


Internal Parametricity for Cubical Type Theory

Evan Cavallo 

Carnegie Mellon University, Pittsburgh, PA, USA
ecavallo@cs.cmu.edu

Robert Harper 

Carnegie Mellon University, Pittsburgh, PA, USA
rwh@cs.cmu.edu

Abstract

We define a computational type theory combining the contentful equality structure of cartesian cubical type theory with internal parametricity primitives. The combined theory supports both univalence and its relational equivalent, which we call *relativity*. We demonstrate the use of the theory by analyzing polymorphic functions between higher inductive types, and we give an account of the identity extension lemma for internal parametricity.

2012 ACM Subject Classification Theory of computation → Type theory

Keywords and phrases parametricity, cubical type theory, higher inductive types

Digital Object Identifier 10.4230/LIPIcs.CSL.2020.13

Related Version <https://arxiv.org/abs/1901.00489>

Funding We gratefully acknowledge the support of the Air Force Office of Scientific Research through MURI grant FA9550-15-1-0053. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the AFOSR.

Acknowledgements We thank Carlo Angiuli, Steve Awodey, Daniel Gratzer, Kuen-Bang Hou (Favonia), Dan Licata, Anders Mörtberg, Emily Riehl, Christian Sattler, and Jonathan Sterling for their comments and insights.

1 Introduction

Cubical type theory [17, 3, 2] is a recent extension of type theory with *contentful equality* (or *paths*). The central concept of cubical type theory, inherited from homotopy type theory [30], is that terms can be equal in multiple ways, each of which is a method of translating results between them. The motivating example of contentful equality is *structured isomorphism*. In informal mathematical practice, it is common to treat isomorphic objects as if they were “the same,” because any interesting property of one will also hold of the other. Cubical type theory makes this informal practice formal: equality of types *is* isomorphism, which is to say that we have Voevodsky’s *univalence axiom* [32]. For this to be possible, it is essential that equality be contentful, because two objects can be isomorphic in many ways. The key feature of cubical type theory, in comparison with homotopy type theory, is that it is a programming language, not just a logical formalism; in particular, uses of contentful equality *compute*.

A user of cubical type theory can also define types with custom equality structure using *higher inductive types* (HITs) [18, 14]. A simultaneous generalization of inductive types and quotient types, a HIT is freely generated by a collection of constructors, each of which may introduce not only elements but also *paths between elements*. For example, the following specification defines a type $\mathbb{Z}/2\mathbb{Z}$ of integers modulo 2 from a type \mathbb{Z} of integers.

```
data  $\mathbb{Z}/2\mathbb{Z} : \mathcal{U}$  where
| in( $n : \mathbb{Z}$ ) :  $\mathbb{Z}/2\mathbb{Z}$ 
| mod( $n : \mathbb{Z}, x : \mathbb{I}$ ) :  $\mathbb{Z}/2\mathbb{Z}$  [ $x = 0 \leftrightarrow \text{in}(n) \mid x = 1 \leftrightarrow \text{in}(n + 2)$ ]
```



© Evan Cavallo and Robert Harper;

licensed under Creative Commons License CC-BY

28th EACSL Annual Conference on Computer Science Logic (CSL 2020).

Editors: Maribel Fernández and Anca Muscholl; Article No. 13; pp. 13:1–13:17



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

This type has two constructors, `in` and `mod`. The first introduces an element of $\mathbb{Z}/2\mathbb{Z}$ for each element of \mathbb{Z} , while the second identifies `in`(n) and `in`($n + 2$) for every $n : \mathbb{Z}$. To construct a function out of $\mathbb{Z}/2\mathbb{Z}$, we simply explain where to send `in` and `mod`, in direct analogy with the induction principle of an ordinary inductive type. In addition to quotients, HITs permit the definition of higher-dimensional objects, enabling the use of type theory as a domain-specific language for formalizing homotopy-theoretic mathematics [30].

While contentful equality creates new possibilities, it also introduces new obligations. The type of equalities between any pair of objects has its own equality structure, which means that every type actually contains an infinite tower of structure: paths, paths between paths, and so on. A user of HITs is often forced to wrestle with this higher-dimensional structure. A particularly vicious example is provided by the *smash product* [30, §6.8], a key construction in homotopy theory. The smash product is a binary operator $-\wedge- : \mathcal{U}_* \rightarrow \mathcal{U}_* \rightarrow \mathcal{U}_*$ on the universe $\mathcal{U}_* := (X : \mathcal{U}) \times X$ of *pointed types*. Defined as a HIT (see Section 4.3), it is the natural notion of tensor product for \mathcal{U}_* , being left adjoint to the pointed function space. We thus expect properties such as commutativity and associativity: for any $X, Y, Z : \mathcal{U}_*$, we hope that $X \wedge Y \simeq Y \wedge X$ and $(X \wedge Y) \wedge Z \simeq X \wedge (Y \wedge Z)$. However, these laws are not so simple to prove. The associator, in particular, involves two stacked applications of \wedge ; this means the programmer must wrangle with two-dimensional structure to define functions back and forth, then three-dimensional structure to prove they are inverses. Worse yet, these are only the first level of an infinite hierarchy of laws satisfied by \wedge . For example, Mac Lane’s pentagon relates the two different ways of re-associating the product of four types; to prove it requires building *four*-dimensional terms. Despite concerted effort [31, 12], a complete formal proof that the smash product is a symmetric monoidal product has yet to be produced.

For all the suffering, these properties seem “obvious” in a way familiar to computer scientists: they look like consequences of *parametricity* [25]. Parametricity is Reynolds’ crystallization of a property enjoyed by many type theories: programs behave uniformly in their type variables. Reynolds captures this uniformity in the existence of a *relational interpretation* of type theory that expresses the invariance of type constructions under a broad class of relations. With parametricity, it is often possible to derive what Wadler dubs “free theorems” [33], naturality properties enjoyed by any term of a given type. In our case, we can hope that there are only so many functions $\alpha : (X, Y, Z : \mathcal{U}_*) \rightarrow (X \wedge Y) \wedge Z \rightarrow X \wedge (Y \wedge Z)$. Perhaps *any* definable function of this type satisfies Mac Lane’s pentagon?

Contributions

We present a cubical type theory with *internal parametricity*, a further extension to type theory introduced by Bernardy and Moulin [8, 9, 6, 21, 23, 22]. Just as cubical type theory makes the fact that all constructions act on isomorphisms available to the user, internally parametric type theory exposes that all constructions act on relations, providing an *operational* account of Reynolds’ denotational presentation of parametricity. In fact, the two are based on the same design principle: the use of *dimension variables*.

Our contribution can be viewed in two ways. On the one hand, we bring parametricity to bear on problems in cubical type theory, giving in particular a characterization of maps between smash products. On the other, we provide an internally parametric theory that enjoys the extensionality principles of cubical type theory. A lack of function extensionality, for one, is acutely visible when working with Church encodings. By relying on univalence, we are able to eliminate the technical device of *I-sets* used by Bernardy, Coquand, and Moulin in their presheaf model [6]. We also explore the use of internal parametricity beyond the initial forays of Bernardy et al., internally developing the sub-universe of *bridge-discrete*

types (which is closed under all type formers except the universe) as a substitute for the traditional identity extension lemma. This is a departure from previous approaches [4, 23] where the goal is typically to *externally* restrict the universe to bridge-discrete types from the start. We also show by example that the relational interpretations of inductive types can be characterized. Finally, we use this paper as an opportunity to compare and contrast the mechanisms underlying cubical and parametric type theory.

We begin in Section 2 by introducing cubical type theory. In Section 3, we mix in the parametricity primitives. With the theory complete, we make use of it in Section 4, proving results about the smash product and probing the status of the identity extension lemma. In Section 5, we go into more detail on the meaning of the judgments and canonicity. In Section 6, we briefly sketch a presheaf model. We close in Section 7 with a discussion of related work. Complete proofs of our results, and in particular a detailed development of the computational interpretation, can be found in our companion technical report [15].

2 Cubical type theory

Cubical type theory, in its various incarnations, is a means of organizing the data of a type equipped with path structure. Homotopy theory suggests various ways of doing this; empirically, *cubical* structure is most convenient for the design of type theories, because n -dimensional elements of a type can be represented by terms in a context of n *dimension variables*. In this section, we recall cartesian cubical type theory [3, 2]; however, one can substitute another cubical type theory (e.g., [17, 24]) in the remainder of this paper without difficulty.

2.1 Path dimensions

Like ordinary type theory, cubical type theory is based on four judgment forms, expressing typehood, type equality, elementhood, and element equality.

$$\Gamma \gg A \text{ type} \quad \Gamma \gg A = B \text{ type} \quad \Gamma \gg M \in A \quad \Gamma \gg M = N \in A$$

For us, these judgments are *behavioral specifications* on terms of an untyped programming language. Roughly, a program A is a type when, for any instantiation of its hypotheses, it computes to a name in some prescribed set of *value types*, while M is in A when its instantiations compute to values in the type named by A . Types and elements are equal when they compute to the same values, where value equality is again prescribed in advance. We use the notation \gg for the behavioral counterpart of the formal \vdash . To more quickly give the reader a feel for the system, however, we will defer precise definitions of the judgments to Section 5, and instead first present a collection of rules they satisfy.

Cubical type theory is distinguished by the addition of *path dimensions*, for which we write r, s . These are specified by judgments $\Gamma \gg r \text{ pdim}$ and $\Gamma \gg r = s \text{ pdim}$ and populated by variables and two distinguished constants.

$$\Gamma, x : \mathbb{I}, \Gamma' \gg x \text{ pdim} \quad \Gamma \gg 0 \text{ pdim} \quad \Gamma \gg 1 \text{ pdim}$$

In the context we write dimension variable assumptions in the form $x : \mathbb{I}$, but this is merely suggestive notation: “ \mathbb{I} ” is not the name of a type. We also allow dimension *equality* assumptions. (Note that dimension equality is decidable.)

13:4 Internal Parametricity for Cubical Type Theory

$$\begin{array}{c}
\frac{\Gamma, x : \mathbb{I} \gg A \text{ type} \quad \Gamma \gg M_0 \in A\langle 0/x \rangle \quad \Gamma \gg M_1 \in A\langle 1/x \rangle}{\Gamma \gg \text{Path}_{x.A}(M_0, M_1) \text{ type}} \\
\\
\frac{\Gamma, x : \mathbb{I} \gg P \in A \quad \Gamma \gg P\langle 0/x \rangle = M_0 \in A\langle 0/x \rangle \quad \Gamma \gg P\langle 1/x \rangle = M_1 \in A\langle 1/x \rangle}{\Gamma \gg \lambda^{\mathbb{I}}x.P \in \text{Path}_{x.A}(M_0, M_1)} \\
\\
\frac{\Gamma \gg Q \in \text{Path}_{x.A}(M_0, M_1) \quad \Gamma \gg r \text{ pdim}}{\Gamma \gg Q@r \in A\langle r/x \rangle} \qquad \frac{\Gamma, x : \mathbb{I} \gg P \in A}{\Gamma \gg (\lambda^{\mathbb{I}}x.P)@r = P\langle r/x \rangle \in A\langle r/x \rangle} \\
\\
\frac{\Gamma \gg Q \in \text{Path}_{x.A}(M_0, M_1) \quad \varepsilon \in \{0, 1\}}{\Gamma \gg Q@\varepsilon = M_\varepsilon \in A\langle \varepsilon/x \rangle} \qquad \frac{\Gamma \gg Q \in \text{Path}_{x.A}(M_0, M_1)}{\Gamma \gg Q = \lambda^{\mathbb{I}}x.Q@x \in \text{Path}_{x.A}(M_0, M_1)}
\end{array}$$

■ **Figure 1** Rules for Path-types.

$$\frac{}{\cdot \text{ ctx}} \qquad \frac{\Gamma \text{ ctx} \quad \Gamma \gg A \text{ type}}{\Gamma, a : A \text{ ctx}} \qquad \frac{\Gamma \text{ ctx}}{\Gamma, x : \mathbb{I} \text{ ctx}} \qquad \frac{\Gamma \text{ ctx} \quad \Gamma \gg r \text{ pdim} \quad \Gamma \gg s \text{ pdim}}{\Gamma, r = s \text{ ctx}}$$

A path dimension variable can be pictured as varying in the real unit interval $[0, 1]$: as x varies in a term $x : \mathbb{I} \gg M \in A$, the term M draws out a “line” in A . The line’s “endpoints” are obtained by substituting 0 and 1 for x : writing $\langle r/x \rangle$ for path dimension substitution, we have $M\langle 0/x \rangle \in A\langle 0/x \rangle$ and $M\langle 1/x \rangle \in A\langle 1/x \rangle$. Path dimension variables support all of the structural rules (weakening, contraction, and exchange) enjoyed by ordinary variables. In contrast to ordinary variables, however, there is more to M than its closed instantiations $M\langle 0/x \rangle$ and $M\langle 1/x \rangle$: there can be many distinct terms with the same endpoints.

Path dimensions provide a judgmental notion of contentful equality: a path between $M_0 \in A$ and $M_1 \in A$ is a term $x : \mathbb{I} \gg P \in A$ such that $P\langle 0/x \rangle = M_0$ and $P\langle 1/x \rangle = M_1$. The judgmental notion is then internalized via *Path-types*, shown in Figure 1. Aside from the indices M_0 and M_1 , they behave as “functions out of \mathbb{I} ”: they are introduced by abstraction, eliminated by application, and satisfy β - and η -rules. In general, Path-types are heterogeneous, meaning that they are *dependent* functions: they take the form $\text{Path}_{x.A}(M_0, M_1)$ where $x : \mathbb{I} \gg A \text{ type}$, and applying $Q \in \text{Path}_{x.A}(M_0, M_1)$ at r yields $Q@r \in A\langle r/x \rangle$. When A does *not* depend on x , we simply write $\text{Path}_A(M_0, M_1)$.

Using just these few principles of cubical type theory, we can already observe that path types validate function extensionality. Given $F_0, F_1 \in (a:A) \rightarrow B$, a pointwise path between them is a term $H \in (a:A) \rightarrow \text{Path}_B(F_0a, F_1a)$. Given such an H , we obtain a path between F_0 and F_1 by simply flipping the order of abstraction: $\lambda^{\mathbb{I}}x.\lambda a.Ha@x \in \text{Path}_{(a:A) \rightarrow B}(F_0, F_1)$.

2.2 Coercion and composition

The path apparatus equips each type with some kind of infinite-dimensional relation. It is reflexive: given any $M \in A$, we have $\lambda^{\mathbb{I}}_.M \in \text{Path}_A(M, M)$. However, there is as yet no reason for it to be symmetric or transitive, nor for all constructions to respect it. These properties are ensured by adding two operations: *coercion* and *composition*.

The coercion operation turns paths between types into isomorphisms,¹ implementing one direction of the correspondence required by the univalence axiom. Given a line $x.A$ and an element $M \in A\langle r/x \rangle$ at some index r , coercion produces an element at any other index s .

$$\frac{\Gamma, x : \mathbb{I} \gg A \text{ type} \quad \Gamma \gg r, s \text{ pdim} \quad \Gamma \gg M \in A\langle r/x \rangle}{\Gamma \gg \text{coe}_{x.A}^{r \rightsquigarrow s}(M) \in A\langle s/x \rangle}$$

We moreover impose the equation $\text{coe}_{x.A}^{r \rightsquigarrow r}(M) = M \in A\langle r/x \rangle$. From this, one can show that $\lambda a. \text{coe}_{x.A}^{r \rightsquigarrow s}(a) \in A\langle r/x \rangle \rightarrow A\langle s/x \rangle$ is in fact an isomorphism. Using coercion, we can see that all constructions respect paths, in the following sense: if we have some property $B \in A \rightarrow \mathcal{U}$, a proof $N \in BM_0$ that B holds of some $M_0 \in A_0$, and a path $Q \in \text{Path}_A(M_0, M_1)$, we get a proof $\text{coe}_{x.B(Q@x)}^{0 \rightsquigarrow 1}(N) \in BM_1$ that B holds of M_1 . This fact can be used to invert and compose paths, establishing that path equality is symmetric and transitive.

While coercion gives us all we need of equality, it is not a strong enough “induction hypothesis.” Operationally, the evaluation of a coercion term is guided by the outermost constructor of type line. To explain the reduction of coercion for Path -types, a second operation, (*homogeneous*) *composition*, is required to obtain a term with the correct endpoints. As the purpose of this operation is essentially technical, however, we defer to [3, 2] for details.

2.3 Paths in the universe: V-types and univalence

Coercion converts paths of types into isomorphisms, but we still need a way to convert isomorphisms into paths. This is accomplished by a new type former: *Glue-types* in [17, 2], *V-types* in [3], and *G-types* in [11]. We use V -types, a sufficient special case of *Glue-types*.

The V constructor does not, strictly speaking, convert an isomorphism into a path. Rather, it takes an isomorphism and a path as input, and produces a *second* path that is the concatenation of the two inputs. Precisely, it takes a dimension r pdim , some $r = 0 \gg A$ type defined at its 0 endpoint, some B type, and an isomorphism $r = 0 \gg E \in A \simeq B$. These inputs form a V -shape, hence the name.

$$\begin{array}{ccc} A & & \\ E \downarrow & \searrow \text{V}_r(A, B, E) & \\ B_0 & \xrightarrow{B} & B_1 \\ r \rightarrow & & \end{array}$$

The output is a type $\text{V}_r(A, B, E)$ satisfying the equations $r = 0 \gg \text{V}_r(A, B, E) = A$ type and $r = 1 \gg \text{V}_r(A, B, E) = B$ type.

To transform an isomorphism $E \in A \simeq B$ of types $A, B \in \mathcal{U}$ into a path, we simply take $\lambda^{\mathbb{I}x}. \text{V}_x(A, B, E) \in \text{Path}_{\mathcal{U}}(A, B)$. This is the special case of the V -type where B is constant in the direction of the output (here x).

2.4 Higher inductive types

As described in the introduction, cubical type theory also supports *higher inductive types*, types generated by constructors that may take dimension arguments and be attached at their boundaries to other elements. We refer to [18, 14] for formal treatments of HITs in cubical type theory; for this paper, we will only need an intuitive understanding.

¹ For us, an *isomorphism* is a function with a left and right inverse up to path equality. That is, we define $A \simeq B$ to be the following type.

$$(f : A \rightarrow B) \times (l : B \rightarrow A) \times (r : B \rightarrow A) \times ((a:A) \rightarrow \text{Path}_A(l(fa), a)) \times ((b:B) \rightarrow \text{Path}_B(f(rb), b))$$

These are often called *equivalences* in the literature, and have several equivalent definitions [30, §4].

3 Internalizing parametricity

We now add *internal parametricity* primitives, closely following Bernardy, Coquand, and Moulin [6]. (Our notation, however, differs substantially from theirs; see [15, Figure 7] for a translation dictionary.) Reynolds’ parametricity captures the vague concept of “uniformity in type variables” by the precise concept of *acting on relations*. To say that all constructions act on relations is essentially to say that type theory has a relational semantics. With internal parametricity, we make that semantics visible inside the theory.

With cubical type theory, the goal was to ensure that all constructions act on isomorphisms; the solution was to equip each type with equality structure via dimension variables, then to identify lines *between* types with isomorphisms. For internal parametricity, we ensure that constructions act on relations with the same technique, but now identifying lines between types with *type-valued relations*. Where we use the word *path* in cubical type theory, we will use *bridge* in parametric type theory, following Nuyts et al. [23].

3.1 Bridge dimensions

Second verse, same as the first: we introduce *bridge dimensions* $\mathbf{r}, \mathbf{s}, \dots$ by judgments $\Gamma \gg \mathbf{r} \text{ bdim}$ and $\Gamma \gg \mathbf{r} = \mathbf{s} \text{ bdim}$ with two constants $\Gamma \gg \mathbf{0}, \mathbf{1} \text{ bdim}$. We use **bold type** to distinguish bridge from path dimensions. We likewise add bridge dimension and equality assumptions – but this time, only equations where one side is a constant.

$$\frac{\Gamma \text{ ctx}}{\Gamma, \mathbf{x} : \mathbf{2} \text{ ctx}} \qquad \frac{\Gamma \text{ ctx} \quad \Gamma \gg \mathbf{r} \text{ bdim} \quad \varepsilon \in \{\mathbf{0}, \mathbf{1}\}}{\Gamma, \mathbf{r} = \varepsilon \text{ ctx}}$$

The distinguishing feature of bridge dimensions is that they are *substructural*, specifically *affine*: they do not support contraction. For $\Gamma \text{ ctx}$ and $(\mathbf{x} : \mathbf{2}) \in \Gamma$, write $\Gamma \setminus^{\mathbf{x}}$ for the result of deleting \mathbf{x} and all term variables that occur beyond it from the context.²

$$(\Gamma, \mathbf{y} : \mathbb{I}) \setminus^{\mathbf{x}} := (\Gamma \setminus^{\mathbf{x}}, \mathbf{y} : \mathbb{I}) \quad (\Gamma, \mathbf{a} : A) \setminus^{\mathbf{x}} := \Gamma \setminus^{\mathbf{x}} \quad (\Gamma, \mathbf{y} : \mathbf{2}) \setminus^{\mathbf{x}} := \begin{cases} \Gamma & \text{if } \mathbf{x} = \mathbf{y} \\ (\Gamma \setminus^{\mathbf{x}}, \mathbf{y} : \mathbf{2}) & \text{if } \mathbf{x} \neq \mathbf{y} \end{cases}$$

Set $\Gamma \setminus^{\varepsilon} = \Gamma$. We then have the following structural rules for bridge dimension variables.

$$\frac{}{\Gamma, \mathbf{x} : \mathbf{2}, \Gamma' \gg \mathbf{x} \text{ bdim}} \text{BHYP} \qquad \frac{\Gamma' \gg \mathcal{J}}{\Gamma, \mathbf{x} : \mathbf{2}, \Gamma' \gg \mathcal{J}} \text{BWEAK}$$

$$\frac{\Gamma \gg \mathbf{r} \text{ bdim} \quad \Gamma \setminus^{\mathbf{r}}, \mathbf{x} : \mathbf{2} \gg \Gamma' \text{ ctx} \quad \Gamma \setminus^{\mathbf{r}}, \mathbf{x} : \mathbf{2}, \Gamma' \gg \mathcal{J}}{\Gamma(\Gamma' \langle \mathbf{r}/\mathbf{x} \rangle) \gg \mathcal{J} \langle \mathbf{r}/\mathbf{x} \rangle} \text{BCUT}$$

The first two rules are unsurprising, but the third contains an essential restriction. To substitute \mathbf{r} for \mathbf{x} in a judgment $\Gamma, \mathbf{x} : \mathbf{2}, \Gamma' \gg \mathcal{J}$, we must know that neither Γ' or \mathcal{J} refers to \mathbf{r} (if it is a variable). In other words, \mathbf{r} must be *fresh* for Γ' and \mathcal{J} .

The Bezem-Coquand-Huber (BCH) cubical sets model [10, 11] also uses affine dimension variables, but recent work on cubical type theories has focused on structural variables. The primary motivation for the shift is to support HITs, whose development remains an open problem in the affine setting; ease of implementation is another factor. For internal

² Here we follow the approach developed by Cheney for nominal dependent type theory [16].

$$\begin{array}{c}
\Gamma \gg \mathbf{r} \text{ bdim} \quad \Gamma \setminus^{\mathbf{r}}, \mathbf{x} : \mathbf{2} \gg A \text{ type} \quad \Gamma \setminus^{\mathbf{r}}, \mathbf{x} : \mathbf{2}, a : A \gg B \text{ type} \quad \Gamma \gg M \in A \langle \mathbf{r}/\mathbf{x} \rangle \\
\Gamma \setminus^{\mathbf{r}}, a_0 : A \langle \mathbf{0}/\mathbf{x} \rangle \gg N_0 \in B \langle \mathbf{0}/\mathbf{x} \rangle [a_0/a] \quad \Gamma \setminus^{\mathbf{r}}, a_1 : A \langle \mathbf{1}/\mathbf{x} \rangle \gg N_1 \in B \langle \mathbf{1}/\mathbf{x} \rangle [a_1/a] \\
\Gamma \setminus^{\mathbf{r}}, a_0 : A \langle \mathbf{0}/\mathbf{x} \rangle, a_1 : A \langle \mathbf{1}/\mathbf{x} \rangle, \bar{a} : \text{Bridge}_{\mathbf{x}.A}(a_0, a_1) \gg \bar{N} \in \text{Bridge}_{\mathbf{x}.B[\bar{a}@\mathbf{x}/a]}(N_0, N_1) \\
\hline
\Gamma \gg \text{extent}_{\mathbf{r}}(M; a_0.N_0, a_1.N_1, a_0.a_1.\bar{a}.\bar{N}) \in B \langle \mathbf{r}/\mathbf{x} \rangle [M/a] \\
\text{extent}_{\varepsilon}(M; \dots) = N_{\varepsilon}[M/a_{\varepsilon}] \in B \langle \varepsilon/\mathbf{x} \rangle [M/a] \\
\Gamma \setminus^{\mathbf{r}}, \mathbf{x} : \mathbf{2} \gg M \in A \\
\hline
\Gamma \gg \text{extent}_{\mathbf{r}}(M \langle \mathbf{r}/\mathbf{x} \rangle; \dots) = \bar{N}[M \langle \mathbf{0}/\mathbf{x} \rangle / a_0][M \langle \mathbf{1}/\mathbf{x} \rangle / a_1][\lambda^{\mathbf{2}}\mathbf{x}.M/\bar{a}]@_{\mathbf{r}} \in B \langle \mathbf{r}/\mathbf{x} \rangle [M/a]
\end{array}$$

■ **Figure 2** The extent operator. We omit straightforwardly inferrable premises for readability.

parametricity, however, affine dimensions are essential to ensure the correct characterization of bridges in function types (Section 3.2) and in the universe (Section 3.3).

As with paths, we introduce types $\text{Bridge}_{\mathbf{x}.A}(M_0, M_1)$ of bridges over $\mathbf{x}.A$ from M_0 to M_1 . We write $\lambda^{\mathbf{2}}\mathbf{x}.P$ for the values of these types. In accordance with the judgmental structure, a bridge can only be applied to a fresh variable: if $\Gamma \setminus^{\mathbf{r}} \gg Q \in \text{Bridge}_{\mathbf{x}.A}(M_0, M_1)$, then $\Gamma \gg Q@_{\mathbf{r}} \in A \langle \mathbf{r}/\mathbf{x} \rangle$. Otherwise, they are exactly like path types; see [15, §8.3] for rules. We now have one direction of our desired correspondence between bridges of types and binary relations: for any $\mathbf{x}.A$, we have the relation $\text{Bridge}_{\mathbf{x}.A}(-, -)$ on $A \langle \mathbf{0}/\mathbf{x} \rangle$ and $A \langle \mathbf{1}/\mathbf{x} \rangle$.

3.2 Bridges at function type: extent

In the standard relational interpretation of type theory [7, 4], two functions are related when they take related arguments to related results. As such, we expect $\text{Bridge}_{\mathbf{x}.(a:A) \rightarrow B}(F_0, F_1)$ to be isomorphic to the following.

$$(a_0 : A \langle \mathbf{0}/\mathbf{x} \rangle)(a_1 : A \langle \mathbf{1}/\mathbf{x} \rangle)(q : \text{Bridge}_{\mathbf{x}.A}(a_0, a_1)) \rightarrow \text{Bridge}_{\mathbf{x}.B[q@\mathbf{x}/a]}(F_0 a_0, F_1 a_1) \quad (1)$$

Although it is simple to define a map from $\text{Bridge}_{\mathbf{x}.(a:A) \rightarrow B}(F_0, F_1)$ to the type (1), the converse is more delicate. In fact, the first role of substructurality is in enabling this principle. As we have seen, structural dimensions give rise to a *different* principle: $\text{Path}_{\mathbf{x}.(a:A) \rightarrow B}(F_0, F_1) \simeq (a:A) \rightarrow \text{Path}_{\mathbf{x}.B}(F_0 a, F_1 a)$ when A does not depend on \mathbf{x} . The proof, sketched in Section 2.1, uses the interchangeability of terms $\lambda a. \lambda^{\mathbb{1}}\mathbf{x}.P$ and $\lambda^{\mathbb{1}}\mathbf{x}. \lambda a.P$. However, $\lambda a. \lambda^{\mathbf{2}}\mathbf{x}.P$ and $\lambda^{\mathbf{2}}\mathbf{x}. \lambda a.P$ are *not* interchangeable: in the former, \mathbf{x} ranges over dimensions that are *fresh for a*, whereas no such restriction is in play in the latter. Conversely, structural dimensions would not allow us to prove that the map from $\text{Bridge}_{\mathbf{x}.(a:A) \rightarrow B}(F_0, F_1)$ to (1) is an isomorphism, as having both principles at once would lead to a contradiction (in the presence of Gel-types, defined below). The two principles *are* both derivable for paths, but only thanks to the presence of coercion – an operation with no equivalent for bridges.

To see how we can get from (1) to $\text{Bridge}_{\mathbf{x}.(a:A) \rightarrow B}(F_0, F_1)$ with affine dimensions, let H in (1) be given. Abstracting \mathbf{x} and a , our goal is to exhibit a term in B that is equal to $F_0 a$ when $\mathbf{x} = \mathbf{0}$ and $F_1 a$ when $\mathbf{x} = \mathbf{1}$. Recall that a , being abstracted when \mathbf{x} is in scope, ranges over terms that may mention \mathbf{x} . We would like, then, to think of a as a *bridge* in direction \mathbf{x} , and to supply that bridge to H . In syntax, we would like to write “ $H(a \langle \mathbf{0}/\mathbf{x} \rangle)(a \langle \mathbf{1}/\mathbf{x} \rangle)(\lambda^{\mathbf{2}}\mathbf{x}.a)@_{\mathbf{x}}$ ”. This does not quite make sense: a is a variable, so $a \langle \mathbf{0}/\mathbf{x} \rangle = a$.

We instead introduce an operator, $\text{extent}_{\mathbf{r}}(M; a_0.N_0, a_1.N_1, a_0.a_1.\bar{a}.\bar{N})$, that performs the substitutions and capture once a (now M) is instantiated. This operator satisfies the rules shown in Figure 2; we call it *extent* because it reveals the extent of the term M as a

$$\begin{array}{c}
 \frac{\Gamma \gg \mathbf{r} \text{ bdim} \quad \Gamma \setminus^{\mathbf{r}} \gg A \text{ type} \quad \Gamma \setminus^{\mathbf{r}} \gg B \text{ type} \quad \Gamma \setminus^{\mathbf{r}}, a : A, b : B \gg R \text{ type}}{\Gamma \gg \text{Gel}_{\mathbf{r}}(A, B, a.b.R) \text{ type}} \\
 \\
 \frac{\Gamma \setminus^{\mathbf{r}} \gg M \in A \quad \Gamma \setminus^{\mathbf{r}} \gg N \in B \quad \Gamma \setminus^{\mathbf{r}} \gg P \in R[M, N/a, b]}{\Gamma \gg \text{gel}_{\mathbf{r}}(M, N, P) \in \text{Gel}_{\mathbf{r}}(A, B, R)} \\
 \\
 \frac{\Gamma, \mathbf{x} : \mathbf{2} \gg Q \in \text{Gel}_{\mathbf{x}}(A, B, E)}{\Gamma \gg \text{ungel}(\mathbf{x}.Q) \in R[Q\langle \mathbf{0}/\mathbf{x} \rangle, Q\langle \mathbf{1}/\mathbf{x} \rangle/a, b]} \\
 \\
 \text{Gel}_{\mathbf{0}}(A, B, a.b.R) = A \quad \text{Gel}_{\mathbf{1}}(A, B, a.b.R) = B \quad \text{gel}_{\mathbf{0}}(M, N, P) = M : A \\
 \text{gel}_{\mathbf{1}}(M, N, P) = N : B \quad \text{ungel}(\mathbf{x}.\text{gel}_{\mathbf{x}}(M, N, P)) = P : R[M, N/a, b] \\
 Q\langle \mathbf{r}/\mathbf{x} \rangle = \text{gel}_{\mathbf{r}}(Q\langle \mathbf{0}/\mathbf{x} \rangle, Q\langle \mathbf{1}/\mathbf{x} \rangle, \mathbf{x}.Q) : \text{Gel}_{\mathbf{r}}(A, B, a.b.R)
 \end{array}$$

■ **Figure 3** Rules for Gel-types.

line in direction \mathbf{r} . There are three cases: either \mathbf{r} is $\mathbf{0}$ or $\mathbf{1}$, in which case M is simply a point, or \mathbf{r} is a variable \mathbf{x} , in which case M is a bridge in direction \mathbf{x} . The operator takes an argument for each case, here N_0 , N_1 , and \bar{N} . The last of these takes as input endpoints a_0, a_1 and a bridge \bar{a} between them and produces a bridge between N_0 and N_1 ; when $\text{extent}_{\mathbf{x}}$ executes, it supplies $M\langle \mathbf{0}/\mathbf{x} \rangle$, $M\langle \mathbf{1}/\mathbf{x} \rangle$, and $\lambda^2 \mathbf{x}.M$ to \bar{N} and outputs its value at \mathbf{x} .

Substructurality is essential to extent because of its use of variable capture: the mapping $\lambda^2 : (\mathbf{x}, M) \rightsquigarrow \lambda^2 \mathbf{x}.M$ is not stable under all dimension substitutions. If $M = M'(\mathbf{x}, \mathbf{y})$, for example, then applying λ^2 after substituting $\langle \mathbf{y}/\mathbf{x} \rangle$ results in $\lambda^2 \mathbf{y}.M'(\mathbf{y}, \mathbf{y})$, while applying λ^2 before substituting $\langle \mathbf{y}/\mathbf{x} \rangle$ results in the inequivalent $\lambda^2 \mathbf{x}.M'(\mathbf{x}, \mathbf{y})$. However, variable capture *does* commute with substitution of *fresh* variables. Returning to the original motivation, extent is exactly what is needed to get from (1) to $\text{Bridge}_{\mathbf{x}.(a:A) \rightarrow B}(F_0, F_1)$.

$$H \rightsquigarrow \lambda^2 \mathbf{x}.\lambda a.\text{extent}_{\mathbf{x}}(a; a_0.F_0 a_0, a_1.F_1 a_1, a_0.a_1.\bar{a}.H a_0 a_1 \bar{a}) : \text{Bridge}_{\mathbf{x}.(a:A) \rightarrow B}(F_0, F_1)$$

It is straightforward to show that this map is in fact an isomorphism [15, Theorem 6.9].

3.3 Bridges in the universe: Gel-types and relativity

The final ingredient is the equivalent of univalence: a characterization of bridges in the universe as binary type-valued relations. We call this property *relativity*.

► **Definition 1.** *A universe \mathcal{U} is relativistic when for every pair of types $A, B : \mathcal{U}$, the map $\lambda C.\text{Bridge}_{\mathbf{x}.C @_{\mathbf{x}}}(-, -) \in \text{Bridge}_{\mathcal{U}}(A, B) \rightarrow (A \times B \rightarrow \mathcal{U})$ is an isomorphism.*

As in cubical type theory, we implement the inverse map with a new type constructor: **Gel**, so named because it resembles the **G**-types of the BCH model but applies to **relations** rather than isomorphisms. Rules for **Gel**-types – omitting those for coercion and composition – are displayed in Figure 3. In stark contrast to **V**- or **Glue**-types, coercion and composition in **Gel**-types are simple; this is because the direction of a coercion or composition is always a path dimension and therefore orthogonal to the direction of the **Gel**-type.

Given a dimension \mathbf{r} and a relation $\Gamma \setminus^{\mathbf{r}}, a : A, b : B \gg R \text{ type}$ for which \mathbf{r} is fresh, we obtain a type $\text{Gel}_{\mathbf{r}}(A, B, a.b.R)$ satisfy $\text{Gel}_{\mathbf{0}}(A, B, a.b.R) = A$ and $\text{Gel}_{\mathbf{1}}(A, B, a.b.R) = B$. Its values take the form $\text{gel}_{\mathbf{r}}(M, N, P)$ where $M \in A$, $N \in B$, and P is a proof the

two are related by R ; we have $\text{gel}_0(M, N, P) = M \in A$ and $\text{gel}_1(M, N, P) = N \in B$. Given a bridge $\Gamma, \mathbf{x} : \mathbf{2} \gg Q \in \text{Gel}_{\mathbf{x}}(A, B, a.b.R)$ over a Gel-type, we can project the proof $\text{ungel}(\mathbf{x}.Q) : R[Q\langle \mathbf{0}/\mathbf{x} \rangle, Q\langle \mathbf{1}/\mathbf{x} \rangle/a, b]$ that its endpoints – elements of A and B respectively – are related by R . When we have $A, B \in \mathcal{U}$ and $R \in A \times B \rightarrow \mathcal{U}$, we write $\text{Gel}_{\mathbf{r}}(A, B, R)$ as shorthand for $\text{Gel}_{\mathbf{r}}(A, B, a.b.R\langle a, b \rangle)$.

Whereas \mathbf{V} -types concatenate an isomorphism and a path to produce a path, Gel-types directly convert relations to bridges. That this is possible is a consequence of substructurality. The constructor $\text{Gel}_{\mathbf{x}}$ performs a dimension shift: it takes A, B , and R in some context Γ , and it produces a type in context $\Gamma, \mathbf{x} : \mathbf{2}$. To express a typing rule for Gel, we must be able to specify that \mathbf{x} is fresh for A, B, R . By contrast, $\mathbf{V}_{\mathbf{x}}$ takes a type in context Γ, \mathbf{x} with an isomorphism at one end and produces a type in context Γ, \mathbf{x} ; there is no dimension shift.

Moreover, a \mathbf{V} -like type would be insufficient for internal parametricity. In cubical type theory, we can turn $E : A \simeq B$ into a path by attaching it to the constant path $\lambda^{\mathbb{I}}_.B$, which corresponds to the identity equivalence at B . For bridges, however, this might not give the desired result: the constant bridge $\lambda^{\mathbf{2}}_.B$ corresponds to the relation $\text{Bridge}_B(-, -)$, which may be distinct from the identity relation $\text{Path}_B(-, -)$. In particular, they fail to coincide when B is a universe: we have $\text{Bridge}_{\mathcal{U}}(A, B) \simeq (A \times B \rightarrow \mathcal{U}) \not\simeq (A \simeq B) \simeq \text{Path}_{\mathcal{U}}(A, B)$.

► **Theorem 2.** *Any universe \mathcal{U} closed under Gel types is relativistic.*

Sketch. Given $a : A$ and $b : B$, the gel and ungel operators constitute an isomorphism between $R\langle a, b \rangle$ and $\text{Bridge}_{\mathbf{x}. \text{Gel}_{\mathbf{x}}(A, B, R)}(a, b)$ by virtue of their β - and η -rules. By univalence, this gives a path in \mathcal{U} between the two. By function extensionality, then, we have a path from R to $\text{Bridge}_{\mathbf{x}. \text{Gel}_{\mathbf{x}}(A, B, R)}(-, -)$. Thus $\lambda R. \text{Gel}_{\mathbf{x}}(A, B, R)$ is a left inverse to $\lambda C. \text{Bridge}_{\mathbf{x}. C@_{\mathbf{x}}}(-, -)$, getting us halfway to a proof that \mathcal{U} is relativistic.

For the right inverse, we need a path from $\lambda^{\mathbf{2}}\mathbf{x}. \text{Gel}_{\mathbf{x}}(A, B, \text{Bridge}_{\mathbf{x}. C@_{\mathbf{x}}}(-, -))$ to C for $C : \text{Bridge}_{\mathcal{U}}(A, B)$. We use the fact that bridges between paths correspond to paths between bridges, a correspondence implemented by swapping binders [15, Theorem 6.2]. It thus suffices to exhibit a bridge over $\mathbf{x}. \text{Path}_{\mathcal{U}}(\text{Gel}_{\mathbf{x}}(A, B, \text{Bridge}_{\mathbf{x}. C@_{\mathbf{x}}}(-, -)), C@_{\mathbf{x}})$ between $\lambda^{\mathbb{I}}_.A$ and $\lambda^{\mathbb{I}}_.B$. Next, we apply univalence, reducing the goal to constructing a bridge over $\mathbf{x}. \text{Gel}_{\mathbf{x}}(A, B, \text{Bridge}_{\mathbf{x}. C@_{\mathbf{x}}}(-, -)) \simeq C@_{\mathbf{x}}$ between the identity equivalences on A and B .

Finally, we can show, using the characterization of bridges at function type, that bridges at an isomorphism type correspond to isomorphisms between bridges in the source and target types [15, Corollary 6.10]. That is, it is enough to show that for every $a : A$ and $b : B$, we have an isomorphism between $\text{Bridge}_{\mathbf{x}. \text{Gel}_{\mathbf{x}}(A, B, \text{Bridge}_{\mathbf{x}. C@_{\mathbf{x}}}(-, -))}(a, b)$ and $\text{Bridge}_{\mathbf{x}. C@_{\mathbf{x}}}(a, b)$. This is a special case of the inverse condition already proven. ◀

4 The practice of internal parametricity

We have completed a formulation of internally parametric cubical type theory. Now, we will use it. As a warm-up, we prove that `bool` is isomorphic to its Church encoding. Then we move on to novel results: a characterization of bridges in `bool`, the definition and applications of bridge-discrete types, and finally, a characterization of maps between smash products.

4.1 The basics: booleans

A classic application of parametricity is the characterization of *Church encodings*, definitions of inductive types by their universal properties. Our universes are predicative, so Church encodings likely cannot be used to obtain data types *ex nihilo*. If we know that a data type exists, however, we can show that it is isomorphic to its Church encoding.

13:10 Internal Parametricity for Cubical Type Theory

► **Theorem 3.** *The type $B := (X:\mathcal{U}) \rightarrow X \rightarrow X \rightarrow X$ is isomorphic to `bool`.*

Proof. It is easy to exhibit functions in either direction.

$$\lambda b.(\lambda X.\lambda t.\lambda f.\text{if}_X(b;t,f)) \in \text{bool} \rightarrow B \qquad \lambda g.g(\text{bool})(\text{true})(\text{false}) \in B \rightarrow \text{bool}$$

One inverse condition is immediate. For the other, we must construct a path from $\lambda X.\lambda t.\lambda f.\text{if}_X(g(\text{bool})(\text{true})(\text{false});t,f)$ to g for all $g : B$. Applying function extensionality, we assume $X : \mathcal{U}$ and $t, f : X$ and aim to connect $\text{if}_X(g(\text{bool})(\text{true})(\text{false});t,f)$ to $gXtf$.

Intuitively, we want to say that g is natural in its type argument; specifically, with respect to the map $\text{if}_X(-;t,f) : \text{bool} \rightarrow X$. Accordingly, we define the relation given by the graph of this map: $R : \text{bool} \times X \rightarrow \mathcal{U}$ given by $R\langle b,a \rangle := \text{Path}_X(\text{if}_X(b;t,f),a)$. To obtain the relational interpretation of g at R , we introduce a dimension \mathbf{x} and apply g at its Gel-type.

$$g(\text{Gel}_{\mathbf{x}}(\text{bool}, X, R)) \in \text{Gel}_{\mathbf{x}}(\text{bool}, X, R) \rightarrow \text{Gel}_{\mathbf{x}}(\text{bool}, X, R) \rightarrow \text{Gel}_{\mathbf{x}}(\text{bool}, X, R)$$

Next, we apply this to the related pairs (true, t) and (false, f) , in the form of $\text{gel}_{\mathbf{x}}$ terms.

$$g(\text{Gel}_{\mathbf{x}}(\text{bool}, X, R))(\text{gel}_{\mathbf{x}}(\text{true}, t, \lambda _ . t))(\text{gel}_{\mathbf{x}}(\text{false}, f, \lambda _ . f)) \in \text{Gel}_{\mathbf{x}}(\text{bool}, X, R)$$

Call this term W . If we substitute $\mathbf{0}$ for \mathbf{x} in W , each Gel or gel term steps to its first argument; thus $W\langle \mathbf{0}/\mathbf{x} \rangle = g(\text{bool})(\text{true})(\text{false})$. Likewise, $W\langle \mathbf{1}/\mathbf{x} \rangle = gXtf$. The term $\text{ungel}(\mathbf{x}.W)$ is then a proof that these are related by R , which is precisely what we need. ◀

We can also characterize the bridges in `bool`. Intuitively, these are all trivial: `bool`'s only elements are the zero-dimensional `true` and `false`. To show this, we use parametricity, an interesting parallel with the use of univalence for characterizing paths in HITs.

► **Theorem 4.** *For any $b_0, b_1 : \text{bool}$, we have $\text{Bridge}_{\text{bool}}(b_0, b_1) \simeq \text{Path}_{\text{bool}}(b_0, b_1)$.*

Sketch. We will only construct the forward map; for a full proof, see [15, Theorem 11.5]. Let $q : \text{Bridge}_{\text{bool}}(b_0, b_1)$. Given \mathbf{x} , we have $P_{\mathbf{x}} := \text{Gel}_{\mathbf{x}}(\text{bool}, \text{bool}, \text{Path}_{\text{bool}}(-, -))$ corresponding to the identity relation on `bool`, as well as a map $F_{\mathbf{x}} : \text{bool} \rightarrow P_{\mathbf{x}}$ defined as follows.

$$F_{\mathbf{x}} := \lambda b.\text{if}_{P_{\mathbf{x}}}(b; \text{gel}_{\mathbf{x}}(\text{true}, \text{true}, \lambda _ . \text{true}), \text{gel}_{\mathbf{x}}(\text{false}, \text{false}, \lambda _ . \text{false}))$$

Note that $F_{\mathbf{0}} = F_{\mathbf{1}} = \lambda b.\text{if}_{\text{bool}}(b; \text{true}, \text{false})$. By applying F pointwise to q , we thus obtain $\lambda^2 \mathbf{x}.F_{\mathbf{x}}(q@_{\mathbf{x}}) \in \text{Bridge}_{\mathbf{x}.P_{\mathbf{x}}}(\text{if}_{\text{bool}}(b_0; \text{true}, \text{false}), \text{if}_{\text{bool}}(b_1; \text{true}, \text{false}))$. It is easy to show that for any $b : \text{bool}$, there is a path from $\text{if}_{\text{bool}}(b; \text{true}, \text{false})$ to b ; using coercion, we obtain some $T \in \text{Bridge}_{\mathbf{x}.P_{\mathbf{x}}}(b_0, b_1)$. Applying ungel gives an element of $\text{Path}_{\text{bool}}(b_0, b_1)$. ◀

The definition of the forward map uses parametricity; showing that it is an isomorphism uses *iterated parametricity*, which is to say two-bridge-dimensional types. Just as a one-dimensional bridge corresponds to a relation indexed by its boundary (the two endpoint types), relativity and the characterization of bridges at function type suffice to show that two-dimensional bridges satisfy the same characterization, the boundary now being given by four types and four relations in a square shape.

4.2 Bridge-discrete types

The booleans are one example of a *bridge-discrete* type, a type with trivial bridge structure.

► **Definition 5.** A type A is bridge-discrete when for all pairs $a_0, a_1 : A$, the canonical map $\lambda p. \text{coe}_{x. \text{Bridge}_A(a_0, p @ x)}^{0 \rightsquigarrow 1}(\lambda^2 _ . a_0) \in \text{Path}_A(a_0, a_1) \rightarrow \text{Bridge}_A(a_0, a_1)$ is an isomorphism. We write $\text{isBDisc}(A)$ for the type of proofs that A is bridge-discrete.

To show that A is bridge-discrete, it suffices to exhibit *any* family of isomorphisms between Path_A and Bridge_A [15, Corollary 10.7]. The type $\text{isBDisc}(A)$ is a *homotopy proposition* [30, §3.3]: all proofs of $\text{isBDisc}(A)$ are equal up to a path.

Bridge-discrete types are worth identifying because they provide an analogue to the *identity extension lemma*, a standard lemma in parametricity stating that the relational interpretation of a closed type is its identity relation. This does not hold in our theory, in the sense that a homogeneous bridge type $\text{Bridge}_A(M_0, M_1)$ is not necessarily isomorphic to $\text{Path}_A(M_0, M_1)$ (take $A = \mathcal{U}$). However, we can instead work by inserting bridge-discreteness hypotheses where the lemma would be required. For example, in imitation of Abel et al. [1], we can prove that path equality in a bridge-discrete type is isomorphic to Leibniz equality.

► **Proposition 6** ([15, §11.2]). *Let A be bridge-discrete. For any $a_0, a_1 : A$, $\text{Path}_A(a_0, a_1)$ is isomorphic to $(P : A \rightarrow \mathcal{U}) \rightarrow Pa_0 \rightarrow Pa_1$.*

Fortunately, most type constructors preserve bridge-discreteness. We can check that $\mathcal{U}_{\text{BDisc}} := (X : \mathcal{U}) \times \text{isBDisc}(X)$ is closed under almost all the type formers, with the obvious exception of universes.

► **Proposition 7.** *The sub-universe $\mathcal{U}_{\text{BDisc}}$ is closed under product, function, Path-, and Bridge-types. It is univalent and relativistic (i.e., closed under \mathbb{V} - and Gel-types).*

Proof. For the first statement, see [15, Lemma 10.9]. Any sub-universe of a univalent universe carved out by a proposition is univalent (see [30, Lemma 3.5.1]). For the proof of relativity, see [15, Theorem 10.12]. ◀

Per Section 4.1, we also expect that $\mathcal{U}_{\text{BDisc}}$ is closed under inductive types. Proposition 7 implies that we can also use parametricity in $\mathcal{U}_{\text{BDisc}}$: for example, one can repeat the proof of Theorem 3 to show that `bool` is isomorphic to $(X : \mathcal{U}) \rightarrow \text{isBDisc}(X) \rightarrow X \rightarrow X \rightarrow X$.

Finally, we observe as a simple consequence of Theorem 4 that the excluded middle for homotopy propositions [30, §3.4] is refuted by internal parametricity. (The excluded middle for *all* types is already refuted by univalence [30, Corollary 3.27].)

► **Definition 8.** Write $\text{isProp}(A) := (a_0, a_1 : A) \rightarrow \text{Path}_A(a_0, a_1)$. The excluded middle for homotopy propositions is $\text{LEM} := (X : \mathcal{U}) \rightarrow \text{isProp}(X) \rightarrow (b : \text{bool}) \times \text{if}_{\mathcal{U}}(b; X, \neg X)$.

► **Lemma 9.** *If A is bridge-discrete, then any function $f : \mathcal{U} \rightarrow A$ is constant.*

Proof. For any pair of types $X, Y : \mathcal{U}$, we have a bridge $B := \lambda^2 \mathbf{x}. \text{Gel}_{\mathbf{x}}(X, Y, \dots, \perp)$ between them, thus a bridge $\lambda^2 \mathbf{x}. f(B @ \mathbf{x}) : \text{Bridge}_A(fX, fY)$ between their images by f , thus a path from fX to fY by bridge-discreteness of A . ◀

► **Theorem 10.** *There is a term of type $\text{LEM} \rightarrow \perp$.*

Proof. We refute the *weak excluded middle* $\text{WLEM} := (X : \mathcal{U}) \rightarrow (b : \text{bool}) \times \text{if}_{\mathcal{U}}(b; \neg X, \neg \neg X)$, the special case of LEM that decides negated types. (Any negated type is a homotopy proposition.) Let $f : \text{WLEM}$. Then $\lambda X. \text{fst}(fX) \in \mathcal{U} \rightarrow \text{bool}$. By Theorem 4 and Lemma 9, this map is constant. But $\text{fst}(f\perp)$ and $\text{fst}(f\top)$ cannot be equal, so we have a contradiction. ◀

4.3 The smash product

We adopt the convention of writing $A_* : \mathcal{U}_* := (X : \mathcal{U}) \times X$ for a pointed type, $A := \text{fst}(A_*)$ for its underlying type, and $a_0 := \text{snd}(A_*)$ for its basepoint. Given $A_*, B_* \in \mathcal{U}_*$, we write $A_* \rightarrow_* B_* := (f : A \rightarrow B) \times \text{Path}_B(fa_0, b_0)$ for the type of basepoint-preserving functions from A_* to B_* . Given $A_*, B_* : \mathcal{U}_*$, their *smash product* is the following HIT.

```

data  $A_* \wedge B_* : \mathcal{U}$  where
| pair( $a : A, b : B$ ) :  $A_* \wedge B_*$ 
| basel :  $A_* \wedge B_*$ 
| baser :  $A_* \wedge B_*$ 
| gluel( $b : B, x : \mathbb{I}$ ) :  $A_* \wedge B_*$  [ $x = 0 \hookrightarrow \text{basel} \mid x = 1 \hookrightarrow \text{pair}(a_0, b)$ ]
| gluer( $a : A, x : \mathbb{I}$ ) :  $A_* \wedge B_*$  [ $x = 0 \hookrightarrow \text{baser} \mid x = 1 \hookrightarrow \text{pair}(a, b_0)$ ]

```

In words, the smash product of A_* and B_* is the cartesian product of their underlying types modulo the relation equating all pairs of the form (a_0, b) or (a, b_0) . The smash product can itself be made a pointed type $A_* \wedge_* B_*$ with basepoint $\text{pair}(a_0, b_0)$.

Our goal is to characterize the polymorphic pointed endofunctions on n -ary smash products. Here, we will only consider the binary case, and we will only sketch the proof. For the binary case, we give detailed pen-and-paper proofs of the arguments that use parametricity directly in [15, Appendix C], and we have formalized the purely cubical arguments in the `redtt` cubical proof assistant [29, `cool.smash`].

► **Theorem 11.** *Any function $f_* : (X_*, Y_* : \mathcal{U}_*) \rightarrow X_* \wedge_* Y_* \rightarrow_* X_* \wedge_* Y_*$ is connected by a path to either the polymorphic identity or the polymorphic constant function.*

That we cannot squeeze a complete proof into this space may seem to undermine our case for parametricity's usefulness. However, our argument is not that it is *easy*, but that it *scales*. After establishing the above, we will argue that no combinatorial explosion results from generalizing to n -ary smash products, in contrast to the more direct approaches.

► **Definition 12.** *Given $f : A \rightarrow B$, write $\text{Gr}_r(A, B, f) := \text{Gel}_r(A, B, a.b.\text{Path}_B(fa, b))$. Given $f_* : A_* \rightarrow_* B_*$, define $\text{Gr}_r^*(A_*, B_*, f_*) := \langle \text{Gr}_r(A, B, f), \text{gel}_r(a_0, b_0, f_0) \rangle \in \mathcal{U}_*$.*

The smash product has an action: from $f_* : A_* \rightarrow_* C_*$ and $g_* : B_* \rightarrow_* D_*$, we obtain $f_* \wedge g_* \in A_* \wedge B_* \rightarrow C_* \wedge D_*$. The following extracts results from products of Gr -types.

► **Lemma 13** (Graph Lemma for \wedge). *Let pointed types $A_*, B_*, C_*, D_* : \mathcal{U}_*$ and pointed functions $f_* : A_* \rightarrow_* C_*$, $g_* : B_* \rightarrow_* D_*$ be given. For any \mathbf{x} , there is a map of type*

$$\text{Gr}_{\mathbf{x}}^*(A, C, f) \wedge \text{Gr}_{\mathbf{x}}^*(B, D, g) \rightarrow \text{Gr}_{\mathbf{x}}(A_* \wedge B_*, C_* \wedge D_*, f_* \wedge g_*)$$

equal to the identity function on $A_ \wedge B_*$ when $\mathbf{x} = \mathbf{0}$ and on $C_* \wedge D_*$ when $\mathbf{x} = \mathbf{1}$.*

Sketch. This is analogous to Theorem 4; we define the map by smash product induction. We use `extent` and `ungel` to extract relation witnesses in the `pair`, `gluel`, and `gluer` cases. ◀

► **Proposition 14.** *Any element of $\text{bool}_* \wedge \text{bool}_*$ is either `pair(true, true)` or `pair(false, false)`.*

► **Lemma 15.** *Any $f : (X_*, Y_* : \mathcal{U}_*) \rightarrow X \rightarrow Y \rightarrow X_* \wedge Y_*$ is connected by a path to either*

1. *the pairing function $\lambda\langle X, x_0 \rangle.\lambda\langle Y, y_0 \rangle.\lambda a.\lambda b.\text{pair}(a, b)$, or*
2. *the constant basepoint function $\lambda\langle X, x_0 \rangle.\lambda\langle Y, y_0 \rangle.\lambda_.\lambda_.\text{pair}(x_0, y_0)$.*

Proof. Let $X_*, Y_* : \mathcal{U}_*$, $a : X$, and $b : Y$ be given. We have a function $g_*^X \in \text{bool}_* \rightarrow_* X_*$ taking `true` to x_0 and `false` to a , likewise $g_*^Y \in \text{bool}_* \rightarrow_* Y_*$ taking `true` to y_0 and `false` to b .

Fix a fresh bridge dimension \mathbf{x} . By taking the Gel-types for the graphs of the above functions, we obtain pointed types $G_*^X := \text{Gr}_{\mathbf{x}}^*(\text{bool}, X, g_*^X)$ and $G_*^Y := \text{Gr}_{\mathbf{x}}^*(\text{bool}, Y, g_*^Y)$. We apply f with the elements of G^X and G^Y corresponding to a and b .

$$f(G_*^X)(G_*^Y)(\text{gel}_{\mathbf{x}}(\text{false}, a, \lambda\mathbb{I}-.a))(\text{gel}_{\mathbf{x}}(\text{false}, b, \lambda\mathbb{I}-.b)) \in G_*^X \wedge G_*^Y$$

At $\mathbf{x} = \mathbf{0}$, this is equal to $f(\text{bool}_*)(\text{bool}_*)(\text{false})(\text{false}) \in \text{bool}_* \wedge \text{bool}_*$; at $\mathbf{x} = \mathbf{1}$, it is equal to $fX_*Y_*ab \in X_* \wedge Y_*$. By Lemma 13, we obtain a term in $\text{Gr}_{\mathbf{x}}(\text{bool}_* \wedge \text{bool}_*, X_* \wedge Y_*, g_*^X \wedge g_*^Y)$ with the same endpoints. Applying `ungel`, we get a proof that these endpoints are in the graph of $g_*^X \wedge g_*^Y$, i.e., that $(g_*^X \wedge g_*^Y)(f(\text{bool}_*)(\text{bool}_*)(\text{false})(\text{false}))$ is path-equal to fX_*Y_*ab . By Proposition 14, $f(\text{bool}_*)(\text{bool}_*)(\text{false})(\text{false})$ has two possible values. If it is `pair(true, true)`, then fX_*Y_*ab must be `pair(x0, y0)`; if it is `pair(false, false)`, then fX_*Y_*ab is `pair(a, b)`. ◀

Sketch of Theorem 11. To characterize $f_* : (X_*, Y_* : \mathcal{U}_*) \rightarrow X_* \wedge_* Y_* \rightarrow_* X_* \wedge_* Y_*$, we must characterize its behavior on each constructor, as well as the proof that it preserves the basepoint of $X_* \wedge_* Y_*$. Given X_*, Y_* , write fX_*Y_* for the function underlying $f_*X_*Y_*$.

Write $P := (X_*, Y_* : \mathcal{U}_*) \rightarrow X \rightarrow Y \rightarrow X_* \wedge Y_*$. First, we isolate the behavior of f_* on the `pair` constructor: $\lambda X_*.\lambda Y_*.\lambda a.\lambda b.fX_*Y_*(\text{pair}(a, b)) \in P$. By Lemma 15, this is one of two functions. We aim to show that this is the only degree of freedom available to f_* .

The values of f on the `basel` and `baser` constructors are uniquely determined up to a path by the fact that $f_*X_*Y_*$ is basepoint-preserving, as `basel` and `baser` are connected to the basepoint of $X_* \wedge_* Y_*$ by `gluel(y0, -)` and `gluer(x0, -)` respectively.

For `gluel`, we consider the term $H := \lambda\mathbb{I}x.\lambda X_*.\lambda Y_*.\lambda a.\lambda b.fX_*Y_*(\text{gluel}(b, x))$, which is a path in P from $\lambda X_*.\lambda Y_*.\lambda a.\lambda b.fX_*Y_*(\text{basel})$ to $\lambda X_*.\lambda Y_*.\lambda a.\lambda b.fX_*Y_*(\text{pair}(x_0, b))$. By Lemma 15, we know that P is isomorphic to `bool`, which means in particular that it is a *homotopy set*: its path types are all homotopy propositions. So H , and therefore the behavior of f_* on `gluel` terms, is uniquely determined (up to a path). The same applies to `gluer`.

Finally, write $f_0 : (X_*, Y_* : \mathcal{U}_*) \rightarrow \text{Path}_{X_* \wedge Y_*}(fX_*Y_*(\text{pair}(x_0, y_0)), \text{pair}(x_0, y_0))$ for the proof that f preserves the basepoint of $X_* \wedge_* Y_*$. As with `gluel`, we prove that f_0 is uniquely determined by recasting it as a path in P , namely the path $\lambda x.\lambda X_*.\lambda Y_*.\lambda a.\lambda b.f_0X_*Y_*@x$ that connects $\lambda X_*.\lambda Y_*.\lambda a.\lambda b.fX_*Y_*(\text{pair}(x_0, y_0))$ to $\lambda X_*.\lambda Y_*.\lambda a.\lambda b.\text{pair}(x_0, y_0)$. ◀

To prove the n -ary generalization of this theorem, we can proceed by an inductive argument, showing for each $i \leq n$ that there are exactly two maps of the following type.

$$(X_{1*}, \dots, X_{n*} : \mathcal{U}_*) \rightarrow X_1 \rightarrow \dots \rightarrow X_{n-i} \rightarrow (X_{n-i+1*} \wedge_* \dots \wedge_* X_{n*}) \rightarrow \bigwedge_{*i} X_{i*}$$

For the base case $i = 0$, we use an easy generalization of Lemma 15; for the inductive step, the argument in the proof of Theorem 11. What is key is that we never use an *iterated* induction argument on the elements of stacked smash products, so we never find ourselves dealing with a two-dimensional case like that of $\text{gluel}(\text{gluel}(c, x), y) \in X_* \wedge (Y_* \wedge Z_*)$.

To see how we can apply the theorem, consider the case of commutativity. If we have $K : (X_*, Y_* : \mathcal{U}) \rightarrow X_* \wedge_* Y_* \rightarrow_* Y_* \wedge_* X_*$, then $\lambda X_*.\lambda Y_* . K_{X_*, Y_*} \circ K_{Y_*, X_*}$ is a polymorphic endofunction on smash products. By Theorem 11, we can show it is the identity simply by showing it is not constant, which we can do by instantiating X_*, Y_* with `bool*`, `bool*` and testing it on `pair(false, false)`. If it is indeed non-constant, we see that K is an isomorphism. Similar techniques show that any non-constant associator is an isomorphism and satisfies Mac Lane's pentagon.

5 Computational meaning of the judgments

We now explain more precisely the meanings of the typing judgments. We follow the work of Angiuli et al. [3]. The central idea is that well-typed terms need not only evaluate to values, but evaluate in a way that is coherent with respect to substitution of dimensions.

First, we have an operational semantics, specified by judgments $M \text{ val}$ and $M \mapsto^* M'$. We write $M \Downarrow V$ when $M \mapsto^* V$ and $V \text{ val}$. The “closed” types and terms – those to which the operational semantics applies – are those whose free variables are all dimensions, i.e., those well-formed in some $\Phi\Psi \text{ ctx}$ where $\Phi = \mathbf{x}_1 : \mathbf{2}, \dots, \mathbf{x}_n : \mathbf{2}$ and $\Psi = y_1 : \mathbb{I}, \dots, y_m : \mathbb{I}$. A *type system* on this language is a five-place relation $\tau(\Phi, \Psi, A_0, A'_0, \varphi)$. It specifies, for each context $\Phi\Psi$, those values A_0 and A'_0 that are equal *value types* in that context, and associates to each such pair a partial equivalence relation (PER) φ on values in context $\Phi\Psi$. To be a type system, τ is required to satisfy laws ensuring symmetry, transitivity, and so forth. To interpret the type formers described in Sections 2 and 3, we can define an appropriate τ by a fixed-point construction populating it with the various constructors.

Two terms A and A' in context $\Phi\Psi$ are then *equal types* when they evaluate to equal type values in a way that commutes with dimension substitution.

► **Definition 16.** *Given τ , define a five-place relation $\text{PTY}(\tau)(\Phi, \Psi, A, A', \alpha)$ on context $\Phi\Psi$, terms A, A' , and families $(\alpha_\psi)_{\psi: \Phi'\Psi' \rightarrow \Phi\Psi}$ indexed by substitutions into $\Phi\Psi$, as follows. $\text{PTY}(\tau)(\Phi, \Psi, A, A', \alpha)$ holds when for all $\psi_1 : \Phi_1\Psi_1 \rightarrow \Phi\Psi$ and $\psi_2 : \Phi_2\Psi_2 \rightarrow \Phi_1\Psi_1$, we have*

$$A\psi_1 \Downarrow A_1 \quad A_1\psi_2 \Downarrow A_2 \quad A\psi_1\psi_2 \Downarrow A_{12} \quad A'\psi_1 \Downarrow A'_1 \quad A'_1\psi_2 \Downarrow A'_2 \quad A'\psi_1\psi_2 \Downarrow A'_{12}$$

with $\tau(\Phi_2, \Psi_2, V, V', \alpha_{\psi_1\psi_2})$ for all $V \in \{A_2, A_{12}\}$ and $V' \in \{A'_2, A'_{12}\}$.

We write $\llbracket A \rrbracket$ for the α such that $\text{PTY}(\tau)(\Phi_2, \Psi_2, A, A, \alpha)$ when it exists; the laws required of type systems ensure its uniqueness. We now say that $\Phi\Psi \gg A = A'$ type when $\text{PTY}(\tau)(\Phi, \Psi, A, A', \alpha)$ for some α (satisfying a certain coherence condition). As a special case, we say that $\Phi\Psi \gg A$ type when $\Phi\Psi \gg A = A$ type. Element equality is defined analogously: M and M' are equal in A when they coherently evaluate to equal values in $\llbracket A \rrbracket$. Finally, the closed judgments are extended to open judgments by functionality: $\Gamma \gg A = A'$ type holds when $\Phi\Psi \gg A\gamma = A'\gamma'$ type for all $\Phi\Psi \gg \gamma = \gamma' \in \Gamma$.

By definition of $\Phi\Psi \gg M \in \text{bool}$, we obtain a canonicity theorem.

► **Theorem 17 (Canonicity).** *If $\Phi\Psi \gg M \in \text{bool}$, then either $M \mapsto^* \text{true}$ or $M \mapsto^* \text{false}$. Additionally, either $\Phi\Psi \gg M = \text{true} \in \text{bool}$ or $\Phi\Psi \gg M = \text{false} \in \text{bool}$.*

6 Presheaf model

As we have said, the rules presented in Sections 2 and 3 can be used as a formalism for reasoning in parametric cubical type theory. Following prior work on cubical type theory [17, 2] and internal parametricity [6], this formalism also supports a presheaf semantics.

The two base categories at play, defined using the notation of Buchholtz and Morehouse [13], are the *BCH cube category* $\mathfrak{CB} := \mathbb{C}_{(\text{we}, \cdot)}$ and the *cartesian cube category* $\mathfrak{CP} := \mathbb{C}_{(\text{wec}, \cdot)}$. (Here w, e, and c stand for weakening, exchange, and contraction respectively.) We model parametric cubical type theory in the category $\mathcal{C} := \text{Set}^{(\mathfrak{CB} \times \mathfrak{CP})^{\text{op}}}$ of presheaves on the product of the two. We can immediately obtain an interpretation of the standard and cubical type formers (including the universe) by applying a result of Angiuli et al. [2, Theorem 1]. For the interval, we take $(\cdot, x : \mathbb{I}) \in \mathfrak{CB} \times \mathfrak{CP}$; for the generating cofibrations $\text{Cof} \subseteq \Omega_{\text{dec}}$, we take finite unions of equations of the form $r = s$ and $\mathbf{r} = \boldsymbol{\varepsilon}$. It is straightforward to check that these choices satisfy the axioms required to apply their theorem.

On the parametric side, we follow the BCH model. Given a context Γ interpreted as some $\llbracket \Gamma \rrbracket : \mathcal{C}$, its extension $\Gamma, \mathbf{x} : \mathbf{2}$ is interpreted as the separated product $\llbracket \Gamma \rrbracket \otimes \mathbf{y}(\mathbf{x} : \mathbf{2}, \cdot)$, whose elements at $(\Phi, \Psi) \in \mathbb{D}_{\mathbb{B}} \times \mathbb{D}_{\mathbb{P}}$ are pairs (γ, \mathbf{r}) with $\mathbf{r} \in \Phi \cup \{\mathbf{0}, \mathbf{1}\}$ and $\gamma \in \llbracket \Gamma \rrbracket(\Phi \setminus \{\mathbf{r}\}, \Psi)$. Bridge-types are interpreted à la BCH path types; the inclusion of equations $\mathbf{r} = \varepsilon$ in Cof ensures that they support coercion and composition. Gel-types can be interpreted similarly to BCH G-types (though coercion and composition are much simpler in our case).

One advantage of univalence is that we can obtain a relativistic universe without replacing sets with *I-sets*, Bernardy et al. do [6]. In their theory, the isomorphism implemented by `gel` and `ungel` is replaced by an equality $\text{Bridge}_{\mathbf{x}, \text{Gel}_{\mathbf{x}}(A, B, R)}(M, N) = R\langle M, N \rangle$. To ensure that the interpretations of these types have *exactly* the same elements, sets are everywhere replaced with *I-sets*. In our notation, these would be Φ -sets: for $\Phi \in \mathbb{D}_{\mathbb{B}}$, a Φ -element is a family indexed by subcontexts $\Phi' \subseteq \Phi$, and a Φ -set is a set of Φ -elements. Interpreting types as families of *I-sets* makes it possible to give an interpretation of Gel- and Bridge-types that validates the above equation. In our work, on the contrary, the equality is replaced by an isomorphism, obviating the need for *I-sets*. By exploiting univalence, we can still obtain a path and so establish the target isomorphism between $\text{Bridge}_{\mathcal{U}}(A, B)$ and $A \times B \rightarrow \mathcal{U}$.

7 Related and future work

Our parametric cubical type theory is, for the most part, simply the union of Angiuli et al.'s cartesian cubical type theory [3] and Bernardy et al.'s internally parametric type theory [6]. We work with binary rather than unary parametricity, but as Bernardy et al. remark, this requires only cosmetic changes. There is little interaction between the two halves of the theory; we need only to check that coercion and composition can be defined for the new types. For bridge types, this requires a minor change to the definition of composition, the addition of $\mathbf{r} = \varepsilon$ tube equations. As discussed in Section 6, our Gel-types satisfy fewer equations than the corresponding types in [6]; accordingly, our proof of relativity is novel.

A second approach to internal parametricity has been developed by Nuyts, Vezzosi, and Devriese [23]. Like ours, their system is based on *paths* and *bridges*. Where ours are almost entirely separate, however, theirs are connected by a modality, which mediates between variable uses in *continuous* and *parametric* positions. Intuitively, their goal is to internalize the independence of element-level calculation from terms at the type level, whereas ours is merely to internalize the relational interpretation. The divergence of aims leads to very different considerations; in particular, their bridge dimensions are structural. In later work [22], Nuyts and Devriese generalize from paths and bridges to an infinite tower of relations.

Parametric cubical type theory also strongly resembles Riehl and Shulman's *directed type theory* [27]. Where ours has semantics in presheaves on $\mathbb{D}_{\mathbb{B}} \times \mathbb{D}_{\mathbb{P}}$, theirs is aimed at presheaves on $\Delta \times \Delta$, two copies of the simplex category. In both cases, one half of the base category is used for equality, while the other endows types with a relational structure. For directed type theory, the goal is to identify those types for which the relational structure is actually a category structure, meaning that concatenable bridges have path-unique composites. Interestingly, their model does not appear to admit a relativistic universe [26, §2].

Stepping outside the realm of internalization, there has been general interest in higher-dimensional generalizations of parametricity, for example in the work of Atkey et al. [4], Benton et al. [5], Ghani et al. [19], and Sojakova and Johann [28]. Atkey et al.'s use of reflexive graphs to construct a parametric model with a discrete universe is to Bernardy-Moulin-style internal parametricity as the Hofmann-Streicher groupoid model [20] is to cubical type theory.

One unsolved problem looms large: to what extent do results in parametric type theory translate into ordinary type theory? Naturally, not all theorems translate – $(f:\mathcal{U} \rightarrow \text{bool}) \rightarrow \text{Path}_{\text{bool}}(f\top, f\perp)$ is provable in parametric type theory but refuted in classical cubical sets – but one reasonable conjecture is that proofs $\Gamma \gg M \in A$ translate when Γ and A use no function types. It appears fairly straightforward to obtain results of this kind in semantics, but syntactic results are much murkier; similar conservativity questions for homotopy and cubical type theory over ordinary type theory remain open.

References

- 1 Andreas Abel, Jesper Cockx, Dominique Devriese, Amin Timany, and Philip Wadler. Leibniz Equality is Isomorphic to Martin-Löf Identity, Parametrically. Submitted to the Journal of Functional Programming, February 2018. URL: <https://jesper.sikanda.be/files/leibniz-equality.pdf>.
- 2 Carlo Angiuli, Guillaume Brunerie, Thierry Coquand, Kuen-Bang Hou (Favonia), Robert Harper, and Daniel R. Licata. Syntax and Models of Cartesian Cubical Type Theory. Unpublished draft, February 2019. URL: <https://github.com/dlicata335/cart-cube>.
- 3 Carlo Angiuli, Kuen-Bang Hou (Favonia), and Robert Harper. Cartesian Cubical Computational Type Theory: Constructive Reasoning with Paths and Equalities. In *27th EACSL Annual Conference on Computer Science Logic, CSL 2018, September 4-7, 2018, Birmingham, UK*, pages 6:1–6:17, 2018. doi:10.4230/LIPIcs.CSL.2018.6.
- 4 Robert Atkey, Neil Ghani, and Patricia Johann. A relationally parametric model of dependent type theory. In *The 41st Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL '14, San Diego, CA, USA, January 20-21, 2014*, pages 503–516, 2014. doi:10.1145/2535838.2535852.
- 5 Nick Benton, Martin Hofmann, and Vivek Nigam. Abstract effects and proof-relevant logical relations. In *The 41st Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL '14, San Diego, CA, USA, January 20-21, 2014*, pages 619–632, 2014. doi:10.1145/2535838.2535869.
- 6 Jean-Philippe Bernardy, Thierry Coquand, and Guilhem Moulin. A Presheaf Model of Parametric Type Theory. *Electr. Notes Theor. Comput. Sci.*, 319:67–82, 2015. doi:10.1016/j.entcs.2015.12.006.
- 7 Jean-Philippe Bernardy, Patrik Jansson, and Ross Paterson. Parametricity and dependent types. In *ICFP 2010, Baltimore, Maryland, USA, September 27-29, 2010*, pages 345–356, 2010.
- 8 Jean-Philippe Bernardy and Guilhem Moulin. A Computational Interpretation of Parametricity. In *LICS 2012, Dubrovnik, Croatia, June 25-28, 2012*, pages 135–144, 2012. doi:10.1109/LICS.2012.25.
- 9 Jean-Philippe Bernardy and Guilhem Moulin. Type-theory in color. In *ICFP 2013, Boston, MA, USA - September 25 - 27, 2013*, pages 61–72, 2013. doi:10.1145/2500365.2500577.
- 10 Marc Bezem, Thierry Coquand, and Simon Huber. A Model of Type Theory in Cubical Sets. In *19th International Conference on Types for Proofs and Programs, TYPES 2013, April 22-26, 2013, Toulouse, France*, pages 107–128, 2013. doi:10.4230/LIPIcs.TYPES.2013.107.
- 11 Marc Bezem, Thierry Coquand, and Simon Huber. The Univalence Axiom in Cubical Sets. *J. Autom. Reasoning*, 63(2):159–171, 2019. doi:10.1007/s10817-018-9472-6.
- 12 Guillaume Brunerie. Computer-generated proofs for the monoidal structure of the smash product. *Homotopy Type Theory Electronic Seminar Talks*, November 2018. URL: <https://www.uwo.ca/math/faculty/kapulkin/seminars/hottest.html>.
- 13 Ulrik Buchholtz and Edward 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*, pages 77–92, 2017. doi:10.1007/978-3-319-57418-9_5.

- 14 Evan Cavallo and Robert Harper. Higher inductive types in cubical computational type theory. *PACMPL*, 3(POPL):1:1–1:27, 2019. doi:10.1145/3290314.
- 15 Evan Cavallo and Robert Harper. Parametric Cubical Type Theory, 2019. arXiv:1901.00489.
- 16 James Cheney. A dependent nominal type theory. *Logical Methods in Computer Science*, 8(1), 2012. doi:10.2168/LMCS-8(1:8)2012.
- 17 Cyril Cohen, Thierry Coquand, Simon Huber, and Anders Mörtberg. Cubical Type Theory: A Constructive Interpretation of the Univalence Axiom. In *21st International Conference on Types for Proofs and Programs, TYPES 2015, May 18-21, 2015, Tallinn, Estonia*, pages 5:1–5:34, 2015. doi:10.4230/LIPIcs.TYPES.2015.5.
- 18 Thierry Coquand, Simon Huber, and Anders Mörtberg. On Higher Inductive Types in Cubical Type Theory. In *LICS 2018, Oxford, UK, July 9-12, 2018*, 2018. doi:10.1145/3209108.3209197.
- 19 Neil Ghani, Patricia Johann, Fredrik Nordvall Forsberg, Federico Orsanigo, and Tim Revell. Bifibrational Functorial Semantics of Parametric Polymorphism. *Electr. Notes Theor. Comput. Sci.*, 319:165–181, 2015. doi:10.1016/j.entcs.2015.12.011.
- 20 Martin Hofmann and Thomas Streicher. The groupoid interpretation of type theory. In *Twenty-five years of constructive type theory (Venice, 1995)*, volume 36 of *Oxford Logic Guides*, pages 83–111. Oxford Univ. Press, New York, 1998.
- 21 Guilhem Moulin. *Internalizing Parametricity*. PhD thesis, Chalmers University of Technology, Gothenburg, Sweden, 2016. URL: <https://research.chalmers.se/en/publication/235758>.
- 22 Andreas Nuyts and Dominique Devriese. Degrees of Relatedness: A Unified Framework for Parametricity, Irrelevance, Ad Hoc Polymorphism, Intersections, Unions and Algebra in Dependent Type Theory. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018, Oxford, UK, July 09-12, 2018*, pages 779–788, 2018. doi:10.1145/3209108.3209119.
- 23 Andreas Nuyts, Andrea Vezzosi, and Dominique Devriese. Parametric quantifiers for dependent type theory. *PACMPL*, 1(ICFP):32:1–32:29, 2017. doi:10.1145/3110276.
- 24 Ian Orton and Andrew M. Pitts. Axioms for Modelling Cubical Type Theory in a Topos. *Logical Methods in Computer Science*, 14(4), 2018. doi:10.23638/LMCS-14(4:23)2018.
- 25 John C. Reynolds. Types, Abstraction and Parametric Polymorphism. In *IFIP Congress*, pages 513–523, 1983.
- 26 Emily Riehl. On the directed univalence axiom. Talk slides, AMS Special Session on Homotopy Type Theory, Joint Mathematics Meetings, January 2018. URL: <http://www.math.jhu.edu/~eriehl/JMM2018-directed-univalence.pdf>.
- 27 Emily Riehl and Michael Shulman. A type theory for synthetic ∞ -categories. *Higher Structures*, 1(1):116–193, 2017. URL: https://journals.mq.edu.au/index.php/higher_structures/article/view/36.
- 28 Kristina Sojakova and Patricia Johann. A General Framework for Relational Parametricity. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018, Oxford, UK, July 09-12, 2018*, pages 869–878, 2018. doi:10.1145/3209108.3209141.
- 29 The RedPRL Development Team. redtt, 2018. URL: <https://github.com/RedPRL/redtt>.
- 30 The Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations of Mathematics*. <https://homotopytypetheory.org/book>, Institute for Advanced Study, 2013.
- 31 Floris van Doorn. *On the Formalization of Higher Inductive Types and Synthetic Homotopy Theory*. PhD thesis, Carnegie Mellon University, 2018.
- 32 Vladimir Voevodsky. The equivalence axiom and univalent models of type theory, 2014. Talk at CMU on February 4, 2010. arXiv:1402.5556.
- 33 Philip Wadler. Theorems for Free! In *FPCA 1989, London, UK, September 11-13, 1989*, pages 347–359, 1989. doi:10.1145/99370.99404.