




QUOTIENTS, INDUCTIVE TYPES, & QUOTIENT INDUCTIVE TYPES

MARCELO P. FIORE , ANDREW M. PITTS , AND S. C. STEENKAMP ^(ES) 

Department of Computer Science and Technology, University of Cambridge, United Kingdom
e-mail address: {marcelo.fiore, andrew.pitts, s.c.steenkamp}@cl.cam.ac.uk

ABSTRACT. This paper introduces an expressive class of indexed quotient-inductive types, called QWI types, within the framework of constructive type theory. They are initial algebras for indexed families of equational theories with possibly infinitary operators and equations. We prove that QWI types can be derived from quotient types and inductive types in the type theory of toposes with natural number object and universes, provided those universes satisfy the Weakly Initial Set of Covers (WISC) axiom. We do so by constructing QWI types as colimits of a family of approximations to them defined by well-founded recursion over a suitable notion of size, whose definition involves the WISC axiom. We developed the proof and checked it using the Agda theorem prover.

1. INTRODUCTION

Inductive types are an essential feature of many type theories and the proof assistants and programming languages that are based upon them. Homotopy Type Theory [Uni13] introduces a powerful extension to the notion of inductive type: higher inductive types (HITs). To define an ordinary inductive type one declares the ways in which its elements are constructed, and these constructions may depend inductively on smaller elements. To define a HIT one not only declares element constructors, but also declares equality constructors valued in identity types (possibly iterated ones), specifying how the element constructors (and possibly the equality constructors) are related. In the language of homotopy theory, these correspond to points, paths, surfaces, and so on.

In this paper, rather than using HoTT, we work in the simpler setting of Extensional Type Theory [Mar82], where any propositional equality is reflected as a definitional equality. However, our results also hold for an intensional type theory with the uniqueness of identity proofs axiom (UIP), as demonstrated by our Agda development which can be found at [FPS21]. In either case identity types are trivial in dimensions higher than one, so that any two proofs of identity are equal. Nevertheless, as Altenkirch and Kaposi [AK16] point out, HITs are still useful in such a one-dimensional setting. They introduced the term *quotient inductive type* (QIT) for this truncated form of HIT.

Key words and phrases: dependent type theory, higher inductive types, quotient types, well-founded relation, weakly initial set of covers, topos theory.

One specific advantage of QITs over inductive types is that they provide a natural way to study universal algebra in type theory, since general algebraic theories cannot be represented in an equation-free way just using inductive notions [MM16]. As an example, the free commutative monoid on carrier X is given by the quotient inductive type of finite multisets¹, $\mathbf{Bag} X$, with constructors:

$$\begin{aligned} & [] : \mathbf{Bag} X \\ & _ :: _ : X \rightarrow \mathbf{Bag} X \rightarrow \mathbf{Bag} X \\ & \mathbf{swap} : \prod_{x,y:X} \prod_{zs:\mathbf{Bag} X} x :: y :: zs = y :: x :: zs \end{aligned} \tag{1.1}$$

This can be seen as the element constructors for lists ($[]$ and $::$), along with an equality constructor \mathbf{swap} that transposes two elements and thereby allows all reorderings of a list to be identified. Notice that the \mathbf{swap} constructor lands in the identity type on $\mathbf{Bag} X$ (just written as $_ = _$), making it an *equation* constructor, compared to the *element* constructors, $[]$ and $::$, which are familiar from ordinary inductive type definitions. There is no need to include a constructor for the congruence property of $::$ since that is automatic from the congruence properties of identity types.

An alternative presentation of multisets, whose equations may be easier to work with, is:

$$\begin{aligned} & [] : \mathbf{Bag}' X \\ & _ :: _ : X \rightarrow \mathbf{Bag}' X \rightarrow \mathbf{Bag}' X \\ & \mathbf{comm} : \prod_{x,y:X} \prod_{as,bs,cs:\mathbf{Bag}' X} as = y :: cs \rightarrow x :: cs = bs \rightarrow x :: as = y :: bs \end{aligned} \tag{1.2}$$

Notice that the \mathbf{comm} equality constructor is *conditional* on two equations, $as = y :: cs$ and $x :: cs = bs$, in order to deduce the final equation $x :: as = y :: bs$. The QWI-types introduced in this paper are not able to encode such conditional QITs (at least, not obviously so); see the discussion in the Conclusion.

Given the parameter X , the types $\mathbf{Bag} X$ and $\mathbf{Bag}' X$ are single quotient-inductive types, rather than indexed families defined quotient-inductively. As an example of the latter, consider the go-to example of an inductively defined family, the vector type (length-indexed lists). This has a commutative variant, $\mathbf{AbVec} X n$ for $n : \mathbb{N}$, which makes it an example of an indexed QIT. It has constructors:

$$\begin{aligned} & [] : \mathbf{AbVec} X 0 \\ & _ :: _ : X \rightarrow \prod_{i:\mathbb{N}} \mathbf{AbVec} X i \rightarrow \mathbf{AbVec} X (i + 1) \\ & \mathbf{swap} : \prod_{x,y:X} \prod_{i:\mathbb{N}} \prod_{zs:\mathbf{AbVec} X i} x :: y :: zs = y :: x :: zs \end{aligned} \tag{1.3}$$

Our final example of a QIT is unordered countably-branching trees over X , called $\omega\mathbf{Tree} X$, with constructors:

$$\begin{aligned} & \mathbf{leaf} : \omega\mathbf{Tree} X \\ & \mathbf{node} : X \rightarrow (\mathbb{N} \rightarrow \omega\mathbf{Tree} X) \rightarrow \omega\mathbf{Tree} X \\ & \mathbf{perm} : \prod_{x:X} \prod_{b:\mathbb{N} \rightarrow \mathbb{N}} \prod_{b':\mathbf{Iso} b} \prod_{f:\mathbb{N} \rightarrow \omega\mathbf{Tree} X} \mathbf{node} x f = \mathbf{node} x (f \circ b) \end{aligned} \tag{1.4}$$

¹Other presentations of multisets are available, aside from the two presentations given here. Incidentally, the algebra of monoids has an equation-free presentation [Kel80, section 23], [Kel92] – the algebra of Lists – and this is why no equation for associativity is needed in either definition of multisets.

where elements of `islob` witness that b is a bijection. (As with `AbVec`, one could similarly consider a depth-indexed variant of `ωTree`.) The significance of this example is that the `node x _` and `perm x b b' _` constructors have infinite arity, making this an example of an *infinitary* QIT. This type is one of the original motivations for considering QITs [AK16], since they can enable constructive versions of structures that classically use non-constructive choice principles, as we explain next.

The examples of QITs in (1.1)–(1.3) only involve element constructors of *finite* arity. For example in (1.1), `[]` is nullary and for each $x, y : X$, `x :: _` and `swap x y _` are unary. Consequently `Bag X` is isomorphic to the type obtained from the ordinary inductive type of finite lists over X by quotienting by the congruence generated by `swap`. By contrast, (1.4) involves element and equality constructors with countably infinite arity. So if one first forms the ordinary inductive type of *ordered* (or planar) countably-branching trees (by dropping the equality constructor `perm` from the declaration) and then quotients by a suitable relation to get the equalities specified by `perm`, one appears to need the axiom of countable choice to be able to lift the `node` element constructor to the quotient [AK16, Section 2.2].

The construction of the Cauchy reals as a higher inductive-inductive type [Uni13, Section 11.3] provides a similar, but more complicated example where use of countable choice is avoided. So it seems that without some form of choice principle, quotient inductive types are strictly more expressive than the combination of ordinary inductive types with quotient types. Indeed, Lumsdaine and Shulman [LS19, Section 9] turn an infinitary equational theory due to Blass [Bla83, Section 9] into a higher-inductive type that cannot be proved to exist in ZF set theory without the Axiom of Choice. In this paper we show that in fact a much weaker and constructively acceptable form of choice is sufficient for constructing indexed quotient inductive types. This is the *Weakly Initial Set of Covers* (WISC) axiom, originally called the “Type Theoretic Collection Axiom” TTCA_f by Streicher [Str05a] and (in the context of constructive set theory) the “Axiom of Multiple Choice” by van den Berg and Moerdijk [vdBM14]. WISC is constructively acceptable in that it is known to hold in the internal logic of a wide range of toposes [vdBM14] (for example, all Grothendieck and all realizability toposes over the topos of classical sets), including those commonly used in the semantics of Type Theory. Section 4 reviews WISC and our motivation for using it herein.

We make two contributions:

- First we define a class of indexed quotient inductive types called *QWI-types* and give elimination and computation rules for them (Section 3). The usual W-types of Martin-Löf [Mar82] are inductive types giving the algebraic terms over a possibly infinitary signature. One specifies a QWI-type by giving a family of equations between such terms. So such types give initial algebras for (families of) possibly infinitary algebraic theories. They can encode a very wide range of examples of, possibly infinitary, indexed quotient inductive types (Section 7). In classical set theory with the Axiom of Choice, QWI-types can be constructed simply as quotients of the underlying Indexed W-type—hence the name.
- Second, our main result (Theorem 6.1) is that with only WISC it is still possible to construct QWI-types from inductive types and quotients in the constructive type theory of toposes, but not simply by quotienting a W-type. Instead, quotienting is interleaved with an inductive construction. To make sense of this and prove that the resulting type has the required universal property, we construct the type as the colimit of a family of approximations to it defined by well-founded recursion over a suitable notion of *size* (a constructive version of what classically would be accomplished with a sequence

of transfinite ordinal length). Our notion of size is developed in Section 5. Sizes are elements of a W-type equipped with the *plump* [Tay96] well-founded order; and WISC allows us to make the W-type big enough that a given polynomial endofunctor preserves size-indexed colimits (Corollary 5.10). This fact is then applied in Section 6 to form QWI-types as size-indexed colimits.

Note. This paper is a greatly revised and expanded version of our earlier paper [FPS20], which introduced QW-types (see Definition 3.2 which is the non-indexed version of Definition 3.5). There we made use of *sized types* [Abe12] as they are implemented in the Agda theorem prover [Agd20]. Though the results of that paper were formalised in Agda version 2.5.2, unfortunately, in the current version of Agda, 2.6.2, sized types are logically unsound. In this paper, not only do we consider the more expressive indexed form of QW-types, namely QWI-types, we also put the construction of QW-types on a firm semantic footing by avoiding Agda-style sized types (and positivity assumptions on quotient types, see Section 6.1) in favour of a well-founded notion of size built from WISC. The Agda formalization of the results in this paper is available at [FPS21].

2. TYPE THEORY

The results in this paper are proved in a version of Extensional Type Theory [Mar82] that is an internal language for toposes [Joh02] with natural numbers object and universes [Str05b]. We use this type theory in an informal way, in the style and notation of the HoTT Book [Uni13]. Thus dependent function types are written $\prod_{x:A} B x$ and non-dependent ones as either $A \rightarrow B$ or B^A . Dependent product types are written $\sum_{x:A} B x$ and non-dependent ones as $A \times B$. Coproduct types are written $A + B$ (with injections $\iota_1 : A \rightarrow A + B$ and $\iota_2 : B \rightarrow A + B$); and finite types are written $\mathbb{0}$ (with no elements), $\mathbb{1}$ (with a single element 0), $\mathbb{2}$ (with two elements 0, 1), etc. The identity type for $x : A$ and $y : A$ is $x =_A y$, or just $x = y$ when the type A is known from the context. When we need to refer to the judgement that x and y are definitionally equal, we write $x \equiv y$. Unlike in the HoTT Book [Uni13], the type $x = y$ is an extensional identity type and so it is inhabited iff $x \equiv y$ holds. In particular, proofs of identity are unique when they exist and so it makes sense to use McBride’s heterogeneous form of identity type [McB99] wherever possible. Given $x : A$ and $y : B$, we denote the heterogeneous identity type by $x == y$; thus this type is inhabited iff $A \equiv B$ and $x \equiv y$.

The type theory has universes of types, $\mathcal{U}_0 : \mathcal{U}_1 : \mathcal{U}_2, \dots$ which we write just as \mathcal{U} when the universe level is immaterial. We use Russell-style universes (there is no syntactic distinction between an element $A : \mathcal{U}$ and the corresponding type of its elements). Influenced by Agda (see below) and unlike the HoTT Book [Uni13], we do not assume universes are cumulative, but instead use Agda-style closure under \prod -types: if $A : \mathcal{U}_i$ and $B : A \rightarrow \mathcal{U}_j$, then $\prod_{x:A} B x : \mathcal{U}_{\max ij}$ (and similarly for \sum -types). Since we only consider toposes that have a natural numbers object, we can assume the universes are closed under forming not just W-types [MP00, Proposition 3.6], but also all inductively defined indexed families of types [GH04]. Indexed W-types are considered in more detail in the next section.

The lowest universe contains an impredicative universe \mathbf{Prop} of propositions, corresponding to the subobject classifier in a topos. \mathbf{Prop} contains the identity types, is closed under

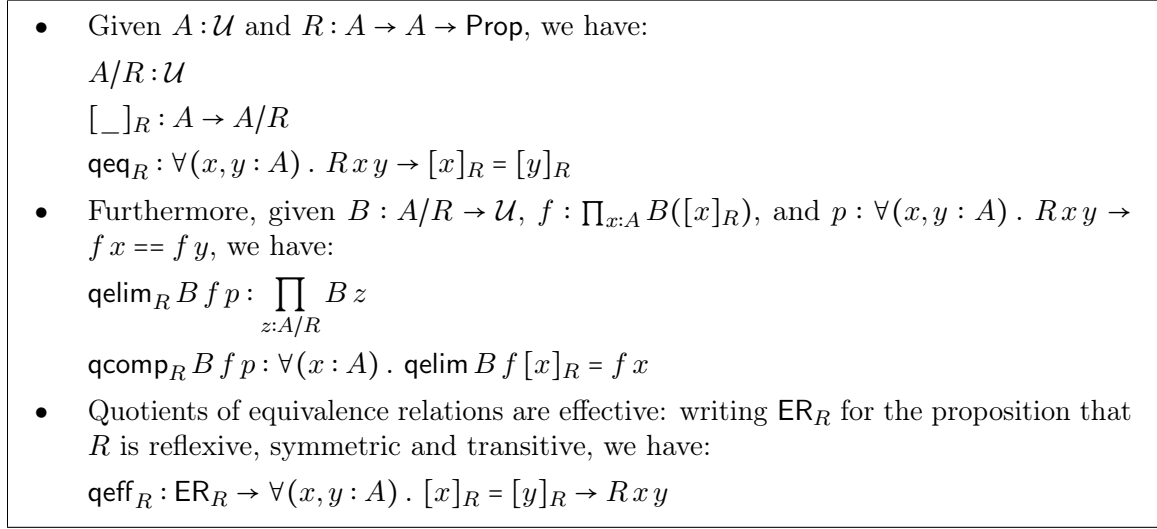


Figure 1: Quotient types

intuitionistic connectives ($\wedge, \vee, \rightarrow, \leftrightarrow$, etc.) and quantifiers ($\forall (x : A) . \phi(x)$ and $\exists (x : A) . \phi(x)$, for A in any universe), and satisfies propositional extensionality:

$$\mathbf{propext} : \forall (p, q : \mathbf{Prop}) . (p \leftrightarrow q) \rightarrow p = q \quad (2.1)$$

Being an extensional type theory, we also have function extensionality:

$$\mathbf{funext} : \forall \left(f, g : \prod_{x:A} B x \right) . (\forall (x : A) . f x = g x) \rightarrow f = g \quad (2.2)$$

Note that like the universes \mathcal{U}_i , \mathbf{Prop} is also a Russell-style universe, in that we do not make a notational distinction between a proposition $p : \mathbf{Prop}$ and the type of its proofs. Given $A : \mathcal{U}$ and $\phi : A \rightarrow \mathbf{Prop}$, we regard the comprehension type $\{x : A \mid \phi x\}$ in \mathcal{U} as synonymous with the dependent product $\sum_{x:A} \phi x$.

Toposes have coequalizers and effective epimorphisms [Joh02, A2.4]. Correspondingly the type theory contains quotient types, with notation as in Figure 1 (recall that $=$ stands for heterogeneous identity). These can be constructed in the usual way via equivalence classes, using \mathbf{Prop} 's impredicative quantifiers and the fact that toposes satisfy unique choice:

$$\mathbf{uniquechoice} : (\exists (x : A) . \forall (y : A) . x = y) \rightarrow A \quad (2.3)$$

Thus $\mathbf{uniquechoice}$ is a function mapping proofs that A has exactly one element to a name for that one element. From $\mathbf{uniquechoice}$ it follows that given $A : \mathcal{U}$, $B : \mathcal{U}^A$ and $\phi : \prod_{x:A} (B x \rightarrow \mathbf{Prop})$, from a proof of $\forall (x : A) \exists ! (y : B x) . \phi x y$ we get there is a unique $f : \prod_{x:A} B x$ such that $\forall (x : A) . \phi x (f x)$.

Agda development. We have developed and checked a version of the results in this paper using the Agda theorem prover [Agd20]. In particular our Agda development gives the full details of the construction of the indexed version of QW-types, whereas in the paper, for readability, we only give the construction in the non-indexed case (Section 6).

Being intensional and predicative, the type theory provided by Agda is weaker than the one described above, but can be soundly interpreted in it. One has to work harder

to establish the results with Agda, since two expressions that are definitionally equal in Extensional Type Theory may not be so in Agda and hence one has to produce a term in a corresponding identity type to prove them equal. On the other hand, when a candidate term is given, Agda can decide whether or not it is a correct proof, because validity of the judgements of the type theory it implements is decidable², in contrast with the situation for Extensional Type Theory [Hof95]. Our development is also made easier by extensive use of the predicative universes of proof-irrelevant propositions that feature in recent versions of Agda. Not only are any two proofs of such propositions definitionally equal, but inductively defined propositions (such as the well-founded ordering on sizes used in Section 6) can be eliminated in proofs using dependent pattern matching, which is a very great convenience. We still need these propositions to satisfy the extensionality (2.1) and unique choice (2.3) properties, so we add them as postulates in Agda. Also, the impredicative construction of quotient types is not available, so we get quotients as in Figure 1 by postulating them and using a user-declared rewrite to make their computation rule a definitional equality. Although function extensionality (2.2) is not provable in Agda’s core type theory, it becomes so once one has such quotient types. Our Agda development is available at [FPS21].

3. INDEXED POLYNOMIAL FUNCTORS AND EQUATIONAL SYSTEMS

In this section we introduce a class of indexed quotient inductive types, called *QWI-types*, which are indexed families of free algebras for possibly infinitary equational theories. To begin with we describe the simpler, non-indexed case (*QW-types*) and then treat the general, indexed case in Section 3.3.

3.1. W-types. We recall some facts about types of well-founded trees, the W-types of Martin-Löf [Mar82]. We take *signatures* (also known as *containers* [AAG05]) to be elements of the dependent product

$$\text{Sig} \stackrel{\text{def}}{=} \sum_{A:\mathcal{U}} (A \rightarrow \mathcal{U}) \quad (3.1)$$

So a signature is given by a pair $\Sigma = (A, B)$ consisting of a type $A : \mathcal{U}$ and a family of types $B : A \rightarrow \mathcal{U}$. Each such signature determines a polynomial endofunctor [GH04; AAG05] $S_\Sigma : \mathcal{U} \rightarrow \mathcal{U}$ whose value at $X : \mathcal{U}$ is the following dependent product

$$S_\Sigma(X) \stackrel{\text{def}}{=} \sum_{a:A} (B a \rightarrow X) \quad (3.2)$$

and whose action on a function $f : X \rightarrow Y$ in \mathcal{U} is the function $S_\Sigma f : S_\Sigma(X) \rightarrow S_\Sigma(Y)$ where

$$S_\Sigma f(a, b) \stackrel{\text{def}}{=} (a, f \circ b) \quad (a : A, b : B a \rightarrow X) \quad (3.3)$$

An S_Σ -algebra is by definition an element of the dependent product

$$\text{Alg}_\Sigma \stackrel{\text{def}}{=} \sum_{X:\mathcal{U}} (S_\Sigma(X) \rightarrow X) \quad (3.4)$$

S_Σ -algebra morphisms $(X, \alpha) \rightarrow (X', \alpha')$ are given by functions $h : X \rightarrow X'$ together with an element of the type

$$\text{isHom}_{\alpha, \alpha'} h \stackrel{\text{def}}{=} \prod_{a:A} \prod_{b:B a \rightarrow X} (\alpha'(a, h \circ b) = h(\alpha(a, b))) \quad (3.5)$$

Then the W-type W_Σ determined by a signature Σ is the underlying type of an initial object in the evident category of S_Σ -algebras. More generally, Dybjer [Dyb97] shows that the

²though not always feasibly so

initial algebra of any non-nested, strictly positive endofunctor on \mathcal{U} is given by a W -type; and Abbott, Altenkirch, and Ghani [AAG05] extend this to the case with nested uses of W -types as part of their work on containers. (These proofs take place in extensional type theory [Mar82], but work just as well in intensional type theory with uniqueness of identity proofs and function extensionality, which is what we use in Agda.)

More concretely, given a signature $\Sigma = (A, B)$, if one thinks of elements $a : A$ as names of operation symbols whose (not necessarily finite) arity is given by the type $B a : \mathcal{U}$, then the elements of the W -type W_Σ represent the closed algebraic terms (i.e. well-founded trees) over the signature. From this point of view it is natural to consider not only closed terms solely built up from operations, but also open terms additionally built up with variables drawn from some type V . As well as allowing operators of possibly infinite arity, we also allow terms involving possibly infinitely many variables (example (1.4) involves such terms). Categorically, the type $T_\Sigma(V)$ of such open terms is the free S_Σ -algebra on V and is another W -type, for the signature obtained from Σ by adding the elements of V as nullary operations. Nevertheless, it is convenient to give a direct inductive definition: the value of $T_\Sigma : \mathcal{U} \rightarrow \mathcal{U}$ at some some $V : \mathcal{U}$, is the inductively defined type with constructors

$$\begin{aligned} \eta &: V \rightarrow T_\Sigma V \\ \sigma &: S_\Sigma(T_\Sigma V) \rightarrow T_\Sigma V \end{aligned} \tag{3.6}$$

Given an S_Σ -algebra $(X, \alpha) : \mathbf{Alg}_\Sigma$ and a function $f : V \rightarrow X$, the unique morphism of S_Σ -algebras from the free S_Σ -algebra on V to (X, α) that extends f , $_ \gg= f : (T_\Sigma(V), \sigma) \rightarrow (X, \alpha)$, has underlying function $T_\Sigma(V) \rightarrow X$ mapping each $t : T_\Sigma(V)$ to the element $t \gg= f$ defined as follows by recursion on the structure of t :

$$\begin{aligned} \eta x \gg= f &\stackrel{\text{def}}{=} f x \\ \sigma(a, b) \gg= f &\stackrel{\text{def}}{=} \alpha(a, \lambda x. b x \gg= f) \end{aligned} \tag{3.7}$$

As the notation suggests, $\gg=$ is the Kleisli lifting operation (“bind”) for a monad structure on T_Σ . Indeed, T_Σ is the free monad on the endofunctor S_Σ and in particular the functorial action of T_Σ on $f : V \rightarrow V'$ yields the function $T_\Sigma f : T_\Sigma(V) \rightarrow T_\Sigma(V')$ given by

$$T_\Sigma f t \stackrel{\text{def}}{=} t \gg= (\eta \circ f) \quad (t : T_\Sigma(V)) \tag{3.8}$$

3.2. QW-types. The notion of QW -type that we introduce in this section is obtained from that of a W -type by considering not only the algebraic terms over a given structure, but also equations between terms. To code equations we use a type-theoretic rendering of a categorical notion of equational system introduced by Fiore and Hur, referred to as *term equational system* [FH09, Section 2] and as *monadic equational system* [Fio13, Section 5], here instantiated to the special case of free monads on signatures.

Definition 3.1. A *system of equations* over a signature $\Sigma : \mathbf{Sig}$ is specified by:

- A type $E : \mathcal{U}$, whose elements $e : E$ can be thought of as names for each equation.
- A function $V : E \rightarrow \mathcal{U}$. For each equation $e : E$, the type $V e : \mathcal{U}$ represents the variables used in the equation e .
- For each equation $e : E$, two elements called $l e$ and $r e$ of type $T_\Sigma(V e)$, which is the free S_Σ -algebra on the variables $V e$. So $l e$ and $r e$ can be thought of as the abstract syntax trees of terms with some leaves being free variables drawn from $V e$.

So systems of equations are elements of the dependent product

$$\text{Syseq}_\Sigma \stackrel{\text{def}}{=} \sum_{E:\mathcal{U}} \sum_{V:E \rightarrow \mathcal{U}} \left(\prod_{e:E} T_\Sigma(V e) \right) \times \left(\prod_{e:E} T_\Sigma(V e) \right) \quad (3.9)$$

An S_Σ -algebra $\alpha : S_\Sigma(X) \rightarrow X$ *satisfies* the system of equations $\varepsilon \equiv (E, V, l, r) : \text{Syseq}_\Sigma$ if there is a proof of

$$\text{Sat}_{\alpha, \varepsilon}(X) \stackrel{\text{def}}{=} \forall (e : E). \forall (\rho : V e \rightarrow X). (l e \gg \rho) = (r e \gg \rho) \quad (3.10)$$

The category-theoretic view of QW-types is that they are simply S_Σ -algebras that are initial among those satisfying a given system of equations:

Definition 3.2 (QW-type). A *QW-type* for a signature $\Sigma \equiv (A, B) : \text{Sig}$ and a system of equations $\varepsilon \equiv (E, V, l, r) : \text{Syseq}_\Sigma$ is given by a type $\text{QW} : \mathcal{U}$ equipped with an S_Σ -algebra structure and a proof that it satisfies the equations. Thus there are functions

$$\text{qwintro} : S_\Sigma(\text{QW}) \rightarrow \text{QW} \quad (3.11)$$

$$\text{qwequate} : \text{Sat}_{\text{qwintro}, \varepsilon}(\text{QW}) \quad (3.12)$$

together with functions that witness that it is an initial such algebra:

$$\text{qwrec} : \prod_{X:\mathcal{U}} \prod_{\alpha:S_\Sigma X \rightarrow X} \text{Sat}_{\alpha, \varepsilon}(X) \rightarrow \text{QW} \rightarrow X \quad (3.13)$$

$$\text{qwrechom} : \prod_{X:\mathcal{U}} \prod_{\alpha:S_\Sigma X \rightarrow X} \prod_{p:\text{Sat}_{\alpha, \varepsilon}(X)} \text{isHom}(\text{qwrec } X \alpha p) \quad (3.14)$$

$$\text{qwuniq} : \prod_{X:\mathcal{U}} \prod_{\alpha:S_\Sigma X \rightarrow X} \prod_{p:\text{Sat}_{\alpha, \varepsilon}(X)} \prod_{f:\text{QW} \rightarrow X} \text{isHom } f \rightarrow \text{qwrec } X \alpha p = f \quad (3.15)$$

Remark 3.3 (Free algebras). Definition 3.2 defines QW-types as initial algebras for systems of equations ε over signatures Σ . More generally, the *free* (Σ, ε) -algebra on a type $X : \mathcal{U}$ is a type $F_{\Sigma, \varepsilon} X : \mathcal{U}$ equipped with an S_Σ -algebra structure $S_\Sigma(F_{\Sigma, \varepsilon} X) \rightarrow F_{\Sigma, \varepsilon} X$ satisfying the system of equations ε and an inclusion of generators $\eta_X : X \rightarrow F_{\Sigma, \varepsilon} X$ which is universal among such data. Initial algebras are the $X = \mathbb{0}$ special case of free algebras. But once one has them, one also has free algebras, by change of signature: $F_{\Sigma, \varepsilon} X$ is the QW-type for the signature Σ_X and system of equations ε_X defined as follows. If $\Sigma = (A, B)$, then $\Sigma_X = (X + A, B_X)$ where $B_X : X + A \rightarrow \mathcal{U}$ satisfies $B_X(\iota_1 x) = \mathbb{0}$ and $B_X(\iota_2 a) = B a$. And if $\varepsilon = (E, V, l, r)$, then $\varepsilon_X = (E, V, l_X, r_X)$ where for each $e : E$, $l_X e = (l e \gg \eta)$ and $r_X e = (r e \gg \eta)$ (using the S_Σ -algebra structure s on $T_\Sigma(V e)$ given by $s(a, b) = \sigma(\iota_2 a, b)$).

The definitions of S_Σ in (3.2) and $\text{Sat}_{\alpha, \varepsilon}$ in (3.10) suggest that QW-types are an instance of the notion of quotient-inductive type [AK16], with `qwintro` as the element constructor and `qwequate` as the equality constructor. To show that QW-types are indeed quotient-inductive types, they need to have the requisite dependently-typed elimination and computation properties³ for these elements and equality constructors. We show that these follow

from (3.13)–(3.15) and function extensionality. To state this proposition we need a dependent version of the bind operation (3.7). For each “motive” P and induction step p ,

$$P : \text{QW} \rightarrow \mathcal{U}$$

$$p : \prod_{a:A} \prod_{b:B a \rightarrow \text{QW}} \left(\prod_{x:Ba} P(bx) \right) \rightarrow P(\text{qwintro}(a, b)) \quad (3.16)$$

as well as a type $X : \mathcal{U}$, a substitution $\rho : (X \rightarrow \sum_{x:\text{QW}} P x)$, and term $t : T_{\Sigma} X$, we have an element $\text{lift}_{P,X} p \rho t : P(t \gg= (\pi_1 \circ \rho))$ defined by recursion on the structure of t :

$$\begin{aligned} \text{lift}_{P,X} p \rho (\eta x) &\stackrel{\text{def}}{=} \pi_2(\rho x) \\ \text{lift}_{P,X} p \rho (\sigma(a, b)) &\stackrel{\text{def}}{=} p a (\lambda x. b x \gg= (\pi_1 \circ \rho)) ((\text{lift}_{P,X} p \rho) \circ b) \end{aligned} \quad (3.17)$$

Note that the substitution ρ gives terms in the “dependent telescope” $\sum_{x:\text{QW}} P x$, which will be written as P' .

Proposition 3.4. *For a QWI-type as defined above, given P and p as in (3.16), and a term p_{resp} of type*

$$\prod_{e:E} \prod_{\rho:V e \rightarrow \sum_{x:\text{QW}} P x} \text{lift}_{P,X} p \rho (l e) == \text{lift}_{P,X} p \rho (r e) \quad (3.18)$$

there are elimination and computation terms

$$\begin{aligned} \text{qwelim} &: \prod_{x:\text{QW}} P x \\ \text{qwcomp} &: \prod_{a:A} \prod_{b:B a \rightarrow \text{QW}} \text{qwelim}(\text{qwintro}(a, b)) = p a b (\text{qwelim} \circ b) \end{aligned} \quad (3.19)$$

(Note that (3.18) uses heterogeneous identity $==$, because $\text{lift}_{P,X} p \rho (l e)$ and $\text{lift}_{P,X} p \rho (r e)$ inhabit different types, namely $P(l e \gg= (\pi_1 \circ \rho))$ and $P(r e \gg= (\pi_1 \circ \rho))$ respectively.)

Proof. To define the eliminator, we must first use the algebra map on the more general type $P' \stackrel{\text{def}}{=} \sum_{x:\text{QW}} P x$. The algebra map, qwrec , requires that this type P' has an algebra structure $\beta : S_{\Sigma}(P') \rightarrow P'$, which is given by:

$$\beta(a, b) \stackrel{\text{def}}{=} (\text{qwintro}(a, \pi_1 \circ b), p a (\pi_1 \circ b) (\pi_2 \circ b)) \quad (3.20)$$

Moreover, we must show that the recursor satisfies the equations by giving a proof $s : \text{Sat}_{\beta, \varepsilon}(P')$, which we do pointwise on the two elements of the dependent product P' . Taking projections distributes (possibly dependently) over $\gg=$; so to construct s it suffices that, given any $e : E$ and $\rho : V e \rightarrow P'$, we have the two terms:

$$\begin{aligned} \text{qwequate } e \rho &: l e \gg= (\pi_1 \circ \rho) = r e \gg= (\pi_1 \circ \rho) \\ p_{\text{resp}} e \rho &: \text{lift } p \rho (l e) == \text{lift } p \rho (r e) \end{aligned} \quad (3.21)$$

Then the eliminator is defined by taking the second projection of this recursor.

$$\text{qwelim} \stackrel{\text{def}}{=} \pi_2 \circ \text{qwrec } P' \beta s \quad (3.22)$$

³In this paper we work with extensional type theory, so the computation property, (3.19), is also a definitional equality. In our Agda development we work in intensional type theory, so there we only establish the computation property up to propositional equality; using the terminology of Shulman [Shu18], those are *typal* indexed quotient-inductive types.

Given an element $(a, b) : S_{\Sigma}(\mathbf{QW})$, we can prove that applying the eliminator to it computes as expected, that is $\mathbf{qwelim}(\mathbf{qwintro}(a, b)) = p a b(\mathbf{qwelim} \circ b)$, as follows, where we write r' for $\mathbf{qwrec} P' \beta s$:

$$\begin{aligned}
& \mathbf{qwelim}(\mathbf{qwintro}(a, b)) \\
& == \pi_2(r'(\mathbf{qwintro}(a, b))) && \text{def. of } \mathbf{qwelim} \\
& == \pi_2(\beta(a, r' \circ b)) && \text{using } \mathbf{qwrechom} \text{ against } \mathbf{qwintro}(a, b) \\
& == p a (\pi_1 \circ r' \circ b) (\pi_2 \circ r' \circ b) && \text{def. of } \beta \\
& == p a b(\pi_2 \circ r' \circ b) && r' \text{ preserves } \mathbf{QW} \text{ in the first component;} \\
& == p a b(\mathbf{qwelim} \circ b) && \text{follows from } \mathbf{qwrechom} \text{ and } \mathbf{qwuniq} \\
& && \text{def. of } \mathbf{qwelim} \quad \square
\end{aligned}$$

3.3. QWI-types. In this section we consider *indexed* quotient inductive types specified by families of (possibly infinitary) equational theories. As for ordinary W-types, the indexed version of QW-types gives convenient expressive power; see Section 7. To describe the indexed case we first introduce some notation for indexed families of types and functions.

Given a type $I : \mathcal{U}$, when we map between two I -indexed types, say $A : \mathcal{U}^I$ to $B : \mathcal{U}^I$, we will write

$$A \rightarrow B \stackrel{\text{def}}{=} \prod_{k:I} A_k \rightarrow B_k \quad (3.23)$$

for the function type. The composition of $f : A \rightarrow B$ and $g : B \rightarrow C$ will just be written as

$$g \circ f \stackrel{\text{def}}{=} \lambda i. \lambda x. g_i(f_i x) \quad (3.24)$$

We also often need to define an indexed family of types $B_i a : \mathcal{U}^I$ which is dependent on some $i : I$ and element $a : A_i$ for $A : \mathcal{U}^I$. The type of such families will be written as

$$A \dashv \mathcal{U}^I \stackrel{\text{def}}{=} \prod_{i:I} (A_i \rightarrow \mathcal{U}^I) \quad (3.25)$$

We take our notion of an indexed signature for a WI-type (indexed W-type) to be a particular case of indexed containers [AAG05], namely an element of the type $\sum_{I:\mathcal{U}} \sum_{A:\mathcal{U}^I} \prod_{i:I} (A_i \rightarrow \mathcal{U}^I)$, which with the above notational conventions we write as

$$\text{Sig} \stackrel{\text{def}}{=} \sum_{I:\mathcal{U}} \sum_{A:\mathcal{U}^I} (A \dashv \mathcal{U}^I) \quad (3.26)$$

We only need indexed containers whose source and target indexing types are the same, since we only need to consider endofunctors on the category of I -indexed families in \mathcal{U} and their algebras. Thus an indexed signature $\Sigma \equiv (I, A, B)$ is a triple that can be thought of as a set of indices (or sorts) I , an I -indexed family of operators A_i , and a function mapping each operator $a : A_i$ to an I -indexed family of arities in \mathcal{U}^I . For instance, the type of vectors over some type D has an indexed signature as follows (see Example 7.6):

- natural numbers as indices;
- for index 0 a parameter-less operator for the empty vector, and for index $n+1$ an operator parametrised by D for cons; and
- for the operator indexed by 0, a family of empty types for arities, and for operators indexed by $n+1$, a family of arities which is the unit type just in the n th index, and empty otherwise.

Each indexed signature $\Sigma \equiv (I, A, B)$ determines a polynomial endofunctor $S_\Sigma : \mathcal{U}^I \rightarrow \mathcal{U}^I$, which is defined at a family $X : \mathcal{U}^I$ by

$$(S_\Sigma(X))_i \stackrel{\text{def}}{=} \sum_{a:A_i} (B_i a \rightarrow X) \quad (\text{for each } i : I) \quad (3.27)$$

An S_Σ -algebra is by definition an element of the dependent product

$$\text{Alg}_\Sigma \stackrel{\text{def}}{=} \sum_{X:\mathcal{U}^I} (S_\Sigma(X) \rightarrow X) \quad (3.28)$$

(For example, when Σ is the signature for vectors over some type D , then an algebra $(X, \alpha) : \text{Alg}_\Sigma$ amounts to $\alpha_0 : \mathbb{1} \rightarrow X_0$ and $\alpha_{n+1} : D \times X_n \rightarrow X_{n+1}$.)

S_Σ -algebra morphisms $(X, \alpha) \rightarrow (X', \alpha')$ are given by an I -indexed family of functions $h : X \rightarrow X'$ together with a family of elements in the types

$$(\text{isHom } h)_i \stackrel{\text{def}}{=} \prod_{a:A_i} \prod_{b:B_i a \rightarrow X} (\alpha'_i(a, h \circ b) = h_i(\alpha_i(a, b))) \quad (3.29)$$

The WI-type $W_{i:I, a:A_i} B_i a$, or W_Σ , determined by $\Sigma \equiv (I, A, B)$ is the underlying type of an initial S_Σ -algebra. The elements of W_Σ represent an I -indexed family of *closed* algebraic terms (i.e. well-founded trees) over the signature Σ . Given a family $V : \mathcal{U}^I$, the family $T_\Sigma(V) : \mathcal{U}^I$ of *open* terms with variables from V is the inductively defined family with constructors

$$\begin{aligned} \eta : V &\rightarrow T_\Sigma V \\ \sigma : S_\Sigma(T_\Sigma V) &\rightarrow T_\Sigma V \end{aligned} \quad (3.30)$$

Given an S_Σ -algebra $(X, \alpha) : \text{Alg}_\Sigma$ and an I -indexed family of functions $f : V \rightarrow X$, the unique morphism of S_Σ -algebras from the S_Σ -algebra $(T_\Sigma(V), \sigma)$ to (X, α) that extends f has an underlying family of functions $T_\Sigma(V) \rightarrow X$ mapping each $t : (T_\Sigma V)_i$ to the element $t \gg f$ defined analogously to (3.7).

Given an indexed signature $\Sigma \equiv (I, A, B) : \text{Sig}$, a *system of equations* for it is an element of the dependent product

$$\text{Syseq}_\Sigma \stackrel{\text{def}}{=} \sum_{E:\mathcal{U}^I} \sum_{V:E \rightarrow \mathcal{U}^I} \left(\prod_{i:I} \prod_{e:E_i} (T_\Sigma(V_i e))_i \right) \times \left(\prod_{i:I} \prod_{e:E_i} (T_\Sigma(V_i e))_i \right) \quad (3.31)$$

An S_Σ -algebra $\alpha : S_\Sigma(X) \rightarrow X$ satisfies the system of equations $\varepsilon \equiv (E, V, l, r) : \text{Syseq}_\Sigma$ if for each $i : I$ there is a proof of

$$(\text{Sat}_{\alpha, \varepsilon}(X))_i \stackrel{\text{def}}{=} \forall (e : E_i). \forall (\rho : V_i e \rightarrow X). ((l_i e) \gg \rho) = ((r_i e) \gg \rho) \quad (3.32)$$

Definition 3.5 (QWI-type). A *QWI-type* for an indexed signature $\Sigma \equiv (I, A, B) : \text{Sig}$ and a system of equations $\varepsilon \equiv (E, V, l, r) : \text{Syseq}_\Sigma$ is given by an I -indexed family of types $\text{QW} : \mathcal{U}^I$ equipped with an S_Σ -algebra structure and a proof that it satisfies the equations

$$\text{qwintro} : S_\Sigma(\text{QW}) \rightarrow \text{QW} \quad (3.33)$$

$$\text{qwequate} : \text{Sat}_{\text{qwintro}, \varepsilon}(\text{QW}) \quad (3.34)$$

together with functions that witness that it is an initial such algebra:

$$\text{qwrec} : \prod_{X:\mathcal{U}^I} \prod_{\alpha:S_\Sigma(X)\rightarrow X} (\text{Sat}_{\alpha,\varepsilon}(X) \rightarrow (\text{QW} \rightarrow X)) \quad (3.35)$$

$$\text{qwrechom} : \prod_{X:\mathcal{U}^I} \prod_{\alpha:S_\Sigma(X)\rightarrow X} \prod_{p:\text{Sat}_{\alpha,\varepsilon}(X)} \text{isHom}(\text{qwrec } X \alpha p) \quad (3.36)$$

$$\text{qwuniq} : \prod_{X:\mathcal{U}^I} \prod_{\alpha:S_\Sigma(X)\rightarrow X} \prod_{p:\text{Sat}_{\alpha,\varepsilon}(X)} \prod_{f:\text{QW}\rightarrow X} \text{isHom } f \rightarrow \text{qwrec } X \alpha p = f \quad (3.37)$$

4. WEAKLY INITIAL SETS OF COVERS

In Section 3 we defined a QWI-type in a topos to be an initial algebra for a given (possibly infinitary) indexed signature and system of equations (Definition 3.5). If one interprets these notions in the topos of sets in classical Zermelo-Fraenkel set theory with the Axiom of Choice (ZFC), one regains the usual notion from universal algebra of initial algebras for equational theories that are multi-sorted [BL70] and infinitary [Lin66]. Thus in ZFC, the QWI-type for a signature $\Sigma = (I, A, B)$ and system of equations $\varepsilon = (E, V, l, r)$ can be constructed by first forming the I -indexed family W_Σ of sets of well-founded trees over Σ and then quotienting by the congruence relation \sim_ε on W_Σ generated by ε . The I -indexed family of quotient sets $(W_\Sigma)/\sim_\varepsilon$ yields the desired initial algebra for (Σ, ε) provided the S_Σ -algebra structure on W_Σ induces one on the quotient sets. It does so, because for each operator in the signature, using the Axiom of Choice (AC) one can pick representatives of the (possibly infinitely many) equivalence classes that are the arguments of the operator, apply the interpretation of the operator in W_Σ and then take the equivalence class of that. So the topos of sets in ZFC has QWI-types.

Is this use of AC really necessary? Blass [Bla83, Section 9] shows that if one drops AC and just works in ZF, then provided a certain large cardinal axiom is consistent with ZFC, it is consistent with ZF that there is an infinitary equational theory with no initial algebra. He shows this by first exhibiting a countably presented equational theory whose initial algebra has to be an uncountable regular cardinal; and secondly appealing to the construction of Gitik [Git80] of a model of ZF with no uncountable regular cardinals (assuming a certain large cardinal axiom). Lumsdaine and Shulman [LS19] turn the infinitary equational theory of Blass into a higher-inductive type that cannot be proved to exist in ZF (and hence cannot be constructed in type theory just using pushouts and the natural numbers). We show in Example 7.10 that this higher inductive type can be presented as a QWI-type.

So one cannot hope to construct QWI-types using a type theory which is interpretable in just ZF. However, the type theory in which we work (Section 2) already requires going beyond ZF to be able to give a classical set-theoretic interpretation of its universes (by assuming the existence of enough strongly inaccessible cardinals, for example). So the above considerations about non-existence of initial algebras for infinitary equational theories in ZF do not necessarily rule out the construction of QWI-types in other toposes with natural numbers object and universes. In Section 6, we show that QWI types exist in a topos if it satisfies a weak form of choice principle that we call IWISC and which is known to hold for a wide range of toposes. IWISC is a version for universes in the type theory in which we are working of a property introduced by van den Berg and Moerdijk [vdBM14], who work in constructive set theory CZF and call it the *Axiom of Multiple Choice* (building

upon previous work by Moerdijk and Palmgren [MP02]); it is related to the *Type Theoretic Collection Axiom* TTCA_f of Streicher [Str05a] (see Remark 4.2).

Definition 4.1 (Indexed WISC). A *family* in a universe \mathcal{U} is just a pair (A, B) where $A : \mathcal{U}$ and $B : A \rightarrow \mathcal{U}$. A family (C, E) in \mathcal{U} is a *wisc* for $Y : \mathcal{U}$ if for all $X : \mathcal{U}$ and surjective functions $q : X \rightarrow Y$ in \mathcal{U} , there exists $c : C$ and $f : E c \rightarrow X$ for which $q \circ f : E c \rightarrow Y$ is surjective. A family (C, E) is a *wisc* for a family (A, B) if it is a *wisc* for each type $B a$ in the family. We say that *the universe \mathcal{U} satisfies IWISC* if for all families (A, B) in \mathcal{U} , there exists a family in \mathcal{U} which is a *wisc* for (A, B) .

Following [<https://ncatlab.org/nlab/show/WISC>], “wisc” stands for “weakly initial set of covers”, with the terminology being justified as follows. A *cover* of $Y : \mathcal{U}$ is just a surjective function $q : X \rightarrow Y$ for some $X : \mathcal{U}$. If (C, E) is a *wisc* for Y as in the above definition, then the family of all covers of the form $E c \rightarrow Y$ with $c : C$ is weakly initial, in the sense that for every cover $q : X \rightarrow Y$ in \mathcal{U} , there is a cover in the family that factors through q .

Note that in classical set theory AC implies IWISC; this is because the Axiom of Choice implies that covers split and hence any family (A, B) is a *wisc* for itself. IWISC is independent of ZF in classical logic [Kar14; Rob15]. The results of van den Berg and Moerdijk [vdBM14] imply that in constructive logic it is preserved by many ways of making new toposes from existing ones, in particular by the formation of sheaf toposes and realizability toposes over a given topos. Thus it holds in all the toposes commonly used for the semantics of Type Theory. It is in this sense that IWISC is a constructively acceptable form of the axiom of choice. (Roberts [Rob15] gives examples of toposes which fail to satisfy it.)

Remark 4.2. IWISC is a property of a single universe, in contrast to Streicher’s type theoretic formulation of a WISC axiom, TTCA_f [Str05a], which uses two universes (and which can be shown to imply IWISC). With a single universe it seems necessary to quantify over indexed families in the universe (hence our choice of terminology), rather than just over types in the universe as in *loc. cit.*; see Levy [Lev21, Section 5.1]. Note that IWISC asks that a *wisc* merely exists for each family in \mathcal{U} . A stronger requirement on \mathcal{U} would be to ask for a function assigning a *wisc* to each family; and this is equivalent to asking for a function mapping each $B : \mathcal{U}$ to a *wisc* for B . (For note that, given a family (A, B) , if we have a function mapping each $a : A$ to a *wisc* (C_a, E_a) for $B a$, then the single family (C, E) with $C = \sum_{a:A} C_a$ and $E(a, c) = E_a c$ is a *wisc* for all the $B a$ simultaneously.) Levy [Lev21] calls this property GLOBAL WISC. Although the topos of sets in ZFC with Grothendieck universes satisfies GLOBAL WISC, it is not known which other toposes do.

The following lemma gives a convenient reformulation of the *wisc* property from Definition 4.1 that we use in the next section.

Lemma 4.3. *Suppose that $A : \mathcal{U}$ has a *wisc* $(C : \mathcal{U}, E : C \rightarrow \mathcal{U})$. If $B : A \rightarrow \mathcal{U}$ and $\phi : \prod_{x:A} (B x \rightarrow \text{Prop})$ satisfy $\forall (x : A). \exists (y : B x). \phi x y$, then there exist $c : C$, a surjection $p : E c \rightarrow A$ and a function $q : \prod_{z:E c} B(p z)$ satisfying $\forall (z : E c). \phi(p z)(q z)$.*

Proof. Consider the type $\Phi \stackrel{\text{def}}{=} \sum_{x:A} \sum_{y:B x} \phi x y$ in \mathcal{U} . By assumption $\pi_1 : \Phi \rightarrow A$ is a surjection, so since (C, E) is a *wisc* for A , there exist $c : C$ and $e : E c \rightarrow \Phi$ with $\pi_1 \circ e$ a surjection. So we can take $p \stackrel{\text{def}}{=} \pi_1 \circ e$ and $q \stackrel{\text{def}}{=} \pi_2 \circ \pi_2 \circ e$; and then $\pi_2(\pi_2(e z))$ is a proof that $\phi(p z)(q z)$ holds. \square

5. SIZE

Swan [Swa18] uses IWISC to construct *W-types with reductions*, which are a simple special case of QWI-types (see Example 7.8). We will see that in fact IWISC implies the existence of all QWI-types, but using a different approach from that of Swan [Swa18]. We show in this section that, starting with a given signature Σ and system of equations ε (Definition 3.1), using IWISC one can construct a well-founded type of “sizes” with the property that colimits of diagrams indexed by such sizes are preserved when taking exponents by the arity types of Σ and ε . This will enable us to construct the QW-type for (Σ, ε) as a colimit in Section 6. We consider the general case of indexed signatures in Remark 5.11; and the construction of QWI-types as a sized-indexed colimit is detailed in the accompanying Agda formalization [FPS21]. Size is here playing a role in the constructive type theory of toposes that in classical set theory would be played by ordinal numbers. The next definition gives the properties of size that we need.

Definition 5.1 (Size). A type `Size` in a universe \mathcal{U} will be called a *type of sizes* if it comes equipped with

- a binary relation $_ < _ : \text{Size} \rightarrow \text{Size} \rightarrow \text{Prop}$ which is transitive

$$\forall i, j, k. i < j \rightarrow j < k \rightarrow i < k \quad (5.1)$$

and well-founded

$$\forall (\phi : \text{Size} \rightarrow \text{Prop}). (\forall i. (\forall j. j < i \rightarrow \phi j) \rightarrow \phi i) \rightarrow \forall i. \phi i \quad (5.2)$$

- a distinguished element 0^s
- a binary operation $_ \sqcup^s _ : \text{Size} \rightarrow \text{Size} \rightarrow \text{Size}$ which gives upper bounds with respect to $<$ for pairs of elements

$$\forall i, j. i < i \sqcup^s j \wedge j < i \sqcup^s j \quad (5.3)$$

Note that defining $i^+ \stackrel{\text{def}}{=} i \sqcup^s i$ we get a form of successor operation satisfying

$$\forall i. i < i^+ \quad (5.4)$$

For the examples of types of sizes given in Example 5.4 this successor operation preserves $<$, but we do not need that property for the results in this paper.⁴

Such a type of sizes has many properties of the classic notion of limit ordinal, except that we do not require the order to be total ($\forall i, j. i < j \vee i = j \vee j < i$); that would be too strong in a constructive setting and indeed does not hold in the examples below. Nor do we have any need for an extensionality property ($\forall i, j. (\forall k. k < i \leftrightarrow k < j) \rightarrow i = j$). In the precursor to this paper [FPS20] we made use of Agda’s built in type `Size` and its version of sized types [Abe12]. This shares only some of the properties of the above definition. In particular, it has a “stationary” size ∞ satisfying $\infty < \infty$ and hence the relation $<$ for Agda’s notion of size is not well-founded.⁵

⁴In fact we do not need the distinguished element 0^s either, but it does no harm to require it and then sizes are directed with respect to $<$ in the usual sense of having upper bounds for all finite subsets, including the empty one.

⁵Unfortunately in the current version of Agda (2.6.2) it is possible to prove well-foundedness of $<$ for its built in type of sizes `Size`, and this leads immediately to logical unsoundness; see github.com/agda/agda/issues/3026. For this reason we avoid the use of Agda’s sized types in the current Agda development accompanying this paper.

Notation 5.2 (Bounded quantification over sizes). We write $\prod_{j < i} _$ for $\prod_{j: (\downarrow i)} _$, where

$$\downarrow i \stackrel{\text{def}}{=} \{j : \text{Size} \mid j < i\} \quad (5.5)$$

is the type of sizes below $i : \text{Size}$. Similarly for other binders involving sizes, such as $\forall j < i. _$ and $\lambda j < i. _$.

In the next section we need the fact that well-founded recursion (5.6) can be reduced to well-founded induction (5.2) by defining the graph of the recursive function and then appealing to unique choice (2.3) and function extensionality (2.2):

Proposition 5.3 (Well-founded recursion). *For any family of types $A : \text{Size} \rightarrow \mathcal{U}$ and function $\alpha : \prod_i ((\prod_{j < i} A j) \rightarrow A i)$, there is a unique function $\text{rec } A \alpha : \prod_i A i$ satisfying*

$$\forall i. \text{rec } A \alpha i = \alpha i (\lambda j < i. \text{rec } A \alpha j) \quad (5.6)$$

Proof. Let $R : \prod_i (A i \rightarrow \text{Prop})$ be the least⁶ family of relations satisfying

$$\forall i. \forall (f : \prod_{j < i} A j). (\forall j < i. R_j (f j)) \rightarrow R_i (\alpha i f) \quad (5.7)$$

The fact that R is least with this property allows one to prove $\forall i. \exists! (x : A_i). R_i x$ by applying (5.2) with $\phi i \stackrel{\text{def}}{=}} \exists! (x : A_i). R_i x$. So by unique choice (2.3) there is $r : \prod_i A i$ satisfying $\forall i. R_i (r i)$ and hence by (5.7) also satisfying $\forall i. R_i (\alpha i (\lambda j < i. r j))$. Since for each i there is a unique x with $R_i x$, it follows that $r i = \alpha i (\lambda j < i. r j)$. So we can take $\text{rec } A \alpha$ to be r . If $r' : \prod_i A i$ also satisfies $\forall i. r' i = \alpha i (\lambda j < i. r' j)$, then $\forall i. r' i = r i$ follows from another application of (5.2), so that $r' = r$ by function extensionality (2.2). \square

Example 5.4 (Plump order). Let $\text{Size} : \mathcal{U}$ be the W-type determined by some type $Op : \mathcal{U}$ and family $Ar : Op \rightarrow \mathcal{U}$ (see Section 3.1). Thus Size is inductively defined with constructor $\sigma : (\sum_{a:Op} (Ar a \rightarrow \text{Size})) \rightarrow \text{Size}$. The *plump* ordering [Tay96] on Size is given by the least relations $_ < _, _ \leq _ : \text{Size} \rightarrow \text{Size} \rightarrow \text{Prop}$ satisfying for all $a : Op, b : Ar a \rightarrow \text{Size}$ and $i : \text{Size}$

$$(\forall (x : Ar a). b x < i) \rightarrow \sigma a b \leq i \quad (5.8)$$

$$(\exists (x : Ar a). i \leq b x) \rightarrow i < \sigma a b \quad (5.9)$$

It is not hard to see that the relation $<$ is transitive (5.1) and well-founded (5.2). Furthermore one can prove that \leq is reflexive and hence from (5.9) we get

$$\forall (x : Ar a). b x < \sigma a b \quad (5.10)$$

So for each operation $a : Op$, every family $b : Ar a \rightarrow \text{Size}$ of elements of Size indexed by its arity $Ar a$ has an upper bound with respect to $<$, namely $\sigma a b$. In particular, if the signature (Op, Ar) contains an operation $a_2 : Op$ whose arity is $Ar a_2 = 2$, then an upper bound for any $i, j : \text{Size}$ with respect to $<$ is given by

$$i \sqcup^s j \stackrel{\text{def}}{=} \sigma a_2 b_{i,j} \quad (5.11)$$

with $b_{i,j} : 2 \rightarrow \text{Size}$ defined by $b_{i,j} 0 = i$ and $b_{i,j} 1 = j$. So (5.3) is satisfied in this case. Similarly, if the signature (Op, Ar) contains an operation $a_0 : Op$ whose arity is $Ar a_0 = 0$, then Size contains a distinguished element

$$0^s \stackrel{\text{def}}{=} \sigma a_0 b \quad (5.12)$$

⁶Instead of the impredicative definition of R (and $<$ and \leq in Example 5.4) as the least relation closed under some rules, in our Agda development such relations are constructed as inductively defined datatypes in the universe Prop , allowing one to use dependent pattern-matching to simplify proofs about them.

with $b : \mathbb{0} \rightarrow \text{Size}$ uniquely determined. Therefore, we have:

Lemma 5.5. *If the signature $Op : \mathcal{U}, Ar : Op \rightarrow \mathcal{U}$ contains a nullary operation ($a_0 : Op$ with $Ar a_0 = \mathbb{0}$) and a binary operation ($a_2 : Op$ with $Ar a_2 = \mathbb{2}$), then the corresponding W -type $\text{Size} : \mathcal{U}$, equipped with the plump order $<$, is a type of sizes in the sense of Definition 5.1. Furthermore for any $a : Op$, every $(Ar a)$ -indexed family of sizes $b : Ar a \rightarrow \text{Size}$ has an upper bound with respect to $<$ (namely $\sigma a b$). \square*

Given a signature Σ in a universe \mathcal{U} satisfying IWISC, the lemma allows us to prove the existence of a type of sizes Size with enough upper bounds to be able to prove a cocontinuity property of the polynomial endofunctor associated with Σ (Corollary 5.10 below); we apply this in Section 6 to construct QW-types for a given signature and system of equations (and QWI-types for indexed signatures and systems of equations as defined in Section 3.3). The cocontinuity property has to do with colimits of Size -indexed diagrams in \mathcal{U} . To state it, we first recall some semi-standard⁷ category-theoretic notions to do with diagrams and colimits.

5.1. Colimits of size-indexed diagrams. By definition, given a type of sizes Size in a universe \mathcal{U} (Definition 5.1), a *Size-indexed diagram* is given by a family of types $D : \text{Size} \rightarrow \mathcal{U}$ equipped with functions $\delta_{i,j} : D_i \rightarrow D_j$ for all $i, j : \text{Size}$ with $i < j$, satisfying

$$\forall (i, j, k : \text{Size}). i < j < k \rightarrow \delta_{i,k} = \delta_{j,k} \circ \delta_{i,j} \quad (5.13)$$

As usual, the *colimit* of such a diagram is a cocone of functions in \mathcal{U}

$$(\nu_D)_i : D_i \rightarrow \text{colim } D \quad \forall (i, j : \text{Size}). i < j \rightarrow (\nu_D)_i = (\nu_D)_j \circ \delta_{i,j} \quad (5.14)$$

with the universal property that for any other cocone

$$f_i : D_i \rightarrow C \quad \forall (i, j : \text{Size}). i < j \rightarrow f_i = f_j \circ \delta_{i,j}$$

there is a unique function $f : \text{colim } D \rightarrow C$ satisfying $\forall i. f_i = f \circ (\nu_D)_i$.

Colimits can be constructed using quotient types: we define a binary relation \sim on $\sum_i D_i$ by:

$$(i, x) \sim (j, y) \stackrel{\text{def}}{=} \exists (k : \text{Size}). i < k \wedge j < k \wedge \delta_{i,k} x = \delta_{j,k} y \quad (5.15)$$

This is an equivalence relation, because $(\text{Size}, <)$ has property (5.3). Quotienting by it yields

$$\text{colim } D \stackrel{\text{def}}{=} (\sum_i D_i) / \sim \quad (5.16)$$

with the universal cocone functions $(\nu_D)_i$ given by mapping each $x : D_i$ to the equivalence class $[i, x]_{\sim}$.

Definition 5.6 (Preservation of colimits by taking a power). Given a Size -indexed diagram (D, δ) in \mathcal{U} and a type $X : \mathcal{U}$, we get a *power* diagram (D^X, δ^X) in \mathcal{U} with

$$\begin{aligned} (D^X)_i &\stackrel{\text{def}}{=} X \rightarrow D_i & (i : \text{Size}) \\ (\delta^X)_{i,j} &\stackrel{\text{def}}{=} \lambda (f : X \rightarrow D_i). \delta_{i,j} \circ f & (i, j : \text{Size}) \end{aligned} \quad (5.17)$$

Post-composition with $(\nu_D)_i$ gives a cocone under the diagram D^X with vertex $X \rightarrow \text{colim } D$ and by universality this induces a function

$$\begin{aligned} \kappa_{D,X} : \text{colim}(D^X) &\rightarrow (X \rightarrow \text{colim } D) \\ \kappa_{D,X} [i, f]_{\sim} &= (\nu_D)_i \circ f \end{aligned} \quad (5.18)$$

⁷They are only *semi*-standard, because $<$ is not reflexive (indeed is irreflexive, because of well-foundedness), so that $(\text{Size}, <)$ is only a (thin) *semi*-category.

One says that *taking a power by $X : \mathcal{U}$ preserves Size-indexed colimits* if for all Size-indexed diagrams (D, δ) in \mathcal{U} the function $\kappa_{D,X}$ is an isomorphism.

Theorem 5.7 (Cocontinuity). *Suppose \mathcal{U} is a universe satisfying IWISC in a topos with natural numbers object. Given $A : \mathcal{U}$ and $B : A \rightarrow \mathcal{U}$, there exists a type of sizes $\mathbf{Size} : \mathcal{U}$ with the property that for all $a : A$, taking a power by the type $B a : \mathcal{U}$ preserves Size-indexed colimits.*

Proof. The proof is in three parts, (1)–(3). In part (1) we show how to find a suitable type of sizes \mathbf{Size} using Lemma 5.5; so \mathbf{Size} is a W-type for a suitable signature derived from $A : \mathcal{U}$ and $B : A \rightarrow \mathcal{U}$. The signature contains arities arising from a wisc whose existence is guaranteed by IWISC. In fact we need to consider not only the covers in such a wisc, but also a wisc for the family of (subtypes of) covers in this wisc, for the same reason that Swan uses “2-cover bases” [Swa18, Section 3.2.2]. Next, we have to prove that (5.18) is an isomorphism when $X : \mathcal{U}$ is $B a$, for any $a : A$; and for this it suffices to prove that it is (2) injective and (3) surjective. For then we can apply unique choice (2.3) to construct a two-sided inverse for $\kappa_{D,X}$. By definition of \sim (5.15), $\kappa_{D,X}$ is injective iff

$$\begin{aligned} \forall (i, j : \mathbf{Size}). \forall (f : X \rightarrow D_i). \forall (f' : X \rightarrow D_j). (\nu_D)_i \circ f = (\nu_D)_j \circ f' \rightarrow \\ \exists (k : \mathbf{Size}). i < k \wedge j < k \wedge \delta_{i,k} \circ f = \delta_{j,k} \circ f' \end{aligned} \quad (5.19)$$

and surjective iff

$$\forall (f : X \rightarrow \text{colim } D). \exists (i : \mathbf{Size}). \exists (f' : X \rightarrow D_i). (\nu_D)_i \circ f' = f \quad (5.20)$$

- (1) *Construction of \mathbf{Size} :* Given $A : \mathcal{U}$ and $B : A \rightarrow \mathcal{U}$, using IWISC let (C, F) be a wisc for the family (A, B) . For each $c : C$, $a : A$ and function $f : F c \rightarrow B a$ we can form the kernel of f

$$\text{Ker } f \stackrel{\text{def}}{=} \sum_{x, x' : F c} (f x = f x') \quad (5.21)$$

Applying IWISC again, let (C', F') be a wisc for this family of kernels indexed by $(c, a, f) : \sum_{(c,a) : C \times A} (F c \rightarrow B a)$. Finally, consider the signature with $Op = \mathbb{1} + \mathbb{1} + A + C + C'$ and $Ar : Op \rightarrow \mathcal{U}$ the function mapping $0 : \mathbb{1}$ in the first summand of Op to 0 , $0 : \mathbb{1}$ in the second summand to $\mathbb{2}$, $a : A$ in the third summand to $B a$, $c : C$ in the fourth summand to $F c$ and $c' : C'$ in the fifth summand to $F' c'$. Then as in Lemma 5.5, the W-type for the signature (Op, Ar) is a type of sizes. Call it \mathbf{Size} .

- (2) *Proof of (5.19):* Recall that $X : \mathcal{U}$ is of the form $B a$ for some $a : A$. Suppose we have $f : X \rightarrow D_i$ and $f' : X \rightarrow D_j$ satisfying $(\nu_D)_i \circ f = (\nu_D)_j \circ f'$. So by definition of $\text{colim } D$ (5.16) we have

$$\forall (x : X). \exists (k : \mathbf{Size}). i < k \wedge j < k \wedge \delta_{i,k}(f x) = \delta_{j,k}(f' x) \quad (5.22)$$

Using the version of the wisc property of (C, F) from Lemma 4.3, there is $c : C$, $p : F c \rightarrow X$ and $s : F c \rightarrow \mathbf{Size}$ with p surjective and satisfying

$$\forall (z : F c). i < s z \wedge j < s z \wedge \delta_{i,s z}(f(p z)) = \delta_{j,s z}(f'(p z)) \quad (5.23)$$

Since $F c$ is one of the arities in the signature of the W-type \mathbf{Size} , by Lemma 5.5 we have that $s : F c \rightarrow \mathbf{Size}$ has an upper bound, i.e. there is $k : \mathbf{Size}$ with $\forall (z : F c). s z < k$; and by (5.3) we can assume further that $i < k$ and $j < k$. Furthermore, from (5.23) and (5.13) we get $\forall (z : F c). \delta_{i,k}(f(p z)) = \delta_{j,k}(f'(p z))$; but p is surjective, so $\delta_{i,k} \circ f = \delta_{j,k} \circ f'$, as required for (5.19).

- (3) *Proof of (5.20)*: Suppose we have $f : X \rightarrow \text{colim } D$. Since the quotient function $[_]\sim : D_i \rightarrow \text{colim } D$ is surjective, using Lemma 4.3 again, there is $c : C, p : Fc \rightarrow X, s : Fc \rightarrow \text{Size}$ and $g : \prod_{z:Fc} D_{sz}$ with p surjective and satisfying $\forall(z : Fc). f(pz) = [sz, gz]_{\sim}$. As before, we have that $s : Fc \rightarrow \text{Size}$ has an upper bound, i.e. there is $j : \text{Size}$ with $\forall(z : Fc). sz < j$. So we get a function $g' : Fc \rightarrow D_j$ by defining $g'z \stackrel{\text{def}}{=} \delta_{sz,j}(gz)$ and hence

$$f(pz) = [sz, gz]_{\sim} = [j, \delta_{sz,j}(gz)]_{\sim} = (\nu_D)_j(g'z) \quad (5.24)$$

Let $Y \stackrel{\text{def}}{=} \text{Ker } p$ be the kernel of p as in (5.21). Then for any $(z, z', _): Y$, since $pz = pz'$, we have $(\nu_D)_j(g'z) = f(pz) = f(pz') = (\nu_D)_j(g'z')$ and hence $\exists k. j < k \wedge \delta_{j,k}(g'z) = \delta_{j,k}(g'z')$. So we can apply Lemma 4.3 again to deduce the existence of $c' : C', \langle p'_1, p'_2, _ \rangle : F'c' \rightarrow Y$ and $s' : F'c' \rightarrow \text{Size}$ with $\langle p'_1, p'_2 \rangle$ surjective and satisfying

$$\forall(z' : F'c'). j < s'z' \wedge \delta_{j,s'z'}(g'(p'_1 z')) = \delta_{j,s'z'}(g'(p'_2 z')) \quad (5.25)$$

Since $F'c'$ is one of the arities in the signature of the W-type Size , by Lemma 5.5 we have that the family of sizes $s' : F'c' \rightarrow \text{Size}$ has an upper bound, i say; and by (5.3) we can assume $j < i$. Let $g'' : Fc \rightarrow D_i$ be $\delta_{j,i} \circ g'$. Thus from (5.24) we have

$$f(pz) = (\nu_D)_i(g''z) \quad (5.26)$$

and from (5.25) also $g'' \circ p'_1 = g'' \circ p'_2$. So altogether we have functions

$$F'c' \xrightarrow{\langle p'_1, p'_2 \rangle} Y \xrightarrow[\pi_2]{\pi_1} Fc \xrightarrow{g''} D_i \quad (5.27)$$

with $g'' \circ \pi_1 \circ \langle p'_1, p'_2 \rangle = g'' \circ p'_1 = g'' \circ p'_2 = g'' \circ \pi_2 \circ \langle p'_1, p'_2 \rangle$. Since $\langle p'_1, p'_2 \rangle$ is surjective, this implies $g'' \circ \pi_1 = g'' \circ \pi_2$. But since $Y = \text{Ker } p$, p is the coequalizer of π_1 and π_2 (since as we noted in Section 2, toposes have effective epimorphisms); therefore there is a unique function $f' : X \rightarrow D_i$ satisfying $f' \circ p = g''$. Since $(\nu_D)_i \circ f' \circ p = (\nu_D)_i \circ g'' = f \circ p$ by (5.26) and p is surjective, we have $(\nu_D)_i \circ f' = f$, completing the proof of (5.20) and hence of Theorem 5.7. \square

Remark 5.8. Note that the type of sizes constructed in the proof of the theorem has the property that for any $a : A, \text{Size}$ has upper bounds (with respect to $<$) for families of sizes indexed by Ba (because by construction Ba is one of the arities in the signature of the W-type Size and so Lemma 5.5 applies). Such upper bounds are not needed in the above proof. We included them because they are needed below in the proof of Theorem 6.1 when proving property (6.11).

Definition 5.9 (Preservation of colimits by polynomial endofunctors). If (D, δ) is a Size -indexed diagram in \mathcal{U} , then we get another such, $(S_\Sigma \circ D, S_\Sigma \circ \delta)$, by composing with the polynomial endofunctor $S_\Sigma : \mathcal{U} \rightarrow \mathcal{U}$ associated with the signature Σ as in (3.2):

$$\begin{aligned} (S_\Sigma \circ D)_i &\stackrel{\text{def}}{=} S_\Sigma(D_i) \quad (i : \text{Size}) \\ (S_\Sigma \circ \delta)_{i,j} &\stackrel{\text{def}}{=} S_\Sigma(\delta_{i,j}) \quad (i, j : \text{Size}) \end{aligned} \quad (5.28)$$

Applying S_Σ to (5.14) gives a cocone under $(S_\Sigma \circ D, S_\Sigma \circ \delta)$ with vertex $S_\Sigma(\text{colim } D)$ and this induces a function in \mathcal{U} ,

$$\kappa_{D,\Sigma} : \text{colim}(S_\Sigma \circ D) \rightarrow S_\Sigma(\text{colim } D) \quad (5.29)$$

One says that *the polynomial endofunctor S_Σ preserves Size -indexed colimits* if $\kappa_{D,\Sigma}$ is an isomorphism for all diagrams (D, δ) .

Corollary 5.10 (Cocontinuity of S_Σ). *In a topos with natural numbers object, given a signature $\Sigma = (A : \mathcal{U}, B : A \rightarrow \mathcal{U})$ in a universe \mathcal{U} satisfying IWISC, there exists a type of sizes \mathbf{Size} with the property that the associated polynomial endofunctor $S_\Sigma : \mathcal{U} \rightarrow \mathcal{U}$ preserves \mathbf{Size} -indexed colimits.*

Proof. We apply Theorem 5.7 to find a type of sizes, \mathbf{Size} for the given $A : \mathcal{U}$ and $B : A \rightarrow \mathcal{U}$. So for each diagram (D, δ) in \mathcal{U} , we have properties (5.19) and (5.20) when X is Ba , for any $a : A$. The function (5.29) satisfies for all $i : \mathbf{Size}$, $a : A$ and $b : Ba \rightarrow D_i$

$$\kappa_{D,\Sigma}[i, (a, b)]_\sim = (a, (\nu_D)_i \circ b) \quad (5.30)$$

We have that $\kappa_{D,\Sigma}$ is injective, because if $(a, (\nu_D)_i \circ b) = (a', (\nu_D)_j \circ b')$, then $a = a'$ and $(\nu_D)_i \circ b = (\nu_D)_j \circ b'$; but then from (5.19) with $X = Ba = Ba'$, there is some k with $i, j < k$ and $\delta_{i,k} \circ b = \delta_{j,k} \circ b'$, so that $[i, (a, b)]_\sim$ and $[j, (a', b')]_\sim$ are equal terms of $\text{colim}(S_\Sigma \circ D)$.

Furthermore, $\kappa_{D,\Sigma}$ is surjective because given $(a, b) : S_\Sigma(\text{colim } D)$, from (5.20) with $X = Ba$, there exist $i : \mathbf{Size}$ and $b' : Ba \rightarrow D_i$ with $(\nu_D)_i \circ b' = b$; so $[i, (a, b')]_\sim$ in $\text{colim}(S_\Sigma \circ D)$ is mapped by $\kappa_{D,\Sigma}$ to (a, b) .

Since $\kappa_{D,\Sigma}$ is both injective and surjective, we can apply unique choice (2.3) to conclude that it is an isomorphism. \square

Remark 5.11 (Cocontinuity for polynomial endofunctors on \mathcal{U}^I). There are indexed versions of Theorem 5.7 and Corollary 5.10. To state them for an indexing type $I : \mathcal{U}$, we need to consider \mathbf{Size} -indexed diagrams in \mathcal{U}^I and their colimits. Since the latter are given pointwise by colimits in \mathcal{U} , this makes what follows a simple extension of the previous development.

Given a \mathbf{Size} -indexed diagram (D, δ) in \mathcal{U}^I and a family $X : \mathcal{U}^I$, the indexed version of (5.17) is a *power* diagram (D^X, δ^X) in \mathcal{U} with

$$\begin{aligned} (D^X)_i &\stackrel{\text{def}}{=} X \rightarrow D_i & (i : \mathbf{Size}) \\ (\delta^X)_{i,j} &\stackrel{\text{def}}{=} \lambda(f : X \rightarrow D_i). \delta_{i,j} \circ f & (i, j : \mathbf{Size}) \end{aligned} \quad (5.31)$$

Post-composition with $(\nu_D)_i$ gives a cocone under the diagram D^X with vertex $X \rightarrow \text{colim } D$ and this induces a function which is the indexed version of (5.18):

$$\begin{aligned} \kappa_{D,X} : \text{colim}(D^X) &\rightarrow (X \rightarrow \text{colim } D) \\ \kappa_{D,X}[i, f]_\sim &= (\nu_D)_i \circ f \end{aligned} \quad (5.32)$$

One says that *taking a power by $X : \mathcal{U}^I$ preserves \mathbf{Size} -indexed colimits* if for all \mathbf{Size} -indexed diagrams (D, δ) in \mathcal{U}^I the function $\kappa_{D,X}$ is an isomorphism. Then we have:

If \mathcal{U} is a universe satisfying IWISC in a topos with natural numbers object, then given $A, I : \mathcal{U}$ and $B : A \rightarrow \mathcal{U}^I$, there exists a type of sizes $\mathbf{Size} : \mathcal{U}$ with the property that for all $a : A$, taking a power by a family $Ba : \mathcal{U}^I$ preserves \mathbf{Size} -indexed colimits.

The proof of this is similar to the proof of Theorem 5.7 and can be found in the Agda development accompanying this paper [FPS21].

Given a \mathbf{Size} -indexed diagram (D, δ) in \mathcal{U}^I , and an I -indexed signature Σ as in (3.27), then we get another \mathbf{Size} -indexed diagram, $(S_\Sigma \circ D, S_\Sigma \circ \delta)$, by composing with the polynomial endofunctor $S_\Sigma : \mathcal{U}^I \rightarrow \mathcal{U}^I$:

$$\begin{aligned} (S_\Sigma \circ D)_i &\stackrel{\text{def}}{=} S_\Sigma(D_i) & (i : \mathbf{Size}) \\ (S_\Sigma \circ \delta)_{i,j} &\stackrel{\text{def}}{=} S_\Sigma(\delta_{i,j}) & (i, j : \mathbf{Size}) \end{aligned} \quad (5.33)$$

Applying S_Σ to the I -indexed version of (5.14) gives a cocone under $(S_\Sigma \circ D, S_\Sigma \circ \delta)$ with vertex $S_\Sigma(\text{colim } D)$ and this induces a family of functions in \mathcal{U}^I

$$\kappa_{D,\Sigma} : \text{colim}(S_\Sigma \circ D) \rightarrow S_\Sigma(\text{colim } D) \quad (5.34)$$

One says that *the polynomial endofunctor $S_\Sigma : \mathcal{U}^I \rightarrow \mathcal{U}^I$ preserves Size-indexed colimits* if $\kappa_{D,\Sigma}$ is a family of isomorphisms, for all diagrams (D, δ) . Then the indexed generalization of Corollary 5.10 is:

*In a topos with natural numbers object, given an indexed signature $\Sigma = (I, A, B)$ in a universe \mathcal{U} satisfying IWISC, there exists a type of sizes **Size** with the property that the associated polynomial endofunctor $S_\Sigma : \mathcal{U}^I \rightarrow \mathcal{U}^I$ preserves Size-indexed colimits.*

This is proved as a corollary of the indexed version of Theorem 5.7 given above, and can also be found in the accompanying Agda development [FPS21].

6. CONSTRUCTION OF QWI-TYPES

We aim to prove the following theorem about existence of QWI-types (Definition 3.5):

Theorem 6.1. *Suppose \mathcal{U} is a universe satisfying IWISC in a topos with natural numbers object. Then for every indexed signature and system of equations in \mathcal{U} , there exists a QWI-type for it.*

The proof follows from the cocontinuity results of the previous section (the indexed versions of Theorem 5.7 and Corollary 5.10 given in Remark 5.11). For simplicity, here we only give the proof for QW-types (Section 3.2), that is, for the non-indexed $I = \mathbb{1}$ case of signatures. The general case is similar, but notationally more involved since there are indexes ranging both over an index type and over sizes. The proof for the general, indexed case can be found in our Agda development [FPS21].

So in this section we fix a signature $\Sigma = (A : \mathcal{U}, B : A \rightarrow \mathcal{U})$ and system of equations $\varepsilon = (E : \mathcal{U}, V : E \rightarrow \mathcal{U}, l, r : \prod_{e:E} T_\Sigma(V e))$ over it in some universe \mathcal{U} satisfying IWISC in a topos with natural numbers object.

6.1. Motivating the construction. We noted at the start of Section 4 that QW-types can be constructed in the category of ZFC sets as initial algebras for possibly infinitary equational theories by first forming the W-type of terms of the theory and then quotienting that by the congruence relation generated by the equations of the theory. AC is used when constructing the algebra structure of the quotient, because the signature's arities may be infinite. To avoid this use of AC, instead of forming all terms in one go and then quotienting, we consider interleaving quotient formation with the construction of terms of free algebras for equational systems (cf. the categorical construction by Fiore and Hur [FH09]).

Figure 2 gives the idea, using the constructors η and σ from (3.6) and Agda-like notation for inductive definitions. We would like to construct the QW-type for (Σ, ε) as a quotient $\text{QW} \stackrel{\text{def}}{=} W/\sim$, but now the type $W : \mathcal{U}$ and the relation $_ \sim _ : W \rightarrow W \rightarrow \mathbf{Prop}$ are mutually inductively defined, with constructors as indicated in the figure. Note that whereas the construction in ZFC uses AC to get an S_Σ -algebra structure for QW, here we get one trivially from the constructor $q\tau : T_\Sigma(W/\sim) \rightarrow W$:

$$S_\Sigma(\text{QW}) \equiv S_\Sigma(W/\sim) \xrightarrow{S_\Sigma \eta} S_\Sigma(T_\Sigma(W/\sim)) \xrightarrow{\sigma} T_\Sigma(W/\sim) \xrightarrow{q\tau} W \xrightarrow{[_]\sim} W/\sim \equiv \text{QW} \quad (6.1)$$

```

mutual
  data W :  $\mathcal{U}$  where
    q $\tau$  :  $T_\Sigma(W/\sim) \rightarrow W$ 
  data  $\_ \sim \_ : W \rightarrow W \rightarrow \text{Prop}$  where
    q $\varepsilon$  :  $\forall (e : E). \forall (\rho : V e \rightarrow W/\sim). q\tau(T_\Sigma \rho (l e)) \sim q\tau(T_\Sigma \rho (r e))$ 
    q $\eta$  :  $\forall (t : T_\Sigma(W/\sim)). q\tau(\eta[q\tau t]_\sim) \sim q\tau t$ 
    q $\sigma$  :  $\forall (a : A). \forall (b : B a \rightarrow T_\Sigma(W/\sim)). q\tau(\sigma(a, b)) \sim q\tau(\sigma(a, \eta \circ [\_ ]_\sim \circ q\tau \circ b))$ 
  QW =  $W/\sim$ 

```

Figure 2: First attempt at constructing QW-types

Furthermore the property $q\varepsilon$ of \sim in Figure 2 ensures that the S_Σ -algebra W/\sim satisfies the equational system ε . The use of T_Σ rather than S_Σ in the domain of $q\tau$ seems necessary for this method of construction to go through (once we have fixed up the problems mentioned in the next paragraph); but it does mean that as well as $q\varepsilon$, we have to impose the conditions $q\eta$ and $q\sigma$ to ensure that W/\sim has a S_Σ -algebra structure, or equivalently, an algebra structure for the monad T_Σ .

Note that the domain of the constructor $q\tau$ combines $T_\Sigma(_)$ with $_/_$. While the first is unproblematic for inductive definitions, the second is not: if one thinks of the semantics of inductively defined types in terms of initial algebras of endofunctors, it is not clear what endofunctor (in some class known to have initial algebras) is involved here, given that both arguments to $_/_$ are being defined simultaneously. Agda uses a notion of “strict positivity” as a conservative approximation for such a class of functors; and one can instruct Agda to regard quotienting as a strictly positive operation through the use of its POLARITY declarations. If one does so, then a definition like the one in Figure 2 is accepted by Agda. The semantic justification for regarding quotients as strictly positive constructions needs further investigation. We avoid the need for that here and replace the attempt to define W and \sim inductively by a size-indexed version that uses definition by well-founded recursion over a type of sizes as in the previous section. This also avoids another difficulty with Figure 2: even if one can define W and \sim inductively, one still has to verify that W/\sim has the universal property (3.13)–(3.15) required of a QW-type. In particular, there is an obvious recursive definition of qwrec following the shape of the inductive definition in Figure 2, but it is not at all clear why this recursive definition is terminating (i.e. gives a well-defined, total function). Well-founded recursion over sizes will solve this problem as well.

6.2. QW-type via sizes. We are given a signature $\Sigma = (A : \mathcal{U}, B : A \rightarrow \mathcal{U})$ and system of equations $\varepsilon = (E : \mathcal{U}, V : E \rightarrow \mathcal{U}, l, r : \prod_{e:E} T_\Sigma(V e))$. Let $\text{Size} : \mathcal{U}$ be the type of sizes whose existence is guaranteed by Theorem 5.7 when in the theorem we take A to be $AE \stackrel{\text{def}}{=} A + E$ and B to be the function $BV : AE \rightarrow \mathcal{U}$ mapping $a : A$ to Ba and $e : E$ to Ve . It follows (as in the proof of Corollary 5.10) that S_Σ preserves Size -indexed colimits.

Definition 6.2. A *Size-indexed Σ -structure* in \mathcal{U} is specified by a family of types $D : \text{Size} \rightarrow \mathcal{U}$ equipped with functions

$$\tau_{j,i} : T_\Sigma(D_j) \rightarrow D_i \quad \text{for all } i, j : \text{Size} \text{ with } j < i \quad (6.2)$$

Similarly, for each $i : \text{Size}$, a *($\downarrow i$)-indexed Σ -structure* is the same thing, except with D only defined on the subsemicategory $\downarrow i$ (5.5) rather than the whole of Size . Clearly, given a

Size-indexed Σ -structure (D, τ) , for each $i : \text{Size}$ we get a $(\downarrow i)$ -indexed one $(D(\downarrow i), \tau(\downarrow i))$ by restriction.

If $i : \text{Size}$ and (D, τ) is a $(\downarrow i)$ -indexed Σ -structure, let $\diamond_i D : \mathcal{U}$ be the quotient type (see Figure 1)

$$\diamond_i D \stackrel{\text{def}}{=} \left(\sum_{j < i} T_\Sigma D_j \right) / R_i \quad (6.3)$$

with the relation $R_i : (\sum_{j < i} T_\Sigma D_j) \rightarrow (\sum_{j < i} T_\Sigma D_j) \rightarrow \text{Prop}$ defined by:

$$\begin{aligned} R_i(j, t)(k, t') &\stackrel{\text{def}}{=} \\ &(k = j \wedge \exists(e : E). \exists(\rho : V e \rightarrow D_j). t = T_\Sigma \rho(l e) \wedge t' = T_\Sigma \rho(r e)) \\ &\vee (k < j \wedge t = \eta(\tau_{k,j} t')) \\ &\vee (k < j \wedge \exists(a : A). \exists(b : B a \rightarrow T_\Sigma D_k). t = \sigma(a, b) \wedge t' = \sigma(a, \eta \circ \tau_{k,j} \circ b)) \end{aligned} \quad (6.4)$$

(The three clauses in the above definition correspond to the three constructors $q\varepsilon$, $q\eta$ and $q\sigma$ in Figure 2.) We will see that the QW-type for (Σ, ε) can be constructed from Size-indexed Σ -structures (D, τ) satisfying the following fixed-point property.

Definition 6.3. A Size-indexed Σ -structure (D, τ) is a \diamond -fixed point if for all $i : \text{Size}$

$$D_i = \diamond_i(D(\downarrow i)) \quad (6.5)$$

and for all $j < i$ and $t : T_\Sigma D_j$

$$\tau_{j,i} t = [j, t]_{R_i} \quad (6.6)$$

Suppose that (D, τ) is a \diamond -fixed point. Because of (6.5) and (6.6), for $i, j : \text{Size}$ with $i < j$ the functions

$$\delta_{i,j} : D_i \rightarrow D_j \quad \delta_{i,j} \stackrel{\text{def}}{=} \tau_{i,j} \circ \eta \quad (6.7)$$

satisfy

$$\delta_{i,j}([k, t]_{R_i}) = [k, t]_{R_j} \quad (\text{for all } k < i \text{ and } t : T_\Sigma D_k) \quad (6.8)$$

In particular they satisfy composition (5.13) and so (D, δ) is a Size-indexed diagram in \mathcal{U} whose colimit we can form as in Section 5.1. We prove that this colimit

$$\text{QW} \stackrel{\text{def}}{=} \text{colim } D \quad (6.9)$$

has the structure (3.11)–(3.15) of a QW-type for the signature (Σ, ε) .

qwintro: Given how we chose Size at the start of this section, from the (proof of) Corollary 5.10 we have that $S_\Sigma : \mathcal{U} \rightarrow \mathcal{U}$ preserves Size-indexed colimits. We claim that the functions

$$S_\Sigma(D_i) \xrightarrow{S_\Sigma \eta} S_\Sigma(T_\Sigma D_i) \xrightarrow{\sigma} T_\Sigma D_i \xrightarrow{\tau_{i,i^+}} D_{i^+} \xrightarrow{(\nu_D)_{i^+}} \text{colim } D \quad (6.10)$$

form a cocone under the diagram $S_\Sigma \circ D$ and hence induce a function $\text{colim}(S_\Sigma \circ D) \rightarrow \text{colim } D$; then we obtain $\text{qwintro} : S_\Sigma(\text{colim } D) \rightarrow \text{colim } D$ by composing this with the isomorphism $S_\Sigma(\text{colim } D) \cong \text{colim}(S_\Sigma \circ D)$ from Corollary 5.10. That the functions in (6.10) form a cocone for $S_\Sigma \circ D$ follows from the fact that the functions in (6.7) satisfy for all sizes $i < j$

$$\forall(t : T_\Sigma(D_i)). \exists(k : \text{Size}). j < k \wedge \tau_{j,k}(T_\Sigma \delta_{i,j}(t)) = \tau_{i,k}(t) \quad (6.11)$$

from which it follows that $(\nu_D)_{j^+} \circ \tau_{j,j^+} \circ T_\Sigma \delta_{i,j} = (\nu_D)_{i^+} \circ \tau_{i,i^+}$ and hence also the cocone property. Property (6.11) can be proved by induction on the structure of $t : T_\Sigma(D_i)$, with the $t = \sigma(a, b)$ case of (3.6) proved using the fact that **Size** has \llcorner -upper bounds for families of sizes indexed by *Ba* (Remark 5.8).

qwequate: Given $e : E$ and $\rho : Ve \rightarrow \text{colim } D$, by Theorem 5.7 there exist $i : \mathbf{Size}$ and $\rho' : Ve \rightarrow D_i$ with $\rho = (\nu_D)_i \circ \rho'$. By standard properties of the bind operation $\gg=$ (3.7) (which depends implicitly on the S_Σ -algebra structure **qwintro** that we have just constructed), we have

$$\begin{aligned} (le \gg= \rho) &= (le \gg= (\nu_D)_i \circ \rho') = (T_\Sigma \rho' (le) \gg= (\nu_D)_i) \\ (re \gg= \rho) &= (re \gg= (\nu_D)_i \circ \rho') = (T_\Sigma \rho' (re) \gg= (\nu_D)_i) \end{aligned} \quad (6.12)$$

From the definition of **qwintro** and (6.6) it follows that for any $t : T_\Sigma D_i$, there is a proof of

$$(t \gg= (\nu_D)_i) = (\nu_D)_{i^+} (\tau_{i,i^+} t) \quad (6.13)$$

So from above we have that $(le \gg= \rho) = (\nu_D)_{i^+} (\tau_{i,i^+} (T_\Sigma \rho' (le)))$ and $(re \gg= \rho) = (\nu_D)_{i^+} (\tau_{i,i^+} (T_\Sigma \rho' (re)))$. Since from (6.6) and the first clause in the definition of R_i (6.4) we also have $\tau_{i,i^+} (T_\Sigma \rho' (le)) = \tau_{i,i^+} (T_\Sigma \rho' (re))$, it follows that there is a proof of $(le \gg= \rho) = (re \gg= \rho)$.

qwrec: Given an S_Σ -algebra $(X, \alpha) : \sum_{X:\mathcal{U}} (S_\Sigma X \rightarrow X)$ satisfying the system of equations ε , the function **qwrec** : $\text{colim } D \rightarrow X$ is induced by a cocone of functions $r : \prod_i (D_i \rightarrow X)$ under the diagram D , defined by well-founded recursion (Proposition 5.3). More precisely, a strengthened ‘‘recursion hypothesis’’ is needed: instead of $\prod_i (D_i \rightarrow X)$ we use $\prod_i F_i$ where

$$F_i \stackrel{\text{def}}{=} \{f : D_i \rightarrow X \mid \forall j < i. \forall (t : T_\Sigma D_j). (t \gg= (f \circ \delta_{j,i})) = f([j, t]_{R_i})\} \quad (6.14)$$

(The definition uses α implicitly in $\gg=$ and relies on the fact (6.5) that $D_i = \diamond_i (D \downarrow i)$.) For each $i : \mathbf{Size}$, if we have $r_j : F_j$ for all $j < i$, then we get a function $r_i : F_i$ well-defined by

$$r_i([j, t]_{R_i}) \stackrel{\text{def}}{=} t \gg= r_j \quad \text{for all } j < i \text{ and } t : T_\Sigma D_j \quad (6.15)$$

(The defining property of F_i is needed to see that the right-hand side of this definition respects the relation R_i .) Hence by well-founded recursion (Proposition 5.3) we get an element $r : \prod_i F_i$. One can prove $\forall i. \forall j < i. r_j = r_i \circ \delta_{j,i}$ by well-founded induction (5.2); so r is a cocone and induces a function **qwrec** : $\text{colim } D \rightarrow X$.

qwrechom: We noted at the start of this section that by choice of **Size**, the functor S_Σ preserves **Size**-indexed colimits. So to prove that

$$\begin{array}{ccc} S_\Sigma(\text{colim } D) & \xrightarrow{\text{qwintro}} & \text{colim } D \\ S_\Sigma \text{qwrec} \downarrow & & \downarrow \text{qwrec} \\ S_\Sigma X & \xrightarrow{\alpha} & X \end{array} \quad (6.16)$$

commutes, by the definitions of **qwintro** and **qwrec**, it suffices to prove that

$$\begin{array}{ccc} S_\Sigma D_i & \xrightarrow{\tau_{i,i^+} \circ \sigma \circ S_\Sigma \eta} & D_{i^+} \\ S_\Sigma r_i \downarrow & & \downarrow r_{i^+} \\ S_\Sigma X & \xrightarrow{\alpha} & X \end{array} \quad (6.17)$$

does for each $i : \mathbf{Size}$. But each $(a, b) : S_\Sigma D_i$ is mapped by $\tau_{i,i^+} \circ \sigma \circ S_\Sigma \eta$ to $[i, \sigma(a, \eta \circ b)]_{R_{i^+}}$ (using the fact that $D_{i^+} = \diamond_{i^+}(D \downarrow i^+)$); and by (6.15), that is mapped by r_{i^+} to $\sigma(a, \eta \circ b) \gg r_i$, which is indeed equal to $\alpha(S_\Sigma r_i(a, b))$ by definition of \gg (3.7) and the action of S_Σ on functions (3.3).

qwuniq: If $h : \text{colim } D \rightarrow X$ is a morphism of S_Σ -algebras, then one can prove by well-founded induction for $<$ that $\forall i. h \circ (\nu_D)_i = r_i$ holds: for if we have $h \circ (\nu_D)_j = r_j$ for all $j < i$, then for any $[j, t]_{R_i}$ in $D_i = \diamond_i(D \downarrow i)$

$$\begin{aligned} r_i([j, t]_{R_i}) &\stackrel{\text{def}}{=} t \gg r_j && (6.15) \\ &= t \gg (h \circ (\nu_D)_j) && \text{by induction hypothesis} \\ &= h(t \gg (\nu_D)_j) && \text{since } h \text{ is a morphism of } S_\Sigma\text{-algebras} \\ &= h((\nu_D)_{j^+}[j, t]_{R_{j^+}}) && \text{by (6.13) and (6.6)} \\ &= h((\nu_D)_i[j, t]_{R_i}) && \text{since by (6.8), } \delta_{j^+, j^+ \sqcup^s i}([j, t]_{R_{j^+}}) = \\ & && [j, t]_{R_{j^+ \sqcup^s i}} = \delta_{i, j^+ \sqcup^s i}([j, t]_{R_i}). \end{aligned}$$

So by well-founded induction, $h \circ (\nu_D)_i = r_i$ holds for all $i : \mathbf{Size}$, and hence by definition of **qwrec** and the uniqueness part of the universal property of colimits we have $h = \mathbf{qwrec}$.

Thus we have proved:

Proposition 6.4. *Let \mathbf{Size} be the type of sizes defined at the start of this section, whose existence is guaranteed by Theorem 5.7. If the \mathbf{Size} -indexed Σ -structure (D, τ) is a \diamond -fixed point, then $\text{colim } D$ has the structure of a QW -type for the signature (Σ, ε) . \square*

Now we can complete the proof of the main theorem:

Proof of non-indexed version of Theorem 6.1. In view of Proposition 6.4, it suffices to construct a \mathbf{Size} -indexed Σ -structure which is a \diamond -fixed point in the sense of Definition 6.3.

For each $i : \mathbf{Size}$, say that a $(\downarrow i)$ -indexed Σ -structure (Definition 6.2) is an *upto- i* \diamond -fixed point if

$$\forall j < i. D_j = \diamond_j(D \downarrow j) \wedge \forall k < j. \forall t. \tau_{k,j} t == [k, t]_{R_j} \quad (6.18)$$

(cf. (6.5) and (6.6)). Note that:

- (A) *Given $j < i$, any upto- i \diamond -fixed point restricts to an upto- j \diamond -fixed point.*
- (B) *For all i , any two upto- i \diamond -fixed points are equal* (proof by well-founded induction (5.2)).

Using these two facts, it follows by well-founded recursion (Proposition 5.3) that there is an upto- i \diamond -fixed point for all $i : \mathbf{Size}$. For if $(D^{(j)}, \tau^{(j)})$ is an upto- j \diamond -fixed point for all $j < i$, then we get $D^{(i)} : \downarrow i \rightarrow \mathcal{U}$ by defining for each $j < i$

$$(D^{(i)})_j \stackrel{\text{def}}{=} \diamond_j(D^{(j)}) \quad (6.19)$$

If $k < j < i$, then by (A) we have that $D^{(j)}(\downarrow k)$ is an upto- k \diamond -fixed point and hence by (B) that $D^{(j)}(\downarrow k) = D^{(k)}$. So together with (6.18) this gives:

$$(D^{(i)})_k \stackrel{\text{def}}{=} \diamond_k(D^{(k)}) = \diamond_k(D^{(j)}(\downarrow k)) = (D^{(j)})_k \quad (6.20)$$

Hence we can define $(\tau^{(i)})_{k,j} : T_\Sigma((D^{(i)})_k) \rightarrow (D^{(i)})_j$ by $(\tau^{(i)})_{k,j} t \stackrel{\text{def}}{=} [k, t]_{R_j}$ and this makes $(D^{(i)}, \tau^{(i)})$ into a $(\downarrow i)$ -indexed Σ -structure which by construction is an upto- i \diamond -fixed point. Thus by well-founded recursion (Proposition 5.3) we have an upto- i \diamond -fixed point $D^{(i)}$ for all $i : \text{Size}$.

Let $D : \text{Size} \rightarrow \mathcal{U}$ be given by $D_i \stackrel{\text{def}}{=} \diamond_i(D^{(i)})$. If $j < i$, then by (A) and (B) we have $D^{(i)}(\downarrow j) = D^{(j)}$ and together with (6.18) this gives:

$$D_j \stackrel{\text{def}}{=} \diamond_j(D^{(j)}) = \diamond_j(D^{(i)}(\downarrow j)) = (D^{(i)})_j \quad (6.21)$$

So we can define $\tau_{j,i} : T_\Sigma D_j \rightarrow D_i$ by $\tau_{j,i} t \stackrel{\text{def}}{=} [j, t]_{R_i}$. This makes (D, τ) into a Σ -structure which by construction is a \diamond -fixed point. \square

7. ENCODING QITs AS QWI-TYPES

The general notion of indexed quotient inductive type (QIT) was discussed by example in the Introduction. We wish to show that a wide variety of QITs can be expressed as QWI-types, namely those which do not use conditional equality constructors (as in the example in (1.2)). We first introduce a schema for such QITs that combines desirable features of ones that occur in the literature. As in the previous section, for simplicity sake we will confine our attention to the non-indexed case of QITs and QW-types.

7.1. General QIT schemas and encodings. Basold, Geuvers, and van der Weide [BGvdW17] present a schema for infinitary QITs that do not support conditional path equations. Constructors are defined by arbitrary polynomial endofunctors built up using (non-dependent) products and sums, which means in particular that parameters and arguments can occur in any order; however, they require constructors to be in uncurried form. They also construct a model of simple 1-cell complexes and other non-recursive QITs.

Dybjer and Moeneclaey [DM18, Sections 3.1 and 3.2] present a schema for finitary QITs that does allow curried constructors (and also supports *conditional* path equations), but requires all parameters to appear before all arguments. This contrasts with the more convenient schema for regular inductive types in Agda, which allows parameters and arguments in any order.

Kaposi, Kovács, and Altenkirch [KKA19] define an encoding of finitary quotient inductive-inductive (QIIT) types, called the *theory of signatures* (ToS), which is a restriction of the *theory of codes* for HIITs [KK18]. In the ToS a QI(I)T is encoded as a context of a small internal type theory. This encoding is not quite as convenient as the schema for regular inductive types in Agda, but, when using named variables, it is much closer to Agda than QWI-types are. They also construct a model for finitary QIITs (and Kovács and Kaposi [KK20] reduce the problem of modelling infinitary QIITs to just one “universal” QIIT – a generalised ToS).

Building on the first two schemas mentioned above, we provide a schema for infinitary, non-conditional QITs combining the arbitrarily ordered parameters and arguments of the first [BGvdW17] with the curried constructors of the second [DM18].

The following definition should be read as an extension of a formalisation of the type theory described in Section 2 in terms of typing contexts (Γ, Δ, \dots) and various judgements-in-context (such as $\Gamma \vdash a : A$, $\Gamma \vdash a = a' : A$, etc.); see the HoTT Book [Uni13, Appendix].

In the following we fix the context Γ in which the QIT, Q , ($Q \notin \Gamma$) is defined, whereas Δ , A , B , H , K , a , b , x , y , etc. are metavariables. When deriving an element or equality constructor, Δ will always be equal to Γ or an extension of it.

A type *strictly-positive in Q* is made up with Π -types and Σ -types and can use Q and constants or variables in the context, provided that Q never occurs on the LHS of a Π -type, and that the RHS of a Σ -type never depends on Q , even if it occurs on the LHS.

$$\begin{array}{c}
 \boxed{\Delta \vdash K \text{ StrPos}} \\
 \\
 \frac{\Delta \vdash B : \mathcal{U}}{\Delta \vdash B \text{ StrPos}} \text{ PARAM} \qquad \frac{\Delta \vdash B : \mathcal{U} \quad \Delta, b : B \vdash K \text{ StrPos}}{\Delta \vdash \prod_{b:B} K \text{ StrPos}} \text{ STRPOSFUN} \\
 \\
 \frac{}{\Delta \vdash Q \text{ StrPos}} \text{ INDARG} \qquad \frac{\Delta \vdash A \text{ StrPos} \quad \Delta, a : A[\mathbb{1}/Q] \vdash K \text{ StrPos}}{\Delta \vdash \sum_{a:A} K' \text{ StrPos}} \text{ PROD}
 \end{array}$$

Where K' is found from K by recursively replacing each sub-term of type $\mathbb{1}$, $A \rightarrow \mathbb{1}$, etc. with the corresponding unique term 0 , $!$, etc.

Note: We can see that these rules give strictly-positive types because: (1) at this point $\Delta \vdash Q : \mathcal{U}$ is not derivable since $Q \notin \Gamma$ and no rule for StrPos adds Q to the context, so Q can never appear in the LHS of a Π -type. Also (2) the PROD rule ensures that abstracting with Σ cannot result in dependencies of Q in the codomain.

The type of an element constructor is an (iterated) Π -type with codomain Q , with zero or more strictly-positive arguments.

$$\begin{array}{c}
 \boxed{\Delta \vdash H \text{ ElConstr}} \\
 \\
 \frac{\Delta, Q : \mathcal{U} \vdash H \text{ ElType}}{\Delta \vdash H \text{ ElConstr}} \text{ ELCONSTR} \\
 \\
 \frac{}{\Delta \vdash Q \text{ ElType}} \text{ ELCoDOM} \qquad \frac{\Delta[\mathbb{1}/Q] \vdash K \text{ StrPos} \quad \Delta, a : K' \vdash H \text{ ElType}}{\Delta \vdash \prod_{a:K'} H \text{ ElType}} \text{ ELARG}
 \end{array}$$

Where $\Delta[\mathbb{1}/Q]$ is the context that results after replacing occurrences of Q in Δ by the type $\mathbb{1}$.

Note: Deriving $\Gamma \vdash \prod_{a:K'} Q \text{ ElConstr}$ via ELARG for some strictly-positive K does not imply $\Gamma \vdash \prod_{a:K'} Q : \mathcal{U}$ since $\Gamma \not\vdash Q : \mathcal{U}$ (since $Q \notin \Gamma$), and also $\Gamma \not\vdash K' : \mathcal{U}$ in general since K' may contain Q . This distinction also applies to the following judgements.

The type of an equality constructor is an (iterated) Π -type with codomain $x =_Q y$, with zero or more strictly-positive arguments. Note that each of the element constructors are added to the context and can be used to derive $x : Q$ and $y : Q$.

$$\begin{array}{c}
 \boxed{\Delta \vdash K \text{ EqConstr}} \\
 \\
 \frac{\Delta, Q : \mathcal{U}, c_1 : C_1, \dots, c_m : C_m \vdash K : \text{EqType}}{\Delta \vdash K \text{ EqConstr}} \text{ EQCONSTR} \\
 \\
 \frac{\Delta \vdash x : Q \quad \Delta \vdash y : Q}{\Delta \vdash x = y \text{ EqType}} \text{ EQCoDOM} \qquad \frac{\Delta[\mathbb{1}/Q] \vdash K \text{ StrPos} \quad \Delta, a : K' \vdash H \text{ EqType}}{\Delta \vdash \prod_{a:K'} H \text{ EqType}} \text{ EQARG}
 \end{array}$$

Figure 3: Rules for QIT element and equality constructors.

Definition 7.1. A (*non-conditional*) QIT, Q , in a context Γ , $Q \notin \Gamma$ is specified by a list of element constructors $c_1 : C_1, \dots, c_m : C_m$, followed by a list of equality constructors $d_1 : D_1, \dots, d_n : D_n$, where the C_i s and D_j s are derived as $\Gamma \vdash C_i \text{ ElConstr}$ and $\Gamma \vdash D_j \text{ EqConstr}$ respectively, according to the rules in Figure 3.

Given a list of element and equality constructors built up according to these rules, the newly-defined QIT, Q , has formation rule

$$\overline{\Gamma \vdash Q : \mathcal{U}}$$

for the specific context Γ in which it was defined. And it has an introduction rule for each element constructor and each equality constructor:

$$\overline{\Gamma \vdash c_1 : C_1} \quad \dots \quad \overline{\Gamma \vdash c_m : C_m} \quad \overline{\Gamma \vdash d_1 : D_1} \quad \dots \quad \overline{\Gamma \vdash d_n : D_n}$$

We show how to derive elimination and computation rules for Q from these formation and introduction rules (compare with Basold, Geuvers, and van der Weide [BGvdW17, Definition 10]) after first giving an example of how the schema is instantiated.

Example 7.2. Consider multisets as in example (1.1) with two element constructors $[]$ and $_ :: _$, and one equality constructor swap . Given the context containing parameter $X : \mathcal{U}$, we define the QIT $\cdot, X : \mathcal{U} \vdash \text{Bag} : \mathcal{U}_1$ by providing those constructors along with their types such that they are derived by the rules in Figure 3:

Let $\Gamma \stackrel{\text{def}}{=} (\cdot, X : \mathcal{U})$ and $\Delta \stackrel{\text{def}}{=} (\Gamma, \text{Bag} : \mathcal{U}) = (\cdot, X : \mathcal{U}, \text{Bag} : \mathcal{U})$; so that, by definition, $\Delta[\mathbb{1}/\text{Bag}] = \Gamma$.

- $[]$

$$\frac{\overline{\Delta \vdash \text{Bag ElType}} \quad \text{ELCoDom}}{\Gamma \vdash \text{Bag ElConstr}} \quad \text{ELCONSTR}$$

- $_ :: _$

$$\frac{\frac{\overline{\Gamma \vdash X : \mathcal{U}} \quad \text{PARAM} \quad \overline{\Gamma, a : X \vdash \text{Bag StrPos}} \quad \text{INDARG} \quad \overline{\Delta, a : X, b : \text{Bag} \vdash \text{Bag ElType}} \quad \text{ELCoDom}}{\Delta, a : X \vdash (\text{Bag} \rightarrow \text{Bag}) \text{ ElType}} \quad \text{ELARG}}{\Gamma \vdash (X \rightarrow \text{Bag} \rightarrow \text{Bag}) \text{ ElConstr}} \quad \text{ELCONSTR}$$

Let

$$\begin{aligned} \Theta &\stackrel{\text{def}}{=} \Delta, [] : \text{Bag}, _ :: _ : X \rightarrow \text{Bag} \rightarrow \text{Bag} \\ \Xi &\stackrel{\text{def}}{=} \Theta[\mathbb{1}/\text{Bag}] \stackrel{\text{def}}{=} \Gamma, [] : \mathbb{1}, _ :: _ : X \rightarrow \mathbb{1} \rightarrow \mathbb{1} \\ \Phi &\stackrel{\text{def}}{=} \Theta, x y : X, zs : \text{Bag} \\ \text{eqn} &\stackrel{\text{def}}{=} x :: y :: zs = y :: x :: zs \end{aligned}$$

- swap

$$\frac{\overline{\Xi \vdash X : \mathcal{U}} \quad \overline{\Xi, x : X \vdash X \text{ StrPos}} \quad \overline{\Xi, x y : X \vdash \text{Bag StrPos}} \quad \overline{\Phi \vdash x :: y :: zs : \text{Bag}} \quad \overline{\Phi \vdash y :: x :: zs : \text{Bag}} \quad \text{EqCoDom}}{\overline{\Theta, x : X \vdash \prod_{y:X} \prod_{zs:\text{Bag}} \text{eqn EqType}} \quad \text{EqARG}} \quad \text{EqARG} \quad \text{EqCONSTR}$$

$$\frac{\overline{\Theta \vdash \prod_{x y : X} \prod_{zs : \text{Bag}} \text{eqn EqType}} \quad \text{EqCONSTR}}{\Gamma \vdash \prod x y : X \prod_{zs : \text{Bag}} \text{eqn EqConstr}} \quad \text{EqCONSTR}$$

Definition 7.3 (Elimination and computation). The *arguments* of a constructor C_j , written $\text{Arg}(C_j)$, is the list of K' for all strictly-positive types K introduced with the rule ELARG.

Given a QIT defined by constructors $c_1, \dots, c_m, d_1, \dots, d_n$ as above, the *underlying inductive type* $[Q]$ is the inductive type defined by only the element constructors c_1, \dots, c_m , ignoring the equalities; then the *underlying W-type* is the W-type that encodes this inductive type. (Recall that every strictly-positive description of a polynomial endofunctor gives rise to a W-type with the same initial algebra; see Dybjer [Dyb97], and for the more general indexed and nested case see Altenkirch et al. [AGHMM15].)

Define *leaf application* on a strictly-positive term $a : A$ (that is, a term with a strictly-positive type) for a function $f : \prod_{q:Q} Rq$ into some type R , written $f \$ a$, by induction on the StrPos structure of A :

$$\begin{array}{ll}
\text{INDARG} & f \$ x \stackrel{\text{def}}{=} f x \\
\text{PARAM} & f \$ b \stackrel{\text{def}}{=} b \\
\text{PROD} & f \$ (a, b) \stackrel{\text{def}}{=} (f \$ a, f \$ b) \\
\text{STRPOSFUN} & f \$ g \stackrel{\text{def}}{=} (f \$ _) \circ g
\end{array} \tag{7.1}$$

In order to define the elimination and computation rules we must first define the induction step. First, given a ‘‘motive’’ $\Gamma \vdash P : Q \rightarrow \mathcal{U}$, define the type $P' \stackrel{\text{def}}{=} \sum_{x:Q} P x$. Now given a strictly-positive argument A , define the type A' (it will be (one of) the induction hypotheses) by induction on the structure of A , replacing each occurrence of Q (INDARG) with P' . Terms $a' : A'$ are also strictly-positive and admit leaf application for functions of type $\prod_{p':P'} S p'$ for some type S .

To find the induction step \widehat{C}_j for each element constructor $c_j : C_j$, replace each of the strictly-positive arguments $\prod_{a_1:A_1} \dots \prod_{a_p:A_p}$ with $\prod_{a_1:A'_1} \dots \prod_{a_p:A'_p}$, as defined above, and replace the target Q of the constructor with $P(c_j(\pi_1 \$ a_1) \dots (\pi_1 \$ a_p))$.

The induction cases for each of the element constructors are then $h_1 : \widehat{C}_1, \dots, h_m : \widehat{C}_m$, and these provide an eliminator $\text{elim } h_1 \dots h_m$ for the underlying inductive type $[Q]$.

Given h_1, \dots, h_m , define \widehat{D}_k , for each equality constructor $d_k : D_k$, in the same way. Replace the homogeneous equality type with a heterogeneous equality type and replace the endpoints l, r of the equality with \widehat{l}, \widehat{r} inductively defined by cases depending on if the abstracted variable is a constructor or not:

$$\begin{array}{l}
\widehat{c}_j \stackrel{\text{def}}{=} h_j \\
\widehat{x} \stackrel{\text{def}}{=} x : P'_i
\end{array} \tag{7.2}$$

The elimination rule is then:

$$\frac{\begin{array}{c} \Gamma \vdash P : Q \rightarrow \mathcal{U} \\ \Gamma \vdash h_1 : \widehat{C}_1 \quad \dots \quad \Gamma \vdash h_m : \widehat{C}_m \\ \Gamma \vdash p_1 : \widehat{D}_1 \quad \dots \quad \Gamma \vdash p_n : \widehat{D}_n \end{array}}{\text{qitelim } h_1 \dots h_m p_1 \dots p_n : \prod_{x:Q} P x} \text{QITELIM}$$

Finally the computation rules are, for each element constructor $c_i : C_i$,

$$\frac{\text{same hypotheses as QITELIM} \quad \Gamma \vdash a_1, \dots, a_p : \text{Arg}(C_i)}{\text{qitelim } h_1 \dots h_m p_1 \dots p_n (c_i a_1 \dots a_p)} \text{QITCOMP} \\
=_{P(c_i a_1 \dots a_p)} h_i(a_1, (\text{qitelim } \dots) \$ a_1) \dots (a_p, (\text{qitelim } \dots) \$ a_p)$$

Example 7.4. The elimination rule for $X : \mathcal{U} \vdash \mathbf{Bag}$ is:

$$\frac{\begin{array}{l} \Gamma \vdash P : \mathbf{Bag} \rightarrow \mathcal{U} \quad \Gamma \vdash \mathit{nil} : P [] \quad \Gamma \vdash \mathit{cons} : \prod_{x:X} \prod_{xs':(\sum_{zs:\mathbf{Bag}} P \, xs)} P(x :: (\pi_1 \, xs')) \\ \Gamma \vdash \mathit{resp} : \prod_{x \, y:X} \prod_{zs':(\sum_{zs:\mathbf{Bag}} P \, zs)} \mathit{cons} \, x \, (y :: (\pi_1 \, zs'), \mathit{cons} \, y \, zs') = \mathit{cons} \, y \, (x :: (\pi_1 \, zs'), \mathit{cons} \, x \, zs') \end{array}}{\mathbf{Bagelim} \, \mathit{nil} \, \mathit{cons} \, \mathit{resp} : \prod_{x:\mathbf{Bag}} P \, x}$$

And it computes:

$$\begin{array}{l} \mathbf{Bagelim} \, \mathit{nil} \, \mathit{cons} \, \mathit{resp} [] \quad =_P [] \quad \mathit{nil} \\ \mathbf{Bagelim} \, \mathit{nil} \, \mathit{cons} \, \mathit{resp} (x :: xs) \quad =_P (x::xs) \quad \mathit{cons} \, x \, (\mathbf{Bagelim} \, \mathit{nil} \, \mathit{cons} \, \mathit{resp} \, xs) \end{array}$$

7.2. From QIT to QWI-type. We claim that any QIT Q in the sense of Definition 7.1 can be constructed as the QW-type for a signature and equational system derived from the declaration of the QIT; and that the same is true for the indexed version of Definition 7.1 and QWI-types. That is, QWI-types are universal for non-conditional QITs in the same sense that W-types are for inductive types in a sufficiently extensional type theory.

We saw above that the data $c_1 : C_1, \dots, c_m : C_m$ in Definition 7.1 gives rise to an underlying inductive type $[Q]$ and hence to a W-type [AGHMM15], with signature $\Sigma = (A, B)$. The parameters and arguments of the equality constructors $d_1 : D_1, \dots, d_n : D_n$ are also encoded in the same way by a signature (E, V) . Then the endpoints of the equality constructors can be encoded by the l and r arguments of an equational system $\varepsilon = (E, V, l, r)$ in the sense of Definition 3.1; the encoding follows the structure of \mathbf{EqType} judgements in Figure 3, using the η constructor of T_Σ (3.6) for variables and the σ constructor for c_1, \dots, c_m introduced by $\mathbf{EQCONSTR}$ in Θ . We illustrate the encoding by example, beginning with the three examples from the Introduction.

Example 7.5 (Finite multisets). The element constructors of the QIT, $\mathbf{Bag} \, X$, of finite multisets over $X : \mathcal{U}$ in (1.1) are encoded exactly as the W-type for \mathbf{List} over X : we take $A : \mathcal{U}$ to be $\mathbb{1} + X$, where $\iota_1 0$ corresponds to $[]$ and $\iota_2 x$ corresponds to $x :: _$ for each $x : X$. The arity of $[]$ is zero, and the arity of each $x :: _$ is one; so we take $B : A \rightarrow \mathcal{U}$ to be the function mapping $\iota_1 0$ to 0 and each $\iota_2 x$ to $\mathbb{1}$. The swap equality constructor is parametrised by elements of $E \stackrel{\text{def}}{=} X \times X$ and for each $(x, y) : E$, $\mathit{swap} \, (x, y)$ yields an equation involving a single free variable (called $zs : \mathbf{Bag} \, X$ in (1.1)); so we define $V \stackrel{\text{def}}{=} \lambda _ . \mathbb{1} : E \rightarrow \mathcal{U}$. Each side of the equation named by $\mathit{swap} \, (x, y)$ is coded by an element of $T_\Sigma (V \, (x, y)) = T_\Sigma \mathbb{1}$. Recalling the definition of T_Σ from Section 3.1, the single free variable, zs , corresponds to $\eta 0 : T_\Sigma \mathbb{1}$. Then the left-hand side of the equation, $x :: y :: zs$, is encoded as $\sigma(\iota_2 x, (\lambda _ . \sigma(\iota_2 y, (\lambda _ . \eta 0))))$, and similarly the right-hand side, $y :: x :: zs$, is encoded as $\sigma(\iota_2 y, (\lambda _ . \sigma(\iota_2 x, (\lambda _ . \eta 0))))$.

So altogether, the encoding of $\mathbf{Bag} \, X$ as a QW-type uses the non-indexed signature $\Sigma = (A, B)$ and equational system $\varepsilon = (E, V, l, r)$, where:

$$\begin{array}{ll} A \stackrel{\text{def}}{=} \mathbb{1} + X & E \stackrel{\text{def}}{=} X \times X \\ B(\iota_1 0) \stackrel{\text{def}}{=} 0 & V(x, y) \stackrel{\text{def}}{=} \mathbb{1} \\ B(\iota_2 x) \stackrel{\text{def}}{=} \mathbb{1} & l(x, y) \stackrel{\text{def}}{=} \sigma(\iota_2 x, (\lambda _ . \sigma(\iota_2 y, (\lambda _ . \eta 0)))) \\ & r(x, y) \stackrel{\text{def}}{=} \sigma(\iota_2 y, (\lambda _ . \sigma(\iota_2 x, (\lambda _ . \eta 0)))) \end{array}$$

Example 7.6 (Length-indexed multisets). The QWI-type encoding the QIT $\mathbf{AbVec} \, X$ of length-indexed multisets in (1.3) is an indexed version of the previous example, using the

index type $I \stackrel{\text{def}}{=} \mathbb{N}$. The indexed signature $\Sigma = (\mathbb{N}, A, B)$ has $A : \mathcal{U}^{\mathbb{N}}$ and $B : \prod_{i:\mathbb{N}} (A_i \rightarrow \mathcal{U}^{\mathbb{N}})$ given by:

$$\begin{aligned} A_0 &\stackrel{\text{def}}{=} \mathbb{1} \\ A_{i+1} &\stackrel{\text{def}}{=} X \\ B_0 0 j &\stackrel{\text{def}}{=} 0 \\ B_{i+1} x j &\stackrel{\text{def}}{=} (i = j) \end{aligned}$$

The indexed system of equations $\varepsilon = (\mathbb{N}, E, V, l, r)$ over Σ has $E : \mathcal{U}^{\mathbb{N}}$, $V : \prod_{i:\mathbb{N}} (E_i \rightarrow \mathcal{U}^{\mathbb{N}})$ and $l, r : \prod_{i:\mathbb{N}} \prod_{e:E_i} (T_{\Sigma}(V_i e))_i$ given by:

$$\begin{aligned} E_0 &\stackrel{\text{def}}{=} 0 \\ E_1 &\stackrel{\text{def}}{=} 0 \\ E_{i+2} &\stackrel{\text{def}}{=} X \times X \\ V_{i+2}(x, y) j &\stackrel{\text{def}}{=} (i = j) \\ l_{i+2}(x, y) &\stackrel{\text{def}}{=} \sigma_{i+2}(x, \lambda _ . \lambda \text{ refl} . \sigma_{i+1}(y, \lambda _ . \lambda \text{ refl} . \eta_i \text{ refl})) \\ r_{i+2}(x, y) &\stackrel{\text{def}}{=} \sigma_{i+2}(y, \lambda _ . \lambda \text{ refl} . \sigma_{i+1}(x, \lambda _ . \lambda \text{ refl} . \eta_i \text{ refl})) \end{aligned}$$

(We have used dependent pattern matching [Coq92] to simplify the formulation of the above definitions; $\text{refl} : x = x$ denotes the proof of reflexivity; V_0, V_1, l_0 and l_1 are uniquely determined.)

Example 7.7 (Unordered countably-branching trees). The parameters of the leaf and node constructors for the QIT in (1.4) look like those for **Bag**, except that the arity of **node** is \mathbb{N} rather than $\mathbb{1}$. So we use the non-indexed signature $\Sigma = (A, B)$ with $A : \mathcal{U}$ and $B : A \rightarrow \mathcal{U}$ given by:

$$\begin{aligned} A &\stackrel{\text{def}}{=} \mathbb{1} + X \\ B(\iota_1 0) &\stackrel{\text{def}}{=} 0 \\ B(\iota_2 x) &\stackrel{\text{def}}{=} \mathbb{N} \end{aligned}$$

The **perm** equality constructor is parameterised by elements of $X \times \sum_{b:\mathbb{N} \rightarrow \mathbb{N}} \text{islsob}$. For each element (x, b, b') of that type, $\text{perm}(x, b, b')$ yields an equation involving an \mathbb{N} -indexed family of variables (called $f : \mathbb{N} \rightarrow \omega\text{Tree } X$ in (1.4)); so we take $V : E \rightarrow \mathcal{U}$ to be $\lambda _ . \mathbb{N}$. Each side of the equation named by $\text{perm}(x, b, b')$ is coded by an element of $T_{\Sigma}(V(x, b, b')) = T_{\Sigma}(\mathbb{N})$. The \mathbb{N} -indexed family of variables is represented by the function $\eta : \mathbb{N} \rightarrow T_{\Sigma}(\mathbb{N})$ and its permuted version by $\eta \circ b$. Thus the left- and right-hand sides of the equation named by $\text{perm}(x, b, b')$ are coded respectively by the elements $\sigma(\iota_2 x, \eta)$ and $\sigma(\iota_2 x, \eta \circ b)$. Finally, the non-indexed system of equations over Σ for the QW-type corresponding to the QIT in (1.4) is $\varepsilon = (E, V, l, r)$ with:

$$\begin{aligned} E &\stackrel{\text{def}}{=} X \times \sum_{b:\mathbb{N} \rightarrow \mathbb{N}} \text{islsob} \\ V(x, b, b') &\stackrel{\text{def}}{=} \mathbb{N} \\ l(x, b, b') &\stackrel{\text{def}}{=} \sigma(\iota_2 x, \eta) \\ r(x, b, b') &\stackrel{\text{def}}{=} \sigma(\iota_2 x, \eta \circ b) \end{aligned}$$

This example is significant since, prior to the introduction of QW-types [FPS20], none of the constructions in the existing literature on subclasses of QITs [Soj15; Swa18; DM18] or QIITs [Dij17; KKA19] supported infinitary QITs.

Example 7.8 (W-suspensions [Soj15]). W-suspensions are also instances of QW-types. The data for one of them is: a type $A' : \mathcal{U}$ and a type family $B' : A' \rightarrow \mathcal{U}$, (that is, the same data as a W-type), together with a type $C' : \mathcal{U}$ and functions $l', r' : C' \rightarrow A'$ that express a restricted subset of equalities. The equivalent QW-type is:

$$\begin{array}{lll} A \stackrel{\text{def}}{=} A' & E \stackrel{\text{def}}{=} C' & l \stackrel{\text{def}}{=} \lambda c . \sigma(l' c, \eta) \\ B \stackrel{\text{def}}{=} B' & V \stackrel{\text{def}}{=} \lambda c . B'(l' c) \times B'(r' c) & r \stackrel{\text{def}}{=} \lambda c . \sigma(r' c, \eta) \end{array}$$

Example 7.9 (W-types with reductions [Swa18]). W-types with reductions are further instances of QWI-types. The data for such a type, using our notational conventions, is: $Z : \mathcal{U}, Y : \mathcal{U}^Z, X : Y \rightarrow \mathcal{U}^Z$, together with, for all $z : Z$ a reindexing map $R_z : \prod_{y:Y_z} (X_z y)_z$. A reduction identifies a term $\sigma_z(y, \alpha)$ for the signature (Z, Y, X) at index z with the argument that was used to construct that term at the re-indexing map, namely $\alpha_z(R_z y)$. We claim that the equivalent QWI-type is given by:

$$\begin{array}{llll} I \stackrel{\text{def}}{=} Z & A \stackrel{\text{def}}{=} Y & E \stackrel{\text{def}}{=} Y & l \stackrel{\text{def}}{=} \lambda z . \lambda y . \sigma_z(y, \eta) \\ & B \stackrel{\text{def}}{=} X & V \stackrel{\text{def}}{=} X & r \stackrel{\text{def}}{=} \lambda z . \lambda y . \eta_z(R_z y) \end{array}$$

Example 7.10 (Blass, Lumsdaine and Shulman's HIT [LS19]). As discussed in Section 4, Blass [Bla83, Section 9] shows that, provided a certain large cardinal axiom is consistent with ZFC, then there is an infinitary equational theory with no initial algebra in ZF. Lumsdaine and Shulman [LS19, Section 9] adapt this theory into a higher-inductive type, called F , that cannot be proved to exist in ZF, and hence cannot be constructed in type theory just using pushouts and natural numbers. Nevertheless it can be constructed in the type theory of Section 2 plus WISC, because of Theorem 6.1 and the fact that F can be expressed as a QWI-type.

The type F can be thought of as a set of notations for countable ordinals. Its definition requires some setup.

Fix the bijection $[\text{even}, \text{odd}] : \mathbb{N} + \mathbb{N} \rightarrow \mathbb{N}$ with inverse par . Fix another bijection $\text{pair} : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$ with inverse $\langle \text{fst}, \text{snd} \rangle$. Given $f, g : \mathbb{N} \rightarrow A$, write $f \cup g : \mathbb{N} \rightarrow A$ for the composite $[f, g] \circ \text{par}$; given $a : A$, write $\{a\} : \mathbb{N} \rightarrow A$ for the constant function at a .

Now F is defined by the three point constructors,

$$\begin{array}{l} 0 : F \\ S : F \rightarrow F \\ \text{sup} : (\mathbb{N} \rightarrow F) \rightarrow F \end{array}$$

and the five path constructors,

$$\begin{aligned}
\mathbf{sup}\{0\} &= 0 \\
\prod_{f,g:\mathbb{N}\rightarrow\mathbb{N}} \mathbf{E}fg &\rightarrow \prod_{h:\mathbb{N}\rightarrow F} \mathbf{sup}(h \circ f) = \mathbf{sup}(h \circ g) \\
\prod_{f,g:\mathbb{N}\rightarrow F} \mathbf{sup}(f \cup \{\mathbf{sup}(f \cup g)\}) &= \mathbf{sup}(f \cup g) \\
\prod_{f,g:\mathbb{N}\rightarrow F} \mathbf{sup}(f \cup \{S(\mathbf{sup}(f \cup g))\}) &= S(\mathbf{sup}(f \cup g)) \\
\prod_{b,c:\mathbb{N}\rightarrow\mathbb{N}} \prod_{L:\mathbb{N}\rightarrow\mathbb{N}\rightarrow\mathbb{N}} \mathbf{JS}bc &\rightarrow \mathbf{Lrel}Lbc \rightarrow \mathbf{Lrel}Lcb \rightarrow \prod_{h:\mathbb{N}\rightarrow F} \mathbf{sup}(\mathbf{iter}hb) = \mathbf{sup}(\mathbf{iter}hc)
\end{aligned}$$

where

$$\begin{aligned}
\mathbf{E}fg &\stackrel{\text{def}}{=} \prod_{n:\mathbb{N}} \left(\sum_{m:\mathbb{N}} f(m) = n \right) \leftrightarrow \left(\sum_{m:\mathbb{N}} g(m) = n \right) \\
\mathbf{JS}bc &\stackrel{\text{def}}{=} \prod_{n:\mathbb{N}} \left(\sum_{m:\mathbb{N}} b(m) = n \right) \vee \left(\sum_{m:\mathbb{N}} c(m) = n \right) \\
\mathbf{Lrel}Lxy &\stackrel{\text{def}}{=} \prod_{n:\mathbb{N}} \sum_{m,l:\mathbb{N}} (L(xm)l = yn) \\
\mathbf{iter}hp &\stackrel{\text{def}}{=} \lambda n . h_{(\mathbf{fst}n)}(p(\mathbf{snd}n)) \\
h_0 &\stackrel{\text{def}}{=} h \\
h_{(x+1)} &\stackrel{\text{def}}{=} \lambda y . \mathbf{sup}(h_x \circ (Ly))
\end{aligned}$$

This can be expressed as a QW-type as follows:

$$\begin{aligned}
A &\stackrel{\text{def}}{=} \mathfrak{3} \\
B &\stackrel{\text{def}}{=} [0, \mathbf{1}, \mathbb{N}] \\
E &\stackrel{\text{def}}{=} \mathbf{1} + \left(\sum_{f,g:\mathbb{N}\rightarrow\mathbb{N}} \mathbf{E}fg \right) + \mathbf{1} + \mathbf{1} + \left(\sum_{b,c:\mathbb{N}\rightarrow\mathbb{N}} \sum_{L:\mathbb{N}\rightarrow\mathbb{N}\rightarrow\mathbb{N}} (\mathbf{JS}bc \times \mathbf{Lrel}Lbc \times \mathbf{Lrel}Lcb) \right) \\
V &\stackrel{\text{def}}{=} [0, \mathbb{N}, \mathbb{N} + \mathbb{N}, \mathbb{N} + \mathbb{N}, \mathbb{N}] \\
l(\iota_1 _) &\stackrel{\text{def}}{=} \sigma(2, \lambda _ . \sigma(0, !)) \\
r(\iota_1 _) &\stackrel{\text{def}}{=} \sigma(0, !) \\
l(\iota_2 fg) &\stackrel{\text{def}}{=} \sigma(2, (\eta \circ f)) \\
r(\iota_2 fg) &\stackrel{\text{def}}{=} \sigma(2, (\eta \circ g)) \\
l(\iota_3 _) &\stackrel{\text{def}}{=} \sigma(2, ((\eta \circ \iota_1) \cup \lambda _ . \sigma(1, ((\eta \circ \iota_1) \cup (\eta \circ \iota_2)))))) \\
r(\iota_3 _) &\stackrel{\text{def}}{=} \sigma(2, ((\eta \circ \iota_1) \cup (\eta \circ \iota_2))) \\
l(\iota_4 _) &\stackrel{\text{def}}{=} \sigma(2, ((\eta \circ \iota_1) \cup \lambda _ . \sigma(1, \sigma(2, ((\eta \circ \iota_1) \cup (\eta \circ \iota_2)))))) \\
r(\iota_4 _) &\stackrel{\text{def}}{=} \sigma(1, \sigma(2, ((\eta \circ \iota_1) \cup (\eta \circ \iota_2)))) \\
l(\iota_5 bcL _ _ _) &\stackrel{\text{def}}{=} \sigma(2, \lambda n . kL(\mathbf{fst}n)(b(\mathbf{snd}n))) \\
r(\iota_5 bcL _ _ _) &\stackrel{\text{def}}{=} \sigma(2, \lambda n . kL(\mathbf{fst}n)(c(\mathbf{snd}n)))
\end{aligned}$$

where

$$\begin{aligned}
kL0 &\stackrel{\text{def}}{=} \eta \\
kL(x+1) &\stackrel{\text{def}}{=} \lambda y . \sigma(2, ((kLx) \circ (Ly)))
\end{aligned}$$

8. CONCLUSION

QWI-types are a general form of indexed quotient inductive types that capture many examples, including simple 1-cell complexes and non-recursive QITs [BGvdW17], non-structural QITs [Soj15], W-types with reductions [Swa18] and also infinitary QITs (e.g. unordered infinitely branching trees [AK16] and the type F of Blass [Bla83], Lumsdaine and Shulman [LS19]). We have shown that it is possible to construct any QWI-type, even infinitary ones, in the extensional type theory of toposes with natural numbers object and universes satisfying the WISC Axiom. (Indeed our Agda development shows that the construction is also possible in weaker, more intensional type theories.)

We conclude by mentioning related work and some possible directions for future work.

Reduction of QWI to QW. WI-types (indexed W-types) can be constructed from W-types: see [GH04, Theorem 12] and [AGHMM15, Section 7.1]. Our main Theorem 6.1 implies indirectly that in the presence of WISC, QWI-types can be constructed from QW-types (since quotients and W-types are instances of the latter). We do not know whether there is a direct reduction of QWI-types to QW-types without any further axioms, extending the reduction of WI-types to W-types.

Conditional path equations. In Section 7.1 we mentioned the fact that Dybjer and Moeneclaey [DM18] give a schema for finitary 1-HITs (and 2-HITs) in which path constructors are allowed to take arguments involving the identity type of the datatype being declared. To cover this case, one could consider generalising QWI-types by replacing the role played by infinitary equational theories by theories whose axioms are infinitary Horn clauses, that is, equations conditioned by (possibly infinite) conjunctions of equations. In fact, we have already extended Theorem 6.1 to cover this case, and this will likely appear in the third author's upcoming thesis.

Quotients of monads. Theorem 6.1 gives a construction of initial algebras for equational systems on the *free* monad T_Σ generated by a signature Σ . By a suitable change of signature (see Remark 3.3) this extends to a construction of free algebras, rather than just initial ones. One can show that the construction works not just for free monads, but for more general ones where the underlying endofunctor is well-behaved (satisfies some suitable colimit preservation), such as the quotient monads one obtains from systems of equations over signatures. Given such a construction one gets a quotient monad morphism from the base monad to the quotient monad. This contravariantly induces a forgetful functor from the algebras of the latter to those of the former. Using the adjoint triangle theorem, one should be able to construct a left adjoint. This would then cover examples such as the free group over a monoid, free ring over a group, etc.

Quotient inductive-inductive types. We do not know whether our analysis of QITs using quotients, inductive types and a well-founded notion of size can be extended to cover the notion of *quotient inductive-inductive* type (QIIT) [ACDKN18; KKA19]. Dijkstra [Dij17] studies such types in depth and in Chapter 6 of his thesis gives a construction for finitary ones in terms of countable colimits, and hence in terms of countable coproducts and quotients. One could hope to pass to the infinitary case by using well-founded sizes as we have done, provided an analogue for QIITs can be found of the construction in Section 6 for our class of QITs, the

QWI-types. Kaposi, Kovács, and Altenkirch [KKA19] and Kovács and Kaposi [KK20] give a specification of QIITs using a domain-specific type theory called the *theory of signatures* (ToS) and, assuming the ToS exists, prove existence of all QIITs matching this specification. It might be possible to encode their ToS using QWI-types (it is already an example of a QIIT), or to extend QWI-types to a notion of “inductive-inductive QW-type” that makes this possible. Together this would provide a construction of all QIITs.

Homotopy Type Theory (HoTT). In this paper we have used an extensional type theory. Our Agda development is more intensional, but nevertheless makes use of the Uniqueness of Identity Proofs (UIP) axiom, which is well-known to be incompatible with the Univalence Axiom [Uni13, Example 3.1.9]. Given the interest in HoTT, it is certainly worth investigating whether an analogue of Theorem 6.1 holds in some model of univalent foundations.

Pattern matching for QITs and HITs. Our reduction of QITs to quotients and inductive types in the presence of WISC is of foundational interest. For applications, one could wish for direct support in systems like Agda, Coq and Lean for the very useful notion of quotient inductive type, or more generally, for higher inductive types. Even having better support for the special case of quotient types would be welcome. It is not hard to envisage the addition of a general schema for declaring QITs; but when it comes to defining functions on them, having to do that with eliminator forms rapidly becomes cumbersome (for example, for functions of several QIT arguments). Some extension of dependently-typed pattern matching to cover equality constructors as well as element constructors is needed. In this context it is worth mentioning that the `cubical` features of recent versions of Agda give access to cubical type theory [VMA19]. This allows for easy declaration of HITs (and hence in particular QITs) and a certain amount of pattern matching when it comes to defining functions on them: the value of a function on a path constructor can be specified by using generic elements of the interval type in point-level patterns; but currently the user is given little mechanised assistance to solve the definitional equality constraints on end-points of paths that are generated by this method.

ACKNOWLEDGEMENT

We would like to acknowledge the contribution Ian Orton made to the initial development of the work described here; he and the first author supervised the third author’s Master’s thesis in which the notion of a QW-type was first introduced. The second author would also like to thank Paul Blain Levy for discussions about the various forms of the *weakly initial sets of covers* axiom (see Remark 4.2); and Andrew Swan for discussions about the role of that axiom in constructive proofs of cocontinuity for polynomial endofunctors (see Corollary 5.10).

REFERENCES

- [AAG05] Michael Abbott, Thorsten Altenkirch, and Neil Ghani. “Containers: Constructing Strictly Positive Types”. In: *Theoretical Computer Science vol. 342.1* (2005), pp. 3–27.
- [Abe12] Andreas Abel. “Type-Based Termination, Inflationary Fixed-Points, and Mixed Inductive-Coinductive Types”. In: *Electronic Proceedings in Theoretical Computer Science vol. 77* (2012), pp. 1–11. DOI: 10.4204/EPTCS.77.1.

- [ACDKN18] Thorsten Altenkirch, Paolo Capriotti, Gabe Dijkstra, Nicolai Kraus, and Fredrik Nordvall Forsberg. “Quotient Inductive-Inductive Types”. In: *Foundations of Software Science and Computation Structures, FoSSaCS 2018*. Ed. by Christel Baier and Ugo Dal Lago. Vol. 10803. Lecture Notes in Computer Science. Springer International Publishing, 2018, pp. 293–310.
- [Agd20] Agda v2.6.1. <https://agda.readthedocs.io/en/v2.6.1.1/index.html>. 2020.
- [AGHMM15] Thorsten Altenkirch, Neil Ghani, Peter Hancock, Conor McBride, and Peter Morris. “Indexed Containers”. In: *Journal of Functional Programming vol. 25* (2015), e5. DOI: 10.1017/S095679681500009X.
- [AK16] Thorsten Altenkirch and Ambrus Kaposi. “Type Theory in Type Theory Using Quotient Inductive Types”. In: *Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages - POPL 2016*. The 43rd Annual ACM SIGPLAN-SIGACT Symposium. St. Petersburg, FL, USA: ACM Press, 2016, pp. 18–29. DOI: 10.1145/2837614.2837638.
- [BGvdW17] Henning Basold, Herman Geuvers, and Niels van der Weide. “Higher Inductive Types in Programming”. In: *Journal of Universal Computer Science vol. 23.1* (2017), p. 27. DOI: 10.3217/jucs-023-01-0063.
- [BL70] Garrett Birkhoff and John D. Lipson. “Heterogeneous algebras”. In: *Journal of Combinatorial Theory vol. 8.1* (1970), pp. 115–133. DOI: 10.1016/S0021-9800(70)80014-X.
- [Bla83] Andreas Blass. “Words, Free Algebras, and Coequalizers”. In: *Fundamenta Mathematicae vol. 117.2* (1983), pp. 117–160.
- [Coq92] Thierry Coquand. “Pattern Matching with Dependent Types”. In: *Proceedings of the 1992 Workshop on Types for Proofs and Programs, Båstad, Sweden*. Ed. by B. Nordström, K. Petersson, and G. D. Plotkin. June 1992, pp. 66–79.
- [Dij17] Gabe Dijkstra. “Quotient Inductive-Inductive Definitions”. PhD thesis. United Kingdom: University of Nottingham, 14th Dec. 2017. URL: <http://eprints.nottingham.ac.uk/42317/1/thesis.pdf> (visited on 14/10/2019).
- [DM18] Peter Dybjer and Hugo Moeneclaey. “Finitary Higher Inductive Types in the Groupoid Model”. In: *Electronic Notes in Theoretical Computer Science vol. 336* (Apr. 2018), pp. 119–134. DOI: 10.1016/j.entcs.2018.03.019.
- [Dyb97] Peter Dybjer. “Representing Inductively Defined Sets by Wellorderings in Martin-Löf’s Type Theory”. In: *Theoretical Computer Science vol. 176.1-2* (1997), pp. 329–335. DOI: 10.1016/S0304-3975(96)00145-4.
- [FH09] Marcelo Fiore and Chung-Kil Hur. “On the Construction of Free Algebras for Equational Systems”. In: *Theoretical Computer Science vol. 410.18* (Apr. 2009), pp. 1704–1729.
- [Fio13] Marcelo P. Fiore. “An Equational Metalogic for Monadic Equational Systems”. In: *Theory and Applications of Categories vol. 27.18* (2013), pp. 464–492. URL: <http://emis.ams.org/journals/TAC/volumes/27/18/27-18.pdf> (visited on 10/09/2019).
- [FPS20] Marcelo P. Fiore, Andrew M. Pitts, and S. C. Steenkamp. “Constructing Infinitary Quotient-Inductive Types”. In: *23rd International Conference on Foundations of Software Science and Computation Structures (FoSSaCS 2020)*. Ed. by J. Goubault-Larrecq and B. König. Vol. 12077. Lecture Notes in

- Computer Science. Springer, 2020, pp. 257–276. DOI: 10.1007/978-3-030-45231-5_14.
- [FPS21] Marcelo P. Fiore, Andrew M. Pitts, and S. C. Steenkamp. *Agda Code Accompanying “Quotients, Inductive Types, and Quotient Inductive Types”*. 2021. DOI: 10.17863/CAM.82943.
- [GH04] Nicola Gambino and Martin Hyland. “Wellfounded Trees and Dependent Polynomial Functors”. In: *Types for Proofs and Programs*. Ed. by S. Berardi, M. Coppo, and F. Damiani. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 210–225.
- [Git80] Mordechy Gitik. “All Uncountable Cardinals Can Be Singular”. In: *Israel J. Math. vol. 35.1–2* (1980), pp. 61–88.
- [Hof95] Martin Hofmann. “Extensional Concepts in Intensional Type Theory”. PhD thesis. University of Edinburgh, 1995.
- [Joh02] Peter T. Johnstone. *Sketches of an Elephant, A Topos Theory Compendium, Volumes 1 and 2*. Oxford Logic Guides 43–44. Oxford University Press, 2002.
- [Kar14] Asaf Karagila. “Embedding Orders into Cardinals with DC_κ ”. In: *Fundamenta Mathematicae vol. 226* (2014), pp. 143–156.
- [Kel80] Gregory Maxwell Kelly. “A Unified Treatment of Transfinite Constructions for Free Algebras, Free Monoids, Colimits, Associated Sheaves, and so on”. In: *Bulletin of the Australian Mathematical Society vol. 22.1* (1980), pp. 1–83. DOI: 10.1017/S0004972700006353.
- [Kel92] Gregory Maxwell Kelly. “On Clubs and Data-Type Constructors”. In: *Applications of Categories in Computer Science, Proceedings LMS Symposium, Durham, UK, 1991*. Ed. by M. P. Fourman, P. T. Johnstone, and A. M. Pitts. Vol. 177. LMS Lecture Note Series. Cambridge University Press, 1992, pp. 95–106.
- [KK18] Ambrus Kaposi and András Kovács. “A Syntax for Higher Inductive-Inductive Types”. In: *FSCD* (2018). In collab. with Michael Wagner, 18 pages. DOI: 10.4230/LIPICS.FSCD.2018.20.
- [KK20] András Kovács and Ambrus Kaposi. “Large and Infinitary Quotient Inductive-Inductive Types”. In: *LICS ’20*. Saarbrücken, Germany: Association for Computing Machinery, 2020, pp. 648–661. DOI: 10.1145/3373718.3394770.
- [KKA19] Ambrus Kaposi, András Kovács, and Thorsten Altenkirch. “Constructing Quotient Inductive-Inductive Types”. In: *Proceedings of the ACM on Programming Languages vol. 3* (POPL 2nd Jan. 2019), 2:1–2:24. DOI: 10.1145/3290315.
- [Lev21] Paul Blain Levy. *Broad Infinity and Generation Principles*. 2021. arXiv: 2101.01698 [math.LO].
- [Lin66] Fred E. J. Linton. “Some Aspects of Equational Categories”. In: *Proceedings of the Conference on Categorical Algebra*. Ed. by S. Eilenberg, D. K. Harrison, S. MacLane, and H. Röhrli. Berlin, Heidelberg: Springer Berlin Heidelberg, 1966, pp. 84–94.
- [LS19] Peter LeFanu Lumsdaine and Michael Shulman. “Semantics of Higher Inductive Types”. In: *Mathematical Proceedings of the Cambridge Philosophical Society* (17th June 2019), pp. 1–50. DOI: 10.1017/S030500411900015X.
- [Mar82] Per Martin-Löf. “Constructive Mathematics and Computer Programming”. In: *Studies in Logic and the Foundations of Mathematics*. Ed. by Laurence

- Jonathan Cohen, Jerzy Łoś, Helmut Pfeiffer, and Klaus-Peter Podewski. *vol. 104*. Elsevier, 1982, pp. 153–175. DOI: 10.1016/S0049-237X(09)70189-2.
- [McB99] Conor McBride. “Dependently Typed Functional Programs and Their Proofs”. PhD thesis. Edinburgh, Scotland, United Kingdom: University of Edinburgh, 1999. 263 pp. URL: <https://era.ed.ac.uk/handle/1842/374> (visited on 09/07/2020).
- [MM16] Philippe Malbos and Samuel Mimram. “Homological Computations for Term Rewriting Systems”. In: *1st International Conference on Formal Structures for Computation and Deduction (FSCD 2016)*. Vol. 52. Porto, Portugal, June 2016, 27:1–27:17. URL: <https://hal.archives-ouvertes.fr/hal-01678175>.
- [MP00] Ieke Moerdijk and Erik Palmgren. “Wellfounded Trees in Categories”. In: *Annals of Pure and Applied Logic vol. 104.1* (2000), pp. 189–218. DOI: 10.1016/S0168-0072(00)00012-9.
- [MP02] Ieke Moerdijk and Erik Palmgren. “Type theories, Toposes and Constructive Set Theory: Predicative Aspects of AST”. In: *Annals of Pure and Applied Logic vol. 114.1* (2002), pp. 155–201.
- [Rob15] David Michael Roberts. “The Weak Choice Principle WISC may Fail in the Category of Sets”. In: *Studia Logica vol. 103* (2015), pp. 1005–1017.
- [Shu18] Michael Shulman. “Brouwer’s Fixed-Point Theorem in Real-Cohesive Homotopy Type Theory”. In: *Mathematical Structures in Computer Science vol. 28.6* (June 2018), pp. 856–941. DOI: 10.1017/S0960129517000147.
- [Soj15] Kristina Sojakova. “Higher Inductive Types as Homotopy-Initial Algebras”. In: *Principles of Programming Languages*. Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium. Mumbai, India: ACM Press, 2015, pp. 31–42. DOI: 10.1145/2676726.2676983.
- [Str05a] Thomas Streicher. “Realizability Models for CZF + \neg Pow”. <http://www2.mathematik.tu-darmstadt.de/~streicher/CIZF/rmczfnp.pdf>. 2005.
- [Str05b] Thomas Streicher. “Universes in Toposes”. In: *From Sets and Types to Topology and Analysis, Towards Practicable Foundations for Constructive Mathematics*. Ed. by L. Crosilla and P. Schuster. *vol. 48*. Oxford Logic Guides. Oxford University Press, 2005. Chap. 4, pp. 78–90. DOI: 10.1093/acprof:oso/9780198566519.001.0001.
- [Swa18] Andrew Swan. “W-Types with Reductions and the Small Object Argument”. In: (21st Feb. 2018). arXiv: 1802.07588 [math].
- [Tay96] Paul Taylor. “Intuitionistic Sets and Ordinals”. In: *Journal of Symbolic Logic vol. 61* (1996), pp. 705–744.
- [Uni13] The Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations for Mathematics*. Institute for Advanced Study: <http://homotopytypetheory.org/book>, 2013.
- [vdBM14] Benno van den Berg and Ieke Moerdijk. “The Axiom of Multiple Choice and Models for Constructive Set Theory”. In: *Journal of Mathematical Logic vol. 14.01* (June 2014), p. 1450005. DOI: 10.1142/S0219061314500056.
- [VMA19] Andrea Vezzosi, Anders Mörtberg, and Andreas Abel. “Cubical Agda: A Dependently Typed Programming Language with Univalence and Higher Inductive Types”. In: *Proc. ACM Program. Lang. vol. 3*. ICFP (July 2019), 87:1–87:29. DOI: 10.1145/3341691.