Semantics for two-dimensional type theory

BENEDIKT AHRENS, Delft University of Technology, The Netherlands and University of Birmingham, United Kingdom
PAIGE RANDALL NORTH, University of Pennsylvania, United States
NIELS VAN DER WEIDE, Radboud University, The Netherlands

We propose a general notion of model for two-dimensional type theory, in the form of *comprehension bicategories*. Examples of comprehension bicategories are plentiful; they include interpretations of directed type theory previously studied in the literature.

From comprehension bicategories, we extract a core syntax, that is, judgment forms and structural inference rules, for a two-dimensional type theory. We prove soundness of the rules by giving an interpretation in any comprehension bicategory.

The semantic aspects of our work are fully checked in the Coq proof assistant, based on the UniMath library.

This work is the first step towards a theory of syntax and semantics for higher-dimensional directed type theory.

Additional Key Words and Phrases: directed type theory, dependent types, comprehension bicategory, computer-checked proof

1 INTRODUCTION

In recent years, efforts have been made to develop *directed* type theory. Roughly, directed type theory should correspond to Martin-Löf type theory (MLTT) as ∞-categories correspond to ∞-groupoids. Besides theoretical interest in directed type theory, it is hoped that such a type theory can serve as a framework for synthetic directed homotopy theory and synthetic ∞-category theory. Applications of those, in turn, include reasoning about concurrent processes [15].

Several proposals for *syntax* for directed type theory have been given (reviewed in Section 2), but are ad-hoc and are not always semantically justified. The *semantic* aspects of directed type theory are particularly underdeveloped; a general notion of model of a directed type theory is still lacking.

In this work, we approach the development of directed type theory from the semantic side. We introduce *comprehension bicategories* as a suitable mathematical structure for higher-dimensional (directed) type theory. Comprehension bicategories capture several different specific mathematical structures that have previously been used to interpret higher-dimensional or directed type theory.

From comprehension bicategories, we extract the core syntax—judgment forms and structural inference rules—of a two-dimensional dependent type theory that can accommodate directed type theory. We also give a soundness proof of our structural rules. In separate work, we will equip our syntax and semantics with variances and type and term formers for directed type theory.

To motivate our approach, we analyze in Section 1.1 how higher-groupoidal structure arises in MLTT through an interplay of judgmental equality and typal identity. Our analysis thus leads to the desiderata listed in Section 1.2. In Section 1.3 we discuss the foundations we work in, and aspects of the computer formalization of some of our results.

1.1 Judgmental and Typal Higher Dimensions

When discussing two- or higher-dimensional type theory, we need to understand how these dimensions are generated.

The judgment forms of traditional MLTT specify types, contexts, terms, and judgmental equality (conversion) between types and terms. There is, prima facie, nothing higher-dimensional about these judgments, and an interpretation of types as sets and terms as elements of sets seems perfectly adequate. In this sense, Martin-Löf type theory is 1-dimensional. However, MLTT is often said to be ∞-dimensional. The higher dimensions are generated by the identity type, which internalizes the judgmental equality; specifically, the well-known **reflexivity** rule generates a typal identity from a judgmental equality. Since the identity type can be iterated, judgmental equality then also becomes available for terms of the identity type itself. This mutual interaction between judgmental equality and typal identity provides the infrastructure to "lift" judgmental equality to higher dimensions without extending the judgmental structure of MLTT. The tower of types $(A, \mathsf{Id}_A, \mathsf{Id}_{\mathsf{Id}_A}, \ldots)$ then can be given the structure of an ∞-groupoid, as shown by [9, 31].

When developing a *directed* type theory, with models in ∞-*categories*, analogous ingredients are required:

I1: A *judgment* of *(directed) reductions* between types and terms, analogous to judgmental equality;

I2: A *type former* for *homomorphisms* between terms, analogous to identity types.

I3: A notion of *model* in which to interpret the judgments and type formers.

Previous work on higher-dimensional and directed type theory has focused on either syntax (I1/I2) or semantics (I3), but not on both. Licata and Harper [28, 29] and Nuyts [33] devise judgmental structure for higher-dimensional and directed type theory. North [32] devises a type former for directed homomorphisms between terms, on top of the judgmental structure of MLTT. None of them proposes an adequate general definition of *model* of directed type theory. Garner [19] defines a notion of higher-dimensional model, but considers only *un*directed type theory.

In the present work, we propose a judgmental framework (I1), and a suitable general notion of semantics (I3), for higher-dimensional and directed type theory. In a separate work we will expand this core by a system of *variances* suitable for accommodating a type former akin to North's hom-types (I2), to build a fully functional higher-dimensional type theory.

1.2 Syntax and Semantics, Semantics and Syntax

Most previous work on directed type theory privileged the development of syntactic aspects over the semantic ones. We choose to approach the challenge from the other direction: we start by devising a suitable categorical structure for directed type theory, and extract from it a syntax. When developing syntax and semantics, we applied the following "quality criteria":

Q1: The obtained syntax should express contexts, types, terms, and reductions between terms.

Q2: The semantics considered by Garner [19], Licata and Harper [28, 29], and North [32] should be *instances* of our semantics (modulo variances and type constructors that are not considered here).

The semantics we propose are described in Section 5, and the extracted syntax is described in Section 8. Both our syntax and semantics are quite general; for instance, our reductions are *proof-relevant*—like those in [28, 29], and unlike judgmental equality in MLTT, which is proof-irrelevant. Syntax and semantics could reasonably be simplified or specialized. Crucially, our work provides a framework to modify syntax and semantics *in lockstep*, with a clear mechanism to analyze changes to the syntax on the semantic side and vice versa. We suggest some possible changes in Section 10.

Following this analysis and workplan, we derive the following goals for our work:

D1: a system of inference rules for dependent types with *directed* reductions between terms;

D2: a definition of *mathematical structures* suitable for the mathematical modelling of the syntactic rules;

D3: an *interpretation* of the inference rules in such a mathematical structure;

D4: a syntax for type and term formers on top of D1;

D5: a semantic structure for the interpretation of type and term formers.

In the present work, we achieve desiderata D1, D2, and D3. The study of variances and type and term constructors will be reported on elsewhere.

1.3 Foundations and Formalization in UniMath

The main results presented here are agnostic to foundations; they can be formalized in both set theory and type theory.

However, some of the notions we employ can economically be formulated using dependent types. In particular, we work with (Grothendieck) fibrations of (bi)categories, whose formulation in set theory usually relies on postulating equality of objects. Using dependent types, a formulation of such concepts can be given that avoids any reasoning about equality of objects; instead, these concepts are formulated in terms of fibers. For this reason, we use type-theoretic language throughout the paper; see also, *e. g.,* Remark 4.5. More precisely, we work in univalent foundations; in particular, we formulate results and examples in terms of **univalent** (bi)categories [2, 3]. These are equivalent to set-theoretic (bi)categories via Voevodsky's model in Kan complexes [26].

We carefully distinguish data and property; specifically, we postulate elements to be explicitly given as data rather than to merely exist. We do not rely on any choice axioms or on excluded middle.

The semantic results of this work are checked in Coq [45], based on the UniMath [46] library of univalent mathematics. Our code has been integrated into UniMath. To document our formalization, we refer to UniMath commit 840ac16. Many definitions are accompanied by a link (*e. g.,* bicat) to the corresponding definition in an HTML version of that commit. The code written specifically for this work comprises approximately 33,600 lines of code; specifically, the coqwc tool counts

```
 spec    proof comments
13143    20553      449 total
```

We build upon an existing library of (bi)category theory [2, 3], and use heavily the *displayed* machinery, developed for (1-)categories in [4] and extended to bicategories in [2]. In particular, the notions of Grothendieck fibration we are using (in the 1-categorical case) and developing (in the bicategorical case) are based on displayed (bi)categories; we can thus discuss these notions without postulating equality of objects.

1.4 Synopsis

In Section 2 we review related work. In Section 3 we review (displayed) bicategories and functors. In Section 4 we define cloven global and local (op/iso)fibrations of bicategories, and we use these notions to define comprehension bicategories. In Section 5 we discuss some examples of comprehension bicategories. In Section 6 we discuss Street (op)fibrations internal to bicategories, which form our main examples of comprehension bicategories. In Section 7 we define display map bicategories, and we show how any display map bicategory gives rise to a comprehension bicategory. In Section 8 we present structural type-theoretic rules for the syntax of a two-dimensional type theory, which we call BTT. In Section 9 we give an interpretation of BTT in any comprehension bicategory. In Section 10 we discuss variations of BTT and the semantic structures these variations correspond to.

1.5 Version History

A shorter version of this paper was published in the proceedings of Logic in Computer Science [5]. Compared to that shorter version, the following material has been added in the present version:

- We provide more instantiations of Example 7.9. In particular, we show that several bicategories of structured categories have pullbacks (Example 3.18).
- In particular, we give more detail on the specific comprehension bicategory in which Licata and Harper's interpretation [28] of their directed type theory takes place. We also present a formalization of that example.
- We introduce display map bicategories and show that any display map bicategory gives rise to a comprehension bicategory.

- We present the type theory BTT in more detail, and specify more of its type-theoretic rules.
- We give more detail in the interpretation of the rules of BTT in any comprehension bicategory.

2 RELATED WORK

In this section, we review work with a similar goal to ours, as well as work we rely on. We pay particular attention to the desiderata outlined in Section 1.2 and to the difference between judgmental and typal dimensions.

2.1 Non-dependent type theories

For the sake of completeness, we include in this section pointers to work on *simple* type theories with reductions. The following works satisfy a non-dependent variant of D1, together with suitable adaptations of D2 and D3. However, due to the absence of type dependency, they are difficult to compare to our work.

Seely's paper [36] presents a syntax for a two-dimensional simply-typed lambda calculus, consisting of types, terms, and reductions between terms. They then construct a 2-category out of that syntax. Tabareau [43] frames aspect-oriented programming in a 2-categorical way, developing a lambda calculus that provides an internal language for 2-categories. Hirschowitz [23] constructs a 2-adjunction between 2-signatures for lambda calculi (where such signatures specify types, terms, and reductions) and the category of Cartesian closed 2-categories. Fiore and Saville [18] construct an internal language for cartesian closed bicategories; the result is a class (parametrized by a notion of signature for constants) of *simple* 2-dimensional type theories or lambda calculi. This last work shares one aspect with ours that the others do not: it uses (weak) *bi*categorical structure, rather than (strict) *2*-categorical structure.

2.2 Theories for Higher Categories

There is a body of work on designing type theories for ω-groupoids and ω-categories. In these type theories, one works, semantically speaking, *within one fixed* ∞-groupoid (or ω-category). Compare this to, *e. g.*, , Martin-Löf type theory, where one manipulates ∞-groupoids (types and identity types) and ∞-functors (functions) between them. Analogously, in our type theory, each type can be thought of as a category. Despite these different goals, we mention some of the work in this area.

Brunerie [10] constructs a type theory whose models are weak ∞-groupoids. Benjamin et al. [8] (see also [16]) design a type theory whose models are precisely ω-categories à la Grothendieck–Maltsiniotis. In [17], the authors study meta-theoretic properties of a language for strictly unital ∞-categories. There are also computer tools implementing such type theories, see, *e. g.*, [6, 34].

2.3 Theories with Dependent Types

In this section we review work on higher-dimensional and directed type theory with dependent types. We start with a review of work on *un*directed type theory.

2.3.1 Undirected Type Theory. The idea of considering higher-dimensional interpretations of type theory was born with Hofmann and Streicher's groupoid interpretation of Martin-Löf type theory [24]. This interpretation was generalized to stacks (poset-indexed groupoids satisfying a sheaf condition) in order to prove the independence of several logical principles by Coquand, Mannaa, and Ruch [13]. It was furthermore generalized, from different angles, to higher dimensions, see, *e. g.*, [9, 26, 31]. Common to all of this work is the restriction to (higher) *groupoids*.

In [29], Licata and Harper developed a two-dimensional dependent type theory with a judgment for *equivalences* $\Gamma \vdash \alpha : M \simeq_A N$ between terms $M, N : A$. These equivalences are postulated to have (strict) inverses. The authors give an interpretation of types as groupoids: terms are (interpreted as) objects in the interpreting groupoid, and equivalences are morphisms, necessarily invertible. No general notion of semantic structure is discussed; this work hence satisfies an *un*directed version of D1, but not D2.

Coraglia and Di Liberti [14] introduce judgemental theories and calculi for them as a general framework to present and study deductive systems. They instantiate their framework to obtain a type theory that describes an internal language of 2-toposes (see [14, Section 5.3]); however, they only consider a one-dimensional type theory à la Martin-Löf, and thus their work does not satisfy D1.

Garner [19] studies a typal two-dimensional type theory à la Martin-Löf: the forms of judgment are the same as in Martin-Löf type theory. Garner calls a type X "discrete" if it satisfies identity reflection (that is, if any identity $p : x = y$ between elements $x, y : X$ induces a judgmental equality $x \equiv y$. They then add rules that turn any identity type into a discrete type, effectively "truncating" intensional Martin-Löf type theory at 1-types (even though in principle, the identity type can of course be iterated any number of times). Garner defines a notion of two-dimensional model based on (strict) *comprehension 2-categories*. Exploiting the restriction to 1-truncated types, they then give a sound and complete interpretation of their two-dimensional type theory in any model. Identity types are automatically "symmetric", i.e., any identity admits an inverse; correspondingly, Garner defines their comprehension 2-categories to consist of *locally groupoidal* 2-categories. Thus, Garner's work satisfies D1 for *undirected* reductions, using the identity type for this purpose. Garner also considers type constructors such as dependent pair types and dependent product types, thus satisfying D4 and D5 in this case.[1]

2.3.2 Directed Type Theory. Licata and Harper [28] (see also [27, Chapter 7]) also designed a *directed* two-dimensional type theory and gave an interpretation for it in the strict 2-category of categories. Their syntax has a judgment for *substitutions* between contexts, written $\Gamma \vdash \theta : \Delta$, and *transformations* between parallel substitutions. An important aspect of their work is *variance* of contexts/types, built into the judgments. The type formers there have a certain variance—*co*variance or *contra*variance—in each of the arguments. They do not define a general notion of model for their theory; this work hence satisfies D1, but not D2.

Nuyts [33, Section 1.3.1] observes that the type theory developed by Licata and Harper [28] does not allow for a non-trivial Martin-Löf identity type—any such type would coincide with the directed transformations. Nuyts thus attempts to generalize the treatment of variance by Licata and Harper, and designs a directed type theory with additional variances, such as isovariance and invariance. Nuyts does not provide any interpretation of their syntax, and thus no proof of (relative) consistency; the work hence does not satisfy D2.

North [32] develops a type former for *directed* types of morphisms, resulting in a typal higher-dimensional directed type theory based on the judgments of MLTT. North's work thus does not satisfy D1. The model given by North is in the 2-category of categories, similar to Licata and Harper's [28].

Shulman, in unfinished work [39], aims to develop 2-categorical logic, including a two-dimensional notion of topos and a suitable internal language for such toposes. Specifically, Shulman sketches two internal languages for 2-toposes. The first language [38] is undirected, consisting only of types and terms. The second language [37] is only described in a short sketch; it is a kind of directed type theory featuring, in particular, variances. Our work is similar to Shulman's in the sense that both start from a (bi)categorical notion and extract a language from it, with the goal of developing a precise correspondence between extensions of the syntax and additional structure on the semantics. Unfortunately, Shulman's work is far from finished, which makes a more complete evaluation difficult. However, it contains several ideas that have influenced the present work. For instance, Shulman [40] emphasizes the usefulness of restricting to (op)fibrations instead of considering all 1-cells when constructing bicategories of arrows—we do this in our main examples of comprehension bicategory, Examples 5.4 and 7.9.

Riehl and Shulman [35] design a *simplicial* type theory (STT) featuring a directed interval type, as a synthetic theory of $(\infty, 1)$-categories. As a notion of model, they introduce "comprehension categories with shapes" [35,

[1]Garner also relies on Hermida's [22] slightly incomplete definition of fibration of 2-categories; see Buckley's work [12, Remark 2.1.9] for details). We have not checked if Garner's work extends to Buckley's corrected definition of 2-fibration.

Def. A.5]. These are (1-categorical) towers of fibrations accounting for several layers of contexts. Further work on STT was done, among others, by [47] and [11]. STT is not higher-dimensional in the sense of [28] or the present work; in particular, reductions, both in the tope layer and in the type layer, are symmetric. This work thus does not satisfy D1.

Summary. In the present work, we define a bicategorical notion of "model" for the interpretation of types, terms, and reductions, and derive from it a system of inference rules and an interpretation of those rules in any model; our work thus satisies D1, D2, and D3. We do not handle D4 and D5 in this work.

Among the described related work, our work is closest to work by Licata and Harper [28] and Garner [19]. Compared to [28], we add a general definition of "model" of a *directed* two-dimensional type theory, and provide many examples of models. Compared to [19], we cover *directed* reductions, and provide many instances of our general definition of model. Compared to both works, we do *not* handle type and term formers.

3 PRELIMINARIES

Here, we sketch some definitions used later on. Many would be very long if given in full; instead, we try to convey some intuition and give pointers to the precise definitions. As a reference for bicategory theory, see Bénabou's article [7]. We use here the vocabulary and notation introduced in [2].

Definition 3.1 (bicat). A **bicategory** consists of a type B_0 of 0-cells (or objects), a type $a \longrightarrow b$ of *1-cells from a to b* for every $a, b : \mathsf{B}_0$, and a *set $f \Rightarrow g$* of *2-cells from f to g* for every $a, b : \mathsf{B}_0$ and $f, g : a \longrightarrow b$. We have identity $\mathrm{id}_1(a) : a \longrightarrow a$ and composition of 1-cells $f \cdot g : a \longrightarrow c$ (also written fg), which we write in diagrammatical order. These operations do *not* satisfy the axioms for a 1-category. Instead, we have, for instance, the *left unitors*, that is, invertible 2-cells $\ell_f : \mathrm{id}_1(a) \cdot f \Rightarrow f$ for any 1-cell f, and similarly right unitors $r_f : f \cdot \mathrm{id}_1(b) \Rightarrow f$. Analogously, we have the *associators*, a family of invertible 2-cells $\alpha(f, g, h) : f \cdot (g \cdot h) \longrightarrow (f \cdot g) \cdot h$. For 2-cells $\theta : f \Rightarrow g$ and $\tau : g \Rightarrow h$ (where $f, g, h : a \longrightarrow b$ for some $a, b : \mathsf{B}_0$), we have a *vertical composition* $\theta \bullet \tau : f \Rightarrow h$. For any 1-cell $f : a \longrightarrow b$, we have an *identity 2-cell* $\mathrm{id}_2(f) : f \Rightarrow f$, which is neutral with respect to vertical composition: $\mathrm{id}_2(f) \bullet \theta = \theta$. For any two objects a and b, the 1-cells from a to b, and 2-cells between them, form the objects and morphisms of the hom-category $\underline{\mathsf{B}(a, b)}$, with composition given by vertical composition of B. We also have *left* and *right whiskering*; given a 2-cell $\theta : f \Rightarrow g : b \longrightarrow c$ and a 1-cell $e : a \longrightarrow b$, we have the *left whiskering* $e \triangleleft \theta : e \cdot f \Rightarrow e \cdot g$, and, similarly, the *right whiskering* $\theta \triangleright h : f \cdot h \Rightarrow g \cdot h$ for $h : c \longrightarrow d$. We do not list the axioms that these operations satisfy; the interested reader can consult, *e. g.,* [1, Def. 2.1].

We occasionally write 1_a for $\mathrm{id}_1(a)$ and 1_f for $\mathrm{id}_2(f)$.

We denote by Cat the bicategory of categories, and by Grpd the bicategory of groupoids. The bicategory B^{op} has the same objects as B, but 1-cells from x to y in B^{op} are 1-cells $f : y \longrightarrow x$ in B. The 2-cells in B^{op} are 2-cells in B. The bicategory B^{co} has the same objects and 1-cells as B, but 2-cells from f to g in B^{co} are the same as 2-cells from g to f in B.

Definition 3.2 (psfunctor). Given two bicategories B and B′, a **pseudofunctor** $F : \mathsf{B} \longrightarrow \mathsf{B}'$ is given by maps $F_0 : \mathsf{B}_0 \longrightarrow \mathsf{B}'_0$, $F_1 : (a \longrightarrow b) \longrightarrow (F_0 a \longrightarrow F_0 b)$,[2] and $p_2 : (f \Rightarrow g) \longrightarrow (F_1 f \Rightarrow F_1 g)$, preserving structure on 1-cells up to invertible 2-cells in B′ (specified as part of the functor F) and preserving structure on 2-cells up to equality.

We build complicated bicategories from simpler ones by adding structure at all dimensions. The additional structure should come with its own composition and identity, which should lie suitably over composition and identity of the original bicategory. This idea is formalized in the notion of *displayed bicategory*—a layer of data over a base bicategory—and the resulting *total bicategory*—the bicategory of pairs $(b, \overline{b})$ of a cell b in the base and

[2]Note that $\longrightarrow$ is used for both 1-cells and function types.

a cell $\overline{b}$ "over" b. We also obtain a pseudofunctor from the total bicategory into the base, given at all dimensions by the first projection.

Definition 3.3 ([1, Def. 4.1], disp_bicat)**.** Let B be a bicategory. A **displayed bicategory** D **over** B consists of

(1) for any $b : \mathsf{B}_0$, a type D_b of **objects over** b;
(2) for any $f : a \to b$ and $x : \mathsf{D}_a$ and $y : \mathsf{D}_b$, a type $x \xrightarrow{f} y$ of **1-cells over** f;
(3) for any $\theta : f \Rightarrow g$ and $\overline{f} : x \xrightarrow{f} y$ and $\overline{g} : x \xrightarrow{g} y$, a *set* $\overline{f} \overset{\theta}{\Rightarrow} \overline{g}$ of **2-cells over** θ;

together with suitably typed composition (over composition in B) and identity (over identity in B) for both 1- and 2-cells. These operations are subject to "axioms over axioms in B".

Definition 3.4 ([1, Def. 4.2], total_bicat)**.** Given a displayed bicategory D over B, we define the **total bicategory** $\int\mathsf{D}$ to have, as cells at dimension i, pairs $(b, \overline{b})$ where b is a cell of B at dimension i and $\overline{b}$ is a cell of D over b, with the obvious source and target.

We define the **projection** pseudofunctor $\pi : \int\mathsf{D} \to \mathsf{B}$ to be given, on any cell, by $(b, \overline{b}) \mapsto b$.

Remark 3.5. Note that all (displayed) (bi)catgories are assumed to be univalent. We do not repeat the definition here, but point instead to [2, Defs. 3.1, 7.3]. Intuitively, univalence means that adjoint equivalence of 0-cells, and isomorphism of 1-cells, coincides with identity between them, respectively. Working with univalent bicategories allows us to simplify some definitions, see Remark 4.5.

Definition 3.6 (disp_subbicat)**.** Suppose that we have a bicategory B, a predicate P_0 on the 0-cells (by which we mean a proposition $P_0(a)$ for every 0-cell a), and a predicate P_1 on the 1-cells. Furthermore, we assume that P_1 is closed under identity and composition; that is, $P_1(\mathrm{id}(x))$ holds for every x satisfying P_0 and that for all $f : x \to y$ and $g : y \to z$ between 0-cells satisfying P_0 we have $P_1(f \cdot g)$ if we have both $P_1(f)$ and $P_1(g)$. Then we define a displayed bicategory $\mathsf{SubBicat}(P_0, P_1)$ on B such that

- the type of displayed objects over x is $P_0(x)$;
- the type of displayed 1-cells over $f : x \to y$ is $P_1(f)$; and
- the type of displayed 2-cells over $\theta : f \Rightarrow g$ is the unit type.

The total bicategory of this bicategory selects 0-cells and 1-cells from the original bicategory B. Its objects are objects $x : \mathsf{B}$ such that $P_0(x)$, its 1-cells are 1-cells $f : x \to y$ in B such that $P_1(f)$, and its 2-cells are the same as 2-cells $\tau : f \Rightarrow g$ in B. The projection pseudofunctor $\pi : \mathsf{SubBicat}(P_0, P_1) \to \mathsf{B}$ is the inclusion. We can instantiate this examples to define bicategories of categories with a certain structure.

Example 3.7 (StructuredCategories.v)**.** We define the following displayed bicategories over Cat:

- $\mathsf{D}_{\mathsf{Terminal}}$ where $P_0(\mathsf{C})$ says that C has a terminal object and $P_1(F)$ says that F preserves terminal objects. We denote its total bicategory by $\mathsf{Cat}_{\mathsf{Terminal}}$.
- $\mathsf{D}_{\mathsf{BinProd}}$ where $P_0(\mathsf{C})$ says that C has binary products and $P_1(F)$ says that F preserves binary products. We denote its total bicategory by $\mathsf{Cat}_{\mathsf{BinProd}}$.
- $\mathsf{D}_{\mathsf{Pullback}}$ where $P_0(\mathsf{C})$ says that C has pullbacks and $P_1(F)$ says that F preserves pullbacks. We denote its total bicategory by $\mathsf{Cat}_{\mathsf{Pullback}}$.
- $\mathsf{D}_{\mathsf{FinLim}}$ where $P_0(\mathsf{C})$ says that C has finite limits and $P_1(F)$ says that F preserves finite limits. We denote its total bicategory by $\mathsf{Cat}_{\mathsf{FinLim}}$.
- $\mathsf{D}_{\mathsf{Initial}}$ where $P_0(\mathsf{C})$ says that C has an initial object and $P_1(F)$ says that F preserves initial objects. We denote its total bicategory by $\mathsf{Cat}_{\mathsf{Initial}}$.
- $\mathsf{D}_{\mathsf{BinSum}}$ where $P_0(\mathsf{C})$ says that C has binary coproducts and $P_1(F)$ says that F preserves binary coproducts. We denote its total bicategory by $\mathsf{Cat}_{\mathsf{BinSum}}$.

Note that in all of these examples, we crucially use that (co)limits are unique if they exist; otherwise P_0 would not consist of fiberwise propositions.

The total bicategories of the displayed bicategories in Example 3.7 are bicategories of categories with certain (co)limits. In addition, we can combine the structure of these by taking the product of the suitable displayed bicategories. Note that we could use similar methods to construct bicategories of extensive categories, regular categories, exact categories, and (pre)toposes.

The following (displayed) bicategories are used:

Example 3.8 (trivial_displayed_bicat)**.** Given bicategories B_1 and B_2, we define a displayed bicategory ${\mathsf{B}_1}^{+\mathsf{B}_2}$ over B_1 as follows:

- The displayed 0-cells over $x : \mathsf{B}_1$ are 0-cells $y : \mathsf{B}_2$.
- The displayed 1-cells over $f : x_1 \to x_2$ from $y_1 : \mathsf{B}_2$ to $y_2 : \mathsf{B}_2$ are 1-cells $g : y_1 \to y_2$ in B_2.
- The displayed 2-cells over $\theta : f \Rightarrow g$ from $g_1 : y_1 \to y_2$ to $g_2 : y_1 \to y_2$ are 2-cells $\tau : g_1 \Rightarrow g_2$ in B_2.

The total bicategory is $\int {\mathsf{B}_1}^{+\mathsf{B}_2} = \mathsf{B}_1 \times \mathsf{B}_2$ with projection $\pi : \mathsf{B}_1 \times \mathsf{B}_2 \to \mathsf{B}_1$.

Example 3.9 (cod_disp_bicat)**.** Let B be a bicategory. Define a displayed bicategory $\mathsf{B}^{\downarrow}$ over B as follows:

- The displayed objects over $y : \mathsf{B}$ are 1-cells $x \to y$.
- The displayed 1-cells over $g : y_1 \to y_2$ from $h_1 : x_1 \to y_1$ to $h_2 : x_2 \to y_2$ are pairs consisting of a 1-cell $f : x_1 \to x_2$ and an invertible 2-cell $\gamma : g \cdot h_2 \Rightarrow h_1 \cdot f$.
- Given displayed 1-cells $f_1 : x_1 \to x_2$ with $\gamma_1 : g_1 \cdot h_2 \Rightarrow h_1 \cdot f_1$, and $f_2 : x_1 \to x_2$ with $\gamma_2 : g_2 \cdot h_2 \Rightarrow h_1 \cdot f_2$, we define the displayed 2-cells over $\theta : g_1 \Rightarrow g_2$ from (f_1, γ_1) to (f_2, γ_2) as 2-cells $\tau : f_1 \Rightarrow f_2$ such that $\gamma_1 \bullet (h_1 \lhd \tau) = (\theta \rhd h_2) \bullet \gamma_2$.

The generated total bicategory is the *arrow bicategory*, $\int \mathsf{B}^{\downarrow} = \mathsf{B}^{\rightarrow}$ with projection given by the codomain, cod : $\mathsf{B}^{\rightarrow} \to \mathsf{B}$.

In the next example, we write StrictCat for the bicategory of strict categories and $\underline{\mathsf{StrictCat}}$ for the category of strict categories.

Example 3.10 (disp_bicat_of_functors_into_cat)**.** We define a displayed bicategory SplitOpCleav over StrictCat as follows:

- The displayed objects over C : StrictCat are functors $G : \mathsf{C} \to \underline{\mathsf{StrictCat}}$.
- The displayed 1-cells over $F : \mathsf{C}_1 \to \mathsf{C}_2$ from $G_1 : \mathsf{C}_1 \to \underline{\mathsf{StrictCat}}$ to $G_2 : \mathsf{C}_2 \to \underline{\mathsf{StrictCat}}$ are natural transformations $\gamma : G_1 \Rightarrow F \cdot G_2$.
- The displayed 2-cells over $n : F_1 \Rightarrow F_2$ from $\gamma_1 : G_1 \Rightarrow F_1 \cdot G_2$ to $\gamma_2 : G_1 \Rightarrow F_2 \cdot G_2$ are proofs that for all $x : \mathsf{C}$ we have $\gamma_1(x) \cdot G_2(n(x)) = \gamma_2(x)$.

The associated projection pseudofunctor $\pi : \int \mathsf{SplitOpCleav} \to \mathsf{StrictCat}$ maps functors $\mathsf{C} \to \underline{\mathsf{StrictCat}}$ to their domain.

Example 3.11 (disp_bicat_of_opcleaving)**.** We define a displayed bicategory OpCleav over Cat as follows:

- The displayed objects over C : Cat are displayed categories D over C together with an opcleaving.
- The displayed 1-cells over $F : \mathsf{C}_1 \to \mathsf{C}_2$ from D_1 to D_2 are displayed functors $\overline{F}$ from D_1 to D_2 that preserve opcartesian morphisms.
- The displayed 2-cells over $\theta : F \Rightarrow G$ from $\overline{F} : \mathsf{D}_1 \xrightarrow{F} \mathsf{D}_2$ to $\overline{G} : \mathsf{D}_1 \xrightarrow{G} \mathsf{D}_2$ are displayed natural transformations from $\overline{F}$ to $\overline{G}$ over θ.

The associated projection pseudofunctor $\pi : \int \mathsf{OpCleav} \to \mathsf{Cat}$ maps any opcleaving to its codomain category.

Similarly, we define displayed bicategories Cleav and IsoFib of cleavings and isocleavings, respectively. Note that Example 3.11 generalizes Example 3.10: while the former contains all opfibrations, the latter only contains the ones that are split.

The idea of displayed (bi)categories transfers to functors:

Definition 3.12 ([2, Def. 8.2], disp_psfunctor)**.** Given $F : \mathsf{B} \longrightarrow \mathsf{B}'$ and D and D' displayed bicategories over B and B', respectively, a **displayed pseudofunctor** $\overline{F}$ over F is

- for all objects $x : \mathsf{B}$ and $\overline{x} : \mathsf{D}_x$ an object $\overline{F}(\overline{x}) : \mathsf{D'}_{F(x)}$;
- for all displayed morphisms $\overline{f} : \overline{x} \xrightarrow{f} \overline{y}$, a displayed 1-cell $\overline{F}(\overline{f}) : \overline{F}(\overline{x}) \xrightarrow{F(f)} \overline{F}(\overline{y})$;
- for all displayed 2-cells $\overline{\theta} : \overline{f} \overset{\theta}{\Rightarrow} \overline{g}$, a displayed 2-cell $\overline{F}(\overline{\theta}) : \overline{F}(\overline{f}) \xRightarrow{F(\theta)} \overline{F}(\overline{g})$.

We denote by $\int\overline{F} : \int\mathsf{D} \longrightarrow \int\mathsf{D}'$ the induced **total pseudofunctor**.

Remark 3.13. The square of pseudofunctors

$$\begin{tikzcd}
\int\mathsf{D} \arrow[r, "\int\overline{F}"] \arrow[d, "\pi_{\mathsf{D}}"'] & \int\mathsf{D}' \arrow[d, "\pi_{\mathsf{D}'}"] \\
\mathsf{B} \arrow[r, "F"] & \mathsf{B}'
\end{tikzcd}$$

induced by $\overline{F}$ over F commutes *up to judgmental equality.*

Furthermore, we need pullbacks and products in bicategories.

Definition 3.14 (has_pb)**.** Let B be a bicategory, and suppose we have two 1-cells $f : a \longrightarrow c$ and $g : b \longrightarrow c$. A **pullback structure for** f **and** g on an object $x : \mathsf{B}$ together with two 1-cells $\pi_1 : x \longrightarrow a$ and $\pi_2 : x \longrightarrow b$ and an invertible 2-cell $\gamma : p \cdot f \Rightarrow q \cdot g$ is given by the following data:

- for all 1-cells $p' : z \longrightarrow a$ and $q' : z \longrightarrow b$ and invertible 2-cells $\gamma' : p' \cdot f \Rightarrow q' \cdot g$, we have a 1-cell $u : z \longrightarrow x$ together with invertible 2-cells $\theta : u \cdot p \Rightarrow p'$ and $\tau : u \cdot q \Rightarrow q'$ such that
$$\alpha \bullet \theta \rhd f \bullet \gamma' = u \lhd \gamma \bullet \alpha^{-1} \bullet \tau \rhd g.$$
- for all 1-cells $u_1, u_2 : z \longrightarrow x$ and 2-cells $\theta : u_1 \bullet p \Rightarrow u_2 \bullet p$ and $\tau : u_1 \bullet q \Rightarrow u_2 \bullet q$ such that
$$u_1 \lhd \gamma \bullet \alpha \bullet \tau \rhd q \bullet \alpha^{-1} = \alpha \bullet \theta \rhd f \bullet \alpha^{-1} \bullet u_2 \lhd \gamma,$$
we have a unique 2-cell $\nu : u_1 \Rightarrow u_2$ such that $\nu \rhd p = \theta$ and $\nu \rhd q = \tau$.

In Definition 3.14 it is irrelevant if we postulate data to be given explicitly or to merely exist:

Proposition 3.15 (isaprop_has_pb_ump). *The type of pullback structures on* $(x, \pi_1, \pi_2, \gamma)$ *is a proposition.*

Remark 3.16. There are different notions of pullback in bicategories depending on whether $p \cdot f$ and $q \cdot g$ are postulated to be related up to an equality, invertible 2-cell or even just a 2-cell. In Definition 3.14, the square commutes up to invertible 2-cell. One could also define *strict pullbacks*: this is done similarly to Definition 3.14, but all involved squares must commute up to equality rather than just up to invertible 2-cell.

Example 3.17 (one_types_has_pb, has_pb_bicat_of_univ_cats)**.** The bicategory of groupoids has pullbacks.

The bicategory Cat also has pullbacks, and they are given by iso-comma categories. These are defined as follows: given categories C_1, C_2, and C_3, and functors $F : \mathsf{C}_1 \longrightarrow \mathsf{C}_3$ and $G : \mathsf{C}_2 \longrightarrow \mathsf{C}_3$, we define the **iso-comma category** $F /_{\simeq} G$

- Its objects consist of objects $x : \mathsf{C}_1$ and $y : \mathsf{C}_2$ together with an isomorphism $f : F(x) \longrightarrow G(y)$
- The morphisms from (x_1, y_1, f_1) to (x_2, y_2, f_2) consists of morphisms $g : x_1 \longrightarrow x_2$ and $h : y_1 \longrightarrow y_2$ such that the following diagram commutes:

$$\begin{tikzcd}
F(x_1) \arrow[r, "f_1"] \arrow[d, "F(g)"'] & G(y_1) \arrow[d, "G(h)"] \\
F(x_2) \arrow[r, "f_2"'] & G(y_2)
\end{tikzcd}$$

We define functors $\pi_1^{\simeq} : F /_{\simeq} G \longrightarrow \mathsf{C}_1$ and $\pi_2^{\simeq} : F /_{\simeq} G \longrightarrow \mathsf{C}_2$ and a natural isomorphism $n^{\simeq} : \pi_1^{\simeq} \cdot F \Rightarrow \pi_2^{\simeq} \cdot G$. The category $F /_{\simeq} G$ together with the functors $\pi_1^{\simeq}, \pi_2^{\simeq}$ and natural isomorphism $n^{\simeq}$ is universal among such data, making it the pullback of F and G.

Example 3.18 (LimitsStructuredCategories.v)**.** The bicategories defined in Example 3.7 all have pullbacks as well. This is because taking the iso-comma category preserves the desired (co)limits and the projections and pairing functors preserve them.

As a special case of pullbacks in the presence of terminal objects, we can define products in bicategories (has_binprod_ump). If B has chosen products, we write $x \times y$ for the product of x and y, and we denote the projections by $\pi_1 : x \times y \longrightarrow x$ and $\pi_2 : x \times y \longrightarrow y$.

Notation 3.19. In the following, given a fibration, we notate the pullback (or reindexing) of A along f by f^*A. Given an opfibration, we notate the pushforward of A along f by f_*A.

4 COMPREHENSION BICATEGORIES

In this section, we define the notion of global cleaving and local (op)cleavings of bicategories. Afterwards, we use these notions to define comprehension bicategories. We are guided by Buckley's paper [12], where local and global fibrations are defined, and we add definitions for local cloven iso- and opfibrations. However, there is an important difference: while Buckley works in a set-theoretic setting, we reformulate the definitions in a type-theoretic setting using the displayed technology developed in [2] and reviewed in Section 3—see also Remark 4.5.

Throughout this section, we assume that B is a bicategory and D is a displayed bicategory over B.

Definition 4.1 ([12, Def. 3.1.1], cartesian_1cell)**.** Let $f : a \longrightarrow b$ be a 1-cell in B, and let $\overline{f} : \overline{a} \xrightarrow{f} \overline{b}$ be a displayed 1-cell over f in D. A **cartesian structure** on $\overline{f}$ consists of the following:

(1) For any $\overline{g} : \overline{c} \xrightarrow{h \cdot f} \overline{b}$, a choice of a displayed morphism $\overline{h} : \overline{c} \xrightarrow{h} \overline{a}$ and a displayed isomorphism θ over the identity isomorphism on $h \cdot f$ in B.

$$\begin{tikzcd}
\overline{c} \arrow[d, dashed, "\overline{h}"'] \arrow[dr, "\overline{g}"] & \\
\overline{a} \arrow[r, "\overline{f}"'] \arrow[ur, phantom, "\overset{\theta}{\Rightarrow}"] & \overline{b}
\end{tikzcd}$$

$$\begin{tikzcd}
c \arrow[d, "h"'] \arrow[dr, "h \cdot f"] & \\
a \arrow[r, "f"'] \arrow[ur, phantom, "\overset{\mathrm{id}_2}{\Rightarrow}"] & b
\end{tikzcd}$$

We call $(\overline{h}, \theta)$ the **lift** of $(h, \overline{g})$.

(2) Given lifts $(\overline{h}_1, \theta_1)$ and $(\overline{h}_2, \theta_2)$ of $(h_1, \overline{g}_1)$ and $(h_2, \overline{g}_2)$, respectively, and $\delta : h_1 \Rightarrow h_2$, and a 2-cell $\overline{\sigma} : \overline{g}_1 \Rightarrow \overline{g}_2$ over $\delta \triangleright f$, we have a unique 2-cell $\overline{\delta} : \overline{h}_1 \Rightarrow \overline{h}_2$ over δ such that $\overline{\delta} \triangleright \overline{f} \bullet \theta_2 = \theta_1 \bullet \overline{\sigma}$.

$$\begin{tikzcd}
\overline{c} \arrow[d, bend right=50, "\overline{h}_1"'] \arrow[d, bend left=50, "\overline{h}_2"] \arrow[d, phantom, "\overset{\overline{\delta}}{\Rightarrow}"] \arrow[dr, bend left=20, "\overline{g}_1"] \arrow[dr, bend left=50, "\overline{g}_2"] \arrow[dr, phantom, bend left=35, "\overset{\overline{\sigma}}{\Rightarrow}"] & \\
\overline{a} \arrow[r, "\overline{f}"'] & \overline{b}
\end{tikzcd}$$

$$\begin{tikzcd}
c \arrow[d, bend right=50, "h_1"'] \arrow[d, bend left=50, "h_2"] \arrow[d, phantom, "\overset{\delta}{\Rightarrow}"] & \\
a \arrow[r, "f"'] & b
\end{tikzcd}$$

Given a cartesian structure on $\overline{f}$, we call $\mathsf{F}_1(\overline{f}, \overline{g}) := \overline{h}$ and $\mathsf{F}_2(\overline{f}, \overline{g}) := \theta$. We also call $\mathsf{E}(\delta, \overline{\sigma}) := \overline{\delta}$.

Proposition 4.2. *We have the following cartesian 1-cells:*

- *(cartesian_1cell_id) The identity $\overline{\mathrm{id}_1} : \overline{a} \longrightarrow \overline{a}$ is cartesian.*
- *(comp_cartesian_1cell) If $\overline{f} : \overline{a} \longrightarrow \overline{b}$ and $\overline{g} : \overline{b} \longrightarrow \overline{c}$ are cartesian, then so is $\overline{f} \cdot \overline{g}$.*

Problem 4.3. Given two cartesian 1-cells $\overline{f} : \overline{a} \longrightarrow \overline{b}$ and $\overline{f}' : \overline{a}' \longrightarrow \overline{b}$, to construct an adjoint equivalence $e : \overline{a} \longrightarrow \overline{a}'$ over the identity and an invertible 2-cell from $\overline{f}$ to $e \cdot \overline{f}'$.

Construction 4.4 (for Problem 4.3; EquivalenceBetweenCartesian.v)**.** The 1-cells of the adjoint equivalence are constructed as cartesian factorizations (first item of Definition 4.1). In that way, we also obtain the desired invertible 2-cell. □

Remark 4.5. Recall that a displayed 1-cell $\overline{f} : \overline{a} \xrightarrow{f} \overline{b}$ in D gives rise to the 1-cell $(f, \overline{f})$ in the total bicategory $\int$D. The definition of cartesian structure on $\overline{f}$ in D of Definition 4.1 gives rise to a notion of cartesian structure for $(f, \overline{f})$ in $\int$D. By expressing the definition of cartesian 1-cell in the displayed bicategory (instead of in the resulting projection $\pi : \int \mathsf{D} \to \mathsf{B}$), we can postulate that a lift in $\int$D lies *definitionally* over a given cell in B, not just modulo an invertible 2-cell. In univalent bicategories, these two formulations are equivalent.

Definition 4.6 (global_cleaving)**.** A **global cleaving on** D is a choice, for any $f : a \to b$ in B and $\overline{b} : \mathsf{D}_b$, of

(1) a displayed object $\overline{a}$ over a;
(2) a displayed 1-cell $\overline{f} : \overline{a} \xrightarrow{f} \overline{b}$;
(3) a cartesian structure on $\overline{f}$.

Given a global cleaving on D, we use the notation $\overline{b}[f] := \overline{a}$ and $L_f(\overline{b}) := \overline{f}$ to denote the choice given by the cleaving.

Remark 4.7. The notion of global cleaving as in Definition 4.6 gives rise to a notion of cloven fibration on the total bicategory $\int$D.

Next we look at local cleavings and opcleavings. A 2-cell is opcartesian if and only if it is opcartesian in the 1-categorical sense for the hom-functor. However, in the formalization we give the following direct definition not relying on hom-categories, and prove the characterization via hom-categories afterwards (opcartesian_2cell_weq_opcartesian). Similarly, we give an unfolded definition of local opcleaving.

Definition 4.8 (is_opcartesian_2cell)**.** Suppose we have 1-cells $f, g : x \to y$, a 2-cell $\theta : f \Rightarrow g$, and displayed objects $\overline{x}$ and $\overline{y}$ over x and y, respectively. Given displayed 1-cells $\overline{f} : \overline{x} \xrightarrow{f} \overline{y}$ and $\overline{g} : \overline{x} \xrightarrow{g} \overline{y}$ and a displayed 2-cell $\overline{\theta} : \overline{f} \overset{\theta}{\Rightarrow} \overline{g}$, we say that $\overline{\theta}$ is **2-opcartesian** (or just **opcartesian**) if for all 1-cells $h : x \to y$, displayed 1-cells $\overline{h} : \overline{x} \xrightarrow{h} \overline{y}$, 2-cells $\tau : g \Rightarrow h$, and displayed 2-cells $\overline{\gamma} : \overline{f} \overset{\tau \bullet \theta}{\Longrightarrow} \overline{h}$, there is a unique displayed 2-cell $\overline{\tau} : \overline{g} \overset{\tau}{\Rightarrow} \overline{h}$ such that $\overline{\theta} \bullet \overline{\tau} = \overline{\gamma}$.

Being an opcartesian 2-cell is always a property. The notion of *cartesian 2-cells* is analogous.

Definition 4.9 (local_opcleaving)**.** A **local opcleaving on** D is given by, for every $\theta : f \Rightarrow g$ and $\overline{f} : \overline{x} \xrightarrow{f} \overline{y}$

(1) a displayed 1-cell $\overline{g} : \overline{x} \xrightarrow{g} \overline{y}$ and
(2) an opcartesian 2-cell $\overline{\theta} : \overline{f} \overset{\theta}{\Rightarrow} \overline{g}$.

The notions of **local cleaving** and **local isocleaving** are defined analogously.

Remark 4.10. Every displayed bicategory on a univalent bicategory has a local isocleaving. The construction is the same as for categories [4, Construction 5.12].

Remark 4.11. The notions of opcartesian 2-cell and of local opcleaving as in Definition 4.9 give rise to notions of opcartesian 2-cell and of cloven local opfibration on the total bicategory $\int$D.

Note that one can construct a local isocleaving from either a local cleaving or a local opcleaving.

Proposition 4.12 (CleavingOfBicatIsAProp.v). *Suppose that* B *is a univalent bicategory and* D *is a univalent displayed bicategory over* D*. Then the types of local (resp. global) (op)cleavings on* D *are propositions.*

In light of Proposition 4.12 and Remark 3.5, we do not distinguish between fibrations and cloven fibrations (cleavings) in the following. When we say that D “is a fibration”, we mean, in particular, that it is equipped with a choice of cleaving. All constructions of cleavings given here are explicit and constructive.

Now let us look at some examples of these notions.

Example 4.13 (TrivialCleaving.v)**.** The trivial displayed bicategory ${\mathsf{B}_1}^{+\mathsf{B}_2}$ over B_1 is a global fibration. Cartesian 1-cells in ${\mathsf{B}_1}^{+\mathsf{B}_2}$ correspond to adjoint equivalences in B_2. As such, we can take the identity 1-cell as the global lift. In addition, ${\mathsf{B}_1}^{+\mathsf{B}_2}$ is both a local fibration and a local opfibration.

Example 4.14 (CodomainCleaving.v)**.** Suppose that B is a locally groupoidal bicategory with pullbacks. Since cartesian 1-cells in $\mathsf{B}^{\downarrow}$ correspond to pullback squares, we can construct a global cleaving for $\mathsf{B}^{\downarrow}$ by taking pullbacks. Note that all 2-cells in $\mathsf{B}^{\downarrow}$ are cartesian, because B is locally groupoidal, and thus $\mathsf{B}^{\downarrow}$ also has a local cleaving and a local opcleaving.

Example 4.15 (FunctorsIntoCatCleaving.v)**.** The displayed bicategory SplitOpCleav has a global cleaving and a local opcleaving. To construct these, we first observe that a displayed 1-cell $\gamma : G_1 \Rightarrow F_1 \cdot G_2$ from $G_1 : \mathsf{C}_1 \to \underline{\mathsf{StrictCat}}$ to $G_2 : \mathsf{C}_2 \to \underline{\mathsf{StrictCat}}$ is cartesian if and only if it is a natural isomorphism. Now we construct a global cleaving for SplitOpCleav as follows: whenever we have functors $F : \mathsf{C}_1 \to \mathsf{C}_2$ and $G : \mathsf{C}_2 \to \underline{\mathsf{StrictCat}}$, then we also get a functor $F \cdot G : \mathsf{C}_1 \to \underline{\mathsf{StrictCat}}$.

Every displayed 2-cell in SplitOpCleav is opcartesian. To find local opcartesian lifts, suppose that we have functors $F_1, F_2 : \mathsf{C}_1 \to \mathsf{C}_2$, $G_1 : \mathsf{C}_1 \to \underline{\mathsf{StrictCat}}$ and $G_2 : \mathsf{C}_2 \to \underline{\mathsf{StrictCat}}$, and natural transformations $n : F_1 \Rightarrow F_2$ and $\gamma : G_1 \Rightarrow F_1 \cdot G_2$. Our goal is to construct a natural transformation $G_1 \Rightarrow F_2 \cdot G_2$, and for the desired transformation we take $\gamma \bullet (n \triangleright G_2)$. Hence, SplitOpCleav has a local opcleaving.

Example 4.16 (OpFibrationCleaving.v)**.** The displayed bicategory OpCleav has a global cleaving and a local opcleaving. Given a functor $F : \mathsf{C}_1 \to \mathsf{C}_2$ and a displayed category D_2 over C_2, we construct a displayed category $F^*(\mathsf{D}_2)$ over C_1:

- The displayed objects over $x : \mathsf{C}_1$ are displayed objects in D_2 over $F(x)$.
- The displayed morphisms over $f : x \to y$ from $\overline{x}$ to $\overline{y}$ are displayed morphisms over $\overline{x} \xrightarrow{F(f)} \overline{y}$.

Note that $F^*(\mathsf{D}_2)$ inherits any opcleaving from D_2. In addition, we have a displayed functor over F from $F^*(\mathsf{D}_2)$ to D_2 that preserves cartesian morphisms.

Opcartesian 2-cells in OpCleav correspond to displayed natural transformations of which all components are opcartesian. We form local lifts pointwise. Since displayed 1-cells in OpCleav preserve cartesian morphisms, cartesian 2-cells are preserved under both left and right whiskering.

Similarly, we can show that the displayed bicategory Cleav has a global and a local cleaving (FibrationCleaving.v). We finish this section by defining comprehension bicategories. Examples of those are given in Section 5.

Definition 4.17 (comprehension_bicat)**.** A **comprehension bicategory** is given by a bicategory B, a displayed bicategory D over B, and a displayed pseudofunctor χ over the identity on B as pictured below,

$$\begin{tikzcd} \mathsf{D} \arrow[rr, "\chi"] & & \mathsf{B}^{\downarrow} \\ & \mathsf{B} & \end{tikzcd}$$

satisfying the following properties (see also Proposition 4.12):

(1) D is a global fibration;
(2) χ preserves cartesian 1-cells;

(3) D is a local opfibration;
(4) opcartesian 2-cells of D are preserved under both left and right whiskering;
(5) χ preserves opcartesian 2-cells.

Remark 4.18. Recall from Remarks 3.13, 4.7 and 4.11 that the displayed pseudofunctor $\chi : \mathsf{D} \to \mathsf{B}^{\downarrow}$ of Definition 4.17 gives rise to a *strictly* commuting diagram of pseudofunctors

$$\begin{tikzcd}
\int \mathsf{D} \arrow[rr, "\int \chi"] \arrow[dr] & & \int \mathsf{B}^{\downarrow} = \mathsf{B}^{\to} \arrow[dl, "\mathrm{cod}"] \\
& \mathsf{B} &
\end{tikzcd}$$

with suitable "classical" structure of global fibration, local opfibration, and preservation of (op)cartesian cells.

Remark 4.19. **Contravariant** and **isovariant** comprehension bicategories are defined analogously with the notions of (iso)cleaving and (iso)cartesian taking the place of opcleaving and opcartesian in Items 3 to 5 of Definition 4.17. Some of our examples of comprehension bicategories can similarly be equipped with a contravariant and isovariant comprehension structure.

5 EXAMPLES OF COMPREHENSION BICATEGORIES

In this section, we give several (classes of) instances of comprehension bicategories. In some of these instances, we recognize structures studied previously in the context of higher-dimensional and directed type theory.

Example 5.1 (trivial_comprehension_bicat)**.** Suppose we have a bicategory B with products. Then we have the following comprehension bicategory

$$\begin{tikzcd}
\mathsf{B}^{+\mathsf{B}} \arrow[rr, "\chi"] & & \mathsf{B}^{\downarrow} \\
& \mathsf{B} &
\end{tikzcd}$$

The displayed pseudofunctor $\chi : \mathsf{B}^{+\mathsf{B}} \to \mathsf{B}^{\downarrow}$ sends the object y_2 over y_1 to the product projection $y_1 \times y_2 \to y_1$, that is, the corresponding total pseudofunctor $\mathsf{B} \times \mathsf{B} \to B^{\downarrow}$ is defined by $(y_1, y_2) \mapsto \pi_1 : y_1 \times y_2 \to y_1$.

Example 5.1 corresponds roughly to the semantics studied by Fiore and Saville [18], but without the type formers.

Another example of a comprehension bicategory comes from locally groupoidal bicategories, such as Grpd.

Example 5.2 (locally_grpd_comprehension_bicat)**.** Let B be a locally groupoidal bicategory with pullbacks. Then we get the following comprehension bicategory

$$\begin{tikzcd}
\mathsf{B}^{\downarrow} \arrow[rr, "\mathrm{id}"] & & \mathsf{B}^{\downarrow} \\
& \mathsf{B} &
\end{tikzcd}$$

Since every 2-cell is opcartesian in $\mathsf{B}^{\downarrow}$ if B is locally groupoidal, the displayed bicategory $\mathsf{B}^{\downarrow}$ has a local opcleaving.

Example 5.2 models *symmetric* reductions between terms. It thus generalizes the groupoid model of type theory [24] and is related to the interpretation of the two-dimensional type theory by Licata and Harper [29], and to the definition of comprehension 2-category by Garner [19].

We can also consider *directed* versions of this example by using categories instead of groupoids.

Example 5.3 (functors_into_cat_comprehension_bicat)**.** We construct the following comprehension bicategory using split opfibrations

$$\mathsf{SplitOpCleav} \xrightarrow{\quad\chi\quad} \mathsf{StrictCat}^{\downarrow}$$

$$\mathsf{StrictCat}$$

To define χ, we use the Grothendieck construction. Specifically, from a functor $G : \mathsf{C} \to \underline{\mathsf{StrictCat}}$ we construct a category $\int G$ as follows

(1) The objects of $\int G$ are pairs (x, y) where $x : \mathsf{C}$ and $y : G(x)$.
(2) The morphisms from (x_1, y_1) to (x_2, y_2) are pairs (f, g) where $f : x_1 \to x_2$ and $g : G(f)(y_1) \to y_2$.

Note that we also have a projection $\pi_1 : \int G \to \mathsf{C}$. In addition, from a displayed 1-cell $\gamma : G_1 \Rightarrow F \cdot G_2$, we get a functor $\int \gamma : \int G_1 \to \int G_2$ and a natural isomorphism $\gamma_c : \pi_1 \cdot F \Rightarrow \int \gamma \cdot \pi_1$. If we have $p : \gamma_1(x) \cdot G_2(n(x)) = \gamma_2(x)$ for every $x : \mathsf{C}_1$, then we get a natural transformation $\int p$ from $\int \gamma_1$ to $\int \gamma_2$.

The pseudofunctor χ preserves cartesian 1-cells, which means that whenever we have a natural isomorphism $\gamma : G_1 \Rightarrow F \cdot G_2$, the following square is a pullback:

$$\begin{tikzcd}
\int G_1 \arrow[r, "\int \gamma"] \arrow[d, "\pi_1"'] & \int G_2 \arrow[d, "\pi_1"] \\
\mathsf{C}_1 \arrow[r, "F"'] & \mathsf{C}_2
\end{tikzcd}$$

The isomorphism witnessing the commutation is γ_c.

Furthermore, χ preserves opcartesian 2-cells. Specifically, suppose that we have functors $F_1, F_2 : \mathsf{C}_1 \to \mathsf{C}_2$, $G_1 : \mathsf{C}_1 \to \underline{\mathsf{StrictCat}}$ and $G_2 : \mathsf{C}_2 \to \underline{\mathsf{StrictCat}}$ and natural transformations $n : F_1 \Rightarrow F_2$, $\gamma_1 : G_1 \Rightarrow F_1 \cdot G_2$ and $\gamma_2 : G_1 \Rightarrow F_2 \cdot G_2$. In addition, we assume that we have $p(x) : \gamma_1(x) \cdot G_2(n(x)) = \gamma_2(x)$ for every $x : \mathsf{C}_1$. Preservation of opcartesian 2-cells by χ means that $(\int p)x$ is an opcartesian 2-cell for the functor $\pi_1 : \int G_2 \to \mathsf{C}_2$ for every x.

Example 5.3 corresponds to the interpretations given by Licata and Harper [28] and North [32] in their works on directed type theories, albeit without considering type formers. Contexts are categories and types over a context C are split opfibrations on C. We also consider a version of this interpretation where there are more types: instead of looking at only those opfibrations that are split, *all* opfibrations are considered.

Example 5.4 (opcleaving_comprehension_bicat)**.** From opfibrations we build the following comprehension bicategory:

$$\mathsf{OpCleav} \xrightarrow{\quad\chi\quad} \mathsf{Cat}^{\downarrow}$$

$$\mathsf{Cat}$$

The pseudofunctor χ sends a displayed category D over C to the functor $\pi_1 : \int \mathsf{D} \to \mathsf{C}$. To check whether χ preserves cartesian 1-cells, it suffices to check whether the chosen lifts are mapped to pullback squares. The chosen lifts are sent to strict pullback square because they are given by reindexing. Since strict pullbacks of isofibrations are weak pullbacks as well [25], we conclude that χ preserves cartesian 1-cells.

Similarly, we can define a *contravariant* comprehension bicategory (cleaving_contravariant_comprehension_bicat) using fibrations instead of opcleavings.

6 INTERNAL STREET (OP)FIBRATIONS

In this section, we discuss Street (op)fibrations internal to a fixed bicategory B. They will yield, in Section 7, many examples of comprehension bicategories, see Example 7.9 and Remark 7.11. The examples of Street opfibrations internal to bicategories of stacks are particularly interesting (see Remark 7.11).

Note that $B^{\downarrow}$ is a *global* fibration if B has pullbacks. However, to obtain a *local* (op)cleaving, we used that B is locally groupoidal in Example 4.14. This assumption is avoided in Example 4.16 where B = Cat: instead of looking at arbitrary functors, one only considers the opfibrations. We can generalize this idea to arbitrary bicategories by using *internal Street (op)fibrations* [12].

The displayed bicategory OpCleav has a local opcleaving where the desired lifts are constructed pointwise. To generalize this to arbitrary bicategories, we need to adjust this definition, so that we can lift arbitrary 2-cells. Furthermore, the notion of Grothendieck opfibration of categories is stricter than appropriate for bicategories. If we have $x \to y$ and an object over y, then a Grothendieck fibration gives an object *strictly* living over x, while a Street fibration only gives an object living *weakly* over x (*i. e.,* up to an isomorphism). More information can be found in work by Loregian and Riehl [30, Example 4.1.2].

Definition 6.1 (internal_sfib). Let $p : e \to b$ be a 1-cell in a bicategory B. Then p is an **internal Street fibration** if

- For every $x : B$, the functor $p_* : \underline{B(x, e)} \to \underline{B(x, b)}$ of hom-categories is a Street fibration.
- For every $f : x \to y$, we have a morphism of Street fibrations

$$\begin{tikzcd} \underline{B(y,e)} \arrow[r, "f^*"] \arrow[d, "p_*"'] & \underline{B(x,e)} \arrow[d, "p_*"] \\ \underline{B(y,b)} \arrow[r, "f^*"] & \underline{B(x,b)} \end{tikzcd}$$

A 1-cell is called an **internal Street opfibration** if it is an internal Street fibration in B^{co} (internal_sopfib).

In Cat, internal Street fibrations are the same as Street fibrations of categories. However, the notion of internal Street fibrations can be applied in a wider variety of settings: for example, one could also look at internal Street fibrations in the bicategories from Example 3.7 or in presheaves or stacks valued in Cat. A classical result on internal Street (op)fibrations is that they are closed under taking pullbacks [20, 41]:

Proposition 6.2 (pb_of_sfib_cleaving). *Street (op-)fibrations are closed under pullback. Concretely: given a pullback square*

$$\begin{tikzcd} e_1 \arrow[r] \arrow[d, "p_1"'] & e_2 \arrow[d, "p_2"] \\ b_1 \arrow[r] & b_2 \end{tikzcd}$$

where p_2 is a Street (op)fibration, then p_1 is so, too.

7 DISPLAY MAP BICATEGORIES

In this section, we present our definition of "display map bicategory" and how they give rise to comprehension bicategories.

Suppose that we have a bicategory B with pullbacks. Our goal is to show that we have a displayed bicategory SOpFib(B) over B that has both a global cleaving and a local opcleaving. We would like to construct this in a modular and general way. That is because using the same techniques we should be able to construct a displayed bicategory $\mathsf{SFib}(\mathsf{B})$ over B with a global and local cleaving. One can imagine even more examples: one could take the intersection of Street opfibrations with discrete or fully faithful 1-cells.

The common pattern among all these examples, is that the displayed bicategory actually represents a subbicategory of the arrow bicategory, In the 1-categorical case, such examples are captured by **display map categories** [44]. For that reason, we would like to adapt that notion to the bicategorical setting.

However, there is a difference between the 1-categorical and the bicategorical case. A display map category on C is a **full** subcategory of the arrow category on C satisfying some requirements. Such a notion would not be useful in the bicategorical case: neither $\mathsf{SOpFib}(\mathsf{B})$ nor $\mathsf{SFib}(\mathsf{B})$ are full subbicategories of $\mathsf{B}^{\downarrow}$, since the 1-cells in $\mathsf{SOpFib}(\mathsf{B})$ and $\mathsf{SFib}(\mathsf{B})$ are required to preserve (op)cartesian cells, which is not needed in $\mathsf{B}^{\downarrow}$. As such, to obtain a useful notion of display map bicategory, some adaptions are needed.

For that reason, our notion of display map bicategory comes in three different flavors.[3]

Definition 7.1 (disp_map_bicat)**.** Let B be a bicategory. A **covariant display map bicategory** of B is a predicate P on the 1-cells of B such that

- If a morphism satisfies P, then it is an internal Street opfibration;
- P is closed under pullback;
- Pullbacks along 1-cells satisfying P exist.

A **display map** is a 1-cell together with a proof of the predicate.

We also define notions of **contravariant display map bicategory** and **display map bicategory**: for the former, the display maps are required to be internal Street fibrations, while for the latter, we do not make such a restriction.

Example 7.2 (sopfib_disp_map_bicat_is_covariant)**.** Let B be a bicategory with pullbacks. Since B has pullbacks, pullbacks exist along all 1-cells, and in particular, along Street opfibrations. In addition, Street opfibrations are closed under pullbacks by Proposition 6.2, and thus we have a covariant display map bicategory.

Analogously, one can define a contravariant display map bicategory of Street fibrations. There are other examples of covariant display map bicategories as well. One of them, would be **discrete Street opfibrations**. To define those, we first need the notion of **discrete morphisms**.[4]

Definition 7.3. A 1-cell $f : a \to b$ in a bicategory B is called

- (faithful_1cell) **faithful** if for every x the functor $f_* : \underline{\mathsf{B}(x,a)} \to \underline{\mathsf{B}(x,b)}$ given by postcomposition is faithful.
- (conservative_1cell) **conservative** if for every x the functor $f_* : \underline{\mathsf{B}(x,a)} \to \underline{\mathsf{B}(x,b)}$ is conservative.
- (discrete_1cell) **discrete** if it is faithful and conservative.

Example 7.4 (discrete_sopfib_disp_map_bicat_is_covariant)**.** Let B be a bicategory with pullbacks. Since both faithful and conservative 1-cells are closed under pullbacks, discrete 1-cells are as well. Hence, we get a covariant displayed map bicategory of discrete Street opfibrations.

Every covariant display map bicategory gives rise to a displayed bicategory as follows.

Definition 7.5 (disp_map_bicat_to_disp_bicat)**.** Let B be a bicategory and let D be a covariant display map bicategory on B. We define a displayed bicategory $\overline{\mathsf{D}}$ over B as follows

- The objects over $b : B$ are pairs $e : \mathsf{B}$ and $p : e \to b$ such that p is a display map;

[3]The definitions in the formalization differ slightly from the definitions in the paper. This is because the focus in the formalization is more general as it considers arbitrary display map bicategories that are not necessary covariant.

[4]In the formalization, we actually use unfolded versions of these definitions, and we prove these two versions are equivalent.

- The 1-cells over $f : b_1 \to b_2$ from $p_1 : e_1 \to b_1$ to $p_2 : e_2 \to b_2$ are pairs of a 1-cell $g : e_1 \to e_2$ and an invertible 2-cell $g \cdot p_2 \Rightarrow p_1 \cdot f$ such that g preserves opcartesian 2-cells;
- The 2-cells are as in Example 3.9

Similarly, every contravariant display map bicategory gives rise to a displayed bicategory: however, the 1-cells are required to preserve cartesian 2-cells. For ordinary display map bicategories, we do not make any restriction on the 1-cells. This way, we obtain displayed bicategories $\mathsf{SOpFib}(\mathsf{B})$ and $\mathsf{SFib}(\mathsf{B})$ over B In $\mathsf{SOpFib}(\mathsf{B})$, 2-cells are the same as 2-cells in $\mathsf{B}^{\to}$. However, 1-cells are a bit different: while 1-cells in $\mathsf{B}^{\to}$ are squares

$$\begin{tikzcd} e_1 \arrow[r, "f_e"] \arrow[d, "p_1"'] & e_2 \arrow[d, "p_2"] \\ b_1 \arrow[r, "f_b"'] & b_2 \end{tikzcd}$$

that commute up to invertible 2-cell, 1-cells in $\mathsf{SOpFib}(\mathsf{B})$ have the additional requirement that whiskering with f_e preserves opcartesian 2-cells. Similarly, we can define the bicategory $\mathsf{SFib}(\mathsf{B})$ where the objects are internal Street fibrations and where the 1-cells preserve cartesian 2-cells.

Now we can construct the desired cleavings.

Example 7.6 (DisplayMapBicatCleaving.v)**.** Let B be a bicategory and let D be a covariant display map bicategory. Similarly to Example 4.14, cartesian 1-cells are the same as pullback squares. Hence, we can construct a global cleaving for D using pullbacks and Proposition 6.2. To construct a local opcleaving for D, we use that it is contained in $\mathsf{SOpFib}(\mathsf{B})$, and then we can use the same construction as for $\mathsf{SFib}(\mathsf{B})$ [12, Example 3.4.6].

Similarly, if one has a contravariant display map bicategory, one gets a both a global and a local cleaving. However, from just a display map bicategory, one only gets a global cleaving. One can instantiate Example 7.6 to internal Street ofpibrations to obtain a global cleaving and a local opcleaving for $\mathsf{SOpFib}(\mathsf{B})$ if B has pullbacks.

Using Example 7.6, we can generalize the example of (op)fibrations to arbitrary covariant display map bicategories. We discuss the interest in this generalization in Remark 7.11.

Problem 7.7. Given a display map bicategory D on a bicategory B, to define a comprehension bicategory.

Construction 7.8 (for Problem 7.7; disp_map_bicat_comprehension_bicat)**.** From bicategory B and a display map bicategory D on B, we construct the comprehension bicategory

$$\begin{tikzcd} \mathsf{D} \arrow[rr, "\chi"] & & \mathsf{B}^{\downarrow} \\ & \mathsf{B} & \end{tikzcd}$$

The (displayed) pseudofunctor χ is the inclusion of D into $\mathsf{B}^{\downarrow}$. From the characterization of cartesian 1-cells and 2-cells in Example 7.6, we conclude that χ preserves opcartesian 1-cells and 2-cells. □

Similarly, we get a contravariant comprehension bicategory from a contravariant display map bicategory. We can instantiate Construction 7.8 to internal Street opfibration to get the following comprehension bicategory.

Example 7.9 (internal_sopfib_comprehension_bicat)**.** For bicategory B with pullbacks, we construct the comprehension bicategory

$$\begin{tikzcd} \mathsf{SOpFib}(\mathsf{B}) \arrow[rr, "\chi"] & & \mathsf{B}^{\downarrow} \\ & \mathsf{B} & \end{tikzcd}$$

The (displayed) pseudofunctor χ forgets that the morphisms in $\mathsf{SOpFib}(\mathsf{B})$ are internal Street opfibrations.

Example 7.10. By Example 7.9 any bicategory with pullbacks, thus in particular any bicategory of stacks [42], gives rise to a comprehension bicategory.

Remark 7.11. Recall from Section 2.3.1 that Coquand, Mannaa, and Ruch [13] constructed models of MLTT in stacks valued in groupoids. Using those models, they proved the independence of countable choice in univalent foundations.

With our comprehension bicategories of stacks (not just ones valued in groupoids), one could follow [13] and study the validity and independence of logical principles in *directed* type theory.

Note that we can specialize Example 7.10 to each of the examples given in Example 3.7. This is because we showed in Example 3.18 that those bicategories have pullbacks. In the case of $\mathsf{Cat}_{\mathsf{Terminal}}$, we get a comprehension bicategory in which

- the "contexts" are categories with a terminal object; and
- the "types" are opfibrations of categories which preserve terminal objects.

Similarly, we can instantiate this to the other categories mentioned in Example 7.10.

Remark 7.12. Note that similarly, we can construct a contravariant comprehension bicategory from $\mathsf{SOpFib}(\mathsf{B})$.

8 THE TYPE THEORY BTT

In this section, we extract a core syntax for two-dimensional type theory from our semantic model. We call the resulting type theory *Bicategorical Type Theory (*BTT*)*. In Section 9 we prove soundness of our syntax by giving an interpretation of the syntax in any comprehension bicategory.

The syntax extracted here is *maximally general*, in the sense that it reflects the structure of a general comprehension bicategory. In Section 10, we propose several orthogonal simplifications to the syntax, along with the corresponding semantic structure and properties. As such, our syntax and semantics are to be viewed as a framework to study different semantic structures and their corresponding internal languages, rather than as one particular pair of syntax and semantics.

In Section 8.1 we present the judgment forms of BTT, as well as the rules extracted from the bicategories of contexts and of types, respectively. In Sections 8.2 and 8.3 we present the comprehension and substitution rules, respectively.

For reasons of space we omit here several rules, namely the naturality rules in Fig. 8 and the coherencies for trifunctors. Such rules, written out in the "linear" syntax used here, would take up several lines and would be difficult to understand. Consequently, the syntax presented here is not *complete*. The presentation of the complete syntax and a suitable completeness theorem is left for future work.

8.1 Judgments and Basic Rules

BTT features contexts, substitutions, types, *generalized* terms, and reductions between terms. As befits the bicategorical semantics, judgmental equality is only postulated between parallel reductions.

There are eight kinds of *judgments* in BTT:

(1) Γ ctx, which is read as 'Γ is a context';
(2) $\Delta \vdash s : \Gamma$ (given Δ, Γ ctx), which is read as 's is a substitution from Δ to Γ';
(3) $\Delta \vdash r : s \rightsquigarrow t : \Gamma$ (where $\Delta \vdash s, t : \Gamma$), which is read as '$r$ is a reduction from s to t';
(4) $\Delta \vdash r \equiv r' : s \rightsquigarrow t : \Gamma$ (where $\Delta \vdash r, r' : s \rightsquigarrow t : \Gamma$), which is read as '$r$ is equal to r'';
(5) $\Gamma \vdash T$ type (where Γ ctx), which is read as 'T is a type in context Γ'
(6) $\Gamma \mid S \vdash t : T$ (where $\Gamma \vdash S, T$ type), which is read as 't is a term in T depending on S in context Γ'

$$\frac{\Gamma\ \mathrm{ctx}}{\Gamma \vdash 1_\Gamma : \Gamma} \qquad \frac{\Gamma, \Delta\ \mathrm{ctx} \quad \Delta \vdash s : \Gamma}{\Delta \vdash 1_s : s \rightsquigarrow s : \Gamma} \qquad \frac{\Gamma, \Delta\ \mathrm{ctx} \quad \Delta \vdash s, s' : \Gamma \quad \Delta \vdash \rho : s \rightsquigarrow s' : \Gamma}{\Delta \vdash \rho \equiv \rho : s \rightsquigarrow s' : \Gamma} \qquad \frac{\Gamma, \Delta, \mathrm{E}\ \mathrm{ctx} \quad \mathrm{E} \vdash s : \Delta \quad \Delta \vdash t : \Gamma}{\mathrm{E} \vdash st : \Gamma}$$

$$\frac{\Gamma, \Delta, \mathrm{E}\ \mathrm{ctx} \quad \mathrm{E} \vdash s : \Delta \quad \Delta \vdash t, t' : \Gamma \quad \Delta \vdash \rho : t \rightsquigarrow t' : \Gamma}{\mathrm{E} \vdash s \lhd \rho : st \rightsquigarrow st' : \Gamma} \qquad \frac{\Gamma, \Delta, \mathrm{E}\ \mathrm{ctx} \quad \mathrm{E} \vdash s, s' : \Delta \quad \Delta \vdash t : \Gamma \quad \mathrm{E} \vdash \sigma : s \rightsquigarrow s' : \Delta}{\mathrm{E} \vdash \sigma \rhd t : st \rightsquigarrow s't : \Gamma}$$

$$\frac{\Gamma, \Delta\ \mathrm{ctx} \quad \Delta \vdash s, s', s'' : \Gamma \quad \Delta \vdash \rho : s \rightsquigarrow s' : \Gamma \quad \Delta \vdash \sigma : s' \rightsquigarrow s'' : \Gamma}{\Delta \vdash \rho \bullet \sigma : s \rightsquigarrow s'' : \Gamma} \qquad \frac{\Gamma, \Delta, \mathrm{E}\ \mathrm{ctx} \quad \mathrm{E} \vdash s : \Delta \quad \Delta \vdash t : \Gamma}{\begin{array}{c} \mathrm{E} \vdash 1_s \rhd t \equiv 1_{st} : st \rightsquigarrow st : \Gamma \\ \mathrm{E} \vdash s \lhd 1_t \equiv 1_{st} : st \rightsquigarrow st : \Gamma \end{array}}$$

$$\frac{\Gamma, \Delta, \mathrm{E}\ \mathrm{ctx} \quad \mathrm{E} \vdash s : \Delta \quad \Delta \vdash t, t', t'' : \Gamma \quad \Delta \vdash \rho : t \rightsquigarrow t' : \Gamma \quad \Delta \vdash \rho' : t' \rightsquigarrow t'' : \Gamma}{\mathrm{E} \vdash (s \lhd \rho) \bullet (s \lhd \rho') \equiv s \lhd (\rho \bullet \rho') : st \rightsquigarrow st'' : \Gamma}$$

$$\frac{\Gamma, \Delta, \mathrm{E}\ \mathrm{ctx} \quad \mathrm{E} \vdash s, s', s'' : \Delta \quad \Delta \vdash t : \Gamma \quad \mathrm{E} \vdash \sigma : s \rightsquigarrow s' : \Delta \quad \mathrm{E} \vdash \sigma' : s' \rightsquigarrow s'' : \Delta}{\mathrm{E} \vdash (\sigma \rhd t) \bullet (\sigma' \rhd t) \equiv (\sigma \bullet \sigma') \rhd t : st \rightsquigarrow s''t : \Gamma}$$

$$\frac{\Gamma, \Delta, \mathrm{E}\ \mathrm{ctx} \quad \mathrm{E} \vdash s, s' : \Delta \quad \Delta \vdash t, t' : \Gamma \quad \mathrm{E} \vdash \sigma : s \rightsquigarrow s' : \Delta \quad \Delta \vdash \rho : t \rightsquigarrow t' : \Gamma}{\mathrm{E} \vdash (\sigma \rhd t) \bullet (s' \lhd \rho) \equiv (s \lhd \rho) \bullet (\sigma \rhd t') : st \rightsquigarrow s't' : \Gamma} \qquad \frac{\Gamma, \Delta\ \mathrm{ctx} \quad \Delta \vdash s : \Gamma}{\begin{array}{c} \Delta \vdash \ell_s : 1_\Delta s \overset{\sim}{\rightsquigarrow} s : \Gamma \\ \Delta \vdash r_s : s 1_\Gamma \overset{\sim}{\rightsquigarrow} s : \Gamma \end{array}}$$

$$\frac{\Gamma, \Delta, \mathrm{E}, \mathrm{Z}\ \mathrm{ctx} \quad \mathrm{Z} \vdash r : \mathrm{E} \quad \mathrm{E} \vdash s : \Delta \quad \Delta \vdash t : \Gamma}{\mathrm{Z} \vdash \alpha_{r,s,t} : r(st) \overset{\sim}{\rightsquigarrow} (rs)t : \Gamma} \qquad \frac{\Gamma, \Delta\ \mathrm{ctx} \quad \Delta \vdash s, s' : \Gamma \quad \Delta \vdash \rho : s \rightsquigarrow s' : \Gamma}{\begin{array}{c} \Delta \vdash r_s^{-1} \bullet (\rho \rhd 1_\Gamma) \bullet r_{s'} \equiv \rho : s \rightsquigarrow s' : \Gamma \\ \Delta \vdash \ell_s^{-1} \bullet (1_\Delta \lhd \rho) \bullet \ell_{s'} \equiv \rho : s \rightsquigarrow s' : \Gamma \end{array}}$$

$$\frac{\Gamma, \Delta, \mathrm{E}, \mathrm{Z}\ \mathrm{ctx} \quad \mathrm{Z} \vdash s : \mathrm{E} \quad \mathrm{E} \vdash t : \Delta \quad \Delta \vdash u, u' : \Gamma \quad \Delta \vdash \rho : u \rightsquigarrow u' : \Gamma}{\mathrm{Z} \vdash \alpha_{s,t,u}^{-1} \bullet (s \lhd (t \lhd \rho)) \bullet \alpha_{s,t,u'} \equiv st \lhd \rho : (st)u \rightsquigarrow (st)u' : \Gamma}$$

$$\frac{\Gamma, \Delta, \mathrm{E}, \mathrm{Z}\ \mathrm{ctx} \quad \mathrm{Z} \vdash s : \mathrm{E} \quad \mathrm{E} \vdash t, t' : \Delta \quad \Delta \vdash u : \Gamma \quad \mathrm{E} \vdash \rho : t \rightsquigarrow t' : \Delta}{\mathrm{Z} \vdash \alpha_{s,t,u}^{-1} \bullet (s \lhd (\rho \rhd u)) \bullet \alpha_{s,t',u} \equiv (s \lhd \rho) \rhd u : (st)u \rightsquigarrow (st')u : \Gamma}$$

$$\frac{\Gamma, \Delta, \mathrm{E}, \mathrm{Z}\ \mathrm{ctx} \quad \mathrm{Z} \vdash s, s' : \mathrm{E} \quad \mathrm{E} \vdash t : \Delta \quad \Delta \vdash u : \Gamma \quad \mathrm{Z} \vdash \rho : s \rightsquigarrow s' : \mathrm{E}}{\mathrm{Z} \vdash \alpha_{s,t,u}^{-1} \bullet (\rho \rhd tu) \bullet \alpha_{s',t,u} \equiv (\rho \rhd t) \rhd u : (st)u \rightsquigarrow (s't)u : \Gamma} \qquad \frac{\Gamma, \Delta\ \mathrm{ctx} \quad \Delta \vdash s, s' : \Gamma \quad \Delta \vdash \rho : s \rightsquigarrow s' : \Gamma}{\begin{array}{c} \Delta \vdash 1_s \bullet \rho \equiv \rho : s \rightsquigarrow s' : \Gamma \\ \Delta \vdash \rho \bullet 1_{s'} \equiv \rho : s \rightsquigarrow s' : \Gamma \end{array}}$$

$$\frac{\Gamma, \Delta\ \mathrm{ctx} \quad \Delta \vdash s, s', s'', s''' : \Gamma \quad \Delta \vdash \rho : s \rightsquigarrow s' : \Gamma \quad \Delta \vdash \sigma : s' \rightsquigarrow s'' : \Gamma \quad \Delta \vdash \tau : s'' \rightsquigarrow s''' : \Gamma}{\Delta \vdash (\rho \bullet \sigma) \bullet \tau \equiv \rho \bullet (\sigma \bullet \tau) : s \rightsquigarrow s''' : \Gamma}$$

$$\frac{\Gamma, \Delta, \mathrm{E}\ \mathrm{ctx} \quad \mathrm{E} \vdash s : \Delta \quad \Delta \vdash t : \Gamma}{\mathrm{E} \vdash \alpha_{s,1_\Delta,t} \bullet (r_s \rhd t) \equiv s \lhd \ell_t : s(1_\Delta t) \rightsquigarrow st : \Gamma}$$

$$\frac{\Gamma, \Delta, \mathrm{E}, \mathrm{Z}, \mathrm{H}\ \mathrm{ctx} \quad \mathrm{H} \vdash s : \mathrm{Z} \quad \mathrm{Z} \vdash t : \mathrm{E} \quad \mathrm{E} \vdash u : \Delta \quad \Delta \vdash v : \Gamma}{\mathrm{H} \vdash \alpha_{s,t,uv} \bullet \alpha_{st,u,v} \equiv (s \lhd \alpha_{t,u,v}) \bullet \alpha_{s,tu,v} \bullet (\alpha_{s,t,u} \rhd v) : s(t(uv)) \rightsquigarrow ((st)u)v : \Gamma}$$

Fig. 1. Rules for the bicategory of contexts

(7) $\Gamma \mid S \vdash \rho : t \rightsquigarrow t' : T$ (where $\Gamma \mid S \vdash t, t' : T$), which is read as '$\rho$ is a reduction from t to t''

(8) $\Gamma \mid S \vdash \rho \equiv \rho' : t \rightsquigarrow t' : T$ (where $\Gamma \mid S \vdash \rho, \rho' : t \rightsquigarrow t' : T$), which is read as '$\rho$ is equal to ρ''.

We often abbreviate the above judgments and write, *e.g.*, just $\rho : t \rightsquigarrow t'$ instead of $\Gamma \mid S \vdash \rho : t \rightsquigarrow t' : T$. For these judgments, we have rules that express the bicategorical structure of contexts and types. Rules are given in Fig. 1 for the bicategory of contexts, and in Fig. 2 for the bicategory of types.

We also introduce symbols which read like judgments but stand for several judgments, using the composition and identities introduced in Fig. 1 and Fig. 2.

(1) $\Delta \vdash \rho : s \overset{\sim}{\rightsquigarrow} t : \Gamma$ stands for the four judgments

- $\Delta \vdash \rho : s \rightsquigarrow t : \Gamma \qquad \Delta \vdash \rho^{-1} : t \rightsquigarrow s : \Gamma$
- $\rho \bullet \rho^{-1} \equiv 1_s \qquad \rho^{-1} \bullet \rho \equiv 1_t$

(2) $\Gamma \mid S \vdash \rho : t \tilde{\rightsquigarrow} t' : T$ stands for the four judgments
- $\Gamma \mid S \vdash \rho : t \rightsquigarrow t' : T \qquad \Gamma \mid S \vdash \rho^{-1} : t' \rightsquigarrow t : T$
- $\rho \bullet \rho^{-1} \equiv 1_t \qquad \rho^{-1} \bullet \rho \equiv 1_{t'}$

(3) $\Delta \ \tilde{\vdash} \ s : \Gamma$ stands for the four judgments
- $\Delta \vdash s : \Gamma \qquad \Gamma \vdash s^{-1} : \Delta$
- $s^{\ell} : ss^{-1} \tilde{\rightsquigarrow} 1_\Delta \qquad s^{\rho} : s^{-1}s \tilde{\rightsquigarrow} 1_\Gamma$

(4) $\Gamma \mid S \ \tilde{\vdash} \ t : T$ stands for the four judgments
- $\Gamma \mid S \vdash t : T \qquad \Gamma \mid T \vdash t^{-1} : S$
- $t^{\ell} : tt^{-1} \tilde{\rightsquigarrow} 1_S \qquad t^{\rho} : t^{-1}t \tilde{\rightsquigarrow} 1_T$

Remark 8.1. By abuse of notation, we write several topically related rules that share all the same hypotheses as one rule with several conclusions. These rules then also share the same name, *e. g.,* extend-con-Ty. When referring to a rule by name, it will be clear from the context which of the possible rules we refer to. The names of inference rules in the text are hyperlinks to the corresponding rules (*e. g.,* map). The equality $\equiv$ is assumed to be a congruence for every other constructor and judgment. For brevity, we have not recorded here the resulting rules. When parentheses are omitted, everything is associated to the left: that is, rst stands for $((rs)t)$. Note also that in several rules in which it is necessary to re-associate several four or more terms or substitutions, we have written α instead of a long composition of whiskered associators $\alpha_{\bullet,\bullet,\bullet}$ in the interest of readability.

8.2 Comprehension Structure

Comprehension, that is, context extension, is extracted from the pseudofunctor χ. The rules for comprehension are given in Fig. 3. There are some notable differences to comprehension in MLTT. First, the rule extend-con-Tm, which forms a substitution, comes together with a reduction that expresses the commutativity of a triangle. Second, we also have a rule extend-con-Red that extends a substitution with a reduction. Since reductions are proof-relevant, this rule comes with a coherency on the commutativity.

8.3 Substitution Structure

Substitution is given, in the semantics, by the global and local (op)cleaving structure. We reflect this into the syntax as *explicit substitution*, as was also used, *e. g.,* in [18, 29] in their respective settings.

The rules for substitution are given in Figs. 4 to 9. We distinguish them based on whether we need the global cleaving or the local opcleaving to interpret them. There are several important observations to be made about these rules. First, in line with our truly bicategorical approach, we do not assume the comprehension bicategory is split. In particular, no equality between $T[\mathrm{id}]$ and T is postulated. Instead, there is an equivalence between them (see sub-id and sub-comp), and terms of these types are transported along the equivalence.

The rule map expresses that each type T behaves 'functorially': for each $\Delta \vdash s : \Gamma$ (*i. e.,* object in 'hom(Δ, Γ)') we get a type $T[s]$ (*i. e.,* object in 'the category of types in context Δ') by sub-ty and for each $r : s \rightsquigarrow s'$ (i.e., morphism in 'hom(Δ, Γ)') we get a term $\Delta \mid T[s] \vdash \mathsf{map}\ T\ \theta : T[s']$ (*i. e.,* morphism in 'the category of types in context Δ') by map. The rules map-id and map-comp ensure that $T[-]$ preserves identity and composition. With rew-tm we can then understand terms to be 'natural transformations'.

Remark 8.2. For contravariant comprehension bicategories (Remark 4.19), the rule map would be in the opposite direction, while for isovariant comprehension bicategories, this rule would be restricted to isomorphisms in the base.

Remark 8.3. Using the Grothendieck construction, we can view the rules from the perspective of fiber bicategories. If $P : \mathsf{E} \longrightarrow \mathsf{B}$ has a global cleaving and a local opcleaving, and furthermore opcartesian 2-cells are preserved under whiskering, then we obtain a trifunctor $\mathsf{B}^{\mathrm{op}} \longrightarrow \mathsf{Bicat}$, which sends every $x : \mathsf{B}$ to the fiber of x along P. The rules given in Fig. 4 say that any 1-cell in the base gives rise to a pseudofunctor. For example, the rules sub-ty, sub-tm, and sub-red represent the actions on objects, 1-cells, and 2-cells, respectively, while sub-pres-id and sub-pres-comp represent the invertible 2-cells that witness the preservation of the identity and composition of 1-cells. The other rules in that table are the coherence laws of pseudofunctors: for example, sub-red-pres-id is the preservation of identity 2-cells.

Tables Fig. 5 and Fig. 6 give pseudonatural equivalences expressing that the assignment of the previously mentioned pseudofunctor is actually pseudofunctorial: the identity and composition are preserved up to pseudonatural equivalence. For example, the rules sub-id and tm-sub-id in Fig. 5 represent the action on objects and the naturality squares. On the other hand, sub-red-id, sub-id-pres-id, and sub-id-pres-comp are the usual coherencies of pseudonatural transformations.

The rules of Fig. 7 state that we obtain a pseudonatural transformation from a 2-cell in the base. The action on objects and 1-cells is given by map and rew-tm, respectively, while rew-red, rew-pres-id, and rew-pres-comp describe the usual coherencies of pseudotransformations. The remaining rules can be found in Fig. 7. In this figure, four invertible modifications are described: map-id and map-comp describe the preservation of identity and composition, respectively, while map-lwhisker and map-rwhisker describe the preservation of both left and right whiskering. Note that we left out the naturality conditions of these modifications. In addition, two additional coherencies are required which express that all this data together forms a trifunctor. We do not write them down, but instead, we refer the reader to [21, Definition 3.3.1].

9 SOUNDNESS: INTERPRETATION IN COMPREHENSION BICATEGORIES

In this section, we give an interpretation of BTT in any comprehension bicategory. To this end, we fix a comprehension bicategory $\mathsf{D} \xrightarrow{\chi} \mathsf{B}^{\downarrow}$ over B. We interpret the judgments as follows.

- Γ ctx is interpreted as an object $[\![\Gamma]\!]$ of B.
- $\Delta \vdash s : \Gamma$ is interpreted as a 1-cell $[\![s]\!] : [\![\Delta]\!] \to [\![\Gamma]\!]$ in B.
- $\Delta \vdash r : s \rightsquigarrow t : \Gamma$ is interpreted as a 2-cell $[\![r]\!] : [\![s]\!] \Rightarrow [\![t]\!]$ in B.
- $\Delta \vdash r \equiv r' : s \rightsquigarrow t : \Gamma$ is interpreted as an equality $[\![r]\!] = [\![r']\!]$.
- $\Gamma \vdash T$ type is interpreted as an object $[\![T]\!]$ in D over $[\![\Gamma]\!]$.
- $\Gamma \mid S \vdash t : T$ is interpreted as a 1-cell $[\![t]\!] : [\![S]\!] \to [\![T]\!]$ over the identity on $[\![\Gamma]\!]$.
- $\Gamma \mid S \vdash r : t \rightsquigarrow t' : T$ is interpreted as a 2-cell $[\![r]\!] : [\![t]\!] \Rightarrow [\![t']\!]$ over the identity 2-cell.
- $\Gamma \mid S \vdash r \equiv r' : t \rightsquigarrow t' : T$ is interpreted as an equality $[\![r]\!] = [\![r']\!]$.

Regarding the "bicategorical" rules of Figs. 1 and 2, each rule is analogous to one of the operations or laws of a bicategory (see, for instance, [2, Definition 3.1]), which also indicates how it is interpreted.

9.1 Comprehension

Next we interpret the rules related to comprehension of Fig. 3. Suppose that we have a context $\Gamma : \mathsf{B}$ and a type T over Γ. Its image $\chi(T)$ in $\mathsf{B}^{\downarrow}$ gives rise to an object $\Gamma.T : \mathsf{B}$ and a 1-cell $\pi_{\Gamma.T} : \Gamma.T \to \Gamma$, which interprets extend-con-Ty. For a 1-cell t from S to T over the identity, the resulting 1-cell $\chi(t)$ is part of a triangle

$$\begin{tikzcd}
\Gamma.S \arrow[rr, "\chi(t)"] \arrow[dr, "\pi_{\Gamma.S}"'] & & \Gamma.T \arrow[dl, "\pi_{\Gamma.T}"] \\
& \Gamma &
\end{tikzcd}$$

which commutes up to invertible 2-cell. This yields the interpretation of the rules in extend-con-Tm. Furthermore, a reduction $r : t \rightsquigarrow t'$ is mapped by χ to a 2-cell from $\chi(t)$ to $\chi(t')$ in $\mathsf{B}^{\downarrow}$, and this is how we interpret extend-con-Red. The rules extend-con-id and extend-con-comp are interpreted by the identitor and compositor of χ, respectively, while the remaining rules in Fig. 3 are satisfied because they translate to the laws that express that this data forms a pseudofunctor.

9.2 Global Substitution

Next, we interpret the rules for substitution. The rule sub-ty follows directly from the global cleaving. The interpretation of the other rules require more explanation.

9.2.1 Interpretation of sub-tm. To interpret sub-tm, we assume that we have a substitution $s : \Delta \longrightarrow \Gamma$ between contexts Δ and Γ. We also assume that we have types S, T over Γ and a morphism $t : S \longrightarrow T$ over 1_Γ. This situation is encapsulated in the following diagram, using the notation introduced in Definition 4.6:

$$\begin{tikzcd}
S[s] \arrow[r, "L_s(S)"] & S \arrow[d, "t"] \\
T[s] \arrow[r, "L_s(T)"'] & T \\
\\
\Delta \arrow[r, "s"'] & \Gamma
\end{tikzcd}$$

Our goal is to construct a 1-cell $t[s] : S[s] \longrightarrow T[s]$ that lies over the identity on Δ. Since the morphism $L_s(T) : T[s] \longrightarrow T$ is cartesian, it suffices to construct a 1-cell, say, $\beta : S[s] \longrightarrow T$ that lies over $1_\Delta \cdot s$; we then obtain the desired 1-cell as the factorization $\mathsf{F}_1(L_s(T), \beta)$ of that 1-cell via $L_s(T)$; recall that $\mathsf{F}_1(\_, \_)$ was defined in Definition 4.1. Note that every displayed bicategory over a univalent bicategory has a local isocleaving by Remark 4.10. We have an invertible 2-cell $r_s \bullet {\ell_s}^{-1} : s \cdot 1_\Gamma \Rightarrow 1_\Delta \cdot s$, and we set $\beta := (r_s \bullet {\ell_s}^{-1})_*(L_s(S) \cdot t)$. Define

$$t[s] := \mathsf{F}_1(L_s(T), \beta).$$

Overall, we can summarize the construction with the following diagram:

$$\begin{tikzcd}
 & S[s] \arrow[r, "L_s(S)"] & S \arrow[dr, "t"] & \\
S[s] \arrow[ur, equal] \arrow[drr, bend left, "\beta"] \arrow[dr, dashed, "t{[s]}"'] & \cong\Uparrow & \Swarrow\cong & T \arrow[dl, equal] \\
 & T[s] \arrow[r, "L_s(T)"'] & T & \\
\Delta \arrow[r, "1_\Delta"'] & \Delta \arrow[r, "s"'] & \Gamma \arrow[r, "1_\Gamma"'] & \Gamma
\end{tikzcd}$$

9.2.2 Interpretation of sub-red. To interpret sub-red, we assume that we have a substitution $s : \Delta \to \Gamma$ and types S and T in context Γ. We also assume that we have 1-cells $t, t' : S \to T$ over 1_Γ and a 2-cell $\rho : t \Rightarrow t'$. Consider the following diagram:

$$\begin{tikzcd}
S[1_\Gamma] \arrow[r, "L_s(S)"] \arrow[d, dashed, bend right, "{t[s]}"'] \arrow[d, dashed, bend left, "{t'[s]}"] & S \arrow[d, bend right=50, "t"', ""{name=A}] \arrow[d, bend left=50, "t'", ""'{name=B}] \arrow[from=A, to=B, Rightarrow, "\rho"'] \\
T[1_\Gamma] \arrow[r, "L_s(T)"'] & T
\end{tikzcd}$$

$$\begin{tikzcd}
\Delta \arrow[r, "s"'] & \Gamma
\end{tikzcd}$$

Our goal is to construct a 2-cell from $t[s]$ to $t[s']$ over the identity on Δ. The source and target, respectively, were constructed by factoring a 1-cell via a cartesian 1-cell, as follows:

$$\begin{tikzcd}
 & S[s] \arrow[r, "L_s(S)"] \arrow[dr, phantom, "\overset{\cong}{\swarrow}"] & S \arrow[dr, "t"] & \\
S[s] \arrow[ur, equal] \arrow[drr, bend left=15, "\beta"] \arrow[dr, dashed, "{t[s]}"'] & & & T \arrow[dl, equal] \\
 & T[s] \arrow[r, "L_s(T)"'] \arrow[u, phantom, "\overset{\cong}{\Uparrow}"] & T & \\
\Delta \arrow[r, "1_\Delta"'] & \Delta \arrow[r, "s"'] & \Gamma \arrow[r, "1_\Gamma"'] & \Gamma
\end{tikzcd}
\qquad
\begin{tikzcd}
 & S[s] \arrow[r, "L_s(S)"] \arrow[dr, phantom, "\overset{\cong}{\swarrow}"] & S \arrow[dr, "t'"] & \\
S[s] \arrow[ur, equal] \arrow[drr, bend left=15, "\beta'"] \arrow[dr, dashed, "{t'[s]}"'] & & & T \arrow[dl, equal] \\
 & T[s] \arrow[r, "L_s(T)"'] \arrow[u, phantom, "\overset{\cong}{\Uparrow}"] & T & \\
\Delta \arrow[r, "1_\Delta"'] & \Delta \arrow[r, "s"'] & \Gamma \arrow[r, "1_\Gamma"'] & \Gamma
\end{tikzcd}$$

To construct the desired 2-cell $\rho[s] : t[s] \Rightarrow t[s']$, we use Item 2 of Definition 4.1. More specifically, it suffices to construct

- a 2-cell δ from 1_Δ to 1_Δ, and
- a 2-cell $\overline{\sigma}$ from β to β'.

Once we have defined those, we define $\rho[s] := \mathsf{E}(\delta, \overline{\sigma})$.

For the first of the two, we take the identity 2-cell 1_{1_Δ}. For the second, we take the composition of 2-cells shown in the following diagram:

$$\begin{tikzcd}
S[s] \arrow[r, "L_s(S)"] \arrow[rr, bend left=60, "\beta"] \arrow[rr, bend right=60, "\beta'"'] & S \arrow[u, phantom, "\overset{\cong}{\Downarrow}"] \arrow[d, phantom, "\overset{\cong}{\Downarrow}"] \arrow[r, bend left, "t", ""'{name=A}] \arrow[r, bend right, "t'"', ""{name=B}] \arrow[from=A, to=B, Rightarrow, "\rho"] & T
\end{tikzcd}$$

The interpretation of the rules sub-red-id and sub-red-comp follows from the uniqueness of factorization 2-cells.

9.2.3 Interpretation of sub-id. Next we interpret sub-id. Suppose we have a context Γ and a type T in Γ. Note that we have a diagram as follows:

$$\begin{tikzcd}
T \arrow[dr, "1_T"] & \\
T[1_\Gamma] \arrow[r, "L_{1_\Gamma}(T)"'] & T \\
\Gamma \arrow[r, "1_\Gamma"'] & \Gamma
\end{tikzcd}$$

We interpret $\mathsf{sub}_{\mathsf{id}}\ T$ by $L_{1_\Gamma}(T)$, so it suffices to show that $L_{1_\Gamma}(T)$ is an equivalence. To construct the inverse, we use the following diagram:

$$\begin{tikzcd}
T \arrow[r, "1_T"] \arrow[dr, dashed, "\mathsf{F}_1(L_{1_\Gamma}(T), 1_T \cdot 1_T)"'] & T \arrow[dr, "1_T"] \arrow[d, phantom, "\overset{\Longrightarrow}{\cong}"] & \\
& T[1_\Gamma] \arrow[r, "L_{1_\Gamma}(T)"'] & T \\
\Gamma \arrow[r, "1_\Gamma"'] & \Gamma \arrow[r, "1_\Gamma"'] & \Gamma
\end{tikzcd}$$

We set $\iota := \mathsf{F}_1(L_{1_\Gamma}(T), 1_T \cdot 1_T)$, and it remains to show that we have an invertible 2-cell from $L_{1_\Gamma}(T) \cdot \iota$ to 1_T. To do so, we use Item 2 of Definition 4.1, and we consider the following diagram

$$\begin{tikzcd}
T[1_\Gamma] \arrow[drr, "L_{1_\Gamma}(T) \cdot \iota"'] & T[1_\Gamma] \arrow[dr, "1_{T[s]}"] & & \\
& & T[1_\Gamma] \arrow[r, "L_{1_\Gamma}(T)"'] & T \\
\Gamma \arrow[r, "1_\Gamma"'] & \Gamma \arrow[r, "1_\Gamma"'] & \Gamma \arrow[r, "1_\Gamma"'] & \Gamma
\end{tikzcd}$$

We need to construct the following:

- An invertible 2-cell δ from $1_\Gamma \cdot 1_\Gamma$ to 1_Γ, and
- An invertible 2-cell $\overline{\sigma}$ from $L_{1_\Gamma}(T) \cdot \iota \cdot L_{1_\Gamma}(T)$ to $1_{T[s]} \cdot L_{1_\Gamma}(T)$.

For δ, we take λ. For $\overline{\sigma}$, we use that $\iota \cdot L_{1_\Gamma}(T) \cong 1_T \cdot 1_T$, and thus we have the following composition of invertible 2-cells:

$$\begin{aligned}
L_{1_\Gamma}(T) \cdot \iota \cdot L_{1_\Gamma}(T) &\cong L_{1_\Gamma}(T) \cdot 1_T \cdot 1_T \\
&\cong L_{1_\Gamma}(T) \\
&\cong 1_{T[s]} \cdot L_{1_\Gamma}(T)
\end{aligned}$$

The desired invertible 2-cell is thus defined by $\mathsf{E}(\delta, \overline{\sigma})$.

9.2.4 Interpretation of sub-comp. To interpret sub-comp, we suppose that we have substitutions $s' : \mathrm{E} \longrightarrow \Delta$ and $s : \Delta \longrightarrow \Gamma$, and a type T in Γ. As such, we have the following diagram:

$$\begin{tikzcd}
T[s \cdot s'] \arrow[rrd, "L_{s \cdot s'}(T)"] & & \\
T[s'][s] \arrow[r, "L_s(T[s'])"'] & T[s'] \arrow[r, "L_{s'}(T)"'] & T \\
\mathrm{E} \arrow[r, "s"'] & \Delta \arrow[r, "s'"'] & \Gamma
\end{tikzcd}$$

Note that both $L_{s \cdot s'}(T)$ and $L_s(T[s']) \cdot L_{s'}(T)$ are cartesian 1-cells over $s \cdot s'$, since cartesian 1-cells are closed under composition by Proposition 4.2. For that reason, we get an adjoint equivalence $\mathsf{sub}_{\mathsf{comp}}(s, s') : T[s'][s] \longrightarrow T[s \cdot s']$ making the diagram below commute up to invertible 2-cell by Construction 4.4.

$$\begin{tikzcd}
T[s \cdot s'] \arrow[rrd, "L_{s \cdot s'}(T)"] \arrow[rd, phantom, "\Downarrow \cong"] & & \\
T[s'][s] \arrow[u, dashed, "\mathsf{sub}_{\mathsf{comp}}(s,s')"] \arrow[r, "L_s(T[s'])"'] & T[s'] \arrow[r, "L_{s'}(T)"'] & T \\
\mathrm{E} \arrow[r, "s"'] & \Delta \arrow[r, "s'"'] & \Gamma
\end{tikzcd}$$

9.2.5 Interpretation of tm-sub-id. Next we interpret tm-sub-id, and for that, we suppose that we have a context Γ, types S and T over Γ, and a 1-cell $t : S \longrightarrow T$ over 1_Γ. We need to construct an invertible 2-cell from $\mathsf{sub}_{\mathsf{id}}^{-1} \cdot t[1_\Gamma] \cdot \mathsf{sub}_{\mathsf{id}}$ to t, and for that, it suffices to construct an invertible 2-cell from $t[1_\Gamma] \cdot \mathsf{sub}_{\mathsf{id}}$ to $\mathsf{sub}_{\mathsf{id}} \cdot t$. Recall that we defined $\mathsf{sub}_{\mathsf{id}}$ to be $L_{1_\Gamma}(T)$, and recall that we defined $t[1_\Gamma]$ as follows:

$$\begin{tikzcd}
& S[s] \arrow[r, "L_{1_\Gamma}(S)"] \arrow[rd, phantom, "\swarrow \cong"] & S \arrow[rd, "t"] & \\
S[1_\Gamma] \arrow[ru, equal] \arrow[rr, bend left, "\beta"] \arrow[rd, dashed, "t{[1_\Gamma]}"'] & \overset{\cong}{\Uparrow} & & T \\
& T[1_\Gamma] \arrow[r, "L_{1_\Gamma}(T)"'] & T \arrow[ru, equal] & \\
\Gamma \arrow[r, "1_\Gamma"'] & \Gamma \arrow[r, "1_\Gamma"'] & \Gamma \arrow[r, "1_\Gamma"'] & \Gamma
\end{tikzcd}$$

By composing the two invertible 2-cells depicted in this diagram, we get the desired invertible 2-cell. The rule tm-sub-comp is interpreted analogously.

9.2.6 Interpretation of sub-pres-id. The rules sub-pres-id and sub-pres-comp are interpreted similarly, and we show the interpretation of sub-pres-id. Suppose that we have a substitution $s : \Delta \longrightarrow \Gamma$ and a type T in context Γ. We consider two morphisms: the identity on $T[s]$ and $1_T[s]$. These are illustrated in the following diagram:

$$\begin{tikzcd}
T[s] \arrow[r, "L_s(S)"] \arrow[d, bend right, "1_T[s]"'] \arrow[d, bend left, "1_{T[s]}"] & T \arrow[d, "1_T"] \\
T[s] \arrow[r, "L_s(T)"'] & T
\end{tikzcd}$$

Our goal is to construct an invertible 2-cell from $1_{T[s]}$ to $1_T[s]$. To do so, we first recall the construction of $1_T[s]$.

$$\begin{tikzcd}
& T[s] \arrow[r, "L_s(T)"] \arrow[dd, phantom, "\overset{\cong}{\Leftarrow}"] & T \arrow[dr, "1_T"] & \\
T[s] \arrow[ur, equal] \arrow[drr, bend left, "\beta"] \arrow[dr, dashed, "1_T[s]"'] & & & T \\
& T[s] \arrow[u, phantom, "\cong\Uparrow"] \arrow[r, "L_s(T)"'] & T \arrow[ur, equal] &
\end{tikzcd}$$

$$\begin{tikzcd}
\Delta \arrow[r, "1_\Delta"'] & \Delta \arrow[r, "s"'] & \Gamma \arrow[r, "1_\Gamma"'] & \Gamma
\end{tikzcd}$$

To construct the desired invertible 2-cell, we again use Item 2 of Definition 4.1. As such, we need to construct the following:

- An invertible 2-cell δ from 1_Δ to 1_Δ; and
- An invertible 2-cell $\overline{\sigma}$ from $1_T[s] \cdot L_s(T)$ to $1_{T[s]} \cdot L_s(T)$.

With those in place, we define $\mathsf{Sub}_\mathsf{I}(s) := \mathsf{E}(\delta, \overline{\sigma})$. For δ, we take the identity, while for $\overline{\sigma}$, we take the following composition

$$1_{T[s]} \cdot L_s(T) \cong L_s(T) \cdot 1_T \cong \beta \cong 1_T[s] \cdot L_s(T).$$

9.2.7 Interpretation of sub-pres-lunitor. The remaining laws stating equalities of 2-cells, such as sub-pres-comp and sub-id-pres-id, are all proven in a similar way. Our goal is to prove an equality $\overline{\alpha} = \overline{\beta}$ with 1-cells and 2-cells as in the diagram below:

$$\begin{tikzcd}
S \arrow[dr, bend left, "t'"] \arrow[dr, bend right, "t"'] \arrow[dr, phantom, "\overline{\alpha}\Nearrow\ \Nearrow\overline{\beta}"] & & \\
& T[s''] \arrow[r, "L_{s''}(T)"'] & T
\end{tikzcd}$$

$$\begin{tikzcd}
E \arrow[r, bend left, "s'"] \arrow[r, bend right, "s"'] \arrow[r, phantom, "\alpha\Uparrow\ \Uparrow\beta"] & \Delta \arrow[r, "s''"'] & \Gamma
\end{tikzcd}$$

Here t and t' live over s and s', respectively, while $\overline{\alpha}$ and $\overline{\beta}$ live over α and β, respectively. To prove the equality, we take the following steps:

(1) We construct an invertible 2-cell θ from $t \cdot L_{s''}(T)$ to $t' \cdot L_{s''}(T)$.
(2) We prove that $\alpha = \beta$.
(3) We prove that $\overline{\alpha} \triangleright L_{s''}(T) = \theta = \overline{\beta} \triangleright L_{s''}(T)$.

From the first item, we get that both t and t' are cartesian factorizations of the same $S \longrightarrow T$. From the second and third item, we get that both $\overline{\alpha}$ and $\overline{\beta}$ are factorization 2-cells from t to t', and since such 2-cells are unique, we get $\overline{\alpha} = \overline{\beta}$.

We demonstrate this for sub-pres-lunitor. Suppose we have a substitution $s : \Delta \longrightarrow \Gamma$, types S and T in context Γ, and a morphism $t : S \longrightarrow T$ over 1_Γ. We are in the following situation:

$$\begin{tikzcd}
S[s] \arrow[rr, bend left, "{t[s]}"] \arrow[rr, bend right, "{1_S[s] \cdot t[s]}"'] \arrow[rr, phantom, "\overline{\alpha} \Uparrow \quad \Uparrow \overline{\beta}"] & & T[s] \arrow[r, "L_s(T)"] & T
\end{tikzcd}$$

$$\begin{tikzcd}
\Delta \arrow[r, bend left, "1_\Delta"] \arrow[r, bend right, "1_\Delta \cdot 1_\Delta"'] \arrow[r, phantom, "\alpha \Uparrow \quad \Uparrow \beta"] & \Delta \arrow[r, "s"'] & \Gamma
\end{tikzcd}$$

Here, we define

$$\alpha := (1_{1_\Delta} \triangleright 1_\Delta) \bullet \ell_{1_\Delta} \qquad \overline{\alpha} := (\mathsf{Sub}_{\mathsf{I}}(s) \triangleright t[s]) \bullet \ell_{t[s]}$$
$$\beta := 1_{1_\Delta \cdot 1_\Delta} \bullet \ell_{1_\Delta} \qquad \overline{\beta} := \mathsf{Sub}_{\mathsf{C}}(1_S, t, s) \bullet \ell_t[s]$$

From how α and β are defined, we see that both are equal to ℓ_{1_Δ}, and hence, $\alpha = \beta$.

Next we construct an invertible 2-cell from $1_S[s] \cdot t[s] \cdot L_s(T)$ to $t[s] \cdot L_s(T)$. By construction of $t[s]$, we have an invertible 2-cell $\theta_1 : t[s] \cdot L_s(T) \cong \cdot L_s(S) \cdot t$. In addition, we have an invertible 2-cell $\theta_2 : 1_S[s] \cdot L_s(S) \cong L_s(S) \cdot 1_S$. This is illustrated in the following diagram:

$$\begin{tikzcd}
S[s] \arrow[r, "L_s(S)"] \arrow[d, "{1_S[s]}"'] & S \arrow[d, "1_S"] \\
S[s] \arrow[r, "L_s(S)" description] \arrow[d, "{t[s]}"'] \arrow[ur, phantom, "\theta_1"] & S \arrow[d, "t"] \\
T[s] \arrow[r, "L_s(T)"'] \arrow[ur, phantom, "\theta_2"] & T
\end{tikzcd}$$

Now we define an invertible 2-cell $\theta_3 : 1_S[s] \cdot t[s] \cdot L_s(T) \cong L_s(S) \cdot t$ as follows

$$\theta_3 := \alpha^{-1} \bullet (1_S[s] \triangleleft \theta_2) \bullet \alpha \bullet (\theta_1 \triangleright t) \bullet (r_\sigma \triangleright t)$$

The desired invertible 2-cell $\theta : 1_S[s] \cdot t[s] \cdot L_s(T) \cong t[s] \cdot L_s(T)$ is $\theta_3 \bullet \theta_2^{-1}$.

For the last step, we only prove that $\overline{\alpha} \rhd L_s(T) = \theta$. We first recall how we constructed $\mathsf{Sub}_{\mathsf{l}}(s)$. For brevity, we write $\sigma := L_s(S)$ and $\tau := L_s(T)$.

$$\begin{tikzcd}
S[s] \arrow[r, "\sigma"] \arrow[d, bend right=60, "1_{S[s]}"'{name=L}] \arrow[d, bend left=60, "1_{S[s]}"{name=R}] & S \arrow[d, "1_S"] \\
S[s] \arrow[r, "\sigma"'] & S
\arrow[from=L, to=R, Rightarrow, "\mathsf{Sub}_{\mathsf{l}}(s)^{-1}"]
\arrow[from=2-1, to=1-2, Rightarrow, shorten=20pt, "\theta_1"']
\end{tikzcd}$$

By construction, we have that $(\mathsf{Sub}_{\mathsf{l}}(s) \rhd \sigma) \bullet \ell_\sigma \bullet r_\sigma^{-1} = \theta_1$. In particular, we have $\mathsf{Sub}_{\mathsf{l}}(s) \rhd \sigma = \theta_1 \bullet r_\sigma \bullet \ell_\sigma^{-1}$.

To show that $((\mathsf{Sub}_{\mathsf{l}}(s) \rhd t[s]) \bullet \ell_{t[s]}) \rhd \tau = \theta_3 \bullet \theta_2^{-1}$, it suffices to show that $((\mathsf{Sub}_{\mathsf{l}}(s) \rhd t[s]) \bullet \ell_{t[s]}) \rhd \tau \bullet \theta_2 = \theta_3$. This follows from the following chain of equalities:

$$\begin{aligned}
&\alpha^{-1} \bullet (1_S[s] \lhd \theta_2^{-1}) \bullet \alpha \bullet ((\mathsf{Sub}_{\mathsf{l}}(s) \rhd t[s]) \bullet \ell_{t[s]}) \rhd \tau \bullet \theta_2 \bullet (r_\sigma \rhd t)^{-1} \\
&= \alpha^{-1} \bullet (1_S[s] \lhd \theta_2^{-1}) \bullet \alpha \bullet (\mathsf{Sub}_{\mathsf{l}}(s) \rhd t[s]) \rhd \tau \bullet (\ell_{t[s]} \rhd \tau) \bullet \theta_2 \bullet (r_\sigma \rhd t)^{-1} \\
&= \alpha^{-1} \bullet (1_S[s] \lhd \theta_2^{-1}) \bullet \mathsf{Sub}_{\mathsf{l}}(s) \rhd (t[s] \cdot \tau) \bullet \alpha \bullet (\ell_{t[s]} \rhd \tau) \bullet \theta_2 \bullet (r_\sigma \rhd t)^{-1} \\
&= \alpha^{-1} \bullet \mathsf{Sub}_{\mathsf{l}}(s) \rhd (\sigma \cdot t) \bullet (1_{S[s]} \lhd \theta_2^{-1}) \bullet \alpha \bullet (\ell_{t[s]} \rhd \tau) \bullet \theta_2 \bullet (r_\sigma \rhd t)^{-1} \\
&= \mathsf{Sub}_{\mathsf{l}}(s) \rhd \sigma \rhd t \bullet \alpha^{-1} \bullet (1_{S[s]} \lhd \theta_2^{-1}) \bullet \alpha \bullet (\ell_{t[s]} \rhd \tau) \bullet \theta_2 \bullet (r_\sigma \rhd t)^{-1} \\
&= (\theta_1 \bullet r_\sigma \bullet \ell_\sigma^{-1}) \rhd t \bullet \alpha^{-1} \bullet (1_{S[s]} \lhd \theta_2^{-1}) \bullet \alpha \bullet (\ell_{t[s]} \rhd \tau) \bullet \theta_2 \bullet (r_\sigma \rhd t)^{-1} \\
&= \theta_1 \rhd t \bullet (r_\sigma \bullet \ell_\sigma^{-1}) \rhd t \bullet \alpha^{-1} \bullet (1_{S[s]} \lhd \theta_2^{-1}) \bullet \alpha \bullet (\ell_{t[s]} \rhd \tau) \bullet \theta_2 \bullet (r_\sigma \rhd t)^{-1} \\
&= \theta_1 \rhd t \bullet (r_\sigma \bullet \ell_\sigma^{-1}) \rhd t \bullet \alpha^{-1} \bullet (1_{S[s]} \lhd \theta_2^{-1}) \bullet \ell_{t[s]\tau} \bullet \theta_2 \bullet (r_\sigma \rhd t)^{-1} \\
&= \theta_1 \rhd t \bullet (r_\sigma \bullet \ell_\sigma^{-1}) \rhd t \bullet \alpha^{-1} \bullet \ell_{\sigma \cdot t} \bullet \theta_2^{-1} \bullet \theta_2 \bullet (r_\sigma \rhd t)^{-1} \\
&= \theta_1 \rhd t \bullet (r_\sigma \bullet \ell_\sigma^{-1}) \rhd t \bullet \alpha^{-1} \bullet \ell_{\sigma \cdot t} \bullet (r_\sigma \rhd t)^{-1} \\
&= \theta_1 \rhd t \bullet (r_\sigma \bullet \ell_\sigma^{-1}) \rhd t \bullet \ell_\sigma \rhd t \bullet (r_\sigma \rhd t)^{-1} \\
&= \theta_1 \rhd t
\end{aligned}$$

9.3 Local Substitution

Next we interpret the rules in Figs. 7 and 8, and for that, we use the local opcleaving.

9.3.1 Interpretation of map. We start with map. Suppose we have two substitutions $s, s' : \Delta \longrightarrow \Gamma$ and a 2-cell $\rho : s \Rightarrow s'$. In addition, we assume that we have a type T in context Γ. Our goal is to construct a 1-cell map $T\ \rho : T[s] \longrightarrow T[s']$. From the local opcleaving, we obtain $\mu_\rho(L_s(T))$ over $1_\Delta \cdot s'$ as the pushforward of $L_s(T)$

along $\rho \bullet \ell_{s'}^{-1}$.

$$\begin{tikzcd}
T[s] \arrow[r, bend left, "L_s(T)"] \arrow[r, bend right, dashed, "\mu_\rho(L_s(T))"'] & T
\end{tikzcd}$$

$$\begin{tikzcd}
\Delta \arrow[r, bend left, "s"{name=U}] \arrow[r, bend right, "1_\Delta \cdot s'"'{name=D}] & \Gamma
\arrow[from=U, to=D, Rightarrow, "\rho \bullet \ell_{s'}^{-1}"']
\end{tikzcd}$$

Since $L_{s'}(T)$ is cartesian, we can factor $\mu_\rho(L_s(T))$ through it. From that, we obtain map $T\ \rho$.

$$\begin{tikzcd}
T[s] \arrow[d, dashed, "\text{map } T\ \rho"'] \arrow[dr, "\mu_\rho(L_s(T))"] & \\
T[s'] \arrow[r, "L_{s'}(T)"'] & T
\end{tikzcd}$$

9.3.2 Interpretation of rew-tm. Next, we give the interpretation for the rule rew-tm. Suppose we have two 1-cells $s, s' : \Delta \longrightarrow \Gamma$ and a 2-cell $\rho : s \longrightarrow s'$. In addition, we assume that we have types S, T in context Γ and a 1-cell $\Gamma \mid S \vdash t : T$. Our goal is to construct an invertible 2-cell between $t[s] \bullet (\text{map } T\ \rho)$ and $(\text{map } S\ \rho) \bullet t[s']$. For that it suffices to construct

(1) An invertible 2-cell δ from $1_\Delta \cdot 1_\Delta$ to $1_\Delta \cdot 1_\Delta$, and
(2) An invertible 2-cell $\overline{\sigma}$ from $t[s] \bullet (\text{map } T\ \rho) \bullet L_{s'}(T)$ to $(\text{map } S\ \rho) \bullet t[s'] \bullet L_{s'}(T)$.

Then we define $t[\rho]$ to be $\mathsf{E}(\delta, \overline{\sigma})$. For δ we take $1_{1_\Delta \cdot 1_\Delta}$.

To construct $\overline{\sigma}$, we first note that by construction of $t[s']$, we have an invertible 2-cell

$$(\text{map } S\ \rho) \bullet t[s'] \bullet L_{s'}(T) \cong (\text{map } S\ \rho) \bullet L_{s'}(S) \bullet t$$

Note that by construction of map $S\ \rho$, the following triangle commutes up to invertible 2-cell

$$\begin{tikzcd}
S[s] \arrow[d, "\text{map } S\ \rho"'] \arrow[dr, "\mu_\rho(L_s(S))"] & \\
S[s'] \arrow[r, "L_{s'}(S)"'] & T
\end{tikzcd}$$

Hence, we get

$$\gamma_1 : (\text{map } S\ \rho) \bullet L_{s'}(S) \bullet t \cong \mu_\rho(L_s(S)) \bullet t$$

Similarly, we have an invertible 2-cell map $T\ \rho \cdot L_{s'}(S) \cong \mu_\rho(L_s(T))$, and thus we obtain

$$\gamma_2 : t[s] \bullet (\mathsf{map}\ T\ \rho) \bullet L_{s'}(T) \cong t[s] \bullet \mu_\rho(L_s(T))$$

It thus suffices to construct an invertible 2-cell from $t[s] \bullet \mu_\rho(L_s(T))$ to $\mu_\rho(L_s(S)) \bullet t$. We can depict the situation with the following square:

$$\begin{tikzcd}
S[s] \arrow[r, bend left, "L_s(S)"{name=A}] \arrow[r, bend right, "\mu_\rho(L_s(S))"'{name=B}] \arrow[d, "t[s]"'] & S \arrow[d, "t"] \\
T[s] \arrow[r, bend left, "\mu_\rho(L_s(T))"{name=C}] \arrow[r, bend right, "L_s(T)"'{name=D}] & T
\arrow[from=A, to=B, Rightarrow, "\tau"]
\arrow[from=D, to=C, Rightarrow, "\tau'"]
\end{tikzcd}$$

In this diagram, τ and τ' are the opcartesian 2-cells coming from the opcartesian lift. Note that by construction of $t[s]$, the outer square of this diagram commutes. More precisely, we have an invertible 2-cell $\theta : t[s] \cdot L_s(T) \cong L_s(S) \cdot t$.

To construct an invertible 2-cell from $\mu_\rho(L_s(S)) \cdot t$ to $t[s] \cdot \mu_\rho(L_s(T))$, we are going to construct two opcartesian 2-cells. First, note that we have a 2-cell

$$\tau \rhd t : L_s(S) \cdot t \Rightarrow \mu_\rho(L_s(S)) \cdot t'$$

that lies over $\beta_1 := (\rho \bullet l^{-1}) \rhd 1_\Gamma : s \cdot 1_\Gamma \Rightarrow (1_\Delta \cdot s') \cdot 1_\Gamma$. This 2-cell is opcartesian, because opcartesian 2-cells are closed under right whiskering.

Note that we also have the 2-cell

$$\theta^{-1} \bullet t[s] \lhd \tau : L_s(S) \cdot t \Rightarrow t[s] \cdot \mu_\rho(L_s(T))$$

which lies over $\beta_2 := r \bullet \ell^{-1} \bullet 1_\Gamma \lhd (\rho \bullet \ell^{-1}) : s \cdot 1_\Gamma \Rightarrow 1_\Delta \cdot (1_\Delta \cdot s')$. It is opcartesian, because left whiskering also preserves opcartesian 2-cells.

We also have the following invertible 2-cell:

$$\beta_3 := r \bullet \ell^{-1} : (1_\Delta \cdot s') \cdot 1_\Gamma \cong 1_\Delta \cdot (1_\Delta \cdot s')$$

and note that $\beta_1 \bullet \beta_3 = \beta_2$ by naturality of the unitors. From all of this we get the desired invertible 2-cell $\mu_\rho(L_s(S)) \cdot t \cong t[s] \cdot \mu_\rho(L_s(T))$.

9.3.3 Interpretation of map-id. To interpret map-id, we assume that we have a 1-cell $s : \Delta \to \Gamma$ and a type T in context Γ. Our goal is to construct an invertible 2-cell between the following compositions.

$$\begin{tikzcd}
T[s] \arrow[r, bend left, "1_{T[s]}"] \arrow[r, bend right, "\mathsf{map}\ T\ 1_s"'] & T[s] \arrow[r, "L_s(T)"] & T
\end{tikzcd}$$

To do so, we need to construct

(1) An invertible 2-cell δ from 1_Δ to 1_Δ, and
(2) An invertible 2-cell $\overline{\sigma}$ from map $T\ 1_s \cdot L_s(T)$ to $L_s(T)$.

The desired 2-cell is then defined to be $\mathsf{E}(\delta, \overline{\sigma})$. For δ we take 1_{1_Δ}.

To construct $\overline{\sigma}$, we first note that by construction of map $T\ \rho$, the following triangle commutes up to an invertible 2-cell $\tau : \text{map } T\ \rho \cdot L_{s'}(T) \cong \mu_\rho(L_s(T))$:

$$\begin{tikzcd}
T[s] \arrow[d, "\text{map } T\ \rho"'] \arrow[dr, "\mu_\rho(L_s(T))"] & \\
T[s'] \arrow[r, "L_{s'}(T)"'] \arrow[ur, phantom, "\overset{\tau}{\Rightarrow}"] & T
\end{tikzcd}$$

Next we recall that $\mu_\rho(L_s(T))$ was constructed as the following pushforward:

$$\begin{tikzcd}
T[s] \arrow[r, bend left, "L_s(T)"] \arrow[r, bend right, dashed, "\mu_{1_s}(L_s(T))"'] \arrow[r, phantom, "\Downarrow \theta"] & T
\end{tikzcd}$$

$$\begin{tikzcd}
\Delta \arrow[r, bend left, "s"] \arrow[r, bend right, "1_\Delta \cdot s"'] \arrow[r, phantom, "\Downarrow \ell_s^{-1}"] & \Gamma
\end{tikzcd}$$

Since ℓ_s^{-1} is invertible, the opcartesian lift θ is invertible as well. Hence, we define $\overline{\sigma} := \tau \bullet \theta^{-1}$. We interpret map-comp similarly.

9.3.4 Interpretation of map-lwhisker. To interpret map-lwhisker and map-rwhisker, we use the same idea, and we need to use that we assumed that opcartesian 2-cells are closed under whiskering. We give the precise details for map-lwhisker. Suppose we have a 1-cell $s : \mathrm{E} \longrightarrow \Delta$, a 2-cell $\rho : s' \Rightarrow s''$ where $s', s'' : \Delta \longrightarrow \Gamma$, and a type T in context Γ. Our goal is to construct an invertible 2-cell from map $T\ (s \lhd \rho)$ to $\mathsf{sub}_{\mathsf{comp}}^{-1}(s, s') \cdot (\text{map } T\ \rho)[s] \cdot \mathsf{sub}_{\mathsf{comp}}(s, s'')$. Since map $T\ \rho$ was constructed as a cartesian factorization, it suffices to construct

(1) An invertible 2-cell δ from 1_Δ to $1_\Delta \cdot 1_\Delta \cdot 1_\Delta$, and
(2) An invertible 2-cell $\overline{\sigma}$ from map $T\ (s \lhd \rho) \cdot L_{s \cdot s''}(T)$ to $\mathsf{sub}_{\mathsf{comp}}^{-1}(s, s') \cdot (\text{map } T\ \rho)[s] \cdot \mathsf{sub}_{\mathsf{comp}}(s, s'') \cdot L_{s \cdot s''}(T)$.

With those in place, the desired invertible 2-cell is defined to be $\mathsf{E}(\delta, \overline{\sigma})$. We define δ to be $\ell_{1_\Delta}^{-1} \cdot \ell_{1_\Delta \cdot 1_\Delta}^{-1}$.

For the construction of $\overline{\sigma}$, let us start by recalling the construction of map $T\ (s \lhd \rho)$.

$$\begin{tikzcd}
T[s \cdot s'] \arrow[d, "\text{map } T\ (s \lhd \rho)"'] \arrow[dr, shift left, "L_{s \cdot s'}(T)"] \arrow[dr, shift right, "\mu_{s \lhd \rho}(L_{s \cdot s'}(T))"'] & \\
T[s \cdot s''] \arrow[r, "L_{s \cdot s''}(T)"'] & T
\end{tikzcd}$$

Next we inspect the definition of $\mathsf{sub}_{\mathsf{comp}}(s, s'')$.

$$\begin{tikzcd}
T[s \cdot s''] \arrow[rrd, "L_{s \cdot s''}(T)"] & & \\
T[s''][s] \arrow[u, "\mathsf{sub}_{\mathsf{comp}}(s{,}s'')"] \arrow[r, "L_s(T[s''])"'] & T[s''] \arrow[r, "L_{s''}(T)"'] \arrow[ul, phantom, "\cong"] & T
\end{tikzcd}$$

From this, we already get an invertible 2-cell

$$\gamma_1 : \mathsf{sub}^{-1}_{\mathsf{comp}}(s, s') \cdot (\mathsf{map}\ T\ \rho)[s] \cdot \mathsf{sub}_{\mathsf{comp}}(s, s'') \cdot L_{s \cdot s''}(T) \cong \mathsf{sub}^{-1}_{\mathsf{comp}}(s, s') \cdot (\mathsf{map}\ T\ \rho)[s] \cdot L_s(T[s'']) \cdot L_{s''}(T)$$

We also inspect the construction of $(\mathsf{map}\ T\ \rho)[s]$, from which we get the following invertible 2-cell:

$$\begin{tikzcd}
T[s'][s] \arrow[r, "L_s(T[s'])"] \arrow[d, "(\mathsf{map}\ T\ \rho)[s]"'] & T[s'] \arrow[d, "\mathsf{map}\ T\ \rho"] \\
T[s''][s] \arrow[r, "L_s(T[s''])"'] \arrow[ur, Rightarrow, "\cong"] & T[s'']
\end{tikzcd}$$

As such, we get another invertible 2-cell

$$\gamma_2 : \mathsf{sub}^{-1}_{\mathsf{comp}}(s, s') \cdot (\mathsf{map}\ T\ \rho)[s] \cdot L_s(T[s'']) \cdot L_{s''}(T) \cong \mathsf{sub}^{-1}_{\mathsf{comp}}(s, s') \cdot L_s(T[s']) \cdot \mathsf{map}\ T\ \rho \cdot L_{s''}(T).$$

Let us recall the construction of $\mathsf{map}\ T\ \rho$ as well.

$$\begin{tikzcd}
T[s'] \arrow[d, "\mathsf{map}\ T\ \rho"'] \arrow[dr, "\mu_\rho(L_{s'}(T))"] & \\
T[s''] \arrow[r, "L_{s''}(T)"'] & T \arrow[phantom, ul, "\cong"]
\end{tikzcd}$$

From this, we obtain another invertible 2-cell

$$\gamma_3 : \mathsf{sub}^{-1}_{\mathsf{comp}}(s, s') \cdot (\mathsf{map}\ T\ \rho)[s] \cdot L_s(T[s'']) \cdot L_{s''}(T) \cong \mathsf{sub}^{-1}_{\mathsf{comp}}(s, s') \cdot L_s(T[s']) \cdot \mu_\rho(L_{s''}(T)).$$

Hence, we achieve our goal if we construct an invertible 2-cell

$$\gamma_4 : \mathsf{sub}^{-1}_{\mathsf{comp}}(s, s') \cdot L_s(T[s']) \cdot \mu_\rho(L_{s'}(T)) \cong \mu_{s \triangleleft \rho}(L_{s \cdot s'}(T)).$$

We can depict this situation as follows

$$\begin{tikzcd}
T[s \cdot s'] \arrow[rr, bend left, "L_{s \cdot s'}(T)"] \arrow[rr, bend right, "\mu_{s \triangleleft \rho}(L_{s \cdot s'}(T))"'] \arrow[d, "\mathsf{sub}^{-1}_{\mathsf{comp}}(s,s')"'] & \Downarrow \tau' & T \arrow[d, equal] \\
T[s'][s] \arrow[r, "L_s(T[s'])"'] & T[s'] \arrow[r, bend left, "L_{s'}(T)"] \arrow[r, bend right, "\mu_\rho(L_{s'}(T))"'] \arrow[r, phantom, "\Downarrow \tau"] & T
\end{tikzcd}$$

$$\begin{tikzcd}
\mathrm{E} \arrow[r, "s"'] & \Delta \arrow[r, bend left, "s'"] \arrow[r, bend right, "s''"'] \arrow[r, phantom, "\Downarrow \rho"] & \Gamma
\end{tikzcd}$$

Here τ and τ' are the 2-cells coming from the opcartesian lifts.

Since whiskering preserves opcartesian 2-cells, the following 2-cell is opcartesian:

$$(\mathsf{sub}^{-1}_{\mathsf{comp}}(s,s') \cdot L_s(T[s'])) \triangleleft \tau : \mathsf{sub}^{-1}_{\mathsf{comp}}(s,s') \cdot L_s(T[s']) \cdot L_{s'}(T) \Rightarrow \mathsf{sub}^{-1}_{\mathsf{comp}}(s,s') \cdot L_s(T[s']) \cdot \mu_\rho(L_{s'}(T))$$

Now we unfold the definition of $\mathsf{sub}_{\mathsf{comp}}(s,s')$, and from that we get the following invertible 2-cell:

$$\begin{tikzcd}
T[s \cdot s'] \arrow[rrd, bend left, "L_{s \cdot s'}(T)"] \arrow[d, "\mathsf{sub}^{-1}_{\mathsf{comp}}(s,s')"'] & \Downarrow \cong & \\
T[s'][s] \arrow[r, "L_s(T[s'])"'] & T[s'] \arrow[r, "L_{s'}(T)"'] & T
\end{tikzcd}$$

Since invertible 2-cells are opcartesian and opcartesian 2-cells are closed under composition, we get an opcartesian 2-cell

$$\theta : L_{s \cdot s'}(T) \Rightarrow \mathsf{sub}^{-1}_{\mathsf{comp}}(s,s') \cdot L_s(T[s']) \cdot \mu_\rho(L_{s'}(T)).$$

Since we already had an opcartesian 2-cell $\tau' : L_{s \cdot s'}(T) \Rightarrow \mu_{s \triangleleft \rho}(L_{s \cdot s'}(T))$, we get an invertible 2-cell between the codomains of θ and τ'. Hence, we obtain an invertible 2-cell

$$\gamma_5 : \mathsf{sub}^{-1}_{\mathsf{comp}}(s,s') \cdot L_s(T[s']) \cdot \mu_\rho(L_{s'}(T)) \cong \mu_{s \triangleleft \rho}(L_{s \cdot s'}(T))$$

By chaining all the invertible 2-cells, we get the desired 2-cell $\overline{\sigma}$.

We omit the description of the verification of several equalities. All in all, we can state the following theorem:

Theorem 9.1 (Soundness). *We can interpret* BTT *in every comprehension bicategory.*

10 VARIATIONS ON SYNTAX

BTT is a very complicated type theory, and might be unfeasible to implement or use in practice. Its main purpose is to serve as a framework for studying specialized syntax and the corresponding semantics. Based on a user's goal, they might adopt one of the following simplifications:

V1: **Splitness.** We could assume the comprehension bicategory is split; thus, the rules sub-id and sub-comp would collapse into ordinary equalities. For this, an equality judgment on types would be added to BTT.

V2: **Strictness.** The rules in Figs. 1 and 2 are aimed at bicategories. When working with strict 2-categories instead, the unitors and associators would become equalities, and as a result, rules for inverse laws, naturality, and the pentagon and triangle equations are not needed.

V3: **Terms.** If we assume that we have a unit type, then we can simplify the judgment for terms. Instead of looking at judgments of the shape $\Gamma \mid S \vdash t : T$, we can take S to be the unit type, thus recovering the judgment $\Gamma \vdash t : T$ for terms in MLTT. Semantically, this amounts to assuming that the fiber in D above any object in B has a terminal object.

V4: **Undirected TT.** We could add a rule postulating inverses of reductions. Semantically, this would amount to working in groupoid-enriched categories.

V5: **Proof-irrelevant reductions.** Our syntax, and the semantics, allow us to distinguish parallel reductions (2-cells). We could instead "truncate" them, by moving to proof-irrelevant reductions, making the judgmental equality on them superfluous. This would yield a directed analog to the judgments of MLTT; semantically, it corresponds to working in poset-enriched categories instead of general bicategories.

V6: **Conflating terms and substitutions.** One could equate terms and substitutions that are left-inverse to projections.

We have developed comprehension bicategories with the explicit goal of encompassing previously defined interpretations of higher-dimensional and directed type theory. In the following remarks we summarize the relationship with two previous works.

Remark 10.1 (Comparison to Garner's work [19])**.** To summarize the differences of Garner's comprehension 2-categories [19] to our comprehension bicategories, the former are full, strict, split, undirected (*i. e.,* locally groupoidal), and incorporate type constructors.

Remark 10.2 (Comparison to Licata and Harper's work)**.** Licata and Harper's interpretation of their two-dimensional type theory into categories [28] takes place in the comprehension bicategory constructed in Example 5.3. Specifically, Licata and Harper interpret a type in context Γ as a (strict 1-)functor from the category interpreting Γ into a 1-category CAT of categories. Formally, they thus consider the slice bicategory Cat/CAT, where Cat is the bicategory of categories, in which CAT is assumed to be a 0-cell. The "domain" pseudofunctor $\mathsf{dom} : \mathsf{Cat}/\mathsf{CAT} \longrightarrow \mathsf{Cat}$ carries the structure of a global cleaving and local opcleaving; this is more generally the case for any "domain" pseudofunctor $\mathsf{dom} : \mathsf{B}/a \longrightarrow \mathsf{B}$ (DomainCleaving.v). To construct the comprehension pseudofunctor, one needs to use specifics of the category of strict categories, namely the Grothendieck construction of functors into CAT.

11 CONCLUSION

We have introduced the notion of comprehension bicategory for the interpretation of two-dimensional and directed type theory. From this semantic notion, we have extracted a two-dimensional core syntax for dependent types, terms, and reductions, and an interpretation of that syntax in comprehension bicategories. Our work is very general; it allows for the modelling of the structural rules of previous suggestions for directed type theory. Furthermore, it can be used as a framework for defining and studying more specialized syntax and semantics, in lockstep.

As outlined in Section 1.1, in separate work we are going to extend our structural rules with variances and a suitable hom-type former à la North [32] on top.

Garner [19] proves completeness of 2-truncated Martin-Löf type theory with respect to semantics in comprehension 2-categories. We would like to give a similar completeness result with respect to our comprehension bicategories.

ACKNOWLEDGMENTS

We gratefully acknowledge the work by the Coq development team in providing the Coq proof assistant and surrounding infrastructure, as well as their support in keeping UniMath compatible with Coq. We are very grateful to Dan Licata and Bob Harper for their help in understanding their interpretation and how it fits into our framework. We thank the anonymous referees of the LICS version for their insightful comments. This work was partially funded by EPSRC under agreement number EP/T000252/1. This material is based upon work supported by the Air Force Office of Scientific Research under award number FA9550-21-1-0334.

COMPETING INTERESTS

Ahrens is employed at Delft University of Technology. North is employed at University of Pennsylvania, and is about to start a job at Utrecht University. Van der Weide is employed at Radboud University.

REFERENCES

[1] Benedikt Ahrens, Dan Frumin, Marco Maggesi, and Niels van der Weide. 2019. Bicategories in Univalent Foundations. In *4th International Conference on Formal Structures for Computation and Deduction (FSCD 2019) (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 131)*, Herman Geuvers (Ed.). Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 5:1–5:17. https://doi.org/10.4230/LIPIcs.FSCD.2019.5

[2] Benedikt Ahrens, Dan Frumin, Marco Maggesi, Niccolò Veltri, and Niels van der Weide. 2022. Bicategories in univalent foundations. *Mathematical Structures in Computer Science* (2022), 1–38. https://doi.org/10.1017/S0960129522000032

[3] Benedikt Ahrens, Krzysztof Kapulkin, and Michael Shulman. 2015. Univalent categories and the Rezk completion. *Math. Struct. Comput. Sci.* 25, 5 (2015), 1010–1039. https://doi.org/10.1017/S0960129514000486

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

[5] Benedikt Ahrens, Paige Randall North, and Niels van der Weide. 2022. Semantics for two-dimensional 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, 12:1–12:14. https://doi.org/10.1145/3531130.3533334

[6] Krzysztof Bar, Aleks Kissinger, and Jamie Vicary. 2018. Globular: an online proof assistant for higher-dimensional rewriting. *Log. Methods Comput. Sci.* 14, 1 (2018). https://doi.org/10.23638/LMCS-14(1:8)2018

[7] Jean Bénabou. 1967. Introduction to bicategories. In *Reports of the Midwest Category Seminar*. Springer Berlin Heidelberg, Berlin, Heidelberg, 1–77. https://doi.org/10.1007/BFb0074299

[8] Thibaut Benjamin, Eric Finster, and Samuel Mimram. 2021. Globular weak ω-categories as models of a type theory. *CoRR* (2021). arXiv:2106.04475

[9] Benno van den Berg and Richard Garner. 2011. Types are weak ω-groupoids. *Proceedings of the London Mathematical Society* 102, 2 (2011), 370–394. https://doi.org/10.1112/plms/pdq026

[10] Guillaume Brunerie. 2016. *On the homotopy groups of spheres in homotopy type theory.* Ph. D. Dissertation. Université Nice Sophia Antipolis. arXiv:1606.05916

[11] Ulrik Buchholtz and Jonathan Weinberger. 2021. Synthetic fibered $(\infty, 1)$-category theory. *CoRR* abs/2105.01724 (2021). arXiv:2105.01724

[12] Mitchell Buckley. 2014. Fibred 2-categories and bicategories. *Journal of Pure and Applied Algebra* 218, 6 (2014), 1034–1074. https://doi.org/10.1016/j.jpaa.2013.11.002

[13] Thierry Coquand, Bassel Mannaa, and Fabian Ruch. 2017. Stack semantics of type theory. In *32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017, Reykjavik, Iceland, June 20-23, 2017.* IEEE Computer Society, 1–11. https://doi.org/10.1109/LICS.2017.8005130

[14] Greta Coraglia and Ivan Di Liberti. 2021. Context, Judgement, Deduction. https://doi.org/10.48550/ARXIV.2111.09438

[15] Lisbeth Fajstrup, Eric Goubault, Emmanuel Haucourt, Samuel Mimram, and Martin Raussen. 2016. *Directed Algebraic Topology and Concurrency.* Springer. https://doi.org/10.1007/978-3-319-15398-8

[16] Eric Finster and Samuel Mimram. 2017. A type-theoretical definition of weak ω-categories. In *32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017, Reykjavik, Iceland, June 20-23, 2017.* IEEE Computer Society, 1–12. https://doi.org/10.1109/LICS.2017.8005124

[17] Eric Finster, David Reutter, Alex Rice, and Jamie Vicary. 2022. A Type Theory for Strictly Unital -Categories. In *37th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS) (LICS '22).* https://doi.org/10.1145/3531130.3533363

[18] Marcelo Fiore and Philip Saville. 2019. A type theory for cartesian closed bicategories (Extended Abstract). In *34th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2019, Vancouver, BC, Canada, June 24-27, 2019.* IEEE, 1–13. https://doi.org/10.1109/LICS.2019.8785708

[19] Richard Garner. 2009. Two-dimensional models of type theory. *Math. Struct. Comput. Sci.* 19, 4 (2009), 687–736. https://doi.org/10.1017/S0960129509007646

[20] John W. Gray. 1966. Fibred and Cofibred Categories. In *Proceedings of the Conference on Categorical Algebra*, S. Eilenberg, D. K. Harrison, S. MacLane, and H. Röhrl (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 21–83. https://doi.org/10.1007/978-3-642-99902-4_2

[21] Michael Nicholas Gurski. 2006. *An algebraic theory of tricategories.* Ph. D. Dissertation. University of Chicago.

[22] Claudio Hermida. 1999. Some properties of Fib as a fibred 2-category. *Journal of Pure and Applied Algebra* 134, 1 (1999), 83–109. https://doi.org/10.1016/S0022-4049(97)00129-1

[23] Tom Hirschowitz. 2013. Cartesian closed 2-categories and permutation equivalence in higher-order rewriting. *Log. Methods Comput. Sci.* 9, 3 (2013). https://doi.org/10.2168/LMCS-9(3:10)2013

[24] Martin Hofmann and Thomas Streicher. 1994. The Groupoid Model Refutes Uniqueness of Identity Proofs. In *Proceedings of the Ninth Annual Symposium on Logic in Computer Science (LICS '94), Paris, France, July 4-7, 1994.* IEEE Computer Society, 208–212. https://doi.org/10.1109/LICS.1994.316071

[25] André Joyal and Ross Street. 1993. Pullbacks equivalent to pseudopullbacks. *Cahiers de topologie et géométrie différentielle catégoriques* 34, 2 (1993), 153–156. http://eudml.org/doc/91518

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

[27] Daniel R. Licata. 2011. *Dependently Typed Programming with Domain-Specific Logics.* Ph. D. Dissertation. USA. Advisor(s) Harper, Robert. https://doi.org/10.5555/2338432 AAI3476124.

[28] Daniel R. Licata and Robert Harper. 2011. 2-Dimensional Directed Type Theory. In *Twenty-seventh Conference on the Mathematical Foundations of Programming Semantics, MFPS 2011, Pittsburgh, PA, USA, May 25-28, 2011 (Electronic Notes in Theoretical Computer Science, Vol. 276)*, Michael W. Mislove and Joël Ouaknine (Eds.). Elsevier, 263–289. https://doi.org/10.1016/j.entcs.2011.09.026

[29] Daniel R. Licata and Robert Harper. 2012. Canonicity for 2-dimensional type theory. In *Proceedings of the 39th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2012, Philadelphia, Pennsylvania, USA, January 22-28, 2012*, John Field and Michael Hicks (Eds.). ACM, 337–348. https://doi.org/10.1145/2103656.2103697

[30] Fosco Loregian and Emily Riehl. 2020. Categorical notions of fibration. *Expositiones Mathematicae* 38, 4 (2020), 496–514. https://doi.org/10.1016/j.exmath.2019.02.004

[31] Peter LeFanu Lumsdaine. 2010. Weak omega-categories from intensional type theory. *Log. Methods Comput. Sci.* 6, 3 (2010). https://doi.org/10.2168/LMCS-6(3:24)2010

[32] 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. https://doi.org/10.1016/j.entcs.2019.09.012

[33] Andreas Nuyts. 2015. *Towards a Directed Homotopy Type Theory based on 4 Kinds of Variance.* Master's thesis. KU Leuven. https://anuyts.github.io/files/mathesis.pdf

[34] David Reutter and Jamie Vicary. 2019. High-level methods for homotopy construction in associative n-categories. In *34th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2019, Vancouver, BC, Canada, June 24-27, 2019.* IEEE, 1–13. https://doi.org/10.1109/LICS.2019.8785895

[35] Emily Riehl and Michael Shulman. 2017. A type theory for synthetic ∞-categories. *Higher Structures* 1, 1 (May 2017), 147–224. https://journals.mq.edu.au/index.php/higher_structures/article/view/36

[36] Robert A. G. Seely. 1987. Modelling Computations: A 2-Categorical Framework. In *Proceedings of the Symposium on Logic in Computer Science (LICS '87), Ithaca, New York, USA, June 22-25, 1987.* IEEE Computer Society, 65–71.

[37] Michael Shulman. 2010. Functorially Dependent Types. https://ncatlab.org/michaelshulman/show/functorially+dependent+types

[38] Michael Shulman. 2011. Internal Logic of a 2-Category. https://ncatlab.org/michaelshulman/show/internal+logic+of+a+2-category

[39] Michael Shulman. 2012. 2-Categorical Logic. https://ncatlab.org/michaelshulman/show/2-categorical+logic

[40] Michael Shulman. 2019. Fibrational Slice. https://ncatlab.org/michaelshulman/show/fibrational+slice

[41] Ross Street. 1980. Fibrations in bicategories. *Cahiers de topologie et géométrie différentielle catégoriques* 21, 2 (1980), 111–160. http://archive.numdam.org/item/CTGDC_1980__21_2_111_0/

[42] Ross Street. 1982. Characterization of bicategories of stacks. In *Category Theory (Lecture Notes in Mathematics, Vol. 962)*, K.H. Kamps, D. Pumplün, and W. Tholen (Eds.). Springer. https://doi.org/10.1007/BFb0066909

[43] Nicolas Tabareau. 2011. Aspect oriented programming: a language for 2-categories. In *Proceedings of the 10th international workshop on Foundations of aspect-oriented languages, FOAL 2011, Porto de Galinhas, Brazil, March 21-25, 2011*, Hridesh Rajan (Ed.). ACM, 13–17. https://doi.org/10.1145/1960510.1960514

[44] Paul Taylor. 1999. *Practical Foundations of Mathematics*. Cambridge studies in advanced mathematics, Vol. 59. Cambridge University Press.

[45] The Coq Development Team. 2022. *The Coq Proof Assistant*. https://doi.org/10.5281/zenodo.5846982

[46] Vladimir Voevodsky, Benedikt Ahrens, Daniel Grayson, et al. 2022. UniMath — a computer-checked library of univalent mathematics. Available at https://unimath.org. https://github.com/UniMath/UniMath

[47] Matthew Z. Weaver and Daniel R. Licata. 2020. A Constructive Model of Directed Univalence in Bicubical Sets. In *LICS '20: 35th Annual ACM/IEEE Symposium on Logic in Computer Science, Saarbrücken, Germany, July 8-11, 2020*, Holger Hermanns, Lijun Zhang, Naoki Kobayashi, and Dale Miller (Eds.). ACM, 915–928. https://doi.org/10.1145/3373718.3394794

$$\frac{\Gamma \vdash T\ \text{type}}{\Gamma \mid T \vdash 1_T : T} \qquad \frac{\Gamma \vdash S,T\ \text{type} \quad \Gamma \mid S \vdash t : T}{\Gamma \mid S \vdash 1_t : t \rightsquigarrow t : T} \qquad \frac{\Gamma \vdash S,T\ \text{type} \quad \Gamma \mid S \vdash t,t' : T \quad \Gamma \mid S \vdash \rho : t \rightsquigarrow t' : T}{\Gamma \mid S \vdash \rho \equiv \rho : t \rightsquigarrow t' : T}$$

$$\frac{\Gamma \vdash R,S,T\ \text{type} \quad \Gamma \mid R \vdash s : S \quad \Gamma \mid S \vdash t : T}{\Gamma \mid R \vdash st : T} \qquad \frac{\Gamma \vdash R,S,T\ \text{type} \quad \Gamma \mid R \vdash s : S \quad \Gamma \mid S \vdash t,t' : T \quad \Gamma \mid S \vdash \rho : t \rightsquigarrow t' : T}{\Gamma \mid R \vdash s \triangleleft \rho : st \rightsquigarrow st' : T}$$

$$\frac{\Gamma \vdash R,S,T\ \text{type} \quad \Gamma \mid R \vdash s,s' : S \quad \Gamma \mid S \vdash t : T \quad \Gamma \mid R \vdash \sigma : s \rightsquigarrow s' : S}{\Gamma \mid R \vdash \sigma \triangleright t : st \rightsquigarrow s't : T}$$

$$\frac{\Gamma \vdash S,T\ \text{type} \quad \Gamma \mid S \vdash t,t',t'' : T \quad \Gamma \mid S \vdash \rho : t \rightsquigarrow t' : T \quad \Gamma \mid S \vdash \sigma : t' \rightsquigarrow t'' : T}{\Gamma \mid S \vdash \rho \bullet \sigma : t \rightsquigarrow t'' : T}$$

$$\frac{\Gamma \vdash R,S,T\ \text{type} \quad \Gamma \mid R \vdash s : S \quad \Gamma \mid S \vdash t : T}{\begin{array}{c}\Gamma \mid R \vdash 1_s \triangleright t \equiv 1_{st} : st \rightsquigarrow st : T \\ \Gamma \mid R \vdash s \triangleleft 1_t \equiv 1_{st} : st \rightsquigarrow st : T\end{array}}$$

$$\frac{\Gamma \vdash R,S,T\ \text{type} \quad \Gamma \mid R \vdash s : S \quad \Gamma \mid S \vdash t,t',t'' : T \quad \Gamma \mid S \vdash \rho : t \rightsquigarrow t' : T \quad \Gamma \mid S \vdash \rho' : t' \rightsquigarrow t'' : T}{\Gamma \mid R \vdash (s \triangleleft \rho) \bullet (s \triangleleft \rho') \equiv s \triangleleft (\rho \bullet \rho') : st \rightsquigarrow st'' : T}$$

$$\frac{\Gamma \vdash R,S,T\ \text{type} \quad \Gamma \mid R \vdash s,s',s'' : S \quad \Gamma \mid S \vdash t : T \quad \Gamma \mid R \vdash \sigma : s \rightsquigarrow s' : S \quad \Gamma \mid R \vdash \sigma' : s' \rightsquigarrow s'' : S}{\Gamma \mid R \vdash (\sigma \triangleright t) \bullet (\sigma' \triangleright t) \equiv (\sigma \bullet \sigma') \triangleright t : st \rightsquigarrow s''t : T}$$

$$\frac{\Gamma \vdash R,S,T\ \text{type} \quad \Gamma \mid R \vdash s,s' : S \quad \Gamma \mid S \vdash t,t' : T \quad \Gamma \mid R \vdash \sigma : s \rightsquigarrow s' : S \quad \Gamma \mid S \vdash \rho : t \rightsquigarrow t' : T}{\Gamma \mid R \vdash (\sigma \triangleright t) \bullet (s' \triangleleft \rho) \equiv (s \triangleleft \rho) \bullet (\sigma \triangleright t') : st \rightsquigarrow s't' : T}$$

$$\frac{\Gamma \vdash S,T\ \text{type} \quad \Gamma \mid S \vdash s : T}{\begin{array}{c}\Gamma \mid S \vdash \ell_s : 1_S s \overset{\sim}{\rightsquigarrow} s : T \\ \Gamma \mid S \vdash r_s : s 1_T \overset{\sim}{\rightsquigarrow} s : T\end{array}} \qquad \frac{\Gamma \vdash Q,R,S,T\ \text{type} \quad \Gamma \mid Q \vdash r : R \quad \Gamma \mid R \vdash s : S \quad \Gamma \mid S \vdash t : T}{\Gamma \mid Q \vdash \alpha_{r,s,t} : r(st) \overset{\sim}{\rightsquigarrow} (rs)t : T}$$

$$\frac{\Gamma \vdash S,T\ \text{type} \quad \Gamma \mid S \vdash s,s' : T \quad \Gamma \mid S \vdash \rho : s \rightsquigarrow s' : T}{\begin{array}{c}\Gamma \mid S \vdash r_s^{-1} \bullet (\rho \triangleright 1_T) \bullet r_{s'} \equiv \rho : s \rightsquigarrow s' : T \\ \Gamma \mid S \vdash \ell_s^{-1} \bullet (1_S \triangleleft \rho) \bullet \ell_{s'} \equiv \rho : s \rightsquigarrow s' : T\end{array}}$$

$$\frac{\Gamma \vdash Q,R,S,T\ \text{type} \quad \Gamma \mid Q \vdash s : R \quad \Gamma \mid R \vdash t : S \quad \Gamma \mid S \vdash u,u' : T \quad \Gamma \mid S \vdash \rho : u \rightsquigarrow u' : T}{\Gamma \mid Q \vdash \alpha_{s,t,u}^{-1} \bullet (s \triangleleft (t \triangleleft \rho)) \bullet \alpha_{s,t,u'} \equiv st \triangleleft \rho : (st)u \rightsquigarrow (st)u' : T}$$

$$\frac{\Gamma \vdash Q,R,S,T\ \text{type} \quad \Gamma \mid Q \vdash s : R \quad \Gamma \mid R \vdash t,t' : S \quad \Gamma \mid S \vdash u : T \quad \Gamma \mid R \vdash \rho : t \rightsquigarrow t' : S}{\Gamma \mid Q \vdash \alpha_{s,t,u}^{-1} \bullet (s \triangleleft (\rho \triangleright u)) \bullet \alpha_{s,t',u} \equiv (s \triangleleft \rho) \triangleright u : (st)u \rightsquigarrow (st')u : T}$$

$$\frac{\Gamma \vdash Q,R,S,T\ \text{type} \quad \Gamma \mid Q \vdash s,s' : R \quad \Gamma \mid R \vdash t : S \quad \Gamma \mid S \vdash u : T \quad \Gamma \mid Q \vdash \rho : s \rightsquigarrow s' : R}{\Gamma \mid Q \vdash \alpha_{s,t,u}^{-1} \bullet (\rho \triangleright tu) \bullet \alpha_{s',t,u} \equiv (\rho \triangleright t) \triangleright u : (st)u \rightsquigarrow (s't)u : T}$$

$$\frac{\Gamma \vdash S,T\ \text{type} \quad \Gamma \mid S \vdash s,s' : T \quad \Gamma \mid S \vdash \rho : s \rightsquigarrow s' : T}{\begin{array}{c}\Gamma \mid S \vdash 1_s \bullet \rho \equiv \rho : s \rightsquigarrow s' : T \\ \Gamma \mid S \vdash \rho \bullet 1_{s'} \equiv \rho : s \rightsquigarrow s' : T\end{array}}$$

$$\frac{\Gamma \vdash S,T\ \text{type} \quad \Gamma \mid S \vdash s,s',s'',s''' : T \quad \Gamma \mid S \vdash \rho : s \rightsquigarrow s' : T \quad \Gamma \mid S \vdash \sigma : s' \rightsquigarrow s'' : T \quad \Gamma \mid S \vdash \tau : s'' \rightsquigarrow s''' : T}{\Gamma \mid S \vdash \rho \bullet (\sigma \bullet \tau) \equiv (\rho \bullet \sigma) \bullet \tau : s \rightsquigarrow s''' : T}$$

$$\frac{\Gamma \vdash R,S,T\ \text{type} \quad \Gamma \mid R \vdash s : S \quad \Gamma \mid S \vdash t : T}{\Gamma \mid R \vdash \alpha_{s,1_S,t} \bullet (r_s \triangleright t) \equiv s \triangleleft \ell_t : s(1_S t) \rightsquigarrow st : T}$$

$$\frac{\Gamma \vdash P,Q,R,S,T\ \text{type} \quad \Gamma \mid P \vdash q : Q \quad \Gamma \mid Q \vdash r : R \quad \Gamma \mid R \vdash s : S \quad \Gamma \mid S \vdash t : T}{\Gamma \mid P \vdash \alpha_{q,r,st} \bullet \alpha_{qr,s,t} \equiv (q \triangleleft \alpha_{r,s,t}) \bullet \alpha_{q,rs,t} \bullet (\alpha_{q,r,s} \triangleright t) : q(r(st)) \rightsquigarrow ((qr)s)t : T}$$

Fig. 2. Rules for the the bicategory of types

$$\dfrac{\Gamma\ \text{ctx} \qquad \Gamma \vdash T\ \text{type}}{\begin{array}{c}\Gamma.T\ \text{ctx}\\ \Gamma.T \vdash \pi_{\Gamma.T} : \Gamma\end{array}}\ \text{extend-con-Ty} \qquad \dfrac{\Gamma\ \text{ctx} \qquad \Gamma \vdash S, T\ \text{type} \qquad \Gamma \mid S \vdash t : T}{\begin{array}{c}\Gamma.S \vdash \Gamma.t : \Gamma.T\\ \Gamma.S \vdash c_{\Gamma.t} : \pi_{\Gamma.S} \mathrel{\tilde{\rightsquigarrow}} (\Gamma.t)\pi_{\Gamma.T} : \Gamma\end{array}}\ \text{extend-con-Tm}$$

$$\dfrac{\Gamma\ \text{ctx} \qquad \Gamma \vdash S, T\ \text{type} \qquad \Gamma \mid S \vdash t, t' : T \qquad \Gamma \mid S \vdash r : t \rightsquigarrow t' : T}{\begin{array}{c}\Gamma.S \vdash \Gamma.r : \Gamma.t \rightsquigarrow \Gamma.t' : \Gamma.T\\ \Gamma.S \vdash c_{\Gamma.t'} \equiv c_{\Gamma.t} \bullet (\Gamma.r \triangleright \pi_{\Gamma.T}) : \pi_{\Gamma.S} \rightsquigarrow (\Gamma.t')\pi_{\Gamma.T} : \Gamma\end{array}}\ \text{extend-con-Red}$$

$$\dfrac{\Gamma\ \text{ctx} \qquad \Gamma \vdash T\ \text{type}}{\begin{array}{c}\Gamma.T \vdash \chi^{\text{id}}_{T} : \Gamma.1_T \mathrel{\tilde{\rightsquigarrow}} 1_{\Gamma.T} : \Gamma.T\\ \Gamma.T \vdash c_{\Gamma.1_T} \bullet (\chi^{\text{id}}_{T} \triangleright \pi_{\Gamma.T}) \bullet \ell_{\pi_{\Gamma.T}} \equiv 1_{\pi_{\Gamma.T}} : \pi_{\Gamma.T} \rightsquigarrow \pi_{\Gamma.T} : \Gamma\end{array}}\ \text{extend-con-id}$$

$$\dfrac{\Gamma\ \text{ctx} \qquad \Gamma \vdash R, S, T\ \text{type} \qquad \Gamma \mid R \vdash s : S \qquad \Gamma \mid S \vdash t : T}{\begin{array}{c}\Gamma.R \vdash \chi^{\text{comp}}_{s,t} : (\Gamma.s)(\Gamma.t) \mathrel{\tilde{\rightsquigarrow}} \Gamma.(st) : \Gamma.T\\ \Gamma.R \vdash c_{\Gamma.s} \bullet (\Gamma.s \triangleleft c_{\Gamma.t}) \bullet \alpha_{\Gamma.s,\Gamma.t,\pi_{\Gamma.T}} \bullet (\chi^{\text{comp}}_{s,t} \triangleright \pi_{\Gamma.T}) \equiv c_{\Gamma.st} : \pi_{\Gamma.R} \rightsquigarrow (\Gamma.st)\pi_{\Gamma.T} : \Gamma\end{array}}\ \text{extend-con-comp}$$

$$\dfrac{\Gamma\ \text{ctx} \qquad \Gamma \vdash S, T\ \text{type} \qquad \Gamma \mid S \vdash t : T}{\Gamma.S \vdash \Gamma.1_t \equiv 1_{\Gamma.t} : \Gamma.t \rightsquigarrow \Gamma.t : \Gamma.T}$$

$$\dfrac{\Gamma\ \text{ctx} \qquad \Gamma \vdash S, T\ \text{type} \qquad \Gamma \mid S \vdash t, t', t'' : T \qquad \Gamma \mid S \vdash \rho : t \rightsquigarrow t' : T \qquad \Gamma \mid S \vdash \rho' : t' \rightsquigarrow t'' : T}{\Gamma.S \vdash \Gamma.(\rho \bullet \rho') \equiv \Gamma.\rho \bullet \Gamma.\rho' : \Gamma.t \rightsquigarrow \Gamma.t'' : \Gamma.T}$$

$$\dfrac{\Gamma\ \text{ctx} \qquad \Gamma \vdash S, T\ \text{type} \qquad \Gamma \mid S \vdash t : T}{\begin{array}{c}\Gamma.S \vdash \left(\chi^{\text{id}}_{S} \triangleright (\Gamma.t)\right) \bullet \ell_{\Gamma.t} \equiv \chi^{\text{comp}}_{1_S,t} \bullet (\Gamma.\ell_t) : (\Gamma.1_S)(\Gamma.t) \rightsquigarrow \Gamma.t : \Gamma.T\\ \Gamma.S \vdash \left((\Gamma.t) \triangleleft \chi^{\text{id}}_{T}\right) \bullet r_{\Gamma.t} \equiv \chi^{\text{comp}}_{t,1_T} \bullet (\Gamma.r_t) : (\Gamma.t)(\Gamma.1_T) \rightsquigarrow \Gamma.t : \Gamma.T\end{array}}$$

$$\dfrac{\Gamma\ \text{ctx} \qquad \Gamma \vdash Q, R, S, T\ \text{type} \qquad \Gamma \mid Q \vdash r : R \qquad \Gamma \mid R \vdash s : S \qquad \Gamma \mid S \vdash t : T}{\Gamma.Q \vdash \left((\Gamma.r) \triangleleft \chi^{\text{comp}}_{s,t}\right) \bullet \chi^{\text{comp}}_{r,(st)} \bullet (\Gamma.\alpha_{r,s,t}) \equiv \alpha_{\Gamma.r,\Gamma.s,\Gamma.t} \bullet \left(\chi^{\text{comp}}_{r,s} \triangleright (\Gamma.t)\right) \bullet \chi^{\text{comp}}_{(rs),t} : (\Gamma.r)((\Gamma.s)(\Gamma.t)) \rightsquigarrow \Gamma.((rs)t) : \Gamma.T}$$

$$\dfrac{\Gamma\ \text{ctx} \qquad \Gamma \vdash R, S, T\ \text{type} \qquad \Gamma \mid R \vdash s, s' : S \qquad \Gamma \mid S \vdash t : T \qquad \Gamma \mid R \vdash \rho : s \rightsquigarrow s' : S}{\Gamma.R \vdash \chi^{\text{comp}}_{s,t} \bullet \Gamma.(\rho \triangleright t) \equiv (\Gamma.\rho \triangleright \Gamma.t) \bullet \chi^{\text{comp}}_{s',t} : (\Gamma.s)(\Gamma.t) \rightsquigarrow \Gamma.(s't) : \Gamma.T}$$

$$\dfrac{\Gamma\ \text{ctx} \qquad \Gamma \vdash R, S, T\ \text{type} \qquad \Gamma \mid R \vdash s : S \qquad \Gamma \mid S \vdash t, t' : T \qquad \Gamma \mid S \vdash \rho : t \rightsquigarrow t' : T}{\Gamma.R \vdash \chi^{\text{comp}}_{s,t} \bullet \Gamma.(s \triangleleft \rho) \equiv (\Gamma.s \triangleleft \Gamma.\rho) \bullet \chi^{\text{comp}}_{s,t'} : (\Gamma.s)(\Gamma.t) \rightsquigarrow \Gamma.(st') : \Gamma.T}$$

Fig. 3. Rules for comprehension

$$\frac{\Gamma, \Delta \ \mathsf{ctx} \quad \Delta \vdash s : \Gamma \quad \Gamma \vdash T \ \mathsf{type}}{\Delta \vdash T[s] \ \mathsf{type}} \ \textsf{sub-ty}$$

$$\frac{\Gamma, \Delta \ \mathsf{ctx} \quad \Delta \vdash s : \Gamma \quad \Gamma \vdash S, T \ \mathsf{type} \quad \Gamma \mid S \vdash t : T}{\Delta \mid S[s] \vdash t[s] : T[s]} \ \textsf{sub-tm}$$

$$\frac{\Gamma, \Delta \ \mathsf{ctx} \quad \Delta \vdash s : \Gamma \quad \Gamma \vdash S, T \ \mathsf{type} \quad \Gamma \mid S \vdash t, t' : T \quad \Gamma \mid S \vdash \rho : t \rightsquigarrow t' : T}{\Delta \mid S[s] \vdash \rho[s] : t[s] \rightsquigarrow t'[s] : T[s]} \ \textsf{sub-red}$$

$$\frac{\Gamma, \Delta \ \mathsf{ctx} \quad \Delta \vdash s : \Gamma \quad \Gamma \vdash T \ \mathsf{type}}{\Delta \mid T[s] \vdash \mathsf{Sub}_{\mathsf{I}}(s) : 1_T[s] \mathrel{\tilde{\rightsquigarrow}} 1_{T[s]} : T[s]} \ \textsf{sub-pres-id}$$

$$\frac{\Gamma, \Delta \ \mathsf{ctx} \quad \Delta \vdash s : \Gamma \quad \Gamma \vdash R, S, T \ \mathsf{type} \quad \Gamma \mid R \vdash r : S \quad \Gamma \mid S \vdash t : T}{\Delta \mid R[s] \vdash \mathsf{Sub}_{\mathsf{C}}(r, t, s) : r[s]t[s] \mathrel{\tilde{\rightsquigarrow}} (rt)[s] : T[s]} \ \textsf{sub-pres-comp}$$

$$\frac{\Gamma, \Delta \ \mathsf{ctx} \quad \Delta \vdash s : \Gamma \quad \Gamma \vdash S, T \ \mathsf{type} \quad \Gamma \mid S \vdash t : T}{\Delta \mid S[s] \vdash 1_t[s] \equiv 1_{t[s]} : t[s] \rightsquigarrow t[s] : T[s]} \ \textsf{sub-red-pres-id}$$

$$\frac{\Gamma, \Delta \ \mathsf{ctx} \quad \Delta \vdash s : \Gamma \quad \Gamma \vdash S, T \ \mathsf{type} \quad \Gamma \mid S \vdash t, t', t'' : T \quad \Gamma \mid S \vdash \rho : t \rightsquigarrow t' : T \quad \Gamma \mid S \vdash \rho' : t' \rightsquigarrow t'' : T}{\Delta \mid S[s] \vdash (\rho \bullet \rho')[s] \equiv \rho[s] \bullet \rho'[s] : t[s] \rightsquigarrow t''[s] : T[s]} \ \textsf{sub-red-pres-comp}$$

$$\frac{\Gamma, \Delta \ \mathsf{ctx} \quad \Delta \vdash s : \Gamma \quad \Gamma \vdash S, T \ \mathsf{type} \quad \Gamma \mid S \vdash t : T}{\Delta \mid S[s] \vdash (\mathsf{Sub}_{\mathsf{I}}(s) \rhd t[s]) \bullet \ell_{t[s]} \equiv \mathsf{Sub}_{\mathsf{C}}(1_S, t, s) \bullet \ell_t[s] : 1_S[s]t[s] \rightsquigarrow t[s] : T[s]} \ \textsf{sub-pres-lunitor}$$

$$\frac{\Gamma, \Delta \ \mathsf{ctx} \quad \Delta \vdash s : \Gamma \quad \Gamma \vdash S, T \ \mathsf{type} \quad \Gamma \mid S \vdash t : T}{\Delta \mid S[s] \vdash (t[s] \lhd \mathsf{Sub}_{\mathsf{I}}(s)) \bullet r_{t[s]} \equiv \mathsf{Sub}_{\mathsf{C}}(t, 1_T, s) \bullet r_t[s] : t[s]1_T[s] \rightsquigarrow t[s] : T[s]} \ \textsf{sub-pres-runitor}$$

$$\frac{\Gamma, \Delta \ \mathsf{ctx} \quad \Delta \vdash s : \Gamma \quad \Gamma \vdash Q, R, S, T \ \mathsf{type} \quad \Gamma \mid Q \vdash q : R \quad \Gamma \mid R \vdash r : S \quad \Gamma \mid S \vdash t : T}{\begin{array}{c}\Delta \mid Q[s] \vdash (q[s] \lhd \mathsf{Sub}_{\mathsf{C}}(r, t, s)) \bullet \mathsf{Sub}_{\mathsf{C}}(q, rt, s) \bullet \alpha_{q,r,t}[s] \equiv \alpha_{q[s],r[s],t[s]} \bullet (\mathsf{Sub}_{\mathsf{C}}(q, r, s) \rhd t[s]) \bullet \mathsf{Sub}_{\mathsf{C}}(qr, t, s) \\ : q[s](r[s]t[s]) \rightsquigarrow ((qr)t)[s] : T[s]\end{array}} \ \textsf{sub-pres-assoc}$$

$$\frac{\Gamma, \Delta \ \mathsf{ctx} \quad \Delta \vdash s : \Gamma \quad \Gamma \vdash R, S, T \ \mathsf{type} \quad \Gamma \mid R \vdash r : S \quad \Gamma \mid S \vdash t, t' : T \quad \Gamma \mid S \vdash \rho : t \rightsquigarrow t' : T}{\Delta \mid R[s] \vdash \mathsf{Sub}_{\mathsf{C}}(r, t, s) \bullet (r \lhd \rho)[s] \equiv (r[s] \lhd \rho[s]) \bullet \mathsf{Sub}_{\mathsf{C}}(r, t', s) : r[s]t[s] \rightsquigarrow rt'[s] : T[s]} \ \textsf{sub-pres-lwhisker}$$

$$\frac{\Gamma, \Delta \ \mathsf{ctx} \quad \Delta \vdash s : \Gamma \quad \Gamma \vdash R, S, T \ \mathsf{type} \quad \Gamma \mid R \vdash r, r' : S \quad \Gamma \mid S \vdash t : T \quad \Gamma \mid R \vdash \rho : r \rightsquigarrow r' : S}{\Delta \mid R[s] \vdash \mathsf{Sub}_{\mathsf{C}}(r, t, s) \bullet (\rho \rhd t)[s] \equiv (\rho[s] \rhd t[s]) \bullet \mathsf{Sub}_{\mathsf{C}}(r', t, s) : r[s]t[s] \rightsquigarrow r't[s] : T[s]} \ \textsf{sub-pres-rwhisker}$$

Fig. 4. Rules for global substitution

$$\frac{\Gamma\ \mathsf{ctx} \qquad \Gamma \vdash T\ \mathsf{type}}{\Gamma \mid T[1_\Gamma] \mathrel{\tilde{\vdash}} \mathsf{sub}_{\mathsf{id}} : T}\ \textsf{sub-id}$$

$$\frac{\Gamma\ \mathsf{ctx} \qquad \Gamma \vdash S, T\ \mathsf{type} \qquad \Gamma \mid S \vdash t : T}{\Gamma \mid S \vdash \mathsf{STm}_{\mathsf{I}}(t) : \mathsf{sub}_{\mathsf{id}}^{-1}\ t[1_\Gamma]\ \mathsf{sub}_{\mathsf{id}} \mathrel{\tilde{\rightsquigarrow}} t : T}\ \textsf{tm-sub-id}$$

$$\frac{\Gamma\ \mathsf{ctx} \qquad \Gamma \vdash S, T\ \mathsf{type} \qquad \Gamma \mid S \vdash t, t' : T \qquad \Gamma \mid S \vdash \rho : t \rightsquigarrow t' : T}{\Gamma \mid S \vdash \mathsf{STm}_{\mathsf{I}}(t) \bullet \rho \equiv (\mathsf{sub}_{\mathsf{id}}^{-1} \lhd \rho[1_\Gamma] \rhd \mathsf{sub}_{\mathsf{id}}) \bullet \mathsf{STm}_{\mathsf{I}}(t') : \mathsf{sub}_{\mathsf{id}}^{-1}\ t[1_\Gamma]\ \mathsf{sub}_{\mathsf{id}} \rightsquigarrow t' : T}\ \textsf{sub-red-id}$$

$$\frac{\Gamma\ \mathsf{ctx} \qquad \Gamma \vdash T\ \mathsf{type}}{\Gamma \mid T \vdash (\mathsf{sub}_{\mathsf{id}}^{-1} \lhd \mathsf{Sub}_{\mathsf{I}}(1_\Gamma) \rhd \mathsf{sub}_{\mathsf{id}}) \bullet (r_{\mathsf{sub}_{\mathsf{id}}^{-1}} \rhd \mathsf{sub}_{\mathsf{id}}) \bullet \mathsf{sub}_{\mathsf{id}}{}^{\rho} \equiv \mathsf{STm}_{\mathsf{I}}(1_T) : \mathsf{sub}_{\mathsf{id}}^{-1}\ 1_T[1_\Gamma]\ \mathsf{sub}_{\mathsf{id}} \rightsquigarrow 1_T : T}\ \textsf{sub-id-pres-id}$$

$$\frac{\Gamma\ \mathsf{ctx} \qquad \Gamma \vdash S, R, T\ \mathsf{type} \qquad \Gamma \mid S \vdash t : T \qquad \Gamma \mid T \vdash r : R}{\Gamma \mid S \vdash (\mathsf{sub}_{\mathsf{id}}^{-1} \lhd \mathsf{Sub}_{\mathsf{C}}(t, r, 1_\Gamma) \rhd \mathsf{sub}_{\mathsf{id}}) \bullet \mathsf{STm}_{\mathsf{I}}(tr) \equiv}\ \textsf{sub-id-pres-comp}$$

$$\alpha \bullet \left(\mathsf{sub}_{\mathsf{id}}^{-1}\ t[1_\Gamma] \lhd \left(\ell_{r[1_\Gamma]}^{-1} \bullet \left((\mathsf{sub}_{\mathsf{id}}{}^{\ell})^{-1} \rhd r[1_\Gamma]\right)\right) \rhd \mathsf{sub}_{\mathsf{id}}\right) \bullet \alpha \bullet \left(\mathsf{sub}_{\mathsf{id}}^{-1}\ t[1_\Gamma]\mathsf{sub}_{\mathsf{id}} \lhd \mathsf{STm}_{\mathsf{I}}(r)\right) \bullet (\mathsf{STm}_{\mathsf{I}}(t) \rhd r)$$

$$: \mathsf{sub}_{\mathsf{id}}^{-1}\ (t[1_\Gamma]\ r[1_\Gamma])\ \mathsf{sub}_{\mathsf{id}} \rightsquigarrow tr : R$$

Fig. 5. Rules for global substitution (preservation of identity)

$$\frac{\Gamma, \Delta, \mathrm{E}\ \mathsf{ctx} \qquad \mathrm{E} \vdash s : \Delta \qquad \Delta \vdash s' : \Gamma \qquad \Gamma \vdash T\ \mathsf{type}}{\mathrm{E} \mid T[s'][s] \mathrel{\tilde{\vdash}} \mathsf{sub}_{\mathsf{comp}}(s, s') : T[ss']}\ \textsf{sub-comp}$$

$$\frac{\Gamma, \Delta, \mathrm{E}\ \mathsf{ctx} \qquad \mathrm{E} \vdash s : \Delta \qquad \Delta \vdash s' : \Gamma \qquad \Gamma \vdash S, T\ \mathsf{type} \qquad \Gamma \mid S \vdash t : T}{\mathrm{E} \mid S[ss'] \vdash \mathsf{STm}_{\mathsf{C}}(t, s, s') : \mathsf{sub}_{\mathsf{comp}}(s, s')^{-1}\ t[s'][s]\ \mathsf{sub}_{\mathsf{comp}}(s, s') \mathrel{\tilde{\rightsquigarrow}} t[ss'] : T[ss']}\ \textsf{tm-sub-comp}$$

$$\frac{\Gamma, \Delta, \mathrm{E}\ \mathsf{ctx} \qquad \mathrm{E} \vdash s : \Delta \qquad \Delta \vdash s' : \Gamma \qquad \Gamma \vdash S, T\ \mathsf{type} \qquad \Gamma \mid S \vdash t, t' : T \qquad \Gamma \mid S \vdash \rho : t \rightsquigarrow t' : T}{\mathrm{E} \mid S[ss'] \vdash \mathsf{STm}_{\mathsf{C}}(t, s, s') \bullet \rho[ss'] \equiv (\mathsf{sub}_{\mathsf{comp}}^{-1}(s, s') \lhd \rho[s'][s] \rhd \mathsf{sub}_{\mathsf{comp}}(s, s')) \bullet \mathsf{STm}_{\mathsf{C}}(t', s, s')}\ \textsf{sub-red-comp}$$

$$: \mathsf{sub}_{\mathsf{comp}}(s, s')^{-1}\ t[s'][s]\ \mathsf{sub}_{\mathsf{comp}}(s, s') \rightsquigarrow t'[ss'] : T[ss']$$

$$\frac{\Gamma, \Delta, \mathrm{E}\ \mathsf{ctx} \qquad \mathrm{E} \vdash s : \Delta \qquad \Delta \vdash s' : \Gamma \qquad \Gamma \vdash T\ \mathsf{type}}{\mathrm{E} \mid T[ss'] \vdash \mathsf{sub}_{\mathsf{comp}}(s, s')^{-1} \lhd ((\mathsf{Sub}_{\mathsf{I}}(s')[s] \bullet \mathsf{Sub}_{\mathsf{I}}(s)) \rhd \mathsf{sub}_{\mathsf{comp}}(s, s') \bullet \ell) \bullet \mathsf{sub}_{\mathsf{comp}}(s, s')^{\rho} \equiv \mathsf{STm}_{\mathsf{C}}(1_T, s, s') \bullet \mathsf{Sub}_{\mathsf{I}}(ss')}\ \textsf{sub-comp-pres-id}$$

$$: \mathsf{sub}_{\mathsf{comp}}(s, s')^{-1}\ 1_T[s'][s]\ \mathsf{sub}_{\mathsf{comp}}(s, s') \rightsquigarrow 1_{T[ss']} : T[ss']$$

$$\frac{\Gamma, \Delta, \mathrm{E}\ \mathsf{ctx} \qquad \mathrm{E} \vdash s : \Delta \qquad \Delta \vdash s' : \Gamma \qquad \Gamma \vdash S, T, R\ \mathsf{type} \qquad \Gamma \mid S \vdash t : T \qquad \Gamma \mid T \vdash r : R}{\mathrm{E} \mid S[ss'] \vdash \left(\mathsf{sub}_{\mathsf{comp}}^{-1}(s, s') \lhd (\mathsf{Sub}_{\mathsf{C}}(t[s'], r[s'], s) \bullet \mathsf{Sub}_{\mathsf{C}}(t, r, s')[s]) \rhd \mathsf{sub}_{\mathsf{comp}}(s, s')\right) \bullet \mathsf{STm}_{\mathsf{C}}(tr, s, s') \equiv}\ \textsf{sub-comp-pres-comp}$$

$$\alpha \bullet \left((\mathsf{sub}_{\mathsf{comp}}^{-1}(s, s') t[s'][s]) \lhd \left(\ell_{r[s'][s]}^{-1} \bullet (\mathsf{sub}_{\mathsf{comp}}(s, s')^{\ell})^{-1} \rhd r[s'][s]\right) \rhd \mathsf{sub}_{\mathsf{comp}}(s, s')\right) \bullet \alpha \bullet$$

$$\left(\mathsf{STm}_{\mathsf{C}}(t, s, s') \rhd \left(\mathsf{sub}_{\mathsf{comp}}^{-1}(s, s') r[s'][s] \mathsf{sub}_{\mathsf{comp}}^{-1}(s, s')\right)\right) \bullet \left(t[ss'] \lhd \mathsf{STm}_{\mathsf{C}}(r, s', s)\right) \bullet \mathsf{Sub}_{\mathsf{C}}(t, r, ss')$$

$$: \mathsf{sub}_{\mathsf{comp}}^{-1}(s, s')\ (t[s'][s]\ r[s'][s])\ \mathsf{sub}_{\mathsf{comp}}(s, s') \rightsquigarrow tr[ss'] : R[ss']$$

Fig. 6. Rules for global substitution (preservation of composition)

$$\frac{\Gamma, \Delta \text{ ctx} \quad \Delta \vdash s, s' : \Gamma \quad \Delta \vdash \rho : s \rightsquigarrow s' : \Gamma \quad \Gamma \vdash T \text{ type}}{\Delta \mid T[s] \vdash \text{map } T\ \rho : T[s']}\ \text{map}$$

$$\frac{\Gamma, \Delta \text{ ctx} \quad \Delta \vdash s, s' : \Gamma \quad \Delta \vdash \rho : s \rightsquigarrow s' : \Gamma \quad \Gamma \vdash S, T \text{ type} \quad \Gamma \mid S \vdash t : T}{\Delta \mid S[s] \vdash \text{map } t\ \rho : t[s](\text{map } T\ \rho) \tilde{\rightsquigarrow} (\text{map } S\ \rho)t[s'] : T[s']}\ \text{rew-tm}$$

$$\frac{\Gamma, \Delta \text{ ctx} \quad \Delta \vdash s, s' : \Gamma \quad \Delta \vdash \rho : s \rightsquigarrow s' : \Gamma \quad \Gamma \vdash S, T \text{ type} \quad \Gamma \mid S \vdash t, t' : T \quad \Gamma \mid S \vdash \tau : t \rightsquigarrow t' : T}{\Delta \mid S[s] \vdash \text{map } t\ \rho \bullet (\text{map } S\ \rho \lhd \tau[s']) \equiv (\tau[s] \rhd \text{map } T\ \rho) \bullet \text{map } t'\ \rho : t[s](\text{map } T\ \rho) \rightsquigarrow (\text{map } S\ \rho)t'[s'] : T[s']}\ \text{rew-red}$$

$$\frac{\Gamma, \Delta \text{ ctx} \quad \Delta \vdash s, s' : \Gamma \quad \Delta \vdash \rho : s \rightsquigarrow s' : \Gamma \quad \Gamma \vdash T \text{ type}}{\Delta \mid T[s] \vdash \text{map } 1_T\ \rho \bullet (\text{map } T\ \rho \lhd \text{Sub}_{\text{l}}(s')) \bullet r_{\text{map } T\ \rho} \equiv \text{Sub}_{\text{l}}(s) \rhd \text{map } T\ \rho \bullet \ell : 1_T[s](\text{map } T\ \rho) \rightsquigarrow \text{map } T\ \rho : T[s']}\ \text{rew-pres-id}$$

$$\frac{\Gamma, \Delta \text{ ctx} \quad \Delta \vdash s, s' : \Gamma \quad \Delta \vdash \rho : s \rightsquigarrow s' : \Gamma \quad \Gamma \vdash S, T, R \text{ type} \quad \Gamma \mid S \vdash t : T \quad \Gamma \mid T \vdash r : R}{\begin{gathered}\Delta \mid S[s] \vdash \text{map } (tr)\ \rho \bullet (\text{map } S\ \rho \lhd \text{Sub}_{\text{C}}(t, r, s')^{-1}) \bullet \alpha \equiv \\ (\text{Sub}_{\text{C}}(t, r, s)^{-1} \rhd \text{map } T\ \rho) \bullet \alpha \bullet (t[s] \lhd \text{map } r\ \rho) \bullet \alpha \bullet (\text{map } t\ \rho \rhd r[s']) \\ : (tr)[s](\text{map } R\ \rho) \rightsquigarrow (\text{map } S\ \rho)t[s']r[s'] : R[s']\end{gathered}}\ \text{rew-pres-comp}$$

Fig. 7. Rules for local substitution

$$\frac{\Gamma, \Delta \text{ ctx} \quad \Delta \vdash s : \Gamma \quad \Gamma \vdash T \text{ type}}{\Delta \mid T[s] \vdash \text{map}^T_{\text{id}}\ s : \text{map } T\ 1_s \tilde{\rightsquigarrow} 1_{T[s]} : T[s]}\ \text{map-id}$$

$$\frac{\Gamma, \Delta \text{ ctx} \quad \Delta \vdash s, s', s'' : \Gamma \quad \Delta \vdash \rho : s \rightsquigarrow s' : \Gamma \quad \Delta \vdash \tau : s' \rightsquigarrow s'' : \Gamma \quad \Gamma \vdash T \text{ type}}{\Delta \mid T[s] \vdash \text{map}^T_{\text{comp}}(\rho, \tau) : \text{map } T\ (\rho \bullet \tau) \tilde{\rightsquigarrow} \text{map } T\ \rho \cdot \text{map } T\ \tau : T[s'']}\ \text{map-comp}$$

$$\frac{\Gamma, \Delta, \mathrm{E} \text{ ctx} \quad \mathrm{E} \vdash s : \Delta \quad \Delta \vdash s', s'' : \Gamma \quad \Delta \vdash \rho : s' \rightsquigarrow s'' : \Gamma \quad \Gamma \vdash T \text{ type}}{\mathrm{E} \mid T[ss'] \vdash \text{map}^T_{\text{l}}(s, \rho) : \text{map } T\ (s \lhd \rho) \tilde{\rightsquigarrow} \text{sub}^{-1}_{\text{comp}}(s, s') \cdot (\text{map } T\ \rho)[s] \cdot \text{sub}_{\text{comp}}(s, s'') : T[ss'']}\ \text{map-lwhisker}$$

$$\frac{\Gamma, \Delta, \mathrm{E} \text{ ctx} \quad \mathrm{E} \vdash s, s' : \Delta \quad \Delta \vdash s'' : \Gamma \quad \mathrm{E} \vdash \rho : s \rightsquigarrow s' : \Delta \quad \Gamma \vdash T \text{ type}}{\mathrm{E} \mid T[ss''] \vdash \text{map}^T_{\text{r}}(\rho, s'') : \text{map } T\ (\rho \rhd s'') \tilde{\rightsquigarrow} \text{sub}^{-1}_{\text{comp}}(s, s'') \cdot \text{map } (T[s''])\ \rho \cdot \text{sub}_{\text{comp}}(s', s'') : T[s's'']}\ \text{map-rwhisker}$$

Fig. 8. Rules for local substitution (preservation)

$$\frac{\Gamma, \Delta \text{ ctx} \quad \Delta \vdash s : \Gamma \quad \Gamma \vdash T \text{ type}}{\Delta \mid T[s][1_\Delta] \vdash \text{sub}_{\text{compid}}(T, s) : \text{sub}_{\text{comp}}(s, 1_\Delta)(\text{map } T\ r_s) \tilde{\rightsquigarrow} \text{sub}_{\text{id}}\ : T[s]}$$

$$\frac{\Gamma, \Delta \text{ ctx} \quad \Delta \vdash s : \Gamma \quad \Gamma \vdash T \text{ type}}{\Delta \mid T[1_\Gamma][s] \vdash \text{sub}_{\text{compid}}(T, s) : \text{sub}_{\text{comp}}(1_\Gamma, s)(\text{map } T\ \ell_s) \tilde{\rightsquigarrow} \text{sub}_{\text{id}}\ [s] : T[s]}$$

$$\frac{\Gamma, \Delta, \mathrm{E}, \mathrm{Z} \text{ ctx} \quad \mathrm{Z} \vdash s : \mathrm{E} \quad \mathrm{E} \vdash t : \Delta \quad \Delta \vdash u : \Gamma \quad \Gamma \vdash T \text{ type}}{\mathrm{Z} \mid T[u][t][s] \vdash \text{sub}_{\text{compcomp}}(T, u, t, s) : \text{sub}_{\text{comp}}(t, u)[s]\ \text{sub}_{\text{comp}}(s, tu)\ \text{map } T\ \alpha \tilde{\rightsquigarrow} \text{sub}_{\text{comp}}(s, t)\ \text{sub}_{\text{comp}}(st, u) : T[stu]}$$

Fig. 9. Rules for coherence of substitution