1 : (x \div U) \times 1 \to U$, and composing it with $(\_, ()) : (x \div U) \to (x \div U) \times 1$ would give us an identity function of type $(x \div U) \to U$ which magically makes irrelevant things relevant and IITT inconsistent. In this article, we will not further consider strong Σ-types with irrelevant components; we leave the in-depth investigation to future work.

Example 5 (Squash type)**.** The *squash* type $\|T\|$ was first introduced in the context of NuPRL [CAB+86]; it contains exactly one inhabitant iff T is inhabited. Semantically, one obtains $\|T\|$ from T by equating all of T's inhabitants. In IITT, we can define $\|T\|$ as

[3] Example suggested by a reviewer of this paper.

internalization of the irrelevance modality, as already suggested by Pfenning [Pfe01]. The first alternative is via the weak irrelevant Σ-type.

$$
\begin{array}{lcl}
\|\_\| & : & \mathsf{Set}_i \to \mathsf{Set}_i \\
\|\_\| & := & \lambda T{:}\mathsf{Set}_i.\,(\_ \mathbin{\dot{\div}} T) \times 1 \\
{[\_]} & : & (x \mathbin{\dot{\div}} T) \to \|T\| \\
{[\_]} & := & \lambda x \mathbin{\dot{\div}} T.\,(x, ()) \\
\mathsf{sqelim} & : & (T{:}\mathsf{Set}_i) \to \\
 & & (P{:}\|T\| \to \mathsf{Set}_j) \to \\
 & & (f{:}(x \mathbin{\dot{\div}} T) \to P\,([x])) \to \\
 & & (t{:}\|T\|) \to \\
 & & P\,t \\
\mathsf{sqelim} & := & \lambda T{:}\mathsf{Set}_i. \\
 & & \lambda P{:}\|T\| \to \mathsf{Set}_j. \\
 & & \lambda f{:}(x \mathbin{\dot{\div}} T) \to P\,([x]). \\
 & & \lambda t{:}\|T\|. \\
 & & \mathsf{let}\ (x, \_) = t\ \mathsf{in}\ f \mathbin{\dot{\div}} x
\end{array}
$$

It is not hard to see that $\|\_\|$ is a monad. All *canonical* inhabitants of $\|T\|$ are definitionally equal:

$$\frac{\Gamma \vdash t, t' \mathbin{\dot{\div}} T}{\Gamma \vdash [t] = [t'] : \|T\|}$$

This is easily shown by expanding the definition of $[\_]$ and using the congruence rule for pairs with an irrelevant first component.

However, we cannot show that *all* inhabitants of $\|T\|$ are definitionally equal, because of the missing extensionality principles for weak Σ. Thus, the second alternative is to add the squash type to IITT via the rules:

$$\frac{\Gamma \vdash T : \mathsf{Set}_i}{\Gamma \vdash \|T\| : \mathsf{Set}_i} \qquad \frac{\Gamma \vdash t \mathbin{\dot{\div}} T}{\Gamma \vdash [t] : \|T\|} \qquad \frac{\Gamma \vdash t : \|T\| \qquad \Gamma.\, x \mathbin{\dot{\div}} T \vdash v : V}{\Gamma \vdash \mathsf{let}\ [x] = t\ \mathsf{in}\ v : V}$$

$$\frac{\Gamma \vdash t, t' : \|T\|}{\Gamma \vdash t = t' : \|T\|} \qquad \frac{\Gamma \vdash t \mathbin{\dot{\div}} T \qquad \Gamma.\, x \mathbin{\dot{\div}} T \vdash v : V}{\Gamma \vdash (\mathsf{let}\ [x] = [t]\ \mathsf{in}\ v) = v[t/x] : V}$$

Our model (Section 4) is ready to interpret these rules, as well as normalization-by-evaluation inspired models [ACP11].

Example 6 (Subset type)**.** The subset type $\{x : U \mid T\}$ is definable from Σ and squash as $(x{:}U) \times \|T\|$.

To discuss the next example, we consider a further extension of IITT by Leibniz equality and natural numbers:

$$
\begin{array}{lll}
a \equiv b & : \mathsf{Set}_i & \text{for } A : \mathsf{Set}_i \text{ and } a, b : A \\
\mathsf{refl} & : a \equiv a & \text{for } A : \mathsf{Set}_i \text{ and } a : A \\
 & & \\
\mathsf{Nat} & : \mathsf{Set}_i & \\
0, 1, \ldots & : \mathsf{Nat} & \\
+, * & : \mathsf{Nat} \to \mathsf{Nat} \to \mathsf{Nat}. &
\end{array}
$$

Example 7 (Composite)**.** [4] Let the set of composite numbers $\{4, 6, 8, 9, 10, 12, 14, 15, \dots\}$ be numbers that are the product of two natural numbers ≥ 2.

$$\mathsf{Composite} = \{n : \mathsf{Nat} \mid (k : \mathsf{Nat}) \times (l : \mathsf{Nat}) \times (n \equiv (k + 2) * (l + 2))\}$$

Most composite numbers have several factorizations, and thanks to irrelevance the specific composition is ignored when handling composite numbers. For instance, 12 as product of 3 and 4 is not distinguished from the 12 as product of 2 and 6.

$$(12, [(1, (2, \mathsf{refl}))]) = (12, [(0, (4, \mathsf{refl}))]) : \mathsf{Composite}.$$

Example 8 (Large eliminations)**.** [5] The ICC* [BB08] or EPTS [MLS08] irrelevant function type $(x \mathbin{\dot{:}} A) \to B$ allows x to appear *relevantly* in B. This extra power raises some issues with large eliminations. Consider

$$\begin{aligned}
\mathsf{T} \quad &: \mathsf{Bool} \to \mathsf{Set}_0 \\
\mathsf{T}\ \mathsf{true} \ &= \mathsf{Bool} \to \mathsf{Bool} \\
\mathsf{T}\ \mathsf{false} \ &= \mathsf{Bool} \\
t \quad &= \lambda F : (b \mathbin{\dot{:}} \mathsf{Bool}) \to (\mathsf{T}\, b \to \mathsf{T}\, b) \to \mathsf{Set}_0. \\
&\quad\ \lambda g : (F \mathbin{\dot{:}} \mathsf{false}\ (\lambda x : \mathsf{Bool}.\, x)) \to \mathsf{Bool}. \\
&\quad\ \lambda a : F \mathbin{\dot{:}} \mathsf{true}\ (\lambda x : \mathsf{Bool} \to \mathsf{Bool}. \lambda y : \mathsf{Bool}.\, x\, y).\ g\, a.
\end{aligned}$$

The term t is well-typed in ICC* + T because the domain type of g and the type of a are $\beta\eta$-equal after erasure $(-)^*$ of type annotations and irrelevant arguments:

$$\begin{aligned}
(F \mathbin{\dot{:}} \mathsf{false}\ (\lambda x : \mathsf{Bool}.\, x))^* &= F\ (\lambda x x) \\
&=_{\beta\eta} F\ (\lambda x \lambda y.\, x\, y) = (F \mathbin{\dot{:}} \mathsf{true}\ (\lambda x : \mathsf{Bool} \to \mathsf{Bool}. \lambda y : \mathsf{Bool}.\, x\, y))^*
\end{aligned}$$

While a Curry view supports this, it is questionable whether identity functions at different types should be viewed as one. It is unclear how a type-directed equality algorithm (see Sec. 3) should proceed here; it needs to recognize that $x : \mathsf{Bool}$ is equal to $\lambda y : \mathsf{Bool}.\, x\, y : \mathsf{Bool} \to \mathsf{Bool}$. This situation is amplified by a unit type 1 with extensional equality. When we change $\mathsf{T}\,\mathsf{true}$ to 1 and the type of a to $F \mathbin{\dot{:}} \mathsf{true}\ (\lambda x : 1.\, ())$ then t should still type-check, because $\lambda x.\, ()$ is the identity function on 1. However, η-equality for 1 cannot be checked without types, and a type-directed algorithm would end up checking (successfully) $x : \mathsf{Bool}$ for equality with $() : 1$. This algorithmic equality cannot be transitive, because then any two booleans would be equal.

Summarizing, we may conclude that the type of F bears trouble and needs to be rejected. IITT does this because it forbids the irrelevant b in relevant positions such as $\mathsf{T}\, b$; ICC* lacks T altogether. Extensions of ICC* should at least make sure that b is never eliminated, such as in $\mathsf{T}\, b$. Technically, T would have to be put in a separate class of *recursive* functions, those that actually compute with their argument. We leave the interaction of the three different function types to future research.

[4] Example suggested by reviewer.

[5] Inspired by discussions with Ulf Norell during the 11th Agda Implementers' Meeting.

3. Algorithmic Equality

The algorithm for checking equality in IITT is inspired by Harper and Pfenning [HP05]. Like theirs, it is type-directed, but we are using the full dependent type and not an erasure to simple types (which would anyway not work due to large eliminations). We give the algorithm in form of judgements and rules in direct correspondence to a functional program.

Algorithmic equality is meant to be used as part of a type checking algorithm. It is the algorithmic counterpart of the definitional conversion rule; in particular, it will only be called on terms that are already know to be well-typed – in fact, types that are well-sorted. We rely on this precondition in the algorithmic formulation.

Algorithmic equality consists of three interleaved judgements. A *type equality* test checks equality between two types, by inspecting their weak head normal forms. Terms found inside dependent types are reduced and the resulting neutral terms are compared by *structural equality*. The head variable of such neutrals provides type information that is then used to check the (non-normal) arguments using *type-directed equality*, by reasoning on the (normalized) type structure to perform η-expansions on product types. After enough expansions, a base type is reached, where structural equality is called again, or a sort, at which we use type equality.

Informally, the interleaved reductions are the algorithmic counterparts of the β-equality axiom, the type and structural equalities account for the compatibility rules, and type-directed equality corresponds to the η-equality axiom. The remaining equivalence rules are emergent global properties of the algorithm.

Weak head reduction. Weak head normal forms (whnfs) are given by the following grammar:

$$\begin{array}{llll} \mathsf{Whnf} & \ni & a, b, f, A, B, F & ::= \quad s \mid (x{\star}U) \overset{s,s'}{\to} T \mid \lambda x{\star}U.\, t \mid n \quad \text{whnf} \\ \mathsf{Wne} & \ni & n, N & ::= \quad x \mid n \mathbin{^\star} u \quad \text{neutral whnf} \end{array}$$

Weak head evaluation $t \searrow a$ and active application $f \mathbin{@^\star} u \searrow a$ are functional relations given by the following rules.

$$\frac{t \searrow f \qquad f \mathbin{@^\star} u \searrow a}{t \mathbin{^\star} u \searrow a} \qquad \frac{}{a \searrow a} \qquad \frac{t[u/x] \searrow a}{(\lambda x{\star}U.\, t) \mathbin{@^\star} u \searrow a} \qquad \frac{}{n \mathbin{@^\star} u \searrow n \mathbin{^\star} u}$$

Instead of writing the propositions $t \searrow a$ and $P[a]$ we will sometimes simply write $P[\downarrow t]$. Similarly, we might write $P[f \mathbin{@^\star} u]$ instead of $f \mathbin{@^\star} u \searrow a$ and $P[a]$. In rules, it is understood that the evaluation judgement is always an extra premise, never an extra conclusion.

Algorithmic equality is given as type equality, structural equality, and type-directed equality, which are mutually recursive. The equality algorithm is only invoked on well-formed expressions of the correct type.

Type equality. Type equality $\Delta \vdash A \Longleftrightarrow A'$, for weak head normal forms, and $\Delta \vdash T \mathrel{\widehat{\Longleftrightarrow}} T'$, for arbitrary well-formed types, checks that two given types are equal in their respective contexts.

$$\frac{\Delta \vdash \downarrow T \Longleftrightarrow \downarrow T'}{\Delta \vdash T \mathrel{\widehat{\Longleftrightarrow}} T'} \qquad \frac{\Delta \vdash N \mathrel{\widehat{\longleftrightarrow}} N' : T}{\Delta \vdash N \Longleftrightarrow N'}$$

$$\frac{}{\Delta \vdash s \Longleftrightarrow s} \qquad \frac{\Delta \vdash U \mathrel{\widehat{\Longleftrightarrow}} U' \qquad \Delta.\, x{:}U \vdash T \mathrel{\widehat{\Longleftrightarrow}} T'}{\Delta \vdash (x{\star}U) \overset{s,s'}{\to} T \Longleftrightarrow (x{\star}U') \overset{s,s'}{\to} T'}$$

Note that when invoking structural equality on neutral types N and N', we do not care which type T is returned, since we know by well-formedness that N and N' must have the same sort.

Structural equality. Structural equality $\Delta \vdash n \longleftrightarrow n' : A$ and $\Delta \vdash n \mathrel{\widehat{\longleftrightarrow}} n' : T$ checks the neutral expressions n and n' for equality and at the same time infers their type, which is returned as output.

$$\frac{\Delta \vdash n \mathrel{\widehat{\longleftrightarrow}} n' : T}{\Delta \vdash n \longleftrightarrow n' : \downarrow T} \qquad \frac{(x{:}T) \in \Delta}{\Delta \vdash x \mathrel{\widehat{\longleftrightarrow}} x : T}$$

$$\frac{\Delta \vdash n \longleftrightarrow n' : (x{:}U) \to T \qquad \Delta \vdash u \mathrel{\widehat{\Longleftrightarrow}} u' : U}{\Delta \vdash n\, u \mathrel{\widehat{\longleftrightarrow}} n'\, u' : T[u/x]} \qquad \frac{\Delta \vdash n \longleftrightarrow n' : (x \div U) \to T}{\Delta \vdash n \div u \mathrel{\widehat{\longleftrightarrow}} n' \div u' : T[u/x]}$$

Type-directed equality. Type-directed equality $\Delta \vdash t \Longleftrightarrow t' : A$ and $\Delta \vdash t \mathrel{\widehat{\Longleftrightarrow}} t' : T$ checks terms t and t' for equality and proceeds by the structure of the supplied type, to account for η.

$$\frac{\Delta \vdash t \Longleftrightarrow t' : \downarrow T}{\Delta \vdash t \mathrel{\widehat{\Longleftrightarrow}} t' : T} \qquad \frac{\Delta.\, x \star U \vdash t\, {}^\star x \mathrel{\widehat{\Longleftrightarrow}} t'\, {}^\star x : T}{\Delta \vdash t \Longleftrightarrow t' : (x \star U) \to T}$$

$$\frac{\Delta \vdash T \mathrel{\widehat{\Longleftrightarrow}} T'}{\Delta \vdash T \Longleftrightarrow T' : s} \qquad \frac{\Delta \vdash \downarrow t \mathrel{\widehat{\longleftrightarrow}} \downarrow t' : T}{\Delta \vdash t \Longleftrightarrow t' : N}$$

Note that in the but-last rule we do not check that the inferred type T of $\downarrow t$ equals the ascribed type N. Since algorithmic equality is only invoked for well-typed t, we know that this must always be the case. Skipping this test is a conceptually important improvement over Harper and Pfenning [HP05].

Due to dependent typing, it is not obvious that algorithmic equality is symmetric and transitive. For instance, consider symmetry in case of application: We have to show that $\Delta \vdash n'\, u' \mathrel{\widehat{\longleftrightarrow}} n\, u : T[u/x]$, but using the induction hypothesis we obtain this equality only at type $T[u'/x]$. To conclude, we need to convert types, which is only valid if we know that u and u' are actually equal. Thus, we need soundness of algorithmic equality to show its transitivity. Soundness w. r. t. declarative equality requires subject reduction, which is not trivial, due to its dependency on function type injectivity. In the next section (4), we construct by a Kripke logical relation which gives us subject reduction and soundness of algorithmic equality (Section 5), and, finally, symmetry and transitivity of algorithmic equality.

A simple fact about algorithmic equality is that the inferred types are unique up to syntactic equality (where we consider α-convertible expressions as identical). Also, they only depend on the left hand side neutral term n.

Lemma 9 (Uniqueness of inferred types)**.**

(1) If $\Delta \vdash n \longleftrightarrow n_1 : A_1$ and $\Delta \vdash n \longleftrightarrow n_2 : A_2$ then $A_1 \equiv A_2$.
(2) If $\Delta \vdash n \mathrel{\widehat{\longleftrightarrow}} n_1 : T_1$ and $\Delta \vdash n \mathrel{\widehat{\longleftrightarrow}} n_2 : T_2$ then $T_1 \equiv T_2$.

Extending structural equality to irrelevance, we let

$$\frac{\Delta^\div \vdash n \longleftrightarrow n : A \qquad \Delta^\div \vdash n' \longleftrightarrow n' : A}{\Delta \vdash n \longleftrightarrow n' \div A}$$

and analogously for $\Delta \vdash n \mathrel{\widehat{\longleftrightarrow}} n' \div T$.

4. A Kripke Logical Relation for Soundness

In this section, we construct a Kripke logical relation in the spirit of Goguen [Gog00] and Vanderwaart and Crary [VC02] that proves weak head normalization, function type injectivity, and subject reduction plus syntactical properties like substitution in judgements and syntactical validity. As an important consequence, we obtain soundness of algorithmic equality w. r. t. definitional equality. This allows us to establish that algorithmic equality on well-typed terms is a partial equivalence relation.

4.1. **An Induction Measure.** Following Goguen [Gog94] and previous work [ACD08], we first define a semantic universe hierarchy U_i whose sole purpose is to provide a measure for defining a logical relation and proving some of its properties. The limit U_ω corresponds to the proof-theoretic strength or ordinal of IITT.

We denote sets of expressions by $\mathcal{A}, \mathcal{B}$ and functions from expressions to sets of expressions by $\mathcal{F}$. Let $\widehat{\mathcal{A}} = \{t \mid \downarrow t \in \mathcal{A}\}$ denote the closure of $\mathcal{A}$ by weak head expansion. The dependent function space is defined as $\Pi\,\mathcal{A}\,\mathcal{F} = \{f \in \mathsf{Whnf} \mid \forall u \in \widehat{\mathcal{A}}.\, f \mathbin{@} u \in \mathcal{F}(u)\}$.

By recursion on $i \in \mathbb{N}$ we define inductively sets $\mathsf{U}_i \subseteq \mathsf{Whnf} \times \mathcal{P}(\mathsf{Whnf})$ as follows [ACD08, Sec. 5.1]:

$$\frac{}{(N, \mathsf{Wne}) \in \mathsf{U}_i} \qquad \frac{}{(\mathsf{Set}_j, |\mathsf{U}_j|) \in \mathsf{U}_i}\ (\mathsf{Set}_j, \mathsf{Set}_i) \in \mathsf{Axiom}$$

$$\frac{(U, \mathcal{A}) \in \widehat{\mathsf{U}_i} \qquad \forall u \in \widehat{\mathcal{A}}.\, (T[u/x], \mathcal{F}(u)) \in \widehat{\mathsf{U}_j}}{((x \star U) \to T, \Pi\,\mathcal{A}\,\mathcal{F}) \in \mathsf{U}_k}\ (\mathsf{Set}_i, \mathsf{Set}_j, \mathsf{Set}_k) \in \mathsf{Rule}$$

Herein, $\widehat{\mathsf{U}_i} = \{(T, \mathcal{A}) \mid (\downarrow T, \mathcal{A}) \in \mathsf{U}_i\}$ and $|\mathsf{U}_j| = \{A \mid (A, \mathcal{A}) \in \mathsf{U}_j \text{ for some } \mathcal{A}\}$. Only interested in computational strength, we treat relevant and irrelevant function spaces alike—at the level of *predicates* $\mathcal{A}$, irrelevance is anyhow not observable, only by *relations* as given later.

The induction measure $A \in \mathsf{Set}_i$ shall now mean the minimum height of a derivation of $(A, \mathcal{A}) \in \mathsf{U}_i$ for some $\mathcal{A}$. Note that due to universe stratification, $A \in \mathsf{Set}_i$ is smaller than $\mathsf{Set}_i \in \mathsf{Set}_j$.

4.2. **A Kripke Logical Relation.** Let $\Delta \vdash t :=: t' \star T$ stand for the conjunction of the propositions

- $\Delta \vdash t \star T$ and $\Delta \vdash t' \star T$, and
- $\Delta \vdash t = t' \star T$.

By induction on $A \in s$ we define two Kripke relations

$$\Delta \vdash A \mathbin{Ⓢ} A' : s$$
$$\Delta \vdash a \mathbin{Ⓢ} a' : A.$$

together with their respective closures $\widehat{Ⓢ}$ and the generalization to $\star$. For better readability, the clauses are given in rule form meaning that the conclusion *is defined as* the conjunction

of the premises. $\forall$ and $\implies$ are meta-level quantification and implication, respectively.

$$\frac{\Delta \vdash N :=: N' : s}{\Delta \vdash N \mathbin{Ⓢ} N' : s} \qquad \frac{\Delta \vdash n :=: n' : N}{\Delta \vdash n \mathbin{Ⓢ} n' : N} \qquad \frac{\vdash \Delta}{\Delta \vdash s \mathbin{Ⓢ} s : s'}\ (s, s')$$

$$\frac{\begin{array}{l}\Delta \vdash U \mathbin{\widehat{Ⓢ}} U' : s_1 \\ \forall \Gamma \leq \Delta,\ \Gamma \vdash u \mathbin{\widehat{Ⓢ}} u' \star U \implies \Gamma \vdash T[u/x] \mathbin{\widehat{Ⓢ}} T'[u'/x] : s_2 \\ \Delta \vdash (x \star U) \overset{s_1,s_2}{\to} T :=: (x \star U') \overset{s_1,s_2}{\to} T' : s_3\end{array}}{\Delta \vdash (x \star U) \overset{s_1,s_2}{\to} T \mathbin{Ⓢ} (x \star U') \overset{s_1,s_2}{\to} T' : s_3}\ (s_1, s_2, s_3)$$

$$\frac{\begin{array}{c}\forall \Gamma \leq \Delta,\ \Gamma \vdash u \mathbin{\widehat{Ⓢ}} u' \star U \implies \Gamma \vdash f \star u \mathbin{\widehat{Ⓢ}} f' \star u' : T[u/x] \\ \Delta \vdash f :=: f' : (x \star U) \overset{s,s'}{\to} T\end{array}}{\Delta \vdash f \mathbin{Ⓢ} f' : (x \star U) \overset{s,s'}{\to} T}$$

$$\frac{\begin{array}{ccccc} & & T \searrow A & \Delta \vdash T = A & \\ t \searrow a & \Delta \vdash t = a : A & & \Delta \vdash t' = a' : A & t' \searrow a' \\ & & \Delta \vdash a \mathbin{Ⓢ} a' : A & & \\ & & \Delta \vdash t :=: t' : T & & \end{array}}{\Delta \vdash t \mathbin{\widehat{Ⓢ}} t' : T}$$

$$\frac{\Delta^{\div} \vdash a \mathbin{Ⓢ} a : A \qquad \Delta^{\div} \vdash a' \mathbin{Ⓢ} a' : A}{\Delta \vdash a \mathbin{Ⓢ} a' \div A} \qquad \frac{\Delta^{\div} \vdash t \mathbin{\widehat{Ⓢ}} t : T \qquad \Delta^{\div} \vdash t' \mathbin{\widehat{Ⓢ}} t' : T}{\Delta \vdash t \mathbin{\widehat{Ⓢ}} t' \div T}$$

It is immediate that the logical relation contains only well-typed and definitionally equal terms. We will demonstrate that it is also closed under weakening and conversion, symmetric and transitive.

Lemma 10 (Weakening)**.**

(1) If $\Delta \vdash a \mathbin{Ⓢ} a' : A$ and $\Gamma \leq \Delta$ then there exists a derivation of $\Gamma \vdash a \mathbin{Ⓢ} a' : A$ with the same height.
(2) Analogously for $\Delta \vdash t \mathbin{\widehat{Ⓢ}} t' : T$.

Proof. By induction on $A \in s$ and $T \in s$, resp. □

Lemma 11 (Type conversion)**.**

(1) If $\Gamma \vdash A \mathbin{Ⓢ} A' : s$ then $\Gamma \vdash a \mathbin{Ⓢ} a' : A$ iff $\Gamma \vdash a \mathbin{Ⓢ} a' : A'$.
(2) If $\Gamma \vdash T \mathbin{\widehat{Ⓢ}} T' : s$ then $\Gamma \vdash t \mathbin{\widehat{Ⓢ}} t' : T$ iff $\Gamma \vdash t \mathbin{\widehat{Ⓢ}} t' : T'$.

Proof. Simultaneously induction in $A \in s$ and $T \in s$, resp. We show the "if" direction, the "only if" follows analogously. The interesting case is the one of functions.

Case

$$\frac{\begin{array}{c}\Delta \vdash U \mathbin{\widehat{\mathsf{S}}} U' : s_1 \\ \forall \Gamma \leq \Delta,\ \Gamma \vdash u \mathbin{\widehat{\mathsf{S}}} u' \star U \implies \Gamma \vdash T[u/x] \mathbin{\widehat{\mathsf{S}}} T'[u'/x] : s_2 \\ \Delta \vdash (x \star U) \overset{s_1,s_2}{\to} T :=: (x \star U') \overset{s_1,s_2}{\to} T' : s_3\end{array}}{\Delta \vdash (x \star U) \overset{s_1,s_2}{\to} T \mathbin{\mathsf{S}} (x \star U') \overset{s_1,s_2}{\to} T' : s_3}$$

$$\frac{\begin{array}{c}\forall \Gamma \leq \Delta,\ \Gamma \vdash u \mathbin{\widehat{\mathsf{S}}} u' \star U \implies \Gamma \vdash f \star u \mathbin{\widehat{\mathsf{S}}} f' \star u' : T[u/x] \\ \Delta \vdash f :=: f' : (x \star U) \overset{s,s'}{\to} T\end{array}}{\Delta \vdash f \mathbin{\mathsf{S}} f' : (x \star U) \overset{s,s'}{\to} T}$$

First, $\Delta \vdash f :=: f' : (x \star U') \overset{s,s'}{\to} T'$, holds because of the conversion rule for typing and equality. Now assume arbitrary $\Gamma \leq \Delta$ and $\Gamma \vdash u \mathbin{\widehat{\mathsf{S}}} u' \star U'$ and show $\Gamma \vdash f \star u \mathbin{\widehat{\mathsf{S}}} f' \star u' : T'[u/x]$. By induction hypothesis on $U \in s_1$ we have $\Gamma \vdash u \mathbin{\widehat{\mathsf{S}}} u' \star U$, thus, $\Gamma \vdash f \star u \mathbin{\widehat{\mathsf{S}}} f' \star u' : T[u/x]$ by assumption. By induction hypothesis on $T[u/x] \in s_2$ we obtain $\Gamma \vdash f \star u \mathbin{\widehat{\mathsf{S}}} f' \star u' : T'[u/x]$.

□

Lemma 12 (Symmetry and Transitivity)**.** Let $\Delta \vdash T \mathbin{\widehat{\mathsf{S}}} T : s$.

(1) If $\Delta \vdash t \mathbin{\widehat{\mathsf{S}}} t' : T$ then $\Delta \vdash t' \mathbin{\widehat{\mathsf{S}}} t : T$.
(2) If $\Delta \vdash t_1 \mathbin{\widehat{\mathsf{S}}} t_2 : T$ and $\Delta \vdash t_2 \mathbin{\widehat{\mathsf{S}}} t_3 : T$ then $\Delta \vdash t_1 \mathbin{\widehat{\mathsf{S}}} t_3 : T$.

Proof. We generalize the two statements to whnfs $\Delta \vdash A \mathbin{\mathsf{S}} A : s$ and prove all four statements simultaneously by induction in $A \in s$ and $T \in s$, resp.

Case Let us look at the case for functions.

$$\frac{\begin{array}{c}\Delta \vdash U \mathbin{\widehat{\mathsf{S}}} U : s_1 \\ \forall \Gamma \leq \Delta,\ \Gamma \vdash u \mathbin{\widehat{\mathsf{S}}} u' \star U \implies \Gamma \vdash T[u/x] \mathbin{\widehat{\mathsf{S}}} T[u'/x] : s_2 \\ \Delta \vdash (x \star U) \overset{s_1,s_2}{\to} T :=: (x \star U) \overset{s_1,s_2}{\to} T : s_3\end{array}}{\Delta \vdash (x \star U) \overset{s_1,s_2}{\to} T \mathbin{\mathsf{S}} (x \star U) \overset{s_1,s_2}{\to} T : s_3}$$

Subcase Symmetry:

$$\frac{\begin{array}{c}\forall \Gamma \leq \Delta,\ \Gamma \vdash u \mathbin{\widehat{\mathsf{S}}} u' \star U \implies \Gamma \vdash f \star u \mathbin{\widehat{\mathsf{S}}} f' \star u' : T[u/x] \\ \Delta \vdash f :=: f' : (x \star U) \to T\end{array}}{\Delta \vdash f \mathbin{\mathsf{S}} f' : (x \star U) \to T}$$

To show $\Delta \vdash f' \mathbin{\mathsf{S}} f : (x \star U) \to T$, assume arbitrary $\Gamma \leq \Delta$ and $\Gamma \vdash u' \mathbin{\widehat{\mathsf{S}}} u \star U$ and show $\Gamma \vdash f' \star u' \mathbin{\widehat{\mathsf{S}}} f \star u : T[u'/x]$. By induction hypothesis on $U \in s_2$, with weakened $\Gamma \vdash U \mathbin{\widehat{\mathsf{S}}} U : s_1$, we have $\Gamma \vdash u \mathbin{\widehat{\mathsf{S}}} u' \star U$, thus, $\Gamma \vdash f \star u \mathbin{\widehat{\mathsf{S}}} f' \star u' : T[u/x]$ by assumption. Using symmetry and transitivity on U we obtain $\Gamma \vdash u \mathbin{\widehat{\mathsf{S}}} u \star U$, thus, $\Gamma \vdash T[u/x] \mathbin{\widehat{\mathsf{S}}} T[u/x] : s_2$. By induction hypothesis on $T[u/x] \in s_2$ we apply symmetry to obtain $\Gamma \vdash f' \star u' \mathbin{\widehat{\mathsf{S}}} f \star u : T[u/x]$, and since $\Gamma \vdash T[u/x] \mathbin{\widehat{\mathsf{S}}} T[u'/x] : s_2$ we conclude by type conversion (Lemma 11).

Subcase Transitivity:

$$\frac{\forall \Gamma \leq \Delta,\ \Gamma \vdash u \mathrel{\widehat{\textcircled{S}}} u' \star U \implies \Gamma \vdash f_1\ {}^{\star}u \mathrel{\widehat{\textcircled{S}}} f_2\ {}^{\star}u' : T[u/x] \qquad \Delta \vdash f_1 :=: f_2 : (x \star U) \overset{s,s'}{\to} T}{\Delta \vdash f_1 \mathbin{\textcircled{S}} f_2 : (x \star U) \overset{s,s'}{\to} T}$$

$$\frac{\forall \Gamma \leq \Delta,\ \Gamma \vdash u \mathrel{\widehat{\textcircled{S}}} u' \star U \implies \Gamma \vdash f_2\ {}^{\star}u \mathrel{\widehat{\textcircled{S}}} f_3\ {}^{\star}u' : T[u/x] \qquad \Delta \vdash f_2 :=: f_3 : (x \star U) \overset{s,s'}{\to} T}{\Delta \vdash f_2 \mathbin{\textcircled{S}} f_3 : (x \star U) \overset{s,s'}{\to} T}$$

We wish to prove that $\Delta \vdash f_1 \mathbin{\textcircled{S}} f_3 : (x \star U) \overset{s,s'}{\to} T$. We get $\Delta \vdash f_1 :=: f_3 : (x \star U) \overset{s,s'}{\to} T$ immediately by transitivity of definitional equality. Given $\Gamma \leq \Delta$ and $\Gamma \vdash u \mathrel{\widehat{\textcircled{S}}} u' \star U$, we need to show that $\Gamma \vdash f_1\ {}^{\star}u \mathrel{\widehat{\textcircled{S}}} f_3\ {}^{\star}u' : T[u/x]$.
As $\Gamma \vdash \_ \mathrel{\widehat{\textcircled{S}}} \_ : U$ is a PER by induction hypothesis, we have $\Gamma \vdash u \mathrel{\widehat{\textcircled{S}}} u \star U$, which entails $f_1\ {}^{\star}u \mathrel{\widehat{\textcircled{S}}} f_2\ {}^{\star}u : T[u/x]$. From $\Gamma \vdash u \mathrel{\widehat{\textcircled{S}}} u' \star U$ also have $\Gamma \vdash f_2\ {}^{\star}u \mathrel{\widehat{\textcircled{S}}} f_3\ {}^{\star}u' : T[u/x]$, which allows to conclude $\Gamma \vdash f_1\ {}^{\star}u \mathrel{\widehat{\textcircled{S}}} f_3\ {}^{\star}u' : T[u/x]$ by transitivity at $T[u/x]$.

Case Now, we consider function spaces:

Subcase Transitivity:

$$\frac{\Delta \vdash U_1 \mathrel{\widehat{\textcircled{S}}} U_2 : s_1 \qquad \forall \Gamma \leq \Delta,\ \Gamma \vdash u \mathrel{\widehat{\textcircled{S}}} u' \star U_1 \implies \Gamma \vdash T_1[u/x] \mathrel{\widehat{\textcircled{S}}} T_2[u'/x] : s_2 \qquad \Delta \vdash (x \star U_1) \overset{s_1,s_2}{\to} T_1 :=: (x \star U_2) \overset{s_1,s_2}{\to} T_2 : s_3}{\Delta \vdash (x \star U_1) \overset{s_1,s_2}{\to} T_1 \mathbin{\textcircled{S}} (x \star U_2) \overset{s_1,s_2}{\to} T_2 : s_3}$$

$$\frac{\Delta \vdash U_2 \mathrel{\widehat{\textcircled{S}}} U_3 : s_1 \qquad \forall \Gamma \leq \Delta,\ \Gamma \vdash u \mathrel{\widehat{\textcircled{S}}} u' \star U_2 \implies \Gamma \vdash T_2[u/x] \mathrel{\widehat{\textcircled{S}}} T_3[u'/x] : s_2 \qquad \Delta \vdash (x \star U_2) \overset{s_1,s_2}{\to} T_2 :=: (x \star U_3) \overset{s_1,s_2}{\to} T_3 : s_3}{\Delta \vdash (x \star U_2) \overset{s_1,s_2}{\to} T_2 \mathbin{\textcircled{S}} (x \star U_3) \overset{s_1,s_2}{\to} T_3 : s_3}$$

By transitivity we have $\Delta \vdash (x \star U_1) \to T_1 :=: (x \star U_3) \to T_3 : s_3$ and $\Delta \vdash U_1 \mathrel{\widehat{\textcircled{S}}} U_3 : s_1$ by induction hypothesis on s_1.
Note that this is where the arrow sort annotations are useful. Without them we would not know that the sorts in both derivations are equal. We could have $\Delta \vdash U_1 \mathrel{\widehat{\textcircled{S}}} U_2 : s_1$ and $\Delta \vdash U_2 \mathrel{\widehat{\textcircled{S}}} U_3 : s_1'$ for apparently unrelated s_1 and s_1', and would therefore be unable to use transitivity.
Given $\Gamma \leq \Delta$ and $\Gamma \vdash u \mathrel{\widehat{\textcircled{S}}} u' \star U_1$, we need to show that $\Gamma \vdash T_1[u/x] \mathrel{\widehat{\textcircled{S}}} T_3[u'/x] : s_3$. As $\mathrel{\widehat{\textcircled{S}}}$ at type U is a PER by induction hypothesis, we have $\Gamma \vdash u \mathrel{\widehat{\textcircled{S}}} u \star U_1$, from which we can deduce $\Gamma \vdash T_1[u/x] \mathrel{\widehat{\textcircled{S}}} T_2[u/x] : s_2$. By conversion using $\Delta \vdash U_1 \mathrel{\widehat{\textcircled{S}}} U_2 : s_1$ – weakened at Γ – we have $\Gamma \vdash u \mathrel{\widehat{\textcircled{S}}} u' \star U_2$, which implies $\Gamma \vdash T_2[u/x] \mathrel{\widehat{\textcircled{S}}} T_3[u'/x] : s_2$. This allows us to conclude by transitivity at type s_2.

□

In the following we show that the variables are in the logical relation, i. e., $\Delta \vdash x \mathbin{Ⓢ} x : \Delta(x)$ for well-formed contexts Δ. As usual, this statement has to be generalized to neutrals n to be proven inductively.

Lemma 13 (Into the logical relation)**.** Let $T \in s$. If $\Delta \vdash n :=: n' \star T$ then $\Delta \vdash n \mathbin{\widehat{Ⓢ}} n' \star T$.

Proof. By induction on $T \in s$.

Case $N \in s$ and $\Delta \vdash n :=: n' \star N$. Then $\Delta \vdash n \mathbin{Ⓢ} n' \star N$ by cases on $\star$, unfolding definitions.

Case $s \in s'$ and $\Delta \vdash N :=: N' \star s$. Then $\Delta \vdash N \mathbin{Ⓢ} N' \star s$ by cases on $\star$.

Case $(x \star U) \to T \in s_3$ and $\Delta \vdash n :=: n' \star_0 (x \star U) \to T$.

First, the case for $\star_0 = :$. We have $\Delta \vdash n :=: n' : (x \star U) \to T$. Assume arbitrary $\Gamma \leq \Delta$ and $\Gamma \vdash u \mathbin{\widehat{Ⓢ}} u' \star U$, which yields $\Gamma \vdash u :=: u' \star U$ and $\Gamma \vdash T[u/x] \mathbin{\widehat{Ⓢ}} T[u/x] : s_2$. By weakening, $\Gamma \vdash n \star u :=: n' \star u' : T[u/x]$, thus, by induction hypothesis, $\Gamma \vdash n \star u \mathbin{Ⓢ} n' \star u' : T[u/x]$, q.e.d.

The case for $\star_0 = \div$ proceeds analogously.

□

4.3. **Validity in the Model.** We now extend our logical relation $\mathbin{\widehat{Ⓢ}}$ to substitutions, by induction on the destination context.

$$\frac{}{\Delta \vdash \sigma \mathbin{\widehat{Ⓢ}} \sigma' : \diamond} \qquad \frac{\Delta \vdash \sigma \mathbin{\widehat{Ⓢ}} \sigma' : \Gamma \qquad \Delta \vdash \sigma(x) \mathbin{\widehat{Ⓢ}} \sigma'(x) \star U\sigma}{\Delta \vdash \sigma \mathbin{\widehat{Ⓢ}} \sigma' : \Gamma.\, x \star U}$$

This relation inherits weakening from $\mathbin{\widehat{Ⓢ}}$ for terms.

We then define the context ($\Vdash \Gamma$), type ($\Gamma \Vdash T = T'$) and term ($\Gamma \Vdash t = t' : T$) validity relations, by induction on the length of contexts.

$$\frac{}{\Vdash \diamond} \qquad \frac{\Vdash \Gamma \quad \Gamma \Vdash U}{\Vdash \Gamma.\, x \star U} \qquad \frac{\Gamma \Vdash T = T' : s}{\Gamma \Vdash T = T'} \qquad \frac{\Gamma \Vdash T = T}{\Gamma \Vdash T}$$

$$\frac{\Vdash \Gamma \qquad (\Gamma \Vdash T \text{ unless } T = s) \qquad \forall \Delta, \sigma, \sigma',\ \Delta \vdash \sigma \mathbin{\widehat{Ⓢ}} \sigma' : \Gamma \implies \Delta \vdash t\sigma \mathbin{\widehat{Ⓢ}} t'\sigma' : T\sigma}{\Gamma \Vdash t = t' : T} \qquad \frac{\Gamma \Vdash t = t : T}{\Gamma \Vdash t : T}$$

Because of its asymmetric definition, the logical relation on substitutions may not be a PER in general, but it is for valid contexts.

Lemma 14 (Substitution relation is a PER)**.** If $\Vdash \Gamma$, then $\Delta \vdash \_ \mathbin{\widehat{Ⓢ}} \_ : \Gamma$ is symmetric and transitive.

Proof. By induction on Γ. We demonstrate symmetry for the case $\Vdash \Gamma.\, x \star U$.

$$\frac{\Delta \vdash \sigma \mathbin{\widehat{Ⓢ}} \sigma' : \Gamma \qquad \Delta \vdash \sigma(x) \mathbin{\widehat{Ⓢ}} \sigma'(x) \star U\sigma}{\Delta \vdash \sigma \mathbin{\widehat{Ⓢ}} \sigma' : \Gamma.\, x \star U}$$

By induction hypothesis, $\Delta \vdash \sigma' \mathbin{\widehat{Ⓢ}} \sigma : \Gamma$, and by symmetry of $\mathbin{\widehat{Ⓢ}}$ for terms (Lemma 12), $\Delta \vdash \sigma'(x) \mathbin{\widehat{Ⓢ}} \sigma(x) \star U\sigma$. We instantiate $\Gamma \Vdash U$ to $\Delta \vdash U\sigma \mathbin{\widehat{Ⓢ}} U\sigma' : s$ and conclude $\Delta \vdash \sigma'(x) \mathbin{\widehat{Ⓢ}} \sigma(x) \star U\sigma'$ by conversion (Lemma 11). □

Lemma 15 (Validity is a PER)**.** The relation $\Gamma \Vdash \_ = \_ : T$ is symmetric and transitive.

Proof. Symmetry requires symmetry of $\widehat{\mathsf{S}}$ for substitutions and conversion with $\Delta \vdash T\sigma \mathrel{\widehat{\mathsf{S}}} T\sigma' : s'$, similar as in Lemma 14.

We demonstrate transitivity in detail. Given $\Gamma \Vdash t_1 = t_2 : T$ and $\Gamma \Vdash t_2 = t_3 : T$ we show $\Gamma \Vdash t_1 = t_3 : T$. Clearly, $\Vdash \Gamma$ and $\Gamma \Vdash T$ or $T = s$ by one of our two assumptions. Assume arbitrary $\Delta \vdash \sigma \mathrel{\widehat{\mathsf{S}}} \sigma' : \Gamma$ and show $\Delta \vdash t_1\sigma \mathrel{\widehat{\mathsf{S}}} t_3\sigma' : T\sigma$. By Lemma 14, $\Delta \vdash \sigma \mathrel{\widehat{\mathsf{S}}} \sigma : \Gamma$, thus $\Delta \vdash t_1\sigma \mathrel{\widehat{\mathsf{S}}} t_2\sigma : T\sigma$. Also, $\Delta \vdash t_2\sigma \mathrel{\widehat{\mathsf{S}}} t_3\sigma' : T\sigma$ which entails our goal by transitivity of $\widehat{\mathsf{S}}$ (Lemma 12). □

Lemma 16 (Function type injectivity is valid)**.** If $\Gamma \Vdash (x{\star}U) \overset{s_1,s_2}{\to} T = (x{\star}U') \overset{s_1',s_2'}{\to} T'$ then $s_1 = s_1'$ and $s_2 = s_2'$ and $\Gamma \Vdash U = U' : s_1$ and $\Gamma.\, x{\star}U' \Vdash T = T' : s_2$.

Proof. Assume arbitrary $\Delta \vdash \sigma \mathrel{\widehat{\mathsf{S}}} \sigma' : \Gamma$. We have $\Delta \vdash (x{\star}U\sigma) \overset{s_1,s_2}{\to} T\sigma \mathrel{\widehat{\mathsf{S}}} (x{\star}U'\sigma') \overset{s_1',s_2'}{\to} T'\sigma' : s_3$, thus by definition $s_1 = s_1'$ and $s_2 = s_2'$ and $\Delta \vdash U'\sigma' \mathrel{\widehat{\mathsf{S}}} U\sigma : s_1$—note that sorts are closed and therefore invariant by substitution. By symmetry of $\widehat{\mathsf{S}}$, and since Δ, σ, σ' were arbitrary, we have $\Gamma \Vdash U = U' : s_1$.

Further, assume arbitrary $\Delta \vdash u \mathrel{\widehat{\mathsf{S}}} u' \star U'\sigma$ and let $\rho = (\sigma, u/x)$ and $\rho' = (\sigma', u'/x)$. Note that w.l.o.g., $x \notin \mathsf{dom}(\Gamma)$ and $x \notin \mathsf{FV}(U')$ and $\Delta \vdash \rho \mathrel{\widehat{\mathsf{S}}} \rho' : \Gamma.\, x{\star}U'$. We have $\Delta \vdash T\rho \mathrel{\widehat{\mathsf{S}}} T'\rho' : s_2$ and since ρ, ρ' were arbitrary, $\Gamma.\, x{\star}U' \Vdash T = T' : s_2$. □

Lemma 17 (Context satisfiable)**.** If $\Vdash \Gamma$ then $\vdash \Gamma$ and $\Gamma \vdash \mathsf{id} \mathrel{\widehat{\mathsf{S}}} \mathsf{id} : \Gamma$.

Proof. By induction on Γ. The $\diamond$ case is immediate. In the $\Gamma.\, x{\star}U$ case, given

$$\frac{\Vdash \Gamma \qquad \Gamma \Vdash U}{\Vdash \Gamma.\, x{\star}U}$$

we can use inference

$$\frac{\Gamma.\, x{\star}U \vdash \mathsf{id} \mathrel{\widehat{\mathsf{S}}} \mathsf{id} : \Gamma \qquad \Gamma.\, x{\star}U \vdash \mathsf{id}(x) \mathrel{\widehat{\mathsf{S}}} \mathsf{id}(x) \star U\mathsf{id}}{\Gamma.\, x{\star}U \vdash \mathsf{id} \mathrel{\widehat{\mathsf{S}}} \mathsf{id} : \Gamma.\, x{\star}U}.$$

From the induction hypothesis $\Gamma \vdash \mathsf{id} \mathrel{\widehat{\mathsf{S}}} \mathsf{id} : \Gamma$, we obtain the first premise by weakening of $\widehat{\mathsf{S}}$. It also yields $\Gamma \vdash U\mathsf{id} : s\mathsf{id}$ for some s by definition of $\Gamma \Vdash U$. Using induction hypothesis, $\vdash \Gamma$, this entails $\vdash \Gamma.\, x{\star}U$. Further, $\Gamma.\, x{\star}U \vdash x = x \star U$, and since trivially $\Gamma.\, x{\star}U \vdash x \longleftrightarrow x \star U$, we can derive $\Gamma.\, x{\star}U \vdash x \mathrel{\widehat{\mathsf{S}}} x \star U$, by the Lemma 13. This concludes the second premise $\Gamma.\, x{\star}U \vdash \mathsf{id}(x) \mathrel{\widehat{\mathsf{S}}} \mathsf{id}(x) \star U\mathsf{id}$. □

We can now show that every equation valid in the model is derivable in IITT.

Theorem 18 (Completeness of IITT rules)**.** If $\Gamma \Vdash t = t' : T$ then both $\Gamma \vdash t : T$ and $\Gamma \vdash t' : T$ and $\Gamma \vdash t = t' : T$ and $\Gamma \vdash T$.

Proof. Using Lemma 17 we obtain $\Gamma \vdash t \mathrel{\widehat{\mathsf{S}}} t' : T$, which entails $\Gamma \vdash t, t' : T$ and $\Gamma \vdash t = t' : T$. Analogously, since our assumption entails $\Gamma \Vdash T$ by definition, we get $\Gamma \vdash T$. □

4.4. **Fundamental theorem.** We prove a series of lemmata which constitute parts of the fundamental theorem for the Kripke logical relation.

Lemma 19 (Resurrection)**.** If $\Vdash \Gamma$ and $\Delta \vdash \sigma \mathrel{\widehat{\mathsf{S}}} \sigma' : \Gamma$ then $\Delta^{\div} \vdash \sigma \mathrel{\widehat{\mathsf{S}}} \sigma : \Gamma^{\div}$ and $\Delta^{\div} \vdash \sigma' \mathrel{\widehat{\mathsf{S}}} \sigma' : \Gamma^{\div}$.

Proof. By induction on Γ, the interesting case being

$$\frac{\Delta \vdash \sigma \mathbin{\widehat{\mathbb{S}}} \sigma' : \Gamma \qquad \Delta \vdash \sigma(x) \mathbin{\widehat{\mathbb{S}}} \sigma'(x) \star U\sigma}{\Delta \vdash \sigma \mathbin{\widehat{\mathbb{S}}} \sigma' : \Gamma.\, x \star U}.$$

First, we show $\Delta^{\div} \vdash \sigma \mathbin{\widehat{\mathbb{S}}} \sigma : (\Gamma^{\div}.\, x : U)$. By induction hypothesis $\Delta^{\div} \vdash \sigma \mathbin{\widehat{\mathbb{S}}} \sigma : \Gamma^{\div}$, and by definition, $\Delta^{\div} \vdash \sigma(x) \mathbin{\widehat{\mathbb{S}}} \sigma(x) : U\sigma$. This immediately entails our goal.

For the second goal $\Delta^{\div} \vdash \sigma' \mathbin{\widehat{\mathbb{S}}} \sigma' : (\Gamma^{\div}.\, x : U)$, observe that $\Gamma \Vdash U$, hence $\Delta \vdash U\sigma \mathbin{\widehat{\mathbb{S}}} U\sigma' : s$ for some sort s. Thus, we can cast our hypothesis $\Delta \vdash \sigma'(x) \mathbin{\widehat{\mathbb{S}}} \sigma'(x) : U\sigma$ to $U\sigma'$ and conclude analogously. □

Corollary 20. If $\Gamma^{\star} \Vdash u : U$ and $\Delta \vdash \sigma \mathbin{\widehat{\mathbb{S}}} \sigma' : \Gamma$ then $\Delta \vdash u\sigma \mathbin{\widehat{\mathbb{S}}} u\sigma' \star U\sigma$.

Proof. In case $\star = :$ it holds by definition, but we need resurrection for $\star = \div$. If $\Delta \vdash \sigma \mathbin{\widehat{\mathbb{S}}} \sigma' : \Gamma$, then by resurrection (Lemma 19) we have $\Delta^{\div} \vdash \sigma \mathbin{\widehat{\mathbb{S}}} \sigma : \Gamma^{\div}$, so from $\Gamma^{\div} \Vdash u : U$ we deduce $\Delta^{\div} \vdash u\sigma \mathbin{\widehat{\mathbb{S}}} u\sigma : U\sigma$. Analogously we get $\Delta^{\div} \vdash u\sigma' \mathbin{\widehat{\mathbb{S}}} u\sigma' : U\sigma'$ which we cast to $\Delta^{\div} \vdash u\sigma' \mathbin{\widehat{\mathbb{S}}} u\sigma' : U\sigma$. □

Lemma 21 (Validity of β-reduction)**.**

$$\frac{\Gamma.\, x \star U \Vdash t : T \qquad \Gamma^{\star} \Vdash u : U}{\Gamma \Vdash (\lambda x \star U.\, t) \star u = t[u/x] : T[u/x]}$$

Proof. $\Vdash \Gamma$ is contained in the first hypothesis $\Gamma \Vdash u \star U$. Then, given $\Delta \vdash \rho \mathbin{\widehat{\mathbb{S}}} \rho' : \Gamma$ we need to show $\Delta \vdash (\lambda x \star U.\, t)\rho \star u\rho \mathbin{\widehat{\mathbb{S}}} t(\rho', u\rho'/x) : T(\rho, u\rho/x)$ and also $\Delta \vdash T[u/x]\rho \mathbin{\widehat{\mathbb{S}}} T[u/x]\rho' : s$ for some s (the latter to get $\Gamma \Vdash T[u/x]$).

Let $\sigma = (\rho, u\rho/x)$ and $\sigma' = (\rho', u\rho'/x)$. From the second hypothesis and Cor. 20 we get $\Delta \vdash u\rho \mathbin{\widehat{\mathbb{S}}} u\rho' \star U\rho$, which gives $\Delta \vdash \sigma \mathbin{\widehat{\mathbb{S}}} \sigma' : \Gamma.\, x \star U$. By instantiating the first hypothesis we get $\Delta \vdash t\sigma \mathbin{\widehat{\mathbb{S}}} t\sigma' : T\sigma$, and also (from the premise $\Gamma.\, x \star U \Vdash T$) $\Delta \vdash T\sigma = T\sigma'$, which gives $\Gamma \Vdash T[u/x]$.

Finally, from $\Delta \vdash t\sigma \mathbin{\widehat{\mathbb{S}}} t\sigma'$ we get the desired $\Delta \vdash (\lambda x \star U.\, t)\rho \star u\rho \mathbin{\widehat{\mathbb{S}}} t\sigma' : T\sigma$, as $\mathbin{\widehat{\mathbb{S}}}$ is closed by weak head expansion to well-typed $\Delta \vdash (\lambda x \star U.\, t)\rho \star u\rho : T\sigma$. □

Lemma 22 (Validity of η)**.**

$$\frac{\Gamma \Vdash t : (x \star U) \to T}{\Gamma \Vdash t = \lambda x \star U.\, t \star x : (x \star U) \to T}$$

Proof. $\Vdash \Gamma$ and $\Gamma \Vdash (x \star U) \to T$ are direct consequences of our hypothesis. Given $\Delta \vdash \rho \mathbin{\widehat{\mathbb{S}}} \rho' : \Gamma$, we need to show $\Delta \vdash t\rho \mathbin{\widehat{\mathbb{S}}} (\lambda x \star U.\, t \star x)\rho' : ((x \star U) \to T)\rho$. W. l. o. g., x is not free in the domain nor range of substitutions ρ and ρ', thus with $t' := t\rho$, $t'' := t\rho'$, $U' := U\rho$, $U'' := U\rho'$, $T' := T\rho$ and $T'' := T\rho'$ it is sufficient to show $\Delta \vdash t' \mathbin{\widehat{\mathbb{S}}} \lambda x \star U''.\, t'' \star x : (x \star U') \to T'$.

First, given (Δ', u, u') such that $\Delta' \leq \Delta$ and $\Delta' \vdash u \mathbin{\widehat{\mathbb{S}}} u' \star U'$, we show $\Delta' \vdash t' \star u \mathbin{\widehat{\mathbb{S}}} (\lambda x \star U''.\, t'' \star x) \star u' : T'[u/x]$. Our hypothesis $\Gamma \Vdash t : (x \star U) \to T$ entails $\Delta \vdash t\rho \mathbin{\widehat{\mathbb{S}}} t\rho' : ((x \star U) \to T)\rho$, that is to say $\Delta \vdash t' \mathbin{\widehat{\mathbb{S}}} t'' : (x \star U') \to T'$. This logical relation at a function type, when instantiated to (Δ', u, u'), gives us $\Delta' \vdash t' \star u \mathbin{\widehat{\mathbb{S}}} t'' \star u' : T'[u/x]$, which weak-head expands to the desired goal.

Second, we show $\Delta \vdash t' :=: \lambda x \star U''.\, t'' \star x : (x \star U') \to T'$.

- $\Gamma \vdash t' : (x \star U') \to T'$ is a simple consequence of our hypothesis $\Gamma \Vdash t : (x \star U) \to T$.

- $\Gamma \vdash \lambda x{\star}U''.\, t''\,{}^{\star}x : (x{\star}U') \to T'$ has the following proof:

$$\dfrac{\dfrac{\dfrac{\Gamma \Vdash t : (x{\star}U) \to T}{\Delta \vdash t'' : (x{\star}U'') \to T''}}{\Delta.\, x{\star}U'' \vdash t'' : (x{\star}U'') \to T''}\,\text{weak} \qquad \dfrac{}{(\Delta.\, x{\star}U'')^{\star} \vdash x : U''}\,\text{var}}{\dfrac{\Delta.\, x{\star}U'' \vdash t''\,{}^{\star}x : T''}{\dfrac{\Delta \vdash \lambda x{\star}U''.\, t''\,{}^{\star}x : (x{\star}U'') \to T''}{\Delta \vdash \lambda x{\star}U''.\, t''\,{}^{\star}x : (x{\star}U') \to T'}\,\text{conv}}} \qquad \dfrac{\Gamma \Vdash (x{\star}U) \to T}{\Delta \vdash (x{\star}U'') \to T''}$$

- $\Delta \vdash t' = \lambda x{\star}U''.\, t''\,{}^{\star}x : (x{\star}U') \to T'$. The η-rule of definitional equality gives us $\Delta \vdash t'' = \lambda x{\star}U''.\, t''\,{}^{\star}x : (x{\star}U'') \to T''$. From $\Gamma \Vdash (x{\star}U) \to T$ we can convert it to the type $(x{\star}U') \to T'$, and then conclude by transitivity using $\Delta \vdash t' = t'' : (x{\star}U') \to T'$, which is a direct consequence of $\Gamma \Vdash t : (x{\star}U) \to T$.

□

Lemma 23 (Validity of function equality)**.**

$$\dfrac{\Gamma \Vdash U = U' \qquad \Gamma.\, x{\star}U \Vdash t = t' : T}{\Gamma \Vdash (\lambda x{\star}U.\, t) = (\lambda x{\star}U'.\, t') : (x{\star}U) \to T}$$

Proof. Again $\Vdash \Gamma$ and $\Gamma \Vdash (x{\star}U) \to T$ are simple consequences of our hypotheses. Given $\Delta \vdash \rho \mathrel{\widehat{\mathbb{S}}} \rho' : \Gamma$ (w. l. o. g., x is not free in ρ, ρ' domain or range), we need to show $\Delta \vdash (\lambda x{\star}U\rho.\, t\rho) \mathrel{\widehat{\mathbb{S}}} (\lambda x{\star}U'\rho'.\, t'\rho') : ((x{\star}U\rho) \to T\rho)$. We will skip the proof of $\Delta \vdash (\lambda x{\star}U\rho.\, t\rho) :=: (\lambda x{\star}U'\rho'.\, t'\rho') : ((x{\star}U\rho) \to T\rho)$, as it is similar to the corresponding part of the η-validity lemma.

Given (Δ', u, u') such that $\Delta' \leq \Delta$ and $\Delta' \vdash u \mathrel{\widehat{\mathbb{S}}} u' \star U\rho$, we have to show that $\Delta' \vdash (\lambda x{\star}U\rho.\, t\rho)\,{}^{\star}u \mathrel{\widehat{\mathbb{S}}} (\lambda x{\star}U'\rho'.\, t'\rho')\,{}^{\star}u' : T\rho[u/x]$. Let $\sigma = (\rho, u/x)$ and $\sigma' = (\rho', u'/x)$. As we supposed $\Delta' \vdash u \mathrel{\widehat{\mathbb{S}}} u' \star U\rho$, we have $\Delta' \vdash \sigma \mathrel{\widehat{\mathbb{S}}} \sigma' : \Gamma.\, x{\star}U\rho$. Instantiating the second hypothesis with Δ', σ, σ' therefore gives us $\Delta' \vdash t\sigma = t'\sigma' : T\sigma$, which can also be written $\Delta' \vdash t\rho[u/x] \mathrel{\widehat{\mathbb{S}}} t'\rho'[u'/x] : T\rho[u/x]$, which is weak-head expansible to our goal. □

Lemma 24 (Validity of irrelevant application)**.**

$$\dfrac{\Gamma \Vdash t = t' : (x \div U) \to T \qquad \Gamma^{\div} \Vdash u : U \qquad \Gamma^{\div} \Vdash u' : U}{\Gamma \Vdash t \div u = t' \div u' : T[u/x]}$$

Proof. Assume arbitrary $\Delta \vdash \rho \mathrel{\widehat{\mathbb{S}}} \rho' : \Gamma$ and show $\Delta \vdash t\rho \div u\rho \mathrel{\widehat{\mathbb{S}}} t'\rho' \div u'\rho' : T(\rho, u\rho/x)$. By the first hypothesis, it is sufficient to show $\Delta \vdash u\rho \mathrel{\widehat{\mathbb{S}}} u'\rho' \div U\rho$, which means $\Delta^{\div} \vdash u\rho \mathrel{\widehat{\mathbb{S}}} u\rho : U\rho$ and $\Delta^{\div} \vdash u'\rho' \mathrel{\widehat{\mathbb{S}}} u'\rho' : U\rho$. By Resurrection (Lemma 19), $\Delta^{\div} \vdash \rho \mathrel{\widehat{\mathbb{S}}} \rho : \Gamma^{\div}$, hence $\Delta^{\div} \vdash u\rho \mathrel{\widehat{\mathbb{S}}} u\rho : U\rho$ from the second hypothesis. Analogously, we obtain $\Delta^{\div} \vdash u'\rho' \mathrel{\widehat{\mathbb{S}}} u'\rho' : U\rho'$ from the third hypothesis which we can cast to $U\rho$ by virtue of $\Gamma \Vdash U$ which we get from $\Gamma \Vdash (x \div U) \to T$ by Lemma 16. □

Theorem 25 (Fundamental theorem of logical relations)**.**

(1) If $\vdash \Gamma$ then $\Vdash \Gamma$.
(2) If $\Gamma \vdash t : T$ then $\Gamma \Vdash t : T$.
(3) If $\Gamma \vdash t = t' : T$ then $\Gamma \Vdash t = t' : T$.

Proof. By induction on the derivation. □

As a simple corollary we obtain syntactic validity, namely that definitional equality implies well-typedness and well-typedness implies well-formedness of the involved type. This lemma could have been proven purely syntactically, but the syntactic proof requires a sequence of carefully arranged lemmata like context conversion, substitution, functionality, and inversion on types [HP05, AC07]. Our "sledgehammer" *semantic* argument is built into the Kripke logical relation, in the spirit of Goguen [Gog00].

Corollary 26 (Syntactic validity)**.**

(1) If $\Gamma \vdash t : T$ then $\Gamma \vdash T$.
(2) If $\Gamma \vdash t = t' : T$ then $\Gamma \vdash t : T$ and $\Gamma \vdash t' : T$.

Proof. By the fundamental theorem, $\Gamma \vdash t = t' : T$ implies $\Gamma \Vdash t = t' : T$, which by Thm. 18 implies $\Gamma \vdash t, t' : T$ and $\Gamma \vdash T$. □

5. Meta-theoretic Consequences of the Model Construction

In this section, we explicate the results established by the Kripke model.

5.1. **Admissibility of Substitution.** Goguen [Gog00] observes that admissibility of substitution for the syntactic judgements can be inherited from the Kripke logical relation, which is closed under substitution by its very definition.

To show that the judgements of IITT are closed under substitution we introduce relations $\Gamma \vdash \sigma : \Gamma'$ for substitution typing and $\Gamma \vdash \sigma = \sigma' : \Gamma'$ for substitution equality which are given inductively by the following rules:

$$\frac{\vdash \Gamma}{\Gamma \vdash \sigma : \diamond} \qquad \frac{\Gamma \vdash \sigma : \Gamma' \qquad \Gamma' \vdash U \qquad \Gamma \vdash \sigma(x) \star U\sigma}{\Gamma \vdash \sigma : \Gamma'.\, x \star U}$$

$$\frac{\vdash \Gamma}{\Gamma \vdash \sigma = \sigma' : \diamond} \qquad \frac{\Gamma \vdash \sigma = \sigma' : \Gamma' \qquad \Gamma' \vdash U \qquad \Gamma \vdash \sigma(x) = \sigma'(x) \star U\sigma}{\Gamma \vdash \sigma = \sigma' : \Gamma'.\, x \star U}$$

Substitution typing and equality are closed under weakening.

Semantically, substitutions are explained by environments. We define substitution validity as follows, again in rule form but not inductively:

$$\frac{\Gamma \Vdash \sigma = \sigma : \Gamma'}{\Gamma \Vdash \sigma : \Gamma'} \qquad \frac{\Vdash \Gamma \qquad \Vdash \Gamma' \qquad \forall \Delta \Vdash \rho \mathbin{\widehat{\textcircled{S}}} \rho' : \Gamma.\ \ \Delta \Vdash \sigma\rho \mathbin{\widehat{\textcircled{S}}} \sigma'\rho' : \Gamma'}{\Gamma \Vdash \sigma = \sigma' : \Gamma'}$$

Lemma 27 (Fundamental lemma for substitutions)**.**

(1) If $\Gamma \vdash \sigma : \Gamma'$ then $\Gamma \Vdash \sigma : \Gamma'$.
(2) If $\Gamma \vdash \sigma = \sigma' : \Gamma'$ then $\Gamma \Vdash \sigma = \sigma' : \Gamma'$.

Proof. We demonstrate 2 by induction on $\Gamma \vdash \sigma = \sigma' : \Gamma'$.

Case

$$\frac{\vdash \Gamma}{\Gamma \vdash \sigma = \sigma' : \diamond}$$

We have $\Vdash \Gamma$ by Thm. 25 and $\Vdash \diamond$ trivially. Also, $\Delta \vdash \sigma\rho \mathbin{\widehat{\textcircled{S}}} \sigma'\rho' : \diamond$ trivially for any $\Delta \vdash \rho \mathbin{\widehat{\textcircled{S}}} \rho' : \Gamma$.

Case

$$\frac{\Gamma \vdash \sigma = \sigma' : \Gamma' \qquad \Gamma' \vdash U \qquad \Gamma \vdash \sigma(x) = \sigma'(x) \star U\sigma}{\Gamma \vdash \sigma = \sigma' : \Gamma'.\, x{\star}U}$$

We have $\Vdash \Gamma$ and $\Vdash \Gamma'$ by induction hypothesis and $\Gamma' \Vdash U$ by Thm. 25, thus, $\Vdash \Gamma'.\, x{\star}U$. Now assume arbitrary $\Delta \vdash \rho \mathbin{\widehat{\textcircled{S}}} \rho' : \Gamma$ and show $\Delta \vdash \sigma\rho \mathbin{\widehat{\textcircled{S}}} \sigma'\rho' : \Gamma'.\, x{\star}U$. First, $\Delta \vdash \sigma\rho \mathbin{\widehat{\textcircled{S}}} \sigma'\rho' : \Gamma'$ follows by induction hypothesis. The second subgoal $\Delta \vdash (\sigma\rho)(x) \mathbin{\widehat{\textcircled{S}}} (\sigma'\rho')(x) \star U\sigma\rho$ is just an instance of the second induction hypothesis.

□

Theorem 28 (Substitution and functionality)**.**

(1) If $\Gamma \vdash \sigma : \Gamma'$ and $\Gamma' \vdash t : T$ then $\Gamma \vdash t\sigma : T\sigma$.
(2) If $\Gamma \vdash \sigma : \Gamma'.$ and $\Gamma' \vdash t = t' : T$ then $\Gamma \vdash t\sigma = t'\sigma : T\sigma$.
(3) If $\Gamma \vdash \sigma = \sigma' : \Gamma'.$ and $\Gamma' \vdash t : T$ then $\Gamma \vdash t\sigma = t\sigma' : T\sigma$.
(4) If $\Gamma \vdash \sigma = \sigma' : \Gamma'.$ and $\Gamma' \vdash t = t' : T$ then $\Gamma \vdash t\sigma = t'\sigma' : T\sigma$.

Proof. We demonstrate 4, the other cases are just variations of the theme. First, from $\Gamma \vdash \sigma = \sigma' : \Gamma'$ we get $\Gamma \vdash \sigma \mathbin{\widehat{\textcircled{S}}} \sigma' : \Gamma'$ by the fundamental lemma for substitutions (Lemma 27), using the identity environment $\Gamma \vdash \mathsf{id} \mathbin{\widehat{\textcircled{S}}} \mathsf{id} : \Gamma$. Now, by the fundamental theorem on $\Gamma \vdash t = t' : T$ we obtain $\Gamma \vdash t\sigma \mathbin{\widehat{\textcircled{S}}} t'\sigma' : T\sigma$, which entails our goal $\Gamma \vdash t\sigma = t'\sigma' : T\sigma$ by Thm. 18. □

5.2. **Context conversion.** Context equality $\vdash \Gamma = \Gamma'$ is defined inductively by the rules

$$\frac{}{\vdash \diamond = \diamond} \qquad \frac{\vdash \Gamma = \Gamma' \qquad \Gamma \vdash U = U'}{\vdash \Gamma.\, x{\star}U = \Gamma'.\, x{\star}U'}.$$

All declarative judgements are closed under context conversion. This fact is easy to prove by induction over derivations, but we get it as just a special case of substitution.

Lemma 29 (Identity substitution)**.** If $\vdash \Gamma = \Gamma'$ then $\Gamma \vdash \mathsf{id} = \mathsf{id} : \Gamma'$.

Proof. By induction on $\vdash \Gamma = \Gamma'$.

Case

$$\frac{\vdash \Gamma = \Gamma' \qquad \Gamma \vdash U = U'}{\vdash \Gamma.\, x{\star}U = \Gamma'.\, x{\star}U'}$$

By induction hypothesis and weakening, $\Gamma.\, x{\star}U \vdash \mathsf{id} = \mathsf{id} : \Gamma'$. Also, $\Gamma.\, x{\star}U \vdash x = x \star U$ and by conversion $\Gamma.\, x{\star}U \vdash x = x \star U'$. Together, $\Gamma.\, x{\star}U \vdash \mathsf{id} = \mathsf{id} : \Gamma'.\, x{\star}U'$.

□

Theorem 30 (Context conversion)**.** Let $\vdash \Gamma' = \Gamma$.

(1) If $\Gamma \vdash t : T$ then $\Gamma' \vdash t : T$.
(2) If $\Gamma \vdash t = t' : T$ then $\Gamma' \vdash t = t' : T$.

Proof. By Thm. 28 with $\Gamma' \vdash \mathsf{id} = \mathsf{id} : \Gamma$. □

As a consequence, context equality is symmetric and transitive (we can trade $\Gamma \vdash U = U'$ for $\Gamma' \vdash U = U'$). Thus, context conversion can be applied in the other direction as well.

5.3. **Inversion, injectivity, and type unicity.** A condition for the decidability of type checking is the ability to invert typing derivations. The proof requires substitution.

Lemma 31 (Inversion)**.**

(1) If $\Gamma \vdash x : T$ then $(x{:}U) \in \Gamma$ for some U with $\Gamma \vdash U = T$.
(2) If $\Gamma \vdash \lambda x{\star}U.\, t : T$ then $\Gamma.\, x{\star}U \vdash t : T'$ for some T' with $\Gamma \vdash (x{\star}U) \to T' = T$.
(3) If $\Gamma \vdash t\,{}^{\star}u : T$ then $\Gamma \vdash t : (x{\star}U) \to T'$ and $\Gamma \vdash u \star U$ for some U, T' with $\Gamma \vdash T'[u/x] = T$.
(4) If $\Gamma \vdash s : T$ then there is $(s, s') \in \mathsf{Axiom}$ such that $\Gamma \vdash s' = T$.
(5) If $\Gamma \vdash (x{\star}U) \overset{s_1,s_2}{\to} T' : T$ then $\Gamma \vdash U : s_1$ and $\Gamma.\, x{\star}U \vdash T' : s_2$, and for some s_3 we have $\Gamma \vdash s_3 = T$ and $(s_1, s_2, s_3) \in \mathsf{Rule}$.

Proof. Each by induction on the typing derivation. □

Remark 32. The need for inversion during type checking is the only good reason to have separate typing rules and not simply define typing $\Gamma \vdash t : T$ as the diagonal $\Gamma \vdash t = t : T$ of equality. While by a logical relation argument we will obtain a suitable inversion result for $\Gamma \vdash (x{\star}U) \to T = (x{\star}U) \to T$—the famous *function type injectivity* (Theorem 33)— it seems hard to get something similar for application $t\,u$.

Injectivity for function types w. r. t. typed equality is known to be tricky. It is connected to subject reduction and required for many meta-theoretic results. We harvest it from our Kripke model.

Theorem 33 (Function type injectivity)**.** If $\Gamma \vdash (x{\star}U) \overset{s_1,s_2}{\to} T = (x{\star}U') \overset{s_1',s_2'}{\to} T' : s_3$ then $s_1 = s_1'$ and $s_2 = s_2'$ and $\Gamma \vdash U = U' : s_1$ and $\Gamma.\, x{\star}U \vdash T = T' : s_2$.

Proof. This follows from Lemma 16. Or we can prove it directly as follows: Since $\Gamma \vdash \mathsf{id}\ \widehat{\mathbb{S}}\ \mathsf{id} : \Gamma$ we have by the fundamental theorem $\Gamma \vdash (x{\star}U) \overset{s_1,s_2}{\to} T\ \widehat{\mathbb{S}}\ (x{\star}U') \overset{s_1,s_2}{\to} T' : s_3$ which by inversion yields first $s_1 = s_1'$ and $s_2 = s_2'$ and $\Gamma \vdash U\ \widehat{\mathbb{S}}\ U' : s_1$ and $\Gamma \vdash U = U' : s_1$. Since $\Gamma.\, x{\star}U \vdash x\ \widehat{\widehat{\mathbb{S}}}\ x{\star}U$, we also obtain $\Gamma.\, x{\star}U \vdash T\ \widehat{\mathbb{S}}\ T' : s_2$ and conclude $\Gamma.\, x{\star}U \vdash T = T' : s_2$. □

From the inversion lemma we can prove uniqueness of types, since we are dealing with a functional PTS, and we have function type injectivity.

Theorem 34 (Type unicity)**.** If $\Gamma \vdash t : T$ and $\Gamma \vdash t : T'$ then $\Gamma \vdash T = T'$.

Proof. By induction on t, using inversion. □

5.4. **Normalization and Subject Reduction.** An immediate consequence of the model construction is that each term has a weak head normal form and that typing and equality is preserved by weak head normalization.

Theorem 35 (Normalization and subject reduction)**.** If $\Gamma \vdash t : T$ then $t \searrow a$ and $\Gamma \vdash t = a : T$.

Proof. By the fundamental theorem, $\Gamma \vdash t\ \widehat{\mathbb{S}}\ t : T$ which by definition contains a derivation of $\Gamma \vdash t = {\downarrow}t : T$. □

5.5. **Consistency.** Importantly, not every type is inhabited in IITT, thus, it can be used as a logic. A prerequisite is that types can be distinguished, which follows immediately from the construction of the logical relation.

Lemma 36 (Type constructor discrimination)**.** Neutral types, sorts and function types are mutually unequal.

(1) $\Gamma \vdash N \neq s$.
(2) $\Gamma \vdash N \neq (x \star U) \to T$.
(3) $\Gamma \vdash s = s'$ implies $s \equiv s'$.
(4) $\Gamma \vdash s \neq (x \star U) \to T$.

Proof. By the fundamental theorem applied to the identity substitution. For instance, assuming $\Gamma \vdash N = s : s'$ we get $\Gamma \vdash N \,Ⓢ\, s : s'$ but this is a contradiction to the definition of Ⓢ. □

From normalization and type constructor discrimination we can show that not every type is inhabited.

Theorem 37 (Consistency)**.** $X : \mathsf{Set}_0 \not\vdash t : X$.

Proof. Let $\Gamma = (X : \mathsf{Set}_0)$. Assuming $\Gamma \vdash t : X$, we have $\Gamma \vdash a : X$ for the whnf a of t. We invert on the typing of a. By Lemma 36, X cannot be equal to a function type or sort, thus, a can neither be a λ nor a function type nor a sort, it can only be neutral. The only variable X must be in the head of a, but since X is not of function type, it cannot be applied. Thus, $a \equiv X$ and $\Gamma \vdash X : X$, implying $\Gamma \vdash X = \mathsf{Set}_0$ by inversion (Lemma 31). This is in contradiction to Lemma 36! □

5.6. **Soundness of Algorithmic Equality.** Soundness of the equality algorithm is a consequence of subject reduction.

Theorem 38 (Soundness of algorithmic equality)**.**

(1) Let $\Delta \vdash t, t' : T$. If $\Delta \vdash t \mathrel{\widehat{\Longleftrightarrow}} t' : T$ then $\Delta \vdash t = t' : T$.
(2) Let $\Delta \vdash n, n' : T$. If $\Delta \vdash n \mathrel{\widehat{\longleftrightarrow}} n' : U$ then $\Delta \vdash n = n' : U$ and $\Delta \vdash U = T$.

Proof. Generalize the theorem to all six algorithmic equality judgments and prove it by induction on the algorithmic equality derivation. Since we have subject reduction, the proof proceeds mechanically, because each algorithmic rule corresponds, modulo weak head normalization, to a declarative rule.

Case $\Delta \vdash T : s$ and $\Delta' \vdash T' : s$ and

$$\frac{\Delta \vdash \downarrow T \Longleftrightarrow \downarrow T'}{\Delta \vdash T \mathrel{\widehat{\Longleftrightarrow}} T'}$$

By induction hypothesis, $\Delta \vdash \downarrow T = \downarrow T' : s$. By subject reduction $\Delta \vdash T = \downarrow T : s$ and $\Delta \vdash T' = \downarrow T' : s$. By transitivity $\Delta \vdash T = T' : s$.

Case

$$\frac{\Delta \vdash T \mathrel{\widehat{\Longleftrightarrow}} T'}{\Delta \vdash T \Longleftrightarrow T' : s}$$

By induction hypothesis, $\Delta \vdash T = T' : s$.

□

5.7. **Symmetry and Transitivity of Algorithmic Equality.** Since algorithmic equality is sound for well-typed terms, it is also symmetric and transitive.

Lemma 39 (Type and context conversion in algorithmic equality)**.** Let $\vdash \Delta = \Delta'$.

(1) If $\Delta \vdash A, A'$ and $\Delta \vdash A \Longleftrightarrow A'$ then $\Delta' \vdash A \Longleftrightarrow A'$.
(2) If $\Delta \vdash n, n' : A$ and $\Delta \vdash n \longleftrightarrow n' : A$ then $\Delta' \vdash n \longleftrightarrow n' : A'$ for some A' with $\Delta \vdash A = A'$.
(3) If $\Delta \vdash t, t' : A$ and $\Delta \vdash t \Longleftrightarrow t' : A$ and $\Delta \vdash A = A'$ then $\Delta' \vdash t \Longleftrightarrow t' : A'$.

Proof. By induction on the derivation of algorithmic equality, where we extend the statements to $\overset{\frown}{\longleftrightarrow}$ and $\overset{\frown}{\Longleftrightarrow}$ accordingly.

(1) Type equality.
Case
$$\frac{\Delta \vdash U \overset{\frown}{\Longleftrightarrow} U' \qquad \Delta.\, x{\star}U \vdash T \overset{\frown}{\Longleftrightarrow} T'}{\Delta \vdash (x{\star}U) \to T \Longleftrightarrow (x{\star}U') \to T'}$$
By inversion, $\Delta \vdash U, U'$ and by induction hypothesis, $\Delta' \vdash U \overset{\frown}{\Longleftrightarrow} U'$. Again by inversion, $\Delta.\, x{\star}U \vdash T$ and $\Delta.\, x{\star}U' \vdash T'$, yet by soundness of algorithmic equality, $\Delta \vdash U = U'$, hence $\Delta.\, x{\star}U \vdash T'$ by context conversion. Further, $\vdash \Delta.\, x{\star}U = \Delta'.\, x{\star}U$. Thus, we can apply the other induction hypothesis to obtain $\Delta'.\, x{\star}U \vdash T \overset{\frown}{\Longleftrightarrow} T'$, which finally yields $\Delta' \vdash (x{\star}U) \to T \Longleftrightarrow (x{\star}U') \to T'$.

(2) Structural equality.
Case
$$\frac{(x : T) \in \Delta}{\Delta \vdash x \overset{\frown}{\longleftrightarrow} x : T}$$
Since $\vdash \Delta = \Delta'$, there is a unique $(x : T') \in \Delta'$ with $\Delta \vdash T = T'$. Hence, $\Delta' \vdash x \overset{\frown}{\longleftrightarrow} x : T'$.
Case
$$\frac{\Delta \vdash n \longleftrightarrow n' : (x : U) \to T \qquad \Delta \vdash u \overset{\frown}{\Longleftrightarrow} u' : U}{\Delta \vdash n\, u \overset{\frown}{\longleftrightarrow} n'\, u' : T[u/x]}$$
Note that since $n\, u$ and $n'\, u'$ are well-typed in Δ, so are n and n', thus, by soundness of algorithmic equality, they have the type returned by structural equality check, $\Delta \vdash n, n' : (x : U) \to T$. By type unicity (Thm. 34), $\Delta \vdash u, u' : U$, thus, we can apply the induction hypotheses. By the first ind. hyp., $\Delta' \vdash n \longleftrightarrow n' : A'$ with $\Delta \vdash (x : U) \overset{s_1,s_2}{\to} T = A'$. Since A' is a whnf, by type constructor discrimination (Lemma 36), $A \equiv (x : U') \overset{s_1,s_2}{\to} T'$. By function type injectivity (Thm. 33), this implies $\Delta \vdash U = U' : s_1$ and $\Delta \vdash T[u/x] = T'[u/x] : s_2$ since by soundness of algorithmic equality (Thm. 38) $\Delta \vdash u = u' : U$. By the second ind. hyp. $\Delta' \vdash u \overset{\frown}{\Longleftrightarrow} u' : U'$, which yields $\Delta' \vdash n\, u \overset{\frown}{\longleftrightarrow} n'\, u' : T'[u/x]$.

(3) Type-directed equality.
Case $\Delta \vdash t, t' : T$ and $\Delta \vdash T = T'$ and
$$\frac{T \searrow A \qquad \Delta \vdash t \Longleftrightarrow t' : A}{\Delta \vdash t \overset{\frown}{\Longleftrightarrow} t' : T}$$
By normalization, $T' \searrow A'$, and subject reduction $\Delta \vdash A = T = T' = A'$. Since by conversion, $\Delta \vdash t, t' : A$, by induction hypothesis $\Delta' \vdash t \Longleftrightarrow t' : A'$. Thus, $\Delta' \vdash t \overset{\frown}{\Longleftrightarrow} t' : T'$.

Case $\Delta \vdash (x{\star}U) \to T = A'$ and

$$\frac{\Delta.\, x{\star}U \vdash t\,{}^{\star}x \mathrel{\widehat{\Longleftrightarrow}} t'\,{}^{\star}x : T}{\Delta \vdash t \Longleftrightarrow t' : (x{\star}U) \to T}$$

By injectivity $A' \equiv (x{\star}U') \to T'$ with $\Delta \vdash U = U'$ and $\Delta.\, x{\star}U \vdash T = T'$. Since $\vdash \Delta.\, x{\star}U = \Delta'.\, x{\star}U'$, by induction hypothesis we have $\Delta'.\, x{\star}U' \vdash t\,{}^{\star}x \mathrel{\widehat{\Longleftrightarrow}} t'\,{}^{\star}x : T'$. We conclude $\Delta' \vdash t \Longleftrightarrow t' : (x{\star}U') \to T'$.

□

Lemma 40 (Algorithmic equality is transitive)**.** Let $\vdash \Delta = \Delta'$. In the following, let the terms submitted to algorithmic equality be well-typed.

(1) If $\Delta \vdash n_1 \mathrel{\widehat{\longleftarrow}} n_2 : T$ and $\Delta' \vdash n_2 \mathrel{\widehat{\longleftarrow}} n_3 : T'$ then $\Delta \vdash n_1 \mathrel{\widehat{\longleftarrow}} n_3 : T$ and $\Delta \vdash T = T'$.
(2) If $\Delta \vdash t_1 \mathrel{\widehat{\Longleftrightarrow}} t_2 : T$ and $\Delta' \vdash t_2 \mathrel{\widehat{\Longleftrightarrow}} t_3 : T'$ and $\Delta \vdash T = T'$ then $\Delta \vdash t_1 \mathrel{\widehat{\Longleftrightarrow}} t_3 : T$.
(3) If $\Delta \vdash T_1 \mathrel{\widehat{\Longleftrightarrow}} T_2 : s$ and $\Delta' \vdash T_2 \mathrel{\widehat{\Longleftrightarrow}} T_3 : s$ then $\Delta \vdash T_1 \mathrel{\widehat{\Longleftrightarrow}} T_3 : s$

Proof. We extend these statements to $\longleftrightarrow$ and $\Longleftrightarrow$ and prove them simultaneously by induction on the first derivation.

Case

$$\frac{\Delta \vdash n_1 \mathrel{\widehat{\longleftarrow}} n_2 : T}{\Delta \vdash n_1 \Longleftrightarrow n_2 : N} \qquad \frac{\Delta' \vdash n_2 \mathrel{\widehat{\longleftarrow}} n_3 : T'}{\Delta' \vdash n_2 \Longleftrightarrow n_3 : N'}$$

By induction hypothesis $\Delta \vdash n_1 \mathrel{\widehat{\longleftarrow}} n_3 : T$, hence, $\Delta \vdash n_1 \Longleftrightarrow n_3 : N$.

Case

$$\frac{\Delta \vdash N_1 \mathrel{\widehat{\longleftarrow}} N_2 : T}{\Delta \vdash N_1 \Longleftrightarrow N_2} \qquad \frac{\Delta \vdash N_2 \mathrel{\widehat{\longleftarrow}} N_3 : T'}{\Delta \vdash N_2 \Longleftrightarrow N_3}$$

Analogously.

Case

$$\frac{\Delta \vdash n_1 \longleftrightarrow n_2 : (x{:}U) \overset{s_1,s_2}{\to} T \qquad \Delta \vdash u_1 \mathrel{\widehat{\Longleftrightarrow}} u_2 : U}{\Delta \vdash n_1\, u_1 \mathrel{\widehat{\longleftarrow}} n_2\, u_2 : T[u_1/x]}$$

$$\frac{\Delta' \vdash n_2 \longleftrightarrow n_3 : (x{:}U') \overset{s_1',s_2'}{\to} T' \qquad \Delta' \vdash u_2 \mathrel{\widehat{\Longleftrightarrow}} u_3 : U'}{\Delta' \vdash n_2\, u_2 \mathrel{\widehat{\longleftarrow}} n_3\, u_3 : T'[u_2/x]}$$

By induction hypothesis we have $\Delta \vdash n_1 \longleftrightarrow n_3 : (x{:}U) \overset{s_1,s_2}{\to} T$ and $\Delta \vdash (x{:}U) \overset{s_1,s_2}{\to} T = (x{:}U') \overset{s_1',s_2'}{\to} T'$ which gives in particular $s_1 = s_1', s_2 = s_2'$, and $\Delta \vdash U = U' : s_1$ by function type injectivity (Thm. 33). By induction hypothesis we can then deduce $\Delta \vdash u_1 \mathrel{\widehat{\Longleftrightarrow}} u_3 : U$, and therefore conclude $\Delta \vdash n_1 u_1 \longleftrightarrow n_3 u_3 : T[u_1/x]$.

Case

$$\frac{\Delta \vdash U_1 \mathrel{\widehat{\Longleftrightarrow}} U_2 : s_1 \quad \Delta.\, x{\star}U_1 \vdash T_1 \mathrel{\widehat{\Longleftrightarrow}} T_2 : s_2}{\Delta \vdash (x{\star}U_1) \overset{s_1,s_2}{\to} T_1 \Longleftrightarrow (x{\star}U_2) \overset{s_1,s_2}{\to} T_2 : s_3}$$

$$\frac{\Delta \vdash U_2 \mathrel{\widehat{\Longleftrightarrow}} U_3 : s_1 \quad \Delta.\, x{\star}U_2 \vdash T_2 \mathrel{\widehat{\Longleftrightarrow}} T_3 : s_2}{\Delta \vdash (x{\star}U_2) \overset{s_1,s_2}{\to} T_2 \Longleftrightarrow (x{\star}U_3) \overset{s_1,s_2}{\to} T_3 : s_2}$$

We get $\Delta \vdash U_1 \mathrel{\widehat{\Longleftrightarrow}} U_3 : s_1$ by transitivity. To also get $\Delta.\, x{\star}U_1 \vdash T_1 \mathrel{\widehat{\Longleftrightarrow}} T_3 : s_2$ we need $\vdash \Delta.\, x{\star}U_2 = \Delta.\, x{\star}U_1$, but this stems from $\Delta \vdash U_1 \mathrel{\widehat{\Longleftrightarrow}} U_2 : s_1$ by soundness of algorithmic equality.

□

Theorem 41. The algorithmic equality relations are PERs on well-typed expressions.

Proof. By Lemma 40 and an analogous proof of symmetry. □

6. A Kripke Logical Relation for Completeness

The only open issues in the meta-theory of IITT are completeness and termination of algorithmic equality. In parts, completeness has been established in the last section already, namely, we have shown injectivity and discrimination for type constructors. What is missing is injectivity and discrimination for neutrals, e. g., if $\Delta \vdash n\, u = n'\, u' : T'$ then necessarily $\Delta \vdash n = n' : (x : U) \to T$ and $\Delta \vdash u = u' : U$, plus $\Delta \vdash T[u/x] = T'$. In untyped λ-calculus, this is an instance of Boehm's theorem [Bar84]. We follow Coquand [Coq91] and Harper and Pfenning [HP05] and prove it by constructing a second Kripke logical relation, Ⓒ, for completeness which is very similar to the first one, Ⓢ, but at base types additionally requires algorithmic equality to hold. After proving the fundamental lemma again, we know that definitionally equal terms are also algorithmically so. As a consequence, equality is decidable in IITT, and so is type checking.

6.1. Another Kripke Logical Relation.

Again, by induction on $A \in s$ we define two Kripke relations

$$\Delta \vdash A \mathrel{Ⓒ} A' : s$$
$$\Delta \vdash a \mathrel{Ⓒ} a' : A.$$

together with their respective closures $\widehat{Ⓒ}$ and the generalization to $\star$. This time, however, at base types we will additionally require algorithmic equality to hold, more precisely, the relation $\Delta \vdash t :\Longleftrightarrow: t' : T$ which stands for the conjunction of the propositions

- $\Delta \vdash t : T$ and $\Delta \vdash t' : T$, and
- $\Delta \vdash t \mathrel{\widehat{\Longleftrightarrow}} t' : T$.

Note that by soundness of algorithmic equality, $:\Longleftrightarrow:$ implies $:=:$.

Again, we allow ourselves rule notation for the defining clauses of Ⓒ.

$$\frac{\Delta \vdash N :\Longleftrightarrow: N' : s}{\Delta \vdash N \mathrel{Ⓒ} N' : s} \qquad \frac{\Delta \vdash n :\Longleftrightarrow: n' : N}{\Delta \vdash n \mathrel{Ⓒ} n' : N} \qquad \frac{\vdash \Delta}{\Delta \vdash s \mathrel{Ⓒ} s : s'}\ (s, s')$$

$$\frac{\begin{array}{l}\Delta \vdash U \mathrel{\widehat{Ⓒ}} U' : s_1 \\ \forall \Gamma \le \Delta,\ \Gamma \vdash u \mathrel{\widehat{Ⓒ}} u' \star U \implies \Gamma \vdash T[u/x] \mathrel{\widehat{Ⓒ}} T'[u'/x] : s_2 \\ \Delta \vdash (x{\star}U) \overset{s_1,s_2}{\to} T :=: (x{\star}U') \overset{s_1,s_2}{\to} T' : s_3\end{array}}{\Delta \vdash (x{\star}U) \overset{s_1,s_2}{\to} T \mathrel{Ⓒ} (x{\star}U') \overset{s_1,s_2}{\to} T' : s_3}\ (s_1, s_2, s_3)$$

$$\frac{\begin{array}{c}\forall \Gamma \le \Delta,\ \Gamma \vdash u \mathrel{\widehat{Ⓒ}} u' \star U \implies \Gamma \vdash f\,{\star}\,u \mathrel{\widehat{Ⓒ}} f'\,{\star}\,u' : T[u/x] \\ \Delta \vdash f :=: f' : (x{\star}U) \overset{s,s'}{\to} T\end{array}}{\Delta \vdash f \mathrel{Ⓒ} f' : (x{\star}U) \overset{s,s'}{\to} T}$$

$$\frac{\Delta \vdash \downarrow t \mathbin{©} \downarrow t' : \downarrow T \qquad \Delta \vdash t :=: t' : T}{\Delta \vdash t \mathbin{\widehat{©}} t' : T}$$

$$\frac{\Delta^{\div} \vdash a \mathbin{©} a : A \qquad \Delta^{\div} \vdash a' \mathbin{©} a' : A}{\Delta \vdash a \mathbin{©} a' \div A} \qquad \frac{\Delta^{\div} \vdash t \mathbin{\widehat{©}} t : T \qquad \Delta^{\div} \vdash t' \mathbin{\widehat{©}} t' : T}{\Delta \vdash t \mathbin{\widehat{©}} t' \div T}$$

This logical relation contains only well-typed and definitionally equal terms. It is symmetric, transitive, and closed under weakening and type conversion. The proofs are in analogy to those of Section 4, which are relying on the fact that the underlying relation :=: is a Kripke PER and closed under type conversion. The relation :⟺: underlying © has the same properties, thanks to soundness of algorithmic equality.

Note that in the definition of $\Delta \vdash f \mathbin{©} f' : (x \star U) \to T$ we did not require f and f' to be algorithmically equal. This would hinder the proof of the fundamental theorem for ©, since algorithmic equality is not closed under application by definition—it will follow from the fundamental theorem, though. In the next lemma we shall prove that f and f' *are* algorithmically equal if they are related by ©. The name *Escape Lemma* was coined by Jeffrey Sarnat [SS08].

Lemma 42 (Escape from the logical relation)**.** Let $\Delta \vdash A \mathbin{©} A' : s$

(1) $\Delta \vdash A \Longleftrightarrow A'$.
(2) If $\Delta \vdash t \mathbin{\widehat{©}} t' : A$ then $\Delta \vdash t \Longleftrightarrow t' : A$.
(3) If $\Delta \vdash n \longleftrightarrow n' \star A$ and $\Delta \vdash n = n' \star A$ then $\Delta \vdash n \mathbin{©} n' \star A$.

Corollary 43. Let $\Delta \vdash T \mathbin{\widehat{©}} T' : s$

(1) $\Delta \vdash T \mathbin{\widehat{\Longleftrightarrow}} T'$.
(2) If $\Delta \vdash t \mathbin{\widehat{©}} t' : T$ then $\Delta \vdash t \mathbin{\widehat{\Longleftrightarrow}} t' : T$.
(3) If $\Delta \vdash n \mathbin{\widehat{\longleftrightarrow}} n' \star T$ and $\Delta \vdash n = n' \star T$ then $\Delta \vdash n \mathbin{\widehat{©}} n' \star T$.

The corollary is a direct, non-inductive consequence of the lemma, so we can use it in the proof of the lemma, quoted as "IH".

Proof of the lemma. Simultaneously by induction on A :⟺: $A' : s$.

Case $\Delta \vdash N \mathbin{©} N' : s$.

Subcase 1. $\Delta \vdash N \Longleftrightarrow N'$ by assumption.

Subcase 2. We have $\Delta \vdash \downarrow t \longleftrightarrow \downarrow t' : \_$, thus $\Delta \vdash t \Longleftrightarrow t' : N$.

Subcase 3.

First, consider $\star = :$. If $\Delta \vdash n = n' : N$ and $\Delta \vdash n \longleftrightarrow n' : N$ then $\Delta \vdash n \Longleftrightarrow n' : N$ and trivially $\Delta \vdash n \mathbin{©} n' : N$.

Then, take $\star = \div$. Note that if $\Delta^{\div} \vdash n = n : N$ and $\Delta^{\div} \vdash n \longleftrightarrow n : N$ then $\Delta^{\div} \vdash n \Longleftrightarrow n : N$ and $\Delta^{\div} \vdash n \mathbin{©} n : N$. This implies that if $\Delta \vdash n = n' \div N$ and $\Delta \vdash n \longleftrightarrow n' \div N$ then $\Delta \vdash n \Longleftrightarrow n' \div N$ and $\Delta \vdash n \mathbin{©} n' \div N$.

Case $\Delta \vdash s \mathbin{©} s : s'$.

Subcase 1. Clearly, $\Delta \vdash s \Longleftrightarrow s$.

Subcase 2. Let $\Delta \vdash T \mathbin{\widehat{©}} T' : s$. Then $\Delta \vdash T \mathbin{\widehat{\Longleftrightarrow}} T'$ by IH 1, thus $\Delta \vdash T \Longleftrightarrow T' : s$

Subcase 3. For $\star = :$ let $\Delta \vdash N \longleftrightarrow N' : s$. By inversion, $\Delta \vdash N \mathbin{\widehat{\longleftrightarrow}} N' : T$ for some T. Then $\Delta \vdash N \Longleftrightarrow N'$ and $\Delta \vdash N \mathbin{©} N' : s$ by definition.

Considering $\star = \div$, it is sufficient to observe that $\Delta^{\div} \vdash N \longleftrightarrow N : s$ implies $\Delta^{\div} \vdash N \Longleftrightarrow N$ and $\Delta^{\div} \vdash N \mathbin{©} N : s$ by definition.

Case $\Delta \vdash (x \star U) \to T \mathbin{©} (x \star U') \to T' : s_3$.

Subcase 1. Similar to 2.

Subcase 2. By assumption, $\Delta \vdash t \mathbin{\widehat{©}} t' : (x \star U) \to T$. It is sufficient to show $\Delta.\, x \star U \vdash t\, ^\star x \mathrel{\widehat{\Longleftrightarrow}} t'\, ^\star x : T$. Since $\Delta \vdash U \mathbin{\widehat{©}} U' : s_1$, which includes $\Delta \vdash U$, we have $\Delta.\, x \star U \vdash x = x \star U$. Since also $\Delta.\, x \star U \vdash x \mathrel{\widehat{\longleftrightarrow}} x \star U$, we obtain $\Delta.\, x \star U \vdash t\, ^\star x \mathbin{\widehat{©}} t'\, ^\star x : \downarrow T$ via IH 3, $\Delta.\, x \star U \vdash x \mathbin{©} x \star U$. IH 2 then entails our goal.

Subcase 3. First, the case for $\star = :$. We reuse variable $\star$ for a different irrelevance marker. We have $\Delta \vdash n \longleftrightarrow n' : (x \star U) \to T$. Assume arbitrary $\Gamma :\Longleftrightarrow: \Gamma \leq \Delta$ and $\Gamma \vdash u \mathbin{\widehat{©}} u' \star U$, which yields $\Gamma \vdash u = u' \star U$ and $\Gamma \vdash T[u/x] \mathbin{\widehat{©}} T[u'/x] : \mathsf{Set}_i$. In case $\star = :$ we have to apply IH 2 for $\Gamma \vdash u \Longleftrightarrow u' : \downarrow U$. Otherwise, we obtain directly $\Gamma \vdash n\, ^\star u \longleftrightarrow n'\, ^\star u' : \downarrow(T[u/x])$. By IH 3, $\Gamma \vdash n\, ^\star u \mathbin{©} n'\, ^\star u' : \downarrow(T[u/x])$.
The case for $\star = \div$ proceeds analogously.

□

In analogy to $\widehat{Ⓢ}$ we extend $\widehat{©}$ to substitutions and define the semantic validity judgements $\Vdash^{\mathsf{c}} \Gamma$ and $\Gamma \Vdash^{\mathsf{c}} t : T$ and $\Gamma \Vdash^{\mathsf{c}} t = t' : T$ based on $\widehat{©}$. Since by the escape lemma, $\Delta \vdash x \mathbin{\widehat{©}} x : \Delta(x)$, we have $\Gamma \vdash \mathsf{id} \mathbin{\widehat{©}} \mathsf{id} : \Gamma$ for $\Vdash^{\mathsf{c}} \Gamma$. Finally, we reprove the fundamental theorem:

Theorem 44 (Fundamental theorem for $\widehat{©}$)**.**

(1) If $\vdash \Gamma$ then $\Vdash^{\mathsf{c}} \Gamma$.
(2) If $\Gamma \vdash t : T$ then $\Gamma \Vdash^{\mathsf{c}} t : T$.
(3) If $\Gamma \vdash t = t' : T$ then $\Gamma \Vdash^{\mathsf{c}} t = t' : T$.

6.2. **Completeness and Decidability of Algorithmic Equality.** Derivations of algorithmic equality can now be obtained by escaping from the logical relation.

Theorem 45 (Completeness of algorithmic equality)**.** If $\Gamma \vdash t = t' : T$ then $\Gamma \vdash t \mathrel{\widehat{\Longleftrightarrow}} t' : T$.

Proof. Since $\Gamma \vdash \mathsf{id} \mathbin{\widehat{©}} \mathsf{id} : \Gamma$, we have $\Gamma \vdash t \mathbin{\widehat{©}} t' : T$ by the fundamental theorem, and conclude with Lemma 42.2. □

Termination of algorithmic equality is a consequence of completeness. When invoking the algorithmic equality check $\Delta \vdash t \mathrel{\widehat{\Longleftrightarrow}} t' : T$ on two well-typed expressions $\Delta \vdash t, t' : T$ we know by completeness that t and t' are related to themselves, i. e., $\Delta \vdash t \mathrel{\widehat{\Longleftrightarrow}} t : T$ and $\Delta \vdash t' \mathrel{\widehat{\Longleftrightarrow}} t' : T$. This means that t, t', and T are weakly normalizing by the strategy the equality algorithm implements: reduce to weak head normal form and recursively continue with the subterms. Running the equality check on t and t' performs, if successful, exactly the same reductions, and if it fails, at most the same reductions in t, t', and T. Hence, testing equality on well-typed terms always terminates. This argument has been applied in previous work to untyped equality [AC07]. Here, we apply it to typed equality; it is an alternative to Goguen's technique of proving termination for typed equality from strong normalization [Gog05], which, in our opinion, does not scale to dependently-typed equality.

Lemma 46 (Termination of algorithmic equality)**.** Let $\vdash \Delta$.

(1) Type equality.

(a) Let $\Delta \vdash A, A'$. If $\mathcal{D} :: \Delta \vdash A \iff A$ and $\Delta \vdash A' \iff A'$ then the query $\Delta \vdash A \iff A'$ terminates.
(b) Let $\Delta \vdash T, T'$. If $\mathcal{D} :: \Delta \vdash T \mathrel{\widehat{\iff}} T$ and $\Delta \vdash T' \mathrel{\widehat{\iff}} T'$ then the query $\Delta \vdash T \mathrel{\widehat{\iff}} T'$ terminates.

(2) Structural equality. Let $\Delta \vdash n : T$ and $\Delta \vdash n' : T'$.
(a) If $\mathcal{D} :: \Delta \vdash n \longleftrightarrow n : A$ and $\Delta \vdash n' \longleftrightarrow n' : A'$ then the query $\Delta \vdash n \longleftrightarrow n' : ?$ terminates. If successfully, it returns A and we have $\Delta \vdash A = T = T' = A$.
(b) If $\mathcal{D} :: \Delta \vdash n \mathrel{\widehat{\longleftrightarrow}} n : T$ and $\Delta \vdash n' \mathrel{\widehat{\longleftrightarrow}} n' : T'$ then the query $\Delta \vdash n \mathrel{\widehat{\longleftrightarrow}} n' : ?$ terminates. If successfully, it returns T and we have $\Delta \vdash T = T'$.

(3) Type-directed equality.
(a) Let $\Delta \vdash t, t' : A$. If $\mathcal{D} :: \Delta \vdash t \iff t : A$ and $\Delta \vdash t' \iff t' : A$ then the query $\Delta \vdash t \iff t' : A$ terminates.
(b) Let $\Delta \vdash t, t' : T$. If $\mathcal{D} :: \Delta \vdash t \mathrel{\widehat{\iff}} t : T$ and $\Delta \vdash t' \mathrel{\widehat{\iff}} t' : T$ then the query $\Delta \vdash t \mathrel{\widehat{\iff}} t' : T$ terminates.

Proof. Simultaneously by induction on derivation $\mathcal{D}$.

(1) Type equality.
Case $A = A' = s$. The query $\Delta \vdash A \iff A'$ terminates successfully.
Case $A = (x{\star}U) \to T$ and $A' = (x{\star}U') \to T'$. First, the query $\Delta \vdash U \mathrel{\widehat{\iff}} U'$ runs. By induction hypothesis, it terminates. If it fails, the whole query fails. Otherwise, the query $\Delta.\, x{\star}U \vdash T \mathrel{\widehat{\iff}} T'$ is run. By induction hypothesis on $\Delta.\, x{\star}U \vdash T \mathrel{\widehat{\iff}} T$ and $\Delta.\, x{\star}U' \vdash T' \mathrel{\widehat{\iff}} T'$, the query terminates.
Case $A = N$ and $A' = N'$ neutral. By induction hypothesis on $\Delta \vdash N \mathrel{\widehat{\longleftrightarrow}} N : T$ and $\Delta \vdash N' \mathrel{\widehat{\longleftrightarrow}} N' : T'$, the query $\Delta \vdash N \mathrel{\widehat{\longleftrightarrow}} N' : ?$ terminates. Hence, the query $\Delta \vdash N \iff N'$ terminates.
Case Weak head normal forms A, A' not covered by previous cases: the query $\Delta \vdash A \iff A'$ fails immediately, since there is no applicable algorithmic type equality rule.
Case The query $\Delta \vdash T \mathrel{\widehat{\iff}} T'$ first invokes weak head normalization on T and T'. Both terminate since $\Delta \vdash T \mathrel{\widehat{\iff}} T$, which implies $T \searrow A$, and analogously $T' \searrow A'$ since $\Delta \vdash T' \mathrel{\widehat{\iff}} T'$ by assumption. Then, the query $\Delta \vdash A \iff A'$ is run, which terminates by induction hypothesis on $\Delta \vdash A \iff A$ and $\Delta \vdash A' \iff A'$.

(2) Structural equality.
Case $n = n' = x$. The query $\Delta \vdash n \mathrel{\widehat{\longleftrightarrow}} n' : ?$ terminates successfully, returning type $\Delta(x)$. Since $\vdash \Delta$, by inversion (Lemma 31) $\Delta \vdash T = T' = \Delta(x)$.
Case Neutral relevant application for $\Delta \vdash n\, u : T_0$ and $\Delta \vdash n'\, u' : T_0'$.

$$\frac{\Delta \vdash n \longleftrightarrow n : (x{:}U) \to T \qquad \Delta \vdash u \mathrel{\widehat{\iff}} u : U}{\Delta \vdash n\, u \mathrel{\widehat{\longleftrightarrow}} n\, u : T[u/x]}$$

$$\frac{\Delta \vdash n' \longleftrightarrow n' : (x{:}U') \to T' \qquad \Delta \vdash u' \mathrel{\widehat{\iff}} u' : U'}{\Delta \vdash n'\, u' \mathrel{\widehat{\longleftrightarrow}} n'\, u' : T'[u'/x]}$$

The query $\Delta \vdash n\, u \mathrel{\widehat{\longleftrightarrow}} n'\, u' : ?$ first invokes query $\Delta \vdash n \longleftrightarrow n' : ?$. By induction hypothesis on $\Delta \vdash n \longleftrightarrow n : (x{:}U) \to T$ and $\Delta \vdash n' \longleftrightarrow n' : (x{:}U') \to T'$ the query terminates. If it fails the whole query fails. Otherwise it returns a type A in weak head normal form, which is identical to $(x{:}U) \to T$ by uniqueness of inferred types (Lemma 9). Further, $\Delta \vdash (x : U) \to T =$

$(x : U') \to T'$, and by function type injectivity (Thm. 33), $\Delta \vdash U = U'$ and $\Delta . x : U \vdash T = T'$. Thus, we can invoke the induction hypothesis on $\Delta \vdash u \mathrel{\widehat{\Longleftrightarrow}} u : U$ and $\Delta \vdash u' \mathrel{\widehat{\Longleftrightarrow}} u' : U$ (cast from $\Delta \vdash u' \mathrel{\widehat{\Longleftrightarrow}} u' : U'$, Lemma 39) to infer that the second subquery $\Delta \vdash u \mathrel{\widehat{\Longleftrightarrow}} u' : U$ terminates. If this one is successful, then by soundness of algorithmic equality, $\Delta \vdash u = u' : U$, which implies $\Delta \vdash T[u/x] = T'[u'/x]$.

Case Neutral irrelevant application with typing

$$\frac{\Delta \vdash n : (x \div U_1) \to T_1 \qquad \Delta \vdash u \div U_1}{\Delta \vdash n \div u : T_1[u/x]} \qquad \frac{\Delta \vdash n' : (x \div U_1') \to T_1' \qquad \Delta \vdash u' \div U_1'}{\Delta \vdash n' \div u' : T_1'[u'/x]}$$

and algorithmic self-equality

$$\frac{\Delta \vdash n \longleftrightarrow n : (x \div U) \to T}{\Delta \vdash n \div u \mathrel{\widehat{\longleftrightarrow}} n \div u : T[u/x]} \qquad \frac{\Delta \vdash n' \longleftrightarrow n' : (x \div U') \to T'}{\Delta \vdash n' \div u' \mathrel{\widehat{\longleftrightarrow}} n' \div u' : T'[u'/x]}$$

The query $\Delta \vdash n \div u \mathrel{\widehat{\longleftrightarrow}} n' \div u' : ?$ invokes query $\Delta \vdash n \longleftrightarrow n' : ?$, which terminates by induction hypothesis. If successfully, then $\Delta \vdash (x \div U_1) \to T_1 (x \div U) \to T = (x \div U') \to T' = (x \div U_1') \to T_1'$. By function type injectivity, $\Delta \vdash U_1 = U = U' = U_1'$ and $\Delta . x \div U \vdash T_1 = T = T' = T_1'$. By conversion $\Delta \vdash u = u' \div U$, thus, $\Delta \vdash T_1[u/x] = T[u/x] = T'[u'/x] = T_1'[u'/x]$.

Case In all other cases, the query $\Delta \vdash n \mathrel{\widehat{\longleftrightarrow}} n' : ?$ fails immediately.

Case The query $\Delta \vdash n \longleftrightarrow n : ?$ spawns subquery $\Delta \vdash n \mathrel{\widehat{\longleftrightarrow}} n' : ?$ which terminates by induction hypothesis on $\Delta \vdash n \mathrel{\widehat{\longleftrightarrow}} n : T$ and $\Delta \vdash n' \mathrel{\widehat{\longleftrightarrow}} n' : T'$. If successfully, it returns type T, and since $T \searrow A$, the original query also terminates, returning A.

(3) Type-directed equality.

Case Function type $\Delta \vdash t, t' : (x \star U) \to T$. The query $\Delta \vdash t \Longleftrightarrow t' : (x \star U) \to T$ spawns subquery $\Delta . x \star U \vdash t\,{}^\star x \mathrel{\widehat{\Longleftrightarrow}} t'\,{}^\star : T$. Since $\Delta . x \star U \vdash t\,{}^\star x, t'\,{}^\star x : T$ and the subquery terminates by induction hypothesis on $\Delta . x \star U \vdash t\,{}^\star x \mathrel{\widehat{\Longleftrightarrow}} t\,{}^\star x : T$ and $\Delta . x \star U \vdash t'\,{}^\star x \mathrel{\widehat{\Longleftrightarrow}} t'\,{}^\star x : T$.

Case Sort $\Delta \vdash T, T' : s$. The query $\Delta \vdash T \Longleftrightarrow T' : s$ calls $\Delta \vdash T \mathrel{\widehat{\Longleftrightarrow}} T'$, which terminates by induction hypothesis on $\Delta \vdash T \mathrel{\widehat{\Longleftrightarrow}} T$ and $\Delta \vdash T' \mathrel{\widehat{\Longleftrightarrow}} T'$.

Case Neutral type N.

$$\frac{t \searrow n \qquad \Delta \vdash n \mathrel{\widehat{\longleftrightarrow}} n : T}{\Delta \vdash t \Longleftrightarrow t : N} \qquad \frac{t' \searrow n' \qquad \Delta \vdash n' \mathrel{\widehat{\longleftrightarrow}} n' : T'}{\Delta \vdash t' \Longleftrightarrow t' : N}$$

The query $\Delta \vdash t \Longleftrightarrow t' : N$ first weak head normalizes t and t'. By assumption, $t \searrow n$ and $t' \searrow n'$, so this terminates. The subquery $\Delta \vdash n \mathrel{\widehat{\longleftrightarrow}} n' : ?$ terminates by induction hypothesis. Thus, the whole query terminates.

Case If A is neither a function type, a sort, or a neutral type, the query $\Delta \vdash t \Longleftrightarrow t' : A$ fails immediately.

Case The query $\Delta \vdash t \mathrel{\widehat{\Longleftrightarrow}} t' : T$ first weak head normalizes T which terminates since $T \searrow A$ by assumption. Then it calls $\Delta \vdash t \Longleftrightarrow t' : A$ which terminates by induction hypothesis.

□

Theorem 47. If $\Delta \vdash t : T$ and $\Delta \vdash t' : T$ then the query $\Delta \vdash t \mathrel{\widehat{\Longleftrightarrow}} t' : T$ terminates.

Proof. From the lemma by completeness of algorithmic equality. □

Thus we have shown that algorithmic equality is correct, i. e., sound, complete, and terminating. Together, this entails decidability of equality in IITT.

Theorem 48 (Decidability of IITT)**.**

(1) $\Gamma \vdash t = t' : T$ is decidable.
(2) $\Gamma \vdash t : T$ is decidable.

Proof. Decidability of equality follows from soundness (Thm. 38), completeness (Thm. 45), and termination (Thm. 47). Decidability of typing follows from decidability of type conversion, weak head normalization, and function type injectivity, using inversion (Lemma 31) on typing derivations. Any reasonable type inference algorithm will do. □

7. Extensions

Data types and recursion. The semantics of IITT is ready to cope with inductive data types like the natural numbers and the associated recursion principles. Recursion into types, aka known as large elimination, is also accounted for since we have universes and a semantics which does not erase dependencies (unlike Pfenning's model [Pfe01]).

Types with extensionality principles. One purpose of having a typed equality algorithm is to handle η-laws that are not connected to the shape of the expression (like η-contraction for functions) but to the shape of the type only. Typically these are types T with at most one inhabitant, i. e., the empty type, the unit type, singleton types or propositions.[6] For such T we have the η-law

$$\frac{\Gamma \vdash t, t' : T}{\Gamma \vdash t = t' : T}$$

which can only be checked in the presence of type T. Realizing such η-laws gives additional "proof" irrelevance which is not covered by Pfenning's irrelevant quantification $(x \div U) \to T$.

Internal erasure. Terms $u \div U$ in irrelevant position are only there to please the type checker, they are ignored during equality checking. This can be inferred from the substitution principle: If $\Gamma.\, x \div U \vdash T$ and $\Gamma \vdash u, u' \div U$, then $\Gamma \vdash T[u/x] = T[u'/x]$; the type T has the same shape regardless of u, u'. Hence, terms like u serve the sole purpose to prove some proposition and could be replaced by a dummy $\bullet$ immediately after type-checking.

Internal erasure can be realized by making $\Gamma \vdash t \div T$ a judgement (as opposed to just a notation for $\Gamma^{\div} \vdash t : T$) and adding the rule

$$\frac{\Gamma \vdash t \div T}{\Gamma \vdash \bullet \div T}.$$

The rule states that if there is already a proof t of T, then $\bullet$ is a new proof of T. This preserves provability while erasing the proof terms. Conservativity of this rule can be proven as in joint work of the author with Coquand and Pagano [ACP11].

[6] Some care is necessary for the type of Leibniz equality [Abe09, Wer08].

8. Conclusions

We have extended Pfenning's notion of irrelevance to a type theory IITT with universes that accommodates types defined by recursion. We have constructed a Kripke model Ⓢ that shows soundness of IITT, yielding normalization, subject reduction and consistency, plus syntactical properties of the judgements of IITT. A second Kripke logical relation Ⓒ has proven correctness of algorithmic equality and, thus, decidability of IITT.

Integrating irrelevance and data types in dependent type theory does not seem without challenges. We have succeeded to treat Pfenning's notion of irrelevance, but our proof does not scale directly to *parametric* function types, a stronger notion of irrelevant function types called implicit quantification by Miquel [Miq01b].[7] Two more type theories build on Miquel's calculus [Miq01a], Barras and Bernardo's ICC* [BB08] and Mishra-Linger and Sheard's *Erasure Pure Type Systems* (EPTS) [MLS08], but none has offered a satisfying account of large eliminations yet. Miquel's model [Miq00] features data types only as impredicative encodings. For irrelevant, parametric, and recursive functions to coexist it seems like three different function types are necessary, e. g., in the style of Pfenning's *irrelevance, extensionality and intensionality*. We would like to solve this puzzle in future work, not least to implement high-performance languages with dependent types.

Acknowledgments. The first author thanks Bruno Barras, Bruno Bernardo, Thierry Coquand, Dan Doel, Hugo Herbelin, Conor McBride, Ulf Norell, and Jason Reed for discussions on irrelevance in type theory. Work on a previous paper has been carried out while he was invited researcher at PPS, Paris, in the INRIA πr^2 team headed by Pierre-Louis Curien and Hugo Herbelin. The second author acknowledges financial support by the École Normale Superiéure de Paris for his internship at the Ludwig-Maximilians-Universität München from May to September 2011. We thank the two anonymous referees, who suggested improvements and examples which improved the presentation, and the patience of the editors.

References

[AB04] Steven Awodey and Andrej Bauer. Propositions as [Types]. *Journal of Logic and Computation*, 14(4):447–471, 2004.

[Abe09] Andreas Abel. Extensional normalization in the logical framework with proof irrelevant equality. In Olivier Danvy, editor, *Workshop on Normalization by Evaluation, affiliated to LiCS 2009, Los Angeles, 15 August 2009*, 2009.

[Abe11] Andreas Abel. Irrelevance in type theory with a heterogeneous equality judgement. In Martin Hofmann, editor, *Foundations of Software Science and Computational Structures, 14th International Conference, FOSSACS 2011, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2011, Saarbrücken, Germany, March 26 - April 3, 2011. Proceedings*, volume 6604 of *Lecture Notes in Computer Science*, pages 57–71. Springer-Verlag, 2011.

[AC07] Andreas Abel and Thierry Coquand. Untyped algorithmic equality for Martin-Löf's logical framework with surjective pairs. *Fundamenta Informaticae*, 77(4):345–395, 2007. TLCA'05 special issue.

[ACD07] Andreas Abel, Thierry Coquand, and Peter Dybjer. 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*, pages 3–12. IEEE Computer Society Press, 2007.

[7] A function argument is parametric if it is irrelevant for computing the function result while the type of the result may depend on it. In Pfenning's notion, the argument must also be irrelevant in the type.

[ACD08] Andreas Abel, Thierry Coquand, and Peter Dybjer. Verifying a semantic $\beta\eta$-conversion test for Martin-Löf type theory. In Philippe Audebaud and Christine Paulin-Mohring, editors, *Mathematics of Program Construction, 9th International Conference, MPC 2008, Marseille, France, July 15-18, 2008. Proceedings*, volume 5133 of *Lecture Notes in Computer Science*, pages 29–56. Springer-Verlag, 2008.

[ACP11] Andreas Abel, Thierry Coquand, and Miguel Pagano. A modular type-checking algorithm for type theory with singleton types and proof irrelevance. *Logical Methods in Computer Science*, 7(2:4):1–57, May 2011.

[Ada06] Robin Adams. Pure type systems with judgemental equality. *Journal of Functional Programming*, 16(2):219–246, 2006.

[All87] Stuart Allen. *A Non-Type-Theoretic Semantics for Type-Theoretic Language*. PhD thesis, Cornell University, 1987.

[Ama08] Roberto M. Amadio, editor. *Foundations of Software Science and Computational Structures, 11th International Conference, FOSSACS 2008, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2008, Budapest, Hungary, March 29 - April 6, 2008. Proceedings*, volume 4962 of *Lecture Notes in Computer Science*. Springer-Verlag, 2008.

[Aug99] Lennart Augustsson. Cayenne - a language with dependent types. In *Proceedings of the third ACM SIGPLAN International Conference on Functional Programming (ICFP '98), Baltimore, Maryland, USA, September 27-29, 1998*, volume 34 of *SIGPLAN Notices*, pages 239–250. ACM Press, 1999.

[Bar84] Henk Barendregt. *The Lambda Calculus: Its Syntax and Semantics*. North Holland, Amsterdam, 1984.

[BB08] Bruno Barras and Bruno Bernardo. The implicit calculus of constructions as a programming language with dependent types. In Amadio [Ama08], pages 365–379.

[BDN09] Ana Bove, Peter Dybjer, and Ulf Norell. A brief overview of Agda - a functional language with dependent types. In Stefan Berghofer, Tobias Nipkow, Christian Urban, and Makarius Wenzel, editors, *Theorem Proving in Higher Order Logics, 22nd International Conference, TPHOLs 2009, Munich, Germany, August 17-20, 2009. Proceedings*, volume 5674 of *Lecture Notes in Computer Science*, pages 73–78. Springer-Verlag, 2009.

[CAB+86] Robert L. Constable, Stuart F. Allen, Mark Bromley, Rance Cleaveland, J. F. Cremer, Robert W. Harper, Douglas J. Howe, Todd B. Knoblock, Nax P. Mendler, Prakash Panangaden, James T. Sasaki, and Scott F. Smith. *Implementing mathematics with the Nuprl proof development system*. Prentice Hall, 1986.

[Coq91] Thierry Coquand. An algorithm for testing conversion in type theory. In G. Huet and G. Plotkin, editors, *Logical Frameworks*, pages 255–279. Cambridge University Press, 1991.

[Coq96] Thierry Coquand. An algorithm for type-checking dependent types. In *Mathematics of Program Construction. Selected Papers from the Third International Conference on the Mathematics of Program Construction (July 17–21, 1995, Kloster Irsee, Germany)*, volume 26 of *Science of Computer Programming*, pages 167–177. Elsevier, May 1996.

[Gog94] Healfdene Goguen. *A Typed Operational Semantics for Type Theory*. PhD thesis, University of Edinburgh, August 1994. Available as LFCS Report ECS-LFCS-94-304.

[Gog00] Healfdene Goguen. A Kripke-style model for the admissibility of structural rules. In Paul Callaghan, Zhaohui Luo, James McKinna, and Robert Pollack, editors, *Types for Proofs and Programs, International Workshop, TYPES 2000, Durham, UK, December 8-12, 2000, Selected Papers*, volume 2277 of *Lecture Notes in Computer Science*, pages 112–124. Springer-Verlag, 2000.

[Gog05] Healfdene Goguen. Justifying algorithms for $\beta\eta$ conversion. In Vladimiro Sassone, editor, *Foundations of Software Science and Computational Structures, 8th International Conference, FoSSaCS 2005, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2005, Edinburgh, UK, April 4-8, 2005, Proceedings*, volume 3441 of *Lecture Notes in Computer Science*, pages 410–424. Springer-Verlag, 2005.

[HP05] Robert Harper and Frank Pfenning. On equivalence and canonical forms in the LF type theory. *ACM Transactions on Computational Logic*, 6(1):61–101, 2005.

[INR10] INRIA. *The Coq Proof Assistant Reference Manual*. INRIA, version 8.3 edition, 2010. http://coq.inria.fr/.

[Let02] Pierre Letouzey. A new extraction for Coq. In Herman Geuvers and Freek Wiedijk, editors, *Types for Proofs and Programs, Second International Workshop, TYPES 2002, Berg en Dal, The Netherlands, April 24-28, 2002, Selected Papers*, volume 2646 of *Lecture Notes in Computer Science*, pages 200–219. Springer-Verlag, 2002.

[Miq00] Alexandre Miquel. A model for impredicative type systems, universes, intersection types and subtyping. In *15th IEEE Symposium on Logic in Computer Science (LICS 2000), 26-29 June 2000, Santa Barbara, California, USA, Proceedings*, pages 18–29, 2000.

[Miq01a] Alexandre Miquel. The implicit calculus of constructions. In Samson Abramsky, editor, *Typed Lambda Calculi and Applications, 5th International Conference, TLCA 2001, Krakow, Poland, May 2-5, 2001, Proceedings*, volume 2044 of *Lecture Notes in Computer Science*, pages 344–359. Springer-Verlag, 2001.

[Miq01b] Alexandre Miquel. *Le Calcul des Constructions implicite: syntaxe et sémantique.* PhD thesis, Université Paris 7, December 2001.

[ML08] Richard Nathan Mishra-Linger. *Irrelevance, Polymorphism, and Erasure in Type Theory.* PhD thesis, Portland State University, 2008.

[MLS08] Nathan Mishra-Linger and Tim Sheard. Erasure and polymorphism in pure type systems. In Amadio [Ama08], pages 350–364.

[MM04] Conor McBride and James McKinna. The view from the left. *Journal of Functional Programming*, 14(1):69–111, 2004.

[Pfe01] Frank Pfenning. Intensionality, extensionality, and proof irrelevance in modal type theory. In *16th IEEE Symposium on Logic in Computer Science (LICS 2001), 16-19 June 2001, Boston University, USA, Proceedings*. IEEE Computer Society Press, 2001.

[PMW93] Christine Paulin-Mohring and Benjamin Werner. Synthesis of ML programs in the system Coq. *Journal of Symbolic Computation*, 15(5/6):607–640, 1993.

[Ree02] Jason Reed. Proof irrelevance and strict definitions in a logical framework, 2002. Senior Thesis, published as Carnegie-Mellon University technical report CMU-CS-02-153.

[Ree03] Jason Reed. Extending higher-order unification to support proof irrelevance. In David A. Basin and Burkhart Wolff, editors, *Theorem Proving in Higher Order Logics, 16th International Conference, TPHOLs 2003, Rom, Italy, September 8-12, 2003, Proceedings*, volume 2758 of *Lecture Notes in Computer Science*, pages 238–252. Springer-Verlag, 2003.

[SS08] Carsten Schürmann and Jeffrey Sarnat. Structural logical relations. In Frank Pfenning, editor, *Proceedings of the Twenty-Third Annual IEEE Symposium on Logic in Computer Science, LICS 2008, 24-27 June 2008, Pittsburgh, PA, USA*, pages 69–80. IEEE Computer Society Press, 2008.

[VC02] Joseph C. Vanderwaart and Karl Crary. A simplified account of the metatheory of Linear LF. In *Third International Workshop on Logical Frameworks and Metalanguages (LFM 2002), FLoC'02 affiliated workshop, Copenhagen, Denmark*, 2002. An extended version appeared as CMU Technical Report CMU-CS-01-154.

[Wer08] Benjamin Werner. On the strength of proof-irrelevant type theories. *Logical Methods in Computer Science*, 4(3), 2008.