Type Theory in Type Theory Using a Strictified Syntax

AMBRUS KAPOSI, Eötvös Loránd University, Hungary
LOÏC PUJET, Stockholm University, France

The metatheory of dependent types has seen a lot of progress in recent years. In particular, the development of categorical gluing finally lets us work with *semantic* presentations of type theory (such as categories with families) to establish fundamental properties of type theory such as canonicity and normalisation. However, proofs by gluing have yet to reach the stage of computer formalisation: formal proofs for the metatheory of dependent types are still stuck in the age of tedious syntactic proofs. The main reason for this is that semantic presentations of type theory are defined using sophisticated indexed inductive types, which are especially prone to “transport hell”. In this paper, we introduce a new technique to work with CwFs in intensional type theory without getting stuck in transport hell. More specifically, we construct an alternative presentation of the initial CwF which encodes the substitutions as metatheoretical functions. This has the effect of strictifying all the equations that are involved in the substitution calculus, which greatly reduces the need for transports. As an application, we use our strictified initial CwF to give a short and elegant proof of canonicity for a type theory with dependent products and booleans with large elimination. The resulting proof is fully formalised in Agda.

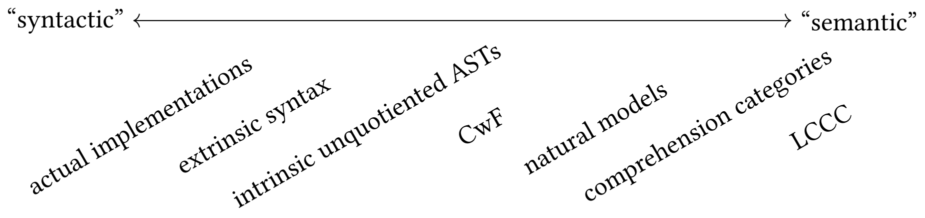
CCS Concepts: • **Theory of computation → Type theory**; *Proof theory*; *Constructive mathematics.*

Additional Key Words and Phrases: Canonicity, Normalisation, Gluing, Proof Assistants, Categories with Families

ACM Reference Format:
Ambrus Kaposi and Loïc Pujet. 2018. Type Theory in Type Theory Using a Strictified Syntax. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation email (Conference acronym ’XX).* ACM, New York, NY, USA, 30 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

What is a dependent type theory? There is no single, universally accepted answer to this question. Dependent type theories can be described in a number of ways, ranging from the most syntactic to the most semantic:



Authors’ addresses: Ambrus Kaposi, akaposi@inf.elte.hu, Eötvös Loránd University, Budapest, Hungary; Loïc Pujet, Stockholm University, Stockholm, France, loic@pujet.fr.

Conference acronym ’XX, June 03–05, 2018, Woodstock, NY
© 2018 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-XXXX-X/2018/06
https://doi.org/XXXXXXX.XXXXXXX

The most syntactic definitions appear in actual implementations [Moura et al. 2015; Agda; Coq] where terms are untyped syntax trees and there is no typing relation, just an (efficient) algorithm for typechecking in some inconsistent metalanguage. A first step towards a more semantic definition is extrinsic syntax, where we work in a consistent metalanguage such as Coq [Adjedj et al. 2024] or Agda [Abel et al. 2017] and use typing relations to carve the meaningful terms out of the untyped ASTs. Intrinsic syntax goes yet a bit further by removing meaningless terms (the ASTs are indexed by their types), but it still requires a separate relation to describe conversion [Chapman 2009; Danielsson 2006]. In the category with families (CwF) approach [Altenkirch and Kaposi 2016b; Castellan et al. 2021; Dybjer 1996], type theory is described by an algebraic theory, and conversion coincides with the metatheoretic equality. In other words, we work with intrinsic terms quotiented by conversion. Awodey's natural models [Awodey 2018] replace the type-indexing of terms with a map from terms to types; comprehension categories [Jacobs 1993] replace the family of types with a category of types and a fibration explaining the connection between contexts and types, while simultaneously weakening substitution so that it is functorial only up to isomorphism; and finally LCCCs [Clairambault and Dybjer 2014; Seely 1984] use slice categories to model types and terms as local objects/morphisms, and they build in extensional identity types.

Moving towards the left on the axis gives more practical implementations and observable computations, while moving towards the right gives elegance, abstraction, and easier metatheoretic proofs. Moving towards the left also involves more ad-hoc choices: for example, one has to decide whether to use lambda abstractions à la Curry or à la Church, or whether the typing rules should be paranoid or economic [Winterhalter 2020, Section 9.2]. Conversely, moving towards the right forces some choices: for example, all notions right of intrinsic unquotiented ASTs use De Bruijn-like combinators instead of named variables, and instantiation (substitution of variables by terms) is not a recursively defined operation, but one of the constructors in the syntax. Going all the way to the far right, it even becomes difficult to see how the semantic constructions relate to actual types/terms. Overall, CwFs occupy a sweet spot where the terms are similar to actual syntax, but there is already a usable notion of model.

1.1 Normalisation and Gluing

Until recently, some properties of dependent type theory were believed to be essentially syntactic, in the sense that it was not clear whether they can be proven while staying on the semantic side of the axis. One such example is *normalisation*. Normalisation traditionally means that any well-typed term can be reduced to a deep normal form, *i.e.*, a term which does not contain any redex. This property plays an important role when implementing a type checker for type theory, as the usual algorithm for type checking performs type comparisons by normalising both types and then checking syntactic equality of their normal forms. In semantic notions of type theory such as CwFs, there is no notion of reduction, but it is nevertheless possible to talk about reduction-free normalisation: we can define normal forms inductively (along with a map from normal forms to terms), and then normalisation states that the map from normal forms to terms has a section – which does imply decidability of equality for quotiented terms. For simple type theory and System F, there are normalisation proofs on the CwF level of abstraction that go back to the 90s [Altenkirch, Hofmann, et al. 1995, 1996], but until recently it was not clear how to scale this technology to dependent types with a universe. Indeed, normalisation is usually proven via some version of Tait's reducibility method, and the traditional definition of reducibility predicates for the universe relies on reduction [Abel 2013]. The breakthrough came when it was realised that the logical relation can be made proof-relevant [Altenkirch and Kaposi 2016a; Shulman 2015], which echoes a standard categorical technique called *gluing* [Kaposi, Huber, et al. 2019].

Today, gluing-style normalisation proofs have been adapted to a wide range of type theories [Altenkirch, Kaposi, and Kovács 2017; Coquand 2019; Gratzer 2022; Sterling and Angiuli 2021], but interestingly enough, they have yet to reach the stage of computer formalisation. Most formalisation efforts still take place on the syntactic and extrinsic side, *e.g.* most recently for Lean [Carneiro 2024]. The resulting normalisation proofs are beasts of tens of thousands of lines of code, contrasting with the elegance of gluing-style proofs which fit on a single page. So why not use gluing in computer formalisation? The answer lies in the fine print: *informal* gluing proofs fit on a single page, but in practice, formalising normalisation using an intrinsic and quotiented syntax is even more difficult than with extrinsic syntax. We identify three reasons for this:

(i) Since the syntax is defined as intrinsically well-typed ASTs quotiented by conversion, one needs a metalanguage which supports quotients (quotient inductive-inductive types, QIITs [Kaposi, Kovács, and Altenkirch 2019] to be precise). In contrast, extrinsic syntax can be implemented in plain Martin-Löf type theory with inductive types.

(ii) Intrinsic representations are especially prone to "transport hell". An analogy: whenever we work with vectors indexed by their length, we end up needing transports to adjust the indices – for instance, the statement of associativity of vector concatenation depends on the associativity of addition, which appears as a transport on one of the vectors. Subsequently, we need to invoke general properties of transport (e.g. that it commutes with function application) every time we manipulate proofs involving transport. This can get out of hand, and turn most of the code into uninteresting reasoning about transports. In fact, it is generally advised to avoid vectors, and instead to work with non-indexed lists and separate proofs about their length. For the same reasons, it is advised to work with non-indexed terms and to use separate proofs of typing.

(iii) Another handicap for intrinsic syntax is the fact that substitution laws such as $(\mathsf{app}\, t\, u)[\gamma] = \mathsf{app}\,(t[\gamma])\,(u[\gamma])$ are weak: these come from equality constructors in the QIIT of syntax, and as such, they are *propositional* equalities. Compare this with extrinsic syntax, where substitution is defined by recursion on terms, and such equations hold *definitionally*. If an equality is definitional, there is no need to transport over it. Combined with indexing, this makes formalising gluing-style normalisation proofs very difficult. For example, even for simple type theory (where there is no type-indexing at all), if all the equations in the syntax are definitional (the syntax is fully strict), the canonicity proof takes 44 lines of Agda code, if all equations are weak, it takes 190 lines of code [Kaposi 2023]. The difference comes from uninteresting boilerplate of transport-reasoning.

1.2 Strictifying the Syntax of Type Theory

If we want to bring the elegance and simplicity of gluing arguments to the world of formalised proofs, we need to find a way around these three issues. Issue (i) is the simplest one, as it can be solved simply by moving to a metalanguage with support for QIITs. Cubical Agda [Vezzosi et al. 2021] is a good candidate, but it is a bit of an overkill, because we usually don't need univalence and h-sets are enough when working with the syntax. A more suitable language is observational type theory (OTT, [Altenkirch, McBride, et al. 2007; Pujet and Tabareau 2022]) which supports quotients and has a definitionally proof-irrelevant equality type. There is an experimental OTT extension for Coq [Pujet, Leray, et al. 2025] but OTT can more generally be implemented in any proof assistant which supports rewrite rules [Cockx et al. 2021; Leray et al. 2024].

Issues (ii) and (iii) are more interesting, and they are the main focus of this paper. Our contribution is a method to transform all the substitution laws of an intrinsic, quotiented, CwF-based syntax to definitional equations. This eliminates the greatest drawback of intrinsic syntaxes over extrinsic presentations. In fact, our tool makes not only the substitution laws definitional, but also almost all equations of the substitution calculus (the CwF equations). For example, the functor law $t[\gamma \circ \delta] = t[\gamma][\delta]$ is also definitional, in contrast with extrinsic syntaxes where it is usually weak. We demonstrate our technique on the syntax of a type theory with Π types and booleans with large elimination, and we conjecture that it works in general for any type theory. More precisely, we expect that for any second-order generalised algebraic theory (SOGAT, [Kaposi and S. Xie 2024]), there is a first-order model equivalent to the syntax, which has a strict substitution calculus. As all non-substructural languages with binders can be described as SOGATs, we expect that our technique works very generically. The only CwF equation which is weak in our strictified syntax is the η law for substitutions saying that any substitution into an extended context is the same as a substitution into the smaller context together with a term.

An analogy for our construction is *difference lists*, which are available in the Haskell Prelude: concatenation of lists is only provably associative, but concatenation of difference lists is definitionally associative. A difference list is a $\mathsf{List} \to \mathsf{List}$ function which prepends a list to its input (*i.e.*, xs is represented by $\lambda ys.\, xs \mathbin{++} ys$), and concatenation of difference lists is just function composition, which is definitionally associative. More abstractly, difference lists can be seen an instance of the Yoneda embedding, which may be used to strictify the equations for composition in an arbitrary category: given a category C, we can replace morphisms $C(J, I)$ by natural transformations between the presheaves $\mathsf{y}\,J$ and $\mathsf{y}\,I$ where $\mathsf{y}\,I\,K := C(K, I)$. Composition of these natural transformations is strictly associative and unital, and by the Yoneda lemma, $C(J, I) \cong (\mathsf{y}\,J \xrightarrow{\cdot} \mathsf{y}\,I)$, hence the strictified category is equivalent to the original one. The technique that we present in this paper roughly provides an extension of this "Yoneda strictification" operation to categories with families.

Another useful point of view comes from the semantics of higher-order abstract syntax (HOAS [Hofmann 1999]), and relatedly the semantics of logical frameworks [Harper et al. 1993] and two-level type theory [Annenkov et al. 2023]. Presheaves over a category are a model of type theory [Hofmann 1997]. If the base category happens to be a CwF, then in addition to the usual universe of presheaves (written internally to the presheaf model as Set), we have another universe $\mathsf{Ty} : \mathrm{Set}$, $\mathsf{Tm} : \mathsf{Ty} \to \mathrm{Set}$ which is defined using the types and terms of the base category. Furthermore, if the base CwF also supports Π types, then this universe is closed under dependent function space – that is, there is a $\Pi : (A : \mathsf{Ty}) \to (\mathsf{Tm}\,A \to \mathsf{Ty}) \to \mathsf{Ty}$ in the presheaf internal language along with lambda-abstraction and application combinators. The situation is similar for other type and term formers. Note that Π is a binder, and the extra variable in the second argument is modelled by the function space of the presheaf model (which acts as the metatheory, as we work internally). Thus the "substitution calculus" in this setting is implemented by the metatheoretic function space. And in type-theoretic metatheories, the function space has nice properties, e.g. function composition is definitionally associative. We make use of this: after internalising the syntax in the internal language of presheaves over the syntax, we *externalise* the universe Ty, Tm. This process replaces the weak substitution calculus (which causes problem (iii) above) by one which comes from constructions in the presheaf model. If our presheaf model is strict, the externalised CwF will also be strict.

In summary, our construction takes as input a weak model of type theory, and then using this internalisation–externalisation construction, we obtain another isomorphic model where (almost) all the CwF equations are strict. Note that we do not strictify equations outside of the substitution calculus, such as the β/η laws for Π types. The result is a model that combines the strengths of syntactic models with the strengths of semantic models. Our model enjoys definitional substitution laws and minimal transport hell just like extrinsic syntax, but it supports the same induction principle as the initial object in the category of CwFs. We demonstrate the efficiency of our technique by implementing a canonicity proof for our object type theory which is as concise as the one-page gluing proof from Kaposi, Huber, et al. [2019, page 11].

1.3 Structure of the paper

After describing related work and our metatheory, we introduce the intrinsic syntax of type theory using CwFs in Section 2. We also illustrate the difficulties that arise when working with intrinsic syntax in an intensional type theory. The heart of the paper is Section 4, in which we unfold our strictification construction in an abstract setting. This construction is parameterised by a strict model of type theory, and uses it to define a substitution-strict model isomorphic to the syntax. Section 3 serves as a warm up for Section 4, where we attempt to do the strictification construction using presheaves, but find that regular presheaves are missing some definitional equations. The strictification is finally completed in Section 5, where we instantiating the requirements of Section 4 with a stricter notion of presheaves called *prefascist sets*. We use our strictified syntax to prove canonicity in Section 6, and we conclude in Section 7.

1.4 Related work

A simpler way to strictify all equations in a CwF-based syntax is shallow embedding [Kaposi, Kovács, and Kraus 2019]. This technique produces a model of type theory which is fully strict, and it can be shown externally to the metatheory that this model is equivalent to the syntax. However, we cannot see this internally, as there is no induction principle for this model. Thus it can be used to typecheck the canonicity proof, but we cannot use it for computation. Another strictification method follows the extrinsic approach directly and replaces substitution in the syntax by a recursively defined substitution [Kaposi 2023]. For dependent types, it is not clear how to do this *at the same time* as defining the syntax, but it can be done as a second step. The technique has been implemented for simple types, but as far as we know, not for dependent types. We also strictify more equations, e.g. the functor laws for substitutions.

A more generic way of strictifying propositional equalities is to use equality reflection (*i.e.*, extensional type theory), which turns propositional equalities into definitional ones, thereby removing the need for transports. There are several options to use equality reflection, besides working within an actual implementation of extensional type theory such as NuPRL [Constable et al. 1985]. For instance, it is possible to circumscribe a useful subsystem of extensional type theory with equality reflection for erasable types [Winterhalter 2024]; one can also use rewrite rules to turn selected equations definitional [Cockx et al. 2021], or one can attempt to translate constructions written in extensional type theory into intensional type theory [Hofmann 1995; Oury 2005; Winterhalter et al. 2019]. Unlike these approaches, our work stays in intensional type theory (more specifically its observational variant).

Synthetic Tait computability [Sterling 2021] and internal sconing [Bocquet et al. 2023] are techniques to prove properties of the syntax of type theory (such as canonicity) in the internal language of glued toposes/presheaves. As they abstract over the particular presentation of the substitution calculus, they don't face the strictness issues that external descriptions have. However an externalisation step is needed to use them as implementations, and this has not been developed yet. Our approach is external and its computational content is clear, e.g. it could be used in an implementation of a typechecker. Our strict substitution calculus removes transport hell, and when comparing to the internal approaches, the only inconvenience that remains is that variables are handled by De Bruijn indices and weakenings are explicit. Our strictification construction is roughly the following three steps of the internal sconing approach merged together: internalisation $\to$ telescopic contextualisation $\to$ externalisation.

Intrinsic formalisations of type theory end up in transport hell [Altenkirch and Kaposi 2016b] or remove term-indexing and thus are further away from the syntax such as [Brunerie and Boer 2020] which used comprehension categories instead of CwFs.

In this paper by strictification we mean the replacement of propositional equalities by definitional equalities. Another meaning of the same word is the replacement of isomorphisms by propositional equalities: this happens when we have a semantic notion of model where substitution is functorial only up to isomorphism, and we want to make it definitional. Bocquet [Bocquet 2021] categorises these constructions into right adjoint splitting [Curien et al. 2014; Hofmann 1994] and left adjoint splitting (the local universe method) [Lumsdaine and Warren 2015]. While the purpose of these methods is different from ours, there are two connections: local universes can be also used to obtain more definitional equalities in formalisation, that is, strictify a model in our sense (see e.g. the formalisation of [Donkó and Kaposi 2021]); right adjoint splitting is related to the notion of prefascist set, which is defined as a subpresheaf of the right Kan extension applied to a presheaf on a discrete category [Pujet 2022, Section 6.3.2].

1.5 Metatheory and formalisation

As the main focus this paper is strictification, a significant part of our exposition will involve considerations about the definitional equality of our metatheory. Therefore, we need to deal with *three* levels of abstraction: the object theory which is modelled by its initial CwF, the metatheory in which we formalise our definitions, and the *meta*-metatheory that we use to reason about the definitional equation of the metatheory. In order to keep track of this hierarchy, we work in a two-level type theory. The inner layer is an observational type theory; this is where our formalisation of the initial CwF and our constructions of gluing models happen. The outer layer is an extensional type theory; we use it to reason about the definitional equality of the inner layer.

We denote the outer universe by Set, and we write $\equiv$ for the equality in the outer layer. This equality validates the reflection rule of extensional type theory, thus it is really the same as the definitional equality of our theory. We denote the inner universe by U : Set, and we equip it with an implicit coercion $\mathsf{U} \to \mathsf{Set}$ (where we omit universe levels for the sake of readability). This inner universe is a model of OTT, and as such it contains a subuniverse of strict propositions $\mathsf{Prop} : \mathsf{U}$ (here, *strict* means that if A : Prop, then for any $a, a' : A$, we have $a \equiv a'$). Every type A in U is equipped with an observational equality relation, which we write as $- =_A - : A \to A \to \mathsf{Prop}$. We write its constructor as refl and transport as $e_* \, u : P \, b$ for $e : a = b$ and $u : P \, a$, with $\mathsf{refl}_* \, u \equiv u$. This inner equality is also called *weak equality*, while the outer equality is called *strict equality*. Both equality types enjoy the definitional uniqueness of identity proofs (UIP). In particular, UIP implies that the J eliminator can be derived from the primitive transport operator. When using either transport or the J eliminator we don't write the family, we don't write congruences (ap or cong operations) and we also don't write symmetries in equality proofs. For example, given $p : b = a$ and $q : f \, b = c$, we write $p \cdot q : f \, a = c$ where $\cdot$ stands for transitivity. We denote the relation between transport and transitivity by $\cdot_* : (e \cdot e')_* \, u = e'_* \, (e_* \, u)$ which is proven by J on e. We write $(a : A) \to B \, a$ for Π types, $(a : A) \times B \, a$ for Σ types and $\top$ for the unit type with constructor tt. All of these have definitional β, η rules. Note that we do not distinguish between the inner Π/Σ types and their outer counterparts, given that they are definitionally isomorphic. We use record types (named iterated Σ types) where projections take the record as a subscript argument, e.g. $\mathsf{Con}_M : \mathsf{U}$ is the context component of an M : CwF. The notation $f : X \cong Y : f^{-1}$ stands for isomorphism in a category where both maps have names. Most of the time this is the category of sets (X, Y : Set) and it means $f : X \to Y$ with $f^{-1} : Y \to X$ with $f^{-1} \, (f \, x) = x$ and $f \, (f^{-1} \, y) = y$ for all x, y.

Sections 5 and 6 were formalised in Agda (this includes the instantiation of Section 4 to prefascist sets). Although Agda does not support OTT natively, it is not too difficult to implement it by taking advantage of the Prop hierarchy and the mechanism for defining custom rewrite rules. We define the observational equality as an inductive equality valued in Prop, and we postulate a type coercion operator $\mathsf{coe} : (A =_{\mathsf{U}} B) \to A \to B$ along with all its reduction rules, following the blueprint in

[Pujet, Leray, et al. 2025]. We also extend Agda with quotient inductive-inductive types (QIITs), by postulating all their constructors, as well as the induction principle and its reduction rules. We use the technique of *fordism* to handle indices. The resulting theory remains fully computational and normalising, which should let us extract executable programs from our gluing proofs. The formalisation is available at [URL removed for reviewing. See anonymised supplementary material], and links to the code will be scattered throughout the paper. Note that there are minor differences between the formal proofs and our presentation – for instance, we found that using a heterogeneous ("John Major") formulation of equality helps with the formalisation, but we stick to a homogeneous equality here.

2 THE INTRINSIC QUOTIENTED SYNTAX OF TYPE THEORY

As explained in the introduction, our goal is to formalise *semantic* proofs for the metatheory of dependent type theory, in particular proofs by gluing. Thus, we abandon the presentation of type theory in terms of raw terms and typing judgements in favour of a presentation based on CwFs. Here, our object theory supports dependent products and booleans with large eliminations, so the CwFs that we consider shall be equipped with Π types and booleans. In this section, we introduce the notions of weak model (based on propositional equalities) and strict model (based on definitional equalities) for the theory of CwFs, and we show why it is difficult to work with weak models. Finally, we will review a handful of concrete model constructions. The last one illustrates the main idea behind the strictification construction in this paper in a very simple setting.

2.1 Categories with families

Definition 1 (CwF, [*model.agda*]). *A weak category with families in* U *(in the inner layer of our two-level metatheory) is defined by the following record, which we denote by* CwF.

Con	$: \mathsf{U}$	$[\circ]$	$: A[\gamma \circ \delta] = A[\gamma][\delta]$
Sub	$: \mathsf{Con} \to \mathsf{Con} \to \mathsf{U}$	$[\mathsf{id}]$	$: A[\mathsf{id}] = A$
Ty	$: \mathsf{Con} \to \mathsf{U}$	$-[-]$	$: \mathsf{Tm}\,\Gamma\,A \to (\gamma : \mathsf{Sub}\,\Delta\,\Gamma) \to \mathsf{Tm}\,\Delta\,(A[\gamma])$
Tm	$: (\Gamma : \mathsf{Con}) \to \mathsf{Ty}\,\Gamma \to \mathsf{U}$	$[\circ]$	$: [\circ]_*\,(a[\gamma \circ \delta]) = a[\gamma][\delta]$
$- \circ -$	$: \mathsf{Sub}\,\Delta\,\Gamma \to \mathsf{Sub}\,\Theta\,\Delta \to \mathsf{Sub}\,\Theta\,\Gamma$	$[\mathsf{id}]$	$: [\mathsf{id}]_*\,(a[\mathsf{id}]) = a$
ass	$: (\gamma \circ \delta) \circ \theta = \gamma \circ (\delta \circ \theta)$	$- \triangleright -$	$: (\Gamma : \mathsf{Con}) \to \mathsf{Ty}\,\Gamma \to \mathsf{Con}$
id	$: \mathsf{Sub}\,\Gamma\,\Gamma$	$-, -$	$: (\gamma : \mathsf{Sub}\,\Delta\,\Gamma) \to \mathsf{Tm}\,\Delta\,(A[\gamma]) \to \mathsf{Sub}\,\Delta\,(\Gamma \triangleright A)$
idl	$: \mathsf{id} \circ \gamma = \gamma$	$,\circ$	$: (\gamma, a) \circ \delta = (\gamma \circ \delta, [\circ]_*\,(a[\delta]))$
idr	$: \gamma \circ \mathsf{id} = \gamma$	p	$: \mathsf{Sub}\,(\Gamma \triangleright A)\,\Gamma$
$\diamond$	$: \mathsf{Con}$	q	$: \mathsf{Tm}\,(\Gamma \triangleright A)\,(A[\mathsf{p}])$
ϵ	$: \mathsf{Sub}\,\Gamma\,\diamond$	$\triangleright\beta_1$	$: \mathsf{p} \circ (\gamma, a) = \gamma$
$\diamond\eta$	$: (\sigma : \mathsf{Sub}\,\Gamma\,\diamond) \to \sigma = \epsilon$	$\triangleright\beta_2$	$: ([\circ] \cdot \triangleright\beta_1)_*\,(\mathsf{q}[\gamma, a]) = a$
$-[-]$	$: \mathsf{Ty}\,\Gamma \to \mathsf{Sub}\,\Delta\,\Gamma \to \mathsf{Ty}\,\Delta$	$\triangleright\eta$	$: \mathsf{id} = (\mathsf{p}, \mathsf{q})$

Note that we overloaded the notation for the substitution operation $-[-]$ and its functor laws for Ty and Tm (the latter refer to the former in the form of a transport on the left). Note also that we make extensive use of implicit arguments: for example, $- \circ -$ takes Γ, Δ, Θ implicitly, p takes Γ and A implicitly, and so on. The constructors p and q let us interpret variables as well-scoped and well-typed De Bruijn indices. The first few are defined as follows:

$$\mathsf{q} : \mathsf{Tm}\,(\Gamma \triangleright A)\,(A[\mathsf{p}]),\ \ \mathsf{q}[\mathsf{p}] : \mathsf{Tm}\,(\Gamma \triangleright A \triangleright B)\,(A[\mathsf{p}][\mathsf{p}]),\ \ \mathsf{q}[\mathsf{p}][\mathsf{p}] : \mathsf{Tm}\,(\Gamma \triangleright A \triangleright B \triangleright C)\,(A[\mathsf{p}][\mathsf{p}][\mathsf{p}]).$$

We introduce substitution lifting and singleton substitutions with the following abbreviations:

$$\begin{aligned}(\gamma : \mathsf{Sub}\,\Delta\,\Gamma)^{\uparrow} &: \mathsf{Sub}\,(\Delta \rhd A[\gamma])\,(\Gamma \rhd A) :\equiv (\gamma \circ \mathsf{p}, [\circ]_*\,\mathsf{q})\\ \langle(a : \mathsf{Tm}\,\Gamma\,A)\rangle &: \mathsf{Sub}\,\Gamma\,(\Gamma \rhd A) \qquad :\equiv (\mathsf{id}, [\mathsf{id}]_*\,a).\end{aligned}$$

We write p^n for the n-times $\mathsf{p} \circ \mathsf{p} \circ \ldots \mathsf{p}$ composition of p. Furthermore, the following three equations can be proved with a simple use of the J eliminator:

(1) given $e : A = A'$ and $a : \mathsf{Tm}\,\Gamma\,A$, we have $(e_*\,a)[\gamma] = e_*\,(a[\gamma])$,
(2) given $e : \Delta = \Delta'$ and $\gamma : \mathsf{Sub}\,\Delta\,\Gamma$ and $\delta : \mathsf{Sub}\,\Theta\,\Delta'$, we have $(e_*\,\gamma) \circ \delta = \gamma \circ e_*\,\delta$,
(3) given $e : A = A'$ and $a : \mathsf{Tm}\,\Gamma\,A$, we have $e_*\,\langle a\rangle = \langle e_*\,a\rangle$.

Lastly, we prove by J on $e' : \gamma = \gamma'$ that $\gamma, a = \gamma', e_*\,a$. We consider this equation as a congruence rule and we apply it implicitly when needed.

Definition 2 ($\overline{\mathsf{CwF}}$). *A strict category with families is a CwF for which all equations are definitional (i.e., stated using ≡ instead of =), except for $\rhd\eta$. As a consequence, all the transports which appear in the definition of a CwF become redundant. For example, we have $[\mathsf{id}] : A[\mathsf{id}] \equiv A$ for all $A : \mathsf{Ty}\,\Gamma$, and thus given an $a : \mathsf{Tm}\,\Gamma\,A$, we have $a[\mathsf{id}] : \mathsf{Tm}\,\Gamma\,(A[\mathsf{id}])$, so by equality reflection, $a[\mathsf{id}] : \mathsf{Tm}\,\Gamma\,A$, meaning that the $[\mathsf{id}]$ law for terms can be stated as $a[\mathsf{id}] \equiv a$. We denote the corresponding record type by $\overline{\mathsf{CwF}}$. Since it uses the strict equality, this record lives in the outer layer of our two-level metatheory, and has no counterpart in the Agda formalisation.*

Definition 3 (Π types, [*model.agda*]). *We say that a* CwF *supports weak dependent products if it is equipped with the following additional components. The extended record type is denoted by* CwF_{Π}.

$$\begin{aligned}\Pi &: (A : \mathsf{Ty}\,\Gamma) \to \mathsf{Ty}\,(\Gamma \rhd A) \to \mathsf{Ty}\,\Gamma & \mathsf{app} &: \mathsf{Tm}\,\Gamma\,(\Pi\,A\,B) \to \mathsf{Tm}\,(\Gamma \rhd A)\,B\\ \Pi[] &: (\Pi\,A\,B)[\gamma] = \Pi\,(A[\gamma])\,(B[\gamma^{\uparrow}]) & \mathsf{app}[] &: (\mathsf{app}\,t)[\gamma^{\uparrow}] = \mathsf{app}\,(\Pi[]_*\,(t[\gamma]))\\ \mathsf{lam} &: \mathsf{Tm}\,(\Gamma \rhd A)\,B \to \mathsf{Tm}\,\Gamma\,(\Pi\,A\,B) & \Pi\beta &: \mathsf{app}\,(\mathsf{lam}\,t) = t\\ \mathsf{lam}[] &: \Pi[]_*\,((\mathsf{lam}\,t)[\gamma]) = \mathsf{lam}\,(t[\gamma^{\uparrow}]) & \Pi\eta &: \mathsf{lam}\,(\mathsf{app}\,t) = t\end{aligned}$$

We introduce two counterparts to this definition in the case of strict CwFs. The most straightforward one is fully strict Π types, for which all 5 equations are definitional. We signify that a strict CwF is equipped with fully strict Π types by using the subscript $_{\overline{\Pi}}$. *We also introduce the weaker notion of substitution-strict Π types, for which only the* $\Pi[]$, $\mathsf{lam}[]$ *and* $\mathsf{app}[]$ *laws are strict, while* $\Pi\beta$ *and* $\Pi\eta$ *remain weak. We denote substitution-strict Π types by using the subscript* $_{\tilde{\Pi}}$. *For example, a* $\overline{\mathsf{CwF}}_{\overline{\Pi}}$ *is a strict CwF with fully strict Π types, a* $\overline{\mathsf{CwF}}_{\tilde{\Pi}}$ *is a strict CwF with substitution-strict Π types, and a* CwF_{Π} *is a weak CwF with weak Π types.*

Definition 4 (Bool, [*model.agda*]). *A* CwF *supports weak booleans with large elimination when it is equipped with the following additional components. The extended record is denoted by* $\mathsf{CwF}_{\mathsf{Bool}}$.

$$
\begin{aligned}
&\mathsf{Bool} &&: \mathsf{Ty}\,\Gamma && \mathsf{ite}^{\mathsf{T}} &&: \mathsf{Ty}\,\Gamma \to \mathsf{Ty}\,\Gamma \to \mathsf{Tm}\,\Gamma\,\mathsf{Bool} \to \mathsf{Ty}\,\Gamma\\
&\mathsf{Bool}[] &&: \mathsf{Bool}[\gamma] = \mathsf{Bool} && \mathsf{Bool}\beta_1^{\mathsf{T}} &&: \mathsf{ite}^{\mathsf{T}}\,A\,A'\,\mathsf{true} = A\\
&\mathsf{true} &&: \mathsf{Tm}\,\Gamma\,\mathsf{Bool} && \mathsf{Bool}\beta_2^{\mathsf{T}} &&: \mathsf{ite}^{\mathsf{T}}\,A\,A'\,\mathsf{false} = A'\\
&\mathsf{true}[] &&: \mathsf{Bool}[]_*\,(\mathsf{true}[\gamma]) = \mathsf{true} && \mathsf{Bool}\beta_1^{\mathsf{t}} &&: \mathsf{ite}^{\mathsf{t}}\,P\,p\,p'\,\mathsf{true} = p\\
&\mathsf{false} &&: \mathsf{Tm}\,\Gamma\,\mathsf{Bool} && \mathsf{Bool}\beta_2^{\mathsf{t}} &&: \mathsf{ite}^{\mathsf{t}}\,P\,p\,p'\,\mathsf{false} = p'\\
&\mathsf{false}[] &&: \mathsf{Bool}[]_*\,(\mathsf{false}[\gamma]) = \mathsf{false}
\end{aligned}
$$

$$
\begin{aligned}
\mathsf{ite}^{\mathsf{t}} \quad &: (P : \mathsf{Ty}\,(\Gamma \triangleright \mathsf{Bool})) \to \mathsf{Tm}\,\Gamma\,(P[\langle\mathsf{true}\rangle]) \to \mathsf{Tm}\,\Gamma\,(P[\langle\mathsf{false}\rangle]) \to\\
&\quad (b : \mathsf{Tm}\,\Gamma\,\mathsf{Bool}) \to \mathsf{Tm}\,\Gamma\,(P[\langle b\rangle])\\
\mathsf{ite}^{\mathsf{T}}[] \quad &: (\mathsf{ite}^{\mathsf{T}}\,A\,A'\,b)[\gamma] = \mathsf{ite}^{\mathsf{T}}\,(A[\gamma])\,(A'[\gamma])\,(\mathsf{Bool}[]_*\,(b[\gamma]))\\
\mathsf{ite}^{\mathsf{t}}[] \quad &: (\alpha\,b)_*\,((\mathsf{ite}^{\mathsf{t}}\,P\,p\,p'\,b)[\gamma]) =\\
&\quad \mathsf{ite}^{\mathsf{t}}\,(\mathsf{Bool}[]_*\,(P[\gamma^{\uparrow}]))\,(\mathsf{true}[]_*\,((\alpha\,\mathsf{true})_*\,p))\,(\mathsf{true}[]_*\,((\alpha\,\mathsf{false})_*\,p'))\,(\mathsf{Bool}[]_*\,(b[\gamma]))
\end{aligned}
$$

In equation $\mathsf{ite}^{\mathsf{t}}[]$, *we used* $\alpha\,(u : \mathsf{Tm}\,\Gamma\,\mathsf{Bool}) : P[\langle u\rangle][\gamma] = P[\mathsf{Bool}[]_*\,(\gamma^{\uparrow})][\langle\mathsf{Bool}[]_*\,(u[\gamma])\rangle]$ *which we prove in Figure 1 in the Appendix. In the case of strict CwFs, we distinguish between strict booleans, for which all 9 equations are definitional, and substitution-strict booleans for which only the 5 substitution laws are strict. These are denoted by* $_{\overline{\mathsf{Bool}}}$ *and* $_{\overset{\text{-\,-\,-}}{\mathsf{Bool}}}$ *subscripts, respectively.*

2.2 Weak models vs. strict models in practice

The weak models are the only ones that we can define in the inner layer of our two-level type theory or in our Agda formalisation, but unfortunately they are especially prone to transport hell. In this subsection, we look at a few examples that illustrate the difference between weak and substitution-strict models (more precisely, between a $\mathsf{CwF}_{\Pi,\mathsf{Bool}}$ and a $\overline{\mathsf{CwF}}_{\overset{\text{-\,-\,-}}{\Pi,\mathsf{Bool}}}$). For our first example, recall that we defined function application as an inverse to the lam operator. We can define a more traditional binary application and prove its β law as follows:

$$
\begin{aligned}
&- \bullet - : \mathsf{Tm}\,\Gamma\,(\Pi\,A\,B) \to (a : \mathsf{Tm}\,\Gamma\,A) \to \mathsf{Tm}\,\Gamma\,(B[\langle a\rangle])\\
&f \bullet a :\equiv (\mathsf{app}\,f)[\langle a\rangle]\\
&\bullet\beta : (\mathsf{lam}\,b) \bullet a \equiv (\mathsf{app}\,(\mathsf{lam}\,b))[\langle a\rangle] \overset{\Pi\beta}{=} b[\langle a\rangle]
\end{aligned}
$$

So far, there is no difference between working in a weak or substitution-strict model.

Problem 5. *The* η *law for binary application can be proved in both* $\mathsf{CwF}_{\Pi,\mathsf{Bool}}$ *and* $\overline{\mathsf{CwF}}_{\overset{\text{-\,-\,-}}{\Pi,\mathsf{Bool}}}$.

Statement and proof of the η law in a $\mathsf{CwF}_{\Pi,\mathsf{Bool}}$. First of all, we have to express in our object theory what we write in the metatheory as $\lambda x.f\,x$. Given an $f : \mathsf{Tm}\,\Gamma\,(\Pi\,A\,B)$, we weaken it to $f[\mathsf{p}] : \mathsf{Tm}\,(\Gamma \triangleright A)\,((\Pi\,A\,B)[\mathsf{p}])$ so that we can apply it to $\mathsf{q} : \mathsf{Tm}\,(\Gamma \triangleright A)\,(A[\mathsf{p}])$ in the same context (q is the zero De Bruijn index). In a $\mathsf{CwF}_{\Pi,\mathsf{Bool}}$, we cannot use binary application on $f[\mathsf{p}]$ because it does not have a Π type, it has an instantiated type. So we transport it along the substitution rule $\Pi[]$. Now we can write $\mathsf{lam}\,(\Pi[]_*\,(f[\mathsf{p}]) \bullet \mathsf{q})$, but this is a $\mathsf{Tm}\,\Gamma\,(\Pi\,A\,(B[\mathsf{p}^{\uparrow}][\langle\mathsf{q}\rangle]))$ rather than a

$\mathsf{Tm}\,\Gamma\,(\Pi\,A\,B)$. Firstly, we show

$$\begin{aligned}
&\mathsf{p}^{\uparrow}\circ\langle\mathsf{q}\rangle && \equiv \\
&(\mathsf{p}\circ\mathsf{p},[\circ]_*\,\mathsf{q})\circ\langle\mathsf{q}\rangle && =(,\circ) \\
&((\mathsf{p}\circ\mathsf{p})\circ\langle\mathsf{q}\rangle,[\circ]_*\,(([\circ]_*\,\mathsf{q})[\langle\mathsf{q}\rangle])) && =(1) \\
&((\mathsf{p}\circ\mathsf{p})\circ\langle\mathsf{q}\rangle,[\circ]_*\,([\circ]_*\,(\mathsf{q}[\langle\mathsf{q}\rangle]))) && =(\mathsf{ass}) \\
&(\mathsf{p}\circ(\mathsf{p}\circ\langle\mathsf{q}\rangle),\mathsf{ass}_*\,([\circ]_*\,([\circ]_*\,(\mathsf{q}[\langle\mathsf{q}\rangle])))) && =(\triangleright\beta_1) \\
&(\mathsf{p}\circ\mathsf{id},\triangleright\beta_{1*}\,(\mathsf{ass}_*\,([\circ]_*\,([\circ]_*\,(\mathsf{q}[\langle\mathsf{q}\rangle]))))) && =(\mathsf{idr}) \\
&(\mathsf{p},\mathsf{idr}_*\,(\triangleright\beta_{1*}\,(\mathsf{ass}_*\,([\circ]_*\,([\circ]_*\,(\mathsf{q}[\langle\mathsf{q}\rangle])))))) && =(\cdot_*) \\
&(\mathsf{p},([\circ]\cdot[\circ]\cdot\mathsf{ass}\cdot\triangleright\beta_1\cdot\mathsf{idr})_*\,(\mathsf{q}[\langle\mathsf{q}\rangle])) && \equiv \\
&(\mathsf{p},([\circ]\cdot[\circ]\cdot\mathsf{ass}\cdot\triangleright\beta_1\cdot\mathsf{idr}\cdot[\mathsf{id}]\cdot[\mathsf{id}])_*\,(\mathsf{q}[\langle\mathsf{q}\rangle])) && =(\cdot_*) \\
&(\mathsf{p},[\mathsf{id}]_*\,(([\circ]\cdot[\circ]\cdot\mathsf{ass}\cdot\triangleright\beta_1\cdot\mathsf{idr}\cdot[\mathsf{id}])_*\,(\mathsf{q}[\langle\mathsf{q}\rangle]))) && \equiv \\
&(\mathsf{p},[\mathsf{id}]_*\,(([\circ]\cdot\triangleright\beta_1)_*\,(\mathsf{q}[\langle\mathsf{q}\rangle]))) && =(\triangleright\beta_2) \\
&(\mathsf{p},[\mathsf{id}]_*\,([\mathsf{id}]_*\,\mathsf{q})) && =(\cdot_*) \\
&(\mathsf{p},([\mathsf{id}]\cdot[\mathsf{id}])_*\,\mathsf{q}) && \equiv \\
&(\mathsf{p},\mathsf{q}) && =(\triangleright\eta) \\
&\mathsf{id},
\end{aligned} \tag{4}$$

and now we can state the η law for traditional application as

$$\bullet\eta : ([\circ]\cdot(4)\cdot[\mathsf{id}])_*\,(\mathsf{lam}\,(\Pi[]_*\,(f[\mathsf{p}])\bullet\mathsf{q})) = f.$$

Note that we had to work quite hard just to be able to *state* the η law. To prove it, we first show

$$e_*\,(\mathsf{lam}\,b) = \mathsf{lam}\,(e_*\,b) \tag{5}$$

by J on $e : B = B'$ for any $b : \mathsf{Tm}\,(\Gamma\triangleright A)\,B$. Now we reason as follows:

$$\begin{aligned}
\bullet\eta :\ &([\circ]\cdot(4)\cdot[\mathsf{id}])_*\,(\mathsf{lam}\,(\Pi[]_*\,(f[\mathsf{p}])\bullet\mathsf{q})) && \equiv \\
&([\circ]\cdot(4)\cdot[\mathsf{id}])_*\,(\mathsf{lam}\,((\mathsf{app}\,(\Pi[]_*\,(f[\mathsf{p}])))[\langle\mathsf{q}\rangle])) && =(\mathsf{app}[]) \\
&([\circ]\cdot(4)\cdot[\mathsf{id}])_*\,(\mathsf{lam}\,((\mathsf{app}\,f)[\mathsf{p}^{\uparrow}][\langle\mathsf{q}\rangle])) && =([\circ]) \\
&([\circ]\cdot(4)\cdot[\mathsf{id}])_*\,(\mathsf{lam}\,([\circ]_*\,((\mathsf{app}\,f)[\mathsf{p}^{\uparrow}\circ\langle\mathsf{q}\rangle]))) && =(4) \\
&([\circ]\cdot(4)\cdot[\mathsf{id}])_*\,(\mathsf{lam}\,([\circ]_*\,((4)_*\,((\mathsf{app}\,f)[\mathsf{id}])))) && =(\cdot_*) \\
&([\circ]\cdot(4)\cdot[\mathsf{id}])_*\,(\mathsf{lam}\,((4)\cdot[\circ])_*\,((\mathsf{app}\,f)[\mathsf{id}])) && =(5) \\
&\mathsf{lam}\,(([\circ]\cdot(4)\cdot[\mathsf{id}])_*\,((4)\cdot[\circ])_*\,((\mathsf{app}\,f)[\mathsf{id}])) && =(\cdot_*) \\
&\mathsf{lam}\,(((4)\cdot[\circ]\cdot[\circ]\cdot(4))_*\,([\mathsf{id}]_*\,((\mathsf{app}\,f)[\mathsf{id}]))) && \equiv \\
&\mathsf{lam}\,([\mathsf{id}]_*\,((\mathsf{app}\,f)[\mathsf{id}])) && =([\mathsf{id}]) \\
&\mathsf{lam}\,(\mathsf{app}\,f) && =(\Pi\eta) \\
&f
\end{aligned}$$

□

Statement and proof of the η law in a $\overline{\mathsf{CwF}}_{\underset{\cdots\cdots}{\Pi,\mathsf{Bool}}}$. When stating $\bullet\eta$, there is no need to transport $f[\mathsf{p}]$ when η-expanding f because $\Pi[]$ is definitional. The type of $\mathsf{lam}\,(f[\mathsf{p}] \bullet \mathsf{q})$ is

$$\begin{aligned}
&\mathsf{Tm}\,\Gamma\,(\Pi\,A\,(B[\mathsf{p}^{\uparrow}][\langle\mathsf{q}\rangle])) && \equiv((4), \text{without the last step which uses the weak} \rhd\eta)\\
&\mathsf{Tm}\,\Gamma\,(\Pi\,A\,(B[\mathsf{p},\mathsf{q}])) && \equiv([\mathsf{id}])\\
&\mathsf{Tm}\,\Gamma\,(\Pi\,(A[\mathsf{id}])\,(B[\mathsf{p},\mathsf{q}])) && \equiv(\mathsf{idl})\\
&\mathsf{Tm}\,\Gamma\,(\Pi\,(A[\mathsf{id}])\,(B[\mathsf{id}\circ\mathsf{p},\mathsf{q}])) && \equiv\\
&\mathsf{Tm}\,\Gamma\,(\Pi\,(A[\mathsf{id}])\,(B[\mathsf{id}^{\uparrow}])) && \equiv(\Pi[])\\
&\mathsf{Tm}\,\Gamma\,((\Pi\,A\,B)[\mathsf{id}]) && \equiv([\mathsf{id}])\\
&\mathsf{Tm}\,\Gamma\,(\Pi\,A\,B),
\end{aligned}$$

so the statement is simply $\mathsf{lam}\,(f[\mathsf{p}] \bullet \mathsf{q}) = f$. Using similar reasoning, its proof is just $\Pi\eta$. $\square$

Remark 6. *It is possible to state and prove* $\bullet\eta$ *in a* $\mathsf{CwF}_{\Pi,\mathsf{Bool}}$ *without relying on* $\rhd\eta$*: we prove* $e_*\,(\mathsf{id}\circ\mathsf{p}\,\{A\}, \mathsf{idl}_*\,(\mathsf{q}\,\{A\})) = (\mathsf{id}\circ(\mathsf{p}\,\{A'\}), (e\cdot\mathsf{idl})_*\,(\mathsf{q}\,\{A'\}))$ *by* J *on* $e : A = A'$*; this implies that* $[\mathsf{id}]_*\,(\mathsf{p},\mathsf{q}) = \mathsf{id}^{\uparrow}$*; now we can show that* $\Pi\,A\,(B[\mathsf{p},\mathsf{q}])$ *is the same as* $\Pi\,(A[\mathsf{id}])\,(B[\mathsf{id}^{\uparrow}])$*, which by* $\Pi[]$ *and* $[\mathsf{id}]$ *is the same as* $\Pi\,A\,B$*. However lots of transport trickery is needed to make this precise.*

For our second example, we use the non-dependent function type $A \Rightarrow B :\equiv \Pi\,A\,(B[\mathsf{p}])$ for $A, B : \mathsf{Ty}\,\Gamma$. In a $\mathsf{CwF}_{\Pi,\mathsf{Bool}}$, the substitution law for $\Rightarrow$ can be proven as follows:

$$(A \Rightarrow B)[\gamma] \equiv (\Pi\,A\,(B[\mathsf{p}]))[\gamma] \overset{\Pi[]}{=} \Pi\,(A[\gamma])\,(B[\mathsf{p}][\gamma^{\uparrow}]) \overset{[\circ]}{=} \Pi\,(A[\gamma])\,(B[\mathsf{p}\circ(\gamma^{\uparrow})]) \overset{\rhd\beta_1}{=}$$
$$\Pi\,(A[\gamma])\,(B[\gamma\circ\mathsf{p}]) \overset{[\circ]}{=} \Pi\,(A[\gamma])\,(B[\gamma][\mathsf{p}]) \equiv (A[\gamma]) \Rightarrow (B[\gamma])$$

while in a $\overline{\mathsf{CwF}}_{\underset{\cdots\cdots}{\Pi,\mathsf{Bool}}}$, we have definitionally $(A \Rightarrow B)[\gamma] \equiv (A[\gamma]) \Rightarrow (B[\gamma])$.

The most dramatic example of transport hell in a $\mathsf{CwF}_{\Pi,\mathsf{Bool}}$ is already there in Definition 4: we need the proof in Figure 1 to even *state* the substitution law for $\mathsf{ite}^{\mathsf{t}}$. Out of the 18 steps in the equational reasoning proof, exactly half are $\mathsf{CwF}_{\Pi,\mathsf{Bool}}$-equations, while the other half are transport-moving equations such as the relationship of transitivity and transport $(\cdot_*)$ and equations commuting transport with instantiation (1), composition (2), single substitution (3). Coming up with this proof consumes lots of brainpower, but no insight comes from delivering it. In contrast, in a $\overline{\mathsf{CwF}}_{\underset{\cdots\cdots}{\Pi,\mathsf{Bool}}}$, the same equation holds definitionally.

As a last example, we define the nondependent eliminator for booleans

$$\mathsf{ifthenelse} : \mathsf{Tm}\,\Gamma\,\mathsf{Bool} \to \mathsf{Tm}\,\Gamma\,A \to \mathsf{Tm}\,\Gamma\,A \to \mathsf{Tm}\,\Gamma\,A$$

in a $\mathsf{CwF}_{\Pi,\mathsf{Bool}}$ as

$$\mathsf{ifthenelse}\,b\,a\,a' :\equiv ([\circ]\cdot\rhd\beta_1\cdot[\mathsf{id}])_*\,(\mathsf{ite}^{\mathsf{t}}\,(A[\mathsf{p}])\,(([\mathsf{id}]\cdot\rhd\beta_1\cdot[\circ])_*\,a)\,(([\mathsf{id}]\cdot\rhd\beta_1\cdot[\circ])_*\,a')\,b)$$

and in a $\overline{\mathsf{CwF}}_{\underset{\cdots\cdots}{\Pi,\mathsf{Bool}}}$ as $\mathsf{ifthenelse}\,b\,a\,a' :\equiv \mathsf{ite}^{\mathsf{t}}\,(A[\mathsf{p}])\,a\,a'\,b$. Proving the substitution law $(\mathsf{ifthenelse}\,b\,a\,a')[\gamma] = \mathsf{ifthenelse}\,(\mathsf{Bool}[]_*\,(b[\gamma]))\,(a[\gamma])\,(a'[\gamma])$ in a $\mathsf{CwF}_{\Pi,\mathsf{Bool}}$ is a great challenge for weak CwFs enthusiasts, but we won't attempt it here. This equation is in fact definitional in a $\overline{\mathsf{CwF}}_{\underset{\cdots\cdots}{\Pi,\mathsf{Bool}}}$.

2.3 Examples of weak and strict models

Now that we have a definition of model, we look at some concrete instances. We will first show a completely weak $\mathsf{CwF}_{\Pi,\mathsf{Bool}}$, then a strict $\overline{\mathsf{CwF}}_{\underline{\Pi,\mathsf{Bool}}}$, then finally a substitution-strict $\overline{\mathsf{CwF}}_{\underset{\cdots\cdots}{\Pi,\mathsf{Bool}}}$.

Definition 7 (I, [*initial.agda*]). *The syntax of our object theory is a* $\mathsf{CwF}_{\Pi,\mathsf{Bool}}$.

Construction. We define the syntax (noted I) as a quotient inductive-inductive type (QIIT) whose constructors are precisely the fields in the record $\mathsf{CwF}_{\Pi,\mathsf{Bool}}$. The induction principle that we get by defining I as a QIIT states that any $\mathsf{CwF}_{\Pi,\mathsf{Bool}}$ indexed over I has a section, which is equivalent to the initiality of I. Note that for this definition to make sense, we need to be able to define this QIIT inside of U. Observational type theory already supports quotient types [Pujet and Tabareau 2022], and furthermore QIITs are supported by the setoid model [Kaposi and Z. Xie 2021] which is the "standard" model for OTT. Thus, the proof of normalisation for OTT [Pujet and Tabareau 2022] should be extensible to cover this particular QIIT, and we feel justified in adding it to our internal layer (and to our formalisation as well). □

Definition 8. *The standard model is a* $\overline{\mathsf{CwF}}_{\overline{\Pi,\mathsf{Bool}}}$ *where even* $\triangleright\eta$ *is definitional.*

Construction. The so-called standard model uses the metatheory as a model. For this reason, it is also known as the *metacircular* model, or the *type* model. It is defined as follows:

$$\mathsf{Con} :\equiv \mathsf{U} \qquad \mathsf{Sub}\,\Delta\,\Gamma :\equiv \Delta \to \Gamma \qquad \mathsf{Ty}\,\Gamma :\equiv \Gamma \to \mathsf{U} \qquad \mathsf{Tm}\,\Gamma\,A :\equiv (\gamma_\bullet : \Gamma) \to A\,\gamma_\bullet$$

Since the substitutions are interpreted as OTT functions, they form a strict category. Dependency on contexts is implemented via function space and instantiation is function composition, so we get definitional functor laws.

$$(\gamma \circ \delta)\,\theta_\bullet :\equiv \gamma\,(\delta\,\theta_\bullet) \qquad \mathsf{id} :\equiv \lambda\gamma_\bullet.\,\gamma_\bullet \qquad A[\gamma]\,\delta_\bullet :\equiv A\,(\gamma\,\delta_\bullet) \qquad a[\gamma]\,\delta_\bullet :\equiv a\,(\gamma\,\delta_\bullet)$$

The empty context is interpreted as OTT's unit type, which needs its definitional η law to implement $\diamond\eta$. Context extension is interpreted by OTT's Σ types, and the corresponding combinators come from the constructor and destructors of Σ:

$$\Gamma \triangleright A :\equiv (\gamma_\bullet : \Gamma) \times A\,\gamma_\bullet \qquad (\gamma, a)\,\delta_\bullet :\equiv (\gamma\,\delta_\bullet, a\,\delta_\bullet) \qquad \mathsf{p}\,(\gamma_\bullet, a_\bullet) :\equiv \gamma_\bullet \qquad \mathsf{q}\,(\gamma_\bullet, a_\bullet) :\equiv a_\bullet$$

Thus we get that the $\triangleright\beta$ and $\triangleright\eta$ laws are definitional. As an illustration, we derive the latter:

$$(\mathsf{p}, \mathsf{q}) \equiv \lambda(\gamma_\bullet, a_\bullet).\,(\mathsf{p}, \mathsf{q})\,(\gamma_\bullet, a_\bullet) \equiv \lambda(\gamma_\bullet, a_\bullet).\,(\mathsf{p}\,(\gamma_\bullet, a_\bullet), \mathsf{q}\,(\gamma_\bullet, a_\bullet)) \equiv \lambda(\gamma_\bullet, a_\bullet).\,(\gamma_\bullet, a_\bullet) \equiv \mathsf{id}.$$

Dependent products and booleans are directly inherited from their metatheoretical counterparts: for example, $\Pi\,A\,B\,\gamma_\bullet :\equiv (a_\bullet : A\,\gamma_\bullet) \to B(\gamma_\bullet, a_\bullet)$, and the substitution rule $\Pi[]$ holds via $(\Pi\,A\,B)[\gamma]\,\delta_\bullet \equiv \Pi\,A\,B\,(\gamma\,\delta_\bullet) \equiv (a_\bullet : A\,(\gamma\,\delta_\bullet)) \to B\,(\gamma\,\delta_\bullet, a_\bullet) \equiv (a_\bullet : A[\gamma]\,\delta_\bullet) \to B[\gamma^\uparrow]\,(\gamma_\bullet, a_\bullet) \equiv \Pi\,(A[\gamma])\,(B[\gamma^\uparrow])\,\delta_\bullet$. □

The construction of the standard model is possible thanks to the internal universe U of our metatheory, which is closed under dependent products and booleans. This encourages us to consider a generalisation of this construction which is parameterised by an arbitrary universe. The following definition is called a higher-order model in [Bocquet et al. 2023].

Definition 9 (A universe closed under Π and Bool). *A universe inside* U *closed under* Π *and* Bool *contains the following components. We denote the type of codes in the universe by* Ty *and its elements function (usually called* El*) by* Tm*. We use brick red colour to disambiguate. Note that the* *equations for* Π *and* Bool *are weak.*

$$\begin{array}{lll@{\qquad}lll}
\mathsf{Ty} & : & \mathsf{U} & \mathsf{false} & : & \mathsf{Tm}\,\mathsf{Bool} \\
\mathsf{Tm} & : & \mathsf{Ty} \to \mathsf{U} & \mathsf{ifTy} & : & \mathsf{Ty} \to \mathsf{Ty} \to \mathsf{Tm}\,\mathsf{Bool} \to \mathsf{Ty} \\
\Pi & : & (A : \mathsf{Ty}) \to (\mathsf{Tm}\,A \to \mathsf{Ty}) \to \mathsf{Ty} & \mathsf{ifTy}\beta_1 & : & \mathsf{ifTy}\,A\,B\,\mathsf{true} = A \\
\mathsf{lam} & : & ((a : \mathsf{Tm}\,A) \to \mathsf{Tm}\,(B\,a)) \to \mathsf{Tm}\,(\Pi\,A\,B) & \mathsf{ifTy}\beta_2 & : & \mathsf{ifTy}\,A\,B\,\mathsf{false} = B \\
\mathsf{app} & : & \mathsf{Tm}\,(\Pi\,A\,B) \to (a : \mathsf{Tm}\,A) \to \mathsf{Tm}\,(B\,a) & \mathsf{ifTm} & : & (P : \mathsf{Tm}\,\mathsf{Bool} \to \mathsf{Ty}) \to \\
\Pi\beta & : & \mathsf{app}\,(\mathsf{lam}\,t) = t & & & \mathsf{Tm}\,(P\,\mathsf{true}) \to \mathsf{Tm}\,(P\,\mathsf{false}) \to \\
\Pi\eta & : & \mathsf{lam}\,(\mathsf{app}\,t) = t & & & (b : \mathsf{Tm}\,\mathsf{Bool}) \to \mathsf{Tm}\,(P\,b) \\
\mathsf{Bool} & : & \mathsf{Ty} & \mathsf{ifTm}\beta_1 & : & \mathsf{ifTm}\,P\,u\,v\,\mathsf{true} = u \\
\mathsf{true} & : & \mathsf{Tm}\,\mathsf{Bool} & \mathsf{ifTm}\beta_1 & : & \mathsf{ifTm}\,P\,u\,v\,\mathsf{false} = v
\end{array}$$

The next model construction illustrates our strictification method in a simple setting: equipping a weak universe with a strict substitution calculus coming from the metatheory. In Problem 14, the strict substitution calculus will come from another model. The following construction is called set-contextualisation in [Bocquet et al. 2023].

Definition 10 (U-contextualisation). *Given a universe* Ty–Tm *as in Definition 9, its* U*-contextualisation is a* $\overline{\mathsf{CwF}}_{\underline{\Pi,\mathsf{Bool}}}$.

Construction. The substitution calculus is defined exactly like in the standard model, except that types and terms come from the universe Ty–Tm and note the appearance of Tm in $\Gamma \triangleright A$:

$$\mathsf{Con} :\equiv \mathsf{U} \qquad \mathsf{Sub}\,\Delta\,\Gamma :\equiv \Delta \to \Gamma \qquad \mathsf{Ty}\,\Gamma :\equiv \Gamma \to \mathsf{Ty} \qquad \mathsf{Tm}\,\Gamma\,A :\equiv (\gamma_\bullet : \Gamma) \to \mathsf{Tm}\,(A\,\gamma_\bullet)$$

$$(\gamma \circ \delta)\,\theta_\bullet :\equiv \gamma\,(\delta\,\theta_\bullet) \qquad \mathsf{id} :\equiv \lambda\gamma_\bullet.\,\gamma_\bullet \qquad A[\gamma]\,\delta_\bullet :\equiv A\,(\gamma\,\delta_\bullet) \qquad a[\gamma]\,\delta_\bullet :\equiv a\,(\gamma\,\delta_\bullet)$$

$$\Gamma \triangleright A :\equiv (\gamma_\bullet : \Gamma) \times \mathsf{Tm}\,(A\,\gamma_\bullet) \qquad (\gamma, a)\,\delta_\bullet :\equiv (\gamma\,\delta_\bullet, a\,\delta_\bullet) \qquad \mathsf{p}\,(\gamma_\bullet, a_\bullet) :\equiv \gamma_\bullet \qquad \mathsf{q}\,(\gamma_\bullet, a_\bullet) :\equiv a_\bullet$$

Π types come from Π-closure of the universe, but context dependency is still OTT-function space, hence we get weak $\Pi\beta$, but strict substitution rules.

$$\Pi\,A\,B\,\gamma_\bullet :\equiv \Pi\,(A\,\gamma_\bullet)\,(\lambda a_\bullet.B\,(\gamma_\bullet, a_\bullet)) \quad \mathsf{app}\,t\,(\gamma_\bullet, a_\bullet) :\equiv \mathsf{app}\,(t\,\gamma_\bullet)\,a_\bullet \quad \mathsf{lam}\,t\,\gamma_\bullet :\equiv \mathsf{lam}\,(\lambda a_\bullet.t\,(\gamma_\bullet, a_\bullet))$$

$$\begin{aligned}
\Pi\beta : \;& \mathsf{app}\,(\mathsf{lam}\,t)\,(\gamma_\bullet, a_\bullet) \equiv \mathsf{app}\,(\mathsf{lam}\,t\,\gamma_\bullet)\,a_\bullet \equiv \mathsf{app}\,(\mathsf{lam}\,(\lambda a_\bullet.t\,(\gamma_\bullet, a_\bullet)))\,a_\bullet \overset{\Pi\beta}{=} (\lambda a_\bullet.t\,(\gamma_\bullet, a_\bullet))\,a_\bullet \equiv \\
& t\,(\gamma_\bullet, a_\bullet) \\
\Pi[] : \;& (\Pi\,A\,B)[\gamma]\,\delta_\bullet \equiv \Pi\,A\,B\,(\gamma\,\delta_\bullet) \equiv \Pi\,(A\,(\gamma\,\delta_\bullet))\,(\lambda a_\bullet.B\,(\gamma\,\delta_\bullet, a_\bullet)) \equiv \\
& \Pi\,(A[\gamma]\,\delta_\bullet)\,(\lambda a_\bullet.B[\gamma^\uparrow]\,(\gamma_\bullet, a_\bullet)) \equiv \Pi\,(A[\gamma])\,(B[\gamma^\uparrow])\,\delta_\bullet
\end{aligned}$$

Booleans are defined analogously e.g. $\mathsf{Bool}\,\gamma_\bullet :\equiv \mathsf{Bool}$. □

The standard model is the special case of U-contextualisation when using the universe from the metatheory $\mathsf{Ty} :\equiv \mathsf{U}$ and $\mathsf{Tm}\,A :\equiv A$.

3 PRESHEAVES

This section is a warm up for our general strictification construction in Section 4. We will replace the substitution calculus in the syntax of type theory by one that comes from presheaves. Although, for presentation purposes, we only do this in a setting with equality reflection, it illustrates how the construction would work in an intensional setting if the presheaf model was stricter. That will be the topic of the next two sections. In this section, first we define the presheaf model in our OTT-setting, and we observe that it does not support strict Π types. Then we switch to a setting with equality reflection and we redefine the syntax of our object theory inside the presheaf model, and use this to obtain another, isomorphic model.

Problem 11 (Presheaf model over C, $\mathsf{PSh}(C)$). *Given a weak category C (the category part of a CwF), presheaves over C form a $\overline{\mathsf{CwF}}$ where even $\triangleright\eta$ is strict.*

Construction. The presheaf model of type theory was introduced by [Hofmann 1997]. Here we analyse it from a strictness perspective, so we only show parts relevant for this.

Contexts are presheaves over C, substitutions are natural transformations:

$$\begin{aligned}
\mathsf{Con} \quad &:\equiv (\Gamma : C \to \mathsf{U}) \times (-[-]_\Gamma : \Gamma\, I \to C(J,I) \to \Gamma\, J) \times \\
&\qquad ([\circ]_\Gamma : \gamma_I[f \circ g]_\Gamma = \gamma_I[f]_\Gamma[g]_\Gamma) \times ([\mathsf{id}]_\Gamma : \gamma_I[f]_\Gamma = \gamma_I) \\
\mathsf{Sub}\,\Delta\,\Gamma &:\equiv (\gamma : (\delta_I : \Delta\, I) \to \Gamma\, I) \times (-[-]_\gamma : (\delta_I : \Delta\, I)(f : C(J,I)) \to (\gamma\,\delta_I)[f]_\Gamma = \gamma\,(\delta_I[f]_\Delta))
\end{aligned}$$

We overload Γ both for the element of Con and its first component, and similarly for Sub. The second component $-[-]_\Gamma$ of a presheaf Γ is called *restriction*. As we express naturality using Prop-valued equalities, when comparing two natural transformations for definitional equality, only their function parts matter, hence we get a strict category. For example, composition of substitutions is definitionally associative:

$$(\gamma \circ \delta)\,\theta_I :\equiv \gamma\,(\delta\,\theta_I) \qquad\qquad (\gamma \circ \delta) \circ \theta \equiv \lambda\xi_*.\,\gamma\,(\delta\,(\theta\,\xi_*)) \equiv \gamma \circ (\delta \circ \theta)$$

The empty context is the constant unit presheaf defined as $\diamond\,\Gamma :\equiv \top$. Types are dependent presheaves, with the trick that restriction takes an equality argument which determines the context-witness index that is returned; terms are dependent natural transformations:

$$\begin{aligned}
\mathsf{Ty}\,\Gamma \quad &:\equiv (A : (I : C) \to \Gamma\, I \to \mathsf{U}) \times (-[-]_A : A\, I\,\gamma_I \to (f : C(J,I)) \to \gamma_I[f]_\Gamma = \gamma_J \to A\, J\,\gamma_J) \times \\
&\qquad ([\circ]_A : a_I[f \circ g]_A\,[\circ]_\Gamma = (a_I[f]_a\,\mathsf{refl})[g]_A\,\mathsf{refl}) \times ([\mathsf{id}]_A : a_I[f]_A\,[\mathsf{id}]_\Gamma = a_I) \\
\mathsf{Tm}\,\Gamma\,A &:\equiv (a : (\gamma_I : \Gamma\, I) \to A\, I\,\gamma_I) \times \\
&\qquad (-[-]_a : (\gamma_I : \Gamma\, I)(f : C(J,I)) \to (a\,\gamma_I)[f]_A\,\mathsf{refl} = a\,(\gamma_I[f]_\Gamma))
\end{aligned}$$

Instantiation of the family part of a type is pointwise, while restriction reuses restriction of A and naturality of γ is only needed to obtain the equality.

$$\begin{aligned}
A[\gamma]\,I\,\delta_I \qquad\qquad &:\equiv A\,I\,(\gamma\,\delta_I) \\
a_I[f]_{A[\gamma]}\,(e : \delta_I[f]_\Delta = \delta_J) &:\equiv a_I[f]_A\,((\gamma\,\delta_I)[f]_\Gamma \overset{\delta_I[f]_\gamma}{=} \gamma\,(\delta_I[f]_\Delta) \overset{e}{=} \gamma\,\delta_J)
\end{aligned}$$

The functor laws for type instantiation are strict (we write underscore for equalities because they are proof irrelevant):

$$\begin{aligned}
&A[\gamma \circ \delta]\,I\,\theta_I \equiv A\,I\,((\gamma \circ \delta)\,\theta_I) \equiv A\,(\gamma\,(\delta\,\theta_I)) \equiv A[\gamma][\delta]\,I\,\theta_I && A[\mathsf{id}]\,I\,\gamma_I \equiv A\,I\,(\mathsf{id}\,\gamma_I) \equiv A\,I\,\gamma_I \\
&a_I[f]_{A[\gamma\circ\delta]}\,\_ \equiv a_I[f]_A\,\_ \equiv a_I[f]_{A[\gamma]}\,\_ \equiv a_I[f]_{A[\gamma][\delta]}\,\_ && a_I[f]_{A[\mathsf{id}]}\,\_ \equiv a_I[f]_A\,\_
\end{aligned}$$

Term instantiation is function composition in its first components and the second component is in Prop, so functor laws for terms are also strict. Context extension is pointwise:

$$(\Gamma \triangleright A)\,I :\equiv (\gamma_I : \Gamma\,I) \times A\,I\,\gamma_I \qquad\qquad (\gamma_I, A_I)[f]_{\Gamma\triangleright A} :\equiv (\gamma_I[f]_\Gamma, a_I[f]_A\,\mathsf{refl})$$

The proof-relevant part of $-,-$, p and q is just pairing of OTT's Σ, first and second projections, hence $\triangleright\beta_1$, $\triangleright\beta_2$, $\triangleright\eta$ are all definitional. Note that the naturality components in p and q are given by refl (they are definitionally natural), and so is $\mathsf{q}[\mathsf{p}]$, $\mathsf{q}[\mathsf{p}][\mathsf{p}]$, and so on. □

Remark 12. *We relied crucially on equalities being in* Prop. *It is not clear whether a $\overline{\mathsf{CwF}}$ of presheaves can be obtained with a* U*-valued equality type (even if it has propositional UIP).*

Although we have a completely strict CwF of presheaves, we don't know how to equip it with strict Π types, or even substitution-strict Π types. We will explain why the usual definition of Π only has a weak substitution rule. But first we need the Yoneda embedding y and the Yoneda lemma's isomorphism yl:

$$\begin{array}{lll} \mathsf{y} : C \to \mathsf{Con} & \mathsf{y} : C(J,I) \to \mathsf{Sub}\,(\mathsf{y}\,J)\,(\mathsf{y}\,I) & \mathsf{yl}_\Gamma : \Gamma\, I \cong \mathsf{Sub}\,(\mathsf{y}\,I)\,\Gamma : \mathsf{yl}_\Gamma^{-1} \\ \mathsf{y}\,I :\equiv \lambda J.C(J,I) & \mathsf{y}\,f\,g :\equiv f \circ g & \mathsf{yl}_\Gamma\,\gamma_I :\equiv \lambda f.\gamma_I[f]_\Gamma \\ f[g]_{\mathsf{y}\,I} :\equiv f \circ g & & \mathsf{yl}_\Gamma^{-1}\,\gamma :\equiv \gamma\,\mathsf{id}_I \end{array}$$

We have that $\mathsf{yl}_\Gamma\,\gamma_I$ is natural both in I and in Γ, but both of these equations are weak:

$$\mathsf{yl}_\Gamma\,\gamma_I \circ \mathsf{y}\,f \equiv \lambda g.\,\gamma_I[f \circ g]_\gamma \overset{[\circ]_\Gamma}{=} \lambda g.\,\gamma_I[f]_\Gamma[g]_\Gamma \equiv \mathsf{yl}_\Gamma\,(\gamma_I[f]_\Gamma)$$
$$\gamma \circ \mathsf{yl}_\Delta\,\delta_I \equiv \lambda f.\,\gamma\,(\delta_I[f]_\Delta) \overset{\delta_I[f]_\gamma}{=} \lambda f.\,(\gamma\,\delta_I)[f]_\Gamma \equiv \mathsf{yl}_\Gamma\,(\gamma\,\delta_I) \tag{6}$$

Π types are defined in the presheaf model as follows:

$$\Pi\,A\,B\,I\,\gamma_I :\equiv \mathsf{Tm}\,(\mathsf{y}\,I \triangleright A[\mathsf{yl}_\Gamma\,\gamma_I])\,(B[(\mathsf{yl}_\Gamma\,\gamma_I)^\uparrow]) \qquad t[f]_{\Pi\,A\,B}\,e :\equiv ([\circ]_\Gamma \cdot e)_*\,(t[(\mathsf{y}\,f)^\uparrow])$$

To show Π[], we need naturality of yl, which is not definitional in general:

$$(\Pi\,A\,B)[\gamma]\,I\,\delta_I \equiv \mathsf{Tm}\,(\mathsf{y}\,I \triangleright A[\mathsf{yl}_\Gamma\,(\gamma\,\delta_I)])\,(B[(\mathsf{yl}_\Gamma\,(\gamma\,\delta_I))^\uparrow]) \overset{(6)}{=}$$
$$\mathsf{Tm}\,(\mathsf{y}\,I \triangleright A[\gamma \circ \mathsf{yl}_\Delta\,\delta_I])\,(B[(\gamma \circ \mathsf{yl}_\Delta\,\delta_I)^\uparrow]) \equiv \mathsf{Tm}\,(\mathsf{y}\,I \triangleright A[\gamma][\mathsf{yl}_\Delta\,\delta_I])\,(B[\gamma^\uparrow][(\mathsf{yl}_\Delta\,\delta_I)^\uparrow]) \equiv$$
$$(\Pi\,(A[\gamma])\,(B[\gamma^\uparrow]))\,I\,\delta_I$$

If the domain is locally representable, there is another, first-order Π type (it can only be iterated in the codomain). Unfortunately that Π type is not strict either. An $A : \mathsf{Ty}\,\Gamma$ is called locally representable if we have an operation

$$- \triangleright_A - : (I : C) \to \Gamma\,I \to C$$

together with the following isomorphism for any I, J, $\gamma_I : \Gamma\,I$ which is natural in J where we give names to the maps in both directions:

$$(\mathsf{p}_A \circ -, \mathsf{q}_A[-]) : C(J, I \triangleright_A \gamma_I) \cong (f : C(J,I)) \times A\,J\,(\gamma_I[f]_\Gamma) : -,_A -,$$

here

$$\mathsf{p}_A : C(I \triangleright_A \gamma_I, I) \qquad \text{and} \qquad \mathsf{q}_A : A\,(I \triangleright_A \gamma_I, I)\,(\gamma_I[\mathsf{p}_A]_\Gamma).$$

If A is locally representable, then so is $A[\gamma]$ by

$$I \triangleright_{A[\gamma]} \delta_I :\equiv I \triangleright_A \gamma\,\delta_I \qquad \text{with} \qquad \mathsf{p}_{A[\gamma]} :\equiv \mathsf{p}_A \qquad \text{and} \qquad \mathsf{q}_{A[\gamma]} :\equiv \mathsf{q}_A.$$

Now we can define the first-order Π type by the dependent presheaf with action on objects as

$$\Pi\,A\,B\,I\,\gamma_I :\equiv B\,(I \triangleright_A \gamma_I)\,(\gamma_I[\mathsf{p}_A]_\Gamma, \mathsf{q}_A).$$

It has the usual universal property $\mathsf{Tm}\,(\Gamma \triangleright A)\,B \cong \mathsf{Tm}\,\Gamma\,(\Pi\,A\,B)$, but its Π[] law is weak:

$$(\Pi\,A\,B)[\gamma]\,I\,\delta_I \equiv B\,(I \triangleright_A \gamma\,\delta_I)\,((\gamma\,\delta_I)[\mathsf{p}_A]_\Gamma, \mathsf{q}_A) \overset{\delta_I[\mathsf{p}_A]_\gamma}{=} B\,(I \triangleright_A \gamma\,\delta_I)\,(\gamma\,(\delta_I[\mathsf{p}_A]_\Gamma), \mathsf{q}_A) \equiv$$
$$B[\gamma^\uparrow]\,(I \triangleright_A \gamma\,\delta_I)\,(\delta_I[\mathsf{p}_A]_\Gamma, \mathsf{q}_A) \equiv B[\gamma^\uparrow]\,(I \triangleright_{A[\gamma]} \delta_I)\,(\delta_I[\mathsf{p}_{A[\gamma]}]_\Gamma, \mathsf{q}_{A[\gamma]}) \equiv \Pi\,(A[\gamma])\,(B[\gamma^\uparrow])\,I\,\delta_I$$

3.1 If we had some definitional equalities...

In this subsection, for readability and motivational reasons, we assume equality reflection (in OTT as well), that is, we won't distinguish $\equiv$ and $=$. The contents of this subsection can be defined in an intensional setting[1] with lots of effort and the result would be unreadable. We only attempt this in Sections 4–5 using a stricter notion of presheaf.

In presheaves over the syntax ($\mathsf{P} :\equiv \mathsf{PSh}(\mathsf{I})$), we have a universe defined by syntactic types and terms, in other words, a closed type and a family over it:

$$\begin{array}{llll} \mathsf{Ty} & : \mathsf{Ty}_\mathsf{P}\,\diamond & \mathsf{Tm} & : \mathsf{Ty}_\mathsf{P}\,(\diamond \triangleright \mathsf{Ty}) \\ \mathsf{Ty}\,\Gamma\,\mathsf{tt} & :\equiv \mathsf{Ty}_\mathsf{I}\,\Gamma & \mathsf{Tm}\,\Gamma\,(\mathsf{tt}, A) & :\equiv \mathsf{Tm}_\mathsf{I}\,\Gamma\,A \\ -[-]_{\mathsf{Ty}} & :\equiv -[-]_\mathsf{I} & -[-]_{\mathsf{Tm}} & :\equiv -[-]_\mathsf{I} \end{array}$$

As $\mathsf{Tm}[\epsilon, A]$ is locally representable for any A, this induces a first-order Π type as explained above. For $A : \mathsf{Tm}_\mathsf{P}\,X\,(\mathsf{Ty}[\epsilon])$ and $F : \mathsf{Ty}_\mathsf{P}\,(X \triangleright \mathsf{Tm}[\epsilon, A])$, we have $\Pi\,(\mathsf{Tm}[\epsilon, A])\,F$ defined as follows.

$$\Pi\,(\mathsf{Tm}[\epsilon, A])\,F\,\Gamma\,x_\Gamma :\equiv F\,(\Gamma \triangleright_\mathsf{I} A\,x_\Gamma)\,(x_\Gamma[\mathsf{p}_\mathsf{I}]_X, \mathsf{q}_\mathsf{I}) \qquad f_\bullet[\gamma]_{\Pi\,(\mathsf{Tm}[\epsilon,A])\,F} :\equiv f_\bullet[\gamma^\uparrow]_F$$

In the above definition we relied heavily on equality reflection to remove transports.

Its universal property $\mathsf{lam} : \mathsf{Tm}_\mathsf{P}\,(X \triangleright \mathsf{Tm}[\epsilon, A])\,F \cong \mathsf{Tm}_\mathsf{P}\,X\,(\Pi\,(\mathsf{Tm}[\epsilon, A])\,F) : \mathsf{app}$ is given by

$$\mathsf{lam}\,f\,x_\Gamma :\equiv f\,(x_\Gamma[\mathsf{p}_\mathsf{I}]_X, \mathsf{q}_\mathsf{I}) \qquad \text{and} \qquad \mathsf{app}\,t\,(x_\Gamma, a_\Gamma) :\equiv (t\,x_\Gamma)[\langle a_\Gamma\rangle_\mathsf{I}]_F.$$

Using this Π type, we can show that in P, the universe Ty–Tm is closed under Π and Bool, in exactly the same way as specified in Definition 9. However, we don't move to an internal language, we show how to state and define the components of Definition 9 externally. We have already seen that the internal $\mathsf{Ty} : \mathsf{U}$ becomes an external $\mathsf{Ty} : \mathsf{Ty}_\mathsf{P}\,\diamond$ and $\mathsf{Tm} : \mathsf{Ty} \to \mathsf{U}$ becomes $\mathsf{Tm} : \mathsf{Ty}_\mathsf{P}\,(\diamond \triangleright \mathsf{Ty})$. Externally, $\Pi : (A : \mathsf{Ty}) \to (\mathsf{Tm}\,A \to \mathsf{Ty}) \to \mathsf{Ty}$ becomes

$$\Pi : \mathsf{Tm}\,(\diamond \triangleright \mathsf{Ty} \triangleright \mathsf{Tm} \Rightarrow \mathsf{Ty}[\epsilon])\,(\mathsf{Ty}[\epsilon]).$$

The $\Rightarrow$ symbol comes from the first-order function space defined above. We define directly $\Pi\,(\mathsf{tt}, A, B) :\equiv \Pi_\mathsf{I}\,A\,B$ which makes sense because $A : \mathsf{Ty}\,\Gamma\,\mathsf{tt} \equiv \mathsf{Ty}_\mathsf{I}\,\Gamma$ and $B : (\mathsf{Tm} \Rightarrow \mathsf{Ty}[\epsilon])\,\Gamma\,(\mathsf{tt}, A) \equiv (\mathsf{Tm}[\epsilon, \mathsf{q}] \Rightarrow \mathsf{Ty}[\epsilon])\,\Gamma\,(\mathsf{tt}, A) \equiv \mathsf{Ty}[\epsilon]\,(\Gamma \triangleright \mathsf{q}\,(\mathsf{tt}, A))\,((\mathsf{tt}, A)[\mathsf{p}], \mathsf{q}) \equiv \mathsf{Ty}\,(\Gamma \triangleright A)\,\mathsf{tt} \equiv \mathsf{Ty}_\mathsf{I}\,(\Gamma \triangleright A)$.

Stating abstraction for Π externally is more involved, but its definition is easy:

$$\begin{array}{l} \mathsf{lam} : \mathsf{Tm}\,(\diamond \triangleright \mathsf{Ty} \triangleright \mathsf{Tm} \Rightarrow \mathsf{Ty}[\epsilon] \triangleright \Pi\,(\mathsf{Tm}[\mathsf{p}])\,(\mathsf{Tm}[\epsilon, \mathsf{q}[\mathsf{p}] \bullet \mathsf{q}]))\,(\mathsf{Tm}[\epsilon, \Pi][\mathsf{p}]) \\ \mathsf{lam}\,(\mathsf{tt}, A, B, t) :\equiv \mathsf{lam}\,t \end{array}$$

What we want is a $\mathsf{Tm}[\epsilon, \Pi]\,\Gamma\,(\mathsf{tt}, A, B, t) \equiv \mathsf{Tm}_\mathsf{I}\,\Gamma\,(\Pi\,A\,B)$, and we have that t has type

$$\begin{array}{llll} \Pi\,(\mathsf{Tm}[\mathsf{p}])\,(\mathsf{Tm}[\epsilon, \mathsf{q}[\mathsf{p}] \bullet \mathsf{q}]) & \equiv & \mathsf{Tm}\,(\Gamma \triangleright A)\,(\mathsf{tt}, \mathsf{q}\,(\mathsf{tt}, A[\mathsf{p}], B[\mathsf{p}])[\langle \mathsf{q}\rangle]) & \equiv \\ \Pi\,(\mathsf{Tm}[\epsilon, \mathsf{q}[\mathsf{p}]])\,(\mathsf{Tm}[\epsilon, \mathsf{q}[\mathsf{p}] \bullet \mathsf{q}]) & \equiv & \mathsf{Tm}\,(\Gamma \triangleright A)\,(\mathsf{tt}, B[\mathsf{p}][\langle \mathsf{q}\rangle]) & \equiv \\ \mathsf{Tm}[\epsilon, \mathsf{q}[\mathsf{p}] \bullet \mathsf{q}]\,(\Gamma \triangleright \mathsf{q}[\mathsf{p}]\,(\mathsf{tt}, A, B))\,((\mathsf{tt}, A, B)[\mathsf{p}], \mathsf{q}) & \equiv & \mathsf{Tm}\,(\Gamma \triangleright A)\,(\mathsf{tt}, B) & \equiv \\ \mathsf{Tm}[\epsilon, \mathsf{app}\,\mathsf{q}]\,(\Gamma \triangleright A)\,(\mathsf{tt}, A[\mathsf{p}], B[\mathsf{p}], \mathsf{q}) & \equiv & \mathsf{Tm}_\mathsf{I}\,(\Gamma \triangleright A)\,B. & \\ \mathsf{Tm}\,(\Gamma \triangleright A)\,(\mathsf{tt}, \mathsf{app}\,\mathsf{q}\,(\mathsf{tt}, A[\mathsf{p}], B[\mathsf{p}], \mathsf{q})) & \equiv & & \end{array}$$

The type of booleans in a P-universe is a $\mathsf{Bool} : \mathsf{Tm}_\mathsf{P}\,\diamond\,(\mathsf{Ty}[\epsilon])$, and is instantiated simply by $\mathsf{Bool}\,\mathsf{tt} :\equiv \mathsf{Bool}_\mathsf{I}$. We state and instantiate the rest of the internal universe analogously, see Definition 13 for a complete version of the statement.

[1]We can also obtain the intensional version of this section using Hofmann's translation [Hofmann 1995; Oury 2005; Winterhalter et al. 2019].

Now we define a new model called the P-contextualisation of the P-universe, it is analogous to Definition 10, but the substitution calculus is not replaced by by substitutions in $\mathsf{P} \equiv \mathsf{PSh}(\mathsf{I})$ (instead of by U-function space). The sorts in this model are as follows.

$$\mathsf{Con} :\equiv \mathsf{Con}_\mathsf{P} \qquad \mathsf{Sub} :\equiv \mathsf{Sub}_\mathsf{P} \qquad \mathsf{Ty}\,\Gamma :\equiv \mathsf{Tm}_\mathsf{P}\,\Gamma\,(\mathsf{Ty}[\epsilon]) \qquad \mathsf{Tm}\,\Gamma\,A :\equiv \mathsf{Tm}_\mathsf{P}\,\Gamma\,(\mathsf{Tm}[\epsilon, A])$$

Types and terms are put together from the corresponding components in the P-universe, e.g. $\Pi\,A\,B :\equiv \Pi[\epsilon, A, B]$ and $\mathsf{Bool} :\equiv \mathsf{Bool}[\epsilon]$.

Substitutions, types and terms in the P-contextualisation model are isomorphic to syntactic types and terms by Yoneda:

$$\begin{aligned}
&\mathsf{y} : \mathsf{Con}_\mathsf{I} \longrightarrow \mathsf{Con}_\mathsf{P} && \text{(the Yoneda embedding)}\\
&\mathsf{y} : \mathsf{Sub}_\mathsf{I}\,\Delta\,\Gamma \cong \mathsf{Sub}_\mathsf{P}\,(\mathsf{y}\,\Delta)\,(\mathsf{y}\,\Gamma) && \text{(consequence of Yoneda lemma: } \mathsf{y} :\equiv \mathsf{yl}_{\mathsf{y}\,\Gamma}\text{)}\\
&\mathsf{y} : \mathsf{Ty}_\mathsf{I}\,\Gamma \cong \mathsf{Tm}_\mathsf{P}\,(\mathsf{y}\,\Gamma)\,(\mathsf{Ty}[\epsilon]) && \text{(dependent version of the Yoneda lemma)}\\
&\mathsf{y} : \mathsf{Tm}_\mathsf{I}\,\Gamma\,A \cong \mathsf{Tm}_\mathsf{P}\,(\mathsf{y}\,\Gamma)\,(\mathsf{Tm}[\epsilon, \mathsf{y}\,A]) && \text{(dependent version of the Yoneda lemma)}
\end{aligned}$$

The dependent version of the Yoneda lemma states that for an $F : \mathsf{Ty}_\mathsf{P}\,X$, we have $F\,\Gamma\,x_\Gamma \cong \mathsf{Tm}_\mathsf{P}\,(\mathsf{y}\,\Gamma)\,(F[\mathsf{yl}_X\,x_\Gamma])$ where $\mathsf{yl}_X : X\,\Gamma \longrightarrow \mathsf{Sub}\,(\mathsf{y}\,\Gamma)\,X$ is the forward direction of the nondependent Yoneda lemma.

In summary, viewing the syntax I as a PSh(I)-universe, and then applying PSh(I)-contextualisation to this universe, we obtained a model, which by Yoneda is isomorphic to the syntax (not for contexts, but this can be solved easily). The substitution calculus in this model is given by presheaves, while types and terms still come from the syntax. This is the essence of the strictification construction in this paper: replacing the substitution calculus of our syntax with something that is intensionally better-behaved. In the next two sections, we will redo this construction in an intensional setting with a stricter notion of presheaves.

4 THE ABSTRACT STRICTIFICATION CONSTRUCTION

In this section, we redo the constructions of the last section in an abstract setting, for any model P. In the last section, P was fixed to be PSh(I). More precisely, we will do the following.

Def. 13. We specify what a P-universe closed under Π and Bool is.
Con. 14. The P-contextualisation of a P-universe gives a substitution-strict model.
Con. 15. We define a telescopic variant of P-contextualisation.
Def. 16. We specify Yoneda embedding from the syntax to a P-universe.
Con. 17. We show that the telescopic P-contextualisation of a P-universe with a Yoneda embedding is isomorphic to the syntax.

Instantiating Definition 13 and 16 provides our strictification construction for the syntax. In Section 3 we saw that presheaves don't suffice. In Section 5, we will see that prefascist sets do.

Definition 9 expresses what a universe closed under Π and Bool is. It is stated in OTT as metatheory. Now we will say the same but making the metatheory explicit as a $\mathsf{P} : \overline{\mathsf{CwF}}_{\overline{\Pi}}$. What does it mean that there is a universe in a model P?

Definition 13 (A universe closed under Π and Bool internal to a $\mathsf{P} : \overline{\mathsf{CwF}_{\Pi}}$). *We have the following components (we stop writing* $_{\mathsf{P}}$ *subscripts after the second line). C.f. Definition 9.*

$$
\begin{aligned}
&\mathsf{Ty} && : \mathsf{Ty}_{\mathsf{P}}\, \diamond_{\mathsf{P}} \\
&\mathsf{Tm} && : \mathsf{Ty}_{\mathsf{P}}\, (\diamond_{\mathsf{P}} \rhd_{\mathsf{P}} \mathsf{Ty}) \\
&\Pi && : \mathsf{Tm}\, (\diamond \rhd \mathsf{Ty} \rhd \mathsf{Tm} \Rightarrow \mathsf{Ty}[\epsilon])\, (\mathsf{Ty}[\epsilon]) \\
&\mathsf{lam} && : \mathsf{Tm}\, (\diamond \rhd \mathsf{Ty} \rhd \mathsf{Tm} \Rightarrow \mathsf{Ty}[\epsilon] \rhd \Pi\, (\mathsf{Tm}[\mathsf{p}])\, (\mathsf{Tm}[\epsilon, \mathsf{q}[\mathsf{p}] \bullet \mathsf{q}]))\, (\mathsf{Tm}[\epsilon, \Pi][\mathsf{p}]) \\
&\mathsf{app} && : \mathsf{Tm}\, (\diamond \rhd \mathsf{Ty} \rhd \mathsf{Tm} \Rightarrow \mathsf{Ty}[\epsilon] \rhd \mathsf{Tm}[\epsilon, \Pi])\, (\Pi\, (\mathsf{Tm}[\mathsf{p}^2])\, (\mathsf{Tm}[\epsilon, \mathsf{q}[\mathsf{p}^2] \bullet \mathsf{q}])) \\
&\Pi\beta && : \mathsf{app}[\epsilon, A, B, \mathsf{lam}[\epsilon, A, B, t]] = t \\
&\Pi\eta && : \mathsf{lam}[\epsilon, A, B, \mathsf{app}[\epsilon, A, B, t]] = t \\
&\mathsf{Bool} && : \mathsf{Tm}\, \diamond\, (\mathsf{Ty}[\epsilon]) \\
&\mathsf{true} && : \mathsf{Tm}\, \diamond\, (\mathsf{Tm}[\epsilon, \mathsf{Bool}]) \\
&\mathsf{false} && : \mathsf{Tm}\, \diamond\, (\mathsf{Tm}[\epsilon, \mathsf{Bool}]) \\
&\mathsf{ite}^{\mathsf{T}} && : \mathsf{Tm}\, (\diamond \rhd \mathsf{Ty} \rhd \mathsf{Ty}[\epsilon] \rhd \mathsf{Tm}[\epsilon, \mathsf{Bool}[\epsilon]])\, (\mathsf{Ty}[\epsilon]) \\
&\mathsf{ite}^{\mathsf{T}}\beta_1 && : \mathsf{ite}^{\mathsf{T}}[\epsilon, A, B, \mathsf{true}[\epsilon]] = A \\
&\mathsf{ite}^{\mathsf{T}}\beta_2 && : \mathsf{ite}^{\mathsf{T}}[\epsilon, A, B, \mathsf{false}[\epsilon]] = B \\
&\mathsf{ite}^{\mathsf{t}} && : \mathsf{Tm}\, (\diamond \rhd \mathsf{Tm}[\epsilon, \mathsf{Bool}] \Rightarrow \mathsf{Ty}[\epsilon] \rhd \mathsf{Tm}[\epsilon, \mathsf{q} \bullet \mathsf{true}[\epsilon]] \rhd \mathsf{Tm}[\epsilon, \mathsf{q}[\mathsf{p}] \bullet \mathsf{false}[\epsilon]] \rhd \mathsf{Tm}[\epsilon, \mathsf{Bool}[\epsilon]]) \\
& && \quad (\mathsf{Tm}[\epsilon, \mathsf{q}[\mathsf{p}^3] \bullet \mathsf{q}]) \\
&\mathsf{ite}^{\mathsf{t}}\beta_1 && : \mathsf{ite}^{\mathsf{t}}[\epsilon, P, u, v, \mathsf{true}[\epsilon]] = u \\
&\mathsf{ite}^{\mathsf{t}}\beta_2 && : \mathsf{ite}^{\mathsf{t}}[\epsilon, P, u, v, \mathsf{false}[\epsilon]] = v
\end{aligned}
$$

Problem 14 (P-contextualisation). *Given a universe* Ty–Tm *internal to a* $\mathsf{P} : \overline{\mathsf{CwF}_{\Pi}}$ *as in Definition 13, its* P*-contextualisation is a* $\overline{\mathsf{CwF}}_{\Pi,\mathsf{Bool}}$*. C.f. Problem 10.*

Construction. The substitution calculus for our new model directly reuses that of P, while types and terms come from the P-universe we started with. We stop writing $_{\mathsf{P}}$ subscripts after the second line.

$$
\begin{array}{llllll}
\mathsf{Con} & :\equiv \mathsf{Con}_{\mathsf{P}} & A[\gamma] & :\equiv A[\gamma] & \Pi\, A\, B & :\equiv \Pi[\epsilon, A, \mathsf{lam}\, B] \\
\mathsf{Sub} & :\equiv \mathsf{Sub}_{\mathsf{P}} & a[\gamma] & :\equiv a[\gamma] & \mathsf{lam}\, \{A\}\, \{B\}\, t & :\equiv \mathsf{lam}[\epsilon, A, \mathsf{lam}\, B, \mathsf{lam}\, t] \\
\mathsf{Ty}\, \Gamma & :\equiv \mathsf{Tm}\, \Gamma\, (\mathsf{Ty}[\epsilon]) & \Gamma \rhd A & :\equiv \Gamma \rhd \mathsf{Tm}[\epsilon, A] & \mathsf{app}\, \{A\}\, \{B\}\, t & :\equiv \mathsf{app}\, (\mathsf{app}[\epsilon, A, \mathsf{lam}\, B, t]) \\
\mathsf{Tm}\, \Gamma\, A & :\equiv \mathsf{Tm}\, \Gamma\, (\mathsf{Tm}[\epsilon, A]) & \mathsf{p} & :\equiv \mathsf{p} & \mathsf{Bool} & :\equiv \mathsf{Bool}[\epsilon] \\
\circ & :\equiv \circ & \mathsf{q} & :\equiv \mathsf{q} & \mathsf{true} & :\equiv \mathsf{true}[\epsilon] \\
\mathsf{id} & :\equiv \mathsf{id} & (\gamma, a) & :\equiv (\gamma\, , a) & \mathsf{false} & :\equiv \mathsf{false}[\epsilon] \\
\diamond & :\equiv \diamond & & & \mathsf{ite}^{\mathsf{T}}\, A\, A'\, b & :\equiv \mathsf{ite}^{\mathsf{T}}[\epsilon, A, B, b] \\
\epsilon & :\equiv \epsilon & & & \mathsf{ite}^{\mathsf{t}}\, P\, p\, p'\, b & :\equiv \mathsf{ite}^{\mathsf{t}}[\epsilon, \mathsf{lam}\, P, p, p', b]
\end{array}
$$

All the CwF-equations and substitution laws are strict, while $\Pi\beta$, $\Pi\eta$, $\mathsf{ite}^{\mathsf{T}}\beta_{1,2}$, $\mathsf{ite}^{\mathsf{t}}\beta_{1,2}$ are weak. Here are some examples:

$$
\begin{aligned}
[\circ] &: A[\gamma \circ \delta] \equiv A[\gamma \circ_{\mathsf{P}} \delta]_{\mathsf{P}} \overset{[\circ]_{\mathsf{P}}}{\equiv} A[\gamma]_{\mathsf{P}}[\delta]_{\mathsf{P}} \equiv A[\gamma][\delta] \\
\Pi[] &: (\Pi\, A\, B)[\gamma] \equiv \Pi[\epsilon, A, \mathsf{lam}\, B][\gamma] \overset{,\circ_{\mathsf{P}}}{\equiv} \Pi[\epsilon \circ \gamma, A[\gamma], (\mathsf{lam}\, B)[\gamma]] \overset{\diamond\eta_{\mathsf{P}}}{\equiv}
\end{aligned}
$$

$$\Pi[\epsilon, A[\gamma], (\mathsf{lam}\,B)[\gamma]] \overset{\mathsf{lam}[]_{\mathsf{P}}}{\equiv} \Pi[\epsilon, A[\gamma], \mathsf{lam}\,(B[\gamma])] \equiv \Pi\,(A[\gamma])\,(B[\gamma^{\uparrow}])$$

$$\Pi\beta \quad : \mathsf{app}\,(\mathsf{lam}\,t) \equiv \mathsf{app}\,(\mathsf{app}[\epsilon, A, \mathsf{lam}\,B, \mathsf{lam}[\epsilon, A, \mathsf{lam}\,B, \mathsf{lam}\,t]]) \overset{\Pi\beta}{=} \mathsf{app}\,(\mathsf{lam}\,t) \overset{\Pi\beta_{\mathsf{P}}}{\equiv} t$$

$$\Pi\eta \quad : \mathsf{lam}\,(\mathsf{app}\,t) \equiv \mathsf{lam}[\epsilon, A, \mathsf{lam}\,B, \mathsf{lam}\,(\mathsf{app}\,(\mathsf{app}[\epsilon, A, \mathsf{lam}\,B, t]))] \overset{\Pi\eta_{\mathsf{P}}}{\equiv}$$
$$\mathsf{lam}[\epsilon, A, \mathsf{lam}\,B, \mathsf{app}[\epsilon, A, \mathsf{lam}\,B, t]] \overset{\Pi\eta}{=} t$$

$$\mathsf{ite}^{\mathsf{t}}[] : (\mathsf{ite}^{\mathsf{t}}\,P\,p\,p'\,t)[\gamma] \equiv \mathsf{ite}^{\mathsf{t}}[\epsilon, \mathsf{lam}\,P, p, p', b][\gamma] \overset{[\circ]_{\mathsf{P}}}{\equiv}$$
$$\mathsf{ite}^{\mathsf{t}}[\epsilon \circ \gamma, \mathsf{lam}\,P[\gamma], p[\gamma], p'[\gamma], b[\gamma]] \overset{\diamond\eta_{\mathsf{P}}, \mathsf{lam}[]_{\mathsf{P}}}{\equiv} \mathsf{ite}^{\mathsf{t}}[\epsilon, \mathsf{lam}\,(P[\gamma^{\uparrow}]), p[\gamma], p'[\gamma], b[\gamma]] \equiv$$
$$\mathsf{ite}^{\mathsf{t}}\,(P[\gamma^{\uparrow}])\,(p[\gamma])\,(p'[\gamma])\,(t[\gamma]) \qquad \square$$

We define a variant of the above construction where contexts are inductively defined from the empty context and context extension.

Problem 15 (Telescopic P-contextualisation). *Given a universe* $\mathsf{Ty}\text{–}\mathsf{Tm}$ *internal to a* $\mathsf{P} : \overline{\mathsf{CwF}_{\Pi}}$ *as in Definition 13, its telescopic* P*-contextualisation is a* $\overline{\mathsf{CwF}_{\Pi,\mathsf{Bool}}}$.

Construction. Contexts are now telescopes of types. Telescopes are defined inductive-recursively by $\mathsf{Tel} : \mathsf{U}$ and $\ulcorner - \urcorner : \mathsf{Tel} \to \mathsf{Con}_{\mathsf{P}}$ with the following constructors:

$$\begin{array}{llll} \diamond_{\mathsf{Tel}} & : \mathsf{Tel} & \ulcorner \diamond \urcorner & :\equiv \diamond_{\mathsf{P}} \\ - \triangleright_{\mathsf{Tel}} - & : (\Gamma : \mathsf{Tel}) \to \mathsf{Tm}\,\ulcorner\Gamma\urcorner\,(\mathsf{Ty}[\epsilon]) \to \mathsf{Tel} & \ulcorner \Gamma \triangleright_{\mathsf{Tel}} A \urcorner & :\equiv \ulcorner\Gamma\urcorner \triangleright_{\mathsf{P}} \mathsf{Tm}[\epsilon, A] \end{array}$$

In the telescopic P-contextualisation model, we define empty context and context extension by $\diamond_{\mathsf{Tel}}$ and $\triangleright_{\mathsf{Tel}}$, and we adjust the rest of the model by adding the $\ulcorner - \urcorner$ operations when referring to contexts. The sorts are thus the following:

$$\mathsf{Con} :\equiv \mathsf{Tel},\ \ \mathsf{Sub}\,\Delta\,\Gamma :\equiv \mathsf{Sub}_{\mathsf{P}}\,\ulcorner\Delta\urcorner\,\ulcorner\Gamma\urcorner,\ \ \mathsf{Ty}\,\Gamma :\equiv \mathsf{Tm}_{\mathsf{P}}\,\ulcorner\Gamma\urcorner\,(\mathsf{Ty}[\epsilon]),\ \ \mathsf{Tm}\,\Gamma\,A :\equiv \mathsf{Tm}_{\mathsf{P}}\,\ulcorner\Gamma\urcorner\,(\mathsf{Tm}[\epsilon, A]).$$

Telescopic P-contextualisation is the same as taking the contextual core of P-contextualisation. □

What is the connection between a CwF C and a P-universe where $\mathsf{P} :\equiv \mathsf{PSh}(C)$? The P-universe is how C is visible in the presheaf model. In other words, the P-universe is the structure with which the presheaf model is equipped when the base category is a CwF. But a $\mathsf{PSh}(C)$-universe contains less information than a CwF: it does not say that that C has a terminal object or that C supports context extension. We make the connection between a model C and a P-universe formal by the following definition. For convenience, we fix the model to be the syntax I.

Definition 16 (Yoneda embedding from I to a P-universe). *The connection between the syntax* $\mathsf{I} : \mathsf{CwF}_{\Pi,\mathsf{Bool}}$ *and its presentation as a* P*-universe is given by the following components. This is also called a weak contextual isomorphism (contextual pseudo-isomorphism) from* I *to the contextualisation of* P*. We overload* y *for* Con, Sub, Ty *and* Tm.

$$\begin{array}{ll} \mathsf{y} \quad : \mathsf{Con}_{\mathsf{I}} \to \mathsf{Con}_{\mathsf{P}} & \mathsf{y}\,(A[\gamma]) \equiv \mathsf{y}\,A[\mathsf{y}\,\gamma] \\ \mathsf{y} \quad : \mathsf{Sub}_{\mathsf{I}}\,\Delta\,\Gamma \cong \mathsf{Sub}_{\mathsf{P}}\,(\mathsf{y}\,\Delta)\,(\mathsf{y}\,\Gamma) & \mathsf{y} \quad : \mathsf{Tm}_{\mathsf{I}}\,\Gamma\,A \cong \mathsf{Tm}_{\mathsf{P}}\,(\mathsf{y}\,\Gamma)\,(\mathsf{Tm}[\epsilon, \mathsf{y}\,A]) \\ \mathsf{y}_{\diamond}^{-1} : \mathsf{Sub}_{\mathsf{P}}\,\diamond_{\mathsf{P}}\,(\mathsf{y}\,\diamond_{\mathsf{I}}) & \mathsf{y}_{\triangleright}^{-1} : \mathsf{Sub}_{\mathsf{P}}\,(\mathsf{y}\,\Gamma \triangleright_{\mathsf{P}} \mathsf{Tm}[\epsilon, \mathsf{y}\,A])\,(\mathsf{y}\,(\Gamma \triangleright_{\mathsf{I}} A)) \\ \mathsf{y}_{\diamond}^{-1} \circ \epsilon = \mathsf{id}\,\{\mathsf{y}\,\diamond\} & \mathsf{y}_{\triangleright}^{-1} \circ (\mathsf{y}\,\mathsf{p}_{\mathsf{I}}\,,_{\mathsf{P}}\,\mathsf{y}\,\mathsf{q}_{\mathsf{I}}) = \mathsf{id} \\ \mathsf{y} \quad : \mathsf{Ty}_{\mathsf{I}}\,\Gamma \cong \mathsf{Tm}_{\mathsf{P}}\,(\mathsf{y}\,\Gamma)\,(\mathsf{Ty}[\epsilon]) & (\mathsf{y}\,\mathsf{p}_{\mathsf{I}}\,,_{\mathsf{P}}\,\mathsf{y}\,\mathsf{q}_{\mathsf{I}}) \circ \mathsf{y}_{\triangleright}^{-1} = \mathsf{id} \end{array}$$

We say that $\epsilon_{\mathsf{P}} : \mathsf{Sub}_{\mathsf{P}}\,(\mathsf{y}\,\diamond_{\mathsf{I}})\,\diamond_{\mathsf{P}}$ *is an isomorphism, and similarly* $(\mathsf{y}\,\mathsf{p}_{\mathsf{I}}\,,_{\mathsf{P}}\,\mathsf{y}\,\mathsf{q}_{\mathsf{I}}) : \mathsf{Sub}_{\mathsf{P}}\,(\mathsf{y}\,(\Gamma \triangleright_{\mathsf{I}} A))\,(\mathsf{y}\,\Gamma \triangleright_{\mathsf{P}} \mathsf{Tm}[\epsilon, \mathsf{y}\,A])$ *is an isomorphism. Equation* $\mathsf{y}\,(A[\gamma]) \equiv \mathsf{y}\,A[\mathsf{y}\,\gamma]$ *is needed to typecheck* $(\mathsf{y}\,\mathsf{p}, \mathsf{y}\,\mathsf{q})$.

Problem 17. *From a Yoneda embedding from* I *to a* P*-universe, we provide an isomorphism between* I *and the telescopic* P*-contextualisation of the universe.*

Construction. Yoneda actually shows that the sorts Sub, Ty and Tm are already isomorphic. We just show that Con_I is isomorphic to Tel, and transport the three Yoneda isomorphisms along this. We will define $f : \mathsf{Con}_\mathsf{I} \cong \mathsf{Tel}$ in multiple steps. First we define its forward map f mutually with an isomorphism e in the category Con_P–Sub_P by induction on syntactic contexts.

$$\begin{aligned}
&f && : \mathsf{Con}_\mathsf{I} \to \mathsf{Tel} \\
&e && : (\Gamma : \mathsf{Con}_\mathsf{I}) \to \ulcorner f\,\Gamma \urcorner \cong \mathsf{y}\,\Gamma \\
&f\,\diamond_\mathsf{I} && :\equiv \diamond_\mathsf{Tel} \\
&f\,(\Gamma \triangleright_\mathsf{I} A) && :\equiv f\,\Gamma \triangleright_\mathsf{Tel} \mathsf{y}\,A[e\,\Gamma]_\mathsf{P} \\
&e\,\diamond_\mathsf{I} && :\equiv \mathsf{y}_\diamond^{-1} : \diamond_\mathsf{P} \cong \mathsf{y}\,\diamond_\mathsf{I}
\end{aligned}$$

$$\begin{aligned}
e\,(\Gamma \triangleright_\mathsf{I} A) : \ulcorner f\,\Gamma \urcorner \triangleright_\mathsf{P} \mathsf{Tm}[\epsilon, \mathsf{y}\,A[e\,\Gamma]] &\cong (e\,\Gamma) \\
\mathsf{y}\,\Gamma \triangleright_\mathsf{P} \mathsf{Tm}[\epsilon, \mathsf{y}\,A[e\,\Gamma]][e\,\Gamma^{-1}] &= \\
\mathsf{y}\,\Gamma \triangleright_\mathsf{P} \mathsf{Tm}[\epsilon, \mathsf{y}\,A] &\cong (\mathsf{y}_\triangleright^{-1}) \\
\mathsf{y}\,(\Gamma \triangleright_\mathsf{I} A) &
\end{aligned}$$

Then we define the opposite direction f^{-1} and one of the round trips by mutual induction on telescopes. Finally we prove the other round trip by induction on contexts.

$$\begin{aligned}
&f^{-1} : \mathsf{Tel} \to \mathsf{Con}_\mathsf{I} && f^{-1}\,\diamond_\mathsf{Tel} :\equiv \diamond_\mathsf{I} \\
&f\eta\ : (\Gamma' : \mathsf{Tel}) \to f\,(f^{-1}\,\Gamma') = \Gamma' && f^{-1}\,(\Gamma' \triangleright_\mathsf{Tel} A) :\equiv f^{-1}\,\Gamma' \triangleright_\mathsf{I} \mathsf{y}^{-1}\,(A[(f\eta\,\Gamma')_*\,(e\,(f^{-1}\,\Gamma')^{-1})]_\mathsf{P})
\end{aligned}$$

Finally we prove the other round trip $f^{-1}\,(f\,\Gamma) = \Gamma$ by induction on Γ. Thus we obtain an isomorphism between the syntax I and the telescopic P-contextualisation of the universe Ty–Tm:

$$\begin{aligned}
&f : \mathsf{Con}_\mathsf{I} \cong \mathsf{Tel} \qquad && \mathsf{Sub}_\mathsf{I}\,\Delta\,\Gamma \overset{\mathsf{y}}{\cong} \mathsf{Sub}_\mathsf{P}\,(\mathsf{y}\,\Delta)\,(\mathsf{y}\,\Gamma) \overset{e\,\Delta, e\,\Gamma}{\cong} \mathsf{Sub}_\mathsf{P}\,\ulcorner f\,\Delta \urcorner\,\ulcorner f\,\Gamma \urcorner \\
& && \mathsf{Ty}_\mathsf{I}\,\Gamma \overset{\mathsf{y}}{\cong} \mathsf{Tm}_\mathsf{P}\,(\mathsf{y}\,\Gamma)\,(\mathsf{Ty}[\epsilon]) \overset{e\,\Gamma}{\cong} \mathsf{Tm}_\mathsf{P}\,\ulcorner f\,\Gamma \urcorner\,(\mathsf{Ty}[\epsilon]) \\
& && \mathsf{Tm}_\mathsf{I}\,\Gamma\,A \overset{\mathsf{y}}{\cong} \mathsf{Tm}_\mathsf{P}\,(\mathsf{y}\,\Gamma)\,(\mathsf{Tm}[\epsilon, \mathsf{y}\,A]) \overset{e\,\Gamma}{\cong} \mathsf{Tm}_\mathsf{P}\,\ulcorner f\,\Gamma \urcorner\,(\mathsf{Tm}[\epsilon, \mathsf{y}\,A[e\,\Gamma]]) \qquad \square
\end{aligned}$$

Now we just need to instantiate P and y to obtain a $\overline{\mathsf{CwF}}_{\Pi,\mathsf{Bool}}$ model which is isomorphic to the syntax I. As we have seen in Section 3, $\mathsf{P} :\equiv \mathsf{PSh}(\mathsf{I})$ is not good enough, because it is not a $\overline{\mathsf{CwF}}_{\Pi}$.

5 SECOND ATTEMPT: STRICTER PRESHEAVES

As explained in Section 3, the CwF of presheaves does not have strict dependent products, and for this reason, we cannot use it to instantiate the strictification construction described in Section 4. The source of this issue is that presheaves, when defined in intensional type theory, only satisfy the functoriality and the naturality equations up to a propositional equality; however, the definition of dependent products makes essential use of these equations. Fortunately for us, this problem has already been solved by Pédrot [2020] with his introduction of *prefascist sets*, an alternative presentation of presheaves which is better suited for intensional type theory. In this section, we provide a brief account Pédrot's construction, and we replay our strictification construction using prefascist sets instead of presheaves. The result is a fully strict $\overline{\mathsf{CwF}}_{\Pi}$, where all the equations that govern the substitution calculus are definitional – except for $\triangleright\eta$.

5.1 The category of prefascist sets

Assume that we have a strict category C, consisting of the following fields:

$$\begin{aligned}
&\mathsf{Obj}\ \ : \mathsf{U} && -\circ- : \mathsf{Hom}\,y\,x \to \mathsf{Hom}\,z\,y \to \mathsf{Hom}\,z\,x && \mathsf{id}\ \ : \mathsf{Hom}\,x\,x \\
&\mathsf{Hom} : (x\,y : \mathsf{Obj}) \to \mathsf{U} && \mathsf{ass}\ \ : f \circ (g \circ h) \equiv (f \circ g) \circ h && \mathsf{idl} : \mathsf{id} \circ f \equiv f \\
& && && \mathsf{idr} : f \circ \mathsf{id} \equiv f
\end{aligned}$$

Now, assume that we want to construct the category of U-valued presheaves over C, and furthermore, that we want the functoriality equations of presheaves and the naturality equations of morphisms to be definitional. For this purpose, we can use the standard definition of presheaves with the strict equality $\equiv$ to obtain a category that we denote by $\mathsf{Psh}(C)$, but this is an *external* category: its objects and its morphisms live in the outer layer of our two-level metatheory. The fundamental insight of prefascist sets is to analyse the external category $\mathsf{Psh}(C)$ through an adjunction with the internal category of families of types indexed over Obj.

Definition 18 (Indexed families). *The objects of the category* $\mathsf{U}^{\mathsf{Obj}}$ *are indexed type families* $\mathsf{Obj} \to \mathsf{U}$, *morphisms between two families P and Q are indexed functions* $(x : \mathsf{Obj}) \to P\ x \to Q\ x$. *The identity and composition are defined pointwise, and they are strictly associative and unital.*

The category $\mathsf{U}^{\mathsf{Obj}}$ fits into an adjunction diagram with the category of strict presheaves $\mathsf{Psh}(C)$.

$$\begin{tikzcd}
\mathsf{Psh}(C) \arrow[r, bend left, "\mathcal{F}"] \arrow[r, phantom, "\perp"] & \mathsf{U}^{\mathsf{Obj}} \arrow[l, bend left, "\mathcal{U}"]
\end{tikzcd}$$

The left adjoint functor $\mathcal{F}$ transforms a presheaf into an indexed family by forgetting about the action of morphisms, and the right adjoint functor $\mathcal{U}$ transforms an indexed family into a presheaf by freely adding all possible morphism actions as follows:

$$\mathcal{U}(P)\ x :\equiv \{y : \mathsf{Obj}\}(f : \mathsf{Hom}\ y\ x) \to P\ y$$

where the action of morphisms on $\mathcal{U}(P)$ is given by precomposition:

$$\begin{aligned}
&-[-] : (t : \mathcal{U}(P)\ x)(f : \mathsf{Hom}\ y\ x) \to \mathcal{U}(P)\ y \\
&t[f] :\equiv \lambda\,(g : \mathsf{Hom}\ z\ y)\,.\,t\ (f \circ g)
\end{aligned}$$

and the functoriality laws for $\mathcal{U}(P)$ are a consequence of the associativity and unitality of morphism composition. The reader may easily verify that $\mathcal{F}$ and $\mathcal{U}$ are indeed adjoint functors. It follows that the composite functor $\mathcal{F} \circ \mathcal{U}$ is a comonad on the category of indexed families, which we denote by $\square : \mathsf{U}^{\mathsf{Obj}} \to \mathsf{U}^{\mathsf{Obj}}$.

Lemma 19. *The $\square$ comonad encodes the equations of presheaves, in the sense that a presheaf is exactly the same thing as a $\square$-coalgebra – that is, a family of sets P (the elements of the presheaf) equipped with a morphism $P \to \square P$ (the action of morphisms on the elements) which is compatible with the counit and comultiplication of $\square$ (the functoriality equations).*

Now, we say that a presheaf is *cofree* if it is of the form $\mathcal{U}(P)$ for some indexed family P. The class of cofree presheaves is particularly interesting for two reasons. Firstly, the action of morphisms on a cofree presheaf is given by composition in the category C, and since this category is definitionally associative and unital, then it follows that the functoriality equations for cofree presheaves are definitional too. Secondly, a natural transformation between two cofree presheaves $\mathcal{U}(P)$ and $\mathcal{U}(Q)$ is the same as a morphism of families $\square P \to Q$, by the adjunction between $\mathcal{F}$ and $\mathcal{U}$. More explicitly, given any morphism of families $\theta : \square P \to Q$, we obtain a natural transformation θ' between $\mathcal{U}(P)$ and $\mathcal{U}(Q)$ as follows:

$$\begin{aligned}
&\theta'_x : \mathcal{U}(P)\ x \to \mathcal{U}(Q)\ x \\
&\theta'_x(t) :\equiv \lambda\,(f : \mathsf{Hom}\ y\ x)\,.\,\theta\ y\ (t[f])
\end{aligned}$$

The naturality equation unfolds to $(f : \mathsf{Sub}\ y\ x)(g : \mathsf{Sub}\ z\ y) \to \theta\ z\ (t[f][g]) \equiv \theta\ z\ (t[f \circ g])$, which holds definitionally because $\mathcal{U}(P)$ is a strict presheaf. In summary, the category of cofree presheaves can be defined as an internal category without needing any propositional equation, and as a result, all of the presheaf equations hold definitionally. As a consequence, cofree presheaves provide a strict model of simple type theory. This is the standard interpretation of simple types in the coKleisli category of the comonad $\Box$.

This is all and well, but we want a strict presentation of *presheaves*, not of a small subcategory of presheaves – cofree presheaves are very restricted, and they do not form a model of Martin-Löf type theory. To arrive at the construction of prefascist sets, we need a final insight, which is the fact that every presheaf can be presented as a subcoalgebra of a cofree presheaf. Indeed, given any presheaf P, the unit of the adjunction $\eta : P \to \mathcal{U}(\mathcal{F}(P))$ is injective, and therefore it exhibits P as a subcoalgebra of the cofree presheaf $\mathcal{U}(\mathcal{F}(P))$. This observation leads us to define a *prefascist set* as a subcoalgebra of a cofree presheaf, *i.e.*, a pair of an indexed family $\Gamma_0 : \mathsf{U}^{\mathsf{Obj}}$ and a predicate $\Gamma_1 : \Box\Gamma_0 \to \Omega$ (where Ω is the constant family $\lambda\, x\, .\, \mathsf{Prop}$):

$$\mathsf{Con} : \mathsf{U} :\equiv (\Gamma_0 : (x : \mathsf{Obj}) \to \mathsf{U}) \times (\Gamma_1 : (x : \mathsf{Obj}) \to (\forall\{y\}\ (f : \mathsf{Hom}\ y\ x) \to \Gamma_0\ y) \to \mathsf{Prop})$$

Note that we use Con for the type of prefascist sets, as we will shortly show that prefascist sets form a CwF. We also define the elements of a prefascist set, which are the inhabitants of $\Box\Gamma_0$ which satisfy Γ_1, and the action of morphisms of C on these elements, which is given by composition.

$$\begin{aligned}
&\mathsf{El}\ (\Gamma : \mathsf{Con})\ (x : \mathsf{Obj}) : \mathsf{U} :\equiv \\
&\quad (\gamma_0 : \forall\{y\}(f : \mathsf{Hom}\ y\ x) \to \Gamma_0\ y) \\
&\quad \times (\gamma_1 : \forall\{y\}(f : \mathsf{Hom}\ y\ x) \to \Gamma_1\ y\ (\lambda\, g\, .\, t_0\ (f \circ g)))
\end{aligned}
\qquad
\begin{aligned}
&-[-] : (\gamma : \mathsf{El}\ \Gamma\ x)\ (f : \mathsf{Hom}\ y\ x) : \mathsf{El}\ \Gamma\ y \\
&(\gamma[f])_0 :\equiv \lambda\, g\, .\, \gamma_0\ (f \circ g) \\
&(\gamma[f])_1 :\equiv \lambda\, g\, .\, \gamma_1\ (f \circ g)
\end{aligned}$$

Since the composition operation is definitionally associative and unital, the action of morphisms on the elements of a prefascist set satisfies the functoriality laws up to definitional equality. We can then define natural transformations between prefascist sets, which are coKleisli maps that preserve the predicate, along with their action on elements.

$$\begin{aligned}
&\mathsf{Sub}\ (\Gamma\ \Delta : \mathsf{Con}) : \mathsf{U} :\equiv \\
&\quad (\sigma_0 : \forall\{x\}(\gamma : \mathsf{El}\ \Gamma\ x) \to \Delta_0\ x) \\
&\quad \times (\sigma_1 : \forall\{x\}(\gamma : \mathsf{El}\ \Gamma\ x) \to \Delta_1\ x\ (\lambda\, f\, .\, \sigma_0(\gamma[f])))
\end{aligned}
\qquad
\begin{aligned}
&- \bullet - : \mathsf{Sub}\ \Gamma\ \Delta \to \mathsf{El}\ \Gamma\ x \to \mathsf{El}\ \Delta\ x \\
&(\sigma \bullet \gamma)_0 :\equiv \lambda\, f\, .\, \sigma_0\ (\gamma[f]) \\
&(\sigma \bullet \gamma)_1 :\equiv \lambda\, f\, .\, \sigma_1\ (\gamma[f])
\end{aligned}$$

The naturality equation $(\sigma \bullet \gamma)[f] \equiv \sigma \bullet (\gamma[f])$ is definitional. Finally, we complete our definition of the category of prefascist sets by defining the composition and the identity morphisms.

$$\begin{aligned}
&- \circ - : \mathsf{Sub}\ \Delta\ \Gamma \to \mathsf{Sub}\ \Theta\ \Delta \to \mathsf{Sub}\ \Theta\ \Gamma \\
&(\sigma \circ \tau)_0\ \gamma :\equiv \sigma_0\ (\tau \bullet \gamma) \\
&(\sigma \circ \tau)_1\ \gamma :\equiv \sigma_1\ (\tau \bullet \gamma)
\end{aligned}
\qquad
\begin{aligned}
&\mathsf{id} : \mathsf{Sub}\ \Gamma\ \Gamma \\
&\mathsf{id}_0\ \gamma :\equiv \gamma_0\ \mathsf{id} \\
&\mathsf{id}_1\ \gamma :\equiv \gamma_1\ \mathsf{id}
\end{aligned}$$

Lemma 20. *The category of prefascist sets can be defined in the inner layer of our two-level metatheory (and thus in our Agda formalisation). Furthermore, there is an equivalence of categories between the category of prefascist sets* $\mathsf{Pfs}(C)$ *and the external category of strict presheaves* $\mathsf{Psh}(C)$.

5.2 The CwF of Prefascist Sets

As we see, the category of prefascist sets on C is an alternative presentation for the category of presheaves on C for which the functoriality and naturality equations are automatically definitional. But the merits of prefascist sets do not stop there: Pédrot [2020] used them to construct a strict category with families (*i.e.*, a $\overline{\mathsf{CwF}_{\overline{\Pi}}}$), where the contexts are prefascist sets and the substitutions are prefascist morphisms, and the types and the terms are respectively dependent prefascist sets and dependent sections, defined as follows:

$$\begin{aligned}
&\mathsf{Ty}\ (\Gamma : \mathsf{Con}) : \mathsf{U} :\equiv \\
&\qquad (A_0 : \{x : \mathsf{Obj}\} \to (\gamma : \mathsf{El}\ \Gamma\ x) \to \mathsf{U}) \\
&\quad \times (A_1 : \{x : \mathsf{Obj}\} \to (\gamma : \mathsf{El}\ \Gamma\ x) \to (\forall \{y\}\ (f : \mathsf{Hom}\ y\ x) \to A_0\ (\gamma[f])) \to \mathsf{Prop})
\end{aligned}$$

$$\begin{aligned}
&\mathsf{Tm}\ (\Gamma : \mathsf{Con})\ (A : \mathsf{Ty}\ \Gamma) : \mathsf{U} :\equiv \\
&\qquad (t_0 : \{x : \mathsf{Obj}\} \to (\gamma : \mathsf{El}\ \Gamma\ x) \to A_0\ x) \\
&\quad \times (t_1 : \{x : \mathsf{Obj}\} \to (\gamma : \mathsf{El}\ \Gamma\ x) \to A_1\ x\ (\lambda f\,.\,\sigma_0(\gamma[f])))
\end{aligned}$$

We also have a notion of element of a dependent prefascist set, which is indexed over an element of the base prefascist set. We overload our notations and denote these as El as well.

$$\begin{aligned}
&\mathsf{El}\ (A : \mathsf{Ty}\ \Gamma)\ (\gamma : \mathsf{El}\ \Gamma\ x) : \mathsf{U} :\equiv \\
&\qquad (a_0 : \forall \{y\}(f : \mathsf{Hom}\ y\ x) \to A_0\ (\gamma[f])) \\
&\quad \times (a_1 : \forall \{y\}(f : \mathsf{Hom}\, y\, x) \to A_1\ (\gamma[f])\ (\lambda g\,.\,a_0\ (f \circ g)))
\end{aligned}$$

Naturally, the morphism of C act on dependent element as well, and the action is definitionally functorial. We now turn ourselves to a different kind of action, the action of prefascist morphisms on dependent prefascist sets and their sections. We denote this action by $-[-]^P$ to distinguish it from the actions of the base category morphisms.

$$\begin{aligned}
&-[-]^P : \mathsf{Ty}\ \Gamma \to \mathsf{Sub}\ \Delta\ \Gamma \to \mathsf{Ty}\ \Delta && -[-]^P : \mathsf{Tm}\ \Gamma\ A \to (\sigma : \mathsf{Sub}\ \Delta\ \Gamma) \to \mathsf{Tm}\ \Delta\ (A[\sigma]^P) \\
&(a[\sigma]^P)_0\ \gamma :\equiv A_0\ (\sigma \bullet \gamma) && (t[\sigma]^P)_0\ \gamma :\equiv t_0\ (\sigma \bullet \gamma) \\
&(A[\sigma]^P)_1\ \gamma :\equiv A_1\ (\sigma \bullet \gamma) && (t[\sigma]^P)_1\ \gamma :\equiv t_1\ (\sigma \bullet \gamma)
\end{aligned}$$

Unsurprisingly, this pullback operation commutes with composition and identity definitionally. We also need a definition for context extensions in the category of prefascist sets, which is defined as a dependent sum of prefascist sets.

$$\begin{aligned}
&- \triangleright - : (\Gamma : \mathsf{Con}) \to (A : \mathsf{Ty}\ \Gamma) \to \mathsf{Con} \\
&(\Gamma \triangleright A)_0\ x :\equiv (\gamma : \mathsf{El}\ \Gamma\ x) \times A_0\ \gamma \\
&(\Gamma \triangleright A)_1\ x\ \theta :\equiv (\gamma_e : \forall f\, g \to (\mathsf{fst}\ (\theta\ f))_0\ g = (\mathsf{fst}\ (\theta\ (f \circ g))_0\ \mathsf{id})) \\
&\qquad\qquad \times (A_1\ (\lambda f\,.\,(\mathsf{fst}\ (\theta\ f))_0\ \mathsf{id})\ (\lambda f\,.\,(A_0\ \gamma_e)\ (\mathsf{snd}\ (\theta\ f))))
\end{aligned}$$

The resulting prefascist set is equipped with two projection operators which give rise to the p and q combinators, and with a pairing operator which gives rise to the substitution extension combinator. Additionally, we can form an element of $\Gamma \triangleright A$ over x given an element $\gamma : \mathsf{El}\ \Gamma\ x$ and a dependent element $a : \mathsf{El}\ A\ \gamma$. We denote the result of this operation by (γ, a).

Lastly, we define dependent products of prefascist sets as follows:

$$\begin{aligned}
&\Pi : (A : \mathsf{Ty}\ \Gamma) \to (B : \mathsf{Ty}\ (\Gamma \triangleright A)) \to \mathsf{Ty}\ \Gamma \\
&(\Pi\ A\ B)_0\ \gamma :\equiv (a : \mathsf{El}\ A\ \gamma) \to B_0\ (\gamma, a) \\
&(\Pi\ A\ B)_1\ \gamma\ \theta :\equiv (a : \mathsf{El}\ A\ \gamma) \to B_1\ (\gamma, a)\ (\lambda f\,.\,\theta\ f\ (a[f]))
\end{aligned}$$

These dependent products are equipped with a function abstraction operator and an application operator, which are a dependent generalisation of the abstraction and application in a coKleisli category. All the laws, including β and η, hold definitionally.

5.3 The internal universe of syntactic types

Now, we want to instantiate the construction described in Section 4 in the CwF of prefascist sets. But before doing so, we need to solve one last strictification problem: the construction from Section 4 is intended for presheaves over the initial model I, which is a *weak* category, while the construction of prefascist sets assumes that the base category C is a *strict* category. To correct this mismatch, we can use the folklore trick of replacing a category by its image under the Yoneda embedding.

Lemma 21 ([*strictifyCat.agda*]). *Internally to* U, *there exists an alternative presentation of the syntax whose underlying category of contexts is a strict category. We denote this new CwF as* I_s.

construction. The contexts of I_s are the same as the contexts of I, but the substitutions are replaced by natural transformations between Yoneda-embedded contexts:

$$\begin{aligned}\mathsf{Sub}_{\mathsf{I}_s}\ x\ y :\equiv (\theta : \forall\{z\}\ (f : \mathsf{Sub}_{\mathsf{I}}\ z\ x) \to \mathsf{Sub}_{\mathsf{I}}\ z\ y)\\ \times\ ((f : \mathsf{Sub}_{\mathsf{I}}\ z\ x)\ (g : \mathsf{Sub}_{\mathsf{I}}\ z'\ z) \to \theta\ (f \circ g) = (\theta\ f) \circ g)\end{aligned}$$

The identity substitutions and the composition of substitutions in I_s are defined as pointwise identity and pointwise composition, respectively. Since these natural transformation consist of pairs of a metatheoretic function and an irrelevant proof of naturality, it follows that associativity and unitality are strict. Furthermore, the Yoneda lemma guarantees that the substitutions in I_s are isomorphic to the substitutions in I. The rest of the structure (types, terms, *etc.*) is left unchanged. □

Let us write $\mathsf{Pfs}(\mathsf{I}_s)$ for the $\overline{\mathsf{CwF}_{\Pi}}$ of prefascist sets over the category of contexts of I_s. Note that I_s is a category with families, which means that it is equipped with a function $\mathsf{Ty}_{\mathsf{I}_s}$ which assigns a set of types to every context in I_s, along with an action of substitutions on these sets of types which is compatible with composition of substitutions and identities. In other words, $\mathsf{Ty}_{\mathsf{I}_s}$ is a *weak* presheaf over the category of contexts of I_s. This weak presheaf gives rise to a prefascist set through the adjunction $\mathcal{U} \dashv \mathcal{F}$, which we denote by Ty. Similarly, the component $\mathsf{Tm}_{\mathsf{I}_s}$ gives rise to a dependent prefascist set over Ty, denoted by Tm.

$$\begin{array}{ll}\mathsf{Ty} : \mathsf{Ty}\ \diamond & \mathsf{Tm} : \mathsf{Ty}\ (\diamond \triangleright \mathsf{Ty})\\ \mathsf{Ty}_0\ \{x\}\ \gamma :\equiv \mathsf{Ty}_{\mathsf{I}_s}\ x & \mathsf{Tm}_0\ \{x\}\ \gamma :\equiv \mathsf{Tm}_{\mathsf{I}_s}\ x\ (\mathsf{snd}\ (\gamma_0\ \mathsf{id}))\\ \mathsf{Ty}_1\ \{x\}\ \gamma\ \theta :\equiv \forall f \to (\theta\ \mathsf{id})[f] = \theta\ f & \mathsf{Tm}_1\ \{x\}\ \gamma\ t :\equiv \forall f \to (t\ \mathsf{id})[f] = t\ f\end{array}$$

Lemma 22. *The pair* $(\mathsf{Ty}, \mathsf{Tm})$ *forms an internal universe closed under dependent products and booleans, as described in Definition 13.*

construction. The CwF combinators that exist in the base category I_s give rise to operators on this universe of syntactic terms. For instance, the Bool operator is defined as follows:

$$\mathsf{Bool} : \mathsf{Tm}\ \diamond\ (\mathsf{Ty}[\epsilon]) \qquad \mathsf{Bool}_0\ \{x\}\ \gamma :\equiv \mathsf{Bool}_{\mathsf{I}_s} \qquad \mathsf{Bool}_1\ \{x\}\ \gamma\ f :\equiv \mathsf{Bool}[]_{\mathsf{I}_s}$$

Remark that the weak equation $\mathsf{Bool}[]_{\mathsf{I}_s}$ does play a role in this definition of "prefascist booleans". However, it is relegated to a coherence condition, while the computational content of substitutions is handled by metatheoretical functions. The Π operator – and more generally all the other operators from Definition 13 – can be defined in a similar fashion, but the coherence conditions become a bit unwieldy when binders are involved, and we must do some reasoning on transports. This definition is where the work really happens! But it happens once and for all in a very controlled manner, instead of cluttering our entire proof with transport hell. □

Since $(\mathsf{Ty}, \mathsf{Tm})$ is an internal universe closed under dependent products and dependent booleans, we can apply telescopic P-contextualisation (Problem 15) to obtain a $\overline{\mathsf{CwF}_{\Pi,\mathsf{Bool}}}$. Now it only remains to show that the result is isomorphic to I_s, and thus that it is a presentation of the initial model. For this purpose, we need to instantiate Definition 16 with the Yoneda embedding from I_s to $\mathsf{Pfs}(\mathsf{I}_s)$. (In the following definitions, we only give the proof-relevant part component of the prefascist sets, *i.e.*, the one with subscript $_0$, given that the second component plays no computational role.)

$$\begin{aligned}
&\mathsf{y} : \mathsf{Con}_{\mathsf{I}_s} \to \mathsf{Con}\\
&(\mathsf{y}\ x)_0\ \gamma :\equiv \mathsf{Sub}_{\mathsf{I}_s}\ \gamma\ x\\
&\mathsf{y} : \mathsf{Sub}_{\mathsf{I}_s}\ x\ y \to \mathsf{Sub}\ (\mathsf{y}\ x)\ (\mathsf{y}\ y)\\
&(\mathsf{y}\ f)_0\ \gamma :\equiv f \circ (\gamma_0\ \mathsf{id})\\
&\mathsf{y} : \mathsf{Ty}_{\mathsf{I}_s}\ x \to \mathsf{Tm}\ (\mathsf{y}\ x)\ (\mathsf{Ty}[\epsilon])\\
&(\mathsf{y}\ a)_0\ \gamma :\equiv a[\gamma_0\ \mathsf{id}]\\
&\mathsf{y} : \mathsf{Tm}_{\mathsf{I}_s}\ x\ a \to \mathsf{Tm}\ (\mathsf{y}\ x)\ (\mathsf{Tm}[\epsilon, \mathsf{y}\ a])\\
&(\mathsf{y}\ t)_0\ \gamma :\equiv t[\gamma_0\ \mathsf{id}]\\
&\mathsf{y}_{\triangleright}^{-1} : \mathsf{Sub}\ (\mathsf{y}\ x \triangleright \mathsf{Tm}[\epsilon, \mathsf{y}\ a])\ (\mathsf{y}\ (x \triangleright_{\mathsf{I}_s} a))\\
&(\mathsf{y}_{\triangleright}^{-1})_0\ \gamma :\equiv (\mathsf{fst}\ (\gamma_0\ \mathsf{id}))_0\ \mathsf{id}\,,\ \mathsf{snd}\ (\gamma_0\ \mathsf{id})
\end{aligned}$$

$$\begin{aligned}
&\mathsf{y}_{\diamond}^{-1} : \mathsf{Sub}\ \diamond\ (\mathsf{y}\ \diamond_{\mathsf{I}_s})\\
&(\mathsf{y}_{\diamond}^{-1})_0\ \gamma :\equiv \epsilon_{\mathsf{I}_s}\\
&\mathsf{y}^{-1} : \mathsf{Sub}\ (\mathsf{y}\ x)\ (\mathsf{y}\ y) \to \mathsf{Sub}_{\mathsf{I}_s}\ x\ y\\
&\mathsf{y}^{-1}\ \sigma :\equiv \sigma_0\ \{x\}\ (\lambda f . f, \lambda f\, g . \mathsf{refl})\\
&\mathsf{y}^{-1} : \mathsf{Tm}\ (\mathsf{y}\ x)\ (\mathsf{Ty}[\epsilon]) \to \mathsf{Ty}_{\mathsf{I}_s}\ x\\
&\mathsf{y}^{-1}\ t :\equiv t_0\ \{x\}\ (\lambda f . f, \lambda f\, g . \mathsf{refl})\\
&\mathsf{y}^{-1} : \mathsf{Tm}\ (\mathsf{y}\ x)\ (\mathsf{Tm}[\epsilon, \mathsf{y}\ a]) \to \mathsf{Tm}_{\mathsf{I}_s}\ x\ a\\
&\mathsf{y}^{-1}\ t :\equiv t_0\ \{x\}\ (\lambda f . f, \lambda f\, g . \mathsf{refl})
\end{aligned}$$

The reader may check for themselves that these definitions satisfy all the equations that appear in Definition 16. It follows that the $\overline{\mathsf{CwF}}_{\Pi,\mathsf{Bool}}$ that we obtained by telescopic P-contextualisation (which we denote by I_{ss}) is isomorphic to I_s, and therefore, that it is isomorphic to I. In particular, we can transport the induction principle for I along this isomorphism to obtain an induction principle for I_{ss}. This substitution-strict model, along with its induction principle, is our *strictified syntax*. The isomorphism is formalised in [*strictifyIso.agda*], although the formal proof is slightly different from the argument that we presented.

6 APPLICATION: CANONICITY BY GLUING

As an application for our strictification construction, we present a short and elegant proof of canonicity for dependent type theory. This is the first time a proof by *gluing* over the initial category with families is fully formalised in intensional type theory.

THEOREM 23 (BOOLEAN CANONICITY, [*canon.agda*]). *In the initial CwF, every term of type* Bool *in the empty context is equal to either* true *or* false.

PROOF. Using the results of Section 5, we may safely assume that the initial CwF is substitution-strict. The idea of the proof is to construct a *glued model* G which equips the syntax with proof-relevant predicates: the contexts of G are pairs of a syntactic context Γ and a predicate on the *closing substitutions* from $\diamond$ to Γ; the types are pairs of a syntactic type A and a predicate over the closed terms of A; the terms are pairs of a syntactic term and a proof of the predicate associated with its type. More formally, the glued model is defined as follows:

$$\begin{aligned}
&\mathsf{Con}_G && :\equiv (\Gamma : \mathsf{Con}) \times (\mathsf{Sub}\ \diamond\ \Gamma \to \mathsf{U})\\
&\mathsf{Sub}_G\ (\Gamma, \Gamma')\ (\Delta, \Delta') && :\equiv (\sigma : \mathsf{Sub}\,\Gamma\,\Delta) \times ((\gamma : \mathsf{Sub}\ \diamond\ \Gamma) \to (\gamma' : \Gamma'\ \gamma) \to \Delta'\ (\sigma \circ \gamma))\\
&(\sigma, \sigma') \circ_G (\tau, \tau') && :\equiv (\sigma \circ \tau\ \ ,\ \ \lambda \gamma\, \gamma' . \sigma'\ (\tau \circ \gamma)\ (\tau'\ \gamma\, \gamma'))\\
&\mathsf{id}_G && :\equiv (\mathsf{id}\ \ ,\ \ \lambda \gamma\, \gamma' . \gamma')\\
&\diamond_G && :\equiv (\diamond\ \ ,\ \ \lambda \gamma . \mathsf{Unit})\\
&\epsilon_G && :\equiv (\epsilon\ \ ,\ \ \lambda \gamma\, \gamma' . \mathsf{unit})\\
&\mathsf{Ty}_G\ (\Gamma, \Gamma') && :\equiv (A : \mathsf{Ty}\ \Gamma) \times ((\gamma : \mathsf{Sub}\ \diamond\ \Gamma) \to (\gamma' : \Gamma'\ \gamma) \to (\mathsf{Tm}\ \diamond\ A[\gamma]) \to \mathsf{U})
\end{aligned}$$

$$
\begin{array}{ll}
(A, A')[(\sigma, \sigma')] & :\equiv (A[\sigma] \ , \ \lambda\, \gamma\, \gamma'\, a\, .\, A'\, (\sigma \circ \gamma)\, (\sigma'\, \gamma\, \gamma')\, a) \\
\mathsf{Tm}_G\ (\Gamma, \Gamma')\ (A, A') & :\equiv (t : \mathsf{Tm}\ \Gamma\ A) \times ((\gamma : \mathsf{Sub}\ \diamond\ \Gamma) \to (\gamma' : \Gamma'\ \gamma) \to A'\ \gamma\ \gamma'\ a[\gamma]) \\
(t, t')[(\sigma, \sigma')] & :\equiv (t[\sigma] \ , \ \lambda\, \gamma\, \gamma'\, .\, a'\, (\sigma \circ \gamma)\, (\sigma'\, \gamma\, \gamma')) \\
(\Gamma, \Gamma') \triangleright_G (A, A') & :\equiv (\Gamma \triangleright A \ , \ \lambda\, \theta\, .\, (\gamma' : \Gamma'\, (\mathsf{p} \circ \theta)) \times (A'\, (\mathsf{p} \circ \theta)\, \gamma'\, \mathsf{q}[\theta])) \\
(\sigma, \sigma')\ ,_G\ (t, t') & :\equiv ((\sigma\ ,\ t) \ , \ \lambda\, \gamma\, \gamma'.\, (\sigma\, \gamma\, \gamma'\ ,\ a'\, \gamma\, \gamma')) \\
\mathsf{p}_G & :\equiv (\mathsf{p} \ , \ \lambda\, \theta\, \theta'.\, \mathsf{fst}\, \theta') \\
\mathsf{q}_G & :\equiv (\mathsf{q} \ , \ \lambda\, \theta\, \theta'.\, \mathsf{snd}\, \theta')
\end{array}
$$

Thanks to the strictness of the syntax, all CwF equations hold definitionally for G. It is remarkable that the definition is very compact and transparent with a strict initial model, yet almost impossible to write down with a weak initial model. For reasons of space, here we only listed the CwF core, but G also supports dependent products and booleans, following the definitions in [Coquand 2019]. In particular, the gluing predicate which is associated to the booleans is $\mathsf{Bool}' :\equiv \lambda\ \gamma\ \gamma'\ b\ .\ (b = \mathsf{true}) + (b = \mathsf{false})$. Now, remark that the first projection π_1 is a morphism of CwFs from G to the initial model – in other words, G is *displayed* over the initial model. By initiality, there is a morphism init which goes in the opposite direction, and furthermore we have that $\pi_1 \circ \mathsf{init} = \mathsf{id}$. As a consequence, given any boolean b defined in the empty context, we have that $\mathsf{init}(b)$ yields a proof of $\mathsf{Bool}' \ \epsilon\ \mathsf{unit}\ b$, that is, a proof that b is canonical. □

7 CONCLUSIONS AND FURTHER WORK

In this paper, we identified and overcame the biggest issue in formalising type theory in an abstract, intrinsic way: the weak substitution calculus. We developed a method to replace the substitution calculus in the syntax with one coming from strict presheaves. This provides a new syntax where all substitution laws and equations concerning variables/weakenings are definitional. We demonstrated practicality of our method by formalising it in Agda for a type theory with Π and Bool. In our formalisation, for the first time, we proved canonicity in a gluing-style way, and this proof is as short and elegant as the pen and paper proofs. The formalisation does not rely on any special feature of the metatheory (except QIITs which are unavoidable for intrinsic quotiented syntax).

Our strictification method is generic, it works for type theories with any choice of type formers, including type theories with no canonicity or normalisation. More generally, we see no problem in applying it to any non-substructural language defined as a SOGAT.

However, as of now, our approach has its downsides: we were not able to do actual computations because Agda loops when trying to compute a nontrivial boolean via the canonicity proof. Reasons for this could be that prefascist internal term representations are very big and that the implementation of OTT via rewrite rules is too slow. We also have ergonomic problems: lots of implicit arguments have to be specified by hand in the strict syntax, and error messages become difficult to read. Maybe these can be overcome by clever library implementations, or we might need extra proof assistant support – in the latter case, our development can be seen as a theoretical underpinning of proof assistants with SOGAT support. This would mean that for any SOGAT, a first-order syntax with strict substitution calculus is available. Even if practical issues are addressed, our approach has a theoretical downside: because of the higher-order representation of the new syntax, we lose most definitional computation rules of the induction principle (they still hold propositionally). The exact computational content of our construction has to be analysed further.

In the future, we plan to implement our method in Coq's OTT extension, and investigate whether it can be replayed in a setting without Prop. Also, we would like to see how the approach scales: can it be used to formalise realistic type theories with (co)inductive types, hierarchies of universes, and so on. Can intrinsic quotiented syntax be used in an implementation of a proof assistant?

REFERENCES

Andreas Abel. 2013. “Normalization by Evaluation: Dependent Types and Impredicativity.” Habilitation thesis. Ludwig-Maximilians-Universität München. http://www2.tcs.ifi.lmu.de/~abel/habil.pdf.

Andreas Abel, Joakim Öhman, and Andrea Vezzosi. Dec. 2017. “Decidability of Conversion for Type Theory in Type Theory.” *Proc. ACM Program. Lang.*, 2, POPL, Article 23, (Dec. 2017), 29 pages. DOI: 10.1145/3158111.

Arthur Adjedj, Meven Lennon-Bertrand, Kenji Maillard, Pierre-Marie Pédrot, and Loïc Pujet. 2024. “Martin-Löf à la Coq.” In: *Proceedings of the 13th ACM SIGPLAN International Conference on Certified Programs and Proofs* (CPP 2024). Association for Computing Machinery, , London, UK, 230–245. ISBN: 9798400704888. DOI: 10.1145/3636501.3636951.

Thorsten Altenkirch, Martin Hofmann, and Thomas Streicher. 1995. “Categorical Reconstruction of a Reduction Free Normalization Proof.” In: *Category Theory and Computer Science, 6th International Conference, CTCS ’95, Cambridge, UK, August 7-11, 1995, Proceedings* (Lecture Notes in Computer Science). Ed. by David H. Pitt, David E. Rydeheard, and Peter T. Johnstone. Vol. 953. Springer, 182–199. DOI: 10.1007/3-540-60164-3_27.

Thorsten Altenkirch, Martin Hofmann, and Thomas Streicher. 1996. “Reduction-Free Normalisation for a Polymorphic System.” In: *Proceedings, 11th Annual IEEE Symposium on Logic in Computer Science, New Brunswick, New Jersey, USA, July 27-30, 1996.* IEEE Computer Society, 98–106. DOI: 10.1109/LICS.1996.561309.

Thorsten Altenkirch and Ambrus Kaposi. 2016a. “Normalisation by Evaluation for Dependent Types.” In: *1st International Conference on Formal Structures for Computation and Deduction (FSCD 2016)* (Leibniz International Proceedings in Informatics (LIPIcs)). Ed. by Delia Kesner and Brigitte Pientka. Vol. 52. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 6:1–6:16. ISBN: 978-3-95977-010-1. DOI: 10.4230/LIPIcs.FSCD.2016.6.

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

Thorsten Altenkirch, Ambrus Kaposi, and András Kovács. 2017. “Normalisation by Evaluation for a Type Theory with Large Elimination.” In: *23rd International Conference on Types for Proofs and Programs, TYPES 2017.* Ed. by Ambrus Kaposi. Eötvös Loránd University. http://types2017.elte.hu/proc.pdf#page=26.

Thorsten Altenkirch, Conor McBride, and Wouter Swierstra. 2007. “Observational Equality, Now!” In: *Proceedings of the 2007 Workshop on Programming Languages Meets Program Verification* (PLPV ’07). ACM, Freiburg, Germany, 57–68. ISBN: 978-1-59593-677-6. DOI: 10.1145/1292597.1292608.

Danil Annenkov, Paolo Capriotti, Nicolai Kraus, and Christian Sattler. 2023. “Two-level type theory and applications.” *Mathematical Structures in Computer Science*, 33, 8, 688–743. DOI: 10.1017/S0960129523000130.

Steve Awodey. 2018. “Natural models of homotopy type theory.” *Math. Struct. Comput. Sci.*, 28, 2, 241–286. DOI: 10.1017/S0960129516000268.

Rafaël Bocquet. 2021. “Strictification of Weakly Stable Type-Theoretic Structures Using Generic Contexts.” In: *27th International Conference on Types for Proofs and Programs, TYPES 2021, June 14-18, 2021, Leiden, The Netherlands (Virtual Conference)* (LIPIcs). Ed. by Henning Basold, Jesper Cockx, and Silvia Ghilezan. Vol. 239. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 3:1–3:23. DOI: 10.4230/LIPICS.TYPES.2021.3.

Rafaël Bocquet, Ambrus Kaposi, and Christian Sattler. 2023. “For the Metatheory of Type Theory, Internal Sconing Is Enough.” In: *8th International Conference on Formal Structures for Computation and Deduction, FSCD 2023, July 3-6, 2023, Rome, Italy* (LIPIcs). Ed. by Marco Gaboardi and Femke van Raamsdonk. Vol. 260. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 18:1–18:23. DOI: 10.4230/LIPICS.FSCD.2023.18.

Guillaume Brunerie and Menno de Boer. 2020. *Formalization of the initiality conjecture.* (2020). Retrieved July 31, 2021 from https://github.com/guillaumebrunerie/initiality.

Mario Carneiro. 2024. *Lean4Lean: Towards a Verified Typechecker for Lean, in Lean.* (2024). https://arxiv.org/abs/2403.14064 arXiv: 2403.14064 `[cs.PL]`.

Ed. by Claudia Casadio and Philip J. Scott. “Categories with Families: Unityped, Simply Typed, and Dependently Typed.” *Joachim Lambek: The Interplay of Mathematics, Logic, and Linguistics.* Springer International Publishing, Cham, 135–180. ISBN: 978-3-030-66545-6. DOI: 10.1007/978-3-030-66545-6_5.

James Chapman. Jan. 2009. “Type Theory Should Eat Itself.” *Electr. Notes Theor. Comput. Sci.*, 228, (Jan. 2009), 21–36. DOI: 10.1016/j.entcs.2008.12.114.

Pierre Clairambault and Peter Dybjer. 2014. “The biequivalence of locally cartesian closed categories and Martin-Löf type theories.” *Math. Struct. Comput. Sci.*, 24, 6. DOI: 10.1017/S0960129513000881.

Jesper Cockx, Nicolas Tabareau, and Théo Winterhalter. Jan. 2021. “The taming of the rew: a type theory with computational assumptions.” *Proc. ACM Program. Lang.*, 5, POPL, Article 60, (Jan. 2021), 29 pages. DOI: 10.1145/3434341.

R. L. Constable et al.. 1985. *Implementing Mathematics with the Nuprl Proof Development Environment.* Prentice-Hall. http://www.nuprl.org/book/.

Thierry Coquand. 2019. “Canonicity and normalization for dependent type theory.” *Theoretical Computer Science*, 777, 184–191. In memory of Maurice Nivat, a founding father of Theoretical Computer Science - Part I. DOI: 10.1016/j.tcs.2019.01.015.

Pierre-Louis Curien, Richard Garner, and Martin Hofmann. 2014. "Revisiting the categorical interpretation of dependent type theory." *Theor. Comput. Sci.*, 546, 99–119. DOI: 10.1016/J.TCS.2014.03.003.

Nils Anders Danielsson. 2006. "A Formalisation of a Dependently Typed Language as an Inductive-Recursive Family." In: *Types for Proofs and Programs, International Workshop, TYPES 2006, Nottingham, UK, April 18-21, 2006, Revised Selected Papers* (Lecture Notes in Computer Science). Ed. by Thorsten Altenkirch and Conor McBride. Vol. 4502. Springer, 93–109. DOI: 10.1007/978-3-540-74464-1_7.

István Donkó and Ambrus Kaposi. 2021. "Internal Strict Propositions Using Point-Free Equations." In: *27th International Conference on Types for Proofs and Programs, TYPES 2021, June 14-18, 2021, Leiden, The Netherlands (Virtual Conference)* (LIPIcs). Ed. by Henning Basold, Jesper Cockx, and Silvia Ghilezan. Vol. 239. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 6:1–6:21. DOI: 10.4230/LIPIcs.TYPES.2021.6.

Peter Dybjer. 1996. "Internal type theory." In: *Types for Proofs and Programs (TYPES 1995)* (Lecture Notes in Computer Science). Ed. by Stefano Berardi and Mario Coppo. Vol. 1158. Springer Berlin Heidelberg, Berlin, Heidelberg, 120–134. ISBN: 978-3-540-70722-6. DOI: 10.1007/3-540-61780-9_66.

Daniel Gratzer. 2022. "Normalization for Multimodal Type Theory." In: *Proceedings of the 37th Annual ACM/IEEE Symposium on Logic in Computer Science* (LICS '22) Article 2. Association for Computing Machinery, Haifa, Israel, 13 pages. ISBN: 9781450393515. DOI: 10.1145/3531130.3532398.

Robert Harper, Furio Honsell, and Gordon D. Plotkin. 1993. "A Framework for Defining Logics." *J. ACM*, 40, 1, 143–184. DOI: 10.1145/138027.138060.

Martin Hofmann. 1995. "Conservativity of Equality Reflection over Intensional Type Theory." In: *Types for Proofs and Programs, International Workshop TYPES'95, Torino, Italy, June 5-8, 1995, Selected Papers* (Lecture Notes in Computer Science). Ed. by Stefano Berardi and Mario Coppo. Vol. 1158. Springer, 153–164. DOI: 10.1007/3-540-61780-9_68.

Martin Hofmann. 1994. "On the Interpretation of Type Theory in Locally Cartesian Closed Categories." In: *Computer Science Logic, 8th International Workshop, CSL '94, Kazimierz, Poland, September 25-30, 1994, Selected Papers* (Lecture Notes in Computer Science). Ed. by Leszek Pacholski and Jerzy Tiuryn. Vol. 933. Springer, 427–441. DOI: 10.1007/BFB0022273.

Martin Hofmann. 1999. "Semantical Analysis of Higher-Order Abstract Syntax." In: *14th Annual IEEE Symposium on Logic in Computer Science, Trento, Italy, July 2-5, 1999.* IEEE Computer Society, 204–213. DOI: 10.1109/LICS.1999.782616.

Martin Hofmann. 1997. "Syntax and Semantics of Dependent Types." In: *Semantics and Logics of Computation.* Cambridge University Press, 79–130.

Bart Jacobs. 1993. "Comprehension Categories and the Semantics of Type Dependency." *Theor. Comput. Sci.*, 107, 2, 169–207. DOI: 10.1016/0304-3975(93)90169-T.

Ambrus Kaposi. 2023. "Towards quotient inductive-inductive-recursive types." In: *29th International Conference on Types for Proofs and Programs (TYPES 2023).* Ed. by Eduardo Hermo Reyes and Alicia Villanueva. Valencia. https://types2023.webs.upv.es/TYPES2023.pdf.

Ambrus Kaposi, Simon Huber, and Christian Sattler. 2019. "Gluing for Type Theory." In: *4th International Conference on Formal Structures for Computation and Deduction (FSCD 2019)* (Leibniz International Proceedings in Informatics (LIPIcs)). Ed. by Herman Geuvers. Vol. 131. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 25:1–25:19. ISBN: 978-3-95977-107-8. DOI: 10.4230/LIPIcs.FSCD.2019.25.

Ambrus Kaposi, András Kovács, and Thorsten Altenkirch. Jan. 2019. "Constructing Quotient Inductive-inductive Types." *Proc. ACM Program. Lang.*, 3, POPL, Article 2, (Jan. 2019), 2:1–2:24. DOI: 10.1145/3290315.

Ambrus Kaposi, András Kovács, and Nicolai Kraus. 2019. "Shallow Embedding of Type Theory is Morally Correct." In: *Mathematics of Program Construction.* Ed. by Graham Hutton. Springer International Publishing, Cham, 329–365. ISBN: 978-3-030-33636-3.

Ambrus Kaposi and Szumi Xie. 2024. "Second-Order Generalised Algebraic Theories: Signatures and First-Order Semantics." In: *9th International Conference on Formal Structures for Computation and Deduction, FSCD 2024, July 10-13, 2024, Tallinn, Estonia* (LIPIcs). Ed. by Jakob Rehof. Vol. 299. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 10:1–10:24. DOI: 10.4230/LIPICS.FSCD.2024.10.

Ambrus Kaposi and Zongpu Xie. 2021. "Quotient inductive-inductive types in the setoid model." In: *27th International Conference on Types for Proofs and Programs, TYPES 2021.* Ed. by Henning Basold. Universiteit Leiden. https://types21.liacs.nl/download/quotient-inductive-inductive-types-in-the-setoid-model/.

Yann Leray, Gaëtan Gilbert, Nicolas Tabareau, and Théo Winterhalter. 2024. "The Rewster: Type Preserving Rewrite Rules for the Coq Proof Assistant." In: *15th International Conference on Interactive Theorem Proving, ITP 2024, September 9-14, 2024, Tbilisi, Georgia* (LIPIcs). Ed. by Yves Bertot, Temur Kutsia, and Michael Norrish. Vol. 309. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 26:1–26:18. DOI: 10.4230/LIPICS.ITP.2024.26.

Peter LeFanu Lumsdaine and Michael A. Warren. 2015. "The Local Universes Model: An Overlooked Coherence Construction for Dependent Type Theories." *ACM Trans. Comput. Log.*, 16, 3, 23:1–23:31. DOI: 10.1145/2754931.

Leonardo de Moura, Soonho Kong, Jeremy Avigad, Floris van Doorn, and Jakob von Raumer. 2015. "The Lean Theorem Prover (System Description)." In: *Automated Deduction – CADE-25*. Ed. by Amy P. Felty and Aart Middeldorp. Springer International Publishing, Berlin, Germany, 378–388. ISBN: 978-3-319-21401-6. DOI: 10.1007/978-3-319-21401-6_26.

Nicolas Oury. 2005. "Extensionality in the Calculus of Constructions." In: *Theorem Proving in Higher Order Logics, 18th International Conference, TPHOLs 2005, Oxford, UK, August 22-25, 2005, Proceedings* (Lecture Notes in Computer Science). Ed. by Joe Hurd and Thomas F. Melham. Vol. 3603. Springer, 278–293. DOI: 10.1007/11541868_18.

Pierre-Marie Pédrot. 2020. "Russian Constructivism in a Prefascist Theory." In: *Proceedings of the 35th Annual ACM/IEEE Symposium on Logic in Computer Science* (LICS '20). Association for Computing Machinery, Saarbrücken, Germany, 782–794. ISBN: 9781450371049. DOI: 10.1145/3373718.3394740.

Loïc Pujet. 2022. "Computing with Extensionality Principles in Type Theory." PhD thesis. Nantes Université. https://github.com/loic-p/PhD-thesis/releases.

Loïc Pujet, Yann Leray, and Nicolas Tabareau. 2025. "Observational Equality meets CIC." *ACM Transactions on Programming Languages and Systems (TOPLAS)*. To appear.

Loïc Pujet and Nicolas Tabareau. Jan. 2022. "Observational equality: now for good." *Proc. ACM Program. Lang.*, 6, POPL, Article 32, (Jan. 2022), 27 pages. DOI: 10.1145/3498693.

R. A. G. Seely. 1984. "Locally cartesian closed categories and type theory." *Mathematical Proceedings of the Cambridge Philosophical Society*, 95, 1, 33–48. DOI: 10.1017/S0305004100061284.

Michael Shulman. 2015. "Univalence for inverse diagrams and homotopy canonicity." *Mathematical Structures in Computer Science*, 25, 5, 1203–1277. DOI: 10.1017/S0960129514000565.

Jonathan Sterling. Nov. 2021. "First Steps in Synthetic Tait Computability: The Objective Metatheory of Cubical Type Theory." Ph.D. Dissertation. Carnegie Mellon University, (Nov. 2021). DOI: 10.5281/zenodo.6990769. Doctoral thesis of Jonathan Sterling, Carnegie Mellon University.

Jonathan Sterling and Carlo Angiuli. 2021. "Normalization for Cubical Type Theory." In: *36th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2021, Rome, Italy, June 29 - July 2, 2021*. IEEE, 1–15. DOI: 10.1109/LICS52264.2021.9470719.

The Agda Development Team. 2023. *The* Agda *Programming Language*. (2023). http://wiki.portal.chalmers.se/agda/pmwiki.php.

The Coq Development Team. 2021. *The* Coq *Proof Assistant*. (2021). https://www.coq.inria.fr.

Andrea Vezzosi, Anders Mörtberg, and Andreas Abel. 2021. "Cubical Agda: A Dependently Typed Programming Language with Univalence and Higher Inductive Types." *Journal of Functional Programming*, 31, e8. DOI: 10.1017/S0956796821000034.

Théo Winterhalter. 2024. "Dependent Ghosts Have a Reflection for Free." *Proc. ACM Program. Lang.*, 8, ICFP, 630–658. DOI: 10.1145/3674647.

Théo Winterhalter. 2020. "Formalisation and meta-theory of type theory." Ph.D. Dissertation. Université de Nantes.

Théo Winterhalter, Matthieu Sozeau, and Nicolas Tabareau. 2019. "Eliminating reflection from type theory." In: *Proceedings of the 8th ACM SIGPLAN International Conference on Certified Programs and Proofs, CPP 2019, Cascais, Portugal, January 14-15, 2019*. Ed. by Assia Mahboubi and Magnus O. Myreen. ACM, 91–103. DOI: 10.1145/3293880.3294095.

A PROOFS WHICH DID NOT FIT IN THE PAPER

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009

$$\begin{aligned}
&P[\langle u\rangle][\gamma] &&=([\circ])\\
&P[\langle u\rangle \circ \gamma] &&\equiv\\
&P[(\mathsf{id}, [\mathsf{id}]_* \, u) \circ \gamma] &&=(,\circ)\\
&P[\mathsf{id} \circ \gamma, [\circ]_* \, (([\mathsf{id}]_* \, u)[\gamma])] &&=(1)\\
&P[\mathsf{id} \circ \gamma, [\circ]_* \, ([\mathsf{id}]_* \, (u[\gamma]))] &&=(\mathsf{idl})\\
&P[\gamma, \mathsf{idl}_* \, ([\circ]_* \, ([\mathsf{id}]_* \, (u[\gamma])))] &&=(\cdot_*)\\
&P[\gamma, ([\mathsf{id}] \cdot [\circ] \cdot \mathsf{idl})_* \, (u[\gamma])] &&\equiv\\
&P[\gamma, u[\gamma]] &&\equiv\\
&P[\gamma, ([\mathsf{id}] \cdot [\mathsf{id}])_* \, (u[\gamma])] &&=(\cdot_*)\\
&P[\gamma, [\mathsf{id}]_* \, ([\mathsf{id}]_* \, (u[\gamma]))] &&=(\triangleright\beta_2)\\
&P[\gamma, [\mathsf{id}]_* \, (([\circ] \cdot \triangleright\beta_1)_* \, (\mathsf{q}[\langle u[\gamma]\rangle]))] &&\equiv\\
&P[\gamma, [\mathsf{id}]_* \, (([\circ] \cdot [\circ] \cdot \mathsf{ass} \cdot \triangleright\beta_1 \cdot \mathsf{idr} \cdot [\mathsf{id}])_* \, (\mathsf{q}[\langle u[\gamma]\rangle]))] &&=(\cdot_*)\\
&P[\gamma, ([\circ] \cdot [\circ] \cdot \mathsf{ass} \cdot \triangleright\beta_1 \cdot \mathsf{idr} \cdot [\mathsf{id}] \cdot [\mathsf{id}])_* \, (\mathsf{q}[\langle u[\gamma]\rangle])] &&\equiv\\
&P[\gamma, ([\circ] \cdot [\circ] \cdot \mathsf{ass} \cdot \triangleright\beta_1 \cdot \mathsf{idr})_* \, (\mathsf{q}[\langle u[\gamma]\rangle])] &&=(\cdot_*)\\
&P[\gamma, \mathsf{idr}_* \, (\triangleright\beta_{1*} \, (\mathsf{ass}_* \, ([\circ]_* \, ([\circ]_* \, (\mathsf{q}[\langle u[\gamma]\rangle])))))] &&=(\mathsf{idr})\\
&P[\gamma \circ \mathsf{id}, \triangleright\beta_{1*} \, (\mathsf{ass}_* \, ([\circ]_* \, ([\circ]_* \, (\mathsf{q}[\langle u[\gamma]\rangle]))))] &&=(\triangleright\beta_1)\\
&P[\gamma \circ (\mathsf{p} \circ \langle u[\gamma]\rangle), \mathsf{ass}_* \, ([\circ]_* \, ([\circ]_* \, (\mathsf{q}[\langle u[\gamma]\rangle])))] &&=(\mathsf{ass})\\
&P[(\gamma \circ \mathsf{p}) \circ \langle u[\gamma]\rangle, [\circ]_* \, ([\circ]_* \, (\mathsf{q}[\langle u[\gamma]\rangle]))] &&=(1)\\
&P[(\gamma \circ \mathsf{p}) \circ \langle u[\gamma]\rangle, [\circ]_* \, (([\circ]_* \, \mathsf{q})[\langle u[\gamma]\rangle])] &&=(,\circ)\\
&P[(\gamma \circ \mathsf{p}, [\circ]_* \, \mathsf{q}) \circ \langle u[\gamma]\rangle] &&\equiv\\
&P[(\gamma \circ \mathsf{p}, [\circ]_* \, \mathsf{q}) \circ \langle (\mathsf{Bool}[] \cdot \mathsf{Bool}[])_* \, (u[\gamma])\rangle] &&=(\cdot_*)\\
&P[(\gamma \circ \mathsf{p}, [\circ]_* \, \mathsf{q}) \circ \langle \mathsf{Bool}[]_* \, (\mathsf{Bool}[]_* \, (u[\gamma]))\rangle] &&=(3)\\
&P[(\gamma \circ \mathsf{p}, [\circ]_* \, \mathsf{q}) \circ \mathsf{Bool}[]_* \, \langle \mathsf{Bool}[]_* \, (u[\gamma])\rangle] &&=(2)\\
&P[(\mathsf{Bool}[]_* \, (\gamma \circ \mathsf{p}, [\circ]_* \, \mathsf{q})) \circ \langle \mathsf{Bool}[]_* \, (u[\gamma])\rangle] &&\equiv\\
&P[(\mathsf{Bool}[]_* \, (\gamma^{\uparrow})) \circ \langle \mathsf{Bool}[]_* \, (u[\gamma])\rangle] &&=([\circ])\\
&P[\mathsf{Bool}[]_* \, (\gamma^{\uparrow})][\langle \mathsf{Bool}[]_* \, (u[\gamma])\rangle] &&
\end{aligned}$$

Fig. 1. Proof of $\alpha\,(u : \mathsf{Tm}\,\Gamma\,\mathsf{Bool}) : P[\langle u\rangle][\gamma] = P[\mathsf{Bool}[]_* \, (\gamma^{\uparrow})][\langle \mathsf{Bool}[]_* \, (u[\gamma])\rangle]$. This α used in Definition 4, in the rule $\mathsf{ite}^{\mathsf{t}}[]$.