Naive cubical type theory

Bruno Bentzen[1]

[1]Carnegie Mellon University,
Pittsburgh, PA, USA
`bbentzen@andrew.cmu.edu`

Abstract

This paper proposes a way of doing type theory informally, assuming a cubical style of reasoning. It can thus be viewed as a first step toward a cubical alternative to the program of informalization of type theory carried out in the homotopy type theory book for dependent type theory augmented with axioms for univalence and higher inductive types. We adopt a cartesian cubical type theory proposed by Angiuli, Brunerie, Coquand, Favonia, Harper, and Licata as the implicit foundation, confining our presentation to elementary results such as function extensionality, the derivation of weak connections and path induction, the groupoid structure of types, and the Eckmman–Hilton duality.

1 Introduction

Cubical type theory is a flavor of higher-dimensional type theory that is more directly amenable to constructive interpretations [11, 3, 4]. It emerged as an alternative to the original formulation of homotopy type theory [18], which is based on an extension of dependent type theory with axioms asserting univalence and the existence of higher inductive types. Unlike this axiomatic formulation, which is well known to block computation since axioms postulate new canonical terms for some types without specifying how to compute with them, cubical type theory relies on cubical methods to properly specify the computational behavior of terms at higher dimensions. But, despite achieving a presentation of higher-dimensional type theory with computational content, the appeal to the sophisticated machinery of cubical reasoning makes cubical type theory even more impenetrable to the uninitiated. As of today, the only way to learn cubical type theory is either through the study of cubical models of type theory or using proof assistants that implement cubical type theory [17, 19]. This can make it hard to present cubical ideas to a wider mathematical audience.

In the spirit of Halmos' seminal book [14], the naive type theory project introduced by Constable [12] and recently advocated by Altenkirch [1] is aimed at making type theory more accessible to mathematicians by introducing the subject in an informal but, in principle, formalizable way, that is, by proposing an intuitive presentation independent of any technical appeal to the rules of inference of the formalism. In this sense, "naive" is in contrast with "formal", meaning that naive type theory can be well understood as formal type theory

approached from a naive perspective [14, 12]. Ideally, it should be possible to make a naive type theory formally precise. This means that underlying a naive type theory one should always be able to find an implicit formal system made up of structural rules, rules of equality, type former rules like formation, introduction, elimination, and computation, and other rules defining a collection of types, their terms, and equality conditions. The point here is that we can almost always forget about those technicalities when assuming the naive point of view, according to which a type theory is viewed as an intuitive semantics rather than a body of axioms and rules of inference.

For higher-dimensional type theories in particular, adopting such a naive approach amounts to developing an informal but rigorous explanation of type-theoretic reasoning at higher dimensions. This is an important enterprise that can be expected to provide at least two major benefits:

(1) If higher-dimensional type theory is to be taken seriously as a foundation for mathematics or research program, then, it should be accessible with a minimum of logical formalism to nonexperts.

(2) Pen-and-paper proofs given in a homotopically-inspired informal language for mathematics could be more closely related to practices of working mathematicians such as the identification of isomorphic structures. Moreover, proof mechanization might require significantly less effort, as type theory is the basis of several proof-assistants.

For homotopy type theory, as originally formulated, this informalization was accomplished in the canonical book on the subject [18], in which the theory is systematically developed from scratch with the use of language and notation that are similar to that of ordinary mathematics, but without making precise reference to the axioms or rules of inference that establish the formal system. Put differently, the so-called homotopy type theory book [18] develops a naive type theory, a rigorous style of doing everyday mathematics informally assuming homotopy type theory as the underlying foundation, which is then used to give informal proofs of theorems from various areas of mathematics, such as logic, set theory, category theory, and homotopy theory throughout the book.

This paper can be viewed as an initial effort to introduce readers to the naive way of doing cubical type theory in an analogous manner. Thus, the main goal here is to propose a naive presentation for cubical type theory that has a comparable degree of rigor, while also highlighting the distinctive aspects of the cubical approach to higher-dimensional type theory by means of proofs of some elementary results. For that reason, the reader may notice that, in this paper, arguments by the principle of path induction, which is used in homotopy type theory as the elimination rule for the path type [18, §1.12], are almost completely ignored in favor of purely cubical arguments. This often results in diagrammatic proofs that are conceptually simpler, cubically speaking, but may require more ingenuity. Now, considering that cubical type theory comes in many different forms, depending on the structure one imposes on the cube category the framework is built on, we should emphasize that our naive presentation is founded on cartesian cubical type theory [3], a formal theory developed by Angiuli, Brunerie, Coquand, Favonia, Harper, and Licata based on the cartesian cube category, that is, the free category with finite products on

an interval object [5], a category of cubes which has faces, degeneracies, symmetries, diagonals, but no connections or reversals—unlike the De Morgan cube category and the cubical type theories based on this structure [11, 13].

The remainder of this paper is organized as follows: in the next section, a short introduction to cubical type theory from the naive point of view is given and its semantics is described informally. Then, Kan operations are discussed in some detail. Next, we derive weak connections and give informal proofs of the groupoid structure of types. Subsequently, we consider dependent paths and heterogeneous composition. Finally, proofs of path induction and the Eckmann–Hilton duality are presented, followed by a section containing some final remarks. This paper is intended to be self-contained, but we assume throughout some general familiarity with the concepts of homotopy type theory.

2   The cubical point of view

Naive cubical type theory is the idea that cubes are the basic shapes used to characterize the structure of higher-dimensional mathematical objects. It is, however, grounded on the same homotopical intuition [6, 20] which regards a type A as a space, a term $a : A$ as a point of the space A, a function $f : A \rightarrow B$ as a continuous map, a path $p : \mathsf{path}_A(a, b)$ as a path from point a to point b in the space A, a universe type $\mathcal{U}$ as a space, the points of which are spaces, and a type family $P : A \rightarrow \mathcal{U}$ as a fibration. Like homotopy type theory, spaces are understood purely from the point of view of homotopy theory, meaning that homotopy equivalent spaces are equal up to a path.

The distinctive cubical perspective of type theory starts with the consideration of an abstraction of the unit interval in the real line, that is, a space consisting of only two points, 0 and 1, which we call the *interval type* $\mathbb{I}$. Thus, an interval variable $i : \mathbb{I}$ is a type-theoretic abstraction of the idea of a point varying continuously in the unit interval. Traditionally, a path in a topological space is a continuous function from the unit interval. This point-set topological description is generalized in cubical type theory in the sense that we represent paths as functions out of the interval. As will be discussed in the remainder of this section, this change of perspective regarding paths is the key ingredient that allows us to visualize them as abstract cubes in higher dimensions.

2.1   The type of paths

It is useful to have a type of paths. Certainly, the most obvious way of obtaining such a type is to use the function type itself. We shall often refer to the type of functions from the interval $\mathbb{I} \rightarrow A$ as the *line type*, and call its terms lines. It is convenient to work with paths with arbitrary endpoints using line types, but as soon as we start considering paths from a particular point to another point, the limitations of this strategy begin to become apparent. In order to better deal with paths with fixed endpoints, we need a slightly more sophisticated variant of the function type called the *path type*, an internalization of functions from the interval that makes their endpoints fully explicit. Hence given any type $A : \mathcal{U}$ and terms $a, b : A$, we can form the type of paths starting from a to b in A, which we denote by $\mathsf{path}_A(a, b)$ and call the path type of A and a and b.[1]

What can we do with paths in the path type? Given a path $p : \mathsf{path}_A(a, b)$ and an interval variable $i : \mathbb{I}$, we can apply the path to obtain a term of the type A depending on i, denoted $p(i)$ and called the value of p at position i. Moreover, applying a path to one of the interval endpoints should result in the corresponding endpoint of the path. This means that for every term $p : \mathsf{path}_A(a, b)$ the following endpoint equalities hold definitionally:

$$p(0) \equiv a : A \quad \text{and} \quad p(1) \equiv b : A.$$

The canonical way to construct paths is by abstraction: given a term $a : A$ which may depend on $i : \mathbb{I}$, we write $\lambda i.a$ to indicate a path from $a[0/i]$ to $a[1/i]$ in A, where, by convention, the scope of the binding $\lambda-.$ extends over the entire expression to its right unless otherwise noted by the use of parentheses.

It should come as no surprise that paths require equalities similar to those common for functions in the lambda calculus. More specifically, when a path abstraction term is applied to an interval point, we require a computation that plays the role of β-reduction:

$$(\lambda i.a)(j) \equiv a[j/i].$$

We also expect that "every path is a path abstraction", the uniqueness principle for the path type, meaning that we also consider the following η-expansion rule:

$$p \equiv \lambda i.(p(i)) \qquad \text{(when } i \text{ does not occur in } p).$$

This identifies any path p with the "path that applies p on the interval", thus endowing paths with an extensional aspect. The use of this equality is crucial to the derivation of path induction in §6.1.

2.2   How we should think of paths

The use of diagrams will play a key role in our visualization of paths, for many cubical type-theoretic problems can be rigorously stated and solved using diagrammatic arguments. How should we think of paths cubically? This view follows naturally from the conception of paths as functions from the interval and the homotopical intuition. First of all, we visualize a term $a : A$ as a point:

$$\cdot\, a$$

In the first dimension we think of a term $p : \mathbb{I} \to A$ as a path from the point $p(0)$ to the point $p(1)$. Given an interval variable $i : \mathbb{I}$, such a path can be visualized as an abstract line $p(i)$ in the "direction" i that goes from the initial to the terminal point of the path, as shown in the following diagram:

$$p(0) \xrightarrow{\quad\quad p(i) \quad\quad} p(1) \qquad\qquad \xrightarrow{\; i \;}$$

Extending the interpretation to higher dimensions, it is natural to think of a function $h : \mathbb{I} \to \mathbb{I} \to A$ as a homotopy of paths. Such a homotopy consists of two simultaneous continuous deformations and it can be visualized as an abstract square having four paths at each edges, as shown in the following diagram, where the lines $h(0)$ and $h(1)$ are the bottom and top faces in the direction i, and $\lambda j.h(j, 0)$ and $\lambda j.h(j, 1)$ the left and right faces in the j-direction, respectively:

$$h(1,0) \xrightarrow{\quad h(1) \quad} h(1,1)$$

$$\lambda j.h(j,0) \Big\uparrow \qquad\qquad h \qquad\qquad \Big\uparrow \lambda j.h(j,1) \qquad j \Big\uparrow \!\!\!\xrightarrow{\quad} i$$

$$h(0,0) \xrightarrow[\quad h(0) \quad]{} h(0,1)$$

In the third dimension, we consider homotopies between homotopies, which, as expected can be pictured as cubes. It is hard enough to visualize paths at even higher dimensions, but, most certainly, the reader can already guess the general pattern here: at dimension $n+1$, we have $n+1$-dimensional paths, which can be visualized as hypercubes with $2(n+1)$ faces formed by n-dimensional cubes. Finally, it is worth noting that this cubical structure applies for types $A : \mathcal{U}$ as well, since types are just terms of a type of universes.

2.3 How can we use paths?

Already at this stage we can see how this novel account of paths as functions out of the interval is a significant departure from the traditional approach taken in homotopy type theory [18], in which the path type is defined as an inductive type generated by a single reflexivity constructor. The following lemmas will serve to show some fundamental differences of the cubical approach.

We start with the fact that a term analogous to this reflexivity constructor is definable by considering constant functions from the interval, thus providing us a way to trivially regard terms as degenerate paths.

Lemma 2.3.1. *For every type A and every $a : A$, there exists a path*

$$\mathsf{path}_A(a,a)$$

called the reflexivity path of a and denoted refl_a.

Proof. Suppose that $i : \mathbb{I}$ is a fresh interval variable. By assumption, a does not depend on i, meaning that $\lambda i.a$ gives us a path starting from a to a in A. $\square$

Since in the homotopy interpretation we view functions as continuous maps, it is natural to expect that functions preserve paths, as shown in the homotopy type theory book [18, §2.2]. Here we state this property with the following lemma for non-dependent functions:

Lemma 2.3.2 (Function application [8, 11])**.** *Given a function $f : A \to B$ and terms $a, b : A$, we have an operation*

$$\mathsf{ap}_f : (\mathsf{path}_A(a,b)) \to (\mathsf{path}_B(f(a), f(b)))$$

such that $\mathsf{ap}_f(\mathsf{refl}_a) \equiv \mathsf{refl}_{f(a)}$.

Proof. Given $p : \mathsf{path}_A(a,b)$ and an interval variable $i : \mathbb{I}$, we have $f(p(i)) : B$, for we can apply $f : A \to B$ to $p(i) : A$. By abstraction, we have a path

$$\mathsf{ap}_f(p) :\equiv \lambda i.f(p(i)).$$

from $f(a)$ to $f(b)$ in B, since p is a path from a to b. Moreover, we have

$$\mathsf{ap}_f(\mathsf{refl}_a) \equiv \lambda i.f(\mathsf{refl}_a(i))$$
$$\equiv \lambda i.f(a)$$
$$\equiv \mathsf{refl}_{f(a)}$$

as requested.   □

We note that `ap` behaves strictly functorially in the sense that the following are definitional equalities for identity functions, compositions of functions, and constant functions [8, 11]:

$$\mathsf{ap}_{id_A}(p) \equiv p$$
$$\mathsf{ap}_{f \circ g}(p) \equiv \mathsf{ap}_f(\mathsf{ap}_g(p))$$
$$\mathsf{ap}_{\lambda\_.a}(p) \equiv \mathsf{refl}_a.$$

In homotopy type theory [18], in which the path type is defined as an inductive family, those equalities only hold up to homotopy.

Another crucial difference is that, with the cubical notion of path, we are able to prove that two pointwise equal functions are equal up to a path. This is known as function extensionality, a property that cannot be obtained without the use of axioms in homotopy type theory [18, §2.9,§4.9]. This lack of function extensionality is a common occurrence shared by many intensional forms of dependent type theory (except for observational type theory).

Theorem 2.3.3 (Function extensionality [11])**.** *Suppose that* $f, g : \prod_{(x:A)} B(x)$ *are functions. There is an operation*

$$\mathsf{funext}_{f,g} : \prod_{(x:A)} \mathsf{path}_{B(x)}(f(x), g(x)) \to \mathsf{path}_{\prod_{(x:A)} B(x)}(f, g)$$

Proof. If we are given $H : \prod_{(x:A)} \mathsf{path}_{B(x)}(f(x), g(x))$ and an arbitrary $x : A$, we have a path $H(x)$ from $f(x)$ to $g(x)$ in $B(x)$. For $i : \mathbb{I}$, $H(x)$ at i gives a path from $\lambda x.f(x)$ to $\lambda x.g(x)$ in $A \to B$,

$$\lambda i.\lambda x.(H(x)(i)) : \prod_{(x:A)} B(x).$$

By η-conversion, it can be seen that this path goes from f to g.   □

3   There are enough paths

So far we have only discussed general properties of paths. However, nothing about what has been said leaves us in a position to transport terms along paths or invert and concatenate paths, to give some examples. Even though compared to homotopy type theory we get closer to the types as spaces intuition in the cubical setting, since paths are viewed as functions out of the interval, this attitude alone cannot provide the required structure to support it. There is even a certain advantage in homotopy type theory because path induction is

accepted as a primitive form of reasoning and all the structure needed to model spaces can be derived from it [18, §2]. In the end path induction guarantees that everything in homotopy type theory can be interpreted in a model of spaces such as that of Kan simplicial sets, and there is nothing strictly simplicial about the spatial intuition required to do homotopy type theory informally.

In cubical type theory, on the other hand, the connection between the theory and the intended model of spaces happens at a much higher level, except for the fact that Kan cubical sets are considered instead [11, 3, 4]. So when doing naive cubical type theory it is no longer possible to maintain the same superficial level of intuition because cubical reasoning becomes imperative. In order to properly model spaces in a cubical setting it is therefore unacceptable to rely on this generic homotopy type-theoretic strategy of using path induction as a fundamental reasoning principle. Instead, to ensure that types have enough structure we impose cubically-inspired primitive operations (Kan operations) such as *transport* and *composition* [3, 4]. They stipulate certain requirements that paths in a type should satisfy, and, to ensure that all terms compute properly, every type must come with its own specific operations, which determine how we should understand a Kan operation on a type in terms of reductions to their constructors or eliminators [3]. The general aspects of transport and composition are discussed in the remainder of this section.

3.1 Transportation along paths

Transport is a cubical generalization of the transport lemma [18, Lem 2.3.1]. Recall that, due to univalence [21], a principle that characterizes the path space of the universe type $\mathcal{U}$, a path between types is (equivalent to) a homotopy equivalence between spaces [18]. In the cubical setting, transport states that, given any path between types $A : \mathbb{I} \to \mathcal{U}$ and any term $a : A(i)$, we have a term of the type $A(j)$, called the transport of a from i to j along A, and denoted by $a_A^{i \rightsquigarrow j} : A(j)$. We also require that static transportations have no effect, which means that, when i is j, we have $a_A^{i \rightsquigarrow i} \equiv a$.

In general, transporting along a degenerate path is not strictly the same as doing nothing, meaning that $a_{\mathsf{refl}_A}^{i \rightsquigarrow j} \not\equiv a$. This is in contrast with the transport operation of homotopy type theory [18], in which transportation along reflexivity is taken to be the identity function. The following lemma shows in what sense a generalization of this operation is taking place:

Lemma 3.1.1. *Given a type family $C : A \to \mathcal{U}$, terms $a, b : A$ and a path $p : \mathsf{path}_A(a, b)$, there is a function*

$$p_* : C(a) \to C(b).$$

Proof. Assume we have $c : C(a)$. Note first that we have a path of types

$$D(p) :\equiv \lambda i.C(p(i)) : \mathbb{I} \to \mathcal{U}$$

from $C(a)$ to $C(b)$. The transportation of c from 0 to 1 along $D(p)$ gives us

$$c_{D(p)}^{0 \rightsquigarrow 1} : C(b)$$

as desired. $\square$

Although $p_*(\mathsf{refl}_a)$ is not the identity function up to definitional equality, this can be shown to be the case up to a path.

Lemma 3.1.2. *For a type family $C : A \to \mathcal{U}$ and $a : A$, we have a path*

$$\mathsf{path}_{C(a) \to C(a)}((\mathsf{refl}_a)_*, \mathsf{Id}_{C(a)}).$$

Proof. The idea is to use function extensionality. By Theorem 2.3.3, we may assume that $c : C(a)$. It remains to find a path from $(\mathsf{refl}_a)_*(c)$ to c. But by definition, $(\mathsf{refl}_a)_*(c) \equiv c^{0 \rightsquigarrow 1}_{\lambda\_.C(a)}$. Now fix $i : \mathbb{I}$. We observe that $c^{i \rightsquigarrow 1}_{\lambda\_.C(a)} : C(a)$, since $(\lambda\_.C(a))(j) \equiv C(a)$ for any $j : \mathbb{I}$. But recall that the static transportation $c^{1 \rightsquigarrow 1}_{\lambda\_.C(a)}$ has no effect, so this term is definitionally equal to c. In other words, $\lambda i . c^{i \rightsquigarrow 1}_{\lambda\_.C(a)}$ is the path we are looking for. $\square$

3.2 Composition of paths

Composition ensures that every open cube in a type can be filled. For example, at dimension one, if we are given adjacent lines $p, q, r : \mathbb{I} \to A$ such that the initial point of p matches the initial point of q and the terminal point of p matches the initial point of r, composition provides a new line, the composite, that goes from the terminal point of q to the terminal point of r.

In fact, composition states more than that. It asserts the existence of the whole square witnessing the filling of the open box (the square in the diagram above). This is called the *filler* of the composition. When j is 1, the filler gives us the composite, the missing face of the open box. The remaining faces of the filler are the faces of the open box. So, when j is 0, the filler equals p (which we may call the cap of the composition) and when i is 0 or 1, it will be q (the left tube) or r (the right tube), respectively. More generally, we refer to any open shape satisfying the relevant adjacency conditions as a *composition scenario*.

To illustrate the use of composition, we shall now prove symmetry and transitivity for paths with the definition of inverse and concatenation functions. Let us consider the former first. In cubical type theories based on a De Morgan structure [11, 13], path inversion comes for free with a primitive (strict) reversal operator but in a cartesian account some work needs to be done to derive it from the Kan operations.

Lemma 3.2.1. *For every type A and every $a, b : A$, there is a function*

$$\mathsf{path}_A(a, b) \to \mathsf{path}_A(b, a)$$

called the inverse function and denoted $p \mapsto p^{-1}$.

Proof. Suppose that $i, j : \mathbb{I}$. The idea of the following proof is to observe that $p : \mathsf{path}_A(a, b)$ gives a "line" $p(j) : A$ in the j-direction from a to b in A. Since

its initial point is a, we have an open box whose faces are formed by the lines $p(j)$ (left tube), $\mathsf{refl}_a(i)$ (cap) and $\mathsf{refl}_a(j)$ (right tube). Note that the latter two lines are degenerate, i.e. by definition, they are the same as a. Degeneracy is indicated using double bars in the following diagram:
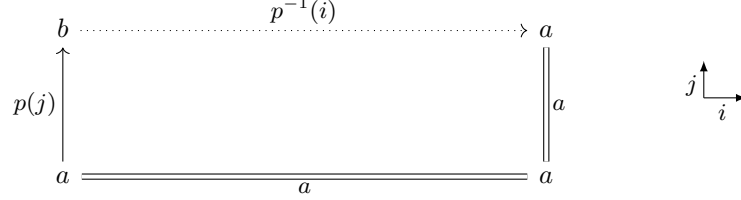
By composition, this open box must have a lid (top), an i-line from b to a in A, which gives us, by path abstraction, the required path p^{-1}. $\qquad\square$

Fillers of compositions will be relevant to our constructions later on, so it can be convenient to have a special symbolism to talk about them directly. That is the purpose of the following notation:

Definition 3.2.2. *Given a composition scenario, $i, j : \mathbb{I}$, and a composite p for an open cube in the i direction, then $\mathsf{fill}_j(p)$ stands for its filler in the j direction.*

When no confusion occurs, we may also write it $\mathsf{fill}(p)$. Thus, the (i, j)-square defined in the proof of Lemma 3.2.1 can be denoted by $\mathsf{fill}(p^{-1}(i))$.

The next lemma defines path concatenation:

Lemma 3.2.3. *For every type A and every $a, b, c : A$, there is a function*

$$\mathsf{path}_A(a, b) \to \mathsf{path}_A(b, c) \to \mathsf{path}_A(a, c)$$

denoted $p \mapsto q \mapsto p \cdot q$. We call $p \cdot q$ the concatenation of p and q.

Proof. Given paths $p : \mathsf{path}_A(a, b)$ and $q : \mathsf{path}_A(b, c)$, we can construct an i-line $p(i)$ from a to b and a j-line $q(j)$ from b to c. Once again, we note that we have an open square,
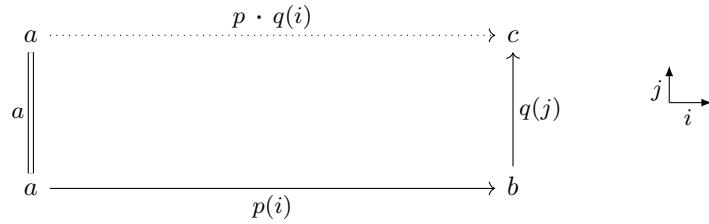
and the path $p \cdot q$ is obtained by path abstraction on its composite. $\qquad\square$

We shall make diagrams as explicit as possible throughout the remainder of this paper, but, for the sake of readability, we omit labels for degenerate lines, since the reader should be able to correctly guess the information by checking the endpoints of the line. It is also important to bear in mind that, while we only considered the simplest composition scenario where we compose lines to form squares in the examples above, we can also compose squares to form cubes and, more generally, n-cubes to form $(n + 1)$-cubes. We shall deal with compositions

at dimension two in §4.1. Composition scenarios at even higher dimensions will not be discussed in this paper.

In addition to the specification of the cap and tubes of an open cube, we also have to admit the specification of diagonals to make composition work properly in a cartesian setting, thereby allowing the diagonal of the filler to be definitionally equal to the designated diagonal [3, 4]. But, as we will not be using diagonals, we do not need to worry about this here.

3.3 The interval is not Kan

We suggested above that every type is Kan. In fact, the interval is the only exception to this rule, since we have been implicitly treating it as a "type", but it actually does not support any Kan operations. Consider, for example, the identity path on the interval, which goes from 0 to 1,

$$\lambda i.i : \mathsf{path}_{\mathbb{I}}(0,1).$$

If the interval were Kan, then the identity path would have an inverse. But what could that be in our cartesian setting? Given that the interval lacks some properties required by typehood in a cubical type theory, it is often referred to as a pretype. To deal with this fact, we adopt the convention that the interval can only occur as the antecedent of a function type.

4 Two-dimensional constructions

Now that we have defined path symmetry and concatenation we can show that they satisfy the groupoid laws up to homotopy, which means that we shall be mainly concerned with paths of one dimension higher than the ones we are given. This is the aim of this section. Two-dimensional paths are determined by squares, which represent a mutual identification of lines on the opposing sides, and, as a result, their construction often requires two-extent compositions.

4.1 Weak connections

In cubical type theory, it is useful to have extra kinds of degeneracies known as connections, which can be thought of as meets and joins on the interval. Just as reversals, connections are built-in in a De Morgan setting [11, 13]. They are not hard to derive in cartesian cubical type theory, but some of the computation rules only hold up to a path (hence we call them "weak" connections).

**Lemma 4.1.1** (Meet). *For every type A and $a, b : A$, there is an operation*

$$-(- \wedge -) : \mathsf{path}_A(a,b) \to \mathbb{I} \to \mathbb{I} \to A$$

*such that, for any $p : \mathsf{path}_A(a,b)$ and any $i, j : \mathbb{I}$, the following holds:*

The proof of this lemma requires a two-extent composition and thus a few remarks are in order here before we proceed. In the one-dimensional case, as we have seen as far, we just have a pair of tubes, so an open square is enough to complete the composition. With two pairs of tubes, each pair corresponding to the dimension in question, we are required to form an open object of the next higher shape, i.e. a cube. In other words, assuming that we want to fill an open (i, j, k)-cube in the j direction, we are expected to determine its (i, k)-face (bottom), two (k, j)-faces (left and right) and two (i, j)-faces (back and front), and those squares must all be adjacent up to definitional equality. If this holds, then the composite is a (i, k)-square. (For the same reasons, performing an n-extent composition requires a construction of an $(n + 1)$-cube).

Finally, it is often convenient to illustrate two-extent compositions using a two-dimensional structure in the form of a cube seen from above.[2] The following diagram indicates how we shall be drawing two-extent compositions in the paper:
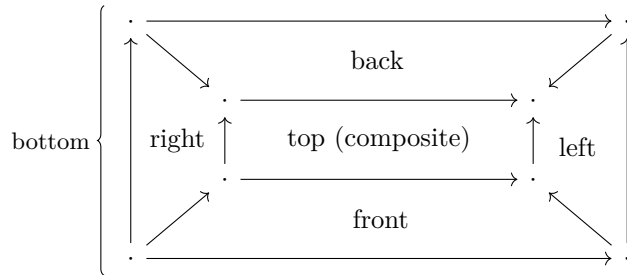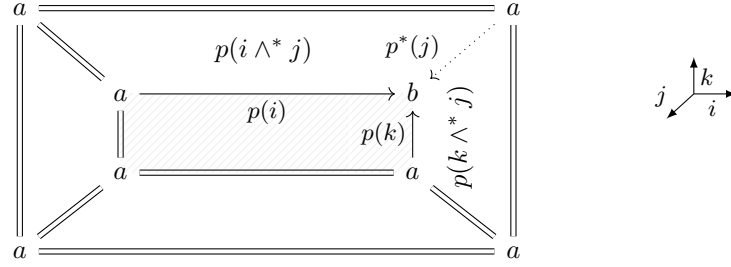

When referring to such diagrams we may call the center and outer squares the top (composite) and the bottom faces of the cube (filler), the top and bottom squares the back and front faces, and the left and right squares the left and right faces, respectively.

Proof. Given $p : \mathsf{path}_A(a, b)$, we are to find a (i, j)-square whose top face is $p(i)$, right face is $p(j)$, left and bottom faces are a. First, by composition, we obtain, for any $i, j : \mathbb{I}$, a square that looks like a "halfway" connection, called $p(i \wedge^* j)$.
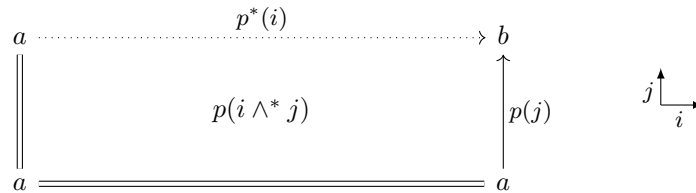
Note that this square is not the one that we are looking for. Its top face is given by a certain path obtained by composition, which we denote p^*, and this path needs not be definitionally equal to p.

But we are able to fix this mismatch in a higher dimension by attaching this square to the top corner of an otherwise degenerate open cube in such a way that $p(- \wedge^* -)$ forms the back and right faces and the back right edge is p^*. This is depicted in the diagram below, where the bottom, front, and left faces of the open cube are a. We complete the proof with a two-extent composition, whose composite is shown as the shaded face in the diagram:

$$a \xlongequal{\hspace{3cm}} a$$

The diagram shows:

$p(i \wedge^* j)$ $p^*(j)$

$a \xrightarrow{\quad p(i) \quad} b$

$p(k)\uparrow$ $p(k \wedge^* j)$

$a \xlongequal{\hspace{2cm}} a$

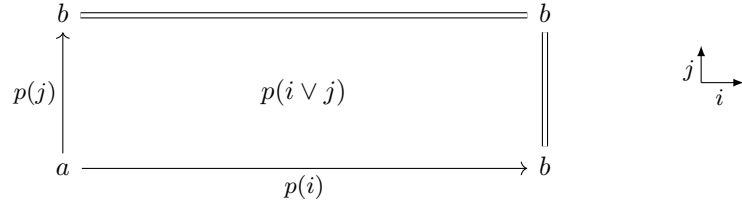$a \xlongequal{\hspace{3cm}} a$

with axes $j \ k \ i$

$\square$

Before moving on to the next connection, it is worth pointing out that the connections derived here could be slightly improved by attaching squares to the diagonal face of the open cubes of their compositions, thereby making the diagonal of the connections definitionally equal to p.

Lemma 4.1.2 (Join). *Given a type A and $a, b : A$, we have an operation:*

$$-(- \vee -) : \mathsf{path}_A(a, b) \to \mathbb{I} \to \mathbb{I} \to A$$

such that, for $p : \mathsf{path}_A(a, b)$ and $i, j : \mathbb{I}$, we have:

$$b \xlongequal{\hspace{4cm}} b$$

$p(j)\uparrow$ ... $p(i \vee j)$ ... $\|$

$a \xrightarrow{\quad p(i) \quad} b$

with axes $j \ i$

Proof. Using the halfway meet connection constructed as part of the previous proof, we perform a two-extent composition on an open cube given by halfway meets (front and left), degenerate squares formed from lines (back and right) and points (bottom). The composite gives us the desired square:

$$a \xlongequal{\hspace{3cm}} a$$

$p^*(j)$ $p^*(j)$

$b \xlongequal{\hspace{2cm}} b$

$p(k)\uparrow$ $\|$

$a \xrightarrow{\quad p(i) \quad} b$

$p^*(j)$

$a \xlongequal{\hspace{3cm}} a$

with axes $j \ k \ i$
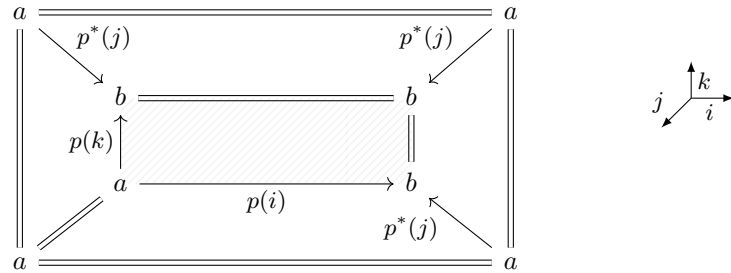
To check that this is a well-formed composition, note that we set a as the bottom (i, k)-face, $p(k \wedge^* j)$ as the left and $p^*(j)$ as the right (k, j)-faces, $p^*(j)$ as the back and $p(i \wedge^* j)$ as the front (k, j)-faces. Those squares are adjacent. $\square$

We hope that the reader is starting to get a feel for proofs by composition and the interplay between two-dimensional paths and squares at this point, so we may omit uses of path abstraction without further comment, and we may

also leave it up to the reader to show that the cap and tubes displayed in the composition diagrams respect the relevant adjacency conditions.

4.2 The groupoid laws

From the cubical perspective, path equality (homotopy) is always relative, since, viewed as squares, the only way to say that two lines are the same is modulo an identification of two other lines. By contrast, homotopy type theory [18] has a globular approach. But we can simulate globular identifications of paths by considering certain squares whose remaining faces are degenerate lines.
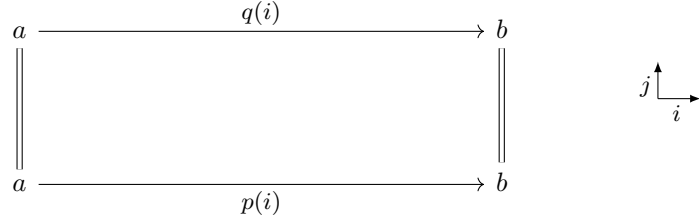
The groupoid laws are stated using this globular representation. Put differently, when we state that reflexivity is a unit for path inversion and concatenation, that inversion is involutive, and concatenation is associative, for example, we mean that there exists a globular identification between them.

Because the proof is simpler, we start by showing that reflexivity is a right and left unit for path concatenation in the next two lemmas.

Lemma 4.2.1. *For every A and every $a, b : A$ we have a path*

$$\mathsf{ru}_p : \mathsf{path}_{\mathsf{path}_A(a,b)}(p, p \cdot \mathsf{refl}_b)$$

for any $p : \mathsf{path}_A(a, b)$.

Proof. We need to construct an (i, j)-square having $p(i)$ and $(p \cdot \mathsf{refl}_b)(i)$ as i-lines and a and b as degenerate j-lines. But this already follows from the filler of concatenation defined in the proof of Lemma 3.2.3. $\square$

**Lemma 4.2.2.** *For every A and every $a, b : A$ we have a path*

$$\mathsf{lu}_p : \mathsf{path}_{\mathsf{path}_A(a,b)}(p, \mathsf{refl}_a \cdot p)$$

for any $p : \mathsf{path}_A(a, b)$.

Proof. By composition, we define a helper (i, j)-square that goes from $p^{-1}(i)$ to b in the i-direction and from b to $p(j)$ in the j-direction. The composition uses the filler of the path inversion of p (front), meet (right), and degenerate squares formed from lines (back and left) and points (bottom). For future reference, we shall call it γ:
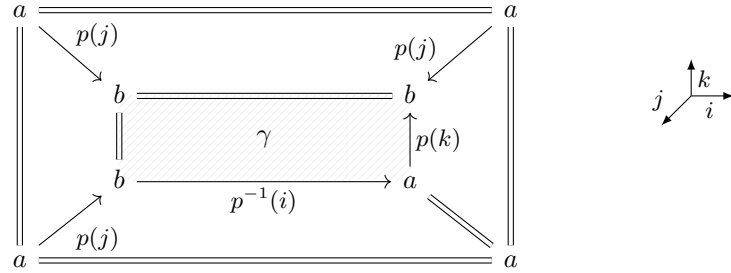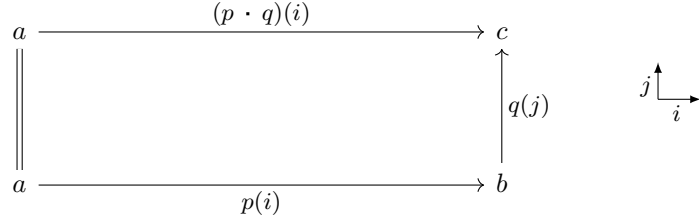
Forming a new open cube, we set γ at the right, the filler of the concatenation of refl_a and p at the back, the filler of the inversion of p at the bottom, and degenerate squares at the other faces.

$\square$
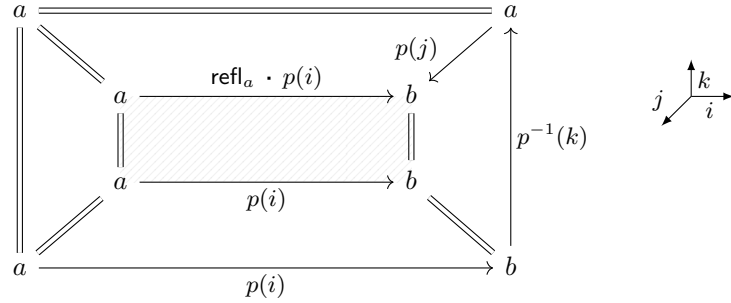
Why is the unit property so much simpler to demonstrate in the right? If we look attentively at the filler of path concatenation (Lemma 3.2.3), for example,

we can see that it forms a simultaneous identification that can be pronounced "let there be a path from p to $p \cdot q$ just in case q is refl_a". Consequently, if we set $q :\equiv \mathsf{refl}_b$, we immediately have a globular path from p to $p \cdot q$. We can thus compare path concatenation with the transitivity operation defined in the homotopy type theory book by path induction on the second argument [18, lem.2.1.2]. The same idea applies to path inversion, "let there be a path from p^{-1} to refl_a just in case p is refl_a", so it is related to the symmetry operation defined by path induction on p [18, lem.2.1.1].

Next we prove that path inversion indeed is a right and left inverse with respect to concatenation:

Lemma 4.2.3. *For every A and every $a, b : A$ we have a path*

$$\mathsf{rc}_p : \mathsf{path}_{\mathsf{path}_A(a,a)}(\mathsf{refl}_a, p \cdot p^{-1})$$

for any $p : \mathsf{path}_A(a, b)$.

Proof. By composition, we must construct a cube whose composite is an (i,k)-square with $\mathsf{refl}_a(i)$ and $p \cdot p^{-1}(i)$ as i-lines and a in both degenerate k-lines. Now consider the following open (i,j,k)-cube
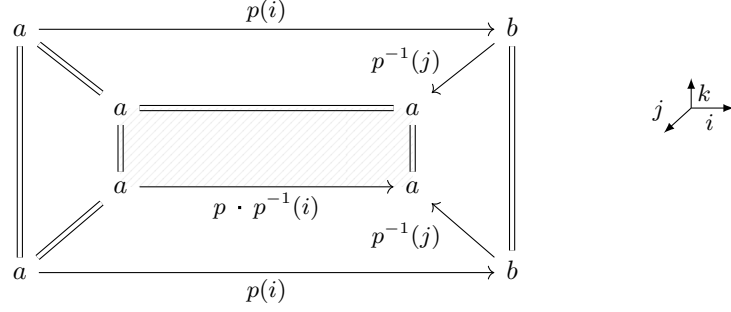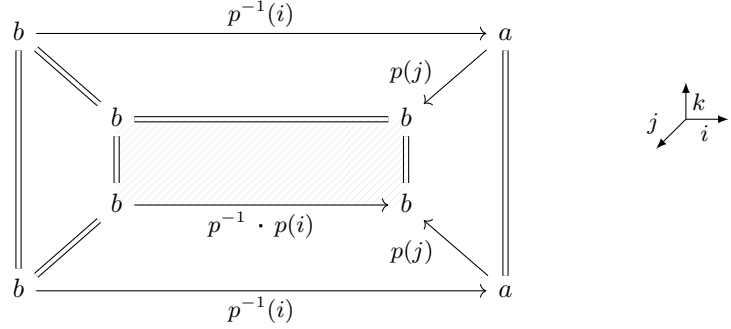

whose bottom, left and right faces are degenerate squares, and back and front squares are respectively the fillers for path inversion and concatenation. $\square$

Lemma 4.2.4. *For every A and every $a,b : A$ we have a path*

$$\mathsf{lc}_p : \mathsf{path}_{\mathsf{path}_A(b,b)}(\mathsf{refl}_b, p^{-1} \cdot p)$$

for any $p : \mathsf{path}_A(a,b)$.

Proof. By composition on the following open cube, whose back face is the γ square defined in the proof of Lemma 4.2.2.
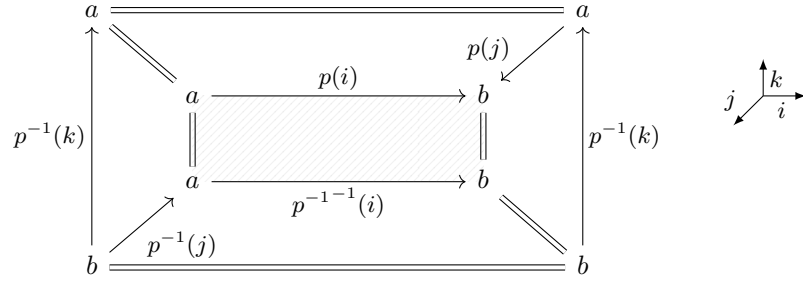

$\square$

The following lemma states that path inversion is involutive:

Lemma 4.2.5. *For every A and every $a,b : A$, we have a path*

$$\mathsf{inv}_p : \mathsf{path}_{\mathsf{path}_A(a,b)}(p, (p^{-1})^{-1})$$

for any $p : \mathsf{path}_A(a,b)$.

Proof. The proof follows by the use of meets, joins, and γ to form the composite:

$\square$

Last but not least, we want to show that path concatenation is associative.

**Lemma 4.2.6.** *For every A and every $a, b, c, d : A$, we have a path*

$$\mathsf{assoc}_{p,q,r} : \mathsf{path}_{\mathsf{path}_A(a,d)}((p \cdot q) \cdot r, p \cdot (q \cdot r))$$

for any $p : \mathsf{path}_A(a,b)$, $q : \mathsf{path}_A(b,c)$, $r : \mathsf{path}_A(c,d)$.

Proof. We use the fact that, for any two squares with the same three faces, there is a square showing that the fourth sides are equal. In particular, given the following two squares with definitionally equal bottom, right, and left faces
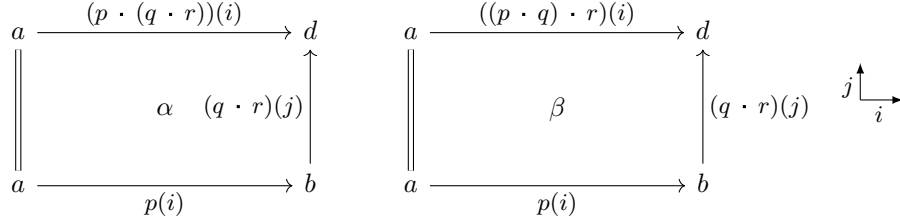
we can construct the desired path homotopy
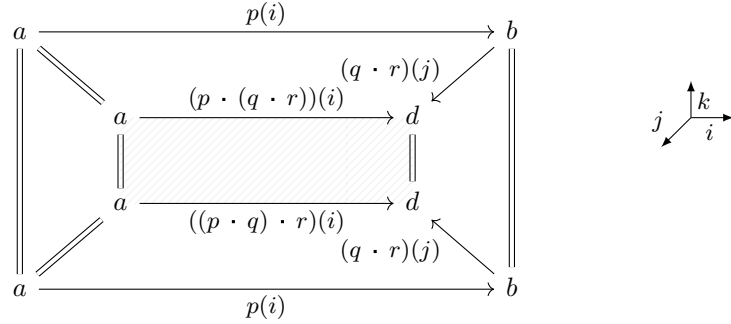
Since α is just the filler of path concatenation $p \cdot (q \cdot r)$, it remains to construct β. But this is easy, because looking at the top and right sides of this square, the filler of path concatenation gives us a canonical construction:
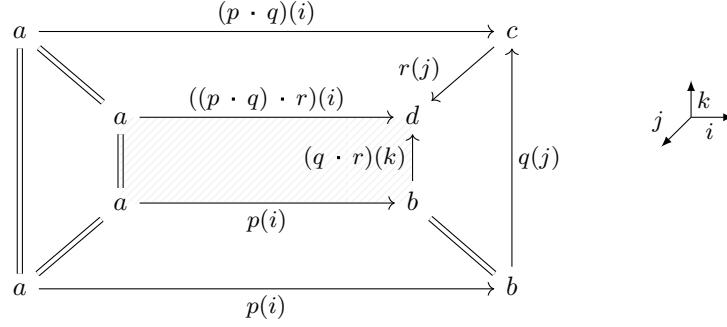
What about the higher groupoid structure of types? What we have shown in this section is that types have a 1-groupoid structure, but since those laws do not hold "on the nose" as definitional equations, they must satisfy some equations of their own. We will not cover this here. Instead we give a proof of path induction in §6.1, which corresponds almost exactly to the elimination rule for the inductive path type in homotopy type theory [18] (except that the computation rule here only holds up to a path), and conjecture that it should be possible to derive them with an approach similar to [16].

5   Dependent paths

The various operations and laws defined in the previous section all share a fundamental limitation: they are only applicable to a specific class of paths. This restriction is imposed by our implicit requirement that a path be non-dependent, which means that the type $\mathsf{path}_A(a, b)$ is not well-formed unless a and b have exactly the type A. While this restriction is important when trying to understand cubical methods, it rules out the formation of paths in paths between types $A : \mathbb{I} \to \mathcal{U}$, which, for the lack of a better name, we shall call *type lines*, types that may change depending on their endpoints.

5.1   The dependent path type

Given a type line $A : \mathbb{I} \to \mathcal{U}$ and terms $a : A(0)$ and $b : A(1)$, the type of dependent paths from a to b is written $\mathsf{pathd}_A(a, b)$. If the underlying type line is a constant function, that is, the reflexivity path, then the dependent path type is the non-dependent path type

$$\mathsf{pathd}_{\lambda\_.A}(a, b) \equiv \mathsf{path}_A(a, b).$$

The only reason why this equality holds is because non-dependent paths are actually defined in terms of dependent paths. So the rules for the dependent path type arise as straightforward generalizations of the ones stated before for path types in virtually the same way the dependent function type generalizes the non-dependent function type. For instance, as with non-dependent paths, we eliminate a term of this type by application, $p(i) : A(i)$, where $p : \mathsf{pathd}_A(a, b)$ and $i : \mathbb{I}$. The constructor of this type is path abstraction, and we say that $\lambda i.a : \mathsf{pathd}_A(a[0/i], a[1/i])$ if $a : A(i)$ assuming that $i : \mathbb{I}$, where $a[0/i] : A(0)$

and $a[1/i] : A(1)$. Moreover, we also require equalities that are no different from the ones for the ordinary path type. In particular, given $p : \mathsf{pathd}_A(a, b)$ and $i : \mathbb{I}$, we have boundary rules $p(0) \equiv a : A(0)$ and $p(1) \equiv b : A(1)$, and the uniqueness rule $\lambda i.(p(i)) \equiv p$ when i does not occur in p.

We need a type for dependent paths in the cubical setting because in their most general form paths connect terms inhabiting the endpoints of a type line. Consider for example the meet operator defined in Lemma 4.1.1. What, exactly, is its type? Starting with an iterated function type

$$\lambda j.\lambda i.p(i \wedge j) : \mathbb{I} \to \mathbb{I} \to A$$

we can obtain a path in the type of paths from a to $p(j)$

$$\lambda j.\lambda i.p(i \wedge j) : \prod_{(j:\mathbb{I})} \mathsf{path}_A(a, p(j)),$$

because, given a fixed j, $\lambda i.p(i \wedge j)$ varies from the point a to $p(j)$. By repeating the process, we obtain a dependent path from refl_a to p in the type of paths from a to $p(j)$

$$\lambda j.\lambda i.p(i \wedge j) : \mathsf{pathd}_{\lambda i.\mathsf{path}_A(a, p(i))}(\mathsf{refl}_a, p)$$

since $\lambda j.\lambda i.p(i \wedge j)$ goes from

$$\lambda i.p(i \wedge 0) \equiv \lambda i.a$$

to

$$\lambda i.p(i \wedge 1) \equiv \lambda i.p(i)$$
$$\equiv p.$$

In homotopy type theory [18] a type for dependent paths is definable using the non-dependent and inductively defined path type. The advantage of the cubical approach to dependent paths is that they are naturally built-in. It is also worth noting that a type for dependent paths can be defined as multi-dimensional extension types instead [2, p.67], a generalization of path types that avoid the intricacies that are innate to the use of iterative path types.

5.2   Heterogeneous composition

As soon as we start considering dependent paths it becomes clear that the path inversion and concatenation operations defined in §3.2 are not general enough. Let us consider the limitations of non-dependent path inversion first. Assuming that $\mathsf{pathd}_A(a, b)$ is a type, where $a : A(0)$ and $b : A(1)$, the type $\mathsf{pathd}_A(b, a)$ of inverse paths from b to a will *not* be well-formed unless it is also the case that $b : A(0)$ and $a : A(1)$. For non-dependent paths this condition turns out to be trivially satisfied since the relevant type line is a constant function, meaning that $A(0) \equiv A \equiv A(1)$. But as this is not true in general, the operation defined in Lemma 3.2.1 fails to produce inverses for dependent paths.

Fortunately, there is a way to deal with this problem using type universes. Suppose we are given a type line $A : \mathbb{I} \to \mathcal{U}$, that is, a non-dependent path from the type $A(0)$ to $A(1)$ in the universe $\mathcal{U}$,

$$A : \mathsf{path}_{\mathcal{U}}(A(0), A(1)).$$

We can assume that this is a non-dependent path because type universes never depend on interval variables, so we have both $A(0), A(1) : \mathcal{U}$.

Now it can be shown by Lemma 3.2.1 that the following inverse exists:

$$
\begin{array}{ccc}
A(1) & \xrightarrow{\quad A^{-1}(i) \quad} & A(0) \\
\Big\uparrow{\scriptstyle A(j)} & & \Big\| \\
A(0) & =\!=\!=\!=\!=\!=\!=\!= & A(0)
\end{array}
\qquad
\begin{array}{c}
j\;\llcorner\!\rightarrow \\
\quad i
\end{array}
$$

In particular, we have
$$A^{-1} : \mathsf{path}_{\mathcal{U}}(A(1), A(0)),$$
which corresponds precisely to the "inverse" type of A, for the initial and terminal points of A^{-1} are respectively $A(1)$ and $A(0)$. Put differently, we have two inferences that hold top/bottom and bottom/top,

$$\frac{a : A^{-1}(1)}{a : A(0)} \quad \text{and} \quad \frac{a : A^{-1}(0)}{a : A(1)}.$$

Thus, assuming that $\mathsf{pathd}_A(a, b)$ is a type, it is now easy to see that $\mathsf{pathd}_{A^{-1}}(b, a)$ will always be a type as well, regardless of whether A is a degenerate line or not: if $\mathsf{pathd}_A(a, b)$ is a type then we have $a : A(0)$ and $b : A(1)$, meaning that $a : A^{-1}(1)$ and $b : A^{-1}(0)$ must be the case.

This motivates the definition of a dependent path inversion operation:

Lemma 5.2.1. *For every type $A : \mathbb{I} \to \mathcal{U}$ and every $a : A(0)$ and $b : A(1)$, there is a function*
$$\mathsf{pathd}_A(a, b) \to \mathsf{pathd}_{A^{-1}}(b, a)$$
called the (dependent) inverse path and denoted $p \mapsto p^{-1}$.

Proof. Dependent operations are typically introduced by repeating, mutatis mutandis, the proofs of their non-dependent counterparts. So, the results which were previously proven by composition will now follow from a "heterogeneous" variant of composition based on the same open cubes. This adaptation is required because we are dealing with non-degenerate type lines and dependent paths, and, for a valid composition, all the faces of an open cube, that is, all the terms of the composition scenario, must have the same type.

More concretely, the open box used in the definition of non-dependent path inversion (the one used in the proof of Lemma 3.2.1) is ill-formed in this context since $p(j) : A(j)$ and $a : A(0)$ are lines with different types:

$$
\begin{array}{ccc}
b & & a \\
\Big\uparrow{\scriptstyle p(j)} & & \Big\| \\
a & =\!=\!=\!=\!=\!=\!=\!= & a
\end{array}
\qquad
\begin{array}{c}
j\;\llcorner\!\rightarrow \\
\quad i
\end{array}
$$

The solution is not to form an open box living in A, but rather to perform a composition taking place in A^{-1}, using transport on the type line $\lambda j.\mathsf{fill}_j(A^{-1}(i))$, the filler of the inverse of A, to adjust the type of the faces of open box.

For the cap, we want to transport a from 0 to 1. This gives us the correct type because $\mathsf{fill}_1(A^{-1}(i)) \equiv A^{-1}(i)$. Now we observe that $a : \mathsf{fill}_0(A^{-1}(i)) \equiv A(0)$, so we have

$$a^{0 \rightsquigarrow 1}_{\lambda j.\mathsf{fill}_j(A^{-1}(i))} : A^{-1}(i).$$

Now for the left tube, i.e. assuming that $i = 0$, we transport p from j to 1. More specifically, since $p(j) : \mathsf{fill}_j(A^{-1}(0)) \equiv A(j)$, and, again, $\mathsf{fill}_1(A^{-1}(0)) \equiv A^{-1}(0)$, we have

$$(p(j))^{j \rightsquigarrow 1}_{\lambda j.\mathsf{fill}_j(A^{-1}(0))} : A^{-1}(0).$$

The right tube is treated similarly, except that we transport a (and $i = 1$),

$$a^{j \rightsquigarrow 1}_{\lambda j.\mathsf{fill}_j(A^{-1}(1))} : A^{-1}(1).$$

It is not hard to show that those lines are adjacent. We want to form a composition from 0 to 1, so the initial point of the left tube should match the initial point of the cap,

$$
\begin{aligned}
(\lambda j.(p(j))^{j \rightsquigarrow 1}_{\lambda j.\mathsf{fill}_j(A^{-1}(0))})(0) &\equiv (p(0))^{0 \rightsquigarrow 1}_{\lambda j.\mathsf{fill}_j(A^{-1}(0))} \\
&\equiv a^{0 \rightsquigarrow 1}_{\lambda j.\mathsf{fill}_j(A^{-1}(0))} \\
&\equiv a^{0 \rightsquigarrow 1}_{\lambda j.\mathsf{fill}_j(A^{-1}(i))}
\end{aligned}
$$

and the initial point of the right tube should be the terminal point of the cap,

$$
\begin{aligned}
(\lambda j.a^{j \rightsquigarrow 1}_{\lambda j.\mathsf{fill}_j(A^{-1}(1))})(0) &\equiv a^{0 \rightsquigarrow 1}_{\lambda j.\mathsf{fill}_j(A^{-1}(1))} \\
&\equiv a^{0 \rightsquigarrow 1}_{\lambda j.\mathsf{fill}_j(A^{-1}(i))}.
\end{aligned}
$$

By composition, we have a line from b to a in $A^{-1}(i)$. This composite has the correct endpoints because

$$(p(1))^{1 \rightsquigarrow 1}_{\lambda j.\mathsf{fill}_j(A^{-1}(0))} \equiv p(1) \equiv b$$

and

$$a^{1 \rightsquigarrow 1}_{\lambda j.\mathsf{fill}_j(A^{-1}(1))} \equiv a.$$

$\square$

It is important to stress that the proof of the previous lemma is based on *heterogeneous composition*, a particular kind of composition in which the types of the cap and composite may differ. As indicated in the proof above, a heterogeneous composition can be obtained from composition and transport. For instance, given a type line $A : \mathbb{I} \to \mathcal{U}$, a heterogeneous composition in A with cap $a : \mathbb{I} \to A(0)$ and tubes $a_0, a_1 : \prod_{(i:\mathbb{I})} A(i)$ is just an abbreviation for a compound composition which combines the two Kan operations into one, a composition with cap $\lambda i.(a(i))^{0 \rightsquigarrow 1}_A : \mathbb{I} \to A(1)$ and tubes $\lambda j.(a_0(j))^{j \rightsquigarrow 1}_A : \mathbb{I} \to A(1)$ and $\lambda j.(a_1(j))^{j \rightsquigarrow 1}_A : \mathbb{I} \to A(1)$. The composite is a term of type $\mathbb{I} \to A(1)$.

So we have already covered the case of path inversion and showed how we can invert dependent paths using heterogeneous composition. One natural question

is whether a similar strategy can be used to concatenate dependent paths. It is easier to see the limitations of the non-dependent operation of path concatenation when we consider two-dimensional paths such as

$$\alpha : \mathsf{pathd}_{\lambda j.\mathsf{path}_A(s(j),t(j))}(p,q) \qquad \text{and} \qquad \beta : \mathsf{pathd}_{\lambda j.\mathsf{path}_A(u(j),v(j))}(q,r),$$

which correspond to two (i,j)-squares in A such as the following:

It is clear that we should be able to concatenate the paths α and β vertically, using the bottom and top faces of the squares, if we concatenate their left and right faces using non-dependent path concatenation. This should give us a j-dependent path from p to r in the type of paths from $s \cdot u(j)$ to $t \cdot v(j)$ in A. This construction is illustrated in the following composition, whose left and right squares can be easily obtained by composition using the fillers of path concatenation and inversion, and meets and joins:

This two-dimensional concatenation is not an instance of the non-dependent operation derived in Lemma 3.2.3: from our definition, it follows that to form the concatenated path $\alpha \cdot \beta$, first α and β must be terms of the same type and that type must be a degenerate line. In this example, the concatenation fails for both reasons, for the equality $\mathsf{path}_A(s(j),t(j)) \equiv \mathsf{path}_A(u(j),v(j))$ need *not* be the case and those types may depend on the interval variable j.

To overcome this problem we need a dependent path concatenation operation. Following the definition of dependent path inversion, we consider the non-dependent path concatenation of line types first. Suppose that $A, B : \mathbb{I} \to \mathcal{U}$ such that $A(1) \equiv B(0)$. The diagram

$$A(0) \cdots\cdots\cdots\xrightarrow{(A \cdot B)(i)}\cdots\cdots\cdots\rightarrow B(1)$$

with vertical arrows: left side $A(0) = A(0)$ (double line), right side $B(j)$ from $A(1) \equiv B(0)$ up to $B(1)$, bottom $A(0) \xrightarrow{A(i)} A(1) \equiv B(0)$, and axes labeled j, i.

illustrates the path concatenation of A and B,

$$A \cdot B \; : \; \mathsf{path}_{\mathcal{U}}(A(0), B(1)).$$

We also have two important inferences that hold top/bottom and bottom/top,

$$\frac{a : (A \cdot B)(0)}{a : A(0)} \quad \text{and} \quad \frac{b : (A \cdot B)(1)}{b : B(1)}.$$

With this, we have all we need to define concatenation for dependent paths:

Lemma 5.2.2. *Suppose that $A, B : \mathbb{I} \to \mathcal{U}$ such that $A(1) \equiv B(0)$. Given any $a : A(0)$, $b : A(1)$ and $c : B(1)$, there is a function*

$$\mathsf{pathd}_A(a,b) \to \mathsf{pathd}_B(b,c) \to \mathsf{pathd}_{A \cdot B}(a,c)$$

written $p \mapsto q \mapsto p \cdot q$ and called the dependent path concatenation function.

Proof. By heterogeneous composition on the open box from Lemma 3.2.3. $\qquad\square$

Dependent path concatenation allows us to concatenate α and β from our example above, but it is worth noting that the resulting path need not be definitionally equal to the operation $\alpha?\beta$ we described. However, it is easy to show by path induction that they are equal up to globular identification.

5.3   The groupoid laws for dependent paths

Now that we are in possession of a new pair of dependent path inverse and concatenation operations, it is possible for us to revisit the groupoid structure from §4.2 and derive more general laws that hold for dependent paths as well. It is helpful to understand how this generalization works with an example.

As a case in point, let us examine the involution law from Lemma 4.2.5. This law actually expresses a fact about paths in degenerate line types. To be exact, given a degenerate $A : \mathbb{I} \to \mathcal{U}$, it states that, for every $a : A(0)$, $b : A(1)$ and $p : \mathsf{pathd}_A(a,b)$, there is a path

$$\mathsf{pathd}_{\lambda \_.\mathsf{pathd}_A(a,b)}((p^{-1})^{-1}, p).$$

If we were to drop the restriction that the type line A be degenerate, meaning that the law would then be founded on dependent path inversion instead, the desired type would have to be stated as something like

$$\mathsf{pathd}_{\lambda j.\mathsf{pathd}_{\lambda i.?(j)(i)}(a,b)}((p^{-1})^{-1}, p).$$

Because we are dealing with dependent paths and their operations, which often result in paths living in a different type line, we run into a problem at this

point: we need a path from $\left(p^{-1}\right)^{-1}$: $\mathsf{pathd}_{(A^{-1})^{-1}}(a,b)$ to p : $\mathsf{pathd}_A(a,b)$, and we must specify in what type line this path lives in. Now, considering that generally $(A^{-1})^{-1} \not\equiv A$, this type line cannot be degenerate. But recall that, by the non-dependent involution law from Lemma 4.2.5, there is a path

$$\mathsf{inv}_A : \mathsf{path}_{\mathsf{path}_{\mathcal{U}}(A(0),A(1))}\left(\left(A^{-1}\right)^{-1}, A\right),$$

so the law we are looking for can be stated as follows:

Lemma 5.3.1. *For every $A : \mathbb{I} \to \mathcal{U}$ with $a : A(0)$ and $b : A(1)$, we have*

$$\mathsf{pathd}_{\lambda j.\mathsf{pathd}_{\lambda i.\mathsf{inv}_A(j)(i)}(a,b)}\left(\left(p^{-1}\right)^{-1}, p\right)$$

for any dependent path $p : \mathsf{pathd}_A(a,b)$.

The proof argument is similar to those given in the definition of dependent path inversion and composition from Lemmas 5.2.1 and 5.2.2, respectively. It thus follows from a straightforward heterogeneous composition on the open cube constructed for the proof of the non-dependent version of the involution law from Lemma 4.2.5. In fact, all dependent counterparts of the laws proven in §4.2 follow the same pattern: they can all be stated by using their non-dependent counterparts and proven by a heterogeneous filling of their open cubes. For this reason we will simply omit those results.

6   Notable properties of paths

Before concluding our naive presentation of cubical type theory, we would like to explore two notable properties of paths from a cubical perspective: path induction and the fact that path concatenation operation on the second loop space is commutative.

6.1   Path induction

We now present a derivation of path induction [18, §1.12.1], a key property that informally states that paths with a free endpoint can be deformed and retracted without changing their essential characteristics. As mentioned before, path induction serves as the elimination rule for the inductive path type in homotopy type theory, where it is taken as primitive. The result considered here corresponds to what is sometimes called based path induction [18]:

Theorem 6.1.1 (Path induction)**.** *Given a type $A : \mathcal{U}$, a term $a : A$ and a type family $C : \prod_{(x:A)} \mathsf{path}_A(a,x) \to \mathcal{U}$, we have a function*

$$\mathsf{pathrec} : \prod_{(x:A)} \prod_{(p:\mathsf{path}_A(a,x))} \prod_{(c:C(a,\mathsf{refl}_a))} C(x,p).$$

Proof. We want to construct, for every $x : A$, $p : \mathsf{path}_A(a,x)$ and $c : C(a, \mathsf{refl}_a)$, a term of type $C(x,p)$. To obtain such a term, we shall transport c along a type line $D : \mathbb{I} \to \mathcal{U}$ that goes from $D(0) :\equiv C(a, \mathsf{refl}_a)$ to $D(1) :\equiv C(x,p)$. For the construction of the desired type line D it suffices to consider the halfway meet connection derived in our proof of Lemma 4.1.1:

Now it suffices to define our type line as:

$$D :\equiv \lambda i.C(p(i \wedge^* 1), \lambda j.p(i \wedge^* j)) : \mathbb{I} \to \mathcal{U}$$

because it goes from

$$
\begin{aligned}
D(0) &\equiv (\lambda i.C(p(i \wedge^* 1), \lambda j.p(i \wedge^* j)))0 \\
&\equiv C(p(0 \wedge^* 1), \lambda j.p(0 \wedge^* j)) \\
&\equiv C(a, \mathsf{refl}_a)
\end{aligned}
$$

to

$$
\begin{aligned}
D(1) &\equiv (\lambda i.C(p(i \wedge^* 1), \lambda j.p(i \wedge^* j)))1 \\
&\equiv C(p(1 \wedge^* 1), \lambda j.p(1 \wedge^* j)) \\
&\equiv C(x, \lambda j.p(j)) \\
&\equiv C(x, p).
\end{aligned}
$$

To complete the proof we transport $c : D(0)$ from 0 to 1.

$\square$

Note that, in the above proof, the η-rule for the path type discussed in §2.1, that is, the requirement that $p \equiv \lambda j.(p(j))$, is crucial to the correct specification of endpoints of the type line we are doing the transportation over. Without this rule, it would not be possible to show that $D(1) \equiv C(x, p)$ and the transportation would give us a term of the wrong type.

In cubical type theory, the computation rule for path induction does not hold "on the nose" like in the case of the eliminator of the inductive path type [18]. Put differently, for a fixed type family $C : \prod_{(x:A)} \mathsf{path}_A(a, x) \to \mathcal{U}$, given $a : A$ and $c : C(a, \mathsf{refl}_a)$, in general,

$$\mathsf{pathrec}(a, \mathsf{refl}_a, c) \not\equiv c.$$

But this equality can be shown to hold up to a path:

Lemma 6.1.2 (Path computation [2])**.** *For every $a : A$ and $c : C(a, \mathsf{refl}_a)$, we have a path of type*

$$\mathsf{path}_{C(a,\mathsf{refl}_a)}(\mathsf{pathrec}(a, \mathsf{refl}_a, c), c).$$

Proof. By the definition of path induction, we have a definitional equality

$$\mathsf{pathrec}(a, \mathsf{refl}_a, c) \equiv c^{0 \rightsquigarrow 1}_{\lambda i.C(\mathsf{refl}_a(i \wedge^* 1), \lambda j.\mathsf{refl}_a(i \wedge^* j))}.$$

We are to find a path from this transported term to c in the type $C(a, \mathsf{refl}_a)$. First, we note that the transportation induces a path

$$\lambda i.c^{i \rightsquigarrow 1}_{\lambda i.C(\mathsf{refl}_a(i \wedge^* 1), \lambda j.\mathsf{refl}_a(i \wedge^* j))}.$$

This path has the right endpoints, since the static transportation from 1 to 1 has no effect, being therefore definitionally equal to c. But we need a path in the type $C(a, \mathsf{refl}_a)$ and when we consider the type line the transportation in the constructed path occurs along, it can be seen that this induced path is actually in the type $C(\mathsf{refl}_a(i \wedge^* 1), \lambda j.\mathsf{refl}_a(i \wedge^* j))$. This is not the type $C(a, \mathsf{refl}_a)$ we are looking for because $a \not\equiv \mathsf{refl}_a(i \wedge^* 1)$ and $\mathsf{refl}_a \not\equiv \lambda j.\mathsf{refl}_a(i \wedge^* j)$.

To conclude this proof we will fix this type mismatch with a heterogeneous composition based on the following composition scenario, where the composite is intended to give us the desired path in the type $C(a, \mathsf{refl}_a)$:

However, before this can be viewed as a valid heterogeneous composition, we must specify in what type line the operation takes place. We therefore have to define a $D_i : \mathbb{I} \to \mathcal{U}$ containing the cap and composite as terms inhabiting its initial and terminal endpoints. More explicitly, we need to satisfy the endpoint conditions that $D_i(0) \equiv C(a, \mathsf{refl}_a) \equiv D_i(1)$. In the i-direction, however, we need $D_0 \equiv \lambda j.C(\mathsf{refl}_a(j \wedge^* 1), \lambda k.\mathsf{refl}_a(j \wedge^* k))$ and $D_1 \equiv \lambda j.C(a, \mathsf{refl}_a)$.

First we construct by composition an otherwise degenerated square with $\mathsf{refl}_a(k \wedge^* 1)$ as the right face. The composition scenario consists of degenerate squares in the bottom, front, back, and left and $\mathsf{refl}_a(k \wedge^* 1)$ in the right:

Finally, we set $D_i :\equiv \lambda k.C(\alpha(k)(i), \lambda j.\mathsf{fill}_j(\alpha(k)(i)))$. Now, to check that the type line D_i has the correct endpoints we observe that for either $\epsilon = 0$ or $\epsilon = 1$:

$$
\begin{aligned}
D_i(\epsilon) &\equiv (\lambda i.C(\alpha(k)(i), \lambda j.\mathsf{fill}_j(\alpha(k)(i))))(\epsilon) \\
&\equiv C(\alpha(\epsilon)(i), \lambda j.\mathsf{fill}_j(\alpha(\epsilon)(i))) \\
&\equiv C(a, \lambda j.a).
\end{aligned}
$$

In the i-direction, for $i = 0$ we have

$$D_0 \equiv \lambda k.C(\alpha(k)(0), \lambda j.\mathsf{fill}_j(\alpha(k)(0)))$$
$$\equiv \lambda k.C(\mathsf{refl}_a(k \wedge^* 1), \lambda j.\mathsf{refl}_a(k \wedge^* j)),$$

and for $i = 1$

$$D_1 \equiv \lambda k.C(\alpha(k)(1), \lambda j.\mathsf{fill}_j(\alpha(k)(1)))$$
$$\equiv \lambda k.C(a, \lambda j.a).$$

$\square$

The proof above is inspired by an argument given by Angiuli [2, pp.54–56], who considered a slightly different definition of path induction based on the folklore result that $\sum_{(x:A)} \mathsf{path}_A(a, x)$ is contractible with center $\langle a, \mathsf{refl}_a \rangle$ and that type families respect paths in their indexing type.

In the presence of higher inductive types [10, 13], the inductive path type from homotopy type theory [18] can be recovered as a higher inductive type freely generated by reflexivity [10]. Path induction then acts strictly on reflexivity, since it is given as a specific generator that can be recognized by the eliminator, and this allows for a strict computation rule. This inductive path type can be shown to be equivalent to the path type [10], meaning that, by univalence, the inductive path type is the path type up to a path.

6.2   The Eckmann–Hilton argument

In homotopy type theory, paths from a point to itself are called loops. Thus, given a type A and a point $a : A$, the loop space $\Omega(A, a)$ is defined to be the type of loops $\mathsf{path}_A(a, a)$. When the loop space of a loop space is considered, we have the second loop space $\Omega^2(A, a)$, which is the type $\mathsf{path}_{\mathsf{path}_A(a,a)}(\mathsf{refl}_a, \mathsf{refl}_a)$.

The Eckmann–Hilton argument is one interesting result about the second loop space that states that path concatenation is commutative. It is inspired by a classical result in homotopy theory and proven by path induction in homotopy type theory [18, thm.2.1.6]. We adapt this proof in what follows:

Theorem 6.2.1 (Eckmann–Hilton). *For any $\alpha, \beta : \Omega^2(A, a)$, there is a path*

$$\mathsf{path}_{\Omega^2(A,a)}(\alpha \cdot \beta, \beta \cdot \alpha)$$

Proof. It is easier to prove a stronger statement that holds more generally for any two-dimensional globular paths by path induction, and then derive the intended claim as a special case. So, given paths $\alpha : \mathsf{path}_{\mathsf{path}_A(a,b)}(p, q)$ and $\beta : \mathsf{path}_{\mathsf{path}_A(b,c)}(r, s)$, first we define a right whiskering operation

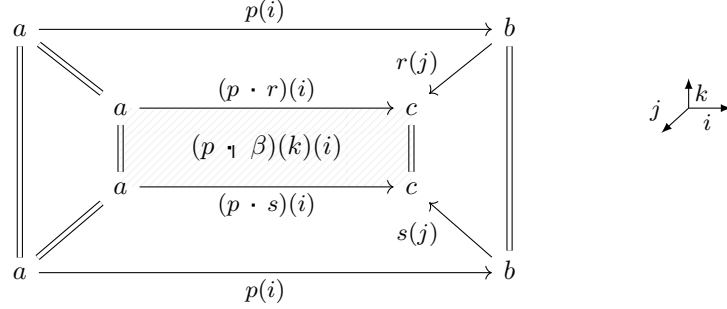$$\alpha \cdot_{\mathsf{r}} r : \mathsf{path}_{\mathsf{path}_A(a,c)}(p \cdot r, q \cdot r)$$

in the obvious way such that the following composition holds, with α at the bottom:
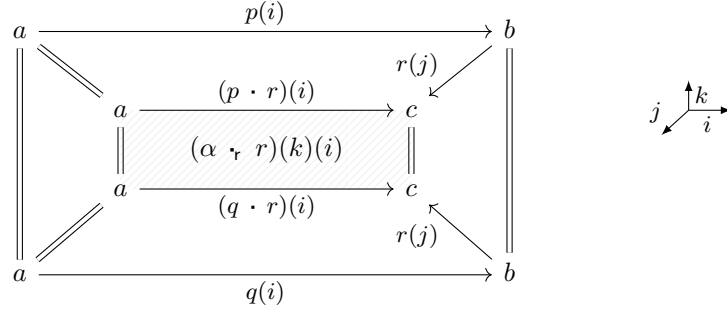
then we define left whiskering

$$p \mathbin{\rceil} \beta : \mathsf{path}_{\mathsf{path}_A(a,c)}(p \cdot r, p \cdot s)$$

by composition on a similar open cube but with β at the right:

It is easy to see by path induction on α, β, p, and r that there exists a path

$$\mathsf{path}_{\mathsf{path}_{\mathsf{path}_A(a,c)}(p \cdot r, q \cdot s)}((\alpha \cdot {}_r r) \cdot (q \cdot {}_l \beta), (p \cdot {}_l \beta) \cdot (\alpha \cdot {}_r s)).$$

Now let $p \equiv q \equiv r \equiv s \equiv \mathsf{refl}_a$. Since reflexivity is both a right and left unit for path concatenation (as shown in Lemmas 4.2.1 and 4.2.2), the above proposition demonstrates our intended claim.

□

The Eckmann–Hilton argument has been proved very recently for De Morgan cubes in a purely cubical way that avoids path induction [9]. The key element that makes this proof possible is a transportation from 0 to 1 along a j-type line of paths from $\alpha_r(j) \cdot \beta_l(j)$ to $\beta_l(j) \cdot \alpha_r(j)$ in $\Omega(A, a)$, constructed using function application, path concatenation, and the right and left unit, where

$$\alpha_r(j) :\equiv \mathsf{ap}_{\lambda p.\mathsf{ru}_p^{-1}(j)}(\alpha) : \mathsf{path}_{\Omega(A,a)}(\mathsf{ru}_{\mathsf{refl}_a}^{-1}(j), \mathsf{ru}_{\mathsf{refl}_a}^{-1}(j))$$

$$\beta_l(j) :\equiv \mathsf{ap}_{\lambda p.\mathsf{lu}_p^{-1}(j)}(\beta) : \mathsf{path}_{\Omega(A,a)}(\mathsf{lu}_{\mathsf{refl}_a}^{-1}(j), \mathsf{lu}_{\mathsf{refl}_a}^{-1}(j)).$$

This type is actually well-formed for De Morgan cubes because it can be shown that $\mathsf{ru}_{\mathsf{refl}_x} \equiv \mathsf{lu}_{\mathsf{refl}_x}$, for all $x : A$, using proofs of the right and left unit laws based on essentially the same compositions, except that they make use of the built-in connections that act strongly on reflexivity terms. This definitional

equality, however, does not hold for cartesian cubes and their weak connections, so a direct cubical proof would have to rely on a different strategy.

7 Directions for future work

There is much to be done yet in order to provide a cubical alternative to the informal type theory project of the homotopy type theory book [18]. We view this paper as opening up many possibilities for future work, including informal cubical accounts of the higher groupoid structure of type formers, univalence, higher inductive types, homotopy n-types, and the development of mathematics such as homotopy theory, category theory, or set theory.

Part of the proofs contained in this paper have been formalized in the proof assistants `Cubical Agda` [19] and `redtt` [17], and are available online.[3]

Notes

[1] In homotopy type theory, the type of paths is usually referred to as the identity type and its inhabitants are treated as witnesses that two objects are equal. We will avoid this terminology here given that it is open to question whether the highly technical conception of a path in higher-dimensional type theory is indeed capable of reflecting ordinary mathematical equality [15, 7]. Because the identity type is inductively defined in homotopy type theory, we shall sometimes refer to it as the "inductive path type" throughout the text.

[2] The author wishes to thank Loïc Pujet for this suggestion.

[3] See e.g. `https://github.com/RedPRL/redtt/blob/master/library/prelude/path.red` for some basic constructions for paths in the cartesian style, including function extensionality and function application, path inversion and concatenation, and path induction and its computation rule. For proofs of the groupoid laws using De Morgan cubes, see also `https://github.com/agda/cubical/blob/master/Cubical/Foundations/GroupoidLaws.agda`.

References

[1] Thorsten Altenkirch. Naïve type theory. In *Reflections on the Foundations of Mathematics*, pages 101–136. Springer, 2019. URL: `https://doi.org/10.1007/978-3-030-15655-8_5`.

[2] Carlo Angiuli. *Computational semantics of Cartesian cubical type theory.* PhD thesis, Carnegie Mellon University, 2019. URL: `https://www.cs.cmu.edu/~cangiuli/thesis/thesis.pdf`.

[3] Carlo Angiuli, Guillaume Brunerie, Thierry Coquand, Kuen-Bang Hou (Favonia), Robert Harper, and Daniel Licata. Syntax and models of cartesian cubical type theory. URL: `https://github.com/dlicata335/cart-cube`, 2019. Preprint.

[4] Carlo Angiuli, Kuen-Bang Hou (Favonia), and Robert Harper. Computational higher type theory iii: Univalent universes and exact equality. URL: https://arxiv.org/abs/1712.01800, 2018. Preprint.

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

[6] Steve Awodey and Michael A Warren. Homotopy theoretic models of identity types. *Mathematical Proceedings of the Cambridge Philosophical Society*, 146(1):45–55, 2009. URL: https://doi.org/10.1017/S0305004108001783.

[7] Bruno Bentzen. On different ways of being equal. *Erkenntnis*, 2020. URL: https://doi.org/10.1007/s10670-020-00275-8.

[8] Marc Bezem, Thierry Coquand, and Simon Huber. A Model of Type Theory in Cubical Sets. In Ralph Matthes and Aleksy Schubert, editors, *19th International Conference on Types for Proofs and Programs (TYPES 2013)*, volume 26 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 107–128, Dagstuhl, Germany, 2014. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[9] Guillaume Brunerie, Axel Ljungström, and Anders Mörtberg. Synthetic cohomology theory in cubical agda. URL: https://staff.math.su.se/anders.mortberg/papers/zcohomology.pdf, 2021.

[10] Evan Cavallo and Robert Harper. Computational Higher Type Theory IV: Inductive Types. URL: https://arxiv.org/pdf/1801.01568.pdf, 01 2018. Preprint.

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

[12] Robert L. Constable. Naïve computational type theory. In Helmut Schwichtenberg and Ralf Steinbrüggen, editors, *Proof and System-Reliability*, pages 213–259. Springer Netherlands, Dordrecht, 2002. URL: https://doi.org/10.1007/978-94-010-0413-8_7.

[13] Thierry Coquand, Simon Huber, and Anders Mörtberg. On higher inductive types in cubical type theory. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '18, pages 255–264, New York, NY, USA, 2018. ACM.

[14] Paul R Halmos. *Naive set theory*. Springer-Verlag, New York, 1974.

[15] James Ladyman and Stuart Presnell. Universes and Univalence in Homotopy Type Theory. *The Review of Symbolic Logic*, 3(12):426–455, 2019. URL: https://doi.org/10.1017/S1755020316000460.

[16] Peter LeFanu Lumsdaine. Weak omega-categories from intensional type theory. *Logical Methods in Computer Science*, Volume 6, Issue 3, September 2010. URL: https://lmcs.episciences.org/1062.

[17] The RedPRL Development Team. The redtt Proof Assistant. URL: `https://github.com/RedPRL/redtt/`, 2018.

[18] The Univalent Foundations Program. Homotopy type theory: Univalent foundations of mathematics, 2013. URL: `https://homotopytypetheory.org/book/`.

[19] Andrea Vezzosi, Anders Mörtberg, and Andreas Abel. Cubical agda: a dependently typed programming language with univalence and higher inductive types. *Proceedings of the ACM on Programming Languages*, 3(ICFP):1–29, 2019. URL: `https://dl.acm.org/doi/10.1145/3341691`.

[20] Vladimir Voevodsky. A very short note on the homotopy λ-calculus. URL: `http://www.math.ias.edu/vladimir/files/2006_09_Hlambda.pdf`, 2006. Unpublished note.

[21] Vladimir Voevodsky. Notes on type systems. URL: `https://www.math.ias.edu/~vladimir/Site3/Univalent_Foundations_files/expressions_current.pdf`, 2009. Unpublished note.