Published: 08 January 2026
Accepted: 06 November 2025
Received: 10 July 2025

RESEARCH-ARTICLE

From Semantics to Syntax: A Type Theory for Comprehension Categories

NIYOUSHA NAJMAEI, Polytechnique School, Palaiseau, Ile-de-France, France

NIELS VAN DER WEIDE, Radboud University, Nijmegen, Gelderland, Netherlands

BENEDIKT AHRENS, Delft University of Technology, Delft, Zuid-Holland, Netherlands

PAIGE RANDALL NORTH, Utrecht University, Utrecht, Netherlands

From Semantics to Syntax: A Type Theory for Comprehension Categories

NIYOUSHA NAJMAEI, École Polytechnique, France
NIELS VAN DER WEIDE, Radboud University Nijmegen, Netherlands
BENEDIKT AHRENS, Delft University of Technology, Netherlands
PAIGE RANDALL NORTH, Utrecht University, Netherlands

Recent models of intensional type theory have been constructed in algebraic weak factorization systems (AWFSs). AWFSs give rise to comprehension categories that feature non-trivial morphisms between types; these morphisms are not used in the standard interpretation of Martin-Löf type theory in comprehension categories.

We develop a type theory that internalizes morphisms between types, reflecting this semantic feature back into syntax. Our type theory comes with Π-, Σ-, and identity types. We discuss how it can be viewed as an extension of Martin-Löf type theory with coercive subtyping, as sketched by Coraglia and Emmenegger. We furthermore define semantic structure that interprets our type theory and prove a soundness result. Finally, we exhibit many examples of the semantic structure, yielding a plethora of interpretations.

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

Additional Key Words and Phrases: categorical semantics, comprehension categories, subtyping, type theory

ACM Reference Format:
Niyousha Najmaei, Niels van der Weide, Benedikt Ahrens, and Paige Randall North. 2026. From Semantics to Syntax: A Type Theory for Comprehension Categories. *Proc. ACM Program. Lang.* 10, POPL, Article 83 (January 2026), 30 pages. https://doi.org/10.1145/3776725

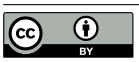
1 Introduction

There is a fruitful synergy between programming languages and mathematics, and more precisely between the study of dependent type theories and category theory. Category theory provides ways to give semantics to type theory, enabling proofs of consistency, normalization, and other desirable properties [28, 36]. On the other hand, type theories provide internal languages for categories, paving the way for concise and computer-verifiable reasoning about the mathematics in a category [16, 20, 23, 29, 46]. In summary, type theory can distill the salient features of semantic structures, and categorical structure can help in the study and development of type theories.

In this paper, we extend the synergy between type theory and category theory by giving a syntax for comprehension categories [34, 35] – one of the most common categorical structures for interpreting Martin-Löf type theory [43]. This syntax adds structure to Martin-Löf type theory.

Furthermore, we give a full analysis of the categorical structure that underlies subtyping, and we recognize this structure in many intensional models of Martin-Löf type theory.

Authors' Contact Information: Niyousha Najmaei, École Polytechnique, Palaiseau, France, najmaei@lix.polytechnique.fr; Niels van der Weide, Radboud University Nijmegen, Nijmegen, Netherlands, niels.vanderweide@ru.nl; Benedikt Ahrens, Delft University of Technology, Delft, Netherlands, B.P.Ahrens@tudelft.nl; Paige Randall North, Utrecht University, Utrecht, Netherlands, p.r.north@uu.nl.



ACM 2475-1421/2026/1-ART83
https://doi.org/10.1145/3776725

1.1 Interpretation of Type Theories in Comprehension Categories

We sketch the interpretation of type theories in categorical structures, contrasting that of MLTT with that of other type theories.

1.1.1 Interpretation of MLTT. Martin-Löf type theory (MLTT), and variations thereof, are typically given semantics in categories with structure, such as categories with families [21], contextual categories [12, 13], and type categories [47]. All of these categorical structures can be viewed as a *full (or discrete) comprehension category* [4].

A comprehension category consists (among other things) of two categories: one whose objects interpret the contexts of the type theory and one which interprets its types. Thus, at first glance, a comprehension category can express both morphisms between contexts and morphisms between types. In MLTT, however, there is only one such notion in the sense that the morphisms between both contexts and types are generated by the terms of the theory. Hence, morphisms between types can in turn be recovered from the context morphisms. The requirements that a comprehension category is discrete or that it is full then represent the two universal ways of deriving the type morphisms from the context morphisms: discreteness assumes that there are as few type morphisms as possible, whereas fullness assumes there are as many type morphisms as possible. Thus, both the requirements of discreteness and fullness are used to 'kill off' this 'extra dimension' of morphisms.

1.1.2 Interpretation of Intensional Type Theories. To give semantics to intensional type theories, such as homotopy type theory [52] and cubical type theories [16], one generally uses more refined ideas, in particular from higher category theory and homotopy theory. Specifically, such type theories are often given semantics in an *algebraic weak factorization system* (or *AWFS* for short) [24, 32]. These AWFSs are usually interesting in their own right, and they endow the category with higher dimensional structure that we can understand synthetically via the identity type. Just like other categorical structures such as categories with families, AWFSs give rise to comprehension categories. However, these comprehension categories are typically neither full nor discrete, nor are they split[1]. In other words, these comprehension categories have morphisms between types that do *not* arise from context morphisms, and their substitution structure is more intricate than that of Martin-Löf type theory itself. We argue that this extra structure in the semantics should not be ignored, and it is captured by the type theory presented here.

1.2 New Type Theories: Reflecting Semantic Features Into the Syntax

We develop a type theory that allows for synthetic reasoning about the comprehension categories arising from AWFSs— thus, in particular, about morphisms of types and about non-split substitution. To this end, we develop a new type theory, which we call *comprehension category type theory* or CCTT for short. This type theory is obtained by "reflecting" some semantic features back into traditional type theory.

We are not the first to develop type theory by reflecting semantic features back into the syntax. For instance, a type theory for non-split substitution has been developed by Curien et al. [20], who developed a type theory with explicit substitutions to reflect the non-split substitution structure of comprehension categories into type theory. Similarly, Coraglia and Emmenegger [19] studied generalized categories with families for the semantics of type theory; they discovered that, syntactically, such generalized structures give rise to a notion of coercive subtyping. In our work, we encounter both of these syntactic features, and, in particular, give a systematic and extensive account of the syntax and semantics of subtyping sketched by Coraglia and Emmenegger [19].

[1] A comprehension category is split if the chosen lifts of identities are identities and the chosen lift of any composite is the composition of the individual lifts (see Definition 2.3).

Specifically, by not imposing the restrictions of discreteness or fullness, our syntax and semantics can capture coercive subtyping [19]. As argued there, every comprehension category is equipped with a notion of subtyping, which is given by morphisms between types. Fullness expresses that coercions and terms are the same, making the subtyping, in one sense, trivial. Discreteness expresses that all coercions arise from identities of types, making the subtyping, in another sense, trivial. To faithfully model coercive subtyping, one thus need to use comprehension categories that are neither full nor discrete.

Moreover, from a practical point of view, having both terms and type morphisms in a type theory allows for tight control over definitional equalities. Specifically, when considering semantics of type theory in AWFSs, types and type morphisms are interpreted as algebras for some monad, where the algebra structure models the transport, $(a = a') \to B(a) \to B(a')$, of structure along a term of an identity type, and morphisms of algebras preserve such transport *strictly*, up to definitional equality; see also Section 2.3. Semantics in AWFSs justify adding syntactic rules expressing such definitional equalities, which only hold up to *propositional* identity in MLTT.

1.3 Contributions and Synopsis

The present paper is organized as follows.

(1) In Section 3, we design judgements and structural rules for CCTT, our language for comprehension categories. We prove soundness for our rules with respect to comprehension categories. We show how our rules can be interpreted as the rules for a type theory with subtyping, extending a sketch by Coraglia and Emmenegger [19].
(2) In Section 4, we design rules for type formers — dependent pairs, dependent functions, and identity types — for our structural rules. We prove soundness for these rules in suitably structured comprehension categories. We show our type formers can be interpreted with respect to subtyping, again extending a sketch by Coraglia and Emmenegger [19].
(3) In Section 5, we discuss a variant of CCTT with strictly functorial substitution that integrates *splitness*. This syntax is easier to work with, at the cost of having fewer models.
(4) In Section 6, we discuss related work.

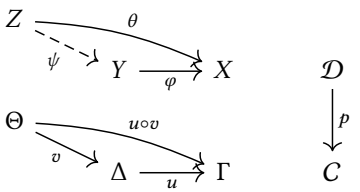
2 Review: Comprehension Categories from Algebraic Weak Factorization Systems

In this section we briefly review the categorical notions used in the remainder of the paper.

2.1 Fibrations

Fibrations can model dependent types in contexts, and substitutions. Intuitively, a fibration consists of a category $\mathcal{C}$ of contexts and context morphisms, a category $\mathcal{D}$ of types depending on contexts, and a substitution operation on types. As a reference on fibrations, see the notes by Streicher [49].

Definition 2.1. Let $p : \mathcal{D} \to \mathcal{C}$ be a functor. A morphism $\varphi : Y \to X$ in $\mathcal{D}$ is called **cartesian** if and only if for all $v : \Theta \to \Delta$ in $\mathcal{C}$ and $\theta : Z \to X$ with $p(\theta) = p(\varphi) \circ v$ there exists a unique morphism $\psi : Z \to Y$ with $p(\psi) = v$ and $\theta = \varphi \circ \psi$.



A morphism α is called **vertical** if and only if $p(\alpha)$ is an identity morphism in C. For $\Gamma \in C$, we write $\mathcal{D}_\Gamma$ for the subcategory of $\mathcal{D}$ consisting of objects X with $p(X) = \Gamma$ and morphisms α with $p(\alpha) = \mathrm{Id}_\Gamma$. The category $\mathcal{D}_\Gamma$ is called the **fiber of** p **over** Γ.

Definition 2.2. A functor $p : \mathcal{D} \to C$ is called a **(cloven) Grothendieck fibration** if and only if for all $u : \Delta \to \Gamma$ in C and $X \in \mathcal{D}_\Gamma$ we have a chosen cartesian arrow $\varphi : Y \to X$ with $p(\varphi) = u$ called a **cartesian lifting** of u to X.

The adjective "cloven" in Definition 2.2 refers to the fact that we assume the cartesian lifts to be *chosen* rather than merely *existing*, as in some other sources. Throughout this paper, all fibrations will be cloven. For brevity, we omit both "cloven" and "Grothendieck" when referring to *fibrations*.

Definition 2.3. A fibration $p : \mathcal{D} \to C$ is **split** if the chosen lifts of identities are identities; and the chosen lift of any composite is the composite of the individual lifts.

Definition 2.4. Given a category C, its **arrow category** $C^\to$ has, as objects, morphisms of C. A morphism in $C^\to$ from $f : a \to b$ to $g : c \to d$ is given by a pair (k, l) of morphisms in C, with $k : a \to c$ and $l : b \to d$, such that $l \circ f = g \circ k$.

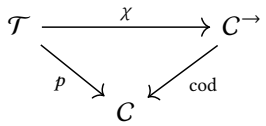
Example 2.5. Let C be a category.

- The forgetful functor $\mathrm{dom} : C^\to \to C$ sending a morphism to its domain is a fibration.
- The forgetful functor $\mathrm{cod} : C^\to \to C$ sending a morphism to its codomain is a fibration if and only if the category C has chosen pullbacks.

2.2 Comprehension Categories

In this section we recall the definition of comprehension categories.

Definition 2.6 ([34, Definition 4.1]). A **comprehension category** consists of a category C, a fibration $p : \mathcal{T} \to C$, and a functor $\chi : \mathcal{T} \to C^\to$ preserving cartesian arrows, such that the following diagram commutes.



Here, χ is called the **comprehension** and $\mathrm{dom} \circ \chi$ is denoted by χ_0.

A comprehension category is called **full** if $\chi : \mathcal{T} \to C^\to$ is fully faithful and is called **split** if $p : \mathcal{T} \to C$ is a split fibration.

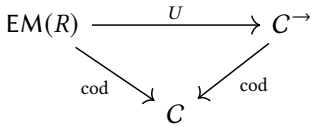
Jacobs [34, 35] uses full split comprehension categories to model type dependency. In the following example, we build intuition about the interpretation of dependent type theories in comprehension categories.

Example 2.7 ([35, Section 10.3],[40, Example 2.1.2]). In the syntactic comprehension category built from MLTT, the base category C is the category of contexts where objects are contexts and morphisms are substitutions between contexts. The fiber $\mathcal{T}_\Gamma$ over a context Γ is the category of types in context Γ. The morphisms in the fibers are such that χ is fully faithful, i.e. the comprehension category is *full*. The cloven fibration $p : \mathcal{T} \to C$ maps each type to its context and the chosen lifts are determined such that reindexing gives type substitution. In particular, for a context morphism (substitution) $s : \Gamma \to \Delta$ and $A \in \mathcal{T}_\Delta$ (type A in context Δ), s^*A is substitution by s in A. Since substitution in MLTT is strictly functorial, the resulting comprehension category is *split*. The functor χ maps $A \in \mathcal{T}_\Delta$ (type A in context Δ) to the projection from an extended context to the original context $\pi_A : \Gamma.A \to \Gamma$. In this example, the sections of such projections are the terms of type A in context Γ.

2.3 Comprehension Categories from Algebraic Weak Factorization Systems

As discussed in Example 2.7, in a comprehension category, we have a notion of morphism between types, namely morphisms in $\mathcal{T}_\Gamma$, and we have a notion of terms, which are sections of projections $\chi(A) : \Gamma.A \to \Gamma$. Often, the comprehension χ is fully faithful, and thus (up to equivalence) the inclusion of a full subcategory. In this case, the objects of $\mathcal{T}$ can be seen as objects of $C^\to$, i.e., morphisms of C, with some property. This is also the case for Example 2.7. One might want, however, to consider a category of morphisms of C, each equipped with some kind of *structure*, not just property.

That is, one could consider a monad R on $C^\to$ and take its Eilenberg-Moore category $\mathsf{EM}(R)$ together with the forgetful functor $U : \mathsf{EM}(R) \to C^\to$.



The composition $\mathsf{EM}(R) \xrightarrow{U} C^\to \xrightarrow{\mathrm{cod}} C$ (also denoted by cod above) is a fibration if C has all pullbacks (so that $C^\to \xrightarrow{\mathrm{cod}} C$ is a fibration) and U is a *discrete pullback-fibration* [11]. The functor U is a *discrete pullback-fibration* [11] if the pullback of the underlying morphism of an R-algebra can itself uniquely be given the structure of an R-algebra making the pullback square an R-algebra morphism. Bourke and Garner [11, Prop. 8] show that U is a discrete pullback-fibration if and only if R is isomorphic to a monad over cod, meaning that R lifts to a monad on $C^\to \xrightarrow{\mathrm{cod}} C$ in the slice Cat/C.

Given such a monad, the components of its unit assemble into an endofunctor on $C^\to$ with a counit – in fact it is a copointed endofunctor on $C^\to$ over $\mathrm{dom} : C^\to \to C$.

There are many examples of such monads coming from an *algebraic weak factorization system* (AWFS)[2]. In brief, an AWFS [26, 27] is a monad on $C^\to$ over $\mathrm{cod} : C^\to \to C$ together with structure making the associated copointed endofunctor on $C^\to$ a comonad (and sometimes a distributive law of the comonad over the monad).

AWFSs are fundamental when constructively studying the semantics of intensional type theory. They have been applied to construct models of both homotopy and cubical type theory [8–10, 14, 22, 24, 51]. Very roughly, the algebras of the constituent monad of one of these AWFSs are objects with *transport* and associated structure, in the terminology of homotopy type theory. Morphisms of algebras, which are morphisms of types in the associated comprehension category, are morphisms that preserve the transport *strictly*, commuting with the transport up to definitional equality.

Example 2.8 ([11, Ex. 29], cf. [32, 46]). There is a monad on the category of groupoids whose algebras are groupoids Γ together with a split (iso)fibration $T : \mathcal{E} \to \Gamma$. Morphisms from (Γ_1, T_1) to (Γ_2, T_2) consist of functors $F : \Gamma_1 \to \Gamma_2$ and $G : \mathcal{E}_1 \to \mathcal{E}_2$ such that $T_2 \circ F = G \circ T_1$ and such that G preserves the chosen lifts up to equality. The comprehension χ is given by taking the underlying functor. In the resulting comprehension category, C is the category of groupoids, the objects in the fiber over a groupoid Γ are split (iso)fibrations, and χ is an inclusion. This comprehension category is not full because functors do not preserve chosen lifts in general.

We can restrict the AWFS on categories to one on the full subcategory of groupoids. Again, the resulting comprehension category is not full.

[2]Note that not all non-full comprehension categories arise from AWFSs. In particular, comprehension categories where χ is not faithful do not arise from AWFSs as described above.

The groupoid model is fundamental in the semantics of type theory, because it refutes the uniqueness of identity proofs [32]. Currently, the groupoid model is being formalized in Lean [33].

Next we look at a different class of examples, where types are interpreted as formulas. These examples are in nature closer to refinement types than to dependent types. Whereas dependent types generally are used in proof relevant settings, refinement types are used in proof irrelevant settings. In our first example, we look at formulas as subobjects.

Example 2.9. Let $\mathcal{E}$ be a topos with subobject classifier Ω. Every topos can be equipped with an orthogonal factorization system whose left class is given by the epimorphisms and whose right class is given by the monomorphisms. Since every orthogonal factorization system also is an algebraic factorization system, we get a comprehension category where $\mathcal{C}$ is $\mathcal{E}$ and where objects of $\mathcal{T}$ are given by an object $x \in \mathcal{E}$ with a morphism $p : x \to \Omega$. The functor χ sends a morphism $p : x \to \Omega$ to the subobject classified by p. The resulting comprehension category is not full in general [18, 35].

Next we specialize Example 2.9 to get a comprehension category whose contexts are sets and whose types are predicates valued in a given Heyting algebra.

Example 2.10. Let H be a Heyting algebra. Note that we have a topos $\mathcal{E}$ of sheaves over H, which can equivalently be described as partial equivalence relations valued in H [30]. We have a fully faithful functor F from Set to $\mathcal{E}$ sending every set X to the partial equivalence relation given by equality. Objects in the image of F are called discrete. From this topos we obtain a comprehension category, which we restrict to the contexts that are discrete. In the resulting comprehension category, $\mathcal{C}$ is the category of sets and objects of $\mathcal{T}$ consist of a set Γ together with a map $p : \Gamma \to H$. Morphisms in $\mathcal{T}$ from (Γ_1, p_1) to (Γ_2, p_2) are given by functions $f : \Gamma_1 \to \Gamma_2$ such that for each $x \in \Gamma_1$ we have $p_1(x) \leq p_2(f(x))$. The functor χ maps every (Γ, p) to the set $\{x \in \Gamma \mid \top \leq p(x)\}$. This comprehension category is not full in general. In fact, if H is the collection of open subsets of some topological space X, then this comprehension category only is full if X is indiscrete.

3 Syntax from Comprehension Categories

We present the judgements and structural rules of a type theory with explicit substitution corresponding to a comprehension category. To be clear, we do not impose any requirements of fullness or splitness on the comprehension category.

For brevity, we leave out certain rules that express coherences and functoriality conditions. For the full list of structural rules, see Appendix A of the preprint of this work [45].

3.1 Judgements

The judgements of the type theory are as follows:

(1) Γ ctx, which is read as 'Γ is a context';
(2) $\Gamma \vdash s : \Delta$, which is read as 's is a substitution from Γ to Δ', where Γ, Δ ctx;
(3) $\Gamma \vdash s \equiv s' : \Delta$, which is read as '$s$ is equal to s'', where $\Gamma \vdash s, s' : \Delta$;
(4) $\Gamma \vdash A$ type, which is read as 'A is a type in context Γ', where Γ ctx;
(5) $\Gamma \mid A \vdash t : B$, which is read as '$t$ is a type morphism from A to B in context Γ', where $\Gamma \vdash A, B$ type;
(6) $\Gamma \mid A \vdash t \equiv t' : B$, which reads as '$t$ and t' are equal', where $\Gamma \mid A \vdash t, t' : B$.

The judgement $\Gamma \vdash s : \Delta$ can also be read as 's is a context morphism from Γ to Δ', since this judgement is interpreted as a morphism $[\![s]\!] : [\![\Gamma]\!] \to [\![\Delta]\!]$ in the category of contexts $\mathcal{C}$ (see Section 3.5). Similarly, the judgement $\Gamma \mid A \vdash t : B$ is read as 't is a type morphism from A to B', since t is interpreted as a morphism from $[\![A]\!]$ to $[\![B]\!]$ in $\mathcal{T}_{[\![\Gamma]\!]}$ (see Section 3.5).

Unlike Martin-Löf type theory, this type theory has explicit substitution in the sense of Abadi et al. [1]. Another difference between this type theory and Martin-Löf type theory is the terms. As we will see in Section 3.5, the terms of this type theory, which correspond to judgements of the form $\Gamma \mid A \vdash t : B$, are interpreted as morphisms in $\mathcal{T}$. The terms of Martin-Löf type theory, however, are interpreted as certain morphisms in C — in particular, as sections of the projection context morphisms. We will refer to these sections as **MLTT terms**. In Notation 3.4, we define a notation for MLTT terms.

Related Work 3.1 (Ahrens et al. [5]). Judgements 1, 2, 4 and 5 are the same as the corresponding judgements in the type theory for comprehension **bi**categories in the work of Ahrens et al. [5]. They read Judgement 5 as 't is a term of type B depending on A in context Γ'. In the present work, we read Judgement 5 as 't is a type morphism from A to B in context Γ' for generality.

3.2 Rules for Context and Type Morphisms

We want contexts and substitutions (context morphisms) to form a category — the category C in a comprehension category $(C, \mathcal{T}, p, \chi)$. Hence, the rules of the type theory concerning substitution follow the usual axioms for a category.

$$\frac{\Gamma \ \mathsf{ctx}}{\Gamma \vdash 1_\Gamma : \Gamma}\ \text{ctx-mor-id} \qquad \frac{\Gamma \vdash s : \Delta \quad \Delta \vdash s' : \Theta}{\Gamma \vdash s' \circ s : \Theta}\ \text{ctx-mor-comp}$$

$$\frac{\Gamma \vdash s : \Delta}{\begin{array}{l}\Gamma \vdash s \circ 1_\Gamma \equiv s : \Delta \\ \Gamma \vdash 1_\Delta \circ s \equiv s : \Delta\end{array}}\ \text{ctx-id-unit} \qquad \frac{\Gamma \vdash s : \Delta \quad \Delta \vdash s' : \Theta \quad \Theta \vdash s'' : \Phi}{\Gamma \vdash s'' \circ (s' \circ s) \equiv (s'' \circ s') \circ s : \Phi}\ \text{ctx-comp-assoc}$$

Similarly, we want types and type morphisms to form a category — the category $\mathcal{T}$ in a comprehension category $(C, \mathcal{T}, p, \chi)$. Hence, we postulate the following rules, which follow the usual axioms for a category.

$$\frac{\Gamma \vdash A \ \mathsf{type}}{\Gamma \mid A \vdash 1_A : A}\ \text{ty-mor-id} \qquad \frac{\Gamma \mid A \vdash t : B \quad \Gamma \mid B \vdash t' : C}{\Gamma \mid A \vdash t' \circ t : C}\ \text{ty-mor-comp}$$

$$\frac{\Gamma \mid A \vdash t : B}{\begin{array}{l}\Gamma \mid A \vdash t \circ 1_A \equiv t : B \\ \Gamma \mid A \vdash 1_B \circ t \equiv t : B\end{array}}\ \text{ty-id-unit} \qquad \frac{\Gamma \mid A \vdash t : B \quad \Gamma \mid B \vdash t' : C \quad \Gamma \mid C \vdash t'' : D}{\Gamma \mid A \vdash t'' \circ (t' \circ t) \equiv (t'' \circ t') \circ t : D}\ \text{ty-comp-assoc}$$

Notation 3.2. Similar to Ahrens et al. [5, Section 8], we define the following notations, which each stand for four judgements.

(1) $\Gamma \ \tilde{\vdash} \ s : \Delta$ stands for the following four judgements.
- $\Gamma \vdash s : \Delta$
- $\Delta \vdash s' : \Gamma$
- $\Delta \vdash s \circ s' \equiv 1_\Delta : \Delta$
- $\Gamma \vdash s' \circ s \equiv 1_\Gamma : \Gamma$

(2) $\Gamma \mid A \ \tilde{\vdash} \ t : B$ stands for the following four judgements.
- $\Gamma \mid A \vdash t : B$
- $\Gamma \mid B \vdash t' : A$
- $\Gamma \mid B \vdash t \circ t' \equiv 1_B : B$
- $\Gamma \mid A \vdash t' \circ t \equiv 1_A : A$

Given $\Gamma \ \tilde{\vdash} \ s : \Delta$, we write $\Delta \vdash s^{-1} : \Gamma$ for the inverse of s. Similarly, t^{-1} denotes the inverse of t.

Remark 3.3. We can postulate an empty context and a unique substitution from each context to it, which corresponds to having a terminal object in the category C. For this, the following rules can be added to the type theory. We do not discuss this further in this paper.

$$\frac{}{\diamond \ \mathsf{ctx}}\ \text{empty-ctx} \qquad \frac{\Gamma \ \mathsf{ctx}}{\Gamma \vdash \langle\rangle_\Gamma : \diamond}\ \text{empty-ctx-mor} \qquad \frac{\Gamma \vdash s : \diamond}{\Gamma \vdash s \equiv \langle\rangle_\Gamma : \diamond}\ \text{empty-ctx-mor-unique}$$

3.3 Rules for Context Extension

The rules for context extension mirror the action of the comprehension functor $\chi : \mathcal{T} \to C^{\to}$ in a comprehension category $(C, \mathcal{T}, p, \chi)$. Particularly, the rules correspond to the functoriality of χ restricted to $\mathcal{T}_\Gamma \to C/\Gamma$ for each $\Gamma \in C$.

$$\frac{\Gamma \vdash A\ \mathsf{type}}{\Gamma.A\ \mathsf{ctx}}\ \text{ext-ty} \qquad \frac{\Gamma \mid A \vdash t : B}{\Gamma.A \vdash \Gamma.t : \Gamma.B}\ \text{ext-tm} \qquad \frac{\Gamma \mid A \vdash t : B \quad \Gamma \mid B \vdash t' : C}{\Gamma.A \vdash \Gamma.(t' \circ t) \equiv \Gamma.t' \circ \Gamma.t : \Gamma.B}\ \text{ext-comp}$$

$$\frac{\Gamma \vdash A\ \mathsf{type}}{\Gamma.A \vdash \Gamma.1_A \equiv 1_{\Gamma.A} : \Gamma.A}\ \text{ext-id} \qquad \frac{\Gamma \vdash A\ \mathsf{type}}{\Gamma.A \vdash \pi_A : \Gamma}\ \text{ext-proj} \qquad \frac{\Gamma \mid A \vdash t : B}{\Gamma.A \vdash \pi_B \circ \Gamma.t \equiv \pi_A : \Gamma}\ \text{ext-c}$$

Before moving on to the rules for substitution, we introduce the following notation for MLTT terms.

Notation 3.4. We define the notation $\Gamma \vdash a : A$ for MLTT terms, which stands for two judgements $\Gamma \vdash a : \Gamma.A$ and $\Gamma \vdash \pi_A \circ a \equiv 1_\Gamma : \Gamma$.

3.4 Rules for Substitution

Unlike standard Martin-Löf type theory, this type theory features explicit substitution in the syntax. By the Grothendieck construction, we know that the fibration $p : \mathcal{T} \to C$, in a comprehension category $(C, \mathcal{T}, p, \chi)$, can be thought of as a pseudofunctor of the form $C^{\mathrm{op}} \to \mathrm{Cat}$. The rules regarding substitution will be interpreted by this pseudofunctor. Specifically, the fibration $p : \mathcal{T} \to C$ gives rise to reindexing functors of the form $s^* : \mathcal{T}_\Delta \to \mathcal{T}_\Gamma$ for each $s : \Gamma \to \Delta$ in C, and two natural isomorphisms corresponding to composition of reindexing functors and reindexing along identity morphisms. Namely, for each $s : \Gamma \to \Delta$ and $s' : \Delta \to \Theta$ in C, we have a natural isomorphism $i^{\mathsf{comp}} : (s' \circ s)^* \cong s^* \circ s'^*$, and for each $A \in \mathcal{T}_\Gamma$ we have an isomorphism $i^{\mathsf{id}}_A : 1^*_\Gamma A \cong A$.

The rules for substitution are as follows.

$$\frac{\Gamma \vdash s : \Delta \quad \Delta \vdash A\ \mathsf{type}}{\Gamma \vdash A[s]\ \mathsf{type}}\ \text{sub-ty} \qquad \frac{\Gamma \vdash s : \Delta \quad \Delta \mid A \vdash t : B}{\Gamma \mid A[s] \vdash t[s] : B[s]}\ \text{sub-tm}$$

$$\frac{\Gamma \vdash s : \Delta \quad \Delta \vdash A\ \mathsf{type}}{\Gamma \mid A[s] \vdash 1_A[s] \equiv 1_{A[s]} : A[s]}\ \text{sub-prs-id} \qquad \frac{\Gamma \vdash s : \Delta \quad \Delta \mid A \vdash t : B \quad \Delta \mid B \vdash t' : C}{\Gamma \mid A[s] \vdash (t' \circ t)[s] \equiv t'[s] \circ t[s] : C[s]}\ \text{sub-prs-comp}$$

$$\frac{\Gamma \vdash A\ \mathsf{type}}{\Gamma \mid A[1_\Gamma] \ \tilde{\vdash}\ i^{\mathsf{id}}_A : A}\ \text{sub-id} \qquad \frac{\Gamma \vdash s : \Delta \quad \Delta \vdash s' : \Theta \quad \Theta \vdash A\ \mathsf{type}}{\Gamma \mid A[s' \circ s] \ \tilde{\vdash}\ i^{\mathsf{comp}}_{A,s',s} : A[s'][s]}\ \text{sub-comp}$$

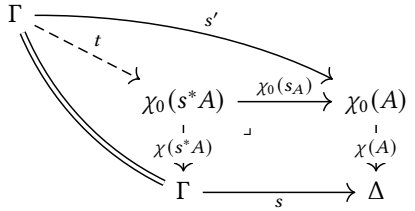
$$\frac{\Gamma \mid A \vdash t : B}{\Gamma \mid A[1_\Gamma] \vdash t[1_\Gamma] \equiv i^{\mathsf{id}^{-1}}_B \circ t \circ i^{\mathsf{id}}_A : B[1_\Gamma]}\ \text{sub-tm-id}$$

$$\frac{\Gamma \vdash s : \Delta \quad \Delta \vdash s' : \Theta \quad \Theta \mid A \vdash t : B}{\Gamma \mid A[s' \circ s] \vdash t[s' \circ s] \equiv i^{\mathsf{comp}^{-1}}_{B,s',s} \circ t[s'][s] \circ i^{\mathsf{comp}}_{A,s',s} : B[s' \circ s]}\ \text{sub-tm-comp}$$

Remark 3.5. In our rules for substitution, composition and identity of context morphisms are only preserved up to isomorphism rather than up to equality. The reason behind this choice is that in many comprehension categories, in particular those arising from AWFSs, substitution laws only hold up to isomorphism. To guarantee that our syntax can be interpreted in such comprehension categories, it is necessary to relax the substitution laws. In Section 5 we present a split variant of our syntax, where we consider these rules up to equality.

In a comprehension category $(C, \mathcal{T}, p, \chi)$, the comprehension of each cartesian lift $s_A : s^*A \to A$ is a pullback square in C, since χ preserves cartesian morphisms. This means that for each morphism

$s : \Gamma \to \Delta$ in $\mathcal{C}$, morphisms $s' : \Gamma \to (\chi_0 A)$ in $\mathcal{C}$ such that $\chi(A) \circ s' = s$ correspond to sections t of $\chi(s^*A)$ such that $\chi_0(s_A) \circ t = s'$.



The following rules capture the idea that context morphisms are built out of MLTT style terms, that is sections of projection context morphisms $\Gamma.A \vdash \pi_A : \Gamma$.

$$\frac{\Delta \vdash A \text{ type} \quad \Gamma \vdash s : \Delta \quad \Gamma \vdash t : A[s]}{\Gamma \vdash (s,t) : \Delta.A}\ \text{sub-ext} \qquad \frac{\Gamma \vdash s : \Delta.A}{\Gamma \vdash p_2(s) : A[\pi_A \circ s]}\ \text{sub-proj}$$

$$\frac{\Delta \vdash A \text{ type} \quad \Gamma \vdash s : \Delta \quad \Gamma \vdash t : A[s]}{\begin{array}{l}\Gamma \vdash \pi_A \circ (s,t) \equiv s : \Delta \\ \Gamma \vdash p_2(s,t) \equiv t : \Gamma.A[s]\end{array}}\ \text{sub-beta} \qquad \frac{\Gamma \vdash s : \Delta.A}{\Gamma \vdash (\pi_A \circ s, p_2(s)) \equiv s : \Delta.A}\ \text{sub-eta}$$

We also have rules for functoriality of $p_2(-)$ (see Rules sub-proj-id and sub-proj-comp in the preprint of this work [45, Appendix A]).

Before concluding the rules, we discuss the following derived rule which is frequently used in the rest of the paper.

Proposition 3.6. *From the above-mentioned rules, we can derive the following rule.*

$$\frac{\Delta \vdash A \text{ type} \quad \Gamma \vdash s : \Delta}{\Gamma.A[s] \vdash s.A : \Delta.A}\ \textit{ctx-mor-lift}$$

Proof. The context morphism $s.A$ is $(s \circ \pi_{A[s]}, \Gamma.(i^{\text{comp}}_{A,s,\pi_{A[s]}}[\pi_{A[s]}]) \circ p_2(1_{\Gamma.A[s]}))$ and Rule ctx-mor-lift can be derived using Rules ctx-mor-comp, ext-proj, sub-comp, sub-tm, ext-tm, sub-proj and sub-ext. □

Lastly, we have the following rule which characterizes the behavior of $\Gamma.t[s]$. In this rule, we use the notation introduced in Proposition 3.6.

$$\frac{\Delta \mid A \vdash t : B \quad \Gamma \vdash s : \Delta}{\Gamma.A[s] \vdash s.B \circ \Gamma.t[s] \equiv \Delta.t \circ s.A : \Delta.B}\ \text{tm-sub-coh}$$

In addition to these rules, we also have the rules related to $\equiv$ being a congruence for all the judgements. For a complete list of the rules, see Appendix A of the preprint [45].

Remark 3.7. In a comprehension category $(\mathcal{C}, \mathcal{T}, p, \chi)$, for each object A in $\mathcal{C}$ and equal morphisms s and s' in $\mathcal{C}$, we have $s^*A = (s')^*A$. As we do not have a judgement for equality of types, particularly a judgement of the form $\Gamma \vdash A[s] \equiv A[s']$, we can not express this idea syntactically. Instead, we add Rule sub-cong (see [45, Figure 5]) to the type theory.

$$\frac{\Delta \vdash A \text{ type} \quad \Gamma \vdash s \equiv s' : \Delta}{\Gamma \mid A[s] \ \tilde{\vdash}\ i^{\text{sub}}_{A,s,s'} : A[s']}\ \text{sub-cong}$$

This rule states that given a type A in context Δ and two equal context morphisms s and s' from Γ to Δ, types $A[s]$ and $A[s']$ are isomorphic in the sense that there are two context morphism $i^{\text{sub}}_{A,s,s'} : A[s] \to A[s']$ and $i^{\text{sub}}_{A,s',s} : A[s'] \to A[s]$ and their compositions are equal to the identity context morphisms.

The coherence rules regarding $i^{\text{sub}}_{A,s,s'}$ are stated in the preprint [45, Figure 5].

Definition 3.8. We define **CCTT** to be the judgements described in Section 3.1 together with the rules in Figure 4 and 5 in Appendix A of the preprint [45].

Remark 3.9. Note that our syntax does not contain any coherence rules that express the commutativity of diagrams built out of $i^{\mathsf{comp}}_{A,s_1,s_2}$ and i^{id}_A. While it is possible to add such coherence equations, we refrain from doing so. This approach is similar to the work by Curien, Garner, and Hofmann who also consider a non-split syntax for type theory [20]. In their syntax, there is an additional operation on terms, which they call explicit coercion. This operation is used to apply the morphisms $i^{\mathsf{comp}}_{A,s_1,s_2}$ and i^{id}_A to terms. Instead of adding coherence rules to syntax, they show that every diagram built out of $i^{\mathsf{comp}}_{A,s_1,s_2}$ and i^{id}_A commutes in every model, and that whenever two terms are equal after removing all coercions, they have the same denotation in every model. We conjecture that their methods can be applied to our setting; hence, we do not dicsuss the coherence rules in this paper.

3.5 Soundness: Interpretation in a Comprehension Category

We establish soundness of the rules of the type theory by giving an interpretation of the type theory in every comprehension category. Note that there is no assumption of fullness or splitness on the comprehension category.

Theorem 3.10 (Soundness of Structural Rules). *Every comprehension category models the rules of CCTT.*

Let $(\mathcal{C}, \mathcal{T}, p, \chi)$ be a comprehension category. The judgements are interpreted as follows:

(1) Γ ctx is interpreted as an object $[\![\Gamma]\!]$ in $\mathcal{C}$;
(2) $\Gamma \vdash s : \Delta$ is interpreted as a morphism $[\![s]\!] : [\![\Gamma]\!] \to [\![\Delta]\!]$ in $\mathcal{C}$;
(3) $\Gamma \vdash s \equiv s' : \Delta$ is interpreted as $[\![s]\!] = [\![s']\!]$;
(4) $\Gamma \vdash A$ type is interpreted as an object $[\![A]\!]$ in $\mathcal{T}_{[\![\Gamma]\!]}$;
(5) $\Gamma \mid A \vdash t : B$ is interpreted as a morphism $[\![t]\!] : [\![A]\!] \to [\![B]\!]$ in $\mathcal{T}_{[\![\Gamma]\!]}$;
(6) $\Gamma \mid A \vdash t \equiv t' : B$ is interpreted as $[\![t]\!] = [\![t']\!]$.

The interpretation of the rules can be found in the preprint [45, Appendix B].

3.6 Subtyping

One can regard type morphisms as witnesses of coercive subtyping. Coraglia and Emmenegger explore this in **generalized categories with families**, a structure equivalent to (not necessarily full) comprehension categories [19]. In this view, a judgement $\Gamma \mid A \vdash t : B$ can be seen as expressing that t is a witness for A being a subtype of B. In Coraglia and Emmenegger's notation, this is expressed as $\Gamma \vdash A \leq_t B$. We focus exclusively on proof-relevant subtyping. A comparison between proof-relevant and proof-irrelevant subtyping is given by Coraglia and Emmenegger [19, Section 2.3].

Proposition 3.11 (Subsumption rule). *From the rules of CCTT, we can derive the following rule.*

$$\frac{\Gamma \mid A \vdash t : B \qquad \Gamma \vdash a : A}{\Gamma \vdash \Gamma.t \circ a : B}$$

Proof. The rule can be derived using Rules ext-tm, ctx-mor-comp and ext-c. □

Proposition 3.11 corresponds to subsumption in coercive subtyping. The rule states that if A is a subtype of B, then a (MLTT) term of type A can be coerced to a term of type B.

Proposition 3.12 (Coercions commute with substitution). *From the rules of CCTT, we can derive the following rule.*

$$\frac{\Delta \mid A \vdash t : B \quad \Delta \vdash a : A \quad \Gamma \vdash s : \Delta}{\Gamma \vdash (\Delta.t \circ a)[s] \equiv \Gamma.(t[s]) \circ a[s] : \Gamma.B[s]}$$

where $a[s] \coloneqq p_2(s \circ a)$.

Proposition 3.12 expresses that substitution and coercion commute. In practice, it allows us to compute substitutions in terms that contain coercions.

Table 1. Meaning of the rules of CCTT when the judgement $\Gamma \mid A \vdash t : B$ expresses subtyping and relation to the rules by Coraglia and Emmenegger [19].

Rule of CCTT	Meaning under Subtyping	Rule in [19]
ty-mor-id	Reflexivity of subtyping witnessed by 1_A	-
ty-mor-comp	$A \leq_f B$ and $B \leq_g C$ give $A \leq_{g \circ f} C$.	*Trans* and *Sbsm*
ty-id-unit	Each 1_A is an identity for witness composition.	-
ty-comp-assoc	Composition of witnesses is associative.	-
ext-tm	$A \leq_t B$ gives a context morphism $\Gamma.A \vdash \Gamma.t : \Gamma.B$.	-
ext-id	Context morphism $\Gamma.1_A$ is equal to the identity.	-
ext-comp	For witnesses f and g, $\Gamma.g \circ f$ is equal to $\Gamma.f \circ \Gamma.g$.	-
sub-tm	Substitution preserves subtyping.	*Wkn* and *Sbst*
sub-prs-id	Substitution preserves the identity witness.	-
sub-prs-comp	Substitution preserves composition of witnesses.	-

In Table 1, we discuss the meaning of the rules of CCTT that involve a judgement of the form $\Gamma \mid A \vdash t : B$ from the subtyping perspective. We also show how these rules relate to the rules discussed by Coraglia and Emmenegger [19].

We now discuss the interpretation of CCTT and the subtyping structure for some of the examples from Section 2.3.

Example 3.13 (Example 2.8 ctd.). In this example, a context Γ is interpreted as a groupoid $[\![\Gamma]\!]$. A type A in context Γ is interpreted as a split isofibration $[\![A]\!] : [\![\Gamma.A]\!] \to [\![\Gamma]\!]$. Type morphisms from A to A' in context Γ, i.e. witnesses for a subtyping relation $A \leq A'$, are interpreted as morphisms of split fibrations of the form $[\![A]\!] \to [\![A']\!]$ preserving the chosen lifts up to equality. A context morphism $\Gamma \vdash s : \Delta$ is interpreted as a functor of the form $[\![s]\!] : [\![\Gamma]\!] \to [\![\Delta]\!]$. Given $\Delta \vdash A$ type and $\Gamma \vdash s : \Delta$, the type $A[s]$ in context Γ is interpreted as the pullback of $[\![A]\!]$ along $[\![s]\!]$.

Example 3.14 (Example 2.10 ctd.). Let H be a Heyting algebra. A context Γ is interpreted as a set $[\![\Gamma]\!]$. A type A in context Γ is interpreted as an H-values predicate $[\![A]\!] : [\![\Gamma]\!] \to H$. We have a type morphism from A to A' in context Γ if $[\![A]\!]$ entails $[\![A']\!]$, i.e. for all $x \in \Gamma$ we have $[\![A]\!](x) \leq [\![A']\!](x)$. The subtyping relation interpreted in this example is proof-irrelevant in the sense that each hom-set in the fibers has at most one element. Given $\Delta \vdash A$ type and $\Gamma \vdash s : \Delta$, the type $A[s]$ in context Γ is interpreted as $[\![A]\!] \circ [\![s]\!]$. Context extension $\Gamma.A$ is interpreted as comprehension of $[\![A]\!]$.

4 Extending CCTT with Type Formers

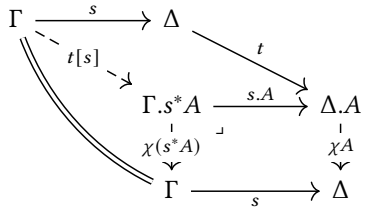
In this section, we develop syntactic rules — on top of CCTT— and semantic structures for interpreting these rules — on top of non-full comprehension categories—, for Π-, Σ-, and identity types, respectively.

In detail, we define the semantic structures for these type formers in Definitions 4.2, 4.16 and 4.29. We then discuss functoriality conditions on those structures that allow us to use type morphisms to interpret subtyping (Definitions 4.6, 4.19 and 4.31). Subsequently, we extend CCTT with functorial Π-, Σ- and identity types (Definitions 4.12, 4.25 and 4.35). We also prove soundness by giving an interpretation of the rules in any comprehension category with suitable structure for each case (Theorems 4.13, 4.26 and 4.36). Finally, we briefly discuss how CCTT with functorial Π-, Σ- and identity types supports subtyping (Related Work 4.15 and 4.28 and Remark 4.37).

See the preprint [45, Appendix B] for the proofs of the soundness theorems (Theorems 4.13, 4.26 and 4.36).

Notation 4.1. In the remainder of this paper, we also use the following notations in comprehension categories to highlight how the syntax relates to the semantics.

(1) We use π_A to denote χA for $A \in \mathcal{T}_\Gamma$.
(2) We use $\Gamma.A$ to denote $\chi_0 A$ for $\Gamma \in C$ and $A \in \mathcal{T}_\Gamma$.
(3) We use $A[s]$ to denote s^*A for $s : \Gamma \to \Delta$ and $A \in \mathcal{T}_\Delta$.
(4) We use $t[s]$ to denote the morphism given by the universal property of the following pullback square:



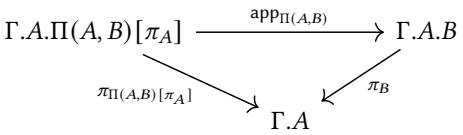
for a section t of χA, $A \in \mathcal{T}_\Delta$ and $s : \Gamma \to \Delta$.

4.1 Functorial Π-Types

In this section, we define semantic structure for Π-types in non-full comprehension categories. We then discuss the necessary functoriality conditions that allow us to use type morphisms to interpret subtyping. We extend CCTT with functorial Π-types and prove soundness by giving an interpretation of the rules in any comprehension category with functorial Π-types. We also discuss how CCTT with functorial Π-types supports subtyping.

Definition 4.2 ([39, Definition 3.2.2.3]). Let $(C, \mathcal{T}, p, \chi)$ be a comprehension category. Given $\Gamma \in C$, $A \in \mathcal{T}_\Gamma$ and $B \in \mathcal{T}_{\Gamma.A}$, a **dependent product** for Γ, A and B consists of:

(1) an object $\Pi(A, B) \in \mathcal{T}_\Gamma$;
(2) a morphism $\mathrm{app}_{\Pi(A,B)} : \Gamma.A.\Pi(A, B)[\pi_A] \to \Gamma.A.B$ making the following diagram commute;



(3) a function $\lambda_{\Pi(A,B)}$ giving for each section $b : \Gamma.A \to \Gamma.A.B$ of π_B, a section $\lambda_{\Pi(A,B)}(b) : \Gamma \to \Gamma.\Pi(A, B)$ of $\pi_{\Pi(A,B)}$ such that $\lambda_{\Pi(A,B)}(-)$ and $\mathrm{app}_{\Pi(A,B)} \circ (-)[\pi_A]$ establish a bijection between sections of π_B and sections of $\pi_{\Pi(A,B)}$.

We briefly draw the connection between Definition 4.2 and Π-types in syntax. The object $\Pi(A, B)$ in $\mathcal{T}_\Gamma$ corresponds to a type in context Γ and is the dependent product of types A and B. The morphism $\mathrm{app}_{\Pi(A,B)} : \Gamma.A.\Pi(A, B)[\pi_A] \to \Gamma.A.B$ in C gives the application of a dependent

function . The function $\lambda_{\Pi(A,B)}$ is the usual λ-abstraction. The bijection between sections of π_B and sections of $\pi_{\Pi(A,B)}$ given by $\mathrm{app}_{\Pi(,)}$ and $\lambda_{\Pi(,)}$ corresponds to the usual β- and η-conversion for dependent products.

Related Work 4.3 (Jacobs [34]). Jacobs interprets Π-types in a full comprehension category with right adjoints to weakening functors and an extra condition that the comprehension preserves products [34, Subsection 5.1]. In a full comprehension category, Definition 4.2 gives structure equivalent to Jacobs' definition.

In a full comprehension category, morphisms in $\mathcal{T}_\Gamma$ can be conflated with morphisms in $\mathcal{C}/\Gamma$, for each Γ. We, however, do not assume fullness. Hence, it is particularly important to make a distinction between the structure added to $\mathcal{C}$ and the structure added to $\mathcal{T}$. In Jacobs' definition, the structure of app and lam is added to the category $\mathcal{T}$ in a comprehension category $(\mathcal{C}, \mathcal{T}, p, \chi)$. Since we work with non-full comprehension categories and since we do not want app and lam to be type morphisms, we add them as a morphism in $\mathcal{C}$ and a function on terms respectively. The structure related to subtyping is added to $\mathcal{T}$. This is why we use Definition 4.2.

Related Work 4.4 (Lumsdaine and Warren [40]). Lumsdaine and Warren define dependent products in a full comprehension category and take $\mathrm{app}_{\Pi(A,B)}$ to be a morphism $\mathrm{app}_{\Pi(A,B)} : \Pi(A,B)[\pi_A] \to B$ in $\mathcal{T}_{\Gamma.A}$ [40, Definition 3.4.2.1]. In Definition 4.2, $\mathrm{app}_{\Pi(A,B)}$ is a morphism from $\Gamma.A.\Pi(A,B)[\pi_A]$ to $\Gamma.A.B$ in $\mathcal{C}$, as we do not assume fullness (see Related Work 4.3). In a full comprehension category, Definition 4.2 is equivalent to the definition of Lumsdaine and Warren.

Related Work 4.5 (Lindgren [39]). Lindgren uses the term **strong** dependent products to refer to what we call dependent products [39, Definition 3.2.2.3]. We do not make this distinction, as we only consider the strong case.

Lindgren shows that Definition 4.2 can equivalently be expressed in terms of relative adjoints [39, Propositions 3.2.4.2 and 3.2.4.4].

To be able to use type morphisms to interpret subtyping, we need to add certain functoriality conditions which formalize the intuition that Π-types act contravariantly on the first argument, and covariantly on the second in the context of subtyping. In particular, given subtyping relations $A' \leq_f A$ and $B[\Gamma.f] \leq_g B'$ we expect to have $\Pi(A,B) \leq_{\Pi(f,g)} \Pi(A',B')$, since Π acts contravariantly on the first argument, and covariantly on the second one. The coercion function for $\Pi(A,B) \leq_{\Pi(f,g)} \Pi(A',B')$ acts as follows. Given a dependent function $t : \Pi(A,B)$, the coerced dependent function $t' : \Pi(A',B')$ takes a term $a' : A'$, coerces it to a term $a : A$ using f, applies t to a to get a term of type B, coerces it to a term of type B' using g, and finally applies a λ-abstraction to get a term of type $\Pi(A',B')$.

Now we define what it means for a comprehension category to have functorial Π-types. For this, we add the structure defined in Definition 4.2 to a comprehension category, postulate a Beck-Chevalley condition, i.e. that this structure is preserved under substitution, and postulate suitable functoriality conditions.

Definition 4.6. A comprehension category $(\mathcal{C}, \mathcal{T}, p, \chi)$ **has functorial Π-types** if it is equipped with a function giving for each $\Gamma \in \mathcal{C}$, $A \in \mathcal{T}_\Gamma$ and $B \in \mathcal{T}_{\Gamma.A}$, $\Pi(A,B)$, $\mathrm{app}_{\Pi(A,B)}$ and $\lambda_{\Pi(A,B)}$ as Definition 4.2 such that:

(1) for each $s : \Gamma \to \Delta$ we have an isomorphism $i_{\Pi(A,B),s} : \Pi(A[s], B[s.A]) \cong \Pi(A,B)[s]$ in $\mathcal{T}_\Gamma$;

(2) for each section $b : \Gamma.A \to \Gamma.A.B$ of π_B the following diagrams commute

$$\begin{array}{ccc} \Gamma.\Pi(A[s], B[s.A]) & \xrightarrow{s.\Pi(s_A, s_B)} & \Delta.\Pi(A, B) \\ \uparrow{\scriptstyle \lambda_{\Pi(A[s],B[s.A])}(b[s.A])} & & \uparrow{\scriptstyle \lambda_{\Pi(A,B)}(b)} \\ \Gamma & \xrightarrow[s]{} & \Delta \end{array}$$

where $s.\Pi(s_A, s_B) \coloneqq s.\Pi(A, B) \circ \Gamma.i_{\Pi(A,B),s}$;

(3) the comprehension category is equipped with a function giving for each $f : A' \to A$ in $\mathcal{T}_\Gamma$ and $g : B[\Gamma.f] \to B'$ in $\mathcal{T}_{\Gamma.A'}$, a morphism $\Pi(f, g) : \Pi(A, B) \to \Pi(A', B')$ in $\mathcal{T}_\Gamma$;

(4) $\Gamma.\Pi(f, g)$ is the following composition in $\mathcal{C}$;

$$\Gamma.\Pi(A, B) \xrightarrow{\lambda(\Gamma.A.g[\mathrm{app}_{\Pi(A,B)}[\Gamma.f]])} \Gamma.\Pi(A, B).\Pi(A', B')[\pi_{\Pi(A,B)}] \xrightarrow{\pi_{\Pi(A,B)}.\Pi(A',B')} \Gamma.\Pi(A', B')$$

(see the preprint [45, Appendix C] for more detail);

(5) $\Pi(-,-)$ preserves identity, i.e. we have $\Pi(1_A, i_B^{\mathrm{id}}) = 1_{\Pi(A,B)}$ for each suitable A and B, where $i_B^{\mathrm{id}} : B[1_{\Gamma.A}] \cong B$;

(6) $\Pi(-,-)$ preserves composition, i.e. we have $\Pi(f \circ f', g' \circ g[\Gamma.f']) = \Pi(f', g') \circ \Pi(f, g)$ for each suitable f' and g';

(7) $i_{\Pi(A,B),-}$ is functorial in that it preserves i^{iso} and i^{comp} (see [45, Appendix D] for more detail).

Items 1 and 2 of Definition 4.6 state that Π and λ are preserved under substitution respectively. Consequently, app is also preserved under substitution. Items 3 to 7 give the functoriality conditions which formalize the variance of Π-types on the arguments for subtyping. Item 4 expresses compatibility of the type morphism structure on the category $\mathcal{T}$ with the type former structure on the category $\mathcal{C}$.

The following proposition states that Definition 4.6 is compatible with Jacobs' definition of comprehension categories with products.

Proposition 4.7 (Relation to Jacobs [35]). *Every **full** comprehension category with products where the comprehension functor preserves products in the sense of Jacobs [34, Section 5.1] has functorial Π-types in the sense of Definition 4.6.*

Related Work 4.8 (Coraglia and Emmenegger [19] and Coraglia and Di Liberti [17]). Coraglia and Emmenegger [19] define Π-types for generalized categories with families, a structure equivalent to comprehension categories [18]. This definition is similar to Definition 4.6 regarding variance on the first and the second argument. In another work, Coraglia and Di Liberti [17] present an alternative definition for Π-types in generalized categories with families (there called DTT), which is covariant in both arguments.

Related Work 4.9 (Gambino and Larrea [24]). Gambino and Larrea [24] also define Π-types for comprehension categories, and there are a couple of differences to note. First, Gambino and Larrea [24] do not require their Π-types to be functorial, whereas we do (Items 3 to 6 in Definition 4.6). Second, while Gambino and Larrea [24] phrase (pseudo) stability under substitution by postulating suitable Cartesian morphisms, we use explicit isomorphisms. These different ways of phrasing preservation are equivalent, and the advantage of our way is that it directly gives us a derivation rule in type theory.

We now discuss examples of comprehension categories with functorial Π-types, including those that arise from AWFSs (see Section 2.3). In a large class of AWFSs, Π-types can be interpreted. More specifically, if an AWFS satisfies the **exponentiability property** and comes equipped with a

functorial Frobenius structure, then its associated comprehension category has Π-types [24, Proposition 4.6]. The functoriality condition discussed in Definition 4.6 is satisfied by the universal property of exponentials. These conditions are satisfied, for instance, by models of cubical type theory. They are also satisfied by the examples in Section 2.3. In what follows, we discuss functorial Π-types in the these examples.

Example 4.10 (Examples 2.8 and 3.13 ctd.). As explained in Example 3.13, in this example, a type A in context Γ is interpreted as a split isofibration $[\![A]\!] : [\![\Gamma.A]\!] \to [\![\Gamma]\!]$. For each groupoid Γ and split isofibrations $A : \Gamma.A \to \Gamma$ and $B : \Gamma.A.B \to \Gamma.A$, we have a split isofibration $\Pi(A,B) : \Gamma.\Pi(A,B) \to \Gamma$, where for each $x \in \Gamma$, the set of objects in the fiber over x is $\Pi(x' : A_x), B_{x'}$, i.e. the elements in the fibers are dependent functions. The functor app is $\mathrm{dom}(\alpha)$, where $\alpha : \Pi(A,B)[\pi_A] \to B$ is the morphism of split fibrations in the fiber over $\Gamma.A$ that maps $(x \in \Gamma, a \in A_x, f \in \Pi(x' : A_x), B_{x'})$ to $(x \in \Gamma, a \in A_x, f(a) \in B_a)$ for each x, a and f. For a section $b : \Gamma.A \to \Gamma.A.B$ of the fibration B over $\Gamma.A$ that maps $(x \in \Gamma, a \in A_x)$ to $(x \in \Gamma, a \in A_x, b(a) \in B_a)$ for each x and a, λb is a section of the fibration $\Pi(A,B)$ over Γ that maps $x \in \Gamma$ to $(x \in \Gamma, \lambda(x' : A_x).b(x'))$.

Now for the functoriality condition, let $f : A' \to A$ be a morphism of split fibrations in the fiber over Γ that maps $(x \in \Gamma, a' \in A'_x)$ to $(x \in \Gamma, f(a') \in A_x)$ for each x and a' and let $g : B[f] \to B'$ be a morphism of split fibrations over $\Gamma.A'$ that maps $(x \in \Gamma, a' \in A'_x, b \in B_{f(a')})$ to $(x \in \Gamma, a' \in A'_x, g(b)B'_{a'})$ for each x, a' and b. The morphism $\Pi(f,g) : \Pi(A,B) \to \Pi(A',B')$ maps $(x \in \Gamma, \in \Pi(x' : A_x), B_{x'})$ to $(x \in \Gamma, \Pi(x' : A'_x), ghf(x'))$. It is easy to see that this assignation satisfies the functoriality conditions discussed in Definition 4.6.

Example 4.11 (Examples 2.10 and 3.14 ctd.). As explained in Example 3.14, in this example, a type A in context Γ is interpreted as an H-valued predicate $[\![A]\!] : [\![\Gamma]\!] \to H$ and $\Gamma.A$ is interpreted as the comprehension. This example has Π-types. For each set Γ and predicates $A : \Gamma \to H$ and $B : \Gamma.A \to H$, where $\Gamma.A$ is the comprehension of A, the predicate $\Pi(A,B) : \Gamma \to H$ is defined to be $B(x)$ if $\top \leq A(x)$ and $\top$ otherwise. We have $\Gamma.A.\Pi(A,B)[\pi_A] = \Gamma.A.B = \{x \in \Gamma | \top \leq A(x) \wedge \top \leq B(x)\}$ and the function app : $\Gamma.A.\Pi(A,B)[\pi_A] \to \Gamma.A.B$ is identity.

Now for the functoriality condition, recall that in this example the hom-sets in the fibers have at most one element. Let Γ be a set, and $A, A' : \Gamma \to H$ be two H-valued predicates. We have a morphism in the fiber over Γ of the form $A' \to A$. This means that for all $x \in \Gamma$ we have $A'(x) \leq A(x)$. Hence, $\Gamma.A' \subseteq \Gamma.A$. We have a morphism in the fiber over $\Gamma.A'$ of the form $B(x) \to B'(x)$, which means that for all $x \in \Gamma$ such that $\top \leq A'(x)$ we have $B(x) \leq B'(x)$. Now we verify that there is a morphism in the fiber over Γ of the form $\Pi(A,B) \to \Pi(A',B')$. For this we need that for each $x \in \Gamma$, $\Pi(A,B)(x) \leq \Pi(A',B')$. If $\top \leq A'(x)$, then we have $\top \leq A(x)$ and $\Pi(A,B)(x) = B(x)$. Since $\top \leq A'(x)$, we have $\Pi(A,B)(x) = B(x) \leq B'(x) = \Pi(A',B')(x)$. If $A'(x) < \top$, then $\Pi(A',B')(x) = \top$ and we have $\Pi(A,B)(x) \leq \Pi(A',B')$ trivially.

Note that in Example 4.11, Π-types do not give universal quantification the way that one would expect in first-order predicate logic. The derivation rules for universal quantification are expressed using proofs, whereas the derivation rules for Π-types in this example are expressed using terms. In this example, proofs are represented by morphisms in the fiber categories, whereas terms are represented as sections of projections, which explains the discrepancy.

Definition 4.12. We define the **extension of CCTT by functorial Π-types** to consist of the rules in Fig. 1.

Theorem 4.13 (Soundness of Rules for Π-types). *Any comprehension category with functorial Π-types models the rules of CCTT and the rules for functorial Π-types in Fig. 1.*

We now see how elimination and computation rules for Π-types similar to those in MLTT can be derived from the rules in Fig. 1.

$$\frac{\Gamma \vdash A\ \mathsf{type} \qquad \Gamma.A \vdash B\ \mathsf{type}}{\Gamma \vdash \Pi(A,B)\ \mathsf{type}}\ \text{pi-form} \qquad \frac{\Gamma.A \vdash b : B}{\Gamma \vdash \lambda b : \Pi(A,B)}\ \text{pi-intro}$$

$$\frac{\Gamma \vdash A\ \mathsf{type} \qquad \Gamma.A \vdash B\ \mathsf{type}}{\Gamma.A.\Pi(A,B)[\pi_A] \vdash \mathrm{app}_{\Pi(A,B)} : \Gamma.A.B}\ \text{pi-elim}$$

$$\Gamma.A.\Pi(A,B)[\pi_A] \vdash \pi_B \circ \mathrm{app}_{\Pi(A,B)} \equiv \pi_{\Pi(A,B)[\pi_A]} : \Gamma.A$$

$$\frac{\Gamma.A \vdash b : B}{\Gamma.A \vdash \mathrm{app}_{\Pi(A,B)} \circ p_2(\lambda b \circ \pi_A) \equiv b : \Gamma.A.B}\ \text{pi-beta} \qquad \frac{\Gamma \vdash f : \Pi(A,B)}{\Gamma \vdash \lambda(\mathrm{app}_{\Pi(A,B)} \circ p_2(f \circ \pi_A)) \equiv f : \Gamma.\Pi(A,B)}\ \text{pi-eta}$$

$$\frac{\Delta \vdash A\ \mathsf{type} \qquad \Delta.A \vdash B\ \mathsf{type} \qquad \Gamma \vdash s : \Delta}{\Gamma \mid \Pi(A[s], B[s.A]) \mathrel{\tilde{\vdash}} i_{\Pi(A,B),s} : \Pi(A,B)[s]}\ \text{pi-sub}$$

$$\frac{\Gamma \vdash s : \Delta \qquad \Gamma \vdash b : \Pi(A,B)}{\Gamma \vdash \lambda_{\Pi(A,B)}(b) \circ s \equiv s.\Pi(A,B) \circ \Gamma.i_{\Pi(A,B),s} \circ \lambda_{\Pi(A[s],B[s.A])}(p_2(b \circ s.A)) : \Delta.\Pi(A,B)}\ \text{sub-lam}$$

$$\frac{\Gamma.A \vdash B\ \mathsf{type} \qquad \Gamma.A' \vdash B'\ \mathsf{type} \qquad \Gamma \mid A' \vdash f : A \qquad \Gamma.A' \mid B[\Gamma.f] \vdash g : B'}{\Gamma \mid \Pi(A,B) \vdash \Pi(f,g) : \Pi(A',B')}\ \text{subt-pi}$$

$$\Gamma.\Pi(A,B) \vdash \Gamma.\Pi(f,g) \equiv \Gamma.A'.g \circ \lambda(p_2(\Gamma.A'.g \circ (\pi_{\Pi(A,B)[\pi_{A'}]}.B[\Gamma.f] \circ p_2(\mathrm{app}_{\Pi(A,B)} \circ (\Gamma.f).\Pi(A,B)[\pi_A] \circ i^{\mathrm{comp}}_{\Pi(A,B)[\pi_A],\pi_A,\Gamma.f} \circ i^{\mathrm{sub}}_{\Pi(A,B)[\pi_A],\pi_{A'},\pi_A \circ \Gamma.f})))) : \Gamma.\Pi(A',B')$$

$$\frac{\Gamma \vdash A\ \mathsf{type} \qquad \Gamma.A \vdash B\ \mathsf{type}}{\Gamma \mid \Pi(A,B) \vdash \Pi(1_A, i^{\mathrm{id}}_B) \equiv 1_{\Pi(A,B)} : \Pi(A,B)}\ \text{subt-pi-id}$$

$$\frac{\begin{array}{c}\Gamma.A \vdash B\ \mathsf{type} \qquad \Gamma.A' \vdash B'\ \mathsf{type} \qquad \Gamma \mid A' \vdash f : A \\ \Gamma.A \mid B[\Gamma.f] \vdash g : B' \qquad \Gamma \mid A'' \vdash f' : A' \qquad \Gamma.A' \mid B'[\Gamma.f'] \vdash g' : B''\end{array}}{\Gamma \mid \Pi(A,B) \vdash \Pi(f \circ f', g' \circ g[\Gamma.f']) = \Pi(f',g') \circ \Pi(f,g) : \Pi(A'',B'')}\ \text{subt-pi-comp}$$

$$\frac{\Gamma \vdash A\ \mathsf{type} \qquad \Gamma.A \vdash B\ \mathsf{type}}{\Gamma \mid \Pi(A[1_\Gamma], B[1.A]) \vdash i^{\mathrm{id}}_{\Pi(A,B)} \circ i_{\Pi(A,B),1_\Gamma} \equiv \Pi(i^{\mathrm{id}^{-1}}_A, i^{\mathrm{id}}_B \circ i^{\mathrm{sub}}_{B,1_\Gamma.A \circ \Gamma.i^{\mathrm{id}^{-1}}_A,1_{\Gamma.A}} \circ i^{\mathrm{comp}^{-1}}_{B,1_\Gamma.A,\Gamma.i^{\mathrm{id}^{-1}}_A}) : \Pi(A,B)}\ \text{pi-sub-id}$$

$$\frac{\Theta \vdash A\ \mathsf{type} \qquad \Theta.A \vdash B\ \mathsf{type} \qquad \Gamma \vdash s' : \Delta \qquad \Delta \vdash s : \Theta}{\Gamma \mid \Pi(A[s \circ s'], B[(s \circ s').A]) \vdash i^{\mathrm{comp}}_{\Pi(A,B),s,s'} \circ i_{\Pi(A,B),s \circ s'} \equiv (i_{\Pi(A,B),s})[s'] \circ i_{\Pi(A[s'],B[s'.A])} \circ \Pi(i^{\mathrm{comp}^{-1}}_{A,s,s'}, i^{\mathrm{comp}}_{B,s.A,s'.A[s]} \circ i^{\mathrm{sub}}_{B,(s \circ s').A \circ \Gamma.i^{\mathrm{comp}^{-1}}_{A,s,s'},s.A \circ s'.A[s]} \circ i^{\mathrm{comp}^{-1}}_{B,(s \circ s').A,\Gamma.i^{\mathrm{comp}^{-1}}_{A,s,s'}}) : \Pi(A,B)[s][s']}\ \text{pi-sub-comp}$$

Fig. 1. Rules for functorial Π-types. Rules pi-sub, sub-lam,p-sub-id and p-sub-comp use the notation introduced in Proposition 3.6.

Proposition 4.14. *From the rules of CCTT and the rules in Fig. 1, we can derive the following rules.*

$$\frac{\Gamma \vdash f : \Pi(A,B) \qquad \Gamma \vdash a : A}{\Gamma \vdash \mathrm{app}_{\Pi(A,B)}(f,a) : B[a]} \qquad \frac{\Gamma.A \vdash b : B \qquad \Gamma \vdash a : A}{\Gamma \vdash \mathrm{app}_{\Pi(A,B)}(\lambda b, a) \equiv p_2(b \circ a) : \Gamma.B[a]}$$

Proof. The context morphism $\mathrm{app}_{\Pi(A,B)}(f,a)$ is $p_2(\mathrm{app}_{\Pi(A,B)} \circ p_2(f \circ \pi_A) \circ a)$. □

Related Work 4.15 (Coraglia and Emmenegger [19]). Rule subt-pi corresponds to the rules in Proposition 20 of Coraglia and Emmenegger's paper [19]. Under the subtyping point of view, this rule states that $A' \leq_f A$ and $B[\Gamma.f] \leq_g B'$ give $\Pi(A,B) \leq_{\Pi(f,g)} \Pi(A',B')$. They do not however, explicitly present rules corresponding to Rules subt-pi-id and subt-pi-comp, which state that $\Pi(-,-)$ preserves identity and composition of subtyping witnesses, respectively. Note that Coraglia and Emmenegger write the rules in [19, Proposition 20] as if the fibrations involved were split, for simplicity. In our case, this equates to removing the i^{comp}, i^{id} and i^{sub} terms from the rules.

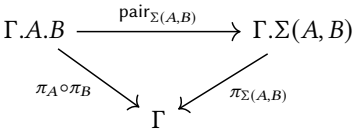
4.2 Functorial Σ-Types

In this section, we define semantic structure for Σ-types in non-full comprehension categories. We then discuss the necessary functoriality conditions that allow us to use type morphisms to interpret subtyping. We extend CCTT with functorial Σ-types and prove soundness by giving an

interpretation of the rules in any comprehension category with functorial Σ-types. We also discuss how CCTT with functorial Σ-types supports subtyping.

Definition 4.16. Let $(\mathcal{C}, \mathcal{T}, p, \chi)$ be a comprehension category. Given $\Gamma \in \mathcal{C}$, $A \in \mathcal{T}_\Gamma$ and $B \in \mathcal{T}_{\Gamma.A}$, **a (strong) dependent sum** for Γ, A and B consists of:

(1) an object $\Sigma(A, B) \in \mathcal{T}_\Gamma$;
(2) an isomorphism $\mathrm{pair}_{\Sigma(A,B)} : \Gamma.A.B \to \Gamma.\Sigma(A, B)$ in $\mathcal{C}$ with inverse $\mathrm{proj}_{\Sigma(A,B)}$ making the following diagram commute.



Just as in the case of Π-types, Definition 4.16 is closely connected to the syntactic representation of Σ-types. The object $\Sigma(A, B)$ in $\mathcal{T}_\Gamma$ corresponds to the dependent sum of types A and B in context Γ. The morphism $\mathrm{pair}_{\Sigma(A,B)} : \Gamma.A.B \to \Gamma.\Sigma(A, B)$ in $\mathcal{C}$ corresponds to pairing of a term of type A with a term of type B that depends on A. The morphism $\mathrm{proj}_{\Sigma(A,B)} : \Gamma.\Sigma(A, B) \to \Gamma.A.B$ corresponds to the second projection and $\pi_B \circ \mathrm{proj}_{\Sigma(A,B)}$ gives the first projection. $\mathrm{proj}_{\Sigma(A,B)}$ being an inverse of $\mathrm{pair}_{\Sigma(A,B)}$ gives the β- and η-rules.

Related Work 4.17 (Jacobs [34]). Jacobs defines dependent sums in a (full) comprehension category as left adjoints to certain reindexing functors — the weakening functors of the form π_A^* [34, Definition 3.10 (i)]. In a full comprehension category, this definition is equivalent to Definition 4.16.

As explained in Related Work 4.3, we do not assume fullness; hence, we make a distinction between the structure added to $\mathcal{C}$ and the structure added to $\mathcal{T}$. In particular, we are interested in having the structure of pair and proj on $\mathcal{C}$ and the structure related to subtyping on $\mathcal{T}$. We use Definition 4.16, where the structure of pair and proj is on the category $\mathcal{C}$. In Jacobs' definition this structure is on the category $\mathcal{T}$.

Related Work 4.18 (Lumsdaine and Warren [40]). Lumsdaine and Warren define Σ-types with an induction rule [40, Definition 3.4.4.1]. In contrast, we describe Σ-types via projections, as this description is simpler.

To be able to use type morphisms to interpret subtyping, we need to add certain functoriality conditions which formalize the intuition that Σ-types act covariantly on both arguments in the context of subtyping. In particular, given subtyping relations $A \leq_f A'$ and $B \leq_g B'[\Gamma.f]$ we have $\Sigma(A, B) \leq_{\Sigma(f,g)} \Sigma(A', B')$, since Σ acts covariantly on both the first and the second arguments. The coercion function for $\Sigma(A, B) \leq_{\Sigma(f,g)} \Sigma(A', B')$ takes a dependent pair $(a, b) : \Sigma(A, B)$ to the coerced dependent pair $(a', b') : \Pi(A', B')$ as follows. The term $a' : A'$ is obtained by coercing a to A' using f, and the term $b' : B'$ is obtained by coercing b to $B'[\Gamma.f]$ using g.

Now we define what it means for a comprehension category to have functorial Σ-types. For this, we add the structure defined in Definition 4.16 to a comprehension category, postulate a Beck-Chevalley condition, i.e. that this structure is preserved under substitution, and postulate suitable functoriality conditions.

Definition 4.19. A comprehension category $(\mathcal{C}, \mathcal{T}, p, \chi)$ **has functorial Σ-types** if it is equipped with a function giving $\Sigma(A, B)$, $\mathrm{pair}_{\Sigma(A,B)}$ and $\mathrm{proj}_{\Sigma(A,B)}$ for each suitable Γ, A, B such that:

(1) for each $s : \Gamma \to \Delta$ in $\mathcal{C}$ we have $i_{\Sigma(A,B),s} : \Sigma(A[s], B[s.A]) \cong \Sigma(A, B)[s]$ in $\mathcal{T}_\Gamma$;

(2) for each $A \in \mathcal{T}_\Delta$, $B \in \mathcal{T}_{\Delta.A}$ the following diagram commutes

$$\begin{array}{ccc} \Gamma.\Sigma(A[s], B[s.A]) & \xrightarrow{s.\Sigma(s_A, s_B)} & \Delta.\Sigma(A, B) \\ {\scriptstyle \mathsf{pair}_{\Sigma(A[s],B[s.A])}}\uparrow & & \uparrow{\scriptstyle \mathsf{pair}_{\Sigma(A,B)}} \\ \Gamma.A[s].B[s.A] & \xrightarrow[s.A.B]{} & \Delta.A.B \end{array}$$

where $s.\Sigma(s_A, s_B) \coloneqq s.\Sigma(A, B) \circ \Gamma.i_{\Sigma(A,B),s}$;

(3) the comprehension category is equipped with a function giving for each $f : A \to A'$ in $\mathcal{T}_\Gamma$ and $g : B \to B'$; $[\Gamma.f]$ in $\mathcal{T}_{\Gamma.A}$, a morphism $\Sigma(f, g) : \Sigma(A, B) \to \Sigma(A', B')$ in $\mathcal{T}_\Gamma$;

(4) $\Gamma.\Sigma(f, g)$ is the following composite,

$$\Gamma.\Sigma(A, B) \xrightarrow{\mathsf{proj}_{\Sigma(A,B)}} \Gamma.A.B \xrightarrow{\Gamma.A.g} \Gamma.A.B'[\Gamma.f] \xrightarrow{\Gamma.f.B'} \Gamma.A'.B' \xrightarrow{\mathsf{pair}_{\Sigma(A',B')}} \Gamma.\Sigma(A', B')$$

(see [45, Appendix C] for more detail);

(5) $\Sigma(-,-)$ preserves identities, i.e. we have $\Sigma(1_A, i_B^{\mathsf{id}^{-1}}) = 1_{\Sigma(A,B)}$ for each suitable A and B, where $i_B^{\mathsf{id}} : B[1_{\Gamma.A}] \cong B$;

(6) $\Sigma(-,-)$ preserves composition, i.e. we have $\Sigma(f' \circ f, g'[\Gamma.f] \circ g) = \Sigma(f', g') \circ \Sigma(f, g)$ for each suitable f' and g';

(7) $i_{\Sigma(A,B),-}$ is functorial in that it preserves i^{iso} and i^{comp} (see [45, Appendix D] for more detail).

Items 1 and 2 of Definition 4.19 state that Σ and pair are preserved under substitution respectively. Consequently, proj is also preserves under substitution. Items 3 to 7 give the functoriality conditions which formalize the variance of Σ-types on the arguments for subtyping. Item 4 expresses compatibility of the type morphism structure on the category $\mathcal{T}$ with the type former structure on the category C.

The following proposition states that Definition 4.19 is compatible with Jacobs' definition of comprehension categories with sums.

Proposition 4.20 (Relation to Jacobs [35]). *Every **full** comprehension category with sums in the sense of Jacobs [34] has functorial Σ-types in the sense of Definition 4.19.*

Related Work 4.21 (Coraglia and Emmenegger [19]). Coraglia and Emmenegger [19] define Σ-types for generalized categories with families, a structure equivalent to comprehension categories [18]. This definition is similar to Definition 4.6 regarding variance on the arguments.

Related Work 4.22 (Gambino and Larrea [24]). Gambino and Larrea [24] also define Σ-types for comprehension categories, but in a different way than we do. As with Π-types, we require Σ-types to be functorial, while they do not, and we phrase stability using isomorphisms instead of Cartesian morphisms. A more notable difference lies in the fact that Gambino and Larrea [24] express Σ-types by giving the usual introduction and elimination rule (i.e., Σ-induction), whereas we express Σ-types equivalently via pairing and projections.

We now discuss examples of comprehension categories with functorial Σ-types, including those that arise from AWFSs (see Section 2.3). Whereas for Π-types we need to require additional assumptions, every AWFS supports Σ-types [24, Proposition 4.3]. This is because the right class of maps is closed under composition. Since Σ-types are interpreted using composition, the functoriality condition of Definition 4.19 is also satisfied. This means that in particular, the examples in Section 2.3 have functorial Σ-types. In what follows, we discuss functorial Σ-types in the these examples.

Example 4.23 (Examples 2.8 and 3.13 ctd.). As explained in Example 3.13, in this example, a type A in context Γ is interpreted as a split isofibration $[\![A]\!] : [\![\Gamma.A]\!] \to [\![\Gamma]\!]$. This example has

Σ-types. For each groupoid Γ and split isofibrations $A : \Gamma.A \to \Gamma$ and $B : \Gamma.A.B \to \Gamma.A$, we have a split isofibration $\Sigma(A,B) : \Gamma.\Sigma(A,B) \to \Gamma$, where for each $x \in \Gamma$, the set of objects in the fiber over x is $\Sigma(x' : A_x), B_{x'}$, i.e. the elements in the fibers are dependent pairs. The functor pair : $\Gamma.A.B \to \Gamma.\Sigma(A,B)$ is $\mathrm{dom}(\alpha)$, where $\alpha : (A \circ B) \to \Sigma(A,B)$ is the morphism of split fibrations over Γ that maps $(x \in \Gamma, a \in A_x, b \in B_a)$ to $(x \in \Gamma, (a,b) \in \Sigma(x' : A_x), B_{x'})$.

Now for the functoriality condition, let $f : A \to A'$ be a morphism of split fibrations in the fiber over Γ that maps $(x \in \Gamma, a \in A_x)$ to $(x \in \Gamma, f(a) \in A'_x)$ and let $g : B \to B'[f]$ be a morphism of split fibrations over $\Gamma.A$ that maps $(x \in \Gamma, a \in A_x, b \in B_a)$ to $(x \in \Gamma, f(a) \in A'_x, g(b) : B'_{f(a)})$. The morphism $\Sigma(f,g) : \Sigma(A,B) \to \Sigma(A',B')$ maps $(x \in \Gamma, (h_1,h_2) \in \Sigma(x' : A_x), B_{x'})$ to $(x \in \Gamma, (f(h_1), g(h_2)) \in \Sigma(x' : A'_x), B'_{x'})$. It is easy to see that this assignation satisfies the properties from Definition 4.19.

Example 4.24 (Examples 2.10 and 3.14 ctd.). As explained in Example 3.14, in this example, a type A in context Γ is interpreted as an H-valued predicate $[\![A]\!] : [\![\Gamma]\!] \to H$ and $\Gamma.A$ is interpreted as the comprehension. This example has Σ-types. For each set Γ and predicates $A : \Gamma \to H$ and $B : \Gamma.A \to H$, where $\Gamma.A$ is the comprehension of A, the predicate $\Sigma(A,B) : \Gamma \to H$ is defined to be $B(x)$ if $A(x) = \top$ and $\bot$ otherwise. We have $\Gamma.A.B = \Gamma.\Sigma(A,B) = \{x \in \Gamma | \top \leq A(x) \wedge \top \leq B(x)\}$ and the function pair : $\Gamma.A.B \to \Gamma.\Sigma(A,B)$ is identity.

Now for the functoriality condition, recall that in this example the hom-sets in the fibers have at most one element. Let Γ be a set, and $A, A' : \Gamma \to H$ be two H-valued predicates. We have a morphism in the fiber over Γ of the form $A \to A'$. This means that for all $x \in \Gamma$ we have $A(x) \leq A'(x)$. Hence, $\Gamma.A \subseteq \Gamma.A'$. We have a morphism in the fiber over $\Gamma.A$ of the form $B(x) \to B'(x)$, which means that for all $x \in \Gamma$ such that $\top \leq A(x)$ we have $B(x) \leq B'(x)$. Now we verify that there is a morphism in the fiber over Γ of the form $\Sigma(A,B) \to \Sigma(A',B')$. For this we need that for each $x \in \Gamma$, $\Sigma(A,B)(x) \leq \Sigma(A',B')$. If $\top \leq A(x)$, then we have $\top \leq A'(x)$. Hence, $\Sigma(A,B)(x) = B(x) \leq B'(x) = \Sigma(A',B')$. If $A(x) < \top$, then $\Sigma(A,B)(x) = \bot$ and we have $\Sigma(A,B)(x) \leq \Sigma(A',B')$ trivially.

Recall that Π-types in Example 4.11 do not correspond to universal quantification. For the same reason, Σ-types in Example 4.24 do not correspond to existential quantification.

Definition 4.25. We define the **extension of CCTT by functorial Σ-types** to consist of the rules in Fig. 2.

Theorem 4.26 (Soundness of Rules for Σ-types). *Any comprehension category with functorial Σ-types models the rules of CCTT and the rules for functorial Σ-types in Fig. 2.*

In the following proposition, we see that we can derive first and second projection rules similar to those of Martin-Löf type theory from the rules in Fig. 2.

Proposition 4.27. *From the rules of CCTT and the rules in Fig. 2, we can derive the following rules.*

$$\frac{\Gamma \vdash p : \Sigma(A,B)}{\Gamma \vdash \mathsf{proj}_1 p : A} \qquad \Gamma \vdash \mathsf{proj}_2 p : B[\mathsf{proj}_1 p]$$

Proof. The context morphism $\mathsf{proj}_1 p$ is $\pi_B \circ \mathsf{proj}_{\Sigma(A,B)} \circ p$ and $\mathsf{proj}_2 p$ is $p_2(\mathsf{proj}_{\Sigma(A,B)} \circ p)$. □

Related Work 4.28 (Coraglia and Emmenegger [19]). Rule subt-sigma corresponds to the rules in Proposition 21 of Coraglia and Emmenegger's paper [19]. Under the subtyping point of view, this rule states that $A \leq_f A'$ and $B \leq_g B'[\Gamma.f]$ give $\Sigma(A,B) \leq_{\Sigma(f,g)} \Sigma(A',B')$. They do not however, explicitly present rules corresponding to Rules subt-sigma-id and subt-sigma-comp, which state that $\Sigma(-,-)$ preserves identity and composition subtyping witnesses, respectively. Note that Coraglia and Emmenegger write the rules in [19, Proposition 21] as if the fibrations involved were split, for simplicity. In our case, this equates to removing the i^{comp} and i^{id} from the rules.

$$\frac{\Gamma \vdash A\ \mathsf{type} \qquad \Gamma.A \vdash B\ \mathsf{type}}{\Gamma \vdash \Sigma(A,B)\ \mathsf{type}}\ \text{sigma-form} \qquad \frac{\Gamma \vdash A\ \mathsf{type} \qquad \Gamma.A \vdash B\ \mathsf{type}}{\Gamma.A.B \vdash \mathsf{pair}_{\Sigma(A,B)} : \Gamma.\Sigma(A,B)}\ \text{sigma-intro}$$

$$\frac{\Gamma \vdash A\ \mathsf{type} \qquad \Gamma.A \vdash B\ \mathsf{type}}{\Gamma.\Sigma(A,B) \vdash \mathsf{proj}_{\Sigma(A,B)} : \Gamma.A.B}\ \text{sigma-elim} \qquad \frac{\Gamma \vdash A\ \mathsf{type} \qquad \Gamma.A \vdash B\ \mathsf{type}}{\Gamma.A.B \vdash \mathsf{proj}_{\Sigma(A,B)} \circ \mathsf{pair}_{\Sigma(A,B)} \equiv 1_{\Gamma.A.B} : \Gamma.A.B}\ \text{sigma-beta}$$

$$\Gamma \vdash \pi_{\Sigma(A,B)} \circ \mathsf{pair}_{\Sigma(A,B)} \equiv \pi_A \circ \pi_B : \Gamma$$

$$\frac{\Gamma \vdash A\ \mathsf{type} \qquad \Gamma.A \vdash B\ \mathsf{type}}{\Gamma.\Sigma(A,B) \vdash \mathsf{pair}_{\Sigma(A,B)} \circ \mathsf{proj}_{\Sigma(A,B)} \equiv 1_{\Gamma.\Sigma(A,B)} : \Gamma.\Sigma(A,B)}\ \text{sigma-eta}$$

$$\frac{\Delta \vdash A\ \mathsf{type} \qquad \Delta.A \vdash B\ \mathsf{type} \qquad \Gamma \vdash s : \Delta}{\Gamma \mid \Sigma(A[s], B[s.A]) \ \tilde{\vdash}\ i_{\Sigma(A,B),s} : \Sigma(A,B)[s]}\ \text{sigma-sub}$$

$$\frac{\Delta \vdash A\ \mathsf{type} \qquad \Delta.A \vdash B\ \mathsf{type} \qquad \Gamma \vdash s : \Delta}{\Gamma.A[s].B[s.A] \vdash s.\Sigma(A,B) \circ \Gamma.i_{\Sigma(A,B),s} \circ \mathsf{pair}_{\Sigma(A[s],B[s.A])} \equiv \mathsf{pair}_{\Sigma(A,B)} \circ s.A.B : \Delta.\Sigma(A,B)}\ \text{sub-pair}$$

$$\frac{\Gamma.A \vdash B\ \mathsf{type} \qquad \Gamma.A' \vdash B'\ \mathsf{type} \qquad \Gamma \mid A \vdash f : A' \qquad \Gamma.A \mid B \vdash g : B'[\Gamma.f]}{\Gamma \mid \Sigma(A,B) \vdash \Sigma(f,g) : \Sigma(A',B')}\ \text{subt-sigma}$$

$$\Gamma.\Sigma(A,B) \vdash \Gamma.\Sigma(f,g) \equiv \mathsf{pair}_{\Sigma(A',B')} \circ (\Gamma.f).B' \circ \Gamma.A.g \circ \mathsf{proj}_{\Sigma(A,B)} : \Gamma.\Sigma(A',B')$$

$$\frac{\Gamma \vdash A\ \mathsf{type} \qquad \Gamma.A \vdash B\ \mathsf{type}}{\Gamma \mid \Sigma(A,B) \vdash \Sigma(1_A, i_B^{\mathsf{id}^{-1}}) \equiv 1_{\Sigma(A,B)} : \Sigma(A,B)}\ \text{subt-sigma-id}$$

$$\frac{\begin{array}{c}\Gamma.A \vdash B\ \mathsf{type} \qquad \Gamma.A' \vdash B'\ \mathsf{type} \qquad \Gamma \mid A \vdash f : A' \\ \Gamma.A \mid B \vdash g : B'[\Gamma.f] \qquad \Gamma \mid A' \vdash f' : A'' \qquad \Gamma.A' \mid B' \vdash g' : B''[\Gamma.f']\end{array}}{\Gamma \mid \Sigma(A,B) \vdash \Sigma(f' \circ f, g'[\Gamma.f] \circ g) \equiv \Sigma(f',g') \circ \Sigma(f,g) : \Sigma(A'',B'')}\ \text{subt-sigma-comp}$$

$$\frac{\Gamma \vdash A\ \mathsf{type} \qquad \Gamma.A \vdash B\ \mathsf{type}}{\Gamma \mid \Sigma(A[1_\Gamma], B[1.A]) \vdash i^{\mathsf{id}}_{\Sigma(A,B)} \circ i_{\Sigma(A,B),1_\Gamma} \equiv \Sigma(i^{\mathsf{id}}_A, i^{\mathsf{id}}_B \circ i^{\mathsf{sub}}_{B,1_\Gamma.A \circ \Gamma.i^{\mathsf{id}}_A, 1_{\Gamma.A}} \circ i^{\mathsf{comp}^{-1}}_{B,1_\Gamma.A,\Gamma.i^{\mathsf{id}}_A}) : \Sigma(A,B)}\ \text{sigma-sub-id}$$

$$\frac{\Theta \vdash A\ \mathsf{type} \qquad \Theta.A \vdash B\ \mathsf{type} \qquad \Gamma \vdash s' : \Delta \qquad \Delta \vdash s : \Theta}{\begin{array}{l}\Gamma \mid \Sigma(A[s \circ s'], B[(s \circ s').A]) \vdash i^{\mathsf{comp}}_{\Sigma(A,B),s,s'} \circ i_{\Sigma(A,B),s \circ s'} \equiv (i_{\Sigma(A,B),s})[s'] \circ i_{\Sigma(A[s'],B[s'.A])} \circ \\ \Sigma(i^{\mathsf{comp}}_{A,s,s'}, i^{\mathsf{comp}}_{B,s.A,s'.A[s]} \circ i^{\mathsf{sub}}_{B,(s\circ s').A \circ \Gamma.i^{\mathsf{comp}}_{A,s,s'}, s.A \circ s'.A[s]} \circ i^{\mathsf{comp}^{-1}}_{B,(s\circ s').A,\Gamma.i^{\mathsf{comp}}_{A,s,s'}}) : \Sigma(A,B)[s][s']\end{array}}\ \text{sigma-sub-comp}$$

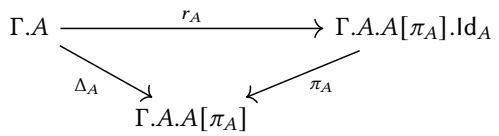
Fig. 2. Rules for functorial Σ-types. Rules sigma-sub, sub-pair,sigma-sub-id and sigma-sub-comp use the notation introduced in Proposition 3.6. For example, in Rule sigma-sub, $s.A$ is $(s \circ \pi_{A[s]}, \Gamma.(i^{\mathsf{comp}}_{A,s,\pi_{A[s]}}[\pi_{A[s]}]) \circ p_2(1_{\Gamma.A[s]}))$.

4.3 Functorial Id-Types

In this section, we define semantic structure for Id-types in non-full comprehension categories. We then discuss the necessary functoriality conditions that allow us to use type morphisms to interpret subtyping. We extend CCTT with functorial Id-types and prove soundness by giving an interpretation of the rules in any comprehension category with functorial Id-types. We also discuss how CCTT with functorial Id-types supports subtyping.

Definition 4.29 ([40, Definition 2.3.1]). Let $(\mathcal{C}, \mathcal{T}, p, \chi)$ be a comprehension category. Given $\Gamma \in \mathcal{C}$, $A \in \mathcal{T}_\Gamma$, **an identity type** for Γ and A consists of:

(1) an object $\mathsf{Id}_A \in \mathcal{T}_{\Gamma.A.A[\pi_A]}$;
(2) a morphism $r_A : \Gamma.A \to \Gamma.A.A[\pi_A].\mathsf{Id}_A$ in $\mathcal{C}$ making the following diagram commute,



where Δ_A is the diagonal morphism of the form $\Gamma.A \to \Gamma.A.A[\pi_A]$;
(3) for each $C \in \mathcal{T}_{\Gamma.A.A[\pi_A].\mathsf{Id}_A}$ and $d : \Gamma.A \to \Gamma.A.A[\pi_A].\mathsf{Id}_A.C$ making the outer square commute, a section $j_{A,C,d} : \Gamma.A.A[\pi_A].\mathsf{Id}_A \to \Gamma.A.A[\pi_A].\mathsf{Id}_A.C$ of π_C making the following two triangles

commute.

$$\begin{array}{ccc} \Gamma.A & \xrightarrow{d} & \Gamma.A.A[\pi_A].\mathsf{Id}_A.C \\ {\scriptstyle r_A}\downarrow & \nearrow_{j_{A,C,d}} & \downarrow{\scriptstyle \pi_C} \\ \Gamma.A.A[\pi_A].\mathsf{Id}_A & = & \Gamma.A.A[\pi_A].\mathsf{Id}_A \end{array}$$

Similar to the case of Π- and Σ-types, we tie Definition 4.29 to identity types in syntax. The object Id_A in $\mathcal{T}_\Gamma$ corresponds the identity type for terms of type A in context $\Gamma.A.A$. The morphism $r : \Gamma.A \to \Gamma.A.A[\pi_A].\mathsf{Id}_A$ in $\mathcal{C}$ gives the reflexivity proof. The morphism $j_{A,B,:}\Gamma.A.A[\pi_A].\mathsf{Id}_A \to \Gamma.A.A[\pi_A].\mathsf{Id}_A.C$ in $\mathcal{C}$ gives the elimination rule for identity types. This rule states that to construct a term of type C in the context $\Gamma.A.A.\mathsf{Id}_A$, it suffices to provide a term of type C when the second and third variables are replaced by the first one and the reflexivity proof, in the context $\Gamma.A$.

Related Work 4.30 (Jacobs [35]). Jacobs defines identity types in a (full) comprehension category as left adjoints to certain reindexing functors — the contraction functors of the form Δ_A^*, where $\Delta_A : \Gamma.A \to \Gamma.A.A$ is a diagonal morphism [35, Definition 10.5.1]. This definition gives an extensional identity type, whereas Definition 4.29 gives an intensional identity type.

To be able to use type morphisms to interpret subtyping, we need to add certain functoriality conditions which formalize the intuition of how Id-types interact with subtyping. In particular, given a subtyping relation $A \leq_t B$ we have $\mathsf{Id}_A \leq_{\mathsf{Id}_t} \mathsf{Id}_B[\Gamma.t.t]$, since Id-types preserve subtyping. In the semantics, this means that for each morphism $t : A \to B$ in $\mathcal{T}_\Gamma$, we have a morphism $t' : \mathsf{Id}_A \to \mathsf{Id}_B[\Gamma.t.t]$ in $\mathcal{T}_{\Gamma.A.A[\pi_A]}$, which is equivalent to having a morphism $\mathsf{Id}_t : \mathsf{Id}_A \to \mathsf{Id}_B$ in $\mathcal{T}$ with $p(\mathsf{Id}_t) = \Gamma.t.t$.

Now we define what it means for a comprehension category to have functorial Id-types. For this, we add the structure defined in Definition 4.29 to a comprehension category, postulate a Beck-Chevalley condition, i.e. that this structure is preserved under substitution, and postulate suitable functoriality conditions.

Definition 4.31. A comprehension category $(\mathcal{C}, \mathcal{T}, p, \chi)$ **has functorial identity types** if it is equipped with a function giving Id_A, r_A and $j_{A,C,d}$ for each suitable Γ, A, C, d such that:

(1) for each $s : \Gamma \to \Delta$ in $\mathcal{C}$, we have an isomorphism $i_{\mathsf{Id}_A,s} : \mathsf{Id}_{A[s]} \cong \mathsf{Id}_A[s.A.A[\pi_A] \circ \chi_0 i_A^{\mathsf{comp}}]$ in $\mathcal{T}_{\Gamma.A[s].A[s][\pi_{A[s]}]}$, where $i_A^{\mathsf{comp}} : A[s][\pi_{A[s]}] \cong A[\pi_A][s.A]$;

(2) for each $A \in \mathcal{T}_\Delta$, $C \in \mathcal{T}_{\Delta.A.A[\pi_A]}$ and $d : \Delta.A.A[\pi_A] \to \Delta.A.A[\pi_A].\mathsf{Id}_A.C$, the following diagram commutes;

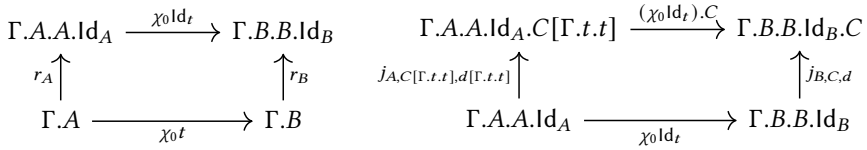
$$\begin{array}{ccc} \Gamma.A[s].A[s][\pi_{A[s]}].\mathsf{Id}_{A[s]}.C[(s.A.A[\pi_A] \circ \chi_0 i_A^{\mathsf{comp}}).\mathsf{Id}_A \circ \chi_0 i_{\mathsf{Id}_A,s}] & \xrightarrow{((s.A.A[\pi_A]\circ\chi_0 i_A^{\mathsf{comp}}).\mathsf{Id}_A\circ\chi_0 i_{\mathsf{Id}_A,s}).C} & \Delta.A.A[\pi_A].\mathsf{Id}_A.C \\ \uparrow{\scriptstyle j_{A[s],C[(s.A.A[\pi_A]\circ\chi_0 i_A^{\mathsf{comp}}).\mathsf{Id}_A \circ \chi_0 i_{\mathsf{Id}_A,s}],d[s.A.A[\pi_A]\circ\chi_0 i_A^{\mathsf{comp}}]}} & & \uparrow{\scriptstyle j_{A,C,d}} \\ \Gamma.A[s].A[s][\pi_{A[s]}].\mathsf{Id}_{A[s]} & \xrightarrow[(s.A.A[\pi_A]\circ\chi_0 i_A^{\mathsf{comp}}).\mathsf{Id}_A\circ\chi_0 i_{\mathsf{Id}_A,s}]{} & \Delta.A.A[\pi_A].\mathsf{Id}_A \\ \uparrow{\scriptstyle r_{A[s]}} & & \uparrow{\scriptstyle r_A} \\ \Gamma.A[s].A[s][\pi_{A[s]}] & \xrightarrow[s.A.A[\pi_A]\circ\chi_0 i_A^{\mathsf{comp}}]{} & \Delta.A.A[\pi_A] \end{array}$$

(3) the comprehension category is equipped with a function giving for each $t : A \to B$ in $\mathcal{T}_\Gamma$, a morphism $\mathsf{Id}_t : \mathsf{Id}_A \to \mathsf{Id}_B$ with $p(\mathsf{Id}_t) = \Gamma.t.t$,;

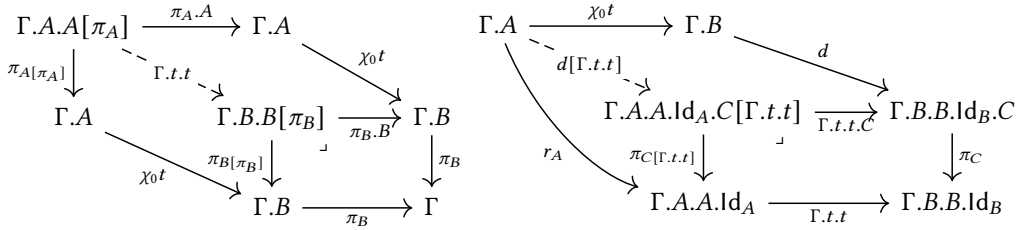
(4) $\mathsf{Id}_{(-)}$ preserves identities, i.e. $\mathsf{Id}_{1_A} = 1_{\mathsf{Id}_A}$ for each $A \in \mathcal{T}_\Gamma$;

(5) $\mathsf{Id}_{(-)}$ preserves composition, i.e. $\mathsf{Id}_{t' \circ t} = \mathsf{Id}_{t'} \circ \mathsf{Id}_t$ for each suitable t and t';

(6) the following diagrams commute for each $t : A \to B$ in $\mathcal{T}_\Gamma$,



where $\Gamma.t.t$ and $d[\Gamma.t.t]$ are given by the universal property of the following pullback square:



(7) $i_{\mathsf{Id}_A,-}$ is functorial in that it preserves i^{iso} and i^{comp} (see [45, Appendix D] for more detail).

Items 1 and 2 of Definition 4.31 state that Id, r and j are preserved under substitution. Items 3 to 7 give the functoriality conditions for expressing the interaction of Id-types with subtyping. Item 6 expresses compatibility of the type morphism structure on the category $\mathcal{T}$ with the type former structure on the category C.

Related Work 4.32 (Gambino and Larrea [24]). Gambino and Larrea [24] also define identity types for comprehension categories. Their introduction and elimination rules are the same as ours. The only difference is that, just like for Π- and Σ-types, we require identity types to be functorial and we phrase stability using isomorphisms instead of Cartesian morphisms.

Related Work 4.33 (Coraglia and Di Liberti [17]). Coraglia and Di Liberti [17] study generalized categories with families, which are equivalent to (non-full) comprehension categories. In that setting, they define *extensional* identity types. Our identity types of Definition 4.31 are *intensional*.

We now discuss examples of comprehension categories with functorial Id-types, including those that arise from AWFSs (see Section 2.3). To interpret identity types in a comprehension category induced by an AWFSs, we need to assume additional structure, namely a **stable functorial choice of path objects** [24, Definition 4.8]. Path objects give us a factorization of the diagonal morphism, and hence, we obtain an interpretation of the identity type [24, Propositon 4.9]. Since the choice of the path object is required to be functorial, comprehension categories induced by weak factorization systems support the functoriality condition in Definition 4.31. These conditions are satisfied by the examples in Section 2.3. In what follows, we discuss functorial Id-types in the these examples.

Example 4.34 (Examples 2.8 and 3.13 ctd.). In the AWFS of groupoids, a type A in context Γ is interpreted as a split isofibration $[\![A]\!] : [\![\Gamma.A]\!] \to [\![\Gamma]\!]$. This example has Id-types. For each groupoid Γ and split isofibration $A : \Gamma.A \to \Gamma$, we have a split isofibration $\mathsf{Id}_A : \Gamma.A.A[\pi_A].\mathsf{Id}_A \to \Gamma.A.A[\pi_A]$, where for each $x \in \Gamma$ and $a, b \in A_x$, the set of objects in the fiber over (x, a, b) is $\hom(a, b)$, i.e. isomorphisms from a to b. Reflexivity is given by the functor $r_A : \Gamma.A \to \Gamma.A.A.\mathsf{Id}_A$ mapping $(x \in \Gamma, a \in A_x)$ to $(x \in \Gamma, a \in A_x, a \in A_x, \mathsf{Id}_a : a \cong a)$. For the elimination rule, for each split isofibration $A : \Gamma.A \to \Gamma$, we have a morphism in $\Gamma.A.A[\pi_A].\mathsf{Id}_A$ as follows:

$$t(\alpha \in a, \beta \in b, p) = (\alpha \in a, p^{-1}(\beta) \in a, \mathsf{Id}_a).$$

For each split isofibration $C : \Gamma.A.A[\pi_A].\mathsf{Id}_A.C \to \Gamma.A.A[\pi_A].\mathsf{Id}_A$ and functor d as follows,

$$d : \Gamma.A \to \Gamma.A.A[\pi_A].\mathsf{Id}_A.C$$
$$(x \in \Gamma, a \in A_x) \mapsto (x \in \Gamma, a \in A_x, b : A_x, \mathsf{Id}_a : a \cong a, d(a) \in C_{a,a,\mathsf{Id}_a}),$$

we have the eliminator $j_{C,d}$:

$$j_{C,d} : \Gamma.A.A[\pi_A].\mathsf{Id}_A \to \Gamma.A.A[\pi_A].\mathsf{Id}_A.C$$
$$(x \in \Gamma, a \in A_x, b \in A_x, p : a \cong b) \mapsto (x \in \Gamma, a \in A_x, b \in A_x, p : a \cong b, t^*(d(a)) \in C_{a,b,p}).$$

Since C is split, we have $j(a, a, \mathsf{Id}_a) = d(a)$ for all $x \in \Gamma$ and $a \in A_x$.

For functoriality, let $t : A \to B$ be a morphism of split fibrations over Γ that maps $(x \in \Gamma, a \in A_x)$ to $(x \in \Gamma, t(a) \in B_x)$. We take $\mathsf{Id}_t : \mathsf{Id}_A \to \mathsf{Id}_B$ to be the following functor over $\Gamma.t.t$:

$$\mathsf{Id}_t : \Gamma.A.A[\pi_A].\mathsf{Id}_A \to \Gamma.B.B[\pi_B].\mathsf{Id}_B$$
$$(x \in \Gamma, a \in A_x, b \in A_x, p : a \cong b) \mapsto (x \in \Gamma, t(a) \in B_x, t(b) \in B_x, t(p) : t(a) \cong t(b)).$$

For the comprehension category built from categories, instead of from groupoids, the set of objects in the fiber over (x, a, b) is $\mathrm{iso}(a, b)$ instead of $\hom(a, b)$, for each $x \in \Gamma$ and $a, b \in A_x$.

The identity type on the example of Heyting algebras (cf. Examples 2.10 and 3.14) is trivial: it denotes equality between proofs, hence gives the singleton type.

Definition 4.35. We define the **extension of CCTT by functorial** Id**-types** to consist of the rules in Fig. 3.

Theorem 4.36 (Soundness of Rules for Id-types). *Any comprehension category with functorial identities models the rules of CCTT and the rules for functorial* Id*-types in Fig. 3.*

Remark 4.37. Under the subtyping point of view Rule subt-id states that $A \leq_t B$ gives $\mathsf{Id}_A \leq_{\mathsf{Id}_t} (\mathsf{Id}_B[\Gamma.t.t])$. Rules subt-id-i and subt-id-c states that $\mathsf{Id}_{(-)}$ preserves identity and composition of subtyping witnesses, respectively.

Coraglia and Emmenegger [19] do not discuss Id-types.

5 Strictly Functorial Substitution: $\mathrm{CCTT}_{\mathrm{split}}$

As discussed in Remark 3.5, substitution in CCTT is functorial only up to *isomorphism*, whereas in many type theories, substitution is functorial up to *equality*. While this makes the type theory easier to use, it comes at a cost; the notion of model of such type theories is more restricted, which makes finding such models more challenging. In comprehension categories, functoriality of substitution is interpreted as splitness of the fibration $p : \mathcal{T} \to \mathcal{C}$. For example, MLTT is interpreted in full *split* comprehension categories.

By replacing those *isomorphisms* that reflect non-splitness of the fibration in the syntax with *equalities*, we obtain $\mathrm{CCTT}_{\mathrm{split}}$, a split version of CCTT with substitution that is functorial up to equality. For this, one needs to also add a judgement $\Gamma \vdash A \equiv B$ for equality of types.

The split version of the judgements and the rules is presented in the preprint [45, Appendix E]. The judgements and rules of $\mathrm{CCTT}_{\mathrm{split}}$ compare to CCTT precisely as follows.

(1) There is a judgement $\Gamma \vdash A \equiv B$ for equality of types in $\mathrm{CCTT}_{\mathrm{split}}$, whereas CCTT does not feature such a judgement.
(2) Isomorphisms of types in the following rules of CCTT are equality of types in $\mathrm{CCTT}_{\mathrm{split}}$: sub-id, sub-comp, sub-cong, pi-sub, sigma-sub, id-sub.
(3) The isomorphism terms in the following rules of CCTT are not present in $\mathrm{CCTT}_{\mathrm{split}}$: sub-tm-id, sub-tm-comp, tm-sub-coh, sub-lam, subt-pi, sub-pair, sub-refl, sub-j.

$$\frac{\Gamma \vdash A\ \mathsf{type}}{\Gamma.A.A[\pi_A] \vdash \mathsf{Id}_A\ \mathsf{type}}\ \text{id-form} \qquad \frac{\Gamma \vdash A\ \mathsf{type}}{\begin{array}{c}\Gamma.A \vdash r_A : \Gamma.A.A[\pi_A].\mathsf{Id}_A \\ \Gamma.A \vdash \pi_{\mathsf{Id}_A} \circ r_A \equiv p_2(1_{\Gamma.A}) : \Gamma.A.A[\pi_A]\end{array}}\ \text{id-intro}$$

$$\frac{\Gamma.A.A[\pi_A].\mathsf{Id}_A \vdash C\ \mathsf{type} \qquad \Gamma.A \vdash d : \Gamma.A.A[\pi_A].\mathsf{Id}_A.C \qquad \Gamma.A \vdash \pi_C \circ d \equiv r_A : \Gamma.A.A.\mathsf{Id}_A}{\Gamma.A.A[\pi_A].\mathsf{Id}_A \vdash j_{A,C,d} : C}\ \text{id-elim}$$

$$\frac{\Gamma.A.A[\pi_A].\mathsf{Id}_A \vdash C\ \mathsf{type} \qquad \Gamma.A \vdash d : \Gamma.A.A[\pi_A].\mathsf{Id}_A.C \qquad \Gamma.A \vdash \pi_C \circ d \equiv r_A : \Gamma.A.A.\mathsf{Id}_A}{\Gamma.A \vdash j_{A,C,d} \circ r_A \equiv d : \Gamma.A.A[\pi_A].\mathsf{Id}_A.C}\ \text{id-beta}$$

$$\frac{\begin{array}{c}\Delta \vdash A\ \mathsf{type} \qquad \Gamma \vdash s : \Delta \\ \Gamma.A[s].A[s][\pi_{A[s]}] \mid \mathsf{Id}_{A[s]} \mathrel{\widetilde{\vdash}} i_{\mathsf{Id}_A,s} : \mathsf{Id}_A[s.A.A[\pi_A] \circ \Gamma.A[s].i_{s.A}]\end{array}}{}\ \text{id-sub}$$

$$\frac{\begin{array}{c}\Delta \vdash A\ \mathsf{type} \qquad \Gamma \vdash s : \Delta \\ \Gamma.A[s].A[s][\pi_{A[s]}] \vdash (s.A.A[\pi_A] \circ \Gamma.A[s].i_{s.A}).\mathsf{Id}_A \circ \Gamma.A[s].A[s][\pi_{A[s]}].i_{\mathsf{Id}_A,s} \circ r_{A[s]} \\ \equiv r_A \circ s.A.A[\pi_A] \circ \Gamma.A[s].i_{s.A} : \Delta.A.A[\pi_A].\mathsf{Id}_A\end{array}}{}\ \text{sub-refl}$$

$$\frac{\begin{array}{c}\Delta \vdash A\ \mathsf{type} \qquad \Gamma \vdash s : \Delta \\ \Gamma.A[s].A[s][\pi_{A[s]}].\mathsf{Id}_{A[s]} \vdash ((s.A.A[\pi_A] \circ \Gamma.A[s].i_{s.A}).\mathsf{Id}_A \circ \Gamma.A[s].A[s][\pi_{A[s]}].i_{\mathsf{Id}_A,s}).C \circ \\ j_{A[s],C[(s.A.A[\pi_A]\circ\Gamma.A[s].i_{s.A}).\mathsf{Id}_A\circ\Gamma.A[s].A[s][\pi_{A[s]}].i_{\mathsf{Id}_A,s}],d[s.A.A[\pi_A]\circ\Gamma.A[s].i_{s.A}]} \equiv j_{A,C,d} \circ \\ (s.A.A[\pi_A] \circ \Gamma.A[s].i_{s.A}).\mathsf{Id}_A \circ \Gamma.A[s].A[s][\pi_{A[s]}].i_{\mathsf{Id}_A,s} : \Delta.A.A[\pi_A].\mathsf{Id}_A.C\end{array}}{}\ \text{sub-j}$$

$$\frac{\Gamma \mid A \vdash t : B}{\Gamma.A.A[\pi_A] \mid \mathsf{Id}_A \vdash \mathsf{Id}_t : \mathsf{Id}_B[\Gamma.t.t]}\ \text{subt-id} \qquad \frac{\Gamma \vdash A\ \mathsf{type}}{\Gamma.A.A[\pi_A].\mathsf{Id}_A \vdash \mathsf{Id}_{1_{\Gamma.A}} \equiv 1_{\mathsf{Id}_A} : \Gamma.A.A[\pi_A].\mathsf{Id}_A[1_{\Gamma.A.A[\pi_A]}]}\ \text{subt-id-i}$$

$$\frac{\Gamma \mid A \vdash t : B \qquad \Gamma \mid B \vdash t' : C}{\Gamma.A.A[\pi_A] \mid \mathsf{Id}_A \vdash \mathsf{Id}_{t'}[\Gamma.t.t] \circ \mathsf{Id}_t \equiv \mathsf{Id}_{t' \circ_t} : \mathsf{Id}_C[\Gamma.t'.t'][\Gamma.t.t]}\ \text{subt-id-c}$$

$$\frac{\Gamma \mid A \vdash t : B}{\Gamma.A \vdash (\Gamma.t.t).\mathsf{Id}_B \circ \Gamma.A.A.\mathsf{Id}_t \circ r_A \equiv r_B \circ \Gamma.t : \Gamma.B.B.\mathsf{Id}_B}\ \text{subt-id-refl}$$

$$\frac{\Gamma \mid A \vdash t : B}{\Gamma.A.A.\mathsf{Id}_A \vdash ((\Gamma.t.t).\mathsf{Id}_B \circ \Gamma.A.A.\mathsf{Id}_t).C \circ j_{A,C[\Gamma.t.t],d[\Gamma.t.t]} \equiv j_{B,C,d} \circ (\Gamma.t.t).\mathsf{Id}_B \circ \Gamma.A.A.\mathsf{Id}_t : \Gamma.B.B.\mathsf{Id}_B.C}\ \text{subt-id-j}$$

$$\frac{\Gamma \vdash A\ \mathsf{type}}{\Gamma \mid \mathsf{Id}_{A[1_\Gamma]} \vdash i^{\mathsf{sub}}_{\mathsf{Id}_A,1_\Gamma.A.A[\pi_A]\circ\Gamma.i_{1_\Gamma.A},1_\Gamma} \circ i_{\mathsf{Id}_A,1_A} \equiv i^{\mathsf{id}^{-1}}_{\mathsf{Id}_A} \circ \mathsf{Id}_{i^{\mathsf{id}}_A} : \mathsf{Id}_A[1_\Gamma]}\ \text{id-sub-id}$$

$$\frac{\Theta \vdash A\ \mathsf{type} \qquad \Gamma \vdash s' : \Delta \qquad \Delta \vdash s : \Theta}{\begin{array}{c}\Gamma \mid \mathsf{Id}_{A[s\circ s']} \vdash i^{\mathsf{comp}}_{\mathsf{Id}_A,s.A.A[\pi_A]\circ\Gamma.i_{s.A},s'.A[s].A[s][\pi_{A[s]}]\circ\Gamma.i_{s'.A[s]}} \circ \\ i^{\mathsf{sub}}_{\mathsf{Id}_A,s\circ s'.A.A[\pi_A]\circ\Gamma.i_{(s\circ s').A},s.A.A[\pi_A]\circ\Gamma.i_{s.A}\circ s'.A[s].A[s][\pi_{A[s]}]\circ\Gamma.i_{s'.A[s]}} \circ i_{\mathsf{Id}_A,s\circ s'} \equiv \\ i_{\mathsf{Id}_A,s}[s'.A[s].A[s][\pi_{A[s]}] \circ \Gamma.i_{s'.A[s]}] \circ i_{\mathsf{Id}_{A[s]},s'} \circ \mathsf{Id}_{i^{\mathsf{comp}}_{A,s,s'}} : \mathsf{Id}_{A[s]}[s'.A[s].A[s][\pi_{A[s]}] \circ \Gamma.i_{s'.A[s]}]\end{array}}\ \text{id-sub-comp}$$

In Rules id-sub, sub-refl, sub-j,id-sub-id and id-sub-comp, $i_{s.A} \coloneqq i^{\mathsf{comp}}_{A,\pi_A,s.A} \circ i^{\mathsf{sub}}_{A,s\circ\pi_{A[s]},\pi_A\circ s.A} \circ i^{\mathsf{comp}^{-1}}_{A,s,\pi_{A[s]}}$. In Rule subt-id-j, $\Gamma.t.t \coloneqq (\Gamma.t \circ \pi_{A[\pi_A]}).B[\pi_B] \circ p_2(\Gamma.t \circ \pi_A.A)$.

Fig. 3. Rules for functorial Id-types. Rules id-sub, sub-refl, sub-j, id-sub-id and id-sub-comp use the notation introduced in Proposition 3.6. For example, in Rule id-sub, $s.A$ is $(s \circ \pi_{A[s]}, \Gamma.(i^{\mathsf{comp}}_{A,s,\pi_{A[s]}}[\pi_{A[s]}]) \circ p_2(1_{\Gamma.A[s]}))$.

(4) Isomorphisms in the following rules of CCTT are identity morphisms in $\mathrm{CCTT}_{\mathrm{split}}$: sub-proj-id, sub-proj-comp, subt-pi-id, subt-sigma-id
(5) The following coherence rules of CCTT, explained in Remark 3.7, are not in $\mathrm{CCTT}_{\mathrm{split}}$: second conclusion of sub-cong, sub-cong-id, sub-cong-comp-1, sub-cong-comp-2. In addition, the following coherence rules are not in $\mathrm{CCTT}_{\mathrm{split}}$: pi-sub-id,pi-sub-comp, sigma-sub-id, sigma-sub-comp, id-sub-id and id-sub-comp.

Theorem 5.1 (Soundness for $\mathrm{CCTT}_{\mathrm{split}}$). *Every split comprehension category models CCTT_{split}.*

Remark 5.2 (About Implicit Substitution). We could also consider a *strict* syntax with *implicit* substitution rather than with *explicit* substitution. Such an implicit substition is automatically functorial up to equality. Gambino and Larrea [24] provide a splitting construction that can be used to turn models of CCTT into models of a variant with implicit substitution. Lumsdaine and Warren [40] also provide a splitting construction, but only for *full* comprehension categories.

6 Related Work

In this section, we discuss related work and the precise relationship to our work.

6.1 Work on Type Formers

Comprehension categories, and semantic structures for the interpretation of type theory, were defined by Jacobs, first in a seminal paper [34] and, later, in a comprehensive book [35]. Jacobs assumes comprehension categories to be full throughout. We give precise comparisons between our work and that of Jacobs throughout this paper, in dedicated environments (Related Work 4.3, 4.17 and 4.30 and Propositions 4.7 and 4.20).

Lindgren [39] defines a semantic structure on non-full comprehension categories suitable for the interpretation of dependent product types; we give more detail in Related Work 4.5.

Lumsdaine and Warren [40] discuss a splitting construction for *full* comprehension categories. They define different versions of categorical structures for the interpretation of type formers suitable for full comprehension categories. For a comparison of our and their structures for type formers, see Related Work 4.4 and 4.18.

Gambino and Larrea [24] shows that splitting construction by Hofmann [31] extends to *non-full* comprehension categories. They consider comprehension categories arising from AWFSs, and suitable structure for type formers. For a comparison of our and their structures for type formers, see Related Work 4.9, 4.22 and 4.32.

Ahrens, North, and Van der Weide [5] develop a syntax for comprehension **bi**categories, with the goal of developing a notion of directed type theory. They do not study type formers, but only structural rules.

Curien, Garner, and Hofmann [20] develop a type theory with explicit substitution to more accurately reflect the intended categorical semantics: explicit substitution is not necessarily strict, and thus better aligns with the interpretation of substitution as pullback. They give an interpretation of their type theory in comprehension categories; this interpretation does not make use of morphisms between types, and the authors note that "[i]t is therefore natural to limit attention to full comprehension categories". Like [20] we have an explicit substitution operation in the rules we develop; however, regarding morphisms between types, we take a different approach by extending the syntax by a corresponding judgement for such morphisms.

6.2 Work on Subtyping

Subtyping in type theory has been studied extensively, from both *semantic* and *syntactic* angles. We discuss what seems to us the most closely related work to ours; the overview below is by no means claimed to be exhaustive.

We first discuss work studying the *semantics* of subtyping.

Firstly, Zeilberger and Melliès [44] give a fibrational view of **subsumptive** subtyping, unlike this paper where we discuss **coercive** subtyping. They interpret type systems as functors from a category of type derivations to a category of underlying terms. In this setting, subtyping derivations are interpreted as vertical morphisms, i.e. the derivations mapped to the identity morphism of the underlying term.

Secondly, Coraglia and Emmenegger [19] study "generalized categories with families", a notion that they show is equivalent to the (non-full) comprehension categories discussed in the present paper [18]. Taking a semantic viewpoint on subtyping, they sketch [19] how generalized categories with families interpret some rules related to **coercive** subtyping, notably the rules of transitivity, subsumption, weakening, substitution, and rules related to the type formers Π and Σ — for details,

see Table 1 and Related Work 4.15 and 4.28. We develop that work further, by presenting more structural rules, and analyzing identity types as well.

We point out one potential source of confusion when comparing the type-theoretic rules presented in the present work with the rules shown by Coraglia and Emmenegger [19, Propositions 20 and 21]. Specifically, "[I]n writing the rules above in Propositions 20 and 21, [Coraglia and Emmenegger] have written the action of reindexing as if the fibrations involved were split." That is, even though Coraglia and Emmenegger study comprehension categories that are not necessarily split, they present a simplified versions of their rules which can be interpreted only in **split** comprehension categories. In our work, we present type-theoretic rules suitable for interpretation in any comprehension category, not necessarily split. As a consequence, some of our rules contain more coherence isomorphisms.

Next, we discuss work studying subtyping from a *syntactic* point of view.

Firstly, Luo and Adams [41] study structural coercive subtyping for inductive data types. They propose functoriality of type formers and prove desirable properties, such as admissibility of transitivity of subtyping, of the resulting theory.

Secondly, Laurent et al. [38] extend MLTT to a type theory with definitionally functorial type formers. They use this functoriality to extend MLTT to two type theories with coercive and subsumptive subtyping, respectively. They show that the functoriality of type formers is sufficient to establish back-and-forth translations between the two type theories, resulting in an equivalence between them. They also study meta-theoretic properties of their systems, e.g. showing that their functorial system is normalizing and has decidable type checking.

A rough comparison of the type theory from Laurent et al. [38] with coercive subtyping, called $\mathrm{MLTT}_{\mathrm{coe}}$, to ours is as follows. The syntax of $\mathrm{MLTT}_{\mathrm{coe}}$ supports at most one coercion between any two types. This corresponds to considering a thin category of types in our semantics. Furthermore, the syntax of $\mathrm{MLTT}_{\mathrm{coe}}$ comes with substitution which is strictly functorial. This corresponds to considering a split fibration in our semantics. The split version of our syntax is discussed in Section 5. If we assume splitness and thinness in our syntax, our rules for Π- and Σ-types imply theirs. A more notable difference is that the identity type of $\mathrm{MLTT}_{\mathrm{coe}}$ does not have a counterpart to our Rule subt-id-j which expresses that the eliminator of identity type is preserved by coercion. As the authors explain in Section 3.3 of another version of their work [37], they make a design choice to not include such a rule as it is not necessary for having their desired functorial equations.

Thirdly, Adjedj et al. [2] develop a type theory they call AdapTT that captures type casting and coercive subtyping. They show that AdapTT is modelled by $\mathrm{NatMod}_{\mathrm{DO}}$, a structure equivalent to *split* generalized categories with families. Similar to $\mathrm{CCTT}_{\mathrm{split}}$, substitution in their syntax is strictly functorial. Furthermore, they provide a general framework, called AdapTT_2, for defining type formers that are automatically functorial, including general inductive types specified by a notion of signature.

Fourthly, Aspinall and Compagnoni [7] study syntactic properties of dependent type theory with subtyping; in particular, they prove subject reduction and decidability of type-checking for a theory with dependent types and subtyping.

Fifthly, Luo et al. [42] study **coercive** subtyping, the form of subtyping analyzed semantically in this paper. They show that coercive subtyping provides a conservative extension of a type theory.

7 Conclusion

We have presented the judgements and rules of CCTT which reflect the structure of comprehension categories. Specifically, we have presented structural rules, and rules for type and term formers for dependent pairs, dependent functions, and identity types. We have also presented categorical structures on comprehension categories that are suitable for the interpretation of the type and term

formers, and we have given a sound interpretation of our rules in such comprehension categories. Furthermore, we have explained how our rules are a form of proof-relevant subtyping, extending work by Coraglia and Emmenegger [19]. We have given an interpretation of the rules in models arising from algebraic weak factorization systems.

We have not touched on the question of whether our syntax is complete for comprehension categories. We conjecture that it is, although we have not yet established this rigorously. One could follow Garner in his development of 2-dimensional models of type theory [25]. Garner constructs an equivalence between, on the one hand, a category of suitable generalized algebraic theories and, on the other hand, the category of models he is studying.

What does this work buy us? From a semantic point of view, the rules of CCTT distill the essence of non-full comprehension categories, a semantic structure arising from AWFSs, which in turn are frequently used for the interpretation of type theories. From a syntactic point of view, CCTT provides a framework for theories with coercive subtyping.

Additionally, CCTT provides a basis for strictifying type theory with additional definitional equalities, in the following sense. As discussed in Section 2.3, in homotopy-theoretic models of Martin-Löf type theory that arise from AWFSs (e.g., Example 2.8), type morphisms are morphisms of algebras: that is, they preserve the algebra structure of types. In other words – using the language of homotopy type theory – in these models, type morphisms are morphisms which preserve transport *strictly*. Thus, one could add rules to CCTT expressing that type morphisms preserve transport strictly, and these rules would be validated by such models. Additionally, many commonly used functions in Martin-Löf type theory are algebra morphisms in these models, and thus could be asserted to be type morphisms in rules added to CCTT. For instance, both the first projection from a Σ-type and constant functions are type morphisms in arbitrary AWFSs; thus one can add rules to CCTT expressing that these functions commute strictly with transport. The value of having such rules comes from the prevalence of calculations with transport in type theory (often that the first projection preserves transport strictly). Indeed, while such calculations are mathematically straightforward and often omitted from accounts in published papers of formalized mathematics (as done, for instance, by Ahrens and Lumsdaine [3]), they "pollute" computer-checked libraries. See Sojakova [48] for an account of a piece of synthetic homotopy theory that explicitly describes the many instances of such calculations. Additional definitional equalities simplify such calculations, because the proof assistants can take over more work from the user. As such, CCTT integrates into recent research in type theory aimed at justifying and implementing more definitional equalities in type theory [6, 15, 16, 38, 50].

We conclude with a question we have left open: How does the subtyping point of view carry over to the bicategorical type theory and its interpretation in comprehension **bi**categories as studied by Ahrens et al. [5]?

Acknowledgments

We thank Ambroise Lafont and Arunava Gantait for feedback on drafts of this paper, and Arthur Adjedj, Kenji Maillard, Noam Zeilberger, Thibaut Benjamin, and Tom de Jong for useful discussions of related work. We also thank the anonymous referees for their careful reading and helpful comments. This research was supported by the NWO project "The Power of Equality" OCENW.M20.380, which is financed by the Dutch Research Council (NWO). This work received government funding managed by the French National Research Agency under the France 2030 program, reference "ANR-22-EXES-0013".

References

[1] Martín Abadi, Luca Cardelli, Pierre-Louis Curien, and Jean-Jacques Lévy. 1991. Explicit Substitutions. *J. Funct. Program.* 1, 4 (1991), 375–416. doi:10.1017/S0956796800000186

[2] Arthur Adjedj, Meven Lennon-Bertrand, Thibaut Benjamin, and Kenji Maillard. 2026. AdapTT: Functoriality for Dependent Type Casts. POPL (2026). doi:10.1145/3776664

[3] Benedikt Ahrens and Peter LeFanu Lumsdaine. 2019. Displayed Categories. *Log. Methods Comput. Sci.* 15, 1 (2019). doi:10.23638/LMCS-15(1:20)2019

[4] Benedikt Ahrens, Peter LeFanu Lumsdaine, and Paige Randall North. 2024. Comparing Semantic Frameworks for Dependently-Sorted Algebraic Theories. In *Programming Languages and Systems - 22nd Asian Symposium, APLAS 2024, Kyoto, Japan, October 22-24, 2024, Proceedings (Lecture Notes in Computer Science, Vol. 15194)*, Oleg Kiselyov (Ed.). Springer, 3–22. doi:10.1007/978-981-97-8943-6_1

[5] Benedikt Ahrens, Paige Randall North, and Niels van der Weide. 2023. Bicategorical type theory: semantics and syntax. *Math. Struct. Comput. Sci.* 33, 10 (2023), 868–912. doi:10.1017/S0960129523000312

[6] Thorsten Altenkirch, Conor McBride, and Wouter Swierstra. 2007. Observational equality, now!. In *Proceedings of the ACM Workshop Programming Languages meets Program Verification, PLPV 2007, Freiburg, Germany, October 5, 2007*, Aaron Stump and Hongwei Xi (Eds.). ACM, 57–68. doi:10.1145/1292597.1292608

[7] David Aspinall and Adriana B. Compagnoni. 2001. Subtyping dependent types. *Theor. Comput. Sci.* 266, 1-2 (2001), 273–309. doi:10.1016/S0304-3975(00)00175-4

[8] Steve Awodey. 2018. A cubical model of homotopy type theory. *Annals of Pure and Applied Logic* 169, 12 (2018), 1270–1294. doi:10.1016/j.apal.2018.08.002

[9] Steve Awodey, Evan Cavallo, Thierry Coquand, Emily Riehl, and Christian Sattler. 2024. The equivariant model structure on cartesian cubical sets. *arXiv:2406.18497* (2024).

[10] Benno van den Berg and Eric Faber. 2022. *Effective Kan fibrations in simplicial sets.* Springer Cham. doi:10.1007/978-3-031-18900-5

[11] John Bourke and Richard Garner. 2016. Algebraic Weak Factorisation Systems I: Accessible AWFS. *Journal of Pure and Applied Algebra* 220, 1 (2016), 108–147. doi:10.1016/j.jpaa.2015.06.002

[12] John Cartmell. 1978. *Generalised Algebraic Theories and Contextual Categories.* Ph. D. Dissertation. Oxford University, UK.

[13] John Cartmell. 1986. Generalised algebraic theories and contextual categories. *Ann. Pure Appl. Log.* 32 (1986), 209–243. doi:10.1016/0168-0072(86)90053-9

[14] Evan Cavallo, Anders Mörtberg, and Andrew W Swan. 2020. Unifying cubical models of univalent type theory. In *28th EACSL Annual Conference on Computer Science Logic (CSL 2020)*. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 14–1. doi:10.4230/LIPIcs.CSL.2020.14

[15] Jesper Cockx. 2019. Type Theory Unchained: Extending Agda with User-Defined Rewrite Rules. In *25th International Conference on Types for Proofs and Programs, TYPES 2019, June 11-14, 2019, Oslo, Norway (LIPIcs, Vol. 175)*, Marc Bezem and Assia Mahboubi (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2:1–2:27. doi:10.4230/LIPICS.TYPES.2019.2

[16] Cyril Cohen, Thierry Coquand, Simon Huber, and Anders Mörtberg. 2018. Cubical Type Theory: A Constructive Interpretation of the Univalence Axiom. In *21st International Conference on Types for Proofs and Programs (TYPES 2015) (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 69)*, Tarmo Uustalu (Ed.). Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 5:1–5:34. doi:10.4230/LIPIcs.TYPES.2015.5

[17] Greta Coraglia and Ivan Di Liberti. 2024. Context, Judgement, Deduction. arXiv:2111.09438 [math.LO] https://arxiv.org/abs/2111.09438 Accepted at the proceedings of CatMi 2023.

[18] Greta Coraglia and Jacopo Emmenegger. 2024. A 2-Categorical Analysis of Context Comprehension. *Theory and Applications of Categories* 41 (2024), Paper No. 42,1476–1512.

[19] Greta Coraglia and Jacopo Emmenegger. 2024. Categorical Models of Subtyping. In *29th International Conference on Types for Proofs and Programs (TYPES 2023) (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 303)*, Delia Kesner, Eduardo Hermo Reyes, and Benno van den Berg (Eds.). Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 3:1–3:19. doi:10.4230/LIPIcs.TYPES.2023.3

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

[21] Peter Dybjer. 1996. Internal Type Theory. In *Types for Proofs and Programs, International Workshop TYPES'95, Torino, Italy, June 5-8, 1995, Selected Papers (Lecture Notes in Computer Science, Vol. 1158)*, Stefano Berardi and Mario Coppo (Eds.). Springer, 120–134. doi:10.1007/3-540-61780-9_66

[22] Jacopo Emmenegger, Fabio Pasquali, and Giuseppe Rosolini. 2022. A characterisation of elementary fibrations. *Annals of Pure and Applied Logic* 173, 6 (2022), 103103. doi:10.1016/j.apal.2022.103103

[23] Marcelo Fiore. 2022. Semantic Analysis of Normalisation by Evaluation for Typed Lambda Calculus. 32, 8 (2022), 1028–1065. doi:10.1017/S0960129522000263

[24] Nicola Gambino and Marco Federico Larrea. 2023. Models of Martin-Löf Type Theory from Algebraic Weak Factorisation Systems. *The Journal of Symbolic Logic* 88, 1 (2023), 242–289. doi:10.1017/jsl.2021.39

[25] Richard Garner. 2009. Two-Dimensional Models of Type Theory. *Mathematical Structures in Computer Science* 19, 4 (2009), 687–736. doi:10.1017/S0960129509007646

[26] Richard Garner. 2009. Understanding the Small Object Argument. *Applied Categorical Structures* 17, 3 (2009), 247–285. doi:10.1007/s10485-008-9137-4

[27] Marco Grandis and Walter Tholen. 2006. Natural Weak Factorization Systems. *Universitatis Masarykianae Brunensis* 42, 4 (2006), 397–408.

[28] Daniel Gratzer. 2022. Normalization for Multimodal Type Theory. In *LICS '22: 37th Annual ACM/IEEE Symposium on Logic in Computer Science, Haifa, Israel, August 2 - 5, 2022*, Christel Baier and Dana Fisman (Eds.). ACM, 2:1–2:13. doi:10.1145/3531130.3532398

[29] Daniel Gratzer, G. A. Kavvos, Andreas Nuyts, and Lars Birkedal. 2021. Multimodal Dependent Type Theory. *Log. Methods Comput. Sci.* 17, 3 (2021). doi:10.46298/LMCS-17(3:11)2021

[30] Denis Higgs. 1984. Injectivity in the Topos of Complete Heyting Algebra Valued Sets. *Canadian Journal of Mathematics* 36, 3 (1984), 550–568. doi:10.4153/CJM-1984-034-4

[31] 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, Vol. 933)*, Leszek Pacholski and Jerzy Tiuryn (Eds.). Springer, 427–441. doi:10.1007/BFB0022273

[32] Martin Hofmann and Thomas Streicher. 1998. The Groupoid Interpretation of Type Theory. In *Twenty-Five Years of Constructive Type Theory (Venice, 1995)*. Oxford Logic Guides, Vol. 36. Oxford Univ. Press, New York, 83–111. doi:10.1093/oso/9780198501275.003.0008

[33] Joseph Hua, Steve Awodey, Mario Carneiro, Sina Hazratpour, Wojciech Nawrocki, Spencer Woolfson, and Yiming Xu. 2025. HoTTLean: Formalizing the Meta-Theory of HoTT in Lean. *TYPES 2025* (2025).

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

[35] Bart Jacobs. 1998. *Categorical Logic and Type Theory*. Studies in logic and the foundations of mathematics, Vol. 141. Elsevier.

[36] Krzysztof Kapulkin and Peter LeFanu Lumsdaine. 2021. The simplicial model of Univalent Foundations (after Voevodsky). *Journal of the European Mathematical Society* 23, 6 (March 2021), 2071–2126. doi:10.4171/jems/1050

[37] Théo Laurent, Meven Lennon-Bertrand, and Kenji Maillard. 2023. Definitional Functoriality for Dependent (Sub)Types. (July 2023). https://hal.science/hal-04160858 working paper or preprint.

[38] Théo Laurent, Meven Lennon-Bertrand, and Kenji Maillard. 2024. Definitional Functoriality for Dependent (Sub)Types *(Lecture Notes in Computer Science, Vol. 14576)*, Stephanie Weirich (Ed.). Springer, 302–331. doi:10.1007/978-3-031-57262-3_13

[39] Michael Lindgren. 2021. *Dependent products as relative adjoints*. Master's thesis. Stockholm University. https://ncatlab.org/nlab/files/Lindgren-DependentProductsAsRelativeAdjoints.pdf

[40] Peter Lefanu Lumsdaine and Michael A. Warren. 2015. The Local Universes Model: An Overlooked Coherence Construction for Dependent Type Theories. *ACM Transactions on Computational Logic* 16, 3 (July 2015), 1–31. doi:10.1145/2754931

[41] Zhaohui Luo and Robin Adams. 2008. Structural subtyping for inductive types with functorial equality rules. *Math. Struct. Comput. Sci.* 18, 5 (2008), 931–972. doi:10.1017/S0960129508006956

[42] Zhaohui Luo, Sergei Soloviev, and Tao Xue. 2013. Coercive subtyping: Theory and implementation. *Inf. Comput.* 223 (2013), 18–42. doi:10.1016/J.IC.2012.10.020

[43] Per Martin-Löf. 1984. *Intuitionistic type theory*. Studies in proof theory, Vol. 1. Bibliopolis.

[44] Paul-André Melliès and Noam Zeilberger. 2015. Functors are Type Refinement Systems, Sriram K. Rajamani and David Walker (Eds.). ACM, 3–16. doi:10.1145/2676726.2676970

[45] Niyousha Najmaei, Niels van der Weide, Benedikt Ahrens, and Paige Randall North. 2025. From Semantics to Syntax: A Type Theory for Comprehension Categories. arXiv:2503.10868v2 [cs.PL] https://arxiv.org/abs/2503.10868v2

[46] Paige Randall North. 2019. Towards a Directed Homotopy Type Theory. In *Proceedings of the Thirty-Fifth Conference on the Mathematical Foundations of Programming Semantics, MFPS 2019, London, UK, June 4-7, 2019 (Electronic Notes in Theoretical Computer Science, Vol. 347)*, Barbara König (Ed.). Elsevier, 223–239. doi:10.1016/J.ENTCS.2019.09.012

[47] Andrew M. Pitts. 2000. Categorical Logic. In *Handbook of Logic in Computer Science, Vol. 5*. Handb. Log. Comput. Sci., Vol. 5. Oxford Univ. Press, New York, 39–128.

[48] Kristina Sojakova. 2016. The Equivalence of the Torus and the Product of Two Circles in Homotopy Type Theory. *ACM Trans. Comput. Log.* 17, 4 (2016), 29. doi:10.1145/2992783

[49] Thomas Streicher. 2018. Fibered Categories à la Jean Bénabou. arXiv:1801.02927 [math.CT] https://arxiv.org/abs/1801.02927

[50] Pierre-Yves Strub. 2010. Coq Modulo Theory. In *Computer Science Logic, 24th International Workshop, CSL 2010, 19th Annual Conference of the EACSL, Brno, Czech Republic, August 23-27, 2010. Proceedings (Lecture Notes in Computer Science, Vol. 6247)*, Anuj Dawar and Helmut Veith (Eds.). Springer, 529–543. doi:10.1007/978-3-642-15205-4_40
[51] Andrew Swan. 2018. Identity types in algebraic model structures and cubical sets. *arXiv:1808.00915* (2018).
[52] The Univalent Foundations Program. 2013. *Homotopy Type Theory: Univalent Foundations of Mathematics*. https://homotopytypetheory.org/book, Institute for Advanced Study.

Received 2025-07-10; accepted 2025-11-06