Syntax and Models of Cartesian Cubical Type Theory

Carlo Angiuli, Guillaume Brunerie, Thierry Coquand,
Kuen-Bang Hou (Favonia), Robert Harper, Daniel R. Licata

Abstract

We present a cubical type theory based on the Cartesian cube category (faces, degeneracies, symmetries, diagonals, but no connections or reversal) with univalent universes, each containing Π, Σ, path, identity, natural number, boolean, pushout, and glue (equivalence extension) types. The type theory includes a syntactic description of a uniform Kan operation, along with judgemental equality rules defining the Kan operation on each type. The Kan operation uses both a different set of generating trivial cofibrations and a different set of generating cofibrations than the Cohen, Coquand, Huber, and Mörtberg (CCHM) model.

Next, we describe a constructive model of this type theory in Cartesian cubical sets. We give a mechanized proof, using the internal language of cubical sets in the style introduced by Orton and Pitts, that glue, Π, Σ, path, identity, boolean, natural number, pushout types and the universe itself are Kan in this model, and that the universe is univalent. An advantage of this formal approach is that our construction can also be interpreted in cubical sets on the connections cube category, and on the de Morgan cube category used in the CCHM model. As a first step towards comparing these approaches, we show that the two Kan operations are interderivable in a setting where both exist (presheaves on the de Morgan cube category, with the additional cofibration required by our construction).

1 Introduction

Cubical type theory is a formal system for Univalent Foundations/homotopy type theory [Voevodsky, 2006; The Univalent Foundations Program, Institute for Advanced Study, 2013]. The goals of cubical type theory are twofold: we would like a computational homotopy type theory, where terms can be interpreted as programs, and we would like a better syntax for synthetic homotopy theory.

1.1 Constructive cubical models and type theories

Bezem, Coquand, and Huber [2014] gave a constructive model of type theory in cubical sets. In this model, a syntactic type is interpreted as a cubical set equipped with a *Kan operation*, which generalizes the elimination rule for the identity type in Martin-Löf type theory. The standard Kan filling condition (given all but one face of a cube, there exists a missing face and the inside) is made algebraic (there is a chosen operation producing the missing face and inside), generalized to permit additional dimensions not involved in the filling problem, and made *uniform*, a naturality condition relating fillings along the maps in the cube category. The stability under reindexing expressed in this *uniform Kan operation* allows a constructive definition of the Kan operation for Π types, and is a generally better fit for type theory, where we expect strict stability under substitutions. The model includes path types defined using the interval object in cubical sets, which validate different judgemental equalities than the inductive identity types of Martin-Löf type theory. For example the identity type's computation rule (J on `refl`) is weakened to a path/propositional/typal equality, but many new equations hold exactly for the path type. An operational semantics based on this model was implemented in the `cubical` proof assistant prototype.[1]

Following the development of this model, efforts began to present cubical ideas in syntactic type theories, which has several benefits. For example, syntax for the intermediate states of the definitions of the Kan operations is required if one wants to apply judgemental equalities to reduce a complex (possibly open) expression, and then manually reason about the reduct—a very common step in type theoretic developments. Moreover, the path types and equalities in the cubical model provide a different language for synthetic homotopy theory, which is more convenient for at least some proofs than Martin-Löf type theory with univalence and higher inductive types as axioms. However, a central goal of synthetic homotopy theory is to do proofs in a model-independent way, which motivates investigating syntactic presentations of cubical ideas that could potentially be given more than one model. One approach to cubical type theory, explored by Altenkirch and Kaposi [2014]; Polonsky [2014], is to do an "internal inductive step" following Altenkirch et al. [2007]: define the identity type so that it computes to another type, so that higher-dimensional operations can be derived from lower-dimensional ones. This involves the fewest additions to type theory, but it is currently unknown whether it can be made to work for univalent universes. Another approach is to extend type theory with syntactic interval variables, which are interpreted using the interval object in cubical sets.

One degree of freedom in the design of constructive cubical models is the choice of cube category on which to take presheaves. Bezem, Coquand, and Huber [2014] use a cube category with faces, degeneracies, and symmetries. This cube category C is the free monoidal category $(C, \otimes, \cdot)$ generated by an object $\mathbb{I}$, with morphisms generated by face maps $0, 1 : \cdot \to \mathbb{I}$, degeneracy $\mathbb{I} \to \cdot$, and a symmetry involution $\mathbb{I} \otimes \mathbb{I} \to \mathbb{I} \otimes \mathbb{I}$ (which makes the category symmetric monoidal). Syntactically, this corresponds to substructural interval variables without contraction, as in nominal logic [Pitts, 2015]. In a syntactic type theory, it would be preferable to also have contraction, so that syntactic interval variables behave like ordinary variables, according to the standard structural rules. Semantically, contraction corresponds to diagonal maps in the cube category. In addition to simplifying the type theory, another motivation for diagonals, which was discovered during the implementation of the Bezem, Coquand, and Huber [2014] model, is that without them it is unclear whether one can give eliminators for higher inductive types that obey exact computation laws on path constructors. Adding diagonals to the cube category used in Bezem, Coquand, and Huber [2014] gives the *Cartesian cube category*, the free finite product category on an interval object (in the notation above, the $\otimes$ becomes a Cartesian product $\times$). Since 2013, Awodey has encouraged the investigation of the Cartesian cube category, and stressed its good mathematical properties, such as the fact that it is the classifying topos of bipointed sets, that (unlike the Bezem, Coquand, and Huber [2014] cube category) geometric realization to topological spaces preserves products, and that the interval is atomic (exponentiation by the interval has a right adjoint)—see Awodey [2016]. The Cartesian cube category is a strict test category in the sense of Grothendieck, which means roughly that it has the same homotopy theory as simplicial sets/topological spaces; Buchholtz and Morehouse [2017] develop a thorough investigation of which cube categories are (strict) test categories.

Coquand developed a uniform Kan operation [Coquand, 2014b] for Cartesian cubical sets, and a uniform Kan operation [Coquand, 2014a] for cubical sets with diagonals plus additional degen-

[1] `https://github.com/simhu/cubical`

eracies called *connections*, which can be thought of as meets and joins on the interval.[2] These two operations began to explore another degree of freedom in the design of constructive cubical models, which is what class of filling problems are allowed, and which composites of filling problems are taken as primitive. For example, the Kan operation designed for the Cartesian cube category allows more general trivial cofibrations, which are not necessary for the connections cube category because they can be encoded. Though there is certainly not a 1-1 correspondence between cube categories and Kan operations (for example, Awodey [2018b] considers the Cartesian cube category with a standard notion of trivial cofibration), for the remainder of this introduction, we will use "the Cartesian model" as shorthand for the Cartesian cube category with the [Coquand, 2014b] Kan operation, and "the connections model" as shorthand for the connections/de Morgan cube category with the Coquand [2014a] Kan operation.

Several syntactic type theories with interval variables were developed in mid-2014 in parallel independent work. Bernardy et al. [2015] developed a presheaf type theory for polymorphism. Coquand [2014a] developed a type theoretic presentation of the connections model. Brunerie and Licata [2014] developed a type theoretic presentation of the Cartesian model. Isaev [2014] developed an interval-based type theory with diagonals and (one) connection, and implemented a prototype proof assistant with path and higher inductive types using the interval; in this work, the interval itself is a "fibrant" type, and to our knowledge a fibrant interval has not yet been investigated from a semantic point of view.

Coquand [2014a] additionally introduced the "glue type" for extending a type by equivalences, a special case of which is univalence, and proposed a definition of a Kan composition structure on the universe. This connections model adopted a *regularity* condition, which is a generalization of the "`transport` on reflexivity is the identity" definitional equality in Martin-Löf type theory. With regularity, the path types of cubical type theory provide both the judgemental equalities of the MLTT identity type, and the useful new judgemental equalities that exponentiation by an interval provides. However, our attempts in 2014–2015 to adapt the glue types and definition of the universe from the connections model to the Cartesian model revealed that the proposed definition of the universe did not actually satisfy regularity.[3] Cohen, Coquand, Huber, and Mörtberg [2018] adapt the connections model to a non-regular setting, and show that the full univalence axiom follows from glue types. However, a solution for the Cartesian model with the more permissive Coquand [2014b] Kan operation remained elusive. Awodey [2018b] constructed a model of identity types in the Cartesian cube category using the standard notion of trivial cofibration (endpoint inclusions), where the identity types are interpreted as path types (exponentiation by the interval), and also satisfy regularity/normality, and therefore the full J on `refl` exact equality—but this has not yet extended to univalent universes.

In 2015 and 2016, both the connections model and the partial Cartesian model were studied from an operational point of view. Angiuli et al. [2016]; Angiuli and Harper [2017]; Angiuli et al. [2017a] developed a computational higher type theory based on Cartesian cubes. (Computational type theory differs from the formal type theory considered here, in that an operational semantics is given first, and a logical relation on programs is taken as the definition of the type system.) Because the computational model validates the rules of the formal type theory, that work implies

[2]More precisely, the *de Morgan* cube category used in Coquand [2014a] also includes a reversal involution $1- : \mathbb{I} \to \mathbb{I}$, giving strict path reversal. We will gloss over the distinction between the connections cube category and the de Morgan one, as the differences between the connections and de Morgan Kan operations are less important to our story than the differences between the Cartesian and connections ones.

[3]See the post `https://goo.gl/btFxZ4` from 5/31/2015 on the Homotopy Type Theory mailing list.

a canonicity result for Cartesian cubical type theory with Π, Σ, path types, booleans, the circle, and "isovalence" (gluing with a strict isomorphism, i.e., two functions that compose to the identity exactly, rather than up to paths). In parallel work, Huber [2018] proved canonicity for de Morgan cubical type theory (with connections and reversal), including Π, Σ, path, gluing (and therefore univalence), the universe, natural numbers, the circle, and propositional truncation.

In Spring 2017, Angiuli, Favonia, and Harper discovered how to construct univalent Kan universes for Cartesian cubes by extending the [Coquand, 2014b] Kan operation. The key idea is to allow the diagonal inclusion $\mathbb{I} \to \mathbb{I} \times \mathbb{I}$ in the cube category as a cofibration, which geometrically corresponds to attaching faces to the diagonals of open boxes. Angiuli et al. [2017b, 2018] use these diagonal cofibrations to define a computational Cartesian cubical type theory with an infinite hierarchy of univalent universes as well as an extensional equality judgment internalized as an equality pre-type.[4] In this paper, we consider diagonal cofibrations from a proof-theoretic and model-theoretic point of view, completing the Cartesian cubical type theory begun by Brunerie and Licata [2014], and the constructive model of univalence in Cartesian cubical sets begun by Coquand [2014b].

Though these two investigations use the same key idea of diagonal cofibrations to construct the universe, Angiuli et al. [2017b]'s computational type theory (AFH) is not a model of the formal type theory that we consider here. The two theories differ on several design choices: Here, Kan composition is described as a single operation, while in AFH it is decomposed into two simpler operations (coercion and homogeneous composition) in all types. Here, all definitions are open-ended with respect to adding new dimension terms (e.g. connections, reversals), while in AFH the definitions of the Kan operation depend on knowing all possible dimension terms (e.g. case-analyzing a term as either 0, 1, or a variable). Here, the definition of the Kan operation for the universe is broken into a series of lemmas, while in AFH some of these steps are "inlined" to expose some optimizations. Here, both univalence and the Kan operation for the universe are combined using the full glue types of Cohen, Coquand, Huber, and Mörtberg [2018], while AFH treats univalence and Kan composition in the universe separately: for univalence it suffices to consider only gluing with a single equivalence, and for composition one can avoid converting every path to an equivalence before composing. In AFH, the Kan operation disallows false cofibrations, thereby avoiding the "empty system compositions" that proliferate in formalizations. In AFH, universes of Kan types are inductively defined as having certain types as members; in the semantics in Section 3, universes are open-ended, in the sense that any type with a Kan operation has a code in the universe of Kan types. In general, AFH optimizes for the computational efficiency of the definitions, which is essential for using the computational content of proofs in practice—for example, we should be able to automatically reduce Brunerie's proof of $\pi_4(S^3)$ to determine that it is $\mathbb{Z}/2$, but this has not yet been possible in any implementation. Here, we optimize for mathematical simplicity, which was helpful for mechanizing the proofs.

Even fixing all of these above, there are still some choices about how to define composition for the universe intensionally (though being Kan is a homotopy proposition, so up to paths all of these choices are the same). For example, a previous draft of this article decomposed the Kan operation for the universe in the same way we presently treat higher inductive types. Here, we avoid this decomposition, and use an algorithm that is close to the one in [Cohen, Coquand, Huber, and Mörtberg, 2018]. At a recent Dagstuhl seminar, a team including some of the authors investigated an additional variant, in order to improve the efficiency of part of the algorithm.

[4] Some equality types are in fact Kan; see Angiuli et al. [2017b].

Since the original draft of this article in December 2017, Cavallo and Harper [2019] have defined a schema for higher inductive types in computational cartesian cubical type theory. Additionally, several experimental implementations of cartesian cubical type theory have been developed, including **RedPRL**,[5] `redtt`,[6] and `yacctt`.[7]

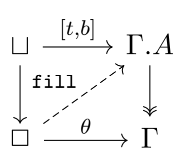
Our goal in this paper is to document the definition and semantics of the type theory, so we refer a reader who is interested in examples of using the theory to the following resources: Bentzen [2018] has developed a number of examples internally to the AFH theory. Some tutorials and courses include Harper and Angiuli [2018]; Harper [2018]. Finally, the above implementations contain many examples coded in the systems.

1.2 Kan operations

The main aspects of our approach are best understood after a bit of background on cubical type theory and the syntax and semantics of uniform Kan operations. Our understanding of the semantics was particularly influenced by Awodey [2018b]; Gambino and Sattler [2017]; Orton and Pitts [2016]; Sattler [2017].

The basic idea of cubical type theory is to consider judgements of the form $\Psi; \Gamma \vdash a : A$, where Ψ is a context of dimension variables $x : \mathbb{I}$. So $\cdot; \Gamma \vdash a : A$ is a point, $x : \mathbb{I}; \Gamma \vdash a : A$ is a line, $x : \mathbb{I}, y : \mathbb{I}; \Gamma \vdash a : A$ is a square, and so on. Semantically, Ψ is an object of a cube category $\mathbb{C}$, which here we take to be the Cartesian cube category, the free finite product category on an interval object $\mathbb{I}$ with maps $\cdot \vdash 0 : \mathbb{I}$ and $\cdot \vdash 1 : \mathbb{I}$. A (closed) type is interpreted as a presheaf in $\hat{\mathbb{C}} := \mathbf{Sets}^{\mathbb{C}^{op}}$, and in the simple case where a, but not A, mentions variables from Ψ (and the context Γ is empty), the judgement $\Psi \vdash a : A$ represents a natural transformation from the representable $\hom_{\mathbb{C}}(-, \Psi)$ to A, or, by Yoneda, an element of $A(\Psi)$. Substitution of special symbols 0 and 1 for interval variables, which we write as $a\langle 0/x \rangle$ and $a\langle 1/x \rangle$, correspond to reindexing using the action of the presheaf A on the maps in the cube category. The structural rules of weakening, exchange, and contraction for interval variables correspond to reindexing by degeneracy, symmetry, and diagonals, respectively. The basic type constructors of type theory (Π, Σ, $\mathbb{N}$, etc.) can be interpreted in any presheaf model in such a way that their rules apply uniformly at every dimension. In a cubical type theory, this justifies stating their rules for an arbitrary context Ψ—e.g. for functions, we have both λ and application in every context Ψ. This generalization gives many constructions on paths, such as `ap` and function extensionality, as special cases of the rules for the basic type constructors.

To make this cubical set theory into a cubical (homotopy) type theory, we want types to be fibrations with respect to the paths given by maps from the dimension variables/interval object $\mathbb{I}$. This is accomplished by adding a *Kan filling operation*, whose basic shape is



$$\frac{\sqcup; \Gamma \vdash [t, b] : A}{\square; \Gamma \vdash \mathtt{fill}_A(t, b) : A} \qquad \sqcup; \Gamma \vdash \mathtt{fill}_A(t, b) \equiv [t, b] : A$$

This says that given a fibration on the right, and a solvable filling problem on the left, a "whole" shape in Γ in the base, and a boundary in $\Gamma.A$ on the top, we obtain a whole shape in $\Gamma.A$. Commutativity of the outer square says that the boundary is in the same fiber in Γ as the whole shape

[5]http://www.redprl.org/

[6]https://github.com/RedPRL/redtt

[7]https://github.com/mortberg/yacctt

in the base. Commutativity of the bottom triangle says that the filler is as well, and commutativity of the top triangle says that the filler agrees with the provided boundary on the base. The standard solvable filling problems introduced by Kan are the inclusions of a "box missing a face" into the whole box, e.g. an endpoint into a line, or three sides into a square. Here $\square$ is an object of the cube category/representable functor on it, and $\sqcup$ is a subpresheaf of it.

In syntax, this corresponds to the rule on the right. The base map θ does not appear in the syntax, but is implicit in the substitution principle for the judgement. The equation says that the top triangle commutes. The fact that this rule applies to any type A imposes an obligation that all types in the type theory "are Kan" or "are fibrant". More concretely, when the filling problem is actually the "left-bottom-right faces included into the square" suggested by the notation, we might write

$$\dfrac{\begin{array}{c} x:\mathbb{I}, y:\mathbb{I}; x=0; \Gamma \vdash t_0 : A \qquad x:\mathbb{I}, y:\mathbb{I}; x=1; \Gamma \vdash t_1 : A \qquad x:\mathbb{I}, y:\mathbb{I}; y=0; \Gamma \vdash b : A \\ x:\mathbb{I}, y:\mathbb{I}; x=0, y=0; \Gamma \vdash t_0 \equiv b : A \qquad x:\mathbb{I}, y:\mathbb{I}; x=1, y=0; \Gamma \vdash t_1 \equiv b : A \end{array}}{\dfrac{x:\mathbb{I}, y:\mathbb{I}; (x=0 \vee x=1) \vee y=0; \Gamma \vdash [[t_0, t_1], b] : A}{\begin{array}{c} x:\mathbb{I}, y:\mathbb{I}; \Gamma \vdash \mathtt{fill}_A(t,b) : A \\ x:\mathbb{I}, y:\mathbb{I}; x=0; \Gamma \vdash \mathtt{fill}_A(t,b) \equiv t_0 : A \\ x:\mathbb{I}, y:\mathbb{I}; x=1; \Gamma \vdash \mathtt{fill}_A(t,b) \equiv t_1 : A \\ x:\mathbb{I}, y:\mathbb{I}; y=0; \Gamma \vdash \mathtt{fill}_A(t,b) \equiv b : A \end{array}}}$$

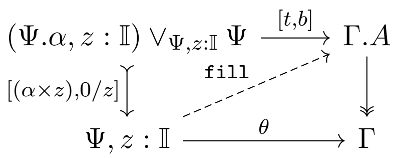
This says that we have a y-path t_0 at $x = 0$ (left), a y-path t_1 at $x = 1$ (right), and an x-path b at $y = 0$ (base), which agree on the corners, and we produce a whole square, which agrees with the provided sides on the boundary.

As this example illustrates, the boundaries of filling problems can be thought of as *formulas* or restrictions on the dimension context, an approach introduced by Cohen, Coquand, Huber, and Mörtberg [2018] and developed by Orton and Pitts [2016]; Birkedal et al. [2018]; Riehl and Shulman [2017]. The boundary formulas are often left-invertible, which means we could without loss of generality reduce the boundary to a sequence of terms. However, it is both convenient and more abstract to have syntax for boundaries. For example, we will define the filling operations in the type theory in such a way that they are *open-ended* with respect to extensions of the allowed filling problems, e.g. with the ones considered in Cohen, Coquand, Huber, and Mörtberg [2018]. Moreover, this approach is essential to presenting cubical models in the internal logic of a topos using techniques developed by Orton and Pitts [2016]; Birkedal et al. [2018].

A key part of the specification of a cubical type theory/model is what filling problems are allowed. For example, it is necessary to preclude non-contractible shapes, fillers of which would make the path type inconsistent (for example, a line filler both of whose endpoints are specified would give a path between `true` and `false`). The standard open boxes considered by Kan have all but one face, which can be represented by a formula with $(x = 0 \vee x = 1)$ for every dimension *except* one, and one face in the remaining ("filling") direction. We call the former the "tube" and the latter the "cap" or "base".

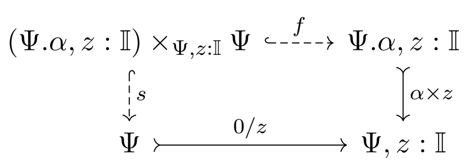
However, cubical type theories represent dimensions as variables, and we generally want admissible and *silent* weakening (i.e., terms need not change when considered in a larger context), which is incompatible with requiring that open boxes have faces for all dimensions but one. Weakening corresponds to the action of degeneracy maps on the cubical sets, so for weakening to be admissible, the uniform Kan filling operations [Bezem et al., 2014] allow filling problems where some, but not necessarily all, interval variables are assigned faces in the tube, and impose equations relating filling problems before and after degeneracy.

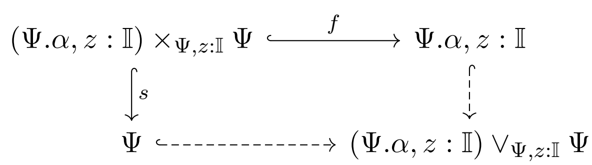
We focus on explaining the generalized shapes of filling problems here; see [Awodey, 2018b; Gambino and Sattler, 2017] for an analysis of the equational aspects of uniformity in semantic terms. Suppose we have a boundary formula α on a context Ψ, which identifies the faces of Ψ that are part of the filling problem. For the standard open boxes, α will select both the 0 and 1 faces of all variables in Ψ, but it may select fewer. Then we can fill a $\Psi, z : \mathbb{I}$ cube if we are given (1) the tube sides specified by α, which all may depend on the filling direction z, and (2) a cap at $z = 0$, such that (3) these are compatible on both α and $z = 0$ (we could have an analogous rule for $z = 1$).



$$\frac{\begin{array}{c}\Psi, z : \mathbb{I} \vdash A\ \mathtt{Type} \\ \Psi.\alpha, z : \mathbb{I}; \Gamma \vdash t : A \\ \Psi; \Gamma \vdash b : A\langle 0/z\rangle \\ \Psi; \alpha; \Gamma \vdash t\langle 0/z\rangle \equiv b : A\langle 0/z\rangle\end{array}}{\begin{array}{c}\Psi, z : \mathbb{I}; \Gamma \vdash \mathtt{fill}_A(t,b) : A \\ \Psi, z : \mathbb{I}; \alpha; \Gamma \vdash \mathtt{fill}_A(t,b) \equiv t : A \\ \Psi; \Gamma \vdash (\mathtt{fill}_A(t,b))\langle 0/z\rangle \equiv b : A\end{array}}$$

The object in the upper-left corner is the pushout of the pullback of $\alpha \times z$ and $0/z$, which captures a tube and cap that fit together as above. That is, we begin with the boundary formula α, which determines a restricted object $\Psi.\alpha$ that includes into Ψ, and then take the pullback on the left, and then the pushout on the right:



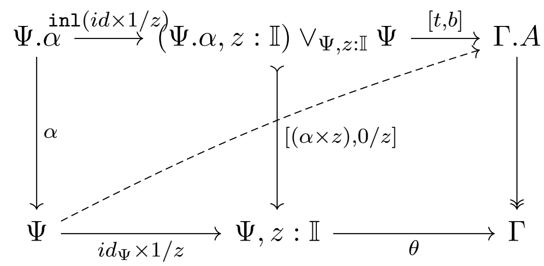


The pushout includes into $\Psi, z : \mathbb{I}$ by the universal property of the pushout applied to $\alpha \times z$ and $0/z$ (the pushout corner map).

The syntactic rule on the right expresses this pushout elimination principle: we need to give a tube t, which does depend on z but is restricted to α, along with a cap b, which is at $z = 0$, which agree on the pullback, i.e. on α and when $z = 0$. This form of the Kan operation is now closed under weakening/degenerating with additional interval variables, because we can inductively weaken all of A, α, t, b, and the boundary constraint, and then reapply the rule in the extended context. The fact that weakening "commutes" with the Kan operation is an aspect of *uniformity* [Bezem, Coquand, and Huber, 2014]; in full, uniformity says that the action of any cube map into Ψ commutes with the filling operation in the same way.

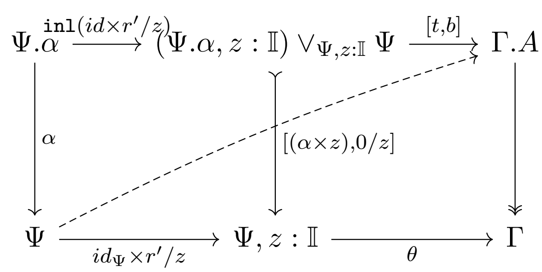
In model category terminology, the boundary formula α is a notion of *cofibration*. For Bezem, Coquand, and Huber [2014] filling problems, α is a disjunction of pairs $x = 0 \vee x = 1$ for variables x, capturing the fact that every dimension involved in the filling problem has both a 0 and 1 face, but some dimensions may not be involved. However, we could also take α to be disjunctions of $x = 0$ and $x = 1$ separately, in which case a dimension can be involved in the filling problem with only one specified tube side. Cohen, Coquand, Huber, and Mörtberg [2018] additionally allow conjunctions (pullbacks) of boundary formulas as cofibrations. The choice of cofibrations is quite flexible; for example, classical Cisinski model structures [Cisinski, 2006] take the cofibrations to be the much larger set of all monomorphisms (in this case, this would mean all monomorphisms in the presheaves, not just the monomorphisms in the cube category), though in a constructive setting, cofibrations need to be restricted to be decidable in a certain sense. Thus, an important degree of freedom in the design of a cubical type theory is what cofibrations are allowed. Next, $[(\alpha \times z), 0/z]$ (an arbitrary collection of faces crossed with an interval, and connected at $z = 0$) is a contractible subshape of Ψ, and is a *trivial cofibration*. The Kan operation says that lifting problems specified by a trivial cofibration can be solved in any type, because all types are intended to be *fibrations*.

Syntactically, a next problem to solve is that the above rule has a designated interval variable z in the rule's *conclusion*'s context, which means substitution will not be admissible in the standard way—we cannot think of the action of the cube maps as simply replacing interval variables with interval terms. The way to resolve this is to define a Kan *composition* operation instead of a Kan *filling* operation, which means that the operation produces only the "missing face" of an open box rather than the "missing face and inside". This is the Kan operation used in Cohen, Coquand, Huber, and Mörtberg [2018], with the de Morgan cube category, and can be drawn as follows (see Sattler [2017]):



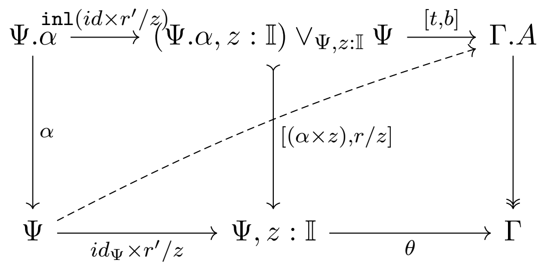
$$\frac{\begin{array}{c}\Psi, z : \mathbb{I} \vdash A\,\mathtt{Type} \\ \Psi.\alpha, z : \mathbb{I}; \Gamma \vdash t : A \\ \Psi; \Gamma \vdash b : A\langle 0/z\rangle \\ \Psi.\alpha; \Gamma \vdash t\langle 0/z\rangle \equiv b : A\langle 0/z\rangle\end{array}}{\begin{array}{c}\Psi; \Gamma \vdash \mathtt{com}_{z.A}(\alpha \mapsto z.t)b : A\langle 1/z\rangle \\ \Psi.\alpha; \Gamma \vdash \mathtt{com}_{z.A}(\alpha \mapsto z.t)b \equiv t\langle 1/z\rangle : A\langle 1/z\rangle\end{array}}$$

Rather than getting a map from the middle-bottom, we get a map from the bottom-left, which is the "1-endpoint", or *composite* of the filling problem. Commutativity of the upper triangle says that at α, this restricts to the 1-endpoint of the tube. The reason that the restriction to composition is sufficient is that filling can in fact be derived from composition using connections.



$$\frac{\begin{array}{c}\Psi \vdash r' : \mathbb{I} \\ \Psi, z : \mathbb{I} \vdash A\,\mathtt{Type} \\ \Psi.\alpha, z : \mathbb{I}; \Gamma \vdash t : A \\ \Psi; \Gamma \vdash b : A\langle 0/z\rangle \\ \Psi.\alpha; \Gamma \vdash t\langle 0/z\rangle \equiv b : A\langle 0/z\rangle\end{array}}{\begin{array}{c}\Psi; \Gamma \vdash \mathtt{com}_{z.A}(\alpha \mapsto z.t)b : A\langle r'/z\rangle \\ \Psi.\alpha; \Gamma \vdash \mathtt{com}_{z.A}(\alpha \mapsto z.t)b \equiv t\langle r'/z\rangle : A\langle r'/z\rangle\end{array}}$$

However, Coquand's earlier Kan operation for diagonals [Coquand, 2014b] generalizes this composition operation (we present a simplified version first):

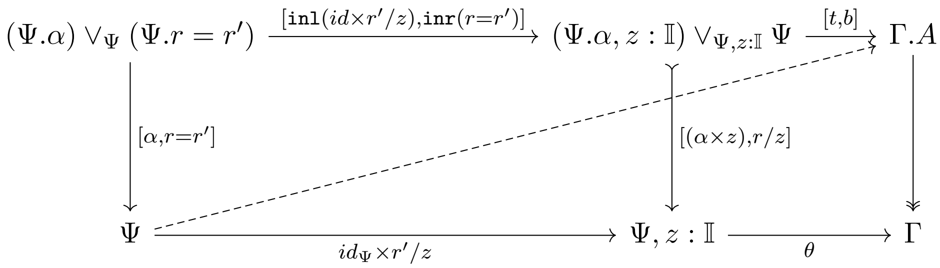


$$\frac{\begin{array}{l}\Psi \vdash r : \mathbb{I}\\ \Psi \vdash r' : \mathbb{I}\\ \Psi, z : \mathbb{I}; \Gamma \vdash A\,\texttt{Type}\\ \Psi.\alpha, z : \mathbb{I}; \Gamma \vdash t : A\\ \Psi; \Gamma \vdash b : A\langle r/z\rangle\\ \Psi; \alpha; \Gamma \vdash t\langle r/z\rangle \equiv b : A\langle r/z\rangle\end{array}}{\begin{array}{c}\Psi; \Gamma \vdash \texttt{com}_A^{z:r\to r'}\,(\alpha \mapsto z.t)\,(b) : A\langle r'/z\rangle\\ \Psi.\alpha; \Gamma \vdash \texttt{com}_A^{z:r\to r'}\,(\alpha \mapsto z.t)\,(b) \equiv t\langle r'/z\rangle : A\langle r'/z\rangle\end{array}}$$

The generalization is that the "source" and "target" of the composition problem are not restricted to 0 and 1, the endpoints of the interval, but can also be other maps $\Psi \to \mathbb{I}$ in the cube category; we write r and r' for arbitrary such maps. Thus, this Kan operation adopts a more permissive notion of both trivial cofibration (the middle map in the diagram) and allowed composite (the left-hand square in the diagram) than the Coquand [2014a] operation (with connections and reversal, the generalization is not necessary because the more general filling problems can be encoded). In the case of the Cartesian cube category, the additional trivial cofibrations/composites that are allowed by this Kan operation are product projections from the context (variables). We read the notation as "compose from r to r' in the z direction, with the tube t (which can also depend on z) on α, starting at r with b." Composing to or from a variable, as in $\texttt{com}_A^{z:0\to x}\,(b)$ or $\texttt{com}_A^{z:x\to 0}\,(b)$, can be thought of as "moving" an element b to or from a diagonal, because the source/target x can also occur in b. For example, consider the case where source is a variable x, the cap depends on x, and the tube is empty, $\texttt{com}_A^{z:x\to 0}\,(b)$. Here, $x : \mathbb{I} \vdash b : A\langle x/z\rangle$ is a heterogeneous path in the line A, while $\texttt{com}_A^{z:x\to 0}\,(b) : A\langle 0/x\rangle$ is a homogeneous path in the fiber over 0, whose endpoints are $\texttt{com}_A^{z:1\to 0}\,(b)$ and b. So, this instance of composition is a *homogenization* operation that turns a heterogeneous path ("path over") into a homogeneous-path-with-a-`transport`. Dually, Kan filling should be $\texttt{com}_A^{z:r\to z'}\,(\alpha \mapsto z.t)\,(b)$ for a fresh variable z'—filling is obtained by degenerating all components of the composition problem in a fresh direction, and then moving to the diagonal between the fresh direction and the original filling direction.

The full operation given by Coquand [2014b] includes an additional equality constraint, which is necessary for deriving filling and homogenization in this way: we need $\texttt{com}_A^{z:r\to r}\,(\alpha \mapsto t)\,(b) \equiv b$. That is, moving from r to itself is the identity, whether r is 0, 1 or a variable. Thus, Coquand [2014b] requires that on $r = r'$, the composite is b, which we can notate as follows:

Definition 1 (Diagonal Kan composition [Coquand, 2014b]).



$$\frac{\begin{array}{c}\Psi \vdash r : \mathbb{I} \quad \Psi \vdash r' : \mathbb{I} \quad \Psi, z : \mathbb{I}; \Gamma \vdash A\,\texttt{Type} \\ \Psi.\alpha, z : \mathbb{I}; \Gamma \vdash t : A \quad \Psi; \Gamma \vdash b : A\langle r/z\rangle \quad \Psi; \alpha; \Gamma \vdash t\langle r/z\rangle \equiv b : A\langle r/z\rangle\end{array}}{\begin{array}{c}\Psi; \Gamma \vdash \texttt{com}_A^{z:r \to r'}\,(\alpha \mapsto z.t)\,(b) : A\langle r'/z\rangle \\ \Psi.\alpha; \Gamma \vdash \texttt{com}_A^{z:r \to r'}\,(\alpha \mapsto z.t)\,(b) \equiv t\langle r'/z\rangle : A\langle r'/z\rangle \\ \Psi.r = r'; \Gamma \vdash \texttt{com}_A^{z:r \to r'}\,t\,(b) \equiv b : A\langle r/z\rangle\end{array}}$$

The syntactic rule is the composition rule given in Brunerie and Licata [2014], updated to use boundary formulas; variations on this rule were also used in Angiuli et al. [2016]; Angiuli and Harper [2017]. The presentation of the composition principle using boundary formulas is inspired by the diagonal constraints of Angiuli et al. [2017b]: In stating the boundary, we have used a more general subobject/formula than above. Before, we had only used equations $x = 0$ and $x = 1$, but here, we also need a general equality $r =_{\mathbb{I}} r'$, which semantically is a subobject of Ψ determined by pulling back the diagonal map $\mathbb{I} \to \mathbb{I} \times \mathbb{I}$. The pushout-of-pullback in the top-left corner of the diagram encodes both the "restricts to t on α" and the "restricts to b on $r = r'$" constraints (and compatibility on both).

Coquand [2014b]; Brunerie and Licata [2014] give definitions of diagonal Kan composition (satisfying this $r = r'$ constraint) for Π, Σ, path, and some higher inductive base types. Angiuli et al. [2016]; Angiuli and Harper [2017] study variations on these definitions in an operational setting, and define this Kan operation for Π, Σ, path, higher inductive base types, and gluing with a strict isomorphism. However, this strict $r = r'$ constraint was a long-time obstacle to defining univalence/gluing with an equivalence, or showing that the universe itself is fibrant. For example, diagonal Kan composition can be encoded using connections and the $0 \to 1$ Kan composition, and when one does so, the $r = r'$ constraint becomes an instance of the regularity condition that proved problematic in that model.

Here, we show that it is possible to constructively define diagonal Kan composition for glue (equivalence extension) types, *if* the cofibrations α include the diagonal map $\mathbb{I} \to \mathbb{I} \times \mathbb{I}$ from the cube category, as in Angiuli et al. [2017b]. Viewing subobjects as propositions (maps into the subobject classifier of the topos), this corresponds to taking the proposition $r =_{\mathbb{I}} r'$, for any maps $r, r' : \Psi \to \mathbb{I}$ in the cube category, to be a cofibration. Geometrically, this corresponds to attaching faces on the diagonal of an open box—interestingly, like connections, this has somewhat of a simplicial flavor. In a classical setting, diagonal maps are cofibrations in, for example, Cisinski model structures, where the cofibrations are all monomorphisms. In a constructive setting, there is an obligation that cofibrations be decidable (in a sense that will be made precise below), but this is true for pullbacks of the diagonal $\mathbb{I} \to \mathbb{I} \times \mathbb{I}$ because equality of maps in the cube category is decidable. Fibrant glue types imply Kan and univalent universes, in the same manner as in Cohen, Coquand, Huber, and Mörtberg [2018].

Relative to the prior work by Awodey [2018b] on constructing models of type theory with the Cartesian cube category, our Kan operation differs by using a different class of generating trivial cofibrations (the source can be an arbitrary map in the cube category, not only an endpoint), by taking composition rather than filling as primitive (but for a general enough notion of composite that it includes filling), and by using a different class of generating cofibrations (allowing the diagonal $\mathbb{I} \to \mathbb{I} \times \mathbb{I}$). Our work could be fully compared with Awodey [2018b]'s model by, for example, building Quillen model categories for both, and comparing the generated classes of cofibrations and fibrations: even though the generating (trivial) cofibrations are different, the full class of (trivial) cofibrations could still be the same (perhaps only classically). We leave such an investigation to future work. Our definition of the universe satisfies univalence but not regularity, and therefore our interval-based path type satisfies the J computation rule of Martin-Löf type theory only weakly (up to a path). Awodey's supports regularity/normality but not yet (to our knowledge) fibrant univalent universes. Thus, it remains open whether it is possible to combine the benefits of both approaches, and have a path type given as exponentiation by an interval, which both satisfies the J computation rule strictly and supports univalent and Kan universes.

1.3 Contributions

In Section 2, we give a formal cubical type theory based on the Cartesian cube category with Π, Σ, path, identity, natural number, boolean, suspension (we comment on the generalization to pushouts), glue, and universe types. For each type, we give a judgemental equality rule "defining" the composition structure on that type. Of course, these "definitions" are just postulated axioms of judgemental equality, so it is important to validate them.

Orton and Pitts [2016]; Birkedal et al. [2018]; Orton and Pitts [2018] have developed a technique for describing cubical models in the internal logic of a 1-topos, by postulating an interval object, cofibrations, and certain other operations. This approach can be mechanized: in principle, one should use an extensional type theory, but in a pinch we can use Agda with function extensionality and uniqueness of identity proofs as a substitute [Hofmann, 1995]. Orton and Pitts [2016] mechanize much of the connections model [Cohen, Coquand, Huber, and Mörtberg, 2018] in this style. A later extension allows the internal description of universes [Licata et al., 2018], so that one can prove that they are fibrant and univalent.

In Section 3, we describe a mechanized constructive model in Cartesian cubical sets using the internal language technique. We have formalized the definition of diagonal Kan composition for glue, Π, Σ, path, identity, natural number, boolean, and pushout types, and the universe itself, and shown that the universe is univalent.[8] Our mechanization postulates the definitions of interval, Π, Σ, positive types, and (exact) equality in the internal logic—i.e. we obtain the formation, introduction, elimination, and $\beta\eta$ rules for these types from the metalanguage, Agda. Then we make certain postulates about an interval type and cofibrations. Relative to the axioms in [Orton and Pitts, 2016], we replace the axioms for connections on the interval with the axiom that the diagonal inclusion $\mathbb{I} \to \mathbb{I} \times \mathbb{I}$ in the cube category $\mathbb{C}$ is a cofibration (i.e. that the proposition $r =_{\mathbb{I}} r'$ in the internal logic is cofibrant, for arbitrary terms r, r' in the interval). We also use "propositional univalence" (interprovable cofibrations are equal) and the axiom that cofibrations are closed under conjunction only to construct identity types (where J on `refl` satisfies an exact equality) from path types using Swan's technique [Cohen, Coquand, Huber, and Mörtberg, 2018]. From these

[8] `https://github.com/dlicata335/cart-cube`

assumptions, our mechanization verifies all of the details of the construction of Kan composition operations for these types, e.g. checking that the constructions in Section 2.11 type check and have the correct boundaries.

The mechanized internal language proof can be interpreted in presheaf toposes. In any presheaf topos, let Ω_{dec} be the presheaf of decidable sieves [Orton and Pitts, 2016, Definition 6.2]: at each stage Ψ, $\Omega_{dec}(\Psi)$ is the set of sieves on Ψ (precomposition-closed of subsets of $\hom_C(-, \Psi)$) with the property that for a given map $\rho : \Psi' \to_C \Psi$ it is decidable whether ρ is in the sieve.[9] Ω_{dec} is a subobject of the subobject classifier Ω. Then the internal language construction implies:

Theorem 1. *Let C be a finite product category with an object $\mathbb{I}$, with maps $0, 1 : 1 \to \mathbb{I}$ with $0 \neq 1$. In $\hat{C} := Sets^{C^{op}}$, suppose Cof is a subobject of Ω_{dec}, which is closed under $=_{\mathbb{I}}$, $\vee$ and $\forall x : \mathbb{I}-$. Then there is (for each size level i) a universe U_i classifying those semantic type families of size i equipped with a diagonal Kan composition structure (Definition 1) for generating cofibrations classified by Cof. U_i is closed under semantic Π, Σ, path, and glue types, and is itself Kan (U_{i+1} has a code for U_i). If $\hat{C}$ has the cubical sets corresponding to boolean, natural number, and pushout types, then U_i is closed under those as well. Finally, if Cof is closed under pullbacks, then U_i is closed under identity types as well.*

For example, we can take C to be the Cartesian cube category, with *Cof* anything from a minimal notion of cofibration, closed under just $\vee$ and $=_{\mathbb{I}}$ with $\forall$ defined by quantifier elimination, to a maximal one consisting of Ω_{dec} itself. We give a proof that the axioms used in our formalization are true in cubical sets on the Cartesian cube category, which is mostly the same as the argument in [Orton and Pitts, 2016; Licata et al., 2018]. We briefly sketch the the interpretation of the syntax in the model, noting that the definitions of the Kan operations in the syntax and internal language model follow each other line by line.

The axioms are also true in cubical sets on the de Morgan cube category, which gives a model in the same category but using a different notion of cofibration than Cohen, Coquand, Huber, and Mörtberg [2018]. However, the two Kan operations are interderivable if we have both *both* connections (and reversal) and diagonal cofibrations, which provides a potential route to comparing the approaches.

For reference, here is a summary of why each feature of our Kan operation, assumption about the interval, and closure condition for cofibrations is necessary for our definitions:

- Path types use the fact that there is an interval with endpoints 0 and 1.
- Composition for Π types use the fact that Kan filling is derivable from composition, and that the source and target of the composition operation can be interchanged. Therefore, this definition of Π types would not work without the generalized trivial cofibrations that we consider, because interchanging a filler results in a composite from a variable.
- Composition for Σ types use Kan filling.
- Composition for path types use the fact that endpoints $x = 0$ and $x = 1$ are cofibrations, and that $\vee$ is.
- Composition for strict base types (natural numbers, booleans) use the connectedness axiom.

[9] This not the same as a proposition being decidable ($\phi \vee \neg\phi$) in the internal logic of the topos.

- Composition for higher inductive types uses the reduction of composition to coercion and homogeneous composition, which in turn uses “homogenization”, i.e. the generalized set of trivial cofibrations.
- Constructing glue types uses the strictification axiom, which in a constructive model requires cofibrations to be decidable sieves.
- Composition for glue types uses closure of cofibrations under $\vee$, $=_{\mathbb{I}}$, and $\forall$.
- Composition for identity types uses closure of cofibrations under $\wedge$, and propositional univalence for cofibrations.

Acknowledgments We are very grateful to the late Vladimir Voevodsky for his vision and leadership, creating a rich research area that has captivated all of us over the past 9 years and inspired the PhD theses of all four junior authors of this paper. We thank Jeremy Avigad, Steve Awodey, Evan Cavallo, Cyril Cohen, Fredrik Forsberg, Simon Huber, Ian Orton, Ed Morehouse, Anders Mörtberg, Andrew Pitts, Christian Sattler, Mike Shulman, Bas Spitters, and Jon Sterling for many helpful discussions and insights over the years leading to this work.

This material is based on research sponsored by The United States Air Force Research Laboratory under agreement number FA9550-15-1-0053 and and FA9550-16-1-0292. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the United States Air Force Research Laboratory, the U.S. Government, or Carnegie Mellon University.

2 Type theory

2.1 Overview of judgements

Cubical type theory extends the judgements of standard Martin-Löf type theory with interval variables and dimension formulas, represented by the following syntactic classes:

- Ψ is a *dimension context* representing an object $\Psi \in \mathbb{C}$.
- $\Psi \vdash r : \mathbb{I}$ is a *dimension term* representing a map $\Psi \to \mathbb{I}$ in $\mathbb{C}$.
- $\Psi \vdash \phi\,\mathsf{formula}$ is a *dimension formula* representing a subobject of Ψ.
- $\Psi \vdash \alpha\,\mathsf{cofib}$ is a *cofibration*, a special kind of dimension formula used in Kan operations.
- $\phi \vdash_{\Psi} \alpha$ is the implication ordering on subobjects of Ψ.
- $\Psi; \phi \vdash \Gamma\ \mathsf{ctx}$ is a (ordinary) *context* relative to a dimension context and dimension formula.
- $\Psi; \phi; \Gamma \vdash A\ \mathtt{Type}_i$ is a type relative to a dimension context, dimension formula, and context. We suppress levels i throughout, but formally types are stratified by size.
- $\Psi; \phi; \Gamma \vdash a : A$ is a term of a type, relative to a dimension context, dimension formula, and context.

We also have two forms of judgemental equality: $\Psi;\phi;\Gamma \vdash A \equiv A'\ \mathtt{Type}$ and $\Psi;\phi;\Gamma \vdash a \equiv a' : A$. We interleave discussion of the syntax and discussion of the "standard" model in presheaves on a cube category, though we plan to investigate whether other models are possible.

2.2 Contexts

We have three kinds of contexts: dimension contexts Ψ, dimension formula contexts ϕ, geometrically representing subshapes of cubes, and ordinary contexts Γ contains term variables. Dimension contexts are non-dependent; Ψ is in scope in ϕ, but ϕ is not internally dependent.

$$\begin{array}{rcl} \Psi & ::= & \cdot \mid \Psi, x : \mathbb{I} \\ \phi & ::= & \cdot \mid \phi, \alpha \\ \Gamma & ::= & \cdot \mid \Gamma, x : A \end{array}$$

The Ψ context has no formation conditions (aside from the usual invariant that variables are distinct, when implemented concretely), while ϕ and Γ require the context entries to be well-formed:

$$\frac{}{\Psi \vdash \cdot\ \mathsf{formula}} \qquad \frac{\Psi \vdash \phi\ \mathsf{formula} \quad \Psi \vdash \alpha\ \mathsf{cofib}}{\Psi \vdash \phi, \alpha\ \mathsf{formula}} \qquad \frac{}{\Psi;\phi \vdash \cdot\ \mathsf{ctx}} \qquad \frac{\Psi;\phi \vdash \Gamma\ \mathsf{ctx} \quad \Psi;\phi;\Gamma \vdash A\ \mathtt{Type}}{\Psi;\phi \vdash \Gamma, x : A\ \mathsf{ctx}}$$

We overload the letter x for both term and dimension variables, because x is traditionally used for term variables, and writing dimension variables as x, y, z is extremely helpful for drawing pictures. When both are in play at once, we may write term variables as a, b, c.

2.3 Dimension terms

Dimension terms are 0, 1, and variables.

$$r ::= 0 \mid 1 \mid x$$

We write $\Psi \vdash r : \mathbb{I}$ to mean that r is either 0 or 1 or a variable from Ψ:

$$\frac{}{\Psi \vdash 0 : \mathbb{I}} \qquad \frac{}{\Psi \vdash 1 : \mathbb{I}} \qquad \frac{x : \mathbb{I} \in \Psi}{\Psi \vdash x : \mathbb{I}}$$

The dimension context behaves like a standard hypothetical judgement in all other judgements—all rules treat dimension variables as placeholders, and do not, for example, inspect whether a term is a variable, or whether two variables are different. Thus, we have silent weakening and exchange (where the dotted line indicates an admissible rule):

$$\frac{\Psi, \Psi' \vdash J}{\Psi, x : \mathbb{I}, \Psi' \vdash J} \qquad \frac{\Psi, x' : \mathbb{I}, x : \mathbb{I}, \Psi' \vdash J}{\Psi, x : \mathbb{I}, x' : \mathbb{I}, \Psi' \vdash J}$$

and substitution, with its usual composition law:

$$\frac{\Psi, x : \mathbb{I}, \Psi' \vdash J \quad \Psi \vdash r : \mathbb{I}}{\Psi, \Psi' \vdash J\langle r/x \rangle} \qquad \frac{}{J\langle r/x\rangle\langle r'/y\rangle \equiv J\langle r'/y\rangle\langle r\langle r'/y\rangle/x\rangle}$$

Substitution of dimension terms is defined in a completely standard way (by induction on syntax, replacing variables with terms).

Semantically, a dimension context Ψ represents an object in the cube category, or the representable presheaf on it. A dimension term $\Psi \vdash r : \mathbb{I}$ is a map $\Psi \to \mathbb{I}$ in the cube category. Because all objects in the cube category are finite products of $\mathbb{I}$, we could define an n-place substitution judgement $\Psi \vdash \sigma : \Psi'$ representing all such maps as $|\Psi'|$-tuples of terms.

2.4 Cofibrations

We use a syntactic notion of cofibration, in the style introduced by Cohen, Coquand, Huber, and Mörtberg [2018]. Our presentation follows Riehl and Shulman [2017], and our discussion of their semantics follows Orton and Pitts [2016]; Sattler [2017].

Our type theory has a notion of *dimension formula*, written $\Psi \vdash \phi\,\mathsf{formula}$. Because Ψ is an object of the cube category $\mathbb{C}$, it gives rise to a representable presheaf $[\Psi] := \hom_{\mathbb{C}}(-, \Psi)$. We can therefore regard a syntactic boundary formula $\Psi \vdash \phi\,\mathsf{formula}$ in several equivalent ways:

1. A map in $\hat{\mathbb{C}}$ (i.e., a natural transformation) from $[\Psi]$ to Ω, where Ω is the subobject classifier of $\hat{\mathbb{C}}$ regarded as a topos.

2. A sieve on Ψ: a set of pairs $(\Psi', \hom_{\mathbb{C}}(\Psi', \Psi))$ closed under precomposition by arbitrary maps in $\mathbb{C}$. Since the subobject classifier in a presheaf topos sends an object Ψ to the set of sieves on Ψ, this is the same as (1) by Yoneda.

3. A subobject (subpresheaf) of $[\Psi]$, i.e., another presheaf $\Psi.\phi$ with a monic natural transformation $\phi : \Psi.\phi \hookrightarrow [\Psi]$ that includes the subobject into the original. (A subobject is technically an isomorphism class of such monos, identifying (A, h) and (B, k) when there is an isomorphism between A and B that sends h to k.) The universal property of the subobject classifier expresses that this is equivalent to (1).

   A map between subobjects ϕ_1 and ϕ_2 is a morphism $f : (\Psi.\phi_1 \to \Psi.\phi_2)$ that commutes with the inclusions into Ψ. Any such commuting map is a monomorphism, because any map f such that $f; \phi_1 = \phi_2$ where ϕ_2 is mono is itself mono. Moreover, because ϕ_1 is mono, any two $f, g : \Psi.\phi_1 \to \Psi.\phi_2$ that satisfy $f; \phi_1 = \phi_2$ and $g; \phi_1 = \phi_2$ are equal—so we have a posetal subobject ordering. We write $\mathbf{Sub}(\Psi)$ for the category (poset) of subobjects.

Cofibrations are a designated subset of dimension formulas used to generate the lifting problems subject to Kan composition. We write $\mathbf{Cofibs}(\Psi)$ for the category of cofibrations into Ψ, whose objects are pairs $(\Psi_0, \Psi_0 \rightarrowtail \Psi)$ such that the map is a mono and cofibration. Equivalently, one can define a cofibration classifier as a subobject of the subobject classifier. The judgement $\Psi \vdash \alpha\,\mathsf{cofib}$ is interpreted as an object $(\Psi.\alpha, \alpha : \Psi.\alpha \hookrightarrow \Psi)$ in $\mathbf{Cofibs}(\Psi)$ or as a map into the cofibration classifier. The rules for this judgement assert that cofibrations are closed under certain operations.

Rules for Dimension Formulas Syntactically, dimension formulas are represented by a context ϕ that is a list of cofibrations α. The empty boundary context is the identity map, while context extension ϕ, α is the pullback:

$$\begin{array}{ccc} \Psi.(\phi, \alpha) & \dashrightarrow & \Psi.\alpha \\ \downarrow & & \downarrow \\ \Psi.\phi & \hookrightarrow & \Psi \end{array}$$

with either of the equal maps $\Psi.(\phi, \alpha) \hookrightarrow \Psi$ (which are mono because they are the diagonal map of the pullback of two monos, making all four maps monos). In any topos, this pullback is the product $\phi \times \alpha$ in $\mathbf{Sub}(\Psi)$—regarding ϕ and α as maps into the subobject classifier, the pullback is their composition with $\wedge : \Omega \times \Omega \to \Omega$—and therefore the rules can treat ϕ, α as context extension.

In contrast with Cohen, Coquand, Huber, and Mörtberg [2018], we do not require these pullbacks to be cofibrations, except to construct identity types with an exact equality on `refl` (see Section 2.16).

Cofibrations Cofibrations are defined by the following rules:

$$\frac{\Psi \vdash r : \mathbb{I} \quad \Psi \vdash r' : \mathbb{I}}{\Psi \vdash r = r' \ \mathsf{cofib}} \qquad \frac{\Psi \vdash \alpha_1 \ \mathsf{cofib} \quad \Psi \vdash \alpha_2 \ \mathsf{cofib}}{\Psi \vdash \alpha_1 \vee \alpha_2 \ \mathsf{cofib}} \qquad \frac{\Psi, x : \mathbb{I} \vdash \alpha \ \mathsf{cofib}}{\Psi \vdash \forall x.\alpha \ \mathsf{cofib}}$$

These rules assert that certain subobjects of Ψ are in the set of cofibrations; as maps into the subobject classifier, these subobjects are exactly the formulas suggested by the notation. The admissible substitution principle for $\Psi, x : \mathbb{I} \vdash \alpha \ \mathsf{cofib}$ says that cofibrations are closed under pullback along an arbitrary map in $\mathbb{C}$.

It is instructive to unpack these maps into the subobject classifier as monomorphisms. The first form of cofibration, $r = r'$, is (the Yoneda embedding of) a face, diagonal, or identity, or the unique map out of $\emptyset$ (the initial object of $\hat{\mathbb{C}}$), depending on the values of r and r':

$$\begin{array}{ccccc}
\underline{x = 0 \text{ and } 0 = x} & \underline{x = 1 \text{ and } 1 = x} & \underline{x = y} & \underline{r = r} & \underline{0 = 1 \text{ and } 1 = 0} \\
\Psi & \Psi & \Psi, x : \mathbb{I} & \Psi & \emptyset \\
\downarrow{\scriptstyle id_\Psi \times 0/x} & \downarrow{\scriptstyle id_\Psi \times 1/x} & \downarrow{\scriptstyle id_\Psi \times x/y} & \downarrow{\scriptstyle id} & \downarrow{\scriptstyle !} \\
\Psi, x : \mathbb{I} & \Psi, x : \mathbb{I} & \Psi, x : \mathbb{I}, y : \mathbb{I} & \Psi & \Psi
\end{array}$$

Admitting the diagonal inclusion / equality on the interval as a cofibration is the key ingredient used to define the diagonal Kan operation for the universe.

The second cofibration rule, $\alpha_1 \vee \alpha_2$, asserts that the coproduct of two cofibrations $\alpha_1 + \alpha_2$ in $\mathbf{Sub}(\Psi)$ is a cofibration. In terms of maps $\alpha_i : \Psi.\alpha_i \to \Psi$, this unpacks in any topos to the pushout of the pullback, which encodes an idea of *coherence* when α_1 and α_2 are both true:

$$\begin{array}{ccc}
\Psi.(\alpha_1, \alpha_2) & \xrightarrow{f} & \Psi.\alpha_1 \\
\downarrow{\scriptstyle s} & & \downarrow{\scriptstyle \alpha_1} \\
\Psi.\alpha_2 & \xrightarrow{\alpha_2} & \Psi
\end{array}
\qquad
\begin{array}{ccc}
\Psi.(\alpha_1, \alpha_2) & \xrightarrow{f} & \Psi.\alpha_1 \\
\downarrow{\scriptstyle s} & & \downarrow \\
\Psi.\alpha_2 & \longrightarrow & \Psi.(\alpha_1 \vee \alpha_2)
\end{array}$$

with the cofibrations $\Psi.(\alpha_1 \vee \alpha_2) \rightarrowtail \Psi$ given by the universal property of the pushout applied to α_1 and α_2.

The third cofibration, $\forall x.\alpha$, takes as argument $\Psi, x : \mathbb{I} \vdash \alpha \ \mathsf{cofib}$, a map $(\Psi, x : \mathbb{I}).\alpha \rightarrowtail (\Psi, x : \mathbb{I})$, or equivalently, an object of $\mathbf{Cofibs}(\Psi, x : \mathbb{I})$. There is a functor $- \times \mathbb{I} : \mathbf{Sub}(\Psi) \to \mathbf{Sub}(\Psi, x : \mathbb{I})$, which given a map $\Psi.\beta \hookrightarrow \Psi$ produces $\beta \times 1 : \Psi.\beta, x : \mathbb{I} \hookrightarrow \Psi, x : \mathbb{I}$. (Syntactically, this is weakening $\Psi \vdash \beta \ \mathsf{cofib}$ to $\Psi, x : \mathbb{I} \vdash \beta \ \mathsf{cofib}$.) This map is a cofibration, because it is a pullback of a cofibration along a map in $\mathbb{C}$. In any topos, the $\forall$ quantifier on subobjects is the right adjoint to pullback along the projection, $\forall : \mathbf{Sub}(\Psi, x : \mathbb{I}) \to \mathbf{Sub}(\Psi)$; therefore, our rule asserts that this right adjoint preserves cofibrations.

For most of our development, we do not need any *equations* between cofibrations, like $(\alpha \vee (0 = 1)) \equiv \alpha$ or $(\alpha_1 \vee \alpha_2) \equiv (\alpha_2 \vee \alpha_1)$, and require only that these cofibrations are interprovable. Semantically, these equations do hold by the "propositional univalence" of the subobject classifier in a topos (i.e., interprovable propositions are equal). We do, however, need these equations when constructing identity types from path types (see Section 2.16).

Subboundaries Next, the judgement $\phi \vdash_\Psi \alpha$ states that there is a map from ϕ to α in $\mathbf{Sub}(\Psi)$, i.e., a morphism $\Psi.\phi \to \Psi.\alpha$ that commutes over Ψ. Geometrically, this captures the idea that ϕ is a subspace of α, both as subspaces of Ψ. For example, $x = 0 \vdash_{x:\mathbb{I},y:\mathbb{I}} (x = 0) \vee (y = 1)$ is the inclusion of the left-hand side of a square into the left and top sides of a square, and $x = 0, y = 1 \vdash_{x:\mathbb{I},y:\mathbb{I}} x = 0$ is the inclusion of the top-left corner of a square into the left-hand side.

Because $\mathbf{Sub}(\Psi)$ is posetal, as previously discussed, we omit proof terms for this judgement in order to identify all derivations of $\phi \vdash_\Psi \alpha$. The rules are just the usual natural deduction rules for these connectives. The rules for $\vee$, for example, express the fact that $\alpha_1 \vee \alpha_2$ is a coproduct in $\mathbf{Sub}(\Psi)$, while the rules for context extension state that it is a product in $\mathbf{Sub}(\Psi)$.

$$\frac{\alpha \in \phi}{\phi \vdash_\Psi \alpha} \qquad \frac{\phi \vdash_\Psi \alpha_1}{\phi \vdash_\Psi \alpha_1 \vee \alpha_2} \qquad \frac{\phi \vdash_\Psi \alpha_2}{\phi \vdash_\Psi \alpha_1 \vee \alpha_2} \qquad \frac{\phi, \alpha_1 \vdash_\Psi \beta \quad \phi, \alpha_2 \vdash_\Psi \beta \quad \phi \vdash_\Psi \alpha_1 \vee \alpha_2}{\phi \vdash_\Psi \beta}$$

$$\frac{\phi \vdash_\Psi 0 = 1}{\phi \vdash_\Psi \beta} \qquad \frac{\Psi \vdash r : \mathbb{I}}{\phi \vdash_\Psi (r = r)} \qquad \frac{\begin{array}{l}\Psi, x : \mathbb{I} \vdash \phi' \ \mathsf{formula} \\ \Psi, x : \mathbb{I} \vdash \alpha \ \mathsf{cofib} \\ \phi \vdash_\Psi r = r' \\ \phi, \phi'\langle r/x \rangle \vdash_\Psi \alpha\langle r/x \rangle\end{array}}{\phi, \phi'\langle r'/x \rangle \vdash_\Psi \alpha\langle r'/x \rangle}$$

$$\frac{\phi \vdash_{\Psi,x:\mathbb{I}} \alpha}{\phi \vdash_\Psi \forall x.\alpha} \qquad \frac{\phi \vdash_\Psi \forall x.\alpha \quad \Psi \vdash r : \mathbb{I}}{\phi \vdash_\Psi \alpha\langle r/x \rangle}$$

Boundaries and partial elements In type and term formation judgements, we use a context $\Psi; \phi; \Gamma \vdash J$. The $(\Psi; \phi)$ part can be understood as the domain $\Psi.\phi$ of the mono $\Psi.\phi \hookrightarrow [\Psi]$, or intuitively, "the subset of Ψ on which ϕ holds." All judgements are compatible with admissible weakening, exchange, and substitution principles:

$$\frac{\Psi; \phi, \phi' \vdash J}{\Psi; \phi, \alpha, \phi' \vdash J} \qquad \frac{\Psi; \phi, \alpha, \beta, \phi' \vdash J}{\Psi; \phi, \beta, \alpha, \phi' \vdash J} \qquad \frac{\Psi; \phi, \alpha \vdash J \quad \phi \vdash_\Psi \alpha}{\Psi; \phi \vdash J}$$

Most of our cofibrations are left-invertible, and require left rules towards other judgements, such as term and type formation. Here, we give rules for both, though we could avoid duplication if we identified types and elements of a universe.

In an ambient context $\Psi; \phi$, a *partial element* of a type is a term $\Psi; \phi, \alpha \vdash a : A$ for some cofibration α. We will sometimes write $\alpha \vdash a : A$, leaving the ambient content implicit, allowing us to conveniently and concisely state many judgemental equality axioms as "$\alpha \vdash t \equiv t'$". Such rules should be understood to mean that, in a general $\Psi; \phi; \Gamma$, if t and t' are well-formed and moreover $\phi \vdash_\Psi \alpha$, the equation holds.

We axiomatize contradiction by the rules:

$$\frac{\phi \vdash_\Psi 0 = 1}{\Psi; \phi; \Gamma \vdash \mathtt{abort} : A} \qquad 0 = 1 \vdash u \equiv \mathtt{abort} \qquad \frac{\phi \vdash_\Psi 0 = 1}{\Psi; \phi; \Gamma \vdash \mathtt{abort}\ \mathtt{Type}} \qquad 0 = 1 \vdash A \equiv \mathtt{abort}$$

For $\vee$, we use the pushout-of-pullback characterization, because $\Psi; \phi$ is the domain of $\Psi.\phi$ of the inclusion, which in this case is (below) a pushout of a pullback:

$$\frac{\phi \vdash_\Psi \alpha \vee \beta \quad \Psi; \phi, \alpha; \Gamma \vdash t : A \quad \Psi; \phi, \beta; \Gamma \vdash u : A \quad \Psi; \phi, \alpha, \beta; \Gamma \vdash t \equiv u : A}{\Psi; \phi; \Gamma \vdash [\alpha \mapsto t, \beta \mapsto u] : A}$$

$$\alpha \vdash [\alpha \mapsto t, \beta \mapsto u] \equiv t$$

$$\beta \vdash [\alpha \mapsto t, \beta \mapsto u] \equiv u$$

$$\alpha \vee \beta \vdash t \equiv [\alpha \mapsto t, \beta \mapsto t]$$

$$\frac{\phi \vdash_\Psi \alpha \vee \beta \quad \Psi;\phi,\alpha;\Gamma \vdash A\ \texttt{Type} \quad \Psi;\phi,\beta;\Gamma \vdash B\ \texttt{Type} \quad \Psi;\phi,\alpha,\beta;\Gamma \vdash A \equiv B\ \texttt{Type}}{\begin{array}{c}\Psi;\phi;\Gamma \vdash [\alpha \mapsto A, \beta \mapsto B]\ \texttt{Type} \\ \alpha \vdash [\alpha \mapsto A, \beta \mapsto B] \equiv A \\ \beta \vdash [\alpha \mapsto A, \beta \mapsto B] \equiv B \\ \alpha \vee \beta \vdash A \equiv [\alpha \mapsto A, \beta \mapsto A]\end{array}}$$

We also have congruence rules for $r = r'$:

$$\frac{\phi \vdash_\Psi r = r' \quad \Psi, x : \mathbb{I}; \phi, \phi'; \Gamma \vdash a : A}{\Psi;\phi,\phi\langle r'/x\rangle;\Gamma \vdash a\langle r/x\rangle \equiv a\langle r'/x\rangle : A\langle r/x\rangle} \qquad \frac{\phi \vdash_\Psi r = r' \quad \Psi, x : \mathbb{I}; \phi, \phi'; \Gamma \vdash A\ \texttt{Type}}{\Psi;\phi,\phi\langle r'/x\rangle;\Gamma \vdash A\langle r/x\rangle \equiv A\langle r'/x\rangle\ \texttt{Type}}$$

From weakening and congruence we can derive the following, in which the double-line indicates a *derivable* (*not* invertible, as it is often used) rule:

$$\frac{\Psi \vdash r : \mathbb{I} \quad \Psi;\phi\langle r/x\rangle;\Gamma\langle r/x\rangle \vdash a : A\langle r/x\rangle}{\Psi, x : \mathbb{I}; \phi, (x = r); \Gamma \vdash a : A}$$

When we have a nested $\vee$ like $(\alpha_1 \vee \alpha_2 \vee \alpha_3 \vee \ldots \vee \alpha_n)$, we write $[\alpha_1 \mapsto t_1, \ldots, \alpha_n \mapsto t_n]$ for a nested case with the same associativity as the type itself.

2.5 Judgemental equality

All judgements respect equality of types and terms in all positions, e.g.

$$\frac{\Psi;\phi;\Gamma \vdash a : A \quad \Psi;\phi;\Gamma \vdash A \equiv A'\ \texttt{Type}}{\Psi;\phi;\Gamma \vdash a : A'}$$

and all equality judgements include reflexivity, symmetry, transitivity, and compatibility/congruence rules for each type or term-former (which we do not write out explicitly).

We will often declare judgemental equality rules by simply writing $u \equiv v$; this should be taken to mean that the rule applies in all contexts, and has typing premises for each meta-variable appearing in the rule, which ensure that u and v are well-typed.

2.6 Term structural rules

Our structural rules for terms are typical:

$$\frac{x : A \in \Gamma}{\Psi;\phi;\Gamma \vdash x : A} \qquad \frac{\Psi;\phi;\Gamma,\Gamma' \vdash J}{\Psi;\phi;\Gamma, x : A, \Gamma' \vdash J}$$

$$\frac{\Psi;\phi;\Gamma, y : B, x : A, \Gamma' \vdash J}{\Psi;\phi;\Gamma, x : A, y : B, \Gamma' \vdash J} \qquad \frac{\Psi;\phi;\Gamma, x : A, \Gamma' \vdash J \quad \Psi;\phi;\Gamma \vdash a : A}{\Psi;\phi;\Gamma, \Gamma'[a/x] \vdash J[a/x]}$$

Composition laws for substitution hold syntactically:

$$\begin{array}{rcll} u[a/x] & \equiv_\alpha & u & \text{when } x \# u \\ u[a/x][b/y] & \equiv_\alpha & u[b/y][a[b/y]/x] & \end{array}$$

We additionally have a composition law for term substitution and dimension substitution:

$$u[a/y]\langle r_0/x_0\rangle \equiv_\alpha u\langle r_0/x_0\rangle[a\langle r_0/x_0\rangle/y]$$

2.7 Kan operation

All types in our syntax are *Kan* (or *fibrant*), i.e., equipped with the following operation:

$$\frac{\begin{array}{l}\Psi \vdash r, r' : \mathbb{I} \\ \Psi, z : \mathbb{I}; \phi; \Gamma \vdash A \; \texttt{Type} \\ \Psi, z : \mathbb{I}; \phi, \alpha; \Gamma \vdash t : A \\ \Psi; \phi; \Gamma \vdash b : A\langle r/z\rangle \\ \Psi; \phi, \alpha; \Gamma \vdash t\langle r/z\rangle \equiv b : A\langle r/z\rangle\end{array}}{\begin{array}{c}\Psi; \phi; \Gamma \vdash \texttt{com}_A^{z:r\to r'}\,(\alpha \mapsto z.t)\,(b) : A\langle r'/z\rangle \\ r = r' \vdash \texttt{com}_A^{z:r\to r'}\,(\alpha \mapsto z.t)\,(b) \equiv b \\ \alpha \vdash \texttt{com}_A^{z:r\to r'}\,(\alpha \mapsto z.t)\,(b) \equiv t\langle r'/z\rangle\end{array}}$$

The boundary condition on $r = r'$ states that transporting from somewhere to itself is the identity. The boundary condition on α states that the Kan composite agrees with the r' side of the specified partial element $z.t$; this ensures, for example, that the boundary of a composition of a $\sqcup$ agrees with the top-left and top-right corners of the $\sqcup$. Congruence of judgemental equality for $\texttt{com}_A^{z:r\to r'}\,(\alpha \mapsto z.t)\,(b)$ requires, in particular, that judgementally equal types have the same Kan operation—this is the major challenge for glue types (equivalently, univalence and universes).

Various derived forms and lemmas are useful when defining Kan composition for certain types.

Filling from composition Kan *filling*, as opposed to Kan *composition*, is the "whole square" instead of just the "missing side". We can derive filling by composing to a fresh variable, or geometrically, degenerating and composing to a diagonal. For emphasis, we sometimes write:

$$\frac{\text{same premises as } \texttt{com} \qquad z\#(r, A, \alpha, z'.t, b)}{\Psi; \phi; \Gamma \vdash \texttt{fill}_A^{z':r\to z}\,(\alpha \mapsto z'.t)\,(b) := \texttt{com}_A^{z':r\to z}\,(\alpha \mapsto z'.t)\,(b) : A\langle z/z'\rangle}$$

One can easily verify the following boundary conditions:

$$\begin{array}{rcl}(\texttt{fill}_A^{z':r\to z}\,(\alpha \mapsto z'.t)\,(b))\langle r/z\rangle & \equiv & b \\ (\texttt{fill}_A^{z':r\to z}\,(\alpha \mapsto z'.t)\,(b))\langle r'/z\rangle & \equiv & \texttt{com}_A^{z':r\to r'}\,(\alpha \mapsto z'.t)\,(b) \\ \alpha \vdash \texttt{fill}_A^{z':r\to z}\,(\alpha \mapsto z'.t)\,(b) & \equiv & t[z \leftrightarrow z']\end{array}$$

Extending partial elements in contractible types Recall that a partial element of a type is an element under the hypothesis of some cofibration α, and that a *contractible* type A is one for which $\mathsf{Contractible}(A) := \Sigma x{:}A.\Pi y{:}A.\texttt{Path}_A\,(x, y)$ holds. Then for every partial element a, there is a total element which restricts to a on α.

$$\frac{c : \mathsf{Contractible}(A) \qquad \alpha \vdash a : A}{\mathsf{contr\_extend\_partial}(\alpha.a) := \texttt{com}_A^{\_:0\to 1}\,(\alpha \mapsto \texttt{snd}(c)\; a)\,(\texttt{fst}(c)) : A[\alpha \mapsto a]}$$

Following Cohen, Coquand, Huber, and Mörtberg [2018], the notation $b : A[\alpha \mapsto a]$ in the conclusion abbreviates the two judgements $a : A$ and $\alpha \vdash b \equiv a : A$. Here, $\texttt{snd}(c)\; a$ has type $\texttt{Path}_A\,(\texttt{fst}(c), a)$, so composing the center of contraction $\texttt{fst}(c)$ with that path on α yields a total element of A that is a on α. In fact, the condition that "any partial element of A extends to a total element" is fibrant, a (-1)-type, and equivalent to $\mathsf{Contractible}(A)$, so it could serve as the *definition* of contractibility in a cubical type theory.

Voevodsky defines $f : T \to B$ to be an *equivalence* when $\Pi b{:}B.\mathsf{Contractible}(\mathsf{HFiber}(f, b))$, where $\mathtt{HFiber}(f, b) := \Sigma t{:}T.\mathtt{Path}_B\,(f(t), b)$ is the *homotopy fiber* of f at b. We therefore obtain the equiv operation of Cohen, Coquand, Huber, and Mörtberg [2018] as a corollary: if $f : A \to B$ is an equivalence, then for any $b : B$, any partial element of $\mathsf{HFiber}(f, b)$ extends to a total element.

Adjusting a composition structure by a partial composition structure The rules of cubical type theory stipulate the existence of a composition operation $\mathtt{com}_A^{z:r \to r'}\,(\alpha \mapsto z.t)\,(b)$ for each type A. However, for any given A, there are terms besides this $\mathtt{com}$ that satisfy the typing and equality rules of $\mathtt{com}$ (for instance, the right-hand side of the defining equations for composition in each type). We call any term satisfying these typing and equality rules a *composition structure* for A, and write com instead of $\mathtt{com}$ to (subtly) differentiate it from the stipulated one:

$$\frac{\begin{array}{l}\Psi \vdash r, r' : \mathbb{I} \\ \Psi, z : \mathbb{I}; \phi; \Gamma \vdash A\ \mathtt{Type} \\ \Psi, z : \mathbb{I}; \phi, \alpha; \Gamma \vdash t : A \\ \Psi; \phi; \Gamma \vdash b : A\langle r/z\rangle[\alpha \mapsto t\langle r/z\rangle]\end{array}}{\Psi; \phi; \Gamma \vdash \mathsf{com}_A^{z:r \to r'}\,(\alpha \mapsto z.t)\,(b) : A\langle r'/z\rangle[\alpha \mapsto t\langle r'/z\rangle, r = r' \mapsto b]}$$

Given any composition structure for a type family $z : \mathbb{I}.A$, and a second partial composition structure on some cofibration β, we can define a (total) composition structure that restricts to the latter on β:

$$\frac{\begin{array}{c}\mathsf{com}_A \text{ is a composition structure for } A \\ \beta \vdash \mathsf{pcom}_A \text{ is a composition structure for } A\end{array}}{\begin{array}{c}\mathsf{adjust\_com}_A \text{ is a (composition structure for } A)[\beta \mapsto \mathsf{pcom}_A] \\ \mathsf{adjust\_com}_A^{z:r \to r'}(\alpha \mapsto z.t)(b) := \mathsf{com}_A^{z:r \to r'}[\alpha \mapsto z.t, \beta \mapsto z.\mathsf{pcom}_A^{z:r \to z}(\alpha \mapsto z.t)(b)](b)\end{array}}$$

Homogeneous composition and coercion A *homogeneous composition (hcom) structure* on a type A is a term as follows, satisfying the indicated boundary conditions:

$$\frac{\begin{array}{l}\Psi \vdash r, r' : \mathbb{I} \\ \Psi; \phi; \Gamma \vdash A : \mathtt{Type} \\ \Psi, z : \mathbb{I}; \phi, \alpha; \Gamma \vdash t : A \\ \Psi; \phi; \Gamma \vdash b : A \\ \Psi; \phi, \alpha; \Gamma \vdash t\langle r/z\rangle \equiv b : A\end{array}}{\begin{array}{c}\Psi; \phi; \Gamma \vdash \mathsf{hcom}_A^{r \to r'}\,(\alpha \mapsto z.t)\,(b) : A \\ r = r' \vdash \mathsf{hcom}_A^{r \to r'}\,(\alpha \mapsto z.t)\,(b) \equiv b \\ \alpha \vdash \mathsf{hcom}_A^{r \to r'}\,(\alpha \mapsto z.t)\,(b) \equiv t\langle r'/z\rangle\end{array}}$$

An hcom structure is a restricted form of composition structure in which the type A is not allowed to depend on the filling direction.

A *coercion structure* on a type $z.A$ is a composition structure with no α boundary constraint:

$$\frac{\begin{array}{l}\Psi \vdash r, r' : \mathbb{I} \\ \Psi, z : \mathbb{I}; \phi; \Gamma \vdash A : \mathtt{Type} \\ \Psi; \phi; \Gamma \vdash b : A\langle r/z\rangle\end{array}}{\begin{array}{c}\Psi; \phi; \Gamma \vdash \mathsf{coe}_A^{z:r \to r'}(b) : A\langle r'/z\rangle \\ r = r' \vdash \mathsf{coe}_A^{z:r \to r'}(b) \equiv b\end{array}}$$

Every composition structure gives rise to hcom and coercion structures:

$$\texttt{hcom}_A^{r\to r'}\,(\alpha \mapsto z.t)\,(b) := \texttt{com}_A^{\_:r\to r'}\,(\alpha \mapsto z.t)\,(b)$$
$$\texttt{coe}_A^{z:r\to r'}(b) := \texttt{com}_A^{z:r\to r'}\,(0=1 \mapsto z.\texttt{abort})\,(b)$$

Conversely, an hcom structure and a coercion structure give rise to a composition structure:

$$\mathsf{com}_A^{z:r\to r'}\,(\alpha \mapsto z.t)\,(b) := \mathsf{hcom}_{A\langle r'/z\rangle}^{r\to r'}\,((\alpha \mapsto z.\mathsf{coe}_A^{z:z\to r'}(t)))\,(\mathsf{coe}_A^{z:r\to r'}(b))$$

This decomposition was first used by Coquand [2015] when defining composition for higher inductive types. Various type theories have adopted this decomposition in *every* type, including Angiuli et al. [2017b] and an implementation of the connections model with regularity [Coquand, 2014a]; recent work on higher inductives in the connections model [Coquand et al., 2018] uses a similar decomposition with a generalized notion of coercion. In this paper, we use this decomposition in only certain types, such as higher inductive types.

Weak coercion A *weak coercion structure* on $z.A$ is similar to a coercion structure, except that on $r = r'$ it is only an identity function up to a path β_b:

$$\frac{\begin{array}{l}\Psi \vdash r, r' : \mathbb{I}\\ \Psi, z : \mathbb{I}; \phi; \Gamma \vdash A : \texttt{Type}\\ \Psi; \phi; \Gamma \vdash b : A\langle r/z\rangle\end{array}}{\Psi; \phi; \Gamma \vdash \mathsf{wcoe}_A^{z:r\to r'}\,(b) : A\langle r'/z\rangle}$$

$$\begin{array}{l}x; r = r' \vdash \beta_b : A\langle r'/z\rangle\\ r = r' \vdash \beta_b\langle 0/x\rangle \equiv \mathsf{wcoe}_A^{z:r\to r'}\,(b)\\ r = r' \vdash \beta_b\langle 1/x\rangle \equiv b\end{array}$$

Importantly, diagonal cofibrations in homogeneous composition allow us to improve weak coercion structures to strict ones, a maneuver we will use in higher inductive types:

$$\mathsf{coe}_A^{w:s\to s'}(b) := \mathsf{hcom}_{A\langle s'/w\rangle}^{0\to 1}\,(s = s' \mapsto x.\beta_b)\,(\mathsf{wcoe}_A^{w:s\to s'}\,(b))$$

A composition structure on a type therefore follows from a weak coercion structure and an hcom structure for each fiber.

Strictly preserving homogeneous compositions Given $x : A \vdash f(x) : B(x)$, an hcom structure on A, and a composition structure on B, f *strictly preserves hcoms* if

$$\begin{array}{ll} & f[(\mathsf{hcom}_A^{r\to r'}\,(\alpha \mapsto z.t)\,(b))/x]\\ \equiv & \mathsf{com}_{B[\mathsf{hcom}_A^{r\to z}\,(\alpha\mapsto z.t)\,(b)/x]}^{z:r\to r'}\,(\alpha \mapsto z.f[t/x])\,(f[b/x]) : C[(\mathsf{hcom}_A^{r\to r'}\,(\alpha \mapsto z.t)\,(b))/x]\end{array}$$

There is always a path between these equands; in some types (HITs) it is a definitional equality.

2.8 Σ-types (as a negative type operator)

The introduction, elimination, β, and η rules are the usual ones, but apply at any dimension; we therefore have β and η not only for pairing of points, but for pairing of paths, squares, etc.

$$\frac{\Psi;\phi;\Gamma\vdash A\ \mathtt{Type}\qquad \Psi;\phi;\Gamma,x:A\vdash B\ \mathtt{Type}}{\Psi;\phi;\Gamma\vdash \Sigma x{:}A.B\ \mathtt{Type}}\qquad \frac{\Psi;\phi;\Gamma\vdash u:A\qquad \Psi;\phi;\Gamma\vdash v:B[u/x]}{\Psi;\phi;\Gamma\vdash (u,v):\Sigma x{:}A.B}$$

$$\frac{\Psi;\phi;\Gamma\vdash u:\Sigma x{:}A.B}{\Psi;\phi;\Gamma\vdash \mathtt{fst}(u):A}\qquad \frac{\Psi;\phi;\Gamma\vdash u:\Sigma x{:}A.B}{\Psi;\phi;\Gamma\vdash \mathtt{snd}(u):B[\mathtt{fst}(u)/x]}$$

$$\begin{array}{rcl}\mathtt{fst}(u,v) & \equiv & u\\ \mathtt{snd}(u,v) & \equiv & v\\ u & \equiv & (\mathtt{fst}(u),\mathtt{snd}(u))\end{array}$$

Kan operation

$$\begin{array}{rcl}\mathtt{com}^{z:r\to r'}_{\Sigma x:A.B}\,(\alpha\mapsto z.t)\,(b) & \equiv & (\mathtt{com}^{z:r\to r'}_{A}\,(\alpha\mapsto z.\mathtt{fst}(t))\,(\mathtt{fst}(b)),\\ & & \mathtt{com}^{z:r\to r'}_{B[\mathtt{fill}^{z:r\to z}_{A}\,(\alpha\mapsto z.\mathtt{fst}(t))\,(\mathtt{fst}(b))/x]}\,(\alpha\mapsto z.\mathtt{snd}(t))\,(\mathtt{snd}(b)))\end{array}$$

The computation rule for composition generalizes the usual rule for `transport` at Σ-types: push into both components, the second over the first. In the second component, we substitute $\mathtt{fill}^{z:r\to z}_{A}\,(\alpha\mapsto z.\mathtt{fst}(t))\,(\mathtt{fst}(b))$ into B, which makes the second component type-check: when r is substituted for z, the filler reduces to $\mathtt{fst}(b)$, which appears in the type of $\mathtt{snd}(b)$; on α, the filler reduces to $\mathtt{fst}(t)$, which occurs in the type of $\mathtt{snd}(t)$; and when r' is substituted for z, the filler is the first component of the pair.

2.9 Π-types

$$\frac{\Psi;\phi;\Gamma\vdash A\ \mathtt{Type}\qquad \Psi;\phi;\Gamma,x:A\vdash B\ \mathtt{Type}}{\Psi;\phi;\Gamma\vdash \Pi x{:}A.B\ \mathtt{Type}}\qquad \frac{\Psi;\phi;\Gamma\vdash f:\Pi x{:}A.B\qquad \Psi;\phi;\Gamma\vdash a:A}{\Psi;\phi;\Gamma\vdash f\,a:B[a/x]}$$

$$\frac{\Psi;\phi;\Gamma,x:A\vdash u:B}{\Psi;\phi;\Gamma\vdash \lambda x.u:\Pi x{:}A.B}$$

$$\begin{array}{rcl}(\lambda x.u)\,a & \equiv & u[a/x]\\ f & \equiv & \lambda x.f\,x\end{array}$$

Kan operation

$$\mathtt{com}^{z:r\to r'}_{\Pi x:A.B}\,(\alpha\mapsto z.t)\,(b)\ \equiv\ \lambda a.\mathtt{com}^{z:r\to r'}_{B[\mathtt{fill}^{z:r'\to z}_{A}\,(a)/x]}\,(\alpha\mapsto z.t\,(\mathtt{fill}^{z:r'\to z}_{A}\,(a)))\,(b\,(\mathtt{coe}^{z:r'\to r}_{A}(a)))$$

The composition on the right-hand side is well-typed because $B[\mathtt{fill}^{z:r'\to z}_{A}\,(a)/x]$ agrees with $B[\mathtt{coe}^{z:r'\to r}_{A}(a)/x]$ under $\langle r/z\rangle$ (so that $b\,(\mathtt{coe}^{z:r'\to r}_{A}(a))$ has the correct type), and agrees with $B[a/x]$ under $\langle r'/z\rangle$ (so that the result type is correct).

In Section 1, we motivated Kan composition *to* an arbitrary r' as a natural way to close Kan filling under substitution. Our definition of `com` from r to r' for Π-types requires the *reverse* coercions and fillers, from r' to r. Therefore, for Π-types to be closed under our Kan operation—at least for the present definition of composition in Π-types—allowing the target of a composition problem to be an arbitrary r' requires us to allow the *source* to also be an arbitrary r'.

2.10 Path types

Paths are boundary-constrained maps out of the interval $\mathbb{I}$. The elimination rules for path types state that an element u can be turned back into a cube in A by instantiating it with a dimension r. When r is a dimension variable that does not occur in u, this operation simply chooses a name for the "hidden" dimension of the path type element. When r is a dimension variable that does occur, this operation takes a diagonal. When r is 0 or 1, $u\,r$ is equal to the element specified by u's type, ensuring that u connects the specified points. The introduction rule inverts the elimination rules: to give an element of the path type, one must give a higher cube with the correct boundary.

$$\frac{\Psi, x : \mathbb{I}; \phi; \Gamma \vdash A\ \texttt{Type} \quad \Psi; \phi; \Gamma \vdash a_0 : A\langle 0/x\rangle \quad \Psi; \phi; \Gamma \vdash a_1 : A\langle 1/x\rangle}{\Psi; \phi; \Gamma \vdash \texttt{Path}_{x.A}\,(a_0, a_1)\ \texttt{Type}}$$

$$\frac{\Psi, x : \mathbb{I}; \phi; \Gamma \vdash u : A \quad \Psi; \phi; \Gamma \vdash u\langle 0/x\rangle \equiv a_0 : A \quad \Psi; \phi; \Gamma \vdash u\langle 1/x\rangle \equiv a_1 : A}{\Psi; \phi; \Gamma \vdash \Lambda x.u : \texttt{Path}_{x.A}\,(a_0, a_1)}$$

$$\frac{\Psi; \phi; \Gamma \vdash u : \texttt{Path}_{x.A}\,(a_0, a_1) \quad \Psi \vdash r : \mathbb{I}}{\Psi; \phi; \Gamma \vdash u\,r : A\langle r/x\rangle}$$

$$\begin{array}{rcl} u\,0 & \equiv & a_0 \\ u\,1 & \equiv & a_1 \\ (\Lambda x.u)\,r & \equiv & u\langle r/x\rangle \\ u & \equiv & \Lambda s.(u\,s) \end{array}$$

Kan operation

$$\begin{array}{rl} & \texttt{com}^{z:r\to r'}_{\texttt{Path}_{x.A}\,(x=r_i\mapsto a)}\,(\alpha \mapsto z.t)\,(b) \\ \equiv & \Lambda x.\texttt{com}^{z:r\to r'}_{A}\,[\alpha \mapsto z.t\,x, (x=0)\mapsto \_.a_0, (x=1)\mapsto \_.a_1]\,(b\,x) \end{array}$$

The Kan operation extends the filling problem with the faces indicated by the type, which is necessary for the right-hand side to have the right faces when $x = 0$ or $x = 1$. Recall that the commas here are notation for the $\vee$ of cofibrations: the cofibration used on the right-hand side of the equation is $\alpha \vee ((x = 0) \vee (x = 1))$, with the $\vee$ left rule used to construct the term $[\alpha \mapsto z.t\,x, [(x = 0) \mapsto \_.a_0, (x = 1) \mapsto \_.a_1]]$. The compatibility condition is satisfied because the terms a_0 and a_1 agree with $t\,x$ when x is 0 or 1 by t's path type.

2.11 Glue types

The glue type $\texttt{Glue}\,(\alpha \mapsto (T, f))\,(B)$, introduced in Coquand [2014a]; Cohen, Coquand, Huber, and Mörtberg [2018], is a type which is equal to the partial type T on α, given a function $\alpha \vdash f : T \to B$ mapping T into the total type B. The formation, introduction, elimination, β, and η rules can be defined for `Glue` types for any function f—in the semantics, we do not insist that f is an equivalence at this stage—but Glue types are only Kan when f is an equivalence. (In the formalization we show that glue types have a weaker *homogeneous composition* structure for any function f.)

Because all syntactic types in our system are Kan, we restrict the formation rule of Glue types to equivalences $f : A \simeq B$, which we define here to be functions $f : A \to B$ whose hfibers are contractible, as described in Section 2.7. (We will implicitly coerce equivalences to functions $A \to B$ when necessary.) In Section 3, we will define glue types for all functions, but the universe of Kan types only includes codes for glue types on equivalences. Angiuli et al. [2017b] do consider non-Kan pre-types, but not Glue with non-equivalences.

The rules below essentially state that $\texttt{Glue}\,(\alpha \mapsto (T, f))\,(B)$ is a Σ type of $b : B$ and $\alpha \vdash t : T$ such that $f(t) \equiv b$: the elimination rule, for example, is a projection to B. However, it is stricter than such a Σ type because under α, the Glue type is judgementally equal to T, its elements restrict to t, and the projection to B restricts to f.

$$\frac{\Psi;\phi;\Gamma \vdash B\ \texttt{Type} \quad \Psi;\phi,\alpha;\Gamma \vdash T\ \texttt{Type} \quad \Psi;\phi,\alpha;\Gamma \vdash f : T \simeq B}{\begin{array}{c}\Psi;\phi;\Gamma \vdash \texttt{Glue}\,(\alpha \mapsto (T, f))\,(B)\ \texttt{Type} \\ \alpha \vdash \texttt{Glue}\,(\alpha \mapsto (T, f))\,(B) \equiv T\end{array}}$$

$$\frac{\begin{array}{l}\Psi;\phi;\Gamma \vdash b : B \\ \Psi;\phi,\alpha;\Gamma \vdash t : T \\ \Psi;\phi,\alpha;\Gamma \vdash f(t) \equiv b : B\end{array}}{\begin{array}{c}\Psi;\phi;\Gamma \vdash \texttt{glue}\,(\alpha \mapsto t)\,(b) : \texttt{Glue}\,(\alpha \mapsto (T, f))\,(B) \\ \alpha \vdash \texttt{glue}\,(\alpha \mapsto t)\,(b) \equiv t\end{array}}$$

$$\frac{\Psi;\phi;\Gamma \vdash g : \texttt{Glue}\,(\alpha \mapsto (T, f))\,(B)}{\begin{array}{c}\Psi;\phi;\Gamma \vdash \texttt{unglue}(g) : B \\ \alpha \vdash \texttt{unglue}(g) \equiv f(g)\end{array}}$$

$$\begin{array}{rcl}\texttt{unglue}(\texttt{glue}\,(\alpha \mapsto t)\,(b)) & \equiv & b \\ g & \equiv & \texttt{glue}\,(\alpha \mapsto g)\,(\texttt{unglue}(g))\end{array}$$

The Kan operation for glue types is quite complicated, and we obtain it in multiple steps.

Weak glue introduction The $\texttt{glue}\,(\alpha \mapsto t)\,(b)$ term pairs together a term $b : B$ and $\alpha \vdash t : T$ such that $\alpha \vdash f(t) \equiv b$. If we weaken this judgemental equality to a path, then t paired with such a path is an element of the homotopy fiber of f at b, $\alpha \vdash h : \texttt{HFiber}(f, b)$. Using homogeneous composition in B, we can derive the following weakened version of the glue introduction rule, in which the constructor takes an element of $\texttt{HFiber}(f, b)$ as input:

$$\frac{b : B \quad \alpha \vdash h : \texttt{HFiber}(f, b)}{\mathsf{wglue}\,(\alpha \mapsto h)\,(b) : \texttt{Glue}\,(\alpha \mapsto (T, f))\,(B) := \texttt{glue}\,(\alpha \mapsto \texttt{fst}(h))\,(\texttt{com}_{\overline{B}}^{:1\to 0}\,(\alpha \mapsto x.\texttt{snd}(h)\,x)\,(b))}$$

We adjust b by $\texttt{snd}(h) : \texttt{Path}_B\,(f(\texttt{fst}(h)), b)$, making an element of B that is $f(\texttt{fst}(h))$ on α.

We will in fact need a slightly better version of this operation. First, if $g : \texttt{Glue}\,(\alpha \mapsto (T, f))\,(B)$, where $f : T \to B$, then g itself is in the fiber of f over $\texttt{unglue}(g)$:

$$\alpha \vdash \mathsf{glue\_to\_fiber}(g) : \texttt{HFiber}(f, \texttt{unglue}(g)) := (g, \Lambda\_.\texttt{unglue}(g))$$

In the above, $g : T$ because $\alpha \vdash \texttt{Glue}\,(\alpha \mapsto (T, f))\,(B) \equiv T$, and the path is well-typed by $\alpha \vdash \texttt{unglue}(g : \texttt{Glue}\,(\alpha \mapsto (T, f))\,(B)) \equiv f(g : T) : B$.

Now, suppose we have a cofibration β (possibly different from α) and a partial element of the glue type $\beta \vdash g : \texttt{Glue}\,(\alpha \mapsto (T, f))\,(B)$. Then, as long as g agrees with b and t on β, we can proceed as above and create an element of the glue type, but now one that restricts to g on β:

$$\frac{\beta \vdash g : \texttt{Glue}\,(\alpha \mapsto (T,f))\,(B) \quad b : B[\beta \mapsto \texttt{unglue}(g)] \quad \alpha \vdash h : \texttt{HFiber}(f,b)[\beta \mapsto \mathsf{glue\_to\_fiber}(g)]}{\mathsf{wglue}\,(\alpha \mapsto h)\,(b)(\beta \mapsto g) : \texttt{Glue}\,(\alpha \mapsto (T,f))\,(B)[\beta \mapsto g]}$$

$$\mathsf{wglue}\,(\alpha \mapsto h)\,(b)(\beta \mapsto g) := \texttt{glue}\,(\alpha \mapsto \texttt{fst}(h))\,(\texttt{com}_{\overline{B}}^{:1\to 0}\,(\alpha \mapsto x.\texttt{snd}(h)\,x, \beta \mapsto \_.\texttt{unglue}(g))\,(b))$$

Incoherent composition for glue types First, we define a composition structure for glue types which is "incoherent" in the sense that it may not restrict to the composition operation for T on α, that is:

$$\frac{z : \mathbb{I} \vdash G := \texttt{Glue}\,(\alpha \mapsto (T,f))\,(B)\;\texttt{Type} \quad s : \mathbb{I} \quad s' : \mathbb{I} \quad z : \mathbb{I}, \beta \vdash u : G \quad v : G\langle s/z\rangle[\beta \mapsto u\langle s/z\rangle]}{\mathsf{icom}_G^{z:s\to s'}(\beta \mapsto z.u)v : G\langle s'/z\rangle[\beta \mapsto u\langle s'/z\rangle, s = s' \mapsto v]}$$

To begin, we project the glue types to the base B by $\texttt{unglue}(-)$ and fill and compose there:

$$\begin{array}{lll} w : \mathbb{I} \vdash & b_{fill} := \texttt{fill}_B^{z:s\to w}\,(\beta \mapsto z.\texttt{unglue}(u))\,(\texttt{unglue}(v)) & : B\langle w/z\rangle \\ & b' := b_{fill}\langle s'/w\rangle & : B\langle s'/z\rangle \end{array}$$

Next, we define a term under the restriction $\alpha\langle s'/z\rangle$, the cofibration under which the conclusion glue type $G\langle s'/z\rangle$ restricts to $T\langle s'/z\rangle$. The data for the glue type includes the witness f_{eq} that

$$\alpha\langle s'/z\rangle \vdash f\langle s'/z\rangle : T\langle s'/z\rangle \to B\langle s'/z\rangle$$

is an equivalence (on $\alpha\langle s'/z\rangle$) , which implies in particular that $\mathsf{Contractible}(\texttt{HFiber}(f\langle s'/z\rangle, b'))$. Thus, using $\mathsf{contr\_extend\_partial}$, any partial element of this hfiber extends to a total one. The partial element we choose is on the cofibration $\beta \vee (s = s')$,

$$[\beta \mapsto \mathsf{glue\_to\_fiber}(u\langle s'/z\rangle), s = s' \mapsto \mathsf{glue\_to\_fiber}(v)]$$

This is one of the key places we use our stipulation that equality on $\mathbb{I}$ is a cofibration. Thus,

$$\begin{array}{l} \alpha\langle s'/z\rangle \vdash c : \texttt{HFiber}(f\langle s'/z\rangle, b')[\beta \mapsto \mathsf{glue\_to\_fiber}(u\langle s'/z\rangle), s = s' \mapsto \mathsf{glue\_to\_fiber}(v)] \\ c := \mathsf{contr\_extend\_partial}(f_{eq}(b'))[\beta \mapsto \mathsf{glue\_to\_fiber}(u\langle s'/z\rangle), s = s' \mapsto \mathsf{glue\_to\_fiber}(v)] \end{array}$$

Finally, we use $\mathsf{glue\_weak\_better}$ to construct an element of $G\langle s'/z\rangle = \texttt{Glue}\,(\alpha\langle s'/z\rangle \mapsto (T\langle s'/z\rangle, f\langle s'/z\rangle))\,(B\langle s'/z\rangle)$. This requires an element of $B\langle s'/z\rangle$, for which we use b', and an element of $\alpha\langle s'/z\rangle \vdash \texttt{HFiber}(f\langle s'/z\rangle, b')$, for which we use c. The cofibration on which we "remember" an existing element of $G\langle s'/z\rangle$ is $\beta \vee (s = s')$, and the existing element is $[\beta \mapsto u\langle s'/z\rangle, s = s' \mapsto v]$. We need to know that on $\beta \vee (s = s')$, b' is equal to $\texttt{unglue}([\beta \mapsto u\langle s'/z\rangle, s = s' \mapsto v])$, which holds by the boundary conditions for b', and that c is equal to $\mathsf{glue\_to\_fiber}([\beta \mapsto \mathsf{u}\langle \mathsf{s'/z}\rangle, \mathsf{s} = \mathsf{s'} \mapsto \mathsf{v}])$, which is the partial element we extended to make c. Thus, overall, we have

$$\mathsf{icom}_G^{z:s\to s'}(\beta \mapsto z.u)v := \mathsf{wglue}\,(\alpha\langle s'/z\rangle \mapsto c)\,(b')[\beta \mapsto u\langle s'/z\rangle, s = s' \mapsto v]$$

Aligning Christian Sattler and Ian Orton (independently) analyzed the CCHM algorithm for composition for glue types [Cohen, Coquand, Huber, and Mörtberg, 2018] and realized that the use of the $\forall$ cofibration could be isolated in a single "aligning" step, which fixes an incoherent composition operation for glue types so that it restricts appropriately to composition in T on α. We use the same method here.

Let $z : \mathbb{I} \vdash G := \mathtt{Glue}\,(\alpha \mapsto (T, f))\,(B)$ $\mathtt{Type}$ as above. Notice that $\forall z : \mathbb{I}.\alpha \vdash G \equiv T$, by $\forall z : \mathbb{I}.\alpha \vdash_{z:\mathbb{I}} \alpha$. Thus, on $\forall z : \mathbb{I}.\alpha$, $\mathtt{com}_T^{z:s\to s'}\,(\beta \mapsto z.u)\,(v)$ is a composition structure for G, and therefore we can adjust the incoherent composition structure from the previous section as follows:

$$\mathtt{com}^{z:s\to s'}_{\mathtt{Glue}\,[\alpha\mapsto(T,f)]\,(B)}\,(\beta \mapsto z.t)\,(b) := \mathsf{adjust\_com}_{z.G}(\mathsf{icom})(\forall z.\alpha \mapsto \mathtt{com}_{z.T})$$

so that it satisfies the following equation, forced by the equation $\alpha \vdash \mathtt{Glue}\,(\alpha \mapsto (T, f))\,(B) \equiv T$:

$$\forall z.\alpha \vdash \mathtt{com}^{z:s\to s'}_{\mathtt{Glue}\,[\alpha\mapsto(T,f)]\,(B)}\,(\beta \mapsto t)\,(b) \equiv \mathtt{com}_T^{z:s\to s'}\,(\beta \mapsto t)\,(b)$$

Comparison with CCHM The CCHM algorithm inlines the glue_weak_better lemma, the adjust_com lemma, and the aligning step; otherwise, the main difference is that here we use the diagonal cofibration $s = s'$ in contr_extend_partial to "remember" that the result should be v on $s = s'$.

2.12 Universe

Depending on technical details of the intended semantics, we could define universes à la Tarski or à la Russell; here, we elide the $El(-)$ for notational simplicity. Universes are specified by the usual rules reflecting types $\Psi; \phi; \Gamma \vdash A\ \mathtt{Type}_i$ as elements $\Psi; \phi; \Gamma \vdash A : U_i$, with $\Psi; \phi; \Gamma \vdash U_i : U_{i+1}$.

Suppose $\mathtt{U}$ is a universe of Kan types closed under glue types; for $\mathtt{U}$ to itself be a Kan type, we need a (homogeneous) composition structure on it, which is to say that whenever we have $B : \mathtt{U}$ and $z, \alpha \vdash T : \mathtt{U}$ with $\alpha \vdash T\langle r/z\rangle \equiv B$, we need a Kan type:

$$\begin{gathered}\mathtt{com}_{\mathtt{U}}^{z:r\to r'}\,(\alpha \mapsto z.T)\,(B)\\ \alpha \vdash \mathtt{com}_{\mathtt{U}}^{z:r\to r'}\,(\alpha \mapsto z.T)\,(B) \equiv T\langle r'/z\rangle\\ r = r' \vdash \mathtt{com}_{\mathtt{U}}^{z:r\to r'}\,(\alpha \mapsto z.T)\,(B) \equiv B\end{gathered}$$

Here, following CCHM, we construct such a type by converting lines in the universe to equivalences and using the fact that $\mathtt{Glue}$ types are Kan for equivalences. (One could instead define a type similar to glue whose input data is lines in the universe rather than equivalences, which permits some different definitions of the Kan operations [Angiuli et al., 2017b].) We define:

$$\mathtt{com}_{\mathtt{U}}^{z:r\to r'}\,(\alpha \mapsto z.T)\,(B) \equiv \mathtt{Glue}\,[\alpha \mapsto (T\langle r'/z\rangle, (\mathtt{coe}_T^{z:r'\to r}(-), e)), r = r' \mapsto (B, (\lambda x.x, e'))]\,(B)$$

where

$$e : \mathtt{isEquiv}(\mathtt{coe}_T^{z:r'\to r}(-)) \qquad e' : \mathtt{isEquiv}(\lambda x.x) \qquad \alpha, r = r' \vdash e \equiv e'$$

e' is defined in the usual way (using contractibility of singletons, which follows from composition and filling), and e is obtained by transporting e':

$$e := \mathtt{coe}^{z:r'\to r}_{\mathtt{isEquiv}(\mathtt{coe}_T^{z:r'\to z}(-))}(e')$$

so that it automatically agrees with e' on $r = r'$.

This type satisfies the desired boundary equations

$$\begin{aligned} r = r' \vdash \mathtt{Glue}\,[\alpha \mapsto (T\langle r'/z\rangle, \mathtt{coe}_T^{z:r'\to r}(-)), r = r' \mapsto (B, \lambda x.x)]\,(B) \quad &\equiv \quad B \\ \alpha \vdash \mathtt{Glue}\,[\alpha \mapsto (T\langle r'/z\rangle, \mathtt{coe}_T^{z:r'\to r}(-)), r = r' \mapsto (B, \lambda x.x)]\,(B) \quad &\equiv \quad T\langle r'/z\rangle \end{aligned}$$

Following Cohen, Coquand, Huber, and Mörtberg [2018], we can then prove the univalence axiom for a fibrant universe containing glue types. For example, an equivalence $f : A \simeq B$ is converted to a path in the universe as follows:

$$\mathtt{ua}(e) := \Lambda x.\mathtt{Glue}\,(x = 0 \mapsto (A, f), x = 1 \mapsto (B, (\lambda x.x, e)))\,(B) : \mathtt{Path}_{\mathtt{U}}\,(A, B)$$

Writing G for this glue type, this path has the correct endpoints because $x = 0 \vdash G \equiv A$ and $x = 1 \vdash G \equiv B$.

Both the universe and univalence use the fact that on α, $\mathtt{Glue}\,(\alpha \mapsto (T, f))\,(B)$ restricts to T. This equation between types forces the equation between Kan operations that requires the final aligning step in the construction of the Kan operation for glue types.

To obtain the full univalence axiom, an internal argument shows that it suffices to give a "β-reduction" $\mathtt{Path}_{A\to B}\,(\mathtt{com}_{\mathtt{ua}(e)\,z}^{z:0\to 1}\,[]\,(-), f)$ that reduces transport along univalence to the provided function. This is because such a $\mathtt{ua}$ with a β path can be "fixed" to satisfy the rest of the univalence axiom, using ideas from Egbert Rijke and Martín Escardó. In our formalization, we construct such a β path for the above definition of composition for glue types.

2.13 Strict booleans

The introduction and elimination rules for natural numbers are the usual ones:

$$\frac{}{\Psi; \phi; \Gamma \vdash \mathtt{bool}\ \mathtt{Type}} \qquad \frac{}{\Psi; \phi; \Gamma \vdash \mathtt{true} : \mathtt{bool}} \qquad \frac{}{\Psi; \phi; \Gamma \vdash \mathtt{false} : \mathtt{bool}}$$

$$\frac{\Psi; \phi; \Gamma, x : \mathtt{bool} \vdash C\ \mathtt{Type} \quad \Psi; \phi; \Gamma \vdash u : \mathtt{bool} \quad \Psi; \phi; \Gamma \vdash v_1 : C[\mathtt{true}/x] \quad \Psi; \phi; \Gamma \vdash v_2 : C[\mathtt{false}/x]}{\Psi; \phi; \Gamma \vdash \mathtt{if}_{x.C}(u, v_1, v_2) : C[u/x]}$$

$$\mathtt{if}_{x.C}(\mathtt{true}, v_1, v_2) \equiv v_1 \qquad \mathtt{if}_{x.C}(\mathtt{false}, v_1, v_2) \equiv v_2$$

Our semantics in cubical sets (in particular, the connectedness of the interval) justifies "equality reflection" for booleans:

$$\frac{\Psi, x : \mathbb{I}; \phi; \Gamma \vdash u : \mathtt{bool} \quad \Psi \vdash r, r' : \mathbb{I}}{\Psi; \phi; \Gamma \vdash t\langle r/x\rangle \equiv t\langle r'/x\rangle : \mathtt{bool}}$$

In particular, given any $\Lambda x.b : \mathtt{Path}_{\mathtt{bool}}\,(b_0, b_1)$, there are judgemental equalities $b_0 \equiv b_1$ (by taking r, r' to be $0, 1$) and $\Lambda x.b \equiv \Lambda\_.b_0$ (by taking r, r' to be $0, x$). In the presence of this rule, it type- and boundary-checks to define

$$\mathtt{com}_{\mathtt{bool}}^{\_:r\to r'}\,(\alpha \mapsto t)\,(b) \equiv b$$

The same definition works for any closed base type with decidable equality.

If one desires type checking to be algorithmic, it is also possible to only apply this equation when both t and b are equal to $\mathtt{true}$ or both are equal to $\mathtt{false}$, following CCHM.

Finally, one can also treat the booleans as a higher inductive type, in which the values include not only $\mathtt{true}$ and $\mathtt{false}$ but also homogeneous compositions. In this paper, such booleans would contain non-standard points—homogeneous compositions with a false face formula—in the empty context. Angiuli et al. [2017b] use a different definition of composition that disallows false cofibrations; there, the homotopy booleans have only `true` and `false` as points in the empty context.

2.14 Strict natural numbers

Again, the introduction and elimination rules are as usual:

$$\frac{}{\Psi;\phi;\Gamma \vdash \mathtt{nat}\ \mathtt{Type}} \qquad \frac{}{\Psi;\phi;\Gamma \vdash \mathtt{zero} : \mathtt{nat}} \qquad \frac{\Psi;\phi;\Gamma \vdash u : \mathtt{nat}}{\Psi;\phi;\Gamma \vdash \mathtt{succ}(u) : \mathtt{nat}}$$

$$\frac{\begin{array}{l}\Psi;\phi;\Gamma, x : \mathtt{nat} \vdash C\ \mathtt{Type}\\ \Psi;\phi;\Gamma \vdash u : \mathtt{nat}\\ \Psi;\phi;\Gamma \vdash v_0 : C[\mathtt{zero}/x]\\ \Psi;\phi;\Gamma, n : \mathtt{nat}, r : C[n/x] \vdash v_1 : C[\mathtt{succ}(n)/x]\end{array}}{\Psi;\phi;\Gamma \vdash \mathbb{N}{-}\mathtt{elim}_{x.C}(u, v_0, x.r.v_1) : C[u/x]} \qquad \mathbb{N}{-}\mathtt{elim}_{x.C}(\mathtt{zero}, v_0, v_1) \equiv v_1$$

$$\mathbb{N}{-}\mathtt{elim}_{x.C}(\mathtt{succ}(n), v_0, x.r.v_1) \equiv v_1\langle n/x\rangle\langle \mathbb{N}{-}\mathtt{elim}_{x.C}(n, v_0, x.r.v_1)/r\rangle$$

Analogously to booleans, one possibility is to have equality reflection:

$$\frac{\Psi, x : \mathbb{I};\phi;\Gamma \vdash u : \mathtt{nat} \quad \Psi \vdash r, r' : \mathbb{I}}{\Psi;\phi;\Gamma \vdash t\langle r/x\rangle \equiv t\langle r'/x\rangle : \mathtt{bool}} \qquad \mathtt{com}^{:r \rightarrow r'}_{\mathtt{nat}}\ (\alpha \mapsto t)\ (b) \equiv b$$

2.15 $\mathbf{\Sigma} A$

We consider suspensions to illustrate higher inductive types; suspensions and booleans suffice to construct all spheres. The introduction forms are the usual HIT constructors, plus a homogeneous composition structure:

$$\frac{\Psi;\phi;\Gamma \vdash A\ \mathtt{Type}}{\Psi;\phi;\Gamma \vdash \mathbf{\Sigma} A\ \mathtt{Type}} \qquad \frac{}{\Psi;\phi;\Gamma \vdash \mathtt{north} : \mathbf{\Sigma} A} \qquad \frac{}{\Psi;\phi;\Gamma \vdash \mathtt{south} : \mathbf{\Sigma} A}$$

$$\frac{\Psi \vdash r : \mathbb{I} \quad \Psi;\phi;\Gamma \vdash u : A}{\Psi;\phi;\Gamma \vdash \mathtt{merid}_r(u) : \mathbf{\Sigma} A} \qquad \mathtt{merid}_0(u) \equiv \mathtt{north} \qquad \mathtt{merid}_1(u) \equiv \mathtt{south}$$

$$\frac{\begin{array}{l}\Psi \vdash r, r' : \mathbb{I}\\ \Psi;\phi;\Gamma \vdash \mathbf{\Sigma} A : \mathtt{Type}\\ \Psi, z : \mathbb{I};\phi,\alpha;\Gamma \vdash t : \mathbf{\Sigma} A\\ \Psi;\phi;\Gamma \vdash b : \mathbf{\Sigma} A\\ \Psi;\phi,\alpha;\Gamma \vdash t\langle r/z\rangle \equiv b : \mathbf{\Sigma} A\end{array}}{\begin{array}{c}\Psi;\phi;\Gamma \vdash \mathtt{vhcom}^{r \rightarrow r'}_{\mathbf{\Sigma} A}\ (\alpha \mapsto t)\ (b) : \mathbf{\Sigma} A\\ r = r' \vdash \mathtt{vhcom}^{r \rightarrow r'}_{\mathbf{\Sigma} A}\ (\alpha \mapsto z.t)\ (b) \equiv b\\ \alpha \vdash \mathtt{vhcom}^{r \rightarrow r'}_{\mathbf{\Sigma} A}\ (\alpha \mapsto z.t)\ (b) \equiv t\langle r'/z\rangle\end{array}}$$

The eliminator computes on all of these:

$$\frac{\begin{array}{l}\Psi;\phi;\Gamma, a : \mathbf{\Sigma}A \vdash C\ \mathtt{Type} \\ \Psi;\phi;\Gamma \vdash u_0 : C[\mathtt{north}/a] \\ \Psi;\phi;\Gamma \vdash u_1 : C[\mathtt{south}/a] \\ \Psi, z : \mathbb{I};\Phi;\Gamma, b : A \vdash u_2 : C[\mathtt{merid}_z(b)/a] \\ \Psi;\Phi;\Gamma, b : A \vdash u_2\langle 0/z\rangle \equiv u_0 : C[\mathtt{north}/a] \\ \Psi;\Phi;\Gamma, b : A \vdash u_2\langle 1/z\rangle \equiv u_1 : C[\mathtt{south}/a]\end{array}}{\Psi;\phi;\Gamma \vdash \mathbf{\Sigma}^{a.C}_{\mathrm{elim}}(u_0; u_1; z.b.u_2; u) : A[u/x]}$$

$$\begin{array}{rcl}\mathbf{\Sigma}^{a.C}_{\mathrm{elim}}(u_0; u_1; z.b.u_2; \mathtt{north}) & \equiv & u_0 \\ \mathbf{\Sigma}^{a.C}_{\mathrm{elim}}(u_0; u_1; z.b.u_2; \mathtt{south}) & \equiv & u_1 \\ \mathbf{\Sigma}^{a.C}_{\mathrm{elim}}(u_0; u_1; z.b.u_2; \mathtt{merid}_r(u')) & \equiv & u\langle r/z\rangle[u'/b] \\ \mathbf{\Sigma}^{a.C}_{\mathrm{elim}}(u_0; u_1; z.b.u_2; \mathtt{vhcom}^{r\rightarrow r'}_{\mathbf{\Sigma}A}\,(\alpha \mapsto t)\,(B)) & \equiv & \mathtt{com}^{z:r\rightarrow r'}_{C[\mathtt{vhcom}^{r\rightarrow z}_{\mathbf{\Sigma}A}\,(\alpha\mapsto z.t)\,(b)/x]}\,(\alpha \mapsto z.E(t))\,(E(b))\end{array}$$

In the final equation, we write $E(x)$ for $\mathbf{\Sigma}^{a.C}_{\mathrm{elim}}(u_0; u_1; z.b.u_2; x)$; the equation says that E strictly preserves hcoms, for $\mathtt{vhcom}^{r\rightarrow r'}_{\mathbf{\Sigma}A}\,(\alpha \mapsto t)\,(B)$ in the domain and the canonical composition structure on C in the range.

In the semantics, for each type A, we first make a cubical set satisfying the above rules, where the elimination rule can be used towards any type that is fibrant over $\mathbf{\Sigma}A$, following Coquand [2015] (i.e. there is a composition structure for $x : \mathbf{\Sigma}A \vdash C\ \mathtt{Type}$ as an additional argument, which in syntax is always the canonical composition structure for C). Next, we show that this cubical set has a Kan operation, using some instances of the elimination rule with a motive C but *not* using the canonical composition structure denoted by the type C. We cannot write these as instances of the elimination rule in syntax (because the translation of $\mathtt{vhcom}^{r\rightarrow r'}_{\mathbf{\Sigma}A}\,(\alpha \mapsto t)\,(B)$ uses the canonical composition structure for C), so we instead give them their own pattern-matching definitions.

We define a composition structure on $\mathbf{\Sigma}A$ by decomposing composition as homogeneous composition and coercion, and coercion as weak coercion and homogeneous composition (Section 2.7). We use $\mathtt{vhcom}^{r\rightarrow r'}_{\mathbf{\Sigma}A}\,(\alpha \mapsto t)\,(b)$ as the homogeneous composition structure, so it suffices to define a weak coercion structure. First, we add the following function, as a new derivable rule of the type theory and additional elimination principle for $\mathbf{\Sigma}A$:

$$\frac{\Psi \vdash r, r' : \mathbb{I} \qquad \Psi, z : \mathbb{I};\phi;\Gamma \vdash A : \mathtt{Type} \qquad \Psi;\phi;\Gamma \vdash b : \mathbf{\Sigma}(A\langle r/z\rangle)}{\Psi;\phi;\Gamma \vdash \mathtt{wcoe}^{z:r\rightarrow r'}_{\mathbf{\Sigma}A}\,(b) : \mathbf{\Sigma}(A\langle r'/z\rangle)}$$

$$\begin{array}{rcl}\mathtt{wcoe}^{z:r\rightarrow r'}_{\mathbf{\Sigma}A}\,(\mathtt{north}) & \equiv & \mathtt{north} \\ \mathtt{wcoe}^{z:r\rightarrow r'}_{\mathbf{\Sigma}A}\,(\mathtt{south}) & \equiv & \mathtt{south} \\ \mathtt{wcoe}^{z:r\rightarrow r'}_{\mathbf{\Sigma}A}\,(\mathtt{merid}_{r_0}(u)) & \equiv & \mathtt{merid}_{r_0}(\mathtt{coe}^{z:r\rightarrow r'}_{A}(u)) \\ \mathtt{wcoe}^{z:r\rightarrow r'}_{\mathbf{\Sigma}A}\,(\mathtt{vhcom}^{r\rightarrow r'}_{\mathbf{\Sigma}A}\,(\alpha \mapsto t)\,(b)) & \equiv & \mathtt{vhcom}^{r\rightarrow r'}_{\mathbf{\Sigma}A}\,(\alpha \mapsto z.\mathsf{wcoe}^{z:r\rightarrow r'}_{\mathbf{\Sigma}A}\,(t))\,(\mathsf{wcoe}^{z:r\rightarrow r'}_{\mathbf{\Sigma}A}\,(b))\end{array}$$

The last equation states that $\mathsf{wcoe}^{z:r\rightarrow r'}_{\mathbf{\Sigma}A}\,(-)$ strictly preserves hcoms, where the hcom structure for the input is $\mathtt{vhcom}^{r\rightarrow r'}_{\mathbf{\Sigma}A}\,(\alpha \mapsto t)\,(b)$, and the composition structure for the output is the one induced by it (as an hcom structure for a non-dependent type gives rise to a composition structure).

On $r = r'$, $\mathsf{wcoe}^{z:r\rightarrow r'}_{\mathbf{\Sigma}A}\,(-)$ is an η-expanded "inductive identity function". Thus, the path needed to complete the definition of a weak coercion structure is also an η-expanded identity, giving rise to another new elimination rule for suspensions:

$$\frac{\Psi \vdash r, r' : \mathbb{I} \qquad \Psi, z : \mathbb{I}; \phi; \Gamma \vdash A : \mathtt{Type} \qquad \Psi; \phi; \Gamma \vdash b : \mathbf{\Sigma}(A\langle r/z\rangle)}{\Psi; \phi; \Gamma \vdash \eta_{\mathbf{\Sigma}A}(b) : \mathtt{Path}_{\mathbf{\Sigma}A}\left(\mathtt{wcoe}^{z:r\to r'}_{\mathbf{\Sigma}A}(b), b\right)}$$

$$\begin{array}{rcl}
\eta(\mathtt{north}) & \equiv & \Lambda_.\mathtt{north} \\
\eta(\mathtt{south}) & \equiv & \Lambda_.\mathtt{south} \\
\eta(\mathtt{merid}_{r_0}(u)) & \equiv & \Lambda_.(\mathtt{merid}_{r_0}(u)) \\
\eta(\mathtt{vhcom}^{r\to r'}_{\mathbf{\Sigma}A}(\alpha \mapsto t)(b)) & \equiv & \Lambda x.\mathtt{vhcom}^{r\to r'}_{\mathbf{\Sigma}A}(\alpha \mapsto z.\eta(t)\,x, 0/x \mapsto _.\mathtt{wcoe}^{z:r\to r'}_{\mathbf{\Sigma}A}(b), 1/x \mapsto _.b)\,(\eta(b)\,x)
\end{array}$$

The final equation states that η strictly preserves hcoms, using, in the codomain, the fact that $\mathtt{Path}_A(x, y)$ is fibrant over (x, y), but not A, given only a homogeneous composition structure in A. (To be fibrant over the type A, as above, requires a full composition structure for A.)

In our formalized model, we have defined the Kan operation for the general pushout of a cospan of functions $f : C \to A$, $g : C \to B$. The definition follows the same steps as above, but the construction of the weak coercion function and η path are a bit more complex, because in the case of the pushout path constructor, it is necessary to adjust by some naturality paths between $f\langle r'/z\rangle(\mathtt{coe}^{z:r\to r'}_C(c))$ and $\mathtt{coe}^{z:r\to r'}_A(f\langle r/z\rangle(c))$ when $x : \mathbb{I} \vdash f : C \to A$ and $c : C\langle r/z\rangle$. Cavallo and Harper [2019] define a schema of higher inductive types and prove that all of them are Kan (for a slightly different formulation of the Kan operation).

2.16 Identity types

Cohen, Coquand, Huber, and Mörtberg [2018] use an idea of Andrew Swan to construct identity types—that is, with a judgemental equality for J on `refl`—from path types. The same definition of the identity type applies in our model, though the definition of J is a bit more complex. Implementing the J elimination rule uses contractibility of singleton types, which in CCHM follows directly from connections. Here, we use the Kan operation, and must check that our proof of singleton contractibility is `refl` on `refl`. Cavallo and Harper [2019] instead define identity types by taking (some) transports in the identity type to be constructors, analogously to taking certain homogeneous compositions to be values for higher inductive types.

For this section only, we add the following cofibration rules:

$$\frac{\phi, \alpha \vdash_\Psi \beta \quad \phi, \beta \vdash_\Psi \alpha}{\Psi; \phi \vdash \alpha \equiv \beta\ \mathsf{cofib}} \quad \frac{\Psi \vdash \alpha\ \mathsf{cofib} \quad \Psi \vdash \beta\ \mathsf{cofib}}{\Psi \vdash \alpha \wedge \beta\ \mathsf{cofib}} \quad \frac{\phi \vdash_\Psi \alpha \quad \phi \vdash_\Psi \beta}{\phi \vdash_\Psi \alpha \wedge \beta} \quad \frac{\phi \vdash_\Psi \alpha \wedge \beta \quad \Psi; \phi, \alpha, \beta \vdash J}{\Psi; \phi \vdash J}$$

The first rule is "propositional univalence" (interprovable cofibrations are equal); the others state that cofibrations are closed under pullback.

An element of the identity type is a path together with a cofibration remembering where the path is constant:

$$\frac{\Psi;\phi;\Gamma\vdash A\ \texttt{Type}\quad \Psi;\phi;\Gamma\vdash a_0 : A\quad \Psi;\phi;\Gamma\vdash a_1 : A}{\Psi;\phi;\Gamma\vdash \texttt{Id}_A(a_0,a_1)\ \texttt{Type}}$$

$$\frac{\Psi;\phi;\Gamma\vdash p : \texttt{Path}_A\,(a_0,a_1)\quad \Psi;\phi,\alpha;\Gamma\vdash p\equiv\Lambda\_.a_0 : \texttt{Path}_A\,(a_0,a_1)}{\Psi;\phi;\Gamma\vdash(\alpha,p) : \texttt{Id}_A(a_0,a_1)}$$

$$\frac{\begin{array}{l}\Psi;\phi;\Gamma\vdash A\ \texttt{Type}\\ \Psi;\phi;\Gamma\vdash a_0,a_1 : A\\ \Psi;\phi;\Gamma\vdash p : \texttt{Id}_A(a_0,a_1)\\ \Psi;\phi;\Gamma,u:\Sigma x{:}A.\texttt{Id}_A(x,a_1)\vdash C\ \texttt{Type}\\ \Psi;\phi;\Gamma\vdash c : C[(a_1,\texttt{refl}_{a_1})/u]\end{array}}{\Psi;\phi;\Gamma\vdash \texttt{J}_{u.C}(p,c) : C[(a_0,p)/u]}$$

$$\texttt{J}_{u.C}(p,c)\equiv\texttt{transport}_{u.C}(\texttt{scontr}(a_0,p),c)$$

$$\begin{array}{l}\texttt{com}^{z:r\to r'}_{\texttt{Id}_A(a_0,a_1)}\,(\alpha\mapsto z.(\alpha_t,t))\,((\alpha_b,b))\\ \equiv((\alpha\wedge(\alpha_t\langle r'/z\rangle))\vee(r=r'\wedge\alpha_b),\texttt{com}^{z:r\to r'}_{\texttt{Path}_A\,(a_0,a_1)}\,(\beta\mapsto z.t)\,(b))\end{array}$$

We write $\texttt{refl}_a$ for $(\top,\Lambda\_.a)$, where $\top$ is $0=0$.

In the equation defining composition, the idea is that the composite is constant when the path composite is $t\langle r'/z\rangle$ (on α) and $t\langle r'/z\rangle$ is constant (α_t at r'), or when the path composite is b ($r=r'$) and b is constant (α_b). To see that our definition of composition has the right boundary, we use the corresponding equations for the path composite, as well as

$$\begin{array}{l}\alpha\vdash((\alpha\wedge(\alpha_t\langle r'/z\rangle))\vee(r=r'\wedge\alpha_b))\equiv(\alpha_t\langle r'/z\rangle)\\ r=r'\vdash((\alpha\wedge(\alpha_t\langle r'/z\rangle))\vee(r=r'\wedge\alpha_b))\equiv\alpha_b\end{array}$$

These follow from propositional univalence for cofibrations, using the fact that $\alpha\vdash\alpha_t\langle r/z\rangle\equiv\alpha_b$ because the original composition problem was well-formed.

The computation rule for J reduces it to transport and singleton contractibility, which are abbreviations defined as follows. For transport, we define

$$\frac{x:A\vdash C\ \texttt{Type}\quad(\alpha,p):\texttt{Id}_A(a_0,a_1)\quad b:C[a_0/x]}{\texttt{transport}_{x.C}((\alpha,p),b):=\texttt{com}^{z:0\to1}_{C[p\,z/x]}\,(\alpha\mapsto\_.b)\,(b):C[a_1/x]}$$

The $(\alpha\mapsto\_.b)$ above is well-typed because p is constant on α, and thus $p\,z$ is equal to a_0 on α. Our definition is constantly b on α, and α is $\top$ for refl, so transport cancels on refl. Thus, to get the computation rule for J, it suffices to define singleton contractibility in such a way that it sends refl to refl. For singleton contractibility, write $S(a_1)$ for $\Sigma x{:}A.\texttt{Id}_A(x,a_1)$. We define

$$\frac{a_0:A\quad p:\texttt{Id}_A(a_0,a_1)}{\texttt{scontr}(a_0,p):\texttt{Id}_{S(a_1)}((a_1,\texttt{refl}_{a_1}),(a_0,p))}$$

by

$$\begin{array}{l}\texttt{scontr}(a_0,(\alpha,p)):=(\alpha,\Lambda x.(s\langle0/y\rangle,((x=0\vee\alpha),\Lambda y.s)))\\ x:\mathbb{I},y:\mathbb{I}\vdash s:=\texttt{hcom}^{1\to y}_A\,(x=0\mapsto\_.a_1,\ x=1\mapsto y.p\,y,\ \alpha\mapsto\_.a_1)\,(a_1)\end{array}$$

Since this definition outputs the same cofibration α as the input path, it sends `refl` to `refl`. Thus, `J` on `refl` cancels.

3 Semantics

The above syntactic presentation is intended to be sound for a model in cubical sets on the Cartesian cube category $\mathbb{C}$, and other cube categories with suitable structure. Presheaves on the Cartesian cube category are a 1-topos, and we construct the model using the internal language of this topos, a technique first investigated by Orton and Pitts [2016, 2018]; Birkedal et al. [2018]. Following Orton and Pitts [2016, 2018], we use the Agda proof assistant as the internal logic of cubical sets, using a special *modal* extension of Agda (developed by Andrea Vezzosi) in which it is possible to internally describe fibrant universes [Licata, Orton, Pitts, and Spitters, 2018].

3.1 Overview of the formalization

In this section, we describe the main definitions used in the formalization, explain what is proved, and give a sample proof.

3.1.1 Assumptions

First, we make Agda into a pseudo-extensional type theory by postulating function extensionality and a (-1)-truncated/squashed disjunction, which we write as $\alpha 1 \vee \alpha 2$. We write Prop for the type of strict propositions, types that satisfy (x y : P) → x = y. (The formalization uses universe polymorphism, so really there are propositions in each universe.)

The following postulates are a subset of those used in Orton and Pitts [2016], except for Cofib-=, which allows diagonal cofibrations. We postulate an interval type

```
𝕀 : Set
'0 : 𝕀
'1 : 𝕀
```

which is non-trivial

```
iabort : '0 = '1 → ⊥
```

and connected

```
iconnected : (P : 𝕀 → Prop)
  → ((i : 𝕀) → (fst (P i) ∨ (fst (P i) → ⊥)))
  → (((i : 𝕀) → fst (P i)) ∨ ((i : 𝕀) → (fst (P i)) → ⊥))
```

Next, we postulate a notion of cofibration closed under ∀, ∨, and == in $\mathbb{I}$:

```
isCofib : Set → Set
isCofib⊥ : isCofib ⊥
isCofib∨ : ∀ {α1 α2} → isCofib α1 → isCofib α2 → isCofib (α1 ∨ α2)
isCofib= : ∀ {r r' : 𝕀} → isCofib (r = r')
isCofib∀ : ∀ {α : 𝕀 → Set} → ((x : 𝕀) → isCofib (α x)) → isCofib ((x : 𝕀) → α x)
```

When α is a cofibration and t : α → A is a partial element of A, we write

```
A [ α ↦ t ] = Σ[b : A] (pα : α) → t pα = b
```

for an element of A that restricts on α to t, and we use an analogous binary version of this notation.

Finally, we have the strictification axiom:

```
strictify : {α : Set} {cα : Cofib α} (A : α → Set l) (B : Set l)
  → (i : (pα : α) → Iso B (A pα))
  → Σ[B' : Set l [ α ↦ A ] ]
    Iso B (fst B') [ α ↦ (λ pα → eqIso (snd B' pα) ∘iso i pα) ]
```

This says that given a type B and a partial equivalence with A, we can make a type B′ that is isomorphic to B and strictly A on α (and the isomorphism with B restricts to the provided t). This is used to construct Glue types.

Following Licata, Orton, Pitts, and Spitters [2018], we also postulate that the interval is *tiny*, i.e. that exponentiation by the interval has a right adjoint (Figure 1 in that paper). In our formalization, we use tininess only via Theorem 5.2 of that paper, which constructs universes of fibrant types; we review the definition of such universes below.

To construct identity types using Swan's technique, we additionally postulate

```
isCofib-prop : ∀ {A : Set} → (p q : Cofib A) → p = q
isCofib∧ : ∀ {α} {α' : α → Set} → isCofib α → ((x : α) → isCofib (α' x)) → isCofib (Σ α')
Cofib-propositional-univalence : ∀ {α α'} {{cα : Cofib α}} {{cα' : Cofib α'}}
  → (α → α') → (α' → α) → α = α'
```

Any two proofs that a type is a cofibration are equal; interprovable cofibrations are equal, and $\wedge$ preserves cofibrations.

3.1.2 Representation of the Kan operation

Our representation of the Kan operation is analogous to [Orton and Pitts, 2016], adapted to the Kan operation that we use here. First, we define what a composition structure for a family dependent on $\mathbb{I}$ is, with a definition that looks much like the syntactic rule:

```
hasCom : ∀ {l} → (𝕀 → Set l) → Set (lsuc lzero ⊔ l)
hasCom A = (r r' : 𝕀) (α : Set) {_ : Cofib α}
  → (t : (z : 𝕀) → α → A z)
  → (b : A r [ α ↦ t r ])
  → A r' [ α ↦ t r' , (r = r') ↦ ⇒ (fst b) ]
```

We write ⇒ for transport/subst along the Agda equality type (this would be silent in an actual extensional type theory). In this case (fst b) : A r and the partial element needs to have type (r = r') → A r'. Set should be thought of as the universe of non-Kan cubical sets, and this defines what a Kan operation on a cubical set is. Then, for a type A dependent on Γ, a composition structure relative to Γ is a composition structure in the above sense for all paths in Γ:

```
relCom : ∀ {l1 l2} {Γ : Set l1} (A : Γ → Set l2) → Set (lsuc lzero ⊔ l1 ⊔ l2)
relCom {Γ} A = (p : 𝕀 → Γ) → hasCom (A o p)
```

This version matches the filling diagrams we drew in the introduction because p itself can have free variables other than the bound $\mathbb{I}$.

3.1.3 Path types

(Heterogeneous) path types are defined as a pair of a function from the interval and a proof of the boundary constraints.

```
PathO : {l : Level} (A : 𝕀 → Set l) (a0 : A ‘0) (a1 : A ‘1) → Set l
PathO A a0 a1 = Σ [p : (x : I) → A x] (p ‘0 = a0) × (p ‘1 = a1))
```

This gives the the formation, introduction, and elimination rules in Section 2.10, except that in Agda there are explicit proofs of the boundary conditions, which are part of the judgements but not proof terms of the above syntax.

3.1.4 Glue types

At the cubical set level, the data for forming a glue type consists of a cofibration α, a top type T defined on the cofibration, a bottom type B, and a function f from T to B on α:

```
{l : Level}
(α : Set) {{_ : Cofib α}}
(T : α → Set l)
(B : Set l)
(f : (u : α) → T u → B)
```

Following Orton and Pitts [2016], we define glue types as a strictification of the pullback

```
Σ [t : (pα : α) → T pα] B [ α ↦ (λ pα → f pα (t pα)) ]
```

of a partial element t of T and an element of B that agrees f t on α. The interface to glue types (dependent on the above) is:

```
Glue : Set l
Glue-α : (u : α) → Glue α T B f = T u
glue : (t : ((u : α) → T u))
       (b : B [ α ↦ (λ u → f u (top u)) ])
       → Glue α T B f
glue-α : (t : ((u : α) → T u))
         (b : B [ α ↦ (λ u → f u (top u)) ])
         (u : α) → coe (Glue-α α T B f u) (glue α T B f top base) = top u
unglue : Glue α T B f → B
unglue-α : (g : Glue α T B f) → (u : α) → f u (coe (Glue-α _ _ _ _ u) g) = unglue g
Glueβ : unglue (glue α T B f t b) = fst b
Glueη : g = (glue α T B f (λ u → coe (Glue-α α T B f u) g) (unglue g , unglue-α g))
```

These correspond to the rules and equations in Section 2.11.

3.1.5 Universes

When interpreting internal language constructions in cubical sets, Agda's universes (Set i) are interpreted as universes of cubical sets (e.g. Hofmann–Streicher universes in a presheaf category [Hofmann and Streicher, 1997]). To model the type theory presented above, where all types are Kan/fibrant, we require universes U that classify fibrant types, in the sense that a map A : Γ → U means a fibrant type over Γ, i.e. a function A' : Γ → Set equipped with a chosen Kan structure of type relCom A'.

Assuming such a universe internally in pure Agda leads to problems (see Licata, Orton, Pitts, and Spitters [2018, Theorem 3.1]), but such a universe can be described using a *modal* extension of Agda, called agda-flat, that allows one to talk about "closed" or "external" elements of a type. The agda-flat notation f : (x : {♭} A) → B (x) refers to a "function with a flat domain"; the force of this restriction is that f can only be applied to terms that have no free variables—or, more generally, have only other flat variables free. When interpreting the internal language in cubical sets, a (closed) Agda type A denotes a cubical set, while ♭ A denotes (the discrete cubical set on) the set of 0-cells of A. Licata, Orton, Pitts, and Spitters [2018] describe the flat variable mechanism in more detail; to understand our formalization here, it is mainly necessary to know that our definition of the universe follows the modal typing for universes developed in that paper, and that agda-flat checks that our constructions of elements of the universe obey the necessary modal restrictions.

In agda-flat, we axiomatize universes of fibrant types as follows (an unpacked version of Licata, Orton, Pitts, and Spitters [2018, (16)]:

```
U : Set (ℓ₂ ⊔ lsuc l)
El : U → Set l
comEl : relCom El
code : {l1 : {♭} Level} (Γ : {♭} Set l1) (A : {♭} Γ → Set l) (comA : {♭} relCom A)
       → Γ → U
code-El : {l1 : {♭} Level} {Γ : {♭} Set l1} {A : {♭} Γ → Set l} {comA : {♭} relCom A}
          → (x : Γ) → El ((code Γ A comA) x) = A x
comEl-β : {l1 : {♭} Level} {Γ : {♭} Set l1} {A : {♭} Γ → Set l} {comA : {♭} relCom A}
          → (comEl' (code Γ A comA)) = comA
code-η : {l1 : {♭} Level} {Γ : {♭} Set l1} (A : {♭} Γ → U)
         → A = code Γ (El o A) (comEl' A)
```

The first line says that U is an Agda type (of universe level at least 2—because the Kan operation quantifies over cofibrations, which are sets of level zero, we represent types by Agda universes of level at least 1). The second gives the decoding function El that interprets each element of U as an Agda type. The third says that El is Kan; because being Kan is closed under precomposition, this implies that for any function A : Γ → U, El o A is Kan: we can define

```
comEl' : {l1 : Level} {Γ : Set l1} (A : Γ → U) → relCom (El o A)
```

The fourth is the introduction rule for the universe, which pairs a type family with an implementation of its Kan operation. The modal annotations here ensure that code is only applied to "closed" families: A is not allowed to have other free variables besides Γ: if it did, the resulting map into the universe would allow proving that it is Kan relative to them, which is not justified by the introduction rule. Finally, we have β and η rules: El and comEl' extract what code provides, and any flat map into the universe is determined by its type family and Kan composition structure.

These universes are open-ended, in the sense that code includes in U any type for which a Kan operation can be defined—it is not necessary to fix a collection of types when defining the universe.

3.1.6 Suspensions and pushouts

For suspension and pushout types (other higher inductive types could be done similarly), we postulate the definition of the type as a cubical set (the intro, elim, and computation rules). For example, for suspensions, we postulate

```
Susp : (A : Set l) → Set l
north : {A : Set l} → Susp A
south : {A : Set l} → Susp A
merid : {A : Set l} → A → I → (Susp A)
merid0 : {A : Set l} (x : A) → merid x ‘0 == north
merid1 : {A : Set l} (x : A) → merid x ‘1 == south
fcomSusp : {A : Set l} → hasCom (λ _ → Susp A)
Susp-elim : {l l' : Level} {A : Set l}
            (C : Susp A → Set l')
            (comC : relCom C)
            (n : C north)
            (s : C south)
            ((a : A) → PathO (λ x → C (merid a x)) n s)
            (x : Susp A) → C x
```

This asserts the suspension type, the point constructors north and south, the path constructor merid with some boundary constraints, and a homogeneous composition constructor fcomSusp, which freely adds homogeneous composites in Susp A. This is represented by saying that the constantly Susp A function is Kan. Finally, we postulate an elimination rule, which eliminates into any type family that is Kan over Susp A.

3.1.7 Main theorem

Theorem 2. *Each universe U {l+1} has codes for Π, Σ, Path, Id, Nat, Bool, Glue, U {l}, Susp types and is univalent:*

```
Πcode    : {Γ : Set l1} (A : Γ → U {l2}) (B : Σ (El o A) → U {l2}) → (Γ → U {l2})
El-Πcode : {Γ : Set l1} (A : Γ → U {l2}) (B : Σ (El o A) → U {l2})
           → (θ : Γ) → El (Πcode A B θ) == ((x : El (A θ)) → El (B (θ , x)))
Σcode    : {Γ : Set l1} (A : Γ → U {l2}) (B : Σ (El o A) → U {l2}) → (Γ → U {l2})
El-Σcode : {Γ : Set l1} (A : Γ → U {l2}) (B : Σ (El o A) → U {l2})
           → (θ : Γ) → El (Σcode A B θ) == (Σ [x : El (A θ) ] El (B (θ , x)))
Path-code    : {Γ : Set l1} (A : Γ × I → U {l2})
               (a0 : (θ : Γ) → El (A (θ , ‘0))) (a1 : (θ : Γ) → El (A (θ , ‘1)))
               → Γ → U {l2}
El-Path-code : {Γ : Set l1} (A : Γ × I → U {l2})
               (a0 : (x : Γ) → El (A (x , ‘0))) (a1 : (x : Γ) → El (A (x , ‘1)))
               (θ : Γ)
               → El (Path-code A a0 a1 θ) == PathO (λ x → El (A (θ , x))) (a0 θ) (a1 θ)
Id-code : {Γ : Set l1} (A : Γ → U {l2})
          (a0 : (θ : Γ) → El (A θ)) (a1 : (θ : Γ) → El (A θ))
          → Γ → U {lsuc lzero ⊔ l2}
refl    : {l : Level} (A : Set l) (a0 : A) → Id A a0 a0
J       : (A : U {l1}) → (a0 : El A)
```

```
(C : (Σ [a : El A] Id (El A) a a0) → U { l2})
(c : El (C (a0 , refl (El A) a0)))
→ Σ [f : (a1 : El A) (p : Id (El A) a1 a0) → El (C (a1 , p)) ]
   f a0 (refl (El A) a0) = c
Nat-code    : {Γ : Set l2} → (Γ → U { l1})
Nat-code-El : {Γ : Set l2} → (θ : Γ) → El (Nat-code θ) = Nat
Bool-code    : {Γ : Set l2} → (Γ → U { l1})
Bool-code-El : {Γ : Set l2} → (θ : Γ) → El (Bool-code θ) = Bool
Glue-code' : {Γ : Set l1}
    (α : Γ → Set) (cα : (θ : Γ) → Cofib (α θ))
    (T : (θ : Γ) → α θ → U { l2}) (B : Γ → U { l2})
    (f : (θ : Γ) (pα : α θ) → El (T θ pα) → El (B θ))
    (feq : (θ : Γ) (pα : α θ) → isEquivFill _ _ (f θ pα))
    → Γ → U { l2}
Glue-code-El : {Γ : Set l1}
    (α : Γ → Set) (cα : (θ : Γ) → Cofib (α θ))
    (T : (θ : Γ) → α θ → U { l2}) (B : Γ → U { l2})
    (f : (θ : Γ) (pα : α θ) → El (T θ pα) → El (B θ))
    (feq : (θ : Γ) (pα : α θ) → isEquivFill _ _ (f θ pα))
    (θ : Γ) → El (Glue-code' α cα T B f feq θ) = Glue (α θ) { { cα θ } } (El o (T θ)) (El (B θ)) (f θ)
U-code    : U { ℓ2 ⊔ lsuc l}
U-code-El : El (U-code { l}) = U
ua  : { A B : U { l} } (e : Equiv (El A) (El B)) → Path U A B
uaβ : { A B : U { l} } (e : Equiv (El A) (El B)) (a : El A)
    → Path _ (coePathU (ua { _} { A} { B} e) a) (fst e a)
Susp-code    : {Γ : Set l1} → (Γ → U { l}) → Γ → U { l}
Susp-code-El : {Γ : Set l1} → (A : Γ → U { l}) (θ : Γ) → El (Susp-code A θ) = Susp (El (A θ))
```

For each type, we show that there is a code for the universe(s) U of fibrant types with the correct type formation rule, and that its elements are the corresponding Agda type (this implies that the necessary cubical-set level constructs exist—e.g. Π decodes to Agda's Π-types, and so has λ and application). Id types satisfy refl and J with an exact equality for J on refl. ua says that an equivalence can be turned into a path in the universe, while uaβ gives a path between coercing along ua and the input equivalence.

3.1.8 Proof

The main work in defining the above is to define the Kan operation for each type. These definitions in our formalization follow exactly the same steps as the equations for each type given in Section 2. We show the definition for Π types as an example.

First, we define the data from which a Π-type is formed: an element of the universe and a family over it:

```
ΠData : Set _
ΠData = Σ [A : U {l} ] (El A → U {l})
```

This determines the obvious Π-type:

```
Π-from-data : ΠData l → Set _
Π-from-data (A , B) = (x : El A) → El (B x)
```

The main work is showing that Π-from-data is Kan, which in Section 2.9 we did with the following term:

$$\lambda a'.\mathtt{com}^{z:r\to r'}_{B[\mathtt{fill}^{z:r'\to z}_{A}\ (a')/x]}\,(\alpha \mapsto z.t\,(\mathtt{fill}^{z:r'\to z}_{A}\ (a')/x))\,(b\,(\mathtt{coe}^{z:r'\to r}_{A}(a')))$$

In Agda, this is rendered as

```
comΠ : relCom {Γ = (ΠData l)} Π-from-data
comΠ AB r r' α t b =
  (λ a' → ... (fst (forward a'))) ,
  ... ,
  ... where
  A = λ x → fst (AB x)
  B = λ xa → snd (AB (fst xa)) (snd xa)
  fillback : (a' : _) (z : I) → _
  fillback a' z = coeU A r' z a'
  forward : (a' : _) → _
  forward a' = comEl (λ z → B (z , (fst (fillback a' z)))) r r'
                     α (λ z pα → t z pα (fst (fillback a' z)))
                     (fst b (fst (fillback a' r)) ,
                       ...)
```

fillback corresponds to the fill in the type of the B composition; fillback a' r coressponds to $\mathtt{coe}^{z:r'\to r}_{A}(a')$. The ellipses elide Agda proof terms for propositional equalities, which witness that the restrictions of the output are correct (on α, it restricts to t r', and on r=r', it restricts to b), and the boundary condition required by the composition in B in forward.

From this, we define a "universal" code for Π-types, like a type constructor $A : U, B : A \to U \vdash \Pi_A B : U$:

```
Πcode-universal : ΠData → U {l}
Πcode-universal = code ΠData (λ AB → (x : El (fst AB)) → El (snd AB x)) comΠ
```

Finally, precomposing into the universal code gives a more standard Π formation rule:

```
Πcode : {Γ : Set l1} (A : Γ → U {l2}) (B : Σ (El o A) → U {l2}) → (Γ → U {l2})
Πcode A B = Πcode-universal o (λ x → (A x , λ y → B (x , y)))
```

3.2 Validating the axioms

The next step is to argue that the Agda formalization is meaningful in cubical sets. Let C be a finite product category with an object $\mathbb{I}$, with maps $0, 1 : 1 \to \mathbb{I}$ with $0 \neq 1$. In particular C might be $\mathbb{C}$, the Cartesian cube category, which has exactly finite products and such an interval.

First, the logical framework is interpreted in essentially the same way as in Orton and Pitts [2016]. Agda's Π and Σ are interpreted as the dependent types in any presheaf model. For propositions, we use Agda's Π (for ∀) and Σ (for ∧), along with equality types (with postulated function extensionality and uniqueness of identity proofs) and a postulated "squashed" disjunction. Like any presheaf category, cubical sets are a topos, and these propositions can be interpreted using the subobject classifier of the topos (though our Agda development is predicative, so we do not need the full power of this). For natural numbers and booleans, we use Agda datatypes, which can be interpreted as the discrete cubical sets whose 0-cells are these types.

What remains is to validate the axioms specific to cubical sets. Relative to Orton and Pitts [2016], we have changed one axiom (`ax5`, by making $r = r'$ a cofibration), so we need to check that the generalized axiom still works. We have also changed the base category of the presheaves, though most of the theorems in Orton and Pitts [2016] are phrased in sufficient generality that they still apply.

- Connectedness says that decidable propositions are constant on the interval:

$$\forall P : \mathbb{I} \to \texttt{Prop}, (\forall i : \mathbb{I}.P(i) \vee \neg(P(i))) \to (\forall i : \mathbb{I}.P(i)) \vee (\forall i : \mathbb{I}.\neg(P(i)))$$

  In our current formalization, this is used only for defining the Kan operations for strict base types (natural numbers, booleans), but it may also be used for (or implied by other axioms used for) the universe. To see that it is true in $\hat{C}$, the argument in Orton and Pitts [2018] applies to presheaves on any category C that is inhabited (in this case by $\cdot$) and has finite products.

- Non-triviality: $0 \neq 1 : \mathbb{I}$. The interval is interpreted as the representable presheaf on the interval $\mathbb{I}$ in C, and $0/x \neq 1/x$ as maps $\Psi \to \mathbb{I}$ in C.

- Strictification: (Orton–Pitts $\texttt{ax}_8$) This axiom is used only once, to construct glue types from Σ-types, in such a way that on α they are equal to T. Theorem 6.3 of Orton and Pitts [2016] shows that strictification holds in any presheaf topos, if the cofibrations are a subset of the decidable sieves. So the result carries over, as long as we show that our cofibrations are decidable sieves.

- Cofibrations: In the Agda formalization, we postulate that cofibrations are closed under $=_{\mathbb{I}}$, $\vee$, and $\forall x : \mathbb{I}.-$. Semantically, the only constraint is that cofibrations are decidable sieves, i.e. that $\texttt{Cof} \hookrightarrow \Omega$ factors $\texttt{Cof} \hookrightarrow \Omega_{dec} \hookrightarrow \Omega$ through the presheaf of decidable sieves Ω_{dec} [Orton and Pitts, 2016, Definition 6.2]. At each stage Ψ, $\Omega_{dec}(\Psi)$ is the set of sieves on Ψ (precomposition-closed of subsets of $\hom_C(-, \Psi)$) with the property that for a given map $\rho : \Psi' \to_C \Psi$ it is decidable whether ρ is in the sieve. So, as in [Orton and Pitts, 2016], we have many choices for how to interpret cofibrations.

  At one extreme, we can take `Cof` to be a "face lattice" closed under only $=_{\mathbb{I}}$ and $\vee$, and observe that $\forall x : \mathbb{I}-$ is admissible for this fragment by a quantifier elimination argument, as in Cohen, Coquand, Huber, and Mörtberg [2018]: define $\forall x.x = x$ to be true, $\forall x.x = r$ to be false if $x \neq r$, $\forall x.r = r'$ to be $r = r'$ if $x \neq r, r'$, and $\forall x.(\alpha \vee \beta) = (\forall x.\alpha) \vee (\forall x.\beta)$. Then $r = r'$ denotes a decidable sieve, because $\hom_{\mathbb{C}}(\Psi, \mathbb{I})$ has decidable equality—equality of maps is just syntactic identity in the $\Psi \vdash r : \mathbb{I}$ judgement. And the $\vee$ of decidable sieves is always decidable. Orton and Pitts [2018] have given an argument that (for the Cohen, Coquand, Huber, and Mörtberg [2018] notion of cofibration) this definition satisfies $\texttt{Cof} \hookrightarrow \Omega_{dec}$.

  At the other extreme, we can take `Cof` to be Ω_{dec} itself, which is closed under $\forall$ for the following reason. Ω_{dec} classifies monomorphisms $m : A \Rightarrow_{\hat{\mathbb{C}}} B$ such that for all Ψ, $A(\Psi) \Rightarrow B(\Psi)$ has a decidable image, i.e. one can decide whether a $b \in B(\Psi)$ is in the image of $m(\Psi)$. Sattler [2017] has shown that cofibrations are closed under $\forall$ iff they are closed under exponentiation by $\mathbb{I}$, i.e. $A \hookrightarrow B$ is a cofibration implies $A^{\hom_{\mathbb{C}}(-,\mathbb{I})} \hookrightarrow B^{\hom_{\mathbb{C}}(-,\mathbb{I})}$ is a cofibration. Here, the class of cofibrations is monos with decidable image, so it suffices to show that these are closed under exponentiation by $\mathbb{I}$. But if $m : A \hookrightarrow B$ has decidable image for all Ψ, then so does $m^{\hom_{\mathbb{C}}(-,\mathbb{I})} : A^{\hom_{\mathbb{C}}(-,\mathbb{I})} \hookrightarrow B^{\hom_{\mathbb{C}}(-,\mathbb{I})}$, because

$$m^{\hom_{\mathbb{C}}(-,\mathbb{I})}(\Psi) : A^{\hom_{\mathbb{C}}(-,\mathbb{I})}(\Psi) \hookrightarrow B^{\hom_{\mathbb{C}}(-,\mathbb{I})}(\Psi) = m(\Psi \times \mathbb{I}) : A(\Psi \times \mathbb{I}) \hookrightarrow B(\Psi \times \mathbb{I})$$

is just a dimension shift. (This argument works for $\forall x : \hom_{\mathbb{C}}(-, J).-$ for any object J of the cube category, as long as the cube category has finite products.)

- Flat variables: Licata, Orton, Pitts, and Spitters [2018, Remark 4.1] argue that the crisp type theory used in agda-flat can be interpreted in any presheaf topos on a base category with a terminal object.

- Tininess (right adjoint to exponentiation by $\mathbb{I}$): in presheaves on a base category with finite products, all representables are tiny, and $\mathbb{I}$ is interpreted as $y_{\mathbb{I}}$.

Relative to Orton and Pitts [2016], we drop `ax3`/`ax4`, which specify connections, and we use `ax7`, which says that cofibrations are closed under $\wedge$, only if we want `Id` types with a definitional computation rule on `refl`.

3.3 Interpreting the syntax

While we do not give a formal interpretation of the syntax from Section 2 in the model from Section 3, the intended interpretation in the internal language is as follows:

- Dimension contexts Ψ are interpreted as $\mathbb{I}^n$ where $n = |\Psi|$.

- Dimension terms $\Psi \vdash r : \mathbb{I}$ are interpreted as functions $[\![\Psi]\!] \to \mathbb{I}$.

- Dimension formulas $\Psi \vdash \phi\,\mathsf{formula}$ are interpreted as $[\![\Psi]\!] \to \mathsf{Set}$.

- Cofibrations $\Psi \vdash \alpha\,\mathsf{cofib}$ are interpreted as functions $\alpha : [\![\Psi]\!] \to \mathsf{Set}$ such that for all $x : [\![\Psi]\!]$, $\mathsf{Cofib}(\alpha(x))$.

- $\phi \vdash_{\Psi} \alpha$ is interpreted as a function $\Pi x{:}[\![\Psi]\!].[\![\phi]\!](x) \to [\![\alpha]\!](x)$.

- $\Psi; \phi \vdash \Gamma\ \mathsf{ctx}$ are interpreted as functions $(\Sigma x{:}[\![\Psi]\!].[\![\alpha]\!](x)) \to Set_i$.

- $\Psi; \phi; \Gamma \vdash A\ \mathtt{Type}_i$ are interpreted as maps $(\Sigma y{:}(\Sigma x{:}[\![\Psi]\!].[\![\alpha]\!](x)).[\![\Gamma]\!]) \to \mathsf{U}_i$.

- $\Psi; \phi; \Gamma \vdash a : A$ are interpreted as terms of type $\Pi z{:}(\Sigma y{:}(\Sigma x{:}[\![\Psi]\!].[\![\alpha]\!](x)).[\![\Gamma]\!]).\mathsf{El}(A(z))$.

- Judgemental equality is interpreted by equality in the internal language (which we represent by Agda's propositional equality).

The interpretation of every judgement is as "crisp" (closed) term of those types. The internal language's formation, introduction, and elimination, and $\beta\eta$ rules for the interpretations of the types then match the syntactic formation, introduction, elimination, and $\beta\eta$ rules; the equations for the Kan operations correspond to the definitions used in the proof of Theorem 2.

3.4 Connection to the connections Kan operation

The assumptions of the argument in Section 3.2 are satisfied by the connections cube category, so we can also interpret our construction there, and in the de Morgan cube category of Cohen, Coquand, Huber, and Mörtberg [2018] (in both cases, the cube category has finite products and an interval with appropriate structure). Consider the de Morgan case, and take $r =_{\mathbb{I}} r'$, pushouts, pullbacks, and $\forall x : \mathbb{I}.-$ to be cofibrations for both Kan operations. This is a setting that includes the structure of both CCHM and our model.

The CCHM Kan operation is $\mathtt{com}_A^{z:0\to 1}\,(\alpha \mapsto t)\,(b)$. We call a type equipped with such an operation endpoint-Kan. We call a type that is Kan in the sense of Definition 1 diagonal-Kan.

If a type is diagonal-Kan, then it is immediately endpoint-Kan as a special case. Conversely, using connections and the reversal, we define

$$
\begin{aligned}
&\mathit{if}(r = 0, r_1, r_2) := (r \vee r_1) \wedge ((1-r) \vee r_2) \wedge (r_1 \vee r_2)\\
&\mathit{if}(0 = 0, r_1, r_2) \equiv r_1\\
&\mathit{if}(1 = 0, r_1, r_2) \equiv r_2\\
&\mathit{if}(r = 0, r_1, r_1) \equiv r_1
\end{aligned}
$$

Then diagonal Kan composition $\mathtt{com}_A^{z:r\to r'}\,(\alpha \mapsto t)\,(b)$ is derived as

$$
\mathtt{com}^{z:0\to 1}_{A\langle \mathit{if}(z=0,r,r')/z\rangle}\,(\alpha \mapsto z.t\langle \mathit{if}(z=0,r,r')/z\rangle, (r = r') \mapsto \_.b)\,(b)
$$

Notice the use of a diagonal cofibration—without it, this definition would not satisfy the strict $r = r'$ constraint without regularity.

Because any two composites of the same filling problem are path-equal, these two translations must be mutually inverse (up to paths), and this should lead to an equivalence between the universe of endpoint-Kan types and the universe of diagonal-Kan types, in a setting where both exist.

4 Future work

Several projects building on this article are ongoing. At a recent Dagstuhl seminar, we used the formalization described here to check a new lemma designed for optimizing the Kan operation for the universe. Weaver and Licata are investigating a directed type theory in the style of Riehl and Shulman [2017], using this work for the underlying homotopy type theory. Analyzing this cubical model in mathematical terms, such as comparing with the model structure on cubical sets given by Awodey [2018b,a], is an important area for future work. Coquand and Sattler have defined a model structure on cartesian cubical sets using diagonal Kan composition, and shown that it is not Quillen-equivalent to simplicial sets [Sattler, 2018]; investigations into whether any cubical type theories can have the same homotopy theory as simplicial sets are ongoing.

References

T. Altenkirch and A. Kaposi. A syntax for cubical type theory. Available from `http://www.cs.nott.ac.uk/~txa/publ/ctt.pdf`, 2014.

T. Altenkirch, C. McBride, and W. Swierstra. Observational equality, now! In *Programming Languages meets Program Verification Workshop*, 2007.

C. Angiuli and R. Harper. Computational higher type theory II: dependent cubical realizability. arXiv:1606.09638, 2017.

C. Angiuli, R. Harper, and T. Wilson. Computational higher type theory I: abstract cubical realizability. arXiv:1604.08873, 2016.

C. Angiuli, R. Harper, and T. Wilson. Computational higher-dimensional type theory. In *ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, 2017a.

C. Angiuli, K.-B. Hou (Favonia), and R. Harper. Computational higher type theory III: univalent universes and exact equality. arXiv:1712.01800, 2017b.

C. Angiuli, K.-B. Hou (Favonia), and R. Harper. Cartesian cubical computational type theory: Constructive reasoning with paths and equalities. In *Computer Science Logic*, 2018.

S. Awodey. Notes on cubical sets. Available from `https://github.com/awodey/math/blob/master/Cubical/cubical.pdf`, 2016.

S. Awodey. An algebraic fibrant replacement for cubical sets. Talk at Category Theory OctoberFest, 2018a.

S. Awodey. A cubical model of homotopy type theory. *Annals of Pure and Applied Logic*, 169(12): 1270–1294, 2018b. doi: 10.1016/j.apal.2018.08.002. Logic Colloquium 2015.

B. Bentzen. Cubical informal type theory: the higher groupoid structure. arXiv:1806.08490, 2018.

J.-P. Bernardy, T. Coquand, and G. Moulin. A presheaf model of parametric type theory. In *Mathematical Foundations of Programming Semantics*, 2015.

M. Bezem, T. Coquand, and S. Huber. A model of type theory in cubical sets. In R. Matthes and A. Schubert, editors, *19th International Conference on Types for Proofs and Programs (TYPES 2013)*, pages 107–128, 2014. doi: 10.4230/LIPIcs.TYPES.2013.107.

L. Birkedal, A. Bizjak, R. Clouston, H. B. Grathwohl, B. Spitters, and A. Vezzosi. Guarded cubical type theory. *Journal of Automated Reasoning*, June 2018. doi: 10.1007/s10817-018-9471-7.

G. Brunerie and D. R. Licata. A cubical infinite-dimensional type theory. Talk at Oxford Workshop on Homotopy Type Theory, November 2014.

U. Buchholtz and E. Morehouse. Varieties of cubical sets. In *Relational and Algebraic Methods in Computer Science: 16th International Conference, RAMiCS 2017, Lyon, France, May 15-18, 2017, Proceedings*. 2017.

E. Cavallo and R. Harper. Higher inductive types in cubical computational type theory. In *ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, 2019.

D.-C. Cisinski. Les préfaisceaux comme modèles des types d'homotopie. *Astérisque, Soc. Math. France*, Volume 308, 2006.

C. Cohen, T. Coquand, S. Huber, and A. Mörtberg. Cubical type theory: A constructive interpretation of the univalence axiom. In T. Uustalu, editor, *21st International Conference on Types for Proofs and Programs (TYPES 2015)*, pages 5:1–5:34, 2018. doi: 10.4230/LIPIcs.TYPES.2015.5.

T. Coquand. Variations on cubical sets. Talk at Oxford Workshop on Homotopy Type Theory. Available from `http://www.cse.chalmers.se/~coquand/comp.pdf`, November 2014a.

T. Coquand. Variations on cubical sets (diagonals version). Available from `http://www.cse.chalmers.se/~coquand/diag.pdf`, 2014b.

T. Coquand. Higher inductive types as cubical sets. Available from `http://www.cse.chalmers.se/~coquand/hit1.pdf`, 2015.

T. Coquand, S. Huber, and A. Mörtberg. On higher inductive types in cubical type theory. In *IEEE Symposium on Logic in Computer Science*, 2018.

N. Gambino and C. Sattler. The Frobenius condition, right properness, and uniform fibrations. *Journal of Pure and Applied Algebra*, 221(12):3027 – 3068, 2017.

R. Harper. Computational type theory, 2018. URL `https://www.cs.uoregon.edu/research/summerschool/summer18/topics.php`. Lectures at Oregon Programming Languages Summer School 2018.

R. Harper and C. Angiuli. Computational (higher) type theory, 2018. URL `http://www.cs.cmu.edu/~rwh/talks/POPL18-Tutorial.pdf`. Tutorial at POPL 2018.

M. Hofmann. *Extensional Concepts in Intensional Type Theory.* PhD thesis, University of Edinburgh, 1995.

M. Hofmann and T. Streicher. Lifting grothendieck universes. Available from `http://www.mathematik.tu-darmstadt.de/~streicher/NOTES/lift.pdf`, 1997.

S. Huber. Canonicity for cubical type theory. *Journal of Automated Reasoning*, 2018. doi: 10.1007/s10817-018-9469-1.

V. Isaev. Homotopy type theory with an interval type. Available from `https://valis.github.io/doc.pdf`, 2014.

D. R. Licata, I. Orton, A. M. Pitts, and B. Spitters. Internal universes in models of homotopy type theory. In *Formal Structures for Computation and Deduction (FSCD)*, 2018. arXiv:1801.07664.

I. Orton and A. M. Pitts. Axioms for modelling cubical type theory in a topos. In *Computer Science Logic*, 2016.

I. Orton and A. M. Pitts. Axioms for modelling cubical type theory in a topos (extended version). *Logical Methods in Computer Science*, 14(4), 2018.

A. M. Pitts. Nominal Presentation of Cubical Sets Models of Type Theory. In *20th International Conference on Types for Proofs and Programs (TYPES 2014)*. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2015.

A. Polonsky. Internalization of extensional equality. Available from `http://arxiv.org/abs/1401.1148`, 2014.

E. Riehl and M. Shulman. A type theory for synthetic ∞-categories. arXiv:1705.07442, 2017.

C. Sattler. The equivalence extension property and model structures. arXiv:1704.06911, 2017.

C. Sattler. Do cubical models of type theory also model homotopy types? Talk at Workshop on Types, Homotopy Type theory, and Verification at Hausdorff Institute, 2018.

The Univalent Foundations Program, Institute for Advanced Study. *Homotopy Type Theory: Univalent Foundations Of Mathematics*. Available from `homotopytypetheory.org/book`, 2013.

V. Voevodsky. A very short note on homotopy λ-calculus. *Unpublished*, page 1–7, September 2006. URL `http://www.math.ias.edu/vladimir/files/2006_09_Hlambda.pdf`.