Observational Equality: Now For Good

Loïc Pujet, Nicolas Tabareau

Observational Equality: Now for Good

LOÏC PUJET, Inria, France
NICOLAS TABAREAU, Inria, France

Building on the recent extension of dependent type theory with a universe of definitionally proof-irrelevant types, we introduce $\mathsf{TT}^{\mathsf{obs}}$, a new type theory based on the setoidal interpretation of dependent type theory. $\mathsf{TT}^{\mathsf{obs}}$ equips every type with an identity relation that satisfies function extensionality, propositional extensionality, and definitional uniqueness of identity proofs (UIP). Compared to other existing proposals to enrich dependent type theory with these principles, our theory features a notion of reduction that is normalizing and provides an algorithmic canonicity result, which we formally prove in Agda using the logical relation framework of Abel et al. Our paper thoroughly develops the meta-theoretical properties of $\mathsf{TT}^{\mathsf{obs}}$, such as the decidability of the conversion and of the type checking, as well as consistency. We also explain how to extend our theory with quotient types, and we introduce a setoidal version of Swan's Id types that turn it into a proper extension of MLTT with inductive equality.

CCS Concepts: • **Theory of computation → Type theory**.

Additional Key Words and Phrases: type theory, dependent types, rewriting theory, confluence, termination

ACM Reference Format:
Loïc Pujet and Nicolas Tabareau. 2022. Observational Equality: Now for Good. *Proc. ACM Program. Lang.* 6, POPL, Article 32 (January 2022), 29 pages. https://doi.org/10.1145/3498693

1 INTRODUCTION

Dependent type theories and in particular MLTT, as originally developed by Martin-Löf [1975], provide an adequate framework for developing constructive mathematics and certifying software. A core aspect of MLTT is the coexistence of two distinct notions of equality: a *definitional* equality that records the equations automated by the system, and a *propositional* equality that is a type internal to the system and thus can be used to do equational reasoning. However, the propositional equality of MLTT lacks some extensionality principles that pervade mathematical reasoning, such as function extensionality (funext). Since they are generally considered desirable, these principles are sometimes added as axioms, but doing so results in a system with weaker computational properties.

Throughout the years, several options to obtain a more extensional version of propositional equality while preserving computation have been explored. The most successful lines of work can be roughly divided into two groups: the ones using an observational equality, and the ones using a cubical equality. Both notions build on the following fundamental idea: in order to obtain sufficient extensionality principles, the behavior of equality in a given type should be explicitly specified for every type instead of using a single definition that is parametric over the types.

The work on observational equality originated from the study of the setoid model of type theory [Altenkirch 1999; Hofmann 1995], and a first attempt at a proper type theory that supports an

Authors' addresses: Loïc Pujet, Inria, Gallinette Project-Team, Nantes, France; Nicolas Tabareau, Inria, Gallinette Project-Team, Nantes, France.

2475-1421/2022/1-ART32
https://doi.org/10.1145/3498693

observational equality satisfying funext has been been proposed in [Altenkirch et al. 2007]. In these systems, observational equality equips every type with a setoid structure. This setoidal equality satisfies the uniqueness of identity proofs (UIP), which states that all proofs of an equality are equal; in other words, equality is proof irrelevant.

The other line of work is more recent and takes its roots in the formulation of the univalence axiom [Kapulkin and Lumsdaine 2018; Univalent Foundations Program 2013], which gives a new meaning to the equality between types: two types are equal when they equivalent. This can be understood as an extensionality principle for the universe of types. In their search for a computational interpretation of univalence, Cohen et al. [2015] developed a notion of cubical equality, which satisfies funext and univalence, but is incompatible with UIP.

Thus, observational equality and cubical equality are two diverging directions for providing propositional equality with more extensionality. Altenkirch et al. [2016]; Capriotti [2017]; Voevodsky [2013] advocate that the two solutions are actually complementary and can be integrated to a single system with two universe hierarchies, one that satisfies univalence and the other that satisfies UIP. While cubical type theory has been thoroughly investigated and even implemented in the Agda proof assistant Vezzosi et al. [2019], observational equality has not reached a comparable level of maturity. Recent attempts to design a type theory based on observational equality are either lacking an algorithm for type checking Sterling et al. [2019], or restricted to a single universe and having computational properties only up to a conjecture Altenkirch et al. [2019]. Indeed, the latter relies on computation in an enriched version of MLTT that features a universe of definitionally proof-irrelevant types (noted hereafter Ω_i) as recently proposed by Gilbert et al. [2019], along with a proof-irrelevant identity type that supports a strong eliminator. This theory has not been justified yet, and it has even been shown not to be normalizing in presence of impredicativity Abel and Coquand [2020].

In this paper, we define $\mathsf{TT}^{\mathrm{obs}}$, the first extension of MLTT + Ω_i with an observational equality that satisfies UIP, funext and propext, and supports quotient types and countably many universes —with a proof of normalization and canonicity formalized in Agda.[1] Firstly, we remark that this theory can only be derived in a system with some level of cumulativity, as it is required for certain computation rules to be well-typed. This makes explicit some difficulties that do not show up in previous works. Second, we remark that our version of observational equality can be equipped with two elimination principles: a notion of type cast (or coercion) for proof relevant types, and the standard eliminator of MLTT for proof irrelevant types, thus making all the setoidal structure derivable from the standard eliminator. Funext and propext are obtained by specifying the right computation rules for observational equality whereas UIP is obtained for free by interpreting observational equality in Ω.

The proof of normalization and canonicity is based on the use of logical relations defined in Agda using induction-recursion as initially developed by Abel et al. [2018] and later extended to Ω by Gilbert et al. [2019]. Compared to previous work on formalized normalization proofs, we have added the support for a cumulative hierarchy with two universes, and we make a clear distinction between inhabitants of proof-irrelevant types, which have no computational behavior, and inhabitants of proof relevant types, which do. This key change allows us to add new principles in Ω without having to supply them with a computational behavior, trivializing for instance the management of higher coherences of the cubical interpretation. In counterpart for this added flexibility, normalization does not directly imply canonicity anymore, and a separate proof of consistency of the theory is required to derive canonicity. This proof is done by defining a model

[1] The formalization is available at https://github.com/CoqHott/logrel-mltt/, references to a particular file are done using *[myfile.agda]* which directly points to the corresponding file on github.

s	::=	$\mathcal{U} \mid \Omega$	Relevances
i, j	$\in$	$\mathbb{N}$	Universe levels
Γ, Δ	::=	$\bullet \mid \Gamma, x :^{s,i} A$	Contexts
t, u, m, n, e, A, B	::=	$x \mid s_i$	Variables and Universes
	$\mid$	$\lambda(x : A).\, t \mid t\ u \mid \Pi^j_{s,i}(x : A).\, B$	Dependent products
	$\mid$	$0 \mid \mathrm{S}\ t \mid \mathbb{N}\text{-}\mathrm{elim}(P, t, u, n) \mid \mathbb{N}$	Natural numbers
	$\mid$	$\langle t, u\rangle \mid \mathrm{fst}(t) \mid \mathrm{snd}(t) \mid \exists^j_i(x : A).\, B$	Existential types
	$\mid$	$\bot\text{-}\mathrm{elim}(A, t) \mid \bot$	Empty type
	$\mid$	$* \mid \top$	Singleton type
	$\mid$	$t \sim_A u \mid \mathrm{refl}(t) \mid \mathrm{transp}(t, B, u, t', e)$	Observational equality
	$\mid$	$\mathrm{cast}(A, B, e, t) \mid \mathrm{castrefl}(A, t)$	Type cast

Fig. 1. Syntax of $\mathsf{TT}^{\mathrm{obs}}$

for $\mathsf{TT}^{\mathrm{obs}}$, but as consistency is the only consequence that we need from the existence of a model, the model can be defined in an extensional setting.

To illustrate the simplicity of extending MLTT +Ω to $\mathsf{TT}^{\mathrm{obs}}$, we have implemented a simple version in Agda using rewrite rules, as recently introduced by Cockx et al. [2021] (file *[setoid_rr.agda]*).

Plan of the paper. In Section 2, we present the syntax and typing rules of $\mathsf{TT}^{\mathrm{obs}}$, focusing on its notions of conversion and reduction. Then in Section 3, we develop the logical relation framework, and the model from which we derive consistency. This shows normalization, canonicity and decidability of typechecking for $\mathsf{TT}^{\mathrm{obs}}$. In Section 4, we explain how to extend $\mathsf{TT}^{\mathrm{obs}}$ to support quotient types, box types and squash types, as well as the standard identity type of MLTT, while preserving meta-theoretical properties.

2 A TYPE THEORY WITH OBSERVATIONAL EQUALITY

$\mathsf{TT}^{\mathrm{obs}}$ has two cumulative hierarchies of universes. The first one, which we write $\mathcal{U}_i$ (where i is a natural number), is the usual universe hierarchy of Martin-Löf type theory. The second family, which we write Ω_i, is the family of *definitionally proof-irrelevant types*, which means that any two inhabitants of a type $A : \Omega_i$ are convertible. This corresponds to the two hierarchies presented in Gilbert et al. [2019] and implemented in Agda, except that we also introduce some cumulativity features that will prove useful when interpreting equality in the universe of propositions and for dependent function types. Having a proof-irrelevant hierarchy of types seems to be crucial to interpret an extensional equality that satisfies UIP. Indeed, it trivializes all of the higher coherences that naturally arise when considering proofs of equality between equalities, and provides a canonical inhabitant for the type that encodes UIP, which is simply given by reflexivity.

2.1 Syntax of $\mathsf{TT}^{\mathrm{obs}}$

The syntax of the sorts, contexts, terms and types of $\mathsf{TT}^{\mathrm{obs}}$ is specified in Fig. 1. The theory features cumulative dependent functions (noted $\Pi^j_{s,i}(x : A).\, B$) with η-equality, natural numbers, as well as proof-irrelevant dependent sums (noted $\exists(x : A).\, B$), a proof-irrelevant empty type (noted $\bot$) and unit type (noted $\top$). When B does not depend on A, we write $A \to B$ instead of $\Pi(x : A).\, B$, and $A \wedge B$ instead of $\exists(x : A).\, B$. The capture-avoiding substitution of a variable x in a term A by the term t is noted $B[x := t]$. $\mathsf{TT}^{\mathrm{obs}}$ can also be extended with more primitives such as quotients or general inductive types, but we defer the treatment of those to Section 4 to better focus on the treatment of equality in the theory.

Compared to MLTT with proof irrelevant types, there are two fundemental new ingredients in $\mathsf{TT}^{\mathrm{obs}}$. The first is a proof-irrelevant observational equality type $t \sim_A u$ that encodes equality of t and u at type A, together with a term $\mathrm{refl}(t)$ that witnesses reflexivity of this equality. The second is an operation that casts a term t from a type A to another type B according to a proof of equality e between A and B, which is noted $\mathrm{cast}(A, B, e, t)$. This operation comes with a witness $\mathrm{castrefl}(A, t)$ that casting between a type A and itself does nothing. Note that since observational equality is proof irrelevant, any proof of equality between A and A is convertible to $\mathrm{refl}(A)$. The theory also features a generic transport $\mathrm{transp}(t, B, u, t', e)$ when A and B are proof irrelevant.

Observational equality is quite different from the usual Martin-Löf Identity Type, which is an inductive type that computes via the J-eliminator. Instead, observational equality $t \sim_A u$ should be understood as an eliminator that reduces the type A to its weak head normal form, and then reduces by pattern-matching on it, much like the path type from Cubical Type Theory. In particular, the reduction rules for observational equality between two dependent products, or two types in Ω_i provide us with function extensionality and propositional extensionality by definition.

2.2 Typing Rules of $\mathsf{TT}^{\mathrm{obs}}$

The typing rules of $\mathsf{TT}^{\mathrm{obs}}$ are presented in Fig. 2. They are based on four kinds of judgments: $\vdash \Gamma$ (well-formedness of a context), $\Gamma \vdash t :^{s,i} A$ (typing of a term), $\Gamma \vdash t \equiv u :^{s,i} A$ (convertibility of terms), and $\Gamma \vdash t \Rightarrow u :^{i} A$ (weak-head reduction of relevant terms). To avoid clutter, we will omit the ^{s,i} annotations that denote the sort and level of a type as that can generally be inferred from the context.[2] Similarly, we will often omit the annotations on the dependent function type and simply write $\Pi(x : A).\, B$; except when the annotations are explicitly required to understand a rule. In all the judgments, s denotes either $\mathcal{U}$ or Ω.

Generic rules and universes. Rules Ctx-Nil, Ctx-Cons and Var describe the usual formation of contexts and typing of variables. Rule conv stipulates that type checking is done modulo conversion of types. The formation rule for universes (Rule Univ) states that both $\mathcal{U}_i$ and Ω_i are relevant types. It is natural for Ω to be proof-relevant, since its inhabitants are types which are not convertible to one another despite being proof-irrelevant.

Dependent products. The formation of dependent products between a domain and a codomain that have different sorts and universe levels is allowed. The resulting type has the same relevance as the codomain, and a universe level that is higher than both the level of the domain and the level of the codomain (Rule Π-Form). The sort of the domain and the levels of both the domain and the codomain are collected as annotations to the dependent function type for the convenience of the type checking algorithm. Note that the introduction of cumulativity in the formation of dependent products is a necessary complication of $\mathsf{TT}^{\mathrm{obs}}$ as the rules that provide us with propositional extensionality and function extensionality can only be well-typed in the presence of cumulativity.

Rules Fun and App are the usual rules of λ-abstraction and application. Note that the sort and level annotation are implicit as they can be inferred from the premises.

Existential types. $\mathsf{TT}^{\mathrm{obs}}$ features proof-irrelevant dependent sums. They are required to describe the behavior of observational equality between two dependent function types. We do not need them to be cumulative so we keep the formation rule as simple as possible (∃-Form).

We give a negative presentation of dependent sums with one introduction rule (Rule Pair) and two projections (Rules Fst and Snd). Note that because of proof irrelevance, the negative and positive presentations are completely equivalent, and we do not need any rule to account for the

[2] The interested reader may look at the Agda formalization where all annotations have been made explicit.

$$\textsc{Ctx-Nil}\ \frac{}{\vdash \bullet} \qquad \textsc{Ctx-Cons}\ \frac{\vdash \Gamma \qquad \Gamma \vdash A : s_i}{\vdash \Gamma, x : A} \qquad \textsc{Var}\ \frac{\vdash \Gamma \qquad x : A \in \Gamma}{\Gamma \vdash x : A} \qquad \textsc{conv}\ \frac{\Gamma \vdash t : A \qquad \Gamma \vdash A \equiv B : s_i}{\Gamma \vdash t : B}$$

$$\textsc{Univ}\ \frac{\vdash \Gamma}{\Gamma \vdash s_i : \mathcal{U}_j}{\scriptstyle i<j} \qquad \Pi\textsc{-Form}\ \frac{\Gamma \vdash A : s_i \qquad \Gamma, x : A \vdash B : s'_j}{\Gamma \vdash \Pi^j_{s,i}(x : A).\, B : s'_k}{\scriptstyle i \le k \quad j \le k} \qquad \exists\textsc{-Form}\ \frac{\Gamma \vdash A : \Omega_i \qquad \Gamma, x : A \vdash B : \Omega_i}{\Gamma \vdash \exists(x : A).\, B : \Omega_i}$$

$$\textsc{Fun}\ \frac{\Gamma \vdash A : s_i \qquad \Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda(x : A).\, t : \Pi(x : A).\, B} \qquad \textsc{App}\ \frac{\Gamma \vdash t : \Pi(x : A).\, B \qquad \Gamma \vdash u : A}{\Gamma \vdash t\ u : B[x := u]}$$

$$\textsc{Pair}\ \frac{\Gamma \vdash t : A \qquad \Gamma \vdash u : B[x := t]}{\Gamma \vdash \langle t, u\rangle : \exists(x : A).\, B} \qquad \textsc{Fst}\ \frac{\Gamma \vdash t : \exists(x : A).\, B}{\Gamma \vdash \mathsf{fst}(t) : A} \qquad \textsc{Snd}\ \frac{\Gamma \vdash t : \exists(x : A).\, B}{\Gamma \vdash \mathsf{snd}(t) : B[x := \mathsf{fst}(t)]}$$

$$\mathbb{N}\textsc{-Form}\ \frac{\vdash \Gamma}{\Gamma \vdash \mathbb{N} : \mathcal{U}_0} \qquad \textsc{Zero}\ \frac{\vdash \Gamma}{\Gamma \vdash 0 : \mathbb{N}} \qquad \textsc{Suc}\ \frac{\Gamma \vdash n : \mathbb{N}}{\Gamma \vdash \mathsf{S}\ n : \mathbb{N}}$$

$$\mathbb{N}\textsc{-Elim}\ \frac{\Gamma \vdash A : \mathbb{N} \to s_i \qquad \Gamma \vdash t_0 : A\ 0 \qquad \Gamma \vdash t_S : \Pi(n : \mathbb{N}).\, A\ n \to A\ (\mathsf{S}\ n) \qquad \Gamma \vdash n : \mathbb{N}}{\Gamma \vdash \mathbb{N}\text{-}\mathsf{elim}(A, t_0, t_S, n) : A\ n}$$

$$\bot\textsc{-Form}\ \frac{\vdash \Gamma}{\Gamma \vdash \bot : \Omega_i} \qquad \bot\textsc{-Elim}\ \frac{\Gamma \vdash A : s_i \qquad \Gamma \vdash t : \bot}{\Gamma \vdash \bot\text{-}\mathsf{elim}(A, t) : A} \qquad \top\textsc{-Form}\ \frac{\vdash \Gamma}{\Gamma \vdash \top : \Omega_i} \qquad \top\textsc{-Intro}\ \frac{\vdash \Gamma}{\Gamma \vdash * : \top}$$

$$\textsc{Eq-Form}\ \frac{\Gamma \vdash A : \mathcal{U}_i \qquad \Gamma \vdash t : A \qquad \Gamma \vdash u : A}{\Gamma \vdash t \sim_A u : \Omega_i} \qquad \textsc{Refl}\ \frac{\Gamma \vdash A : \mathcal{U}_i \qquad \Gamma \vdash t : A}{\Gamma \vdash \mathsf{refl}(t) : t \sim_A t}$$

$$\textsc{Transport-}\Omega\ \frac{\Gamma \vdash A : \mathcal{U}_i \quad \Gamma \vdash t : A \quad \Gamma \vdash B : \Pi(x : A).\, t \sim_A x \to \Omega_j \quad \Gamma \vdash u : B\ t\ \mathsf{refl}(t) \quad \Gamma \vdash t' : A \quad \Gamma \vdash e : t \sim_A t'}{\Gamma \vdash \mathsf{transp}(t, B, u, t', e) : B\ t'\ e}$$

$$\textsc{Cast}\ \frac{\Gamma \vdash A : s_i \quad \Gamma \vdash B : s_i \quad \Gamma \vdash e : A \sim_s B \quad \Gamma \vdash t : A}{\Gamma \vdash \mathsf{cast}(A, B, e, t) : B} \qquad \textsc{Cast-Refl}\ \frac{\Gamma \vdash A : s_i \qquad \Gamma \vdash t : A \qquad \Gamma \vdash e : A \sim_s A}{\Gamma \vdash \mathsf{castrefl}(A, t) : t \sim_A \mathsf{cast}(A, A, e, t)}$$

Fig. 2. $\mathsf{TT}^{\mathsf{obs}}$ Typing Rules

computational behavior of projections, nor η-equality. This is because any well-typed equality between two inhabitants of a proof-irrelevant type is proven by reflexivity (modulo Rule CONV).

Natural numbers. Strictly speaking, all inductive types could be excluded from the core of $\mathsf{TT}^{\mathsf{obs}}$, and be considered as extensions. However, we decided to treat the example of natural numbers as we feel it is both very simple and a good illustration.

The type $\mathbb{N}$ comes with a formation rule $\mathbb{N}$-FORM, constructors 0 (Rule ZERO) and S (Rule SUC) and an elimination/induction principle given by Rule $\mathbb{N}$-ELIM. Since we do not require cumulativity

for the type of natural numbers, we decided to only allow it in the lowest universe $\mathcal{U}_0$, but it could be made to inhabit all relevant universes without added difficulty. Extensions to other inductive types and quotient types are presented in Section 4.

Empty and Unit type. To encode respectively the true equalities and the absurd equalities, $\mathsf{TT}^{\mathrm{obs}}$ features the unit type $\top$ and the empty type $\bot$. Besides its formation rule (Rule $\bot$-Form), the empty type comes with an elimination principle (Rule $\bot$-Elim) that can eliminate it into both proof-irrelevant and proof-relevant types. In MLTT with a proof irrelevant universe hierarchy, this constitutes the unique connection between the proof-irrelevant and relevant types. This means that the only way to use information from a proof-irrelevant term to build a proof-relevant term is by using a proof of $\bot$, which amounts to proving that actually we are in an inaccessible branch or impossible case. In $\mathsf{TT}^{\mathrm{obs}}$ however, there is another very different way of using proof irrelevant information to build a proof relevant term, by transporting (or casting) along a proof of equality.

The unit type on the other hand only requires a formation rule (Rule $\top$-Form) and a constructor $*$ (Rule $\top$-Intro). It does not require an eliminator, since the usual one can be derived from proof irrelevance. Equivalently, we could have defined the unit type as $\bot \to \bot$. Note that both the unit type and the empty types are made cumulative, that is because they are used as terminal cases for the computation of observational equality, which can live at any level.

Setoid equality and Type Casts. A central feature of $\mathsf{TT}^{\mathrm{obs}}$ is that every proof-relevant type comes equipped with a proof-irrelevant equality type, noted $t \sim_A u$ (Rule Eq-Form) and a canonical way to inhabit it, by the reflexivity term $\mathsf{refl}(t)$ (Rule Refl). It is not necessary to equip proof-irrelevant types with a propositional equality, as two terms would always be propositional equal by reflexivity. As we will see in the definition of reduction (Fig. 4), the equality type of $\mathsf{TT}^{\mathrm{obs}}$ should not be seen as a type constructor like the identity type of MLTT, but rather as a type eliminator that computes on its type argument and possibly on its two end points.

For this equality type to be of any use, we need to be able to eliminate it as well. The usual J-eliminator defined in MLTT can be defined in our context, but if the predicate is proof-relevant, there will be no hope for the eliminator to compute on reflexivity: since equality types are proof irrelevant, there is no reduction of equality proofs, and we cannot do pattern matching on the equality proof either. In order to avoid this issue, we restrict the use of J to proof-irrelevant predicates only, and note it transp (Rule Transport-Ω). Note that in this setting, the non-dependent version of the eliminator would be enough, as one can prove the contractibility of singletons for free thanks to proof-irrelevance [Univalent Foundations Program 2013]. Using transp, we can derive the usual groupoid laws provided by the J eliminator, and we note e^{-1} for the inverse of e, $e \cdot e'$ for the transitivity, and $\mathsf{ap}\ f\ e$ for the preservation of equality by non-dependent function.

To deal with elimination of equality in a proof relevant context, we introduce a cast primitive that handles transport between two propositionally equal proof relevant types. Note that our cast operation also applies to proof-irrelevant types (Rule Cast). This is unnecessary as Rule Transport-Ω subsumes this case, but it will allow us to write more uniform reduction rules. The term $\mathsf{cast}(A, B, e, t)$ is an eliminator that reduces the types A and B in weak-head normal form, then reduces by pattern-matching on their head constructors. This will be explained in more detail in the next section. When applied to reflexivity, cast does not compute as the identity function, but this equality is propositionally admissible as witnessed by $\mathsf{castrefl}$ (Rule Cast-Refl).

2.3 Conversion and Reduction to Weak-Head Normal Forms

Conversion (Fig. 3) subsumes reduction (Red-Conv), η-equality of functions (Rule η-Eq), and proof-irrelevance (Rule Proof-Irrelevance). It is also closed under reflexivity, symmetry, transitivity and

$$\textsc{Refl}\ \frac{\Gamma \vdash t : A}{\Gamma \vdash t \equiv t : A} \qquad \textsc{Sym}\ \frac{\Gamma \vdash t \equiv u : A}{\Gamma \vdash u \equiv t : A} \qquad \textsc{Trans}\ \frac{\Gamma \vdash t \equiv t' : A \qquad \Gamma \vdash t' \equiv u : A}{\Gamma \vdash t \equiv u : A} \qquad \textsc{Red-Conv}\ \frac{\Gamma \vdash t \Rightarrow u : A}{\Gamma \vdash t \equiv u : A}$$

$$\eta\text{-Eq}\ \frac{\Gamma \vdash t, u : \Pi(x : A).\, B \qquad \Gamma, x : A \vdash t\ x \equiv u\ x : B}{\Gamma \vdash t \equiv u : \Pi(x : A).\, B} \qquad \textsc{Proof-Irrelevance}\ \frac{\Gamma \vdash A : \Omega_i \qquad \Gamma \vdash t : A \qquad \Gamma \vdash u : A}{\Gamma \vdash t \equiv u : A}$$

Fig. 3. $\mathsf{TT}^{\mathsf{obs}}$ Conversion Rules (except congruence)

congruence. Note that η-equality does not need to be defined when the codomain is in Ω, because of proof-irrelevance. Congruence rules are standard and presented in Appendix A, Fig. 9.

Fig. 4 describes the reduction in $\mathsf{TT}^{\mathsf{obs}}$. Note that since reduction is typed, one needs to add a rule for type conversion (Rule Conv-Red). Reduction features the usual notion of β-reduction (Rule β-red), which corresponds to the computation rule of application (the eliminator of dependent functions) applied to a λ-term (the constructor of dependent functions). Similarly, rules $\mathbb{N}$-Elim-Zero and $\mathbb{N}$-Elim-Suc describe the two computation rules of the eliminator of $\mathbb{N}$, one for each constructor of $\mathbb{N}$.

As mentioned previously, the equality type and type cast operation of $\mathsf{TT}^{\mathsf{obs}}$ are eliminators, and thus must come with their computation rules. Since $t \sim_A u$ is an eliminator for the universe, it needs rules for every type constructor. Rule Eq-Fun is the computation rule for dependent functions which stipulates that two functions are equal when they are pointwise equal, thus naturally providing $\mathsf{TT}^{\mathsf{obs}}$ with function extensionality. Rule Eq-Ω says that two proof irrelevant types are equal when they are logically equivalent, thus providing $\mathsf{TT}^{\mathsf{obs}}$ with propositional extensionality. Note that cumulativity is required for this reduction rule to preserve typing, as the types A and B live in a universe below $A \sim_{\Omega_i} B$.

There are three reduction rules for equality in $\mathcal{U}$. Rule Eq-Univ stipulates that when the two endpoints are both $\mathbb{N}$ or the same universe, the equality holds, which is captured by the fact that it reduces to the unit type. Rule Eq-Univ-$\neq$ says that when the two end points do not have the same head hd, the equality does not hold, which is captured by the fact that it reduces to the empty type. The function hd A is simply equal to $\mathbb{N}$ or s_i when A is the type of natural numbers or a universe respectively, and is defined as (Π, s, i, j) when A is the type $\Pi^j_{s,i}(x : A).\, B$. The third rule (Rule Eq-Π) says that two dependent function types are equal when their domain are equal, and their codomain are pointwise equal (as type families) up to the equality on their domain. As for Rule Eq-Ω, this rule is well-typed only in presence of cumulativity.

There are four rules for equality in $\mathbb{N}$, corresponding to every possible normal forms of the endpoints. Equality holds when both endpoints are 0 (Rule Eq-zero), computes to the equality of arguments when both endpoints are successors (Rule Eq-Suc), and does not hold otherwise (Rules Eq-zero-suc and Eq-suc-zero). This concludes the reduction rules for equality.

We can now turn to the description of the computation rules for type cast, which only need to be defined when the types are compatible—otherwise the cast will be stuck.[3] Casting from $\mathbb{N}$ to $\mathbb{N}$ is defined by recursion to be the identity on constructors (Rules Cast-Zero and Cast-Suc), and casting a type from a universe to the same universe is the identity[4] (Rule Cast-Univ). Finally,

[3] An alternative design would be to reduce to an elimination on the proof of equality which in this case has type $\bot$.
[4] We could also define it by recursion. It is not clear whether it makes any significant difference.

$$\beta\text{-red}\quad \frac{\Gamma, x : A \vdash t : B \qquad \Gamma \vdash u : A}{\Gamma \vdash (\lambda(x : A).\, t)\ u \Rightarrow t[x := u] : B[x := u]}$$

$$\textsc{Conv-Red}\quad \frac{\Gamma \vdash t \Rightarrow u : A \qquad \Gamma \vdash A \equiv B : s_i}{\Gamma \vdash t \Rightarrow u : B}$$

$$\mathbb{N}\text{-Elim-Zero}\quad \frac{\Gamma \vdash A : \mathbb{N} \to \mathcal{U}_i \qquad \Gamma \vdash t_0 : A\ 0 \qquad \Gamma \vdash t_S : \Pi(n : \mathbb{N}).\, A\ n \to A\ (\mathrm{S}\ n)}{\Gamma \vdash \mathbb{N}\text{-elim}(A, t_0, t_S, 0) \Rightarrow t_0 : A\ 0}$$

$$\mathbb{N}\text{-Elim-Suc}\quad \frac{\Gamma \vdash A : \mathbb{N} \to \mathcal{U}_i \qquad \Gamma \vdash t_0 : A\ 0 \qquad \Gamma \vdash t_S : \Pi(n : \mathbb{N}).\, A\ n \to A\ (\mathrm{S}\ n) \qquad \Gamma \vdash n : \mathbb{N}}{\Gamma \vdash \mathbb{N}\text{-elim}(A, t_0, t_S, \mathrm{S}\ n) \Rightarrow t_S\ \mathbb{N}\text{-elim}(A, t_0, t_S, n) : A\ (\mathrm{S}\ n)}$$

$$\textsc{Eq-Fun}\quad \frac{\Gamma \vdash f : \Pi(x : A).\, B \qquad \Gamma \vdash g : \Pi(x : A).\, B}{\Gamma \vdash f \sim_{\Pi AB} g \Rightarrow \Pi(x : A).\, f\ x \sim_B g\ x : \Omega_i}$$

$$\textsc{Eq-}\Omega\quad \frac{\Gamma \vdash A : \Omega_i \qquad \Gamma \vdash B : \Omega_i}{\Gamma \vdash A \sim_{\Omega_i} B \Rightarrow (A \to B) \wedge (B \to A) : \Omega_j}\ i<j$$

$$\textsc{Eq-Univ}\quad \frac{\vdash \Gamma \qquad A \in \{\mathbb{N}, s_i\}}{\Gamma \vdash A \sim_{\mathcal{U}_j} A \Rightarrow \top : \Omega_k}\ i<j<k$$

$$\textsc{Eq-Univ-}\neq\quad \frac{\vdash \Gamma \qquad A, B \in \{\mathbb{N}, \Pi A.B, s_i\} \qquad \mathrm{hd}\, A \neq \mathrm{hd}\, B}{\Gamma \vdash A \sim_{\mathcal{U}_j} B \Rightarrow \bot : \Omega_k}\ i<j<k$$

$$\textsc{Eq-}\Pi\quad \frac{\Gamma \vdash A, A' : s_i \qquad \Gamma, x : A \vdash B : s'_j \qquad \Gamma, x : A' \vdash B' : s'_j \qquad a := \mathrm{cast}(A', A, e^{-1}, a')}{\Gamma \vdash \begin{array}{l}\Pi(x : A).\, B \sim_{\mathcal{U}k} \Pi(x : A').\, B' \Rightarrow \\ \exists(e : A \sim_{\mathcal{U}i} A').\, \Pi(a' : A').\, B[x := a] \sim_{\mathcal{U}j} B'[x := a']\end{array} : \Omega_i}\ \begin{array}{l} i \le k \\ j \le k \end{array}$$

$$\textsc{Eq-zero}\quad \frac{\vdash \Gamma}{\Gamma \vdash 0 \sim_{\mathbb{N}} 0 \Rightarrow \top : \Omega_i} \qquad \textsc{Eq-zero-suc}\quad \frac{\Gamma \vdash n : \mathbb{N}}{\Gamma \vdash 0 \sim_{\mathbb{N}} \mathrm{S}\ n \Rightarrow \bot : \Omega_i} \qquad \textsc{Eq-suc-zero}\quad \frac{\Gamma \vdash n : \mathbb{N}}{\Gamma \vdash \mathrm{S}\ n \sim_{\mathbb{N}} 0 \Rightarrow \bot : \Omega_i}$$

$$\textsc{Eq-Suc}\quad \frac{\Gamma \vdash n : \mathbb{N} \qquad \Gamma \vdash m : \mathbb{N}}{\Gamma \vdash \mathrm{S}\ m \sim_{\mathbb{N}} \mathrm{S}\ n \Rightarrow m \sim_{\mathbb{N}} n : \Omega_i} \qquad \textsc{Cast-Zero}\quad \frac{\Gamma \vdash e : \mathbb{N} \sim_{\mathcal{U}} \mathbb{N}}{\Gamma \vdash \mathrm{cast}(\mathbb{N}, \mathbb{N}, e, 0) \Rightarrow 0 : \mathbb{N}}$$

$$\textsc{Cast-Suc}\quad \frac{\Gamma \vdash e : \mathbb{N} \sim_{\mathcal{U}} \mathbb{N} \qquad \Gamma \vdash n : \mathbb{N}}{\Gamma \vdash \mathrm{cast}(\mathbb{N}, \mathbb{N}, e, \mathrm{S}\ n) \Rightarrow \mathrm{S}\ \mathrm{cast}(\mathbb{N}, \mathbb{N}, e, n) : \mathbb{N}} \qquad \textsc{Cast-Univ}\quad \frac{\Gamma \vdash e : s_i \sim_{\mathcal{U}} s_i \qquad \Gamma \vdash A : s_i}{\Gamma \vdash \mathrm{cast}(s_i, s_i, e, A) \Rightarrow A : s_i}$$

$$\textsc{Cast-}\Pi\quad \frac{\Gamma \vdash e : \Pi(x : A).\, B \sim_{\mathcal{U}} \Pi(x : A').\, B' \qquad \Gamma \vdash f : \Pi(x : A).\, B \qquad a := \mathrm{cast}(A', A, \mathrm{fst}(e)^{-1}, a')}{\Gamma \vdash \begin{array}{l}\mathrm{cast}(\Pi(x : A).\, B, \Pi(x : A').\, B', e, f) \Rightarrow \\ \lambda(a' : A').\, \mathrm{cast}(B[x := a], B'[x := a'], \mathrm{snd}(e)\ a', f\ a)\end{array} : \Pi(x : A').\, B'}$$

Fig. 4. $\mathrm{TT}^{\mathrm{obs}}$ Reduction Rules (except substitutions)

casting between two dependent products is more involved: it produces a new function by casting back and forth the argument and return value of the original function (Rule Cast-Π).

For our definition of reduction to be complete, we need to add rules that reduce the scrutinees of eliminators to *weak-head normal form* (whnf), so the eliminator can then reduce by case analysis on the whnf. Weak-head normal forms correspond to relevant terms that can *not* be reduced (Fig. 5).

$$
\begin{array}{llll}
\text{whnf} & w & ::= & ne \mid \Pi(x : A).\, B \mid s_i \mid \exists(x : A).\, B \mid \mathbb{N} \mid \bot \mid \top \mid \lambda(x : A).\, t \mid 0 \mid \mathsf{S}\, n \\
\text{neutral} & ne & ::= & x \mid ne\ t \mid \bot\text{-elim}(A, e) \mid \mathbb{N}\text{-elim}(P, t, u, ne) \\
 & & & \mid t \sim_{ne} u \mid ne \sim_{\mathbb{N}} m \mid 0 \sim_{\mathbb{N}} ne \mid \mathsf{S}\, m \sim_{\mathbb{N}} ne \\
 & & & \mid ne \sim_{\mathcal{U}_i} B \mid \mathbb{N} \sim_{\mathcal{U}_i} ne \mid \Pi(x : A).\, B \sim_{\mathcal{U}_i} ne \mid \mathsf{cast}(\mathbb{N}, \mathbb{N}, e, ne) \\
 & & & \mid \mathsf{cast}(ne, B, e, t) \mid \mathsf{cast}(\mathbb{N}, ne, e, t) \mid \mathsf{cast}(\Pi(x : A).\, B, ne, e, t) \\
 & & & \mid \mathsf{cast}(w, w', e, t) \qquad (\text{where } w, w' \in \{\mathbb{N}, \Pi A.B, s_i\}, \mathrm{hd}\, w \neq \mathrm{hd}\, w')
\end{array}
$$

Fig. 5. Weak-head normal and neutral forms

They are either terms with a constructor in head position, or a neutral term, which correspond to the stuck terms that can not exist in an empty context. Neutral terms are either variables, eliminators applied to another neutral term, or obtained from an inhabitant of $\bot$. In $\mathsf{TT}^{\mathsf{obs}}$, inhabitants of a proof-irrelevant type are never considered as whnf, as there is no notion of reduction of proof-irrelevant terms. Substitution rules that implement reduction of scrutinees to weak-head normal form are described in Appendix A, Fig. 10.

3 METATHEORETICAL PROPERTIES

In order for a type theory to be a reasonable candidate for implementation in a proof assistant, it is desirable to have some control over its behavior and its semantics. There is a wide variety of properties we can ask of a type theory; we will describe four of them and prove they apply to $\mathsf{TT}^{\mathsf{obs}}$.

Consistency. A theory is called consistent if it is impossible to use it to derive a contradiction. It goes without saying that this property is of utmost importance if one wants to build interesting mathematics. When proving a theory consistent, one should pay some attention to the meta-theory where reasoning takes place, as a consistency result is really a reduction of the consistency of the theory to the consistency of the meta-theory.

Normalization. If a type theory features reduction rules, one may want to make sure that by repeatedly applying said rules, all terms eventually reach a normal form that cannot be reduced further. One says that the reduction strategy is normalizing.

Canonicity. A term belonging to a type is called canonical when it can be explicitly built up using the constructors of that type. For instance, an inhabitant of the type $\mathbb{N}$ of natural numbers is canonical if it is an explicit numeral. If all terms of type $\mathbb{N}$ in an empty context normalize to a canonical form, then the theory is said to enjoy canonicity for natural numbers.

Decidability of Type Checking. Finally, in order to implement a type theory in a proof assistant, it is desirable to have an algorithm that, given a context Γ and terms t and A, decides whether t is an inhabitant of A in context Γ. This ensures users that when they find a proof for a statement, the proof assistant can automatically check that their proof is correct.

In this section, we will prove these four properties for $\mathsf{TT}^{\mathsf{obs}}$. We first present our extension of the logical relations framework developed by Abel et al. [2018] and later extended to Ω by Gilbert et al. [2019]. The framework provides a proof of normalization and canonicity from consistency, as well as decidability of conversion and thus decidability of type checking. It has been formalized in the Agda development. Then, to get consistency and therefore canonicity, we develop a model of $\mathsf{TT}^{\mathsf{obs}}$ in a constructive set theory that supports inductive-recursive definitions and one Grothendieck universe.

3.1 Normalization and Canonicity from Consistency

We now explain the proof by reducibility as initially formalized in Agda by Abel et al. [2018] and extended to Ω by Gilbert et al. [2019]. The proof starts from untyped syntax *[Untyped.agda]*,

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

Fig. 6. The four judgments of the logical relation

with syntactical operations such as weakening and substitution, which are used to define typing judgments *[Typed.agda]*. Abel et al. [2018] also identify the weak head normal forms: define a neutral term to be a term that has a variable in head position, which blocks reduction. Weak head normal forms are then defined to be either neutral terms or terms that have a constructor in head position. This is the notion that we extended to $\mathsf{TT}^{\mathrm{obs}}$ in Fig. 5.

From there, Abel et al. [2018] use induction-recursion to define a Kripke logical relation that exhibits the structure and properties of well-typed terms, and implies weak-head normalization. The proof builds up to the *fundamental lemma*, which states that any well-typed term is reducible by induction on the typing derivation. Abel et al. [2018] prove this for a dependent type theory with dependent functions, natural numbers and one universe. Gilbert et al. [2019] extends it with one universe of definitionally proof irrelevant types.

In our work, we added support for two cumulative universe levels, existential types, proof irrelevant axioms, setoid equality and cast. This doubles the size of the proof, getting it up to 20,000 lines of Agda code.

Our most novel contribution to the proof is our treatment of proof-irrelevant types. Even though the theory studied by Gilbert et al. [2019] features proof-irrelevant types, their reducibility proof extends reduction rules to the inhabitants of these types, and the normalization result applies to them too. From this, Gilbert et al. [2019] derive an easy proof of consistency: any proof of $\bot$ in an an empty context will reduce to a weak head normal form, and the only weak head normal forms that can inhabit $\bot$ are neutral terms. But since there are no variables in the empty context, neutral terms cannot exist, so $\bot$ has no inhabitant in the empty context.

However, this strategy is not applicable in our setting: it seems difficult to devise reduction rules for some terms that we postulated, such as castrefl, so that they reduce to a normal form. Therefore we have dropped the notion of reduction rules for inhabitants of proof-irrelevant types in $\mathsf{TT}^{\mathrm{obs}}$, and only prove normalization for proof-relevant types. We argue that this is more faithful to the philosophy of proof irrelevance, and results in a proof that is completely agnostic about the proof irrelevant content of the theory—we could postulate any consistent proof-irrelevant axiom and the normalization proof would carry through.

Of course, this means that consistency and canonicity do not follow from normalization anymore. The reason is simple: since we gave up any kind of control on the proof-irrelevant terms, we are forced to consider them as neutral. This also adds neutrals to the proof-*relevant* world, as $\bot$-elim can build inhabitants of proof-relevant types from inhabitants of $\bot$. Therefore, we need consistency of $\mathsf{TT}^{\mathrm{obs}}$ to show canonicity.

3.2 Definition of the Logical Relation

In this section, we define the (Kripke) logical relation that we used *[LogicalRelation.agda]*, closely following Abel et al. [2018]. For pedagogical reasons, we will present a somewhat informal version of the logical relation, and refer the interested reader to the formalization.

The logical relation is indexed by a level ℓ, which reflects the predicative nature of the universe hierarchies: we first define reducibility at level 0 to characterize judgments that do not mention

any universe, then we use this relation to define reducibility at level 1 for judgments that mention $\mathcal{U}_0$ at most, and so on. Therefore, the whole definition is done by induction on ℓ. While our formal proof only features three levels, we describe a full proof with levels that range over $\mathbb{N}$ as this does not add significant complexity to the argument. The logical relation features four kinds of judgments presented in Fig. 6. As is customary in reducibility proofs [Girard 1972], these judgments imply normalization, are closed under weak head expansion, and are verified by all neutral terms. The judgment $\Gamma \Vdash_\ell A$ is defined inductively, while the three others are simultaneously defined by recursion on a derivation of $\Gamma \Vdash_\ell A$. Thus, these judgments technically depend on the specific derivation of $\Gamma \Vdash_\ell A$, but since two different derivations will give rise to equivalent judgments we freely ignore this dependence.

We now present the six cases in the inductive definition $\Gamma \Vdash_\ell A$, together with the definition of the other three judgments.

Neutral Types.

$$\frac{\Gamma \vdash A \Rightarrow^* N : s_i \qquad \text{neutral}\, N}{\Gamma \Vdash_\ell A}$$

This rule states that A is reducible to a neutral type: $\Gamma \vdash A \Rightarrow^* N : s_i$ means that A reduces to a neutral term N in a finite number of steps (possibly zero), and that both A and N are of type s_i in context Γ. When A is reducible to a neutral type, we define:

- $\Gamma \Vdash_\ell A \equiv B$ if there is a neutral term M such that $\Gamma \vdash B \Rightarrow^* M : s_i$ and $\Gamma \vdash N \equiv M : s_i$.

If A is a proof-relevant type, then we also define:

- $\Gamma \Vdash_\ell t : A$ if there is a neutral term n such that $\Gamma \vdash t \Rightarrow^* n : N$.
- $\Gamma \Vdash_\ell t \equiv u : A$ if there are neutral terms n, m such that $\Gamma \vdash t \Rightarrow^* n : N$ and $\Gamma \vdash u \Rightarrow^* m : N$, and $\Gamma \vdash n \equiv m : N$.

If A is a proof-irrelevant type, then we define instead:

- $\Gamma \Vdash_\ell t : A$ if $\Gamma \vdash t : A$.
- $\Gamma \Vdash_\ell t \equiv u : A$ if $\Gamma \vdash t : A$ and $\Gamma \vdash u : A$.

More generally, inhabitants of proof-irrelevant types are always reducible when they are well-typed.

Universes.

$$\frac{\Gamma \vdash A \Rightarrow^* s_i : \mathcal{U}_j \;{}_{i<j}^{i<\ell}}{\Gamma \Vdash_\ell A}$$

When A is reducible to a universe, we define:

- $\Gamma \Vdash_\ell A \equiv B$ if $\Gamma \vdash B \Rightarrow^* s_i : \mathcal{U}_j$.
- $\Gamma \Vdash_\ell t : A$ if there is a normal form t' such that $\Gamma \vdash t \Rightarrow^* t' : s_i$, and $\Gamma \Vdash_i t$ (which is already defined by induction hypothesis, since $i < \ell$).
- $\Gamma \Vdash_\ell t \equiv u : A$ if there are normal forms t', u' such that $\Gamma \vdash t \Rightarrow^* t' : s_i$ and $\Gamma \vdash u \Rightarrow^* u' : s_i$, and $\Gamma \Vdash_i t$, $\Gamma \Vdash_i u$, and $\Gamma \Vdash_i t \equiv u$.

Natural Numbers.

$$\frac{\Gamma \vdash A \Rightarrow^* \mathbb{N} : \mathcal{U}_i}{\Gamma \Vdash_\ell A}$$

When A is reducible to $\mathbb{N}$, we define:

- $\Gamma \Vdash_\ell A \equiv B$ if $\Gamma \vdash B \Rightarrow^* \mathbb{N} : \mathcal{U}_i$.

- $\Gamma \Vdash_\ell t : A$ if there is a normal form t' such that $\Gamma \vdash t \Rightarrow^* t' : \mathbb{N}$ and $\Gamma \Vdash_{\mathbb{N}} t'$, which is inductively defined by

$$\frac{}{\Gamma \Vdash_{\mathbb{N}} 0} \qquad \frac{\Gamma \Vdash_{\mathbb{N}} t}{\Gamma \Vdash_{\mathbb{N}} \mathsf{S}\, t} \qquad \frac{\Gamma \vdash n : \mathbb{N} \qquad n \text{ is neutral}}{\Gamma \Vdash_{\mathbb{N}} n}$$

- $\Gamma \Vdash_\ell t \equiv u : A$ if there are normal forms t', u' such that $\Gamma \vdash t \Rightarrow^* t' : \mathbb{N}$ and $\Gamma \vdash u \Rightarrow^* u' : \mathbb{N}$, and $\Gamma \Vdash_{\mathbb{N}} t' \equiv u'$, which is inductively defined by

$$\frac{}{\Gamma \Vdash_{\mathbb{N}} 0 \equiv 0} \qquad \frac{\Gamma \Vdash_{\mathbb{N}} t \equiv u}{\Gamma \Vdash_{\mathbb{N}} \mathsf{S}\, t \equiv \mathsf{S}\, u} \qquad \frac{\Gamma \vdash n \equiv m : \mathbb{N} \qquad n, m \text{ are neutral}}{\Gamma \Vdash_{\mathbb{N}} n \equiv m}$$

Dependent Function Types.

$$\frac{\begin{array}{c}\Gamma \vdash A \Rightarrow^* \Pi(x : F).\, G : s_k \\ \Gamma \vdash F : s'_i \qquad \Gamma, x : F \vdash G : s_j \qquad \forall \Delta\, \rho.\ \Delta \Vdash_\ell F[\rho] \qquad \forall \Delta\, \rho.\ \Delta \Vdash_\ell a : F[\rho] \implies \Delta \Vdash_\ell G[\rho, a] \\ \forall \Delta\, \rho.\ \Delta \Vdash_\ell a : F[\rho] \implies \Delta \Vdash_\ell b : F[\rho] \implies \Delta \Vdash_\ell a \equiv b : F[\rho] \implies \Delta \Vdash_\ell G[\rho, a] \equiv G[\rho, b]\end{array}}{\Gamma \Vdash_\ell A}\ \begin{array}{l} i \leq k \\ j \leq k \end{array}$$

This rule states that A is reducible to a dependent function type. We introduced quite a bit of notation here. $\implies$ is implication in the meta theory, *i.e.,* the theory of Agda. $\forall \Delta \rho$ is a meta-theoretical quantification on an arbitrary well-formed context Δ and a weakening ρ that turns Δ into Γ. Applying such a weakening on the free variables of a term F that is a well-typed in context Γ results in a term $F[\rho]$ that is well-typed in context Δ. Given a term G in the extended context $\Gamma, x : F$ and a term $\Delta \vdash a : F[\rho]$, we can apply ρ on the free variables of G except for x to get a term in $\Delta, x : F[\rho]$, and then substitute x with a to get a term in Δ. The result is noted $G[\rho, a]$.

When A is reducible to a dependent function type, we define:

- $\Gamma \Vdash_\ell A \equiv B$ if there are terms F' and G' such that
  - $\Gamma \vdash B \Rightarrow^* \Pi(x : F').\, G' : s_k$ and $\Gamma \vdash \Pi(x : F).\, G \equiv \Pi(x : F').\, G' : s_k$
  - $\forall \Delta\, \rho.\ \Delta \Vdash_\ell F[\rho] \equiv F'[\rho]$
  - $\forall \Delta\, \rho.\ \Delta \Vdash_\ell a : F[\rho] \implies \Delta \Vdash_\ell G[\rho, a] \equiv G'[\rho, a]$.

If A is a proof-relevant type, then we also define:

- $\Gamma \Vdash_\ell t : A$ if there is a normal form t' such that
  - $\Gamma \vdash t \Rightarrow^* t' : \Pi(x : F).\, G$
  - $\forall \Delta\, \rho.\ \Delta \Vdash_\ell a : F[\rho] \implies \Delta \Vdash_\ell t'[\rho]\ a : G[\rho, a]$
  - $\forall \Delta\, \rho.\ \Delta \Vdash_\ell a : F[\rho] \implies \Delta \Vdash_\ell b : F[\rho] \implies \Delta \Vdash_\ell a \equiv b : F[\rho] \implies \Delta \Vdash_\ell t'[\rho]\ a \equiv t'[\rho]\ b : G[\rho, a]$.
- $\Gamma \Vdash_\ell t \equiv u : A$ if there are normal forms t', u' such that
  - $\Gamma \vdash t \Rightarrow^* t' : \Pi(x : F).\, G$ and $\Gamma \vdash u \Rightarrow^* u' : \Pi(x : F).\, G$
  - $\Gamma \vdash t' \equiv u' : \Pi(x : F).\, G$
  - $\Gamma \Vdash_\ell t : A$ and $\Gamma \Vdash_\ell u : A$
  - $\forall \Delta\, \rho.\ \Delta \Vdash_\ell a : F[\rho] \implies \Delta \Vdash_\ell t'[\rho]\ a \equiv u'[\rho]\ a : G[\rho, a]$.

If A is a proof-irrelevant type, then we define instead:

- $\Gamma \Vdash_\ell t : A$ if $\Gamma \vdash t : \Pi(x : F).\, G$.
- $\Gamma \Vdash_\ell t \equiv u : A$ if $\Gamma \vdash t : \Pi(x : F).\, G$ and $\Gamma \vdash u : \Pi(x : F).\, G$.

Existential types. The definition of a type A reducible to an existential type is similar to those for dependent function types, and the same holds for the notion of reducibly equal types. However, the definition of judgments for terms is much simpler: as existential types are proof-irrelevant we only ask them to be well-typed.

- $\Gamma \Vdash_\ell A \equiv B$ if there are terms F' and G' such that

  - $\Gamma \vdash B \Rightarrow^* \exists(x : F').\, G' : \Omega_k$ and $\Gamma \vdash \exists(x : F).\, G \equiv \exists(x : F').\, G' : \Omega_k$
  - $\forall \Delta\, \rho.\ \Delta \Vdash_\ell F[\rho] \equiv F'[\rho]$
  - $\forall \Delta\, \rho.\ \Delta \Vdash_\ell a : F[\rho] \implies \Delta \Vdash_\ell G[\rho, a] \equiv G'[\rho, a]$.
- $\Gamma \Vdash_\ell t : A$ if $\Gamma \vdash t : \exists(x : F).\, G$.
- $\Gamma \Vdash_\ell t \equiv u : A$ if $\Gamma \vdash t : \exists(x : F).\, G$ and $\Gamma \vdash u : \exists(x : F).\, G$.

Empty type.

$$\frac{\Gamma \vdash A \Rightarrow^* \bot : \Omega_i}{\Gamma \Vdash_\ell A}$$

When A is reducible to the empty type, we define:

- $\Gamma \Vdash_\ell A \equiv B$ if $\Gamma \vdash B \Rightarrow^* \bot : \Omega_i$.
- $\Gamma \Vdash_\ell t : A$ if $\Gamma \vdash t : \bot$.
- $\Gamma \Vdash_\ell t \equiv u : A$ if $\Gamma \vdash t : \bot$ and $\Gamma \vdash u : \bot$.

We do not have to add a case for $\top$, since it can be encoded as $\bot \to \bot$. There is no case for $t \sim_A u$ either, because setoid equality is not a type constructor, but rather a type eliminator that does pattern matching on A and its two endpoints t and u. Consequently, the only whnfs of the form $t \sim_A u$ are neutral, and are handled by the rule for proof-irrelevant neutral types.

Embedding.

$$\frac{\Gamma \Vdash_\ell A}{\Gamma \Vdash_{\ell'} A}\ell<\ell'$$

This case ensures that the logical relation is monotonic with respect to the level.

3.3 The fundamental lemma

Before stating the fundamental lemma for the logical relation, which basically amounts to completeness of the logical relation with respect to the typing relation, we establish the correctness of logical relation, which is called the escape lemma by Abel et al. [2018].

Lemma 3.1 (Escape lemma). *Given* $\Gamma \Vdash_\ell A$,

(1) $\Gamma \vdash A : s$.
(2) *If* $\Gamma \Vdash_\ell A \equiv B$ *then* $\Gamma \vdash A \equiv B : s$.
(3) *If* $\Gamma \Vdash_\ell t : A$ *then* $\Gamma \vdash A : s$.
(4) *If* $\Gamma \Vdash_\ell t \equiv u : A$ *then* $\Gamma \vdash t \equiv u : A$.

Proof. Straightforward induction on the logical relation. □

We can now state the completeness of the logical relation, a.k.a. the fundamental lemma *[Fundamental.agda]*. We state it here in a simple form but the proof, done by induction on the typing derivation, requires us to generalize the induction hypothesis to consider reducibility under an arbitrary reducible substitution. This is achieved, in [Abel et al. 2018] as well as in our formalization *[Substitution.agda]*, by defining the notion of validity, but we deliberately ignore this technicality in the rest of the section.

Lemma 3.2 (Fundamental lemma).

(1) *If* $\Gamma \vdash t : A$, *then there is a* ℓ *such that* $\Gamma \Vdash_\ell A$ *and* $\Gamma \Vdash_\ell t : A$.
(2) *If* $\Gamma \vdash t \equiv u : A$, *then there is a* ℓ *such that* $\Gamma \Vdash_\ell A$ *and* $\Gamma \Vdash_\ell t \equiv u : A$.

The fundamental lemma is proved by induction on the typing derivations. For most of the rules, the proof is roughly the same as in [Abel et al. 2018; Gilbert et al. 2019]—with simplified proofs for inhabitants of proof-irrelevant types. The proof relies heavily on reflexivity, symmetry and

transitivity of the reducible equality, stability of reducibility under reducible type conversion, reducibility of neutral terms, and the fact that reducibility is stable by weak-head expansion. Our extension to a cumulative hierarchy of universes and to proof-irrelevant existential quantification requires numerous changes, but these do not pose any major difficulty—albeit the proof is by no means trivial, as the arguments have to be spelled out in excruciating detail. We now focus on the two main typing derivation introduced by $\mathsf{TT}^{\mathrm{obs}}$, Rules Eq-Form and Cast. Those two cases are the consequence of the following lemmas.

Lemma 3.3 (Reducibility of cast). *For all levels $\ell_1, \ell_2, \ell_3, \ell_4$, in a well formed context Γ, if:*

(1) $\Gamma \Vdash_{\ell_1} A$ *and* $\Gamma \Vdash_{\ell_2} A'$ *and* $\Gamma \Vdash_{\ell_1} A \equiv A'$
(2) $\Gamma \Vdash_{\ell_3} B$ *and* $\Gamma \Vdash_{\ell_4} B'$ *and* $\Gamma \Vdash_{\ell_3} B \equiv B'$
(3) $\Gamma \vdash e : A \sim_s B$ *and* $\Gamma \vdash e' : A' \sim_s B'$
(4) $\Gamma \Vdash_{\ell_1} t : A$ *and* $\Gamma \Vdash_{\ell_2} t' : A'$ *and* $\Gamma \Vdash_{\ell_1} t \equiv t' : A$

then $\Gamma \Vdash_{\ell_3} \mathrm{cast}(A, B, e, t) : B$ *and* $\Gamma \Vdash_{\ell_3} \mathrm{cast}(A, B, e, t) \equiv \mathrm{cast}(A', B', e', t') : B$.

Proof. The proof is by case analysis on the reducibility proofs of A, A', B, B'. If one of these proofs is introduced by embedding, then we recursively apply the lemma on the embedded proof. Thus, it suffices to prove the lemma when all reducibility proofs correspond to a normal form. From the proofs of reducible equality, one obtains that the reducibility proofs of A and A' (resp. B and B') are introduced by the same rule. Then, if one of the normal forms is neutral, or if the normal forms of A and B have different head constructors, then $\mathrm{cast}(A, B, e, t)$ and $\mathrm{cast}(A', B', e', t')$ are neutral and easily seen to be reducible. Therefore, it suffices to prove the lemma when all the reducibility proofs are introduced by the same rule. We now detail the case of dependent products, as it is the most interesting.

We know that A reduces to a term of the form $\Pi(x : F_A).\, G_A$, as do A', B, B'. Therefore, we know that $\mathrm{cast}(A, B, e, t)$ reduces to $\lambda(a' : F_B).\, \mathrm{cast}(G_A[x := a], G_B[x := a'], \mathrm{snd}(e)\ a', t\ a)$ where a is a shorthand for $\mathrm{cast}(F_B, F_A, \mathrm{fst}(e)^{-1}, a')$, and $\mathrm{cast}(A', B', e', t')$ reduces similarly. By the weak head expansion lemma, it suffices to prove that these normal forms are reducible, and reducibly equal at type $\Pi(x : F_B).\, G_B$—that is, applying them to a reducible term a' under any weakening results in reducible terms that are reducibly equal, and applying the first to two reducibly equal terms produce new reducibly equal terms.

To prove the first obligation, we first recursively apply the lemma to $\mathrm{cast}(F_B, F_A, \mathrm{fst}(e)^{-1}, a')$ so that we obtain reducibility of a, and then recursively apply it to $\mathrm{cast}(G_A[x := a], G_B[x := a'], \mathrm{snd}(e)\ a', t\ a)$. The other two obligations are proved in the exact same manner. □

Lemma 3.4 (Reducibility of Id in the universe). *For all levels $\ell_1, \ell_2, \ell_3, \ell_4$, in a well formed context Γ, if:*

(1) $\Gamma \Vdash_{\ell_1} A$ *and* $\Gamma \Vdash_{\ell_2} A'$ *and* $\Gamma \Vdash_{\ell_1} A \equiv A'$
(2) $\Gamma \Vdash_{\ell_3} B$ *and* $\Gamma \Vdash_{\ell_4} B'$ *and* $\Gamma \Vdash_{\ell_3} B \equiv B'$

then $\Gamma \Vdash_{\ell_{\max}} A \sim_s B$ *and* $\Gamma \Vdash_{\ell_{\max}} A \sim_s B \equiv A' \sim_s B'$ *where* $\ell_{\max} = \max(\ell_1, \ell_2, \ell_3, \ell_4)$.

Proof. The proof is by case analysis on the reducibility proofs of A, A', B, B'. As in the proof of Lemma 3.3, we reduce the proof to introduction rules that correspond to the same kind of normal form. Most cases are straightforward, the difficult case being again the dependent products.

From reducibility proofs, A reduces to a term of the form $\Pi(x : F_A).\, G_A$, and so for A', B and B'. Thus, we know that $A \sim_s B$ reduces to $\exists(e : F_A \sim_{\mathcal{U}} F_B).\, \Pi(a' : F_B).\, G_A[x := a] \sim_{\mathcal{U}} G_B[x := a']$ where a is a shorthand for $\mathrm{cast}(F_B, F_A, \mathrm{fst}(e)^{-1}, a')$, and $A' \sim_s B'$ reduces similarly. By the weak head expansion lemma, it suffices to prove that the first normal form is reducible, and reducibly equal to the second one. We do this by applying reducibility of cast to get reducibility of a, and

then doing recursive calls on $F_A \sim_{\mathcal{U}} F_B$ and $G_A[x := a] \sim_{\mathcal{U}} G_B[x := a']$ under the appropriate weakenings and substitutions. □

Lemma 3.5 (Reducibility of Id). *For all levels ℓ_1, ℓ_2, in a well formed context Γ, if:*

(1) $\Gamma \Vdash_{\ell_1} A$ *and* $\Gamma \Vdash_{\ell_2} A'$ *and* $\Gamma \Vdash_{\ell_1} A \equiv A'$
(2) $\Gamma \Vdash_{\ell_1} t : A$ *and* $\Gamma \Vdash_{\ell_2} t' : A'$ *and* $\Gamma \Vdash_{\ell_1} t \equiv t' : A$
(3) $\Gamma \Vdash_{\ell_1} u : A$ *and* $\Gamma \Vdash_{\ell_2} u' : A'$ *and* $\Gamma \Vdash_{\ell_1} u \equiv u' : A$

then $\Gamma \Vdash_{\ell_1} t \sim_A u$ *and* $\Gamma \Vdash_{\ell_1} t \sim_A u \equiv t' \sim_A u'$.

Proof. By case analysis on A and A'. The most difficult case is the universe, and is handled by Lemma 3.4. The case of dependent products requires doing recursive calls on the domain and the codomain, as in the previous lemmas. □

These lemmas constitute the bulk of the proof: they occupy approximately 5,000 lines of Agda code and most of the time required to check the whole proof is spent on them.

Corollary 3.6 (Normalization). *A direct consequence of the fundamental lemma is that any well-typed term has a weak-head normal form.*

Another direct consequence of the fundamental lemma is that any closed term of type $\mathbb{N}$ reduces to a whnf of type $\mathbb{N}$. Thus, to conclude canonicity, we just need to know that there is no neutral term of type $\mathbb{N}$ in an empty context. Unfortunately, as the notion of reduction is absent for terms of proof-irrelevant types, and in particular for the empty type, this cannot be proven by induction on the syntax of the neutral terms. However, assuming consistency of $\mathsf{TT}^{\mathsf{obs}}$ (proved in Section 3.5), or in other words that there is no closed term of type $\bot$, we can derive canonicity of $\mathsf{TT}^{\mathsf{obs}}$ *[Canonicity.agda]*.

3.4 Decidability of typing

We first need to show that the conversion judgment is decidable. This is done by defining an algorithmic version of the conversion of two terms t and u which basically amounts to computing the whnf of t and u, comparing their head, and applying the algorithm recursively if necessary *[Conversion.agda]*. Correctness of this algorithmic conversion is easily obtained, as the rules used are special cases of the conversion judgement *[Soundness.agda]*. Then, we can show that algorithmic conversion is also complete by replaying the fundamental lemma with a definition of the logical relation that uses algorithmic conversion instead of typed conversion *[Completeness.agda]*. To do so, the main difficulty is to show that algorithmic conversion is reflexive, symmetric and transitive. In our formal proof, we follow Abel et al. [2018] in factoring the two instances of the fundamental lemma by defininig a generic interface for both algorithmic conversion and typed conversion, and using this interface in the definition of the logical relation *[EqualityRelation.agda]*.

Then, to get decidability of type checking, we can simply rely on the work of Lennon-Bertrand [2021] on bidirectional type-checking, which defines an algorithmic version of type-checking provided that the theory enjoys subject reduction and decidability of conversion.

3.5 Consistency: The Setoid Model

$\mathsf{TT}^{\mathsf{obs}}$ is designed to describe the behavior of constructive setoids. In order to justify this claim, we build a model of $\mathsf{TT}^{\mathsf{obs}}$ where every context and proof-relevant type is interpreted as a setoid in a constructive set theory with induction-recursion [Aczel 1978], following the seminal proof of Hofmann [1995]. Then, by interpreting $\mathsf{TT}^{\mathsf{obs}}$ in this model, we can prove the following theorem, which is the central missing piece of the metatheory of $\mathsf{TT}^{\mathsf{obs}}$:

Theorem 3.7 (Consistency of $\mathsf{TT}^{\mathsf{obs}}$). *There is no inhabitant of $\bot$ in the empty context.*

We work in a constructive set theory that supports inductive-recursive definitions and one Grothendieck universe. While it would be more satisfying to use a constructive type theory, as in the normalization proof, it is significantly more challenging. This is because the interpretation of the syntax in our model is only defined on well-typed terms, but should not depend on the typing derivation. While properly establishing this in a typed context is non-trivial, the added flexibility of set theory makes the task straightforward. We will write N for the set of natural numbers in our meta-theory, $(x \in A) \to B(x)$ for dependent functions and $(x \in A) \times B(x)$ for dependent products.

We define a *setoid* to be a *carrier* set A and a *setoid equality* predicate on $A \times A$ that is reflexive, symmetric and transitive, which we write $\approx$. Alternatively, we can define setoids to be small categories such that morphism sets are subsingletons, and every morphism is invertible. We will make use of both perspectives, and use the notation $a \approx b$ to mean either the proposition or the corresponding subsingleton. Given two setoids A and B, we write $A \to_s B$ for the set of functions that preserve setoid equality. This set inherits a natural setoid structure from pointwise setoid equality. $A \leftrightarrow_s B$ denotes the set of setoid equivalences, which are pairs of setoid functions in both directions that cancel. Given $e \in A \leftrightarrow_s B$, the induced function from A to B is simply noted e, while the other direction is noted e^{-1}.

The setoid equivalence turns the (large) set of all small setoids Setoid into a *groupoid*, or in other words a category where all morphisms are invertible. Given any setoid A, this allows us to define dependent setoids as functions from A to Setoid that send setoidal equalities to setoid equivalences, in a way that is compatible with composition and identity. Then, given a setoid A and a dependent setoid $B : A \to_s \mathsf{Setoid}$, we write $(x \in A) \to_s B(x)$ for the setoid of dependent functions, with pointwise equality. We also form a setoid Prop by using the set of subsingletons as the carrier, with logical equivalence as the equality.

The central object in the setoid model is a dual hierarchy of small setoids $\mathbf{U}_i$ and $\mathbf{\Omega}_i$, which contain codes for the setoids and propositions that can be constructed in $\mathsf{TT}^{\mathrm{obs}}$. We build these universes of codes using induction-recursion: we define the sets $\mathbf{U}_i$ and $\mathbf{\Omega}_i$ by induction, and we simultaneously use recursion on them to define interpretation functions $\mathsf{el} : \mathbf{U}_i \to \mathsf{Setoid}$ and $\mathsf{val} : \mathbf{\Omega}_i \to \mathsf{Prop}$, setoid equality on $\mathbf{U}_i$ and $\mathbf{\Omega}_i$, as well as proofs that el and val preserve that setoid equality. All of this is done under an external induction on i and is given in Fig. 7.

We have omitted quite a few bureaucratic proofs from this definition, such as the reflexivity, symmetry and transitivity of $\approx$, or the fact that el-eq is compatible with reflexivity and transitivity. We also freely make use of the fact that equality is decidable on integers.

With this, we obtain two families of setoids $\mathbf{U}_i$ and $\mathbf{\Omega}_i$, along with a dependent setoid $\mathsf{el} : \mathbf{U}_i \to_s \mathsf{Setoid}$ and a setoid function $\mathsf{val} : \mathbf{\Omega}_i \to_s \mathsf{Prop}$. This construction turns the category of setoids and setoid functions into a *universe category* in the sense of Voevodsky [2015]. Our model is a contextual category/C-system built from this universe category. In the following, we describe the construction of the setoid model without assuming familiarity with contextual categories.

We now use $\mathbf{U}_i$ and $\mathbf{\Omega}_i$ to define the types and the terms of the setoid model. From now on, we will write $\mathbf{s}$ to mean either $\mathbf{U}$ or $\mathbf{\Omega}$ and we will also write c_Π for any of $\mathsf{c}_{\Pi\Omega\Omega}$, $\mathsf{c}_{\Pi\Omega\mathbf{U}}$, $\mathsf{c}_{\Pi\mathbf{U}\Omega}$ and $\mathsf{c}_{\Pi\mathbf{U}\mathbf{U}}$. Given a setoid Γ, a *semantic type* of level i over Γ is simply a setoid function $\Gamma \to_s \mathbf{s}_i$. We write these sets as $\mathsf{Ty}_{s,i}\ \Gamma$. Then, given a semantic type A over Γ, we define the set of *semantic terms* of type A to be the setoidal dependent product $(x \in \Gamma) \to_s \mathsf{el}\ A(x)$. We note this set $\mathsf{tm}\ A$.

The contexts of our model are setoids which are inductively built from the terminal setoid as sequences of types:

$$
\begin{array}{lcll}
\mathsf{Con} & ::= & \bullet : \mathsf{Con} & \\
& | & \_,\_ : (\Gamma \in \mathsf{Con}) \to \mathsf{Ty}_{\mathcal{U},i}\ \Gamma \to \mathsf{Con} & (\forall i \in \mathsf{N}) \\
& | & \_,\_ : (\Gamma \in \mathsf{Con}) \to \mathsf{Ty}_{\Omega,i}\ \Gamma \to \mathsf{Con} & (\forall i \in \mathsf{N})
\end{array}
$$

$$
\begin{array}{lll}
\mathbf{U}_i & ::= & \mathsf{c}_{\mathsf{N}} : \mathbf{U}_i \\
& \mid & \mathsf{c}_{\Pi\mathsf{U}\mathsf{U}} : (A \in \mathbf{U}_j) \to (\mathsf{el}\ A \to_s \mathbf{U}_k) \to \mathbf{U}_i \qquad \text{where } j,k < i \\
& \mid & \mathsf{c}_{\Pi\Omega\mathsf{U}} : (A \in \mathbf{\Omega}_j) \to (\mathsf{el}\ A \to_s \mathbf{U}_k) \to \mathbf{U}_i \qquad \text{where } j,k < i \\
& \mid & \mathsf{c}_{\mathsf{U}} : \{j \in \mathsf{N} \mid j < i\} \to \mathbf{U}_i \\
& \mid & \mathsf{c}_{\Omega} : \{j \in \mathsf{N} \mid j < i\} \to \mathbf{U}_i \\
\mathbf{\Omega}_i & ::= & \mathsf{c}_{\bot} : \mathbf{\Omega}_i \\
& \mid & \mathsf{c}_{\Pi\mathsf{U}\Omega} : (A \in \mathbf{U}_j) \to (\mathsf{el}\ A \to_s \mathbf{\Omega}_k) \to \mathbf{\Omega}_i \qquad \text{where } j,k < i \\
& \mid & \mathsf{c}_{\Pi\Omega\Omega} : (A \in \mathbf{\Omega}_j) \to (\mathsf{el}\ A \to_s \mathbf{\Omega}_k) \to \mathbf{\Omega}_i \qquad \text{where } j,k < i \\
& \mid & \mathsf{c}_{\exists} : (A \in \mathbf{\Omega}_i) \to (\mathsf{el}\ A \to_s \mathbf{\Omega}_i) \to \mathbf{\Omega}_i
\end{array}
$$

$$
\begin{array}{l}
\mathsf{el} : \mathbf{U}_i \to \mathsf{Setoid} \\
\begin{array}{lcl}
\mathsf{el}\ \mathsf{c}_{\mathsf{N}} & := & \mathsf{N} \\
\mathsf{el}\ (\mathsf{c}_{\Pi\mathsf{U}\mathsf{U}}\ A\ B) & := & (x \in \mathsf{el}\ A) \to_s \mathsf{el}\ B(x) \\
\mathsf{el}\ (\mathsf{c}_{\Pi\Omega\mathsf{U}}\ A\ B) & := & (x \in \mathsf{val}\ A) \to \mathsf{el}\ B(x) \\
\mathsf{el}\ (\mathsf{c}_{\mathsf{U}}\ j) & := & \mathbf{U}_j \\
\mathsf{el}\ (\mathsf{c}_{\Omega}\ j) & := & \mathbf{\Omega}_j
\end{array}
\qquad
\begin{array}{l}
\mathsf{val} : \mathbf{\Omega}_i \to \mathsf{Prop} \\
\begin{array}{lcl}
\mathsf{val}\ \mathsf{c}_{\bot} & := & \bot \\
\mathsf{val}\ (\mathsf{c}_{\Pi\mathsf{U}\Omega}\ A\ B) & := & (x \in \mathsf{el}\ A) \to_s \mathsf{val}\ B(x) \\
\mathsf{val}\ (\mathsf{c}_{\Pi\Omega\Omega}\ A\ B) & := & (x \in \mathsf{val}\ A) \to \mathsf{val}\ B(x) \\
\mathsf{val}\ (\mathsf{c}_{\exists}\ A\ B) & := & (x \in \mathsf{val}\ A) \times \mathsf{val}\ B(x)
\end{array}
\end{array}
\end{array}
$$

$$
\begin{array}{lcll}
\_ \approx \_ : \mathbf{U}_i \to \mathbf{U}_i \to \mathsf{Prop} & & & \\
\mathsf{c}_{\mathsf{N}} \approx \mathsf{c}_{\mathsf{N}} & := & \top & \\
\mathsf{c}_{\Pi\mathsf{U}\mathsf{U}}\ A\ B \approx \mathsf{c}_{\Pi\mathsf{U}\mathsf{U}}\ A'\ B' & := & (e \in A \approx A') \times ((x \in \mathsf{el}\ A) \to_s B(x) \approx B'(\mathsf{el\text{-}eq}\ e\ x)) & \\
 & & & \text{when } j = j' \text{ and } k = k' \\
\mathsf{c}_{\Pi\Omega\mathsf{U}}\ A\ B \approx \mathsf{c}_{\Pi\Omega\mathsf{U}}\ A'\ B' & := & (e \in A \approx A') \times ((x \in \mathsf{val}\ A) \to B(x) \approx B'(\mathsf{val\text{-}eq}\ e\ x)) & \\
 & & & \text{when } j = j' \text{ and } k = k' \\
\mathsf{c}_{\mathsf{U}}\ j \approx \mathsf{c}_{\mathsf{U}}\ j' & := & j = j' & \\
\mathsf{c}_{\Omega}\ j \approx \mathsf{c}_{\Omega}\ j' & := & j = j' & \\
\_ \approx \_ & := & \bot \qquad \text{otherwise} &
\end{array}
$$

$$
\begin{array}{l}
\_ \approx \_ : \mathbf{\Omega}_i \to \mathbf{\Omega}_i \to \mathsf{Prop} \\
A \approx B := \mathsf{val}\ A \Leftrightarrow \mathsf{val}\ B
\end{array}
$$

$$
\begin{array}{lcl}
\mathsf{el\text{-}eq} : (A \in \mathbf{U}_i) \to (B \in \mathbf{U}_i) \to A \approx B \to \mathsf{el}\ A \leftrightarrow_s \mathsf{el}\ B & & \\
\mathsf{el\text{-}eq}\ \mathsf{c}_{\mathsf{N}}\ \mathsf{c}_{\mathsf{N}}\ e & := & (x \mapsto x\,,\ x \mapsto x) \\
\mathsf{el\text{-}eq}\ (\mathsf{c}_{\Pi\mathsf{U}\mathsf{U}}\ A\ B)\ (\mathsf{c}_{\Pi\mathsf{U}\mathsf{U}}\ A'\ B')\ e & := & (f \mapsto (x \mapsto \mathsf{el\text{-}eq}\ (e.2\ (\mathsf{el\text{-}eq}^{-1}\ e.1\ x))\ (f\ (\mathsf{el\text{-}eq}^{-1}\ e.1\ x))) \\
 & & \quad , f \mapsto (x \mapsto \mathsf{el\text{-}eq}^{-1}\ (e.2\ (\mathsf{el\text{-}eq}\ e.1\ x))\ (f\ (\mathsf{el\text{-}eq}\ e.1\ x)))) \\
\mathsf{el\text{-}eq}\ (\mathsf{c}_{\Pi\Omega\mathsf{U}}\ A\ B)\ (\mathsf{c}_{\Pi\Omega\mathsf{U}}\ A'\ B')\ e & := & (f \mapsto (x \mapsto \mathsf{el\text{-}eq}\ (e.2\ (\mathsf{val\text{-}eq}^{-1}\ e.1\ x))\ (f\ (\mathsf{val\text{-}eq}^{-1}\ e.1\ x))) \\
 & & \quad , f \mapsto (x \mapsto \mathsf{el\text{-}eq}^{-1}\ (e.2\ (\mathsf{val\text{-}eq}\ e.1\ x))\ (f\ (\mathsf{val\text{-}eq}\ e.1\ x)))) \\
\mathsf{el\text{-}eq}\ (\mathsf{c}_{\mathsf{U}}\ j)\ (\mathsf{c}_{\mathsf{U}}\ j)\ e & := & (x \mapsto x\,,\ x \mapsto x) \\
\mathsf{el\text{-}eq}\ (\mathsf{c}_{\Omega}\ j)\ (\mathsf{c}_{\Omega}\ j)\ e & := & (x \mapsto x\,,\ x \mapsto x)
\end{array}
$$

$$
\begin{array}{l}
\mathsf{val\text{-}eq} : (A \in \mathbf{\Omega}_i) \to (B \in \mathbf{\Omega}_i) \to A \approx B \to \mathsf{val}\ A \Leftrightarrow \mathsf{val}\ B \\
\mathsf{val\text{-}eq}\ A\ B\ e\ :=\ e
\end{array}
$$

Fig. 7. The Setoid Model

Being telescopes of setoids, inhabitants of Con have an obvious interpretation as setoids: the elements of $\Gamma \in \mathsf{Con}$ are dependent lists of inhabitants for all of the types and propositions in Γ, and the setoid equality is inherited from pointwise setoid equality of lists.

We have all of the required blocks to get the structure of a category with families [Dybjer 1996] and thus a model of MLTT.

$$
\begin{array}{rcl}
[\![\mathcal{U}_i]\!]_\Gamma\ x & := & \mathsf{c}_\mathsf{U}\ i \\
[\![\Omega_i]\!]_\Gamma\ x & := & \mathsf{c}_\Omega\ i \\
[\![\Pi(y:F).\,G]\!]_\Gamma\ x & := & \mathsf{c}_\Pi\ ([\![F]\!]_\Gamma\ x)\ (y \mapsto [\![G]\!]_{\Gamma,y:F}\ x, y) \\
[\![\lambda(y:F).\,t]\!]_\Gamma\ x & := & y \mapsto ([\![t]\!]_{\Gamma,y:F}\ x, y) \\
[\![t\ u]\!]_\Gamma\ x & := & ([\![t]\!]_\Gamma\ x)\ ([\![u]\!]_\Gamma\ x) \\
[\![\mathbb{N}]\!]_\Gamma\ x & := & \mathsf{c}_\mathsf{N} \\
[\![0]\!]_\Gamma\ x & := & 0 \\
[\![\mathsf{S}\ t]\!]_\Gamma\ x & := & \mathsf{S}\ ([\![t]\!]_\Gamma\ x) \\
[\![\mathbb{N}\text{-elim}(P,t_0,t_S,n)]\!]_\Gamma\ x & := & \mathsf{N}\text{-elim}(\mathsf{el} \circ ([\![P]\!]_\Gamma\ x), [\![t_0]\!]_\Gamma\ x, [\![t_S]\!]_\Gamma\ x, [\![n]\!]_\Gamma\ x) \\
[\![\exists(y:F).\,G]\!]_\Gamma\ x & := & \mathsf{c}_\exists\ ([\![F]\!]_\Gamma\ x)\ (y \mapsto [\![G]\!]_{\Gamma,y:F}\ x, y) \\
[\![\langle t,u\rangle]\!]_\Gamma\ x & := & ([\![t]\!]_\Gamma\ x, [\![u]\!]_\Gamma\ x) \\
[\![\mathsf{fst}(t)]\!]_\Gamma\ x & := & ([\![t]\!]_\Gamma\ x).1 \\
[\![\mathsf{snd}(t)]\!]_\Gamma\ x & := & ([\![t]\!]_\Gamma\ x).2 \\
[\![\bot]\!]_\Gamma\ x & := & \mathsf{c}_\bot \\
[\![\bot\text{-elim}(A,t)]\!]_\Gamma\ x & := & \bot\text{-elim}(\mathsf{el}([\![A]\!]_\Gamma\ x), [\![t]\!]_\Gamma\ x) \\
[\![t \sim_A u]\!]_\Gamma\ x & := & [\![t]\!]_\Gamma\ x \approx [\![u]\!]_\Gamma\ x \qquad \text{in } \mathsf{el}([\![A]\!]_\Gamma\ x) \\
[\![\mathsf{refl}(t)]\!]_\Gamma\ x & := & e_\mathsf{refl} \\
[\![\mathsf{transp}(t,G,u,t',e)]\!]_\Gamma\ x & := & \mathsf{val\text{-}eq} \\
[\![\mathsf{cast}(A,B,e,t)]\!]_\Gamma\ x & := & \mathsf{el\text{-}eq}([\![A]\!]_\Gamma\ x, [\![B]\!]_\Gamma\ x, [\![e]\!]_\Gamma\ x, [\![t]\!]_\Gamma\ x) \\
[\![\mathsf{castrefl}(A,t)]\!]_\Gamma\ x & := & e_\mathsf{id}
\end{array}
$$

Fig. 8. Interpretation of $\mathsf{TT}^{\mathsf{obs}}$ in the Setoid Model

3.6 Interpreting $\mathsf{TT}^{\mathsf{obs}}$ in the Setoid Model

We now describe how to interpret the judgments of $\mathsf{TT}^{\mathsf{obs}}$ in the setoid model where: (i) a well-formedness judgment $\vdash \Gamma$ will be interpreted as a setoid in Con (ii) a type judgment $\Gamma \vdash A : s_i$ will be interpreted as a semantic type over the interpretation of Γ (iii) a typing judgment $\Gamma \vdash t : A$ will be interpreted as a semantic term of the corresponding semantic type (iv) a convertibility judgment $\Gamma \vdash t \equiv u : B$ will be interpreted as meta-theoretical equality of the interpretations of t and u. Since reduction is contained in convertibility, the model will necessarily interpret reduction as equality—it is not fine enough to distinguish the two notions.

Our interpretation is defined by induction on the syntax of the terms, following Hofmann [1993]. The context interpretation turns a syntactical context Γ into a setoid $[\![\Gamma]\!] \in \mathsf{Con}$. The type interpretation turns a syntactical context Γ and a syntactical term A into an element $[\![A]\!]_\Gamma \in \mathsf{tm}\ \mathbf{U}_i$ for some i. The term interpretation turns a syntactical context Γ and a syntactical term t of type A into an element $[\![t]\!]_\Gamma \in \mathsf{tm}\ (\mathsf{el} \circ [\![A]\!]_\Gamma)$.

The reader might notice that the type interpretation is really an instance of the term interpretation. This is to be expected, since $\mathsf{TT}^{\mathsf{obs}}$ features Russel-style universes, that do not separate types and terms. As we cannot hope for every syntactical term to have an interpretation, we define the interpretation as *partial* functions from the syntax to the model. In the definition by recursion, we consider that whenever a term reduces to an expression that does not make sense, the interpretation is not defined. It is possible to prove that the interpretation function is total when restricted to the well-typed terms afterwards.

The interpretation of contexts is given by $[\![\bullet]\!] := 1$ and $[\![\Gamma, x : A]\!] := [\![\Gamma]\!], \mathsf{el} \circ [\![A]\!]_\Gamma$ and variables are interpreted as projections:

$$
\begin{array}{rcl}
[\![x]\!]_{\Gamma,x:A}\ (x_1,\ldots,x_n) & := & x_n \\
[\![x]\!]_{\Gamma,y:A}\ (x_1,\ldots,x_n) & := & [\![x]\!]_\Gamma\ (x_1,\ldots,x_{n-1})
\end{array}
$$

Then, the interpretation of the basic theory is not too surprising. Since terms correspond to setoid functions, we define them by their behavior when applied to an arbitrary $x \in [\![\Gamma]\!]$. This is defined in the first part of Fig. 8. Finally, to interpret observational equality and its cast operation, we use the setoid structure of the universe: where e_{refl} is obtained from the reflexivity of setoid equality, and e_{id} is obtained from the behavior of el-eq on reflexivity.

In order to prove the soundness of our interpretation, we need to extend it to weakenings and substitutions between contexts: Assume Γ and Δ are a syntactical contexts, and A and t are syntactical terms. In case $[\![\Gamma, x : A, \Delta]\!]$ and $[\![\Gamma, \Delta]\!]$ are well-defined, let π_A be the projection:

$$\begin{aligned} \pi_A : [\![\Gamma, x : A, \Delta]\!] &\to_s [\![\Gamma, \Delta]\!] \\ (\vec{x_\Gamma}, x_A, \vec{x_\Delta}) &\mapsto (\vec{x_\Gamma}, \vec{x_\Delta}). \end{aligned}$$

In case $[\![\Gamma, \Delta[x := t]]\!]$ and $[\![\Gamma, x : A, \Delta]\!]$ are well-defined, we define the setoid function σ_t by:

$$\begin{aligned} \sigma_t : [\![\Gamma, \Delta[x := t]]\!] &\to_s [\![\Gamma, x : A, \Delta]\!] \\ (\vec{x_\Gamma}, \vec{x_\Delta}) &\mapsto (\vec{x_\Gamma}, [\![t]\!]_\Gamma, \vec{x_\Delta}). \end{aligned}$$

Lemma 3.8 (Weakening). *π_A is the semantic counterpart to the weakening of A: for all terms u, when both sides are well defined, we have:*

$$[\![u]\!]_{\Gamma, x:A, \Delta} = [\![u]\!]_{\Gamma,\Delta} \circ \pi_A$$

Lemma 3.9 (Substitution). *σ_t is the semantic counterpart to the substitution by t: for all terms u, when both sides are well defined, we have:*

$$[\![u[x := t]]\!]_{\Gamma,\Delta[x:=t]} = [\![u]\!]_{\Gamma,x:A,\Delta} \circ \sigma_t$$

Theorem 3.10 (Soundness of the Setoid Model).

(1) *If $\vdash \Gamma$ then $[\![\Gamma]\!] \in \mathsf{Con}$.*
(2) *If $\Gamma \vdash A : s_i$ then $\mathsf{el} \circ [\![A]\!]_\Gamma \in \mathsf{Ty}_{s,i}\ \Gamma$.*
(3) *If $\Gamma \vdash t : A$ then $[\![t]\!]_\Gamma \in \mathsf{tm}\ (\mathsf{el} \circ [\![A]\!]_\Gamma)$.*
(4) *If $\Gamma \vdash t \equiv u : A$ then $[\![t]\!]_\Gamma = [\![u]\!]_\Gamma$*

Proof. By induction on the typing derivations. □

Consistency of TT^{obs} (Theorem 3.7) follows immediately from the soundness theorem: any inhabitant of $\bot$ in the empty context is interpreted as a setoid function from the one-element setoid to the empty setoid, but no such function exists.

4 EXTENSIONS TO QUOTIENTS, ID TYPES AND INDUCTIVE TYPES

So far, we have defined and studied a minimal version of TT^{obs}. In this section, we consider several extensions of the theory. In regular MLTT, adding a type generally means giving rules for type formation, introduction, elimination and computation of the eliminator. In our case, we also need to provide computation rules for equality of types, equality of terms, and cast. These extensions have not been formalized in the companion Agda development.

4.1 Quotient Types

Quotients are a ubiquitous construction in mathematics, and one that is famously difficult to handle smoothly in MLTT. The usual way to handle quotients is *via* setoids, but since this structure is not built in MLTT, all the functions between setoids, all the predicates, etc... have to be supplemented with equality preservation lemmas—which appears to be quickly unmanageable.

In $\mathsf{TT}^{\mathsf{obs}}$ however, every type is already a setoid and every term preserves the setoid equality by construction. This is a very comfortable setting for quotients, that can thus be added to the theory—provided the relation that induces the quotient is proof-irrelevant. This can be seen as a limitation, as noticed by Sterling et al. [2019], as it is generally impossible to extract proof-relevant information from equality in the quotient type. This is in contrast with the development of higher inductive types in the cubical setting [Coquand et al. 2018]. On the other hand, the positive consequence of this limitation is that the elimination principle of the quotient types is fairly easy to manipulate in $\mathsf{TT}^{\mathsf{obs}}$.

Quotient types are defined on a type A equipped with an equivalence relation on A

$$\frac{\Gamma \vdash A : \mathcal{U}_i \qquad \Gamma \vdash R : A \to A \to \Omega \qquad \Gamma \vdash R_r : \Pi(x : A).\, R\ x\ x \qquad \Gamma \vdash R_s : \Pi(x, y : A).\, R\ x\ y \to R\ y\ x \qquad \Gamma \vdash R_t : \Pi(x, y, z : A).\, R\ x\ y \to R\ y\ z \to R\ x\ z}{\Gamma \vdash A/(R, R_r, R_s, R_t) : \mathcal{U}_i}$$

Since the proofs of reflexivity, symmetry and transitivity appear everywhere but are proof-irrelevant, we will generally omit them in the assumptions of the rules, and write A/R instead of $A/(R, R_r, R_s, R_t)$. The only constructor of quotient types is the canonical projection: from an element t of A, one obtains an element $\pi(t)$ of A/R that is whnf, and equality between two canonical projections reduces to R.

$$\frac{\Gamma \vdash t : A}{\Gamma \vdash \pi(t) : A/R} \qquad \frac{\Gamma \vdash t : A \qquad \Gamma \vdash u : A}{\Gamma \vdash \pi(t) \sim_{A/R} \pi(u) \Rightarrow R\ t\ u : \Omega_i}$$

The definition of cast between two quotient types reduces when the casted term is a canonical projection.

$$\frac{\Gamma \vdash e : A/R \sim_{\mathcal{U}} A'/R' \qquad \Gamma \vdash t : A}{\Gamma \vdash \mathsf{cast}(A/R, A'/R', e, \pi(t)) \Rightarrow \pi(\mathsf{cast}(A, A', \mathsf{fst}(e), t)) : A/R'}$$

Observational equality between two quotient types reduces to equality of the (proof-relevant part of the) telescopes that define each quotient:

$$\frac{\Gamma \vdash A : \mathcal{U}_i \qquad \Gamma \vdash R : A \to A \to \Omega \qquad \Gamma \vdash A' : \mathcal{U}_i \qquad \Gamma \vdash R' : A' \to A' \to \Omega}{\Gamma \vdash A/R \sim_{\mathcal{U}} A'/R' \Rightarrow \exists(e : A \sim_{\mathcal{U}} A').\, \Pi(x\ y : A).\, R\ x\ y \sim_{\Omega} R\ \mathsf{cast}(A, A', e, x)\ \mathsf{cast}(A, A', e, y) : \Omega_i}$$

The eliminator for quotient types encodes the universal property of quotients: to construct a function out of a quotient A/R, it suffices to give a function t_π out of A such that if $R\ x\ y$, then their images under t_π are equal.

$$\frac{\Gamma \vdash B : A/R \to s_i \qquad \Gamma \vdash t_\pi : \Pi(x : A).\, B\ \pi(x) \qquad \Gamma \vdash t_\sim : \Pi(x, y : A).\, \Pi(e : R\ x\ y).\, (t_\pi\ x) \sim_{B\ \pi(x)} \mathsf{cast}(B\ \pi(y), B\ \pi(x), B\ e^{-1}, t_\pi\ y) \qquad \Gamma \vdash u : A/R}{\Gamma \vdash \mathsf{Q\text{–}elim}(B, t_\pi, t_\sim, u) : B\ u}$$

The eliminator for quotient types has the obvious computation rule

$$\frac{\Gamma \vdash B : A/R \to s_i \qquad \Gamma \vdash t_\pi : \Pi(x : A).\, B\ \pi(x) \qquad \Gamma \vdash t_\sim : \Pi(x, y : A).\, \Pi(e : R\ x\ y).\, (t_\pi\ x) \sim_{B\ \pi(x)} \mathsf{cast}(B\ \pi(y), B\ \pi(x), B\ e^{-1}, t_\pi\ y) \qquad \Gamma \vdash u : A}{\Gamma \vdash \mathsf{Q\text{–}elim}(B, t_\pi, t_\sim, \pi(u)) \Rightarrow t_\pi\ u : B\ (\pi(u))}$$

There are also reduction rules that reduce terms to a weak head normal form under equality (of the form $A/R \sim_{\mathcal{U}} X$, and $x \sim_{A/R} y$, $\pi(a) \sim_{A/R} y$), and similarly for cast and quotient elimination, as well as new congruence rules as defined in Figs. 9 and 10.

We now turn to the proof that the metatheoretical properties of $\mathsf{TT}^{\mathsf{obs}}$ are preserved by the addition of quotient types. More explicitly, we extend the logical relation framework and its fundamental lemma, as well as the setoid model, to quotient types. From there, we can replay our proofs of consistency, normalization, canonicity and decidability.

Reducibility. We first extend the reducibility proof to quotient types. Since they add a new family of types in normal form, we need to add a case to the logical relation:

$$\frac{\begin{array}{c}\Gamma \vdash A \Rightarrow^* A'/(R, R_r, R_s, R_t) : \mathcal{U}_i \\ \Gamma \vdash A' : \mathcal{U}_i \qquad \forall \Delta\, \rho.\ \Delta \Vdash_\ell A'[\rho] \qquad \Gamma \Vdash_\ell R : A' \to A' \to \Omega \qquad \Gamma \vdash R_r : \Pi(x : A').\, R\ x\ x \\ \Gamma \vdash R_s : \Pi(x, y : A').\, R\ x\ y \to R\ y\ x \qquad \Gamma \vdash R_t : \Pi(x, y, z : A').\, R\ x\ y \to R\ y\ z \to R\ x\ z\end{array}}{\Gamma \Vdash_\ell A}$$

Given a type A reducible to a quotient type, we define:

- $\Gamma \Vdash_\ell A \equiv B$ if there are terms B', Q, Q_r, Q_s, Q_t such that
  - $\Gamma \vdash B \Rightarrow^* B'/(Q, Q_r, Q_s, Q_t) : \mathcal{U}_i$
  - $\forall \Delta\, \rho.\ \Delta \Vdash_\ell A'[\rho] \equiv B'[\rho]$
  - $\Gamma \Vdash_\ell R \equiv Q : A' \to A' \to \Omega$.
- $\Gamma \Vdash_\ell t : A$ if there is a normal form t' such that $\Gamma \vdash t \Rightarrow^* t' : A'/R$ and $\Gamma \Vdash_Q t'$, which is defined by
$$\frac{\Gamma \Vdash_\ell t : A'}{\Gamma \Vdash_Q \pi(t)} \qquad \frac{\Gamma \vdash n : A'/R \qquad n \text{ is neutral}}{\Gamma \Vdash_Q n}$$
- $\Gamma \Vdash_\ell t \equiv u : A$ if there are normal forms t', u' such that $\Gamma \vdash t \Rightarrow^* t' : A'/R$ and $\Gamma \vdash u \Rightarrow^* u' : A'/R$, and $\Gamma \Vdash_Q t' \equiv u'$, which is inductively defined by
$$\frac{\Gamma \Vdash_\ell t \equiv u : A'}{\Gamma \Vdash_Q \pi(t) \equiv \pi(u)} \qquad \frac{\Gamma \vdash n \equiv m : A'/R \qquad n, m \text{ are neutral}}{\Gamma \Vdash_Q n \equiv m}$$

Then, the proof of the fundamental lemma can be extended to handle the new typing rules, as well as the new cases of logical relation.

Interpretation in the Model. The type Setoid of setoids is naturally closed under quotients. Indeed, given a setoid A, providing A with a relation $R : A \times A \to_s \mathsf{Prop}$ that is reflexive, symmetric and transitive is exactly the same thing as defining an equivalence relation on the carrier set of A that extends setoidal equality. Then, A/R is simply defined as a setoid having A for its carrier set, and the relation R as its setoidal equality. One can then easily show that A/R satisfies the universal property of a mathematical quotient, which tells us that this construction is the right candidate to interpret our quotient types.

However, our universe hierarchy $\mathbf{U}_i$ is not closed under quotients, since its elements are inductively built from N, universes and function types. Therefore, we need to modify our inductive-recursive definition to account for them. In the definition of $\mathbf{U}_i$, we add a constructor

$$\begin{array}{lcl} \mathsf{c}_Q & : & (A : \mathbf{U}_i) \\ & \to & (R : \mathsf{el}\ A \to_s \mathsf{el}\ A \to_s \mathbf{\Omega}_i) \\ & \to & (R_r : (x : \mathsf{el}\ A) \to_s \mathsf{val}\ (R\ x\ x)) \\ & \to & (R_s : (x\ y : \mathsf{el}\ A) \to_s \mathsf{val}\ (R\ x\ y) \to \mathsf{val}\ (R\ y\ x)) \\ & \to & (R_t : (x\ y\ z : \mathsf{el}\ A) \to_s \mathsf{val}\ (R\ x\ y) \to \mathsf{val}\ (R\ y\ z) \to \mathsf{val}\ (R\ x\ z)) \\ & \to & \mathbf{U}_i \end{array}$$

In the definition of el, our new constructor is handled as follows:

$$\mathsf{el}\ (\mathsf{c}_Q\ A\ R\ R_r\ R_s\ R_t) \ :=\ (\mathsf{el}\ A)/(\lambda\, x\, y.\, \mathsf{val}\ (R\ x\ y))$$

In the definition of the setoidal equality on $\mathbf{U}_i$:

$$\mathsf{c}_Q\ A\ R\ R_r\ R_s\ R_t \approx \mathsf{c}_Q\ A'\ R'\ R'_r\ R'_s\ R'_t \ :=\ (e : A \approx A') \times ((x\ y : A) \to_s R\ x\ y \approx R'\ (\mathsf{el\text{-}eq}\ e\ x)\ (\mathsf{el\text{-}eq}\ e\ y))$$

and all of the other cases that involve c_Q and a different constructor reduce to $\bot$. We also need to update the definition of el-eq:

$$\mathsf{el\text{-}eq}\ (\mathsf{c}_Q\ A\ R\ R_r\ R_s\ R_t)\ (\mathsf{c}_Q\ A'\ R'\ R'_r\ R'_s\ R'_t)\ e \ :=\ (\lambda x.\mathsf{el\text{-}eq}\ A\ A'\ e.1\ x\ ,\ \lambda x.\mathsf{el\text{-}eq}^{-1}\ A\ A'\ e.1\ x)$$

Finally, the reader can check that we can extend the proofs of reflexivity, symmetry and transitivity for $\approx$ on $\mathbf{U}_i$, as well as the proofs that el-eq is compatible with reflexivity and transitivity. This defines a new universe that is closed under quotient type formation. We let the reader check that the interpretation of the syntax that defines quotients can be done in this extended universe $\mathbf{U}_i$.

4.2 Id Types

The reader may wonder if $\mathsf{TT}^{\mathrm{obs}}$ extends Martin-Löf type theory with inductive types: that is, whether a judgment of MLTT is a judgment in the proof-relevant fragment of $\mathsf{TT}^{\mathrm{obs}}$. Our rules handle the universe hierarchy, dependent products, the natural numbers in the exact same way. But there are some difficulties with Martin-Löf identity type inductive equality, and more generally with indexed inductive types.

The first idea that might come to the reader's mind is to use the proof equality types of $\mathsf{TT}^{\mathrm{obs}}$ to interpret the I-types of MLTT. Sure enough, proof irrelevance will provide us with more definitional equalities than what we require. However, we need to explain how we interpret the J eliminator for proof-relevant predicates.

It is not too hard to design a term that satisfies the correct typing rule, for instance, the term

$$\frac{\Gamma \vdash t : A \qquad \Gamma \vdash B : \Pi(x : A).\, t \sim_A x \to \mathcal{U}_j \qquad \Gamma \vdash b : B\ t\ \mathrm{refl}(t) \qquad \Gamma \vdash t' : A \qquad \Gamma \vdash e : t \sim_A t' \qquad \Gamma \vdash A : \mathcal{U}_i}{\Gamma \vdash \mathrm{cast}(B\ t\ \mathrm{refl}(t), B\ t'\ e, \mathrm{eq_J}(A, t, u, t', e), b) : B\ t'\ e}$$

where

$$\mathrm{eq_J}(A, t, B, u, t')e := \mathrm{transp}(t, \lambda(x : A).\, \lambda(e' : t \sim x).\, B\ t\ \mathrm{refl}(A) \sim B\ x\ e', \mathrm{refl}(B\ t\ \mathrm{refl}(A)), t', e).$$

But the computational behavior is not preserved: in general, this term will not reduce to u when we substitute $t' = t$ and $e = \mathrm{refl}(t)$. It might do when B is a closed term, but it certainly wil not if B is a neutral term. More generally, there is no hope to interpret I-types as proof-irrelevant types: I-types compute by doing reduction and pattern-matching on the equality proof, which cannot happen in a proof-irrelevant context.

A very similar problem was encountered by Cohen et al. [2015] in Cubical Type Theory with equality defined using the Path type, and solved by Swan [2016]. Following his ideas, we introduce Id-types:

$$\frac{\Gamma \vdash A : \mathcal{U}_i \qquad \Gamma \vdash t : A \qquad \Gamma \vdash u : A}{\Gamma \vdash \mathrm{Id}(A, t, u) : \mathcal{U}_i} \qquad \frac{\Gamma \vdash A : \mathcal{U}_i \qquad \Gamma \vdash t : A}{\Gamma \vdash \mathrm{Idrefl}(t) : \mathrm{Id}(A, t, t)} \qquad \frac{\Gamma \vdash e, e' : \mathrm{Id}(A, t, u)}{\Gamma \vdash e \sim_{\mathrm{Id}(A,t,u)} e' \Rightarrow \top : \Omega_i}$$

$$\frac{\Gamma \vdash t : A \qquad \Gamma \vdash B : \Pi(x : A).\, \mathrm{Id}(A, t, x) \to s_j \qquad \Gamma \vdash u : B\ t\ \mathrm{Idrefl}(t) \qquad \Gamma \vdash t' : A \qquad \Gamma \vdash e : \mathrm{Id}(A, t, t') \qquad \Gamma \vdash A : \mathcal{U}_i}{\Gamma \vdash \mathrm{J}(A, t, B, u, t', e) : B\ t'\ e}$$

$$\frac{\Gamma \vdash A : \mathcal{U}_i \qquad \Gamma \vdash t : A \qquad \Gamma \vdash B : \Pi(x : A).\, \mathrm{Id}(A, t, x) \to s_j \qquad \Gamma \vdash b : B\ t\ \mathrm{Idrefl}(t)}{\Gamma \vdash \mathrm{J}(A, t, B, b, t, \mathrm{Idrefl}(t)) \Rightarrow u : B\ t\ \mathrm{Idrefl}(t)}$$

These rules mimic the behavior of inductive I-types, quotiented so they contain only one inhabitant up to propositional equality. Observational equality between two identity types is defined as equality of the telescopes of arguments, as for the quotient type:

$$\frac{\Gamma \vdash A : \mathcal{U}_i \qquad \Gamma \vdash t : A \qquad \Gamma \vdash u : A \qquad \Gamma \vdash A' : \mathcal{U}_i \qquad \Gamma \vdash t' : A' \qquad \Gamma \vdash u' : A'}{\Gamma \vdash \begin{array}{l}\mathrm{Id}(A, t, u) \sim_{\mathcal{U}} \mathrm{Id}(A', t', u') \Rightarrow \\ \exists (e : A \sim_{\mathcal{U}} A').\, \mathrm{cast}(A, A', e, t) \sim_{A'} t' \wedge \mathrm{cast}(A, A', e, u) \sim_{A'} u'\end{array} : \Omega_j}$$

We may hope to define computation rules for cast by simply reducing the equality proof to a weak head normal form, and then commuting cast with the head constructor like we did for other

positive types. However, we quickly run into a problem:

$$\frac{\Gamma \vdash A, A' : \mathcal{U}_i \qquad \Gamma \vdash t : A \qquad \Gamma \vdash t, u : A' \qquad \Gamma \vdash e : \mathrm{Id}(A,t,t) \sim \mathrm{Id}(A',t',u')}{\Gamma \vdash \mathrm{cast}(\mathrm{Id}(A,t,t), \mathrm{Id}(A',t',u'), e, \mathrm{Idrefl}(t)) \Rightarrow ? : \mathrm{Id}(A',t',u')}$$

We cannot reduce this term to Idrefl, because t' and u' are not convertible in general—e only provides us with a propositional equality $t' \sim_{A'} u'$. In order to fix this, we add a term $\mathrm{Idpath}(e)$ that turns any inhabitant of $e : t \sim_A u$ into an inhabitant of $\mathrm{Id}(A,t,u)$.

$$\frac{\Gamma \vdash A : \mathcal{U}_i \qquad \Gamma \vdash t : A \qquad \Gamma \vdash u : A \qquad \Gamma \vdash e : t \sim_A u}{\Gamma \vdash \mathrm{Idpath}(e) : \mathrm{Id}(A,t,u)}$$

Then, the computation rule for cast on $\mathrm{Idrefl}(t)$ can be defined as:

$$\frac{\Gamma \vdash A, A' : \mathcal{U}_i \qquad \Gamma \vdash t : A \qquad \Gamma \vdash t', u' : A' \qquad \Gamma \vdash e : \mathrm{Id}(A,t,t) \sim \mathrm{Id}(A',t',u')}{\Gamma \vdash \begin{array}{l}\mathrm{cast}(\mathrm{Id}(A,t,t), \mathrm{Id}(A',t',u'), e, \mathrm{Idrefl}(t)) \Rightarrow \\ \mathrm{Idpath}(\mathrm{fst}(\mathrm{snd}(e))^{-1} \cdot \mathrm{snd}(\mathrm{snd}(e)))\end{array} : \mathrm{Id}(A',t',u')}$$

We also need to account for this additional constructor in reduction rules:

$$\frac{\begin{array}{c}\Gamma \vdash A : \mathcal{U}_i \\ \Gamma \vdash t : A \qquad \Gamma \vdash B : \Pi(x : A).\, \mathrm{Id}(A,t,x) \to s_j \qquad \Gamma \vdash b : B\ t\ \mathrm{Idrefl}(t) \qquad \Gamma \vdash t' : A \qquad \Gamma \vdash e : t \sim_A t'\end{array}}{\Gamma \vdash \begin{array}{l}\mathrm{J}(A,t,B,b,t',\mathrm{Idpath}(e)) \Rightarrow \\ \mathrm{cast}(B\ t\ \mathrm{Idrefl}(t), B\ t'\ \mathrm{Idpath}(e), \mathrm{eq_J}(A,t,B,u,t')e, b)\end{array} : B\ t'\ \mathrm{Idpath}(e)}$$

$$\frac{\Gamma \vdash A, A' : \mathcal{U}_i \qquad \Gamma \vdash t, u : A \qquad \Gamma \vdash e : t \sim_A u \qquad \Gamma \vdash t', u' : A' \qquad \Gamma \vdash e' : \mathrm{Id}(A,t,t) \sim \mathrm{Id}(A',t',u')}{\Gamma \vdash \begin{array}{l}\mathrm{cast}(\mathrm{Id}(A,t,u), \mathrm{Id}(A',t',u'), e', \mathrm{Idpath}(e)) \Rightarrow \\ \mathrm{Idpath}(\mathrm{fst}(\mathrm{snd}(e'))^{-1} \cdot \mathrm{ap}\ (\mathrm{cast}(A,A',\mathrm{fst}(e'),-))\ e \cdot \mathrm{snd}(\mathrm{snd}(e')))\end{array} : \mathrm{Id}(A',t',u')}$$

Along with congruence rules and reduction of the scrutinee of J and cast, these form the rules for Id types.

Relation with Setoid Equality. The introduction rule for IdPath proves that $t \sim_A u$ imples $\mathrm{Id}(A,t,u)$. Conversely, if we have an inhabitant of $\mathrm{Id}(A,t,u)$, we can get a proof of $t \sim_A u$ by combining J to reduce it to the case $u = t$, and $\mathrm{refl}(t)$ to inhabit $t \sim_A t$. Since both types are also contractible (for both notions of equality), they are equivalent.

Reducibility Proof. In order to fit identity types in the normalization proof, we add another case to the logical relation:

$$\frac{\Gamma \vdash A \Rightarrow^* \mathrm{Id}(A',t,u) : \mathcal{U}_i \qquad \Gamma \Vdash_\ell A' \qquad \Gamma \Vdash_\ell t : A' \qquad \Gamma \Vdash_\ell u : A'}{\Gamma \Vdash_\ell A}$$

When A is reducible to an identity type, we define:

- $\Gamma \Vdash_\ell A \equiv B$ if there are terms B', t', u' such that
  - $\Gamma \vdash B \Rightarrow^* \mathrm{Id}(B',t',u') : \mathcal{U}_i$
  - $\Gamma \Vdash_\ell A' \equiv B'$
  - $\Gamma \Vdash_\ell t \equiv t' : A'$
  - $\Gamma \Vdash_\ell u \equiv u' : A'$.
- $\Gamma \Vdash_\ell e : A$ if there is a normal form e' such that $\Gamma \vdash e \Rightarrow^* e' : \mathrm{Id}(A',t,u)$ and $\Gamma \Vdash_{\mathrm{Id}} e'$, which is defined by

$$\frac{}{\Gamma \Vdash_{\mathrm{Id}} \mathrm{Idrefl}(t)} \qquad \frac{\Gamma \vdash e : t \sim_{A'} u}{\Gamma \Vdash_{\mathrm{Id}} \mathrm{Idpath}(e)} \qquad \frac{\Gamma \vdash n : \mathrm{Id}(A',t,u) \qquad n \text{ is neutral}}{\Gamma \Vdash_{\mathrm{Id}} n}$$

- $\Gamma \Vdash_\ell e \equiv f : A$ if there are normal forms e', f' such that $\Gamma \vdash e \Rightarrow^* e' : \mathrm{Id}(A', t, u)$ and $\Gamma \vdash f \Rightarrow^* f' : \mathrm{Id}(A', t, u)$, and $\Gamma \Vdash_{\mathrm{Id}} e' \equiv f'$, which is inductively defined by

$$\frac{}{\Gamma \Vdash_{\mathrm{Id}} \mathrm{Idrefl}(t) \equiv \mathrm{Idrefl}(t)} \qquad \frac{\Gamma \vdash e, f : t \sim_{A'} u}{\Gamma \Vdash_{\mathrm{Id}} \mathrm{Idpath}(e) \equiv \mathrm{Idpath}(f)}$$

$$\frac{\Gamma \vdash n \equiv m : \mathrm{Id}(A', t, u) \qquad n, m \text{ are neutral}}{\Gamma \Vdash_{\mathrm{Id}} n \equiv m}$$

Again, the proof of the fundamental lemma can be extended to handle the new typing rules, as well as the new cases of the logical relation.

4.3 Box and Squash

Box types embed the proof-irrelevant types into the proof-relevant world. They are useful for a number of constructions: for instance, from a type $A : \mathcal{U}_i$ and a proof-irrelevant predicate $P : A \to \Omega_i$, one can build a *subset type* $\Sigma(x : A).\, \Box(P\, x) : \mathcal{U}_i$ whose inhabitants come with a proof of P, but retain the computational behavior of inhabitants of A.

Another typical use of $\Box$ is to define a singleton type on which it is possible to reason about equality. Indeed, although $\top$ has only one inhabitant up-to conversion, it is not possible to state this internally as equality on propositions is not defined. However, one can state contractibility of the type $\Box\top$ and prove it, as it lives in $\mathcal{U}$.

Box types can be defined as:

$$\frac{\Gamma \vdash A : \Omega_i}{\Gamma \vdash \Box A : \mathcal{U}_i} \qquad \frac{\Gamma \vdash A : \Omega_i \qquad \Gamma \vdash t : A}{\Gamma \vdash \diamond t : \Box A} \qquad \frac{\Gamma \vdash t, u : \Box A}{\Gamma \vdash t \sim_{\Box A} u \Rightarrow \top : \Omega_i} \qquad \frac{\Gamma \vdash A : \Omega_i \qquad \Gamma \vdash t : \Box A}{\Gamma \vdash \Box\text{–elim}(t) : A}$$

$$\frac{\Gamma \vdash A, B : \Omega_i}{\Gamma \vdash \Box A \sim_{\mathcal{U}} \Box B \Rightarrow A \sim_{\Omega} B : \Omega_j} \qquad \frac{\Gamma \vdash A, B : \Omega_i \qquad \Gamma \vdash t : A \qquad \Gamma \vdash e : \Box A \sim_{\mathcal{U}} \Box B}{\Gamma \vdash \mathrm{cast}(\Box A, \Box B, e, \diamond t) \Rightarrow \diamond \mathrm{cast}(A, B, e, t) : \Box B}$$

Conversely, Squash types embed the proof-relevant world into the proof-irrelevant world.

$$\frac{\Gamma \vdash A : \mathcal{U}_i}{\Gamma \vdash \|A\| : \Omega_i} \qquad \frac{\Gamma \vdash A : \mathcal{U}_i \qquad \Gamma \vdash t : A}{\Gamma \vdash |t| : \|A\|}$$

$$\frac{\Gamma \vdash A : \mathcal{U}_i \qquad \Gamma \vdash P : \|A\| \to \Omega_j \qquad \Gamma \vdash t_A : \Pi(x : A).\, P\, |x| \qquad \Gamma \vdash t : \|A\|}{\Gamma \vdash \mathrm{S}\text{–elim}(P, t_A, t) : P\, t}$$

It is not very difficult to check that we can extend the logical relation, as well as Lemmas 3.3 and 3.4 so that they handle Box types and Squash types. Likewise, we can extend the setoid model and its interpretation function, and obtain that these additions preserve the metatheoretical properties of $\mathsf{TT}^{\mathrm{obs}}$.

4.4 Other Standard Inductive Types

So far, we have explained how to integrate integers and the identity type to $\mathsf{TT}^{\mathrm{obs}}$, but it is not difficult to integrate Σ-types as well. The rules for Σ-types are similar to the rules for $\exists$-types, except for the fact that the types live in $\mathcal{U}$, which means that we also need to define equality and cast on Σ-types. We note $(a; b)$ for pairs in Σ-types to distinguish them from pairs in $\exists$-types.

$$\textsc{Eq-Pair}\quad \frac{\Gamma \vdash a : A \qquad \Gamma \vdash a' : A \qquad \Gamma \vdash b : B[x := a] \qquad \Gamma \vdash b' : B[x := a'] \qquad e' := \mathsf{ap}\ B\ e}{\Gamma \vdash (a; b) \sim_{\Sigma AB} (a'; b') \Rightarrow \exists (e : a \sim_A a').\, \mathsf{cast}(B[x := a], B[x := a'], e', b) \sim_B b' : \Omega_i}$$

$$\textsc{Eq-}\Sigma\quad \frac{\Gamma \vdash A, A' : s_i \qquad \Gamma, x : A \vdash B : s'_j \qquad \Gamma, x : A' \vdash B' : s'_j \qquad a := \mathsf{cast}(A', A, e^{-1}, a')}{\Gamma \vdash \begin{array}{l}\Sigma(x : A).\, B \sim_{\mathcal{U}k} \Sigma(x : A').\, B' \Rightarrow \\ \exists (e : A \sim_{\mathcal{U}i} A').\, \Pi(a' : A').\, B[x := a] \sim_{\mathcal{U}j} B'[x := a']\end{array} : \Omega_i}\ \begin{array}{l} i \le k \\ j \le k \end{array}$$

$$\textsc{Cast-}\Sigma\quad \frac{\Gamma \vdash e : \Sigma(x : A).\, B \sim_{\mathcal{U}} \Sigma(x : A').\, B' \qquad \Gamma \vdash a : A \qquad \Gamma \vdash b : B[x := a] \qquad a' := \mathsf{cast}(A, A', \mathsf{fst}(e), a)}{\Gamma \vdash \begin{array}{l}\mathsf{cast}(\Sigma(x : A).\, B, \Sigma(x : A').\, B', e, (a; b)) \Rightarrow \\ (a'; \mathsf{cast}(B[x := a], B[x := a'], \mathsf{snd}(e), b))\end{array} : \Sigma(x : A').\, B'}$$

We conjecture that the reduction of equality and cast for any indexed inductive types as defined in CIC can be described, although we leave the general construction for future work.

5 RELATED AND FUTURE WORK

Compared to [Altenkirch et al. 2007], the most important ingredient in $\mathsf{TT}^{\mathsf{obs}}$ is the use of definitional proof irrelevance. This added flexibility in computations allows our recursors to enjoy proper computational behavior on open terms, and it also lets us seamlessly treat universe hierarchies. Moreover, the normalization proof for OTT relies on a normalization conjecture for a different theory, unlike the normalization proof for $\mathsf{TT}^{\mathsf{obs}}$.

In [Altenkirch et al. 2019], the authors define a setoid model in $\mathsf{MLTT} + \Omega$. Then, they interpret a version of MLTT with proof-irrelevant identity types that support propext and funext in their model, thereby providing a computational interpretation of these principles. However, handling universes in their model requires some additions to $\mathsf{MLTT} + \Omega$, and the resulting theory is only conjectured to be normalizing. In contrast to this, $\mathsf{TT}^{\mathsf{obs}}$ is a full-fledged type theory and does not require any external model to compute.

Compared to XTT [Sterling et al. 2019], the strengths of $\mathsf{TT}^{\mathsf{obs}}$ are a normalization strategy that exhibits canonicity, as well a full proof that conversion and typing are decidable. These properties allow us to present a concrete implementation of our system in a proof assistant. In XTT however, Sterling *et al.* show that typing cannot be decidable, as there is no way to deduce $A \sim A'$ and $B \sim B'$ from a proof of $A \times B \sim A' \times B'$. In order to fix this shortcoming, they suggest adding a "typecase" operator, but argue against it since it forces the universe to be closed, thereby severely constraining the possible semantics. In $\mathsf{TT}^{\mathsf{obs}}$, we obtain the injectivity of type contructors from the behavior of observational equality in the universe. These rules somewhat constrain the semantics—for instance, we cannot interpret $\mathsf{TT}^{\mathsf{obs}}$ in set theory using a Grothendieck universe as the interpretation of $\mathcal{U}$—but our universe remains open to the addition of arbitrary types.

The natural next step of our work is to implement $\mathsf{TT}^{\mathsf{obs}}$ inside Coq, Lean or Agda, which should not be too difficult as all of them already feature a proof-irrelevant universe of propositions. The main missing ingredient for a concrete implementation is a general description of the reduction of equality and cast on arbitrary indexed inductive types, as explained in Section 4.4.

Another interesting line of work is the marriage of $\mathsf{TT}^{\mathsf{obs}}$ with cubical type theory in a 2-level type theory setting [Altenkirch et al. 2016; Capriotti 2017; Voevodsky 2013], which could lead to an implementation of $\mathsf{TT}^{\mathsf{obs}}$ in the cubical extension of Agda [Vezzosi et al. 2019].

REFERENCES

Andreas Abel and Thierry Coquand. 2020. Failure of Normalization in Impredicative Type Theory with Proof-Irrelevant Propositional Equality. *Logical Methods in Computer Science* Volume 16, Issue 2 (June 2020). https://doi.org/10.23638/LMCS-16(2:14)2020

Andreas Abel, Joakim Öhman, and Andrea Vezzosi. 2018. Decidability of Conversion for Type Theory in Type Theory. *Proceedings of the ACM on Programming Languages* 2, POPL, Article 23 (Jan. 2018), 29 pages. https://doi.org/10.1145/3158111

Peter Aczel. 1978. The Type Theoretic Interpretation of Constructive Set Theory. In *Logic Colloquium '77*, Angus Macintyre, Leszek Pacholski, and Jeff Paris (Eds.). Studies in Logic and the Foundations of Mathematics, Vol. 96. Elsevier, 55–66. https://doi.org/10.1016/S0049-237X(08)71989-X

T. Altenkirch. 1999. Extensional equality in intensional type theory. In *Proceedings. 14th Symposium on Logic in Computer Science (Cat. No. PR00158)*. 412–420. https://doi.org/10.1109/LICS.1999.782636

Thorsten Altenkirch, Simon Boulier, Ambrus Kaposi, and Nicolas Tabareau. 2019. Setoid type theory - a syntactic translation. In *MPC 2019 - 13th International Conference on Mathematics of Program Construction (LNCS)*, Vol. 11825. Springer, 155–196. https://doi.org/10.1007/978-3-030-33636-3_7

Thorsten Altenkirch, Paolo Capriotti, and Nicolai Kraus. 2016. Extending Homotopy Type Theory with Strict Equality. In *CSL*.

Thorsten Altenkirch, Conor McBride, and Wouter Swierstra. 2007. Observational equality, now!. In *Proceedings of the Workshop on Programming Languages meets Program Verification (PLPV 2007)*. 57–68.

Paolo Capriotti. 2017. *Models of type theory with strict equality*. Ph.D. Dissertation. University of Nottingham.

Jesper Cockx, Nicolas Tabareau, and Théo Winterhalter. 2021. The Taming of the Rew: A Type Theory with Computational Assumptions. *Proc. ACM Program. Lang.* 5, POPL, Article 60 (Jan. 2021), 29 pages. https://doi.org/10.1145/3434341

Cyril Cohen, Thierry Coquand, Simon Huber, and Anders Mörtberg. 2015. Cubical Type Theory: a constructive interpretation of the univalence axiom. In *21st International Conference on Types for Proofs and Programs (21st International Conference on Types for Proofs and Programs)*. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Tallinn, Estonia, 262. https://doi.org/10.4230/LIPIcs.TYPES.2015.5

Thierry Coquand, Simon Huber, and Anders Mörtberg. 2018. On Higher Inductive Types in Cubical Type Theory. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science* (Oxford, United Kingdom) *(LICS '18)*. Association for Computing Machinery, New York, NY, USA, 255–264. https://doi.org/10.1145/3209108.3209197

Peter Dybjer. 1996. Internal type theory. In *Types for Proofs and Programs*, Stefano Berardi and Mario Coppo (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 120–134.

Gaëtan Gilbert, Jesper Cockx, Matthieu Sozeau, and Nicolas Tabareau. 2019. Definitional Proof-Irrelevance without K. *Proceedings of the ACM on Programming Languages* 3 (Jan. 2019), 1–28. https://doi.org/10.1145/329031610.1145/3290316

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

Martin Hofmann. 1993. Non Strictly Positive Datatypes in System F. Email on the Types mailing list. http://www.seas.upenn.edu/~sweirich/types/archive/1993/msg00027.html

Martin Hofmann. 1995. *Extensional concepts in intensional type theory*. Ph.D. Dissertation. University of Edinburgh.

Chris Kapulkin and Peter LeFanu Lumsdaine. 2018. The simplicial model of Univalent Foundations (after Voevodsky). (2018). arXiv:1211.2851 [math.LO]

Meven Lennon-Bertrand. 2021. Complete Bidirectional Typing for the Calculus of Inductive Constructions. In *12th International Conference on Interactive Theorem Proving (ITP 2021) (Leibniz International Proceedings in Informatics (LIPIcs))*, Liron Cohen and Cezary Kaliszyk (Eds.), Vol. 193. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. https://doi.org/10.4230/LIPIcs.ITP.2021.24

Per Martin-Löf. 1975. An Intuitionistic Theory of Types: Predicative Part. In *Logic Colloquium '73*, H.E. Rose and J.C. Shepherdson (Eds.). Studies in Logic and the Foundations of Mathematics, Vol. 80. Elsevier, 73 – 118. https://doi.org/10.1016/S0049-237X(08)71945-1

Jonathan Sterling, Carlo Angiuli, and Daniel Gratzer. 2019. Cubical Syntax for Reflection-Free Extensional Equality. In *4th International Conference on Formal Structures for Computation and Deduction (FSCD 2019) (Leibniz International Proceedings in Informatics (LIPIcs))*, Herman Geuvers (Ed.), Vol. 131. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 31:1–31:25. https://doi.org/10.4230/LIPIcs.FSCD.2019.31

Andrew Swan. 2016. An algebraic weak factorisation system on 01-substitution sets: a constructive proof. *Journal of Logic and Analysis* (2016). https://doi.org/10.4115/jla.2016.8.1

The Univalent Foundations Program. 2013. *Homotopy Type Theory: Univalent Foundations of Mathematics*. Institute for Advanced Study.

Andrea Vezzosi, Anders Mörtberg, and Andreas Abel. 2019. Cubical Agda: A Dependently Typed Programming Language with Univalence and Higher Inductive Types. *Proc. ACM Program. Lang.* 3, ICFP, Article 87 (July 2019), 29 pages.

https://doi.org/10.1145/3341691
Vladimir Voevodsky. 2013. A simple type system with two identity types. https://ncatlab.org/homotopytypetheory/files/HTS.pdf
Vladimir Voevodsky. 2015. A C-system defined by a universe category. *Theory Appl. Categ.* 30 (2015), No. 37, 1181–1215. http://www.tac.mta.ca/tac/volumes/30/37/30-37abs.html

A CONGRUENCE AND SUBSTITUTION RULES OF $\mathrm{TT}^{\mathrm{obs}}$

$$\textsc{Π-Cong}\quad \frac{\Gamma \vdash A \equiv A' : s_i \qquad \Gamma, x : A \vdash B \equiv B' : s'_j}{\Gamma \vdash \Pi(x : A).\, B \equiv \Pi(x : A').\, B' : s'_k}\ \frac{i \leq k}{j \leq k}$$

$$\textsc{∃-Cong}\quad \frac{\Gamma \vdash A \equiv A' : \Omega_i \qquad \Gamma, x : A \vdash B \equiv B' : \Omega_i}{\Gamma \vdash \exists(x : A).\, B \equiv \exists(x : A').\, B' : \Omega_i}$$

$$\textsc{App-Cong}\quad \frac{\Gamma \vdash t \equiv t' : \Pi(x : A).\, B \qquad \Gamma \vdash u \equiv u' : A}{\Gamma \vdash t\ u \equiv t'\ u' : B[x := u]}$$

$$\textsc{Suc-Cong}\quad \frac{\Gamma \vdash n \equiv n' : \mathbb{N}}{\Gamma \vdash \mathrm{S}\ n \equiv \mathrm{S}\ n' : \mathbb{N}}$$

$$\mathbb{N}\textsc{-Elim-Cong}\quad \frac{\Gamma \vdash A \equiv A' : \mathbb{N} \to s_i \qquad \Gamma \vdash t_0 \equiv t'_0 : A\ 0 \qquad \Gamma \vdash t_S \equiv t'_S : \Pi(n : \mathbb{N}).\, A\ n \to A\ (\mathrm{S}\ n) \qquad \Gamma \vdash n \equiv n' : \mathbb{N}}{\Gamma \vdash \mathbb{N}\text{-elim}(A, t_0, t_S, n) \equiv \mathbb{N}\text{-elim}(A', t'_0, t'_S, n') : A\ n}$$

$$\bot\textsc{-Elim-Cong}\quad \frac{\Gamma \vdash A \equiv A' : s_i \qquad \Gamma \vdash t \equiv t' : \bot}{\Gamma \vdash \bot\text{-elim}(A, t) \equiv \bot\text{-elim}(A', t') : A}$$

$$\textsc{Eq-Cong}\quad \frac{\Gamma \vdash A \equiv A' : \mathcal{U}_i \qquad \Gamma \vdash t \equiv t' : A \qquad \Gamma \vdash u \equiv u' : A}{\Gamma \vdash t \sim_A u \equiv t' \sim_{A'} u' : \Omega_i}$$

$$\textsc{Cast-Cong}\quad \frac{\Gamma \vdash A \equiv A' : s_i \qquad \Gamma \vdash B \equiv B' : s_i \qquad \Gamma \vdash e \equiv e' : A \sim_s B \qquad \Gamma \vdash t \equiv t' : A}{\Gamma \vdash \mathrm{cast}(A, B, e, t) \equiv \mathrm{cast}(A', B', e', t') : B}$$

Fig. 9. $\mathrm{TT}^{\mathrm{obs}}$ Conversion Rules (congruence only)

$$\textsc{App-Subst}\quad \frac{\Gamma \vdash t \Rightarrow t' : \Pi(x : A).\, B \qquad \Gamma \vdash u : A}{\Gamma \vdash t\ u \Rightarrow t'\ u : B[x := u]}$$

$$\mathbb{N}\textsc{-Elim-Subst}\quad \frac{\Gamma \vdash A : \mathbb{N} \to s_i \qquad \Gamma \vdash t_0 : A\ 0 \qquad \Gamma \vdash t_S : \Pi(n : \mathbb{N}).\, A\ n \to A\ (\mathrm{S}\ n) \qquad \Gamma \vdash n \Rightarrow n' : \mathbb{N}}{\Gamma \vdash \mathbb{N}\text{–elim}(A, t_0, t_S, n) \Rightarrow \mathbb{N}\text{–elim}(A, t_0, t_S, n') : A\ n}$$

$$\textsc{Eq-Subst}\quad \frac{\Gamma \vdash A \Rightarrow A' : \mathcal{U}_i \Gamma \vdash t : A \qquad \Gamma \vdash u : A}{\Gamma \vdash t \sim_A u \Rightarrow t \sim_{A'} u : \Omega_i}$$

$$\textsc{Eq-Univ-Subst}\quad \frac{\Gamma \vdash A \Rightarrow A' : \mathcal{U}_i \qquad \Gamma \vdash B : \mathcal{U}_i}{\Gamma \vdash A \sim_{\mathcal{U}_i} B \Rightarrow A' \sim_{\mathcal{U}_i} B : \Omega_j}\ i<j$$

$$\textsc{Eq-Univ-}\mathbb{N}\textsc{-Subst}\quad \frac{\Gamma \vdash B \Rightarrow B' : \mathcal{U}_i}{\Gamma \vdash \mathbb{N} \sim_{\mathcal{U}_i} B \Rightarrow \mathbb{N} \sim_{\mathcal{U}_i} B' : \Omega_j}\ i<j$$

$$\textsc{Eq-Univ-}\Pi\textsc{-Subst}\quad \frac{\Gamma \vdash \Pi(x : A).\, P : \mathcal{U}_i \Gamma \vdash B \Rightarrow B' : \mathcal{U}_i}{\Gamma \vdash \Pi(x : A).\, P \sim_{\mathcal{U}_i} B \Rightarrow \Pi(x : A).\, P \sim_{\mathcal{U}_i} B' : \Omega_j}\ i<j$$

$$\textsc{Eq-}\mathbb{N}\textsc{-Subst}\quad \frac{\Gamma \vdash n \Rightarrow n' : \mathbb{N} \qquad \Gamma \vdash m : \mathbb{N}}{\Gamma \vdash n \sim_{\mathbb{N}} m \Rightarrow n' \sim_{\mathbb{N}} m : \Omega_i}$$

$$\textsc{Eq-}\mathbb{N}\textsc{-Zero-Subst}\quad \frac{\Gamma \vdash n \Rightarrow n' : \mathbb{N}}{\Gamma \vdash 0 \sim_{\mathbb{N}} n \Rightarrow 0 \sim_{\mathbb{N}} n' : \Omega_i}$$

$$\textsc{Eq-}\mathbb{N}\textsc{-Suc-Subst}\quad \frac{\Gamma \vdash m : \mathbb{N} \qquad \Gamma \vdash n \Rightarrow n' : \mathbb{N}}{\Gamma \vdash \mathrm{S}\ m \sim_{\mathbb{N}} n \Rightarrow \mathrm{S}\ m \sim_{\mathbb{N}} n' : \Omega_i}$$

$$\textsc{Cast-Subst}\quad \frac{\Gamma \vdash A \Rightarrow A' : s_i \qquad \Gamma \vdash B : s_i \qquad \Gamma \vdash e : A \sim_{s_i} B \qquad \Gamma \vdash t : A}{\Gamma \vdash \mathrm{cast}(A, B, e, t) \Rightarrow \mathrm{cast}(A', B, e, t) : B}$$

$$\textsc{Cast-}\mathcal{U}\textsc{-Subst}\quad \frac{\Gamma \vdash A : \mathcal{U}_i \qquad \Gamma \vdash e : \mathcal{U}_i \sim_{\mathcal{U}_j} B \qquad \Gamma \vdash B \Rightarrow B' : \mathcal{U}_j}{\Gamma \vdash \mathrm{cast}(\mathcal{U}_i, B, e, A) \Rightarrow \mathrm{cast}(\mathbb{N}, B', e, A) : B}\ i<j$$

$$\textsc{Cast-}\mathbb{N}\textsc{-Subst}\quad \frac{\Gamma \vdash n : \mathbb{N} \qquad \Gamma \vdash e : \mathbb{N} \sim_{\mathcal{U}_i} B \qquad \Gamma \vdash B \Rightarrow B' : \mathcal{U}_i}{\Gamma \vdash \mathrm{cast}(\mathbb{N}, B, e, n) \Rightarrow \mathrm{cast}(\mathbb{N}, B', e, n) : B}$$

$$\textsc{Cast-}\Pi\textsc{-Subst}\quad \frac{\Gamma \vdash \Pi(x : A).\, P : \mathcal{U}_i \qquad \Gamma \vdash f : \Pi(x : A).\, P \qquad \Gamma \vdash e : \Pi(x : A).\, P \sim_{\mathcal{U}_i} B \qquad \Gamma \vdash B \Rightarrow B' : \mathcal{U}_i}{\Gamma \vdash \mathrm{cast}(\Pi(x : A).\, P, B, e, f) \Rightarrow \mathrm{cast}(\Pi(x : A).\, P, B', e, f) : B}$$

$$\textsc{Cast-}\mathbb{N}\textsc{-}\mathbb{N}\textsc{-subst}\quad \frac{\Gamma \vdash e : \mathbb{N} \sim_{\mathcal{U}} \mathbb{N} \qquad \Gamma \vdash n \Rightarrow n' : \mathbb{N}}{\Gamma \vdash \mathrm{cast}(\mathbb{N}, \mathbb{N}, e, n) \Rightarrow \mathrm{cast}(\mathbb{N}, \mathbb{N}, e, n') : \mathbb{N}}$$

Fig. 10. $\mathrm{TT}^{\mathrm{obs}}$ Reduction Rules (substitutions only)