Type-Theoretic Constructions of the Final Coalgebra of the Finite Powerset Functor

Niccolò Veltri
Department of Software Science, Tallinn University of Technology, Estonia

Abstract

The finite powerset functor is a construct frequently employed for the specification of nondeterministic transition systems as coalgebras. The final coalgebra of the finite powerset functor, whose elements characterize the dynamical behavior of transition systems, is a well-understood object which enjoys many equivalent presentations in set-theoretic foundations based on classical logic.

In this paper, we discuss various constructions of the final coalgebra of the finite powerset functor in constructive type theory, and we formalize our results in the Cubical Agda proof assistant. Using setoids, the final coalgebra of the finite powerset functor can be defined from the final coalgebra of the list functor. Using types instead of setoids, as it is common in homotopy type theory, one can specify the finite powerset datatype as a higher inductive type and define its final coalgebra as a coinductive type. Another construction is obtained by quotienting the final coalgebra of the list functor, but the proof of finality requires the assumption of the axiom of choice. We conclude the paper with an analysis of a classical construction by James Worrell, and show that its adaptation to our constructive setting requires the presence of classical axioms such as countable choice and the lesser limited principle of omniscience.

2012 ACM Subject Classification Theory of computation → Type theory; Theory of computation → Constructive mathematics

Keywords and phrases type theory, finite powerset, final coalgebra, Cubical Agda

Digital Object Identifier 10.4230/LIPIcs.FSCD.2021.22

Funding This work was supported by the Estonian Research Council grant PSG659 and by the ESF funded Estonian IT Academy research measure (project 2014-2020.4.05.19-0001).

Acknowledgements We thank Henning Basold, Tarmo Uustalu, Andrea Vezzosi and Niels van der Weide for valuable discussions.

1 Introduction

The powerset functor, delivering the set of subsets of a given set, plays a fundamental role in the behavioral analysis of nondeterministic systems [26], which include process calculi such as Milner's calculus of communicating systems [23] and π-calculus [24]. A nondeterministic system is determined by a function $c : S \to \mathsf{P}\ S$, called a coalgebra, from a set of states S to the set $\mathsf{P}\ S$ of subsets of S. The function c associates to each state $x : S$ a set of new states $c\ x$ reachable from x, so it represents the transition relation of an unlabelled transition system. Adding labels to transitions is easy, just consider coalgebras of the form $c : S \to \mathsf{P}\ (A \times S)$ or $c : S \to (A \to \mathsf{P}\ S)$ instead, where A is a set of labels. In many applications, the set of reachable states is known to be finite, so the powerset functor P can be replaced by the finite powerset functor Pfin delivering only the set of finite subsets.

The behavior of a finitely nondeterministic system starting from a given initial state is fully captured by the final coalgebra of Pfin. Elements of the final coalgebra are execution traces obtained by iteratively running the coalgebra function modelling the system on the initial state. The resulting traces are possibly infinite trees with finite unordered branching. Several formal constructions of the final coalgebra of Pfin and other finitary set functors exist in the literature, developed using various different techniques [6, 2, 32, 33, 3]. Adámek et al.


© Niccolò Veltri;
licensed under Creative Commons License CC-BY 4.0

collect and compare all these characterizations in their recent book draft [4, Chapter 4]. All these constructions take place in set theory, and reasoning is based on classical logic.

In this work we present various definitions of the final coalgebra of the finite powerset functor in constructive type theory, which have all been formalized in the Cubical Agda proof assistant [30]. Cubical Agda is an implementation of cubical type theory [10], which in turn is a particular presentation of homotopy type theory with support for univalence and higher inductive types (HITs). The choice of Cubical Agda as our foundational setting, over other proof assistants based on Martin-Löf type theory or the calculus of constructions such as plain Agda, Coq or Lean, lays in the fact that both univalence and HITs play an important role for both encoding and reasoning with the finite powerset datatype in homotopy type theory [17]. In our development we also take advantage of Cubical Agda's support for coinductive types [30].

First, we study the construction of the finite powerset as a setoid [7], i.e. a pair of a carrier type and an equivalence relation on the carrier. This is inspired by Danielsson's setoid of finite multisubsets [13]. The final coalgebra of the finite powerset in this setting arises as a setoid with the final coalgebra of the List functor as carrier, whose elements are non-wellfounded trees with finite ordered branching. The equivalence relation on the latter type relates trees that differ only in the order and multiplicity of their subtrees.

Working with setoids, therefore associating a specific equality relation to each type and ensuring that all constructions respect this relation, is not in the spirit of homotopy type theory, where the spotlight is on the notion of propositional equality, also called path equality in this setting. We then consider Frumin et al.'s presentation of the finite powerset datatype as a HIT, Pfin A, formally delivering the free join semilattice on A [17]. It is well-known that coinductive types can be employed for the construction of M-types, i.e. final coalgebras of polynomial functors. We show that coinductive types can be used in a similar way for defining the final Pfin-coalgebra. This construction works since in Cubical Agda HITs are implemented as usual inductive types, in which higher path constructors depend on additional interval names and satisfy two matching conditions on endpoints [11, 9]. In other words, HITs are part of the grammar of strictly positive types and as such they are allowed to appear in the domain type of destructors of coinductive types.

An alternative construction of the final coalgebra of the finite powerset functor (as a type) is obtainable by performing a quotient operation on the final setoid coalgebra, i.e. quotienting the final List-coalgebra by the equivalence relation relating trees containing the same subtrees, possibly in different order and with different multiplicity. This construction is possible in homotopy type theory due to the existence of a set quotient operation definable as a HIT [27]. We show that the resulting quotient type is indeed a fixpoint of Pfin, but the proof of its finality requires the assumption of the full axiom of choice.

The last part of the paper is devoted to the analysis of a classical set-theoretic construction of the final Pfin-coalgebra by James Worrell [33]. It is well known that the chain of iterated applications of Pfin on the singleton set does not stabilize after ω steps [2]. This is in antithesis with the case of polynomial functors, whose final coalgebras (a.k.a. M-types in type theory) always arise as ω-limits, a fact that can also be proved in homotopy type theory [5]. Worrell showed that the final Pfin-coalgebra can be obtained by iterating applications of Pfin for extra ω steps, i.e. as the $(\omega + \omega)$-limit of the chain. Elements of the ω-limit are represented by non-wellfounded trees with unordered but possibly infinite branching, while the $(\omega + \omega)$-limit corresponds to the subset of these trees with finite branching at all levels. We study Worrell's construction in our constructive setting and show that the $(\omega + \omega)$-limit is indeed the final Pfin-coalgebra modulo the assumption of classical principles such as axiom of countable

choice and the lesser limited principle of omniscience (LLPO). Notably, Worrell's iterated construction is inherently classical: the injectivity of the canonical Pfin-algebra on the ω-limit is equivalent to LLPO. In particular, it is impossible to prove that the $(\omega + \omega)$-limit is a subset of the ω-limit, as in Worrell's construction, without the assumption of LLPO.

All the material presented in the paper have been formalized in the Cubical Agda proof assistant. The code is freely available at `https://github.com/niccoloveltri/final-pfin`.

2 Type Theory and Cubical Agda

Our work takes place in homotopy type theory (HoTT) [27]. Practically, we formalize our constructions in Cubical Agda [30]. This is an implementation of cubical type theory [10], a particular flavor of HoTT with support for univalence, function extensionality and higher inductive types. What follows is a brief description of basic notions employed in our work. More details on programming in Cubical Agda can be found in Vezzosi et al.'s paper [30].

A few words on notation. We write Type for the universe of small types. We use Agda notation for dependent function types $(x : A) \to B\ x$, where B is a type family of type $A \to \mathsf{Type}$. Implicit arguments of functions are enclosed in curly brackets. We write $=_{\mathrm{df}}$ for definitional equality and we denote judgemental equality by $\equiv$. We reserve the use of the equality symbol $=$ for path equality. Given an element of a dependent sum type $\sum x : A.\ B\ x$, we denote its two projections by fst and snd. The unit type is 1 with unique inhabitant tt, the empty type is $\bot$. The type of Boolean values is Bool with elements true and false, and the binary sum of types A and B is $A + B$. The type of natural numbers is $\mathbb{N}$ with constructors zero and suc, the type of lists with entries in A is $\mathsf{List}\ A$ with constructors $[\,]$ and $(::)$. The unique function from a type A into the unit type is called $! : A \to 1$.

2.1 Univalence, Path Types, Higher Inductive Types

In cubical type theory, and therefore Cubical Agda, *univalence* is a theorem stating that equality of types corresponds to equivalence. A function $f : A \to B$ is an *equivalence* if it has contractible fibers, i.e. if the preimage of any element in B under f is a singleton type. Any function underlying a type isomorphism defines an equivalence. Writing $A \simeq B$ for the type of equivalences between A and B, univalence states that the canonical function of type $A = B \to A \simeq B$ is an equivalence. In particular, there is a function $\mathsf{ua} : A \simeq B \to A = B$ which turns equivalences into equalities. From any proof of equality built as $\mathsf{ua}\ e$ we need to be able to extract the equivalence e, so the representation of equality needs to accommodate such information. Cubical type theory takes inspiration from the cubical interpretation of HoTT [10] and represents equalities as paths, i.e. functions out of an interval object.

In Cubical Agda there is a primitive interval type $\mathbb{I}$ required to be a De Morgan algebra with two endpoints i_0 and i_1. This is used in the implementation of the primitive type $\mathsf{Path}\ A\ \ x\ y$ of path equalities between elements $x : A$ and $y : A$, which we always denote by $x = y$. A path type is similar to a function type with domain $\mathbb{I}$: an element $p : x = y$ is eliminated by application to an interval element $r : \mathbb{I}$, returning $p\ r : A$. Unlike a function type, this application can compute even when p is unknown by using the endpoints x and y stored in the type: $p\ \mathsf{i}_0$ reduces to x, while $p\ \mathsf{i}_1$ reduces to y. The introduction of a path is done via lambda abstraction $\lambda\ i : \mathbb{I}.\ t : x = y$, but this causes the extra requirement to match the endpoints: $t[\mathsf{i}_0/i] \equiv x$ and $t[\mathsf{i}_1/i] \equiv y$.

The identification of equalities with special functions from an interval type allows the provability of the *function extensionality* principle, stating that pointwise equal functions are

equal. The proof consists of simply swapping the order of the two input arguments:

$$\mathsf{funExt} : \{f\ g : A \to B\} \to ((x : A) \to f\ x = g\ x) \to f = g$$
$$\mathsf{funExt}\ p\ i\ x =_{\mathrm{df}} p\ x\ i$$

A characteristic feature of homotopy type theory, together with Voevodsky's univalence, is the presence of higher inductive types (HITs) [27]. A HIT is a type whose constructors inductively generate both its elements and its (higher) paths. We introduce three HITs: propositional truncation, set quotient and finite powerset (the latter in Section 4.1).

First, we introduce three classes of types: the *contractible types*, which have a unique inhabitant up to path equality, the *(mere) propositions*, for which any two elements are path equal, and the *sets*, whose path types are propositions. The collections of propositions is called $\mathsf{hProp} =_{\mathrm{df}} \sum A : \mathsf{Type}.\ \mathsf{isProp}\ A$. We follow the informal convention of identifying a proposition with its underlying type (i.e. its first projection).

$$\mathsf{isContr}\ A =_{\mathrm{df}} \sum x : A.\ (y : A) \to x = y$$
$$\mathsf{isProp}\ A =_{\mathrm{df}} (x\ y : A) \to x = y$$
$$\mathsf{isSet}\ A =_{\mathrm{df}} (x\ y : A) \to \mathsf{isProp}\ (x = y)$$

The *propositional truncation* of a type A is the HIT generated by the following constructors:

$$\frac{x : A}{|x| : \|A\|} \qquad \frac{x, y : \|A\|}{\mathsf{squash}\ x\ y : x = y}$$

$\|A\|$ is the proposition associated to type A, in which all elements of A have been unified thanks to the path constructor squash. Using propositional truncation, we can define an uninformative *existential quantifier* $\exists x : A.\ B\ x =_{\mathrm{df}} \| \sum x : A.\ B\ x\|$.

The *set quotient* of a type A by a (proof-relevant) relation $R : A \to A \to \mathsf{Type}$ is the HIT generated by the following constructors:

$$\frac{x : A}{[x] : A/R} \qquad \frac{x, y : A \quad r : R\ x\ y}{\mathsf{eq}/\ r : x = y} \qquad \frac{x, y : A/R \quad p, q : x = y}{\mathsf{squash}/\ p\ q : p = q}$$

The element $[x]$ is the R-equivalence class of x, while the path constructor eq/ states that R-related elements have path equal equivalence classes. The 2-path constructor squash/ forces A/R to be a set.

HITs are supported in cubical type theory [11] and have been implemented in Cubical Agda, where they can be introduced using the syntax of inductive types. Path constructors are considered as point constructors depending on extra interval names and satisfying the required matching conditions on endpoints. Functions out of HITs can be defined via pattern matching, where now the user has to deal with the extra cases of higher path constructors. For example, propositional truncation is a functor, and its action on functions is defined as

$$\mathsf{map}_{\|} : (A \to B) \to \|A\| \to \|B\|$$
$$\mathsf{map}_{\|}\ f\ |x| =_{\mathrm{df}} |f\ x|$$
$$\mathsf{map}_{\|}\ f\ (\mathsf{squash}\ x\ y\ i) =_{\mathrm{df}} \mathsf{squash}\ (\mathsf{map}_{\|}\ f\ x)(\mathsf{map}_{\|}\ f\ y)\ i$$

2.2 Coinductive Types

Agda has native support for coinductive types specified by strictly positive functors, and this support has been extended to Cubical Agda as well. As an example, which will be employed in the successive sections, consider the type Tree consisting of finitely-branching non-wellfounded trees defined as the final coalgebra of the List functor. In Agda, the latter is

```
record Tree : Type where
  coinductive
  field
    subtrees_L : List Tree
```

```
record TreeB (t u : Tree) : Type where
  coinductive
  field
    subtreesB_L : List̄ TreeB (subtrees_L t) (subtrees_L u)
```

Figure 1 Agda definitions of infinite trees and tree bisimilarity.

encoded as a coinductive record with one destructor $\mathsf{subtrees_L}$, returning the subtrees of the root, see the left code in Figure 1. The type Tree, together with the destructor $\mathsf{subtrees_L}$, is a coalgebra of the List functor. Elements of coinductive types are characterized by the result of the application of destructors, which means that an element of type Tree is specified by the list of its subtrees. This is dual to the construction of elements of inductive types in terms of constructors. For example, the infinite binary tree is corecursively defined as: $\mathsf{subtrees_L}\ \mathsf{binTree} =_{\mathrm{df}} \mathsf{binTree} :: \mathsf{binTree} :: [\,]$.

An important advantage of working in Cubical Agda is the possibility to prove the *coinduction principle* [30]. For the type of trees, this states that tree bisimilarity is equivalent to path equality. Bisimilarity can be defined as a coinductive relation on trees, and as such it can be encoded in Agda as a coinductive record, see the right code in Figure 1. In the codomain of the destructor $\mathsf{subtreesB_L}$ we employ the *lifting* of a type family $R : A \to B \to \mathsf{Type}$ to lists, inductively generated by two constructors:

$$\frac{}{[\,] : \overline{\mathsf{List}}\ R\ [\,]\ [\,]} \qquad \frac{p : R\ a\ b \quad r : \overline{\mathsf{List}}\ R\ l\ m}{p :: r : \overline{\mathsf{List}}\ R\ (a :: l)\ (b :: m)} \tag{1}$$

The proof of the coinduction principle $\mathsf{bisim_L}$ fundamentally employs copatterns [1] and lambda abstraction of interval variables, i.e. the introduction rule of path types. The coinduction principle $\mathsf{bisim_L}$ is simultaneously constructed with an auxiliary proof $\mathsf{bisim'_L}$, stating that ($\overline{\mathsf{List}}\ \mathsf{TreeB}$)-related lists of trees are path equal.

$$\begin{aligned}&\mathsf{bisim_L} : \{t\,u : \mathsf{Tree}\} \to \mathsf{TreeB}\ t\ u \to t = u\\ &\mathsf{subtrees_L}\ (\mathsf{bisim_L}\ b\ i) =_{\mathrm{df}} \mathsf{bisim'_L}\ (\mathsf{subtreesB_L}\ b)\ i\end{aligned}$$

$$\begin{aligned}&\mathsf{bisim'_L} : \{l\,m : \mathsf{List\ Tree}\} \to \overline{\mathsf{List}}\ \mathsf{TreeB}\ l\ m \to l = m\\ &\mathsf{bisim'_L}\ [\,] \qquad i =_{\mathrm{df}} [\,]\\ &\mathsf{bisim'_L}\ (b :: r)\ i =_{\mathrm{df}} \mathsf{bisim_L}\ b\ i :: \mathsf{bisim'_L}\ r\ i\end{aligned}$$

The productivity, i.e. well-definiteness, of the function $\mathsf{bisim_L}$ is guaranteed by the presence of list constructors $[\,]$ and $::$ at top level in the definition of $\mathsf{bisim'_L}$. More generally, corecursively defined terms are accepted as valid by Agda's productivity checker only when recursive calls appear directly under the application of a constructor. This syntactic restriction, while indeed sufficient for ensuring the productivity of corecursive definitions, makes programming and reasoning with coinductive types in Agda quite cumbersome. For example, the following construction of the unique coalgebra morphism from the carrier of a coalgebra $c : X \to \mathsf{List}\ X$ to Tree is not accepted in Agda ($\mathsf{map_{List}}$ is the action on functions of the List functor):

$$\begin{aligned}&\mathsf{anaTree} : (c : X \to \mathsf{List}\ X) \to X \to \mathsf{Tree}\\ &\mathsf{subtrees_L}\ (\mathsf{anaTree}\ c\ x) =_{\mathrm{df}} \mathsf{map_{List}}\ (\mathsf{anaTree}\ c)\ (c\ x)\end{aligned} \tag{2}$$

For this reason, in our code we parameterize our coinductive types with *sizes*, to ease the productivity checking of corecursive definitions [18, 14]. For example, the function $\mathsf{anaTree}$ is accepted by Agda if the type of trees is decorated with size information. Notice that we use sized types for mere practical convenience: we believe possible, with some extra work, to

massage the corecursive definitions in our Agda code and obtain equivalent characterizations able to overtake Agda's syntactic productivity checker. In the paper, all mentions to sizes have been removed.

Since HITs are implemented in Agda as a particular kind of inductive types, Cubical Agda also allows the construction of coinductive types specified by functors defined as HITs. This is a somehow experimental feature: the existence of such objects would need to be verified in the cubical set model, which we leave to future work. We will show an example of such a coinductive type in Section 4.2.

3 The Finite Powerset and Its Final Coalgebra as a Setoid

Here we introduce the finite powerset construction as a setoid and study its final coalgebra.

A *setoid* [7], or *Bishop set*, is a pair (A, R) consisting of a type A and a proof-irrelevant (i.e. valued in propositions) equivalence relation R on A. We write Setoid for the type of setoids and, given $S : \mathsf{Setoid}$, we write $\mathsf{carr}\ S$ and $\mathsf{eqr}\ S$ for the carrier and the equivalence relation of S, respectively. A *setoid morphism* between setoids (A, R) and (B, S) is a function $f : A \to B$ which is compatible with the equivalence relations: for all $x, y : A$, if $R\ x\ y$ then $S\ (f\ x)\ (f\ y)$. We write $\mathsf{SetoidMor}\ S\ T$ for the type of setoid morphisms between setoids S and T, and, given $h : \mathsf{SetoidMor}\ S\ T$, we write $\mathsf{fun}\ h : \mathsf{carr}\ S \to \mathsf{carr}\ T$ for its underlying function. Setoids and their morphisms form a category Setoid, but this is not the framework typically employed as a foundational setting for constructive mathematics, since in this category equality of morphisms is given by path equality, not equivalence relation. Bishop-style constructive mathematics is instead developed in SetoidRel, which is the category Setoid enriched in the category of sets and equivalence relations [19]. In this setting, two setoid morphisms f and g between setoids (A, R) and (B, S) are considered equal whenever, for all $x : A$, $S\ (f\ x)\ (g\ x)$.

In SetoidRel, given an endofunctor F with action on setoid morphisms $\mathsf{map_F}$ (satisfying the functor laws up to the appropriate equivalence relation), the types of F-coalgebras and F-coalgebra morphisms between two F-coalgebras (S, s) and (T, t) are defined as follows:

$$\begin{aligned}&\mathsf{Coalg_s}\ F =_{\mathrm{df}} \textstyle\sum S : \mathsf{Setoid}.\ \mathsf{SetoidMor}\ S\ (F\ S)\\ &\mathsf{CoalgMor_s}\ F\ (S, s)\ (T, t) =_{\mathrm{df}}\\ &\qquad \textstyle\sum h : \mathsf{SetoidMor}\ S\ T.\ (x : \mathsf{carr}\ S) \to \mathsf{eqr}\ T\ (\mathsf{fun}\ t\ (\mathsf{fun}\ h\ x))\ (\mathsf{fun}\ (\mathsf{map_F}\ h)\ (\mathsf{fun}\ s\ x))\end{aligned} \tag{3}$$

A coalgebra in SetoidRel is final if there exists a unique coalgebra morphism from any other coalgebra, up to equivalence relation:

$$\mathsf{Final_s}\ F =_{\mathrm{df}} \textstyle\sum C : \mathsf{Coalg_s}\ F.\ (D : \mathsf{Coalg_s}\ F) \to \mathsf{isContr_s}\ (\mathsf{CoalgMor_s}\ F\ C\ D) \tag{4}$$

where elements of $\mathsf{isContr_s}\ (\mathsf{CoalgMor_s}\ F\ C\ D)$ are pairs consisting of a coalgebra morphism h and, for any other coalgebra morphism h', a proof that h and h' are equivalent as setoid morphisms.

3.1 The Setoid of Finite Subsets

Given a setoid (A, R), its setoid of finite subsets is defined as $\mathsf{Pfin_s}\ (A, R) =_{\mathrm{df}} (\mathsf{List}\ A, \widehat{\mathsf{List}}\ R)$, where $\widehat{\mathsf{List}}$ is a lifting of List to relations, alternative to the lifting given in (1). $\widehat{\mathsf{List}}$ is sometimes called a *relator* and plays an important role in the study of applicative bisimilarity for functional programming languages with nondeterministic choice [21]. Given a type family

```
record TreeR (t u : Tree) : Type where
  coinductive
  field
    subtreesR : List^ TreeR (subtrees_L t) (subtrees_L u)
```

Figure 2 Agda definition of the coinductive closure of the relator $\widehat{\mathsf{List}}$.

$R : A \to B \to \mathsf{Type}$, the type family $\widehat{\mathsf{List}}\ R : \mathsf{List}\ A \to \mathsf{List}\ B \to \mathsf{Type}$ is defined as

$$\begin{aligned} \widehat{\mathsf{List}}\ R\ l\ m =_{\mathrm{df}} \quad & ((x : A) \to x \in_{\mathsf{L}} l \to \exists y : B.\ y \in_{\mathsf{L}} m \times R\ x\ y) \\ & \times \\ & ((y : B) \to y \in_{\mathsf{L}} m \to \exists x : A.\ x \in_{\mathsf{L}} l \times R\ y\ x) \end{aligned} \tag{5}$$

So two lists are related by $\widehat{\mathsf{List}}\ R$ when each element of a list is R-related to an element of the other list. The type family $\in_{\mathsf{L}}$ is the inductive (proof-relevant) membership relation on lists, the subscript L distinguishes this to the membership relation on the type Pfin introduced in Section 4.1. $\mathsf{Pfin}_{\mathsf{s}}$ is an endofunctor on SetoidRel. Its action on setoid morphisms $\mathsf{map}_{\mathsf{Pfin}_{\mathsf{s}}} : \mathsf{SetoidMor}\ S\ T \to \mathsf{SetoidMor}\ (\mathsf{Pfin}_{\mathsf{s}}\ S)\ (\mathsf{Pfin}_{\mathsf{s}}\ T)$ has underlying function $\mathsf{map}_{\mathsf{List}}$.

Notice the presence of existential quantifications $\exists$ in the definition of $\widehat{\mathsf{List}}$. If we were to replace them with $\sum$, we would obtain a setoid of finite multisubsets instead, as the one considered by Danielsson [13].

3.2 The Final Coalgebra

The final coalgebra of the final powerset functor in SetoidRel can be constructed using coinductive types. Consider the coinductive relation of Figure 2 obtained by replacing the lifting $\overline{\mathsf{List}}$ with the lifting $\widehat{\mathsf{List}}$ in the destructor of the tree bisimilarity relation TreeB in Figure 1. Two trees are related by TreeR if, for each subtree of one tree, there merely exists a TreeR-related subtree of the other tree. The setoid $\nu\mathsf{Pfin}_{\mathsf{s}} =_{\mathrm{df}} (\mathsf{Tree}, \mathsf{TreeR})$ is a $\mathsf{Pfin}_{\mathsf{s}}$-coalgebra:

```
coalg_s : SetoidMor νPfin_s (Pfin_s νPfin_s)
coalg_s = (subtrees_L , subtreesR)
```

▶ **Theorem 1.** *The* $\mathsf{Pfin}_{\mathsf{s}}$*-coalgebra* $(\nu\mathsf{Pfin}_{\mathsf{s}}, \mathsf{coalg}_{\mathsf{s}})$ *is final in* SetoidRel.

Proof. We only show the existence of a coalgebra morphism into $(\nu\mathsf{Pfin}_{\mathsf{s}}, \mathsf{coalg}_{\mathsf{s}})$. Given another $\mathsf{Pfin}_{\mathsf{s}}$-coalgebra (S, s), there is a setoid morphism h from S to $(\mathsf{Tree}, \mathsf{TreeR})$ with underlying function $\mathsf{anaTree}\ (\mathsf{fun}\ s)$.

This function is compatible with equivalence relations. Assume given $x, y : \mathsf{carr}\ S$ such that $\mathsf{eqr}\ S\ x\ y$. We prove $\mathsf{TreeR}\ (\mathsf{anaTree}\ (\mathsf{fun}\ s)\ x)\ (\mathsf{anaTree}\ (\mathsf{fun}\ s)\ y)$. This is a coinductive type, so we proceed by applying the destructor of TreeR and we are left to show that $\mathsf{subtrees}_{\mathsf{L}}\ (\mathsf{anaTree}\ (\mathsf{fun}\ s)\ x)$ is $(\widehat{\mathsf{List}}\ \mathsf{TreeR})$-related to $\mathsf{subtrees}_{\mathsf{L}}\ (\mathsf{anaTree}\ (\mathsf{fun}\ s)\ y)$. The definition of the lifting $\widehat{\mathsf{List}}$ in (5) is symmetric, so it is sufficient to prove the following lemma (in which we unfold the definition of $\mathsf{anaTree}$ as in (2)):

$$\begin{aligned} (x' : \mathsf{Tree}) \to x' \in_{\mathsf{L}} \mathsf{map}_{\mathsf{List}}\ (\mathsf{anaTree}\ (\mathsf{fun}\ s))\ (\mathsf{fun}\ s\ x) \\ \to \exists y' : \mathsf{Tree}.\ y' \in_{\mathsf{L}} \mathsf{map}_{\mathsf{List}}\ (\mathsf{anaTree}\ (\mathsf{fun}\ s))\ (\mathsf{fun}\ s\ y) \times \mathsf{TreeR}\ x'\ y' \end{aligned} \tag{6}$$

Given a tree x' as in the hypotheses of (6), it is possible to construct another tree x'' such that $x'' \in_{\mathsf{L}} \mathsf{fun}\ s\ x$ and $\mathsf{anaTree}\ (\mathsf{fun}\ s)\ x'' = x'$. Remember that s is a setoid morphism and

$\mathsf{eqr}\ S\ x\ y$ holds by assumption. This implies that, for each element in the list $\mathsf{fun}\ s\ x$, there exists a ($\mathsf{eqr}\ S$)-related element in the list $\mathsf{fun}\ s\ y$. Since $x'' \in_{\mathsf{L}} \mathsf{fun}\ s\ x$, we obtain a tree y'' such that $y'' \in \mathsf{fun}\ s\ y$ and $\mathsf{eqr}\ S\ x''\ y''$. The required tree y' in the goal of (6) is defined as $\mathsf{anaTree}\ (\mathsf{fun}\ s)\ y''$. The proof of y' being TreeR-related to x' is given by the corecursive hypothesis applied to arguments x'',y'' and the proof of $\mathsf{eqr}\ S\ x''\ y''$. ◀

Differently from the case of polynomial functors, the final $\mathsf{Pfin_s}$-coalgebra cannot be constructed as an ω-limit in SetoidRel. In fact, the ω-limit of the sequence obtained by iterated application of $\mathsf{Pfin_s}$ on the unit setoid is not a fixpoint of $\mathsf{Pfin_s}$. This is in analogy with the situation in classical set theory described by Adámek and Koubek [2], for which we give a constructive account in Section 5.

It is possible to prove a version of Theorem 1 for Setoid instead of SetoidRel: $(\nu\mathsf{Pfin_s}, \mathsf{coalg_s})$ is also the final $\mathsf{Pfin_s}$-coalgebra in Setoid, where one first needs to appropriately adapt the definitions of coalgebra morphism and final coalgebra in (3) and (4) to Setoid.

4 The Finite Powerset and Its Final Coalgebra as a Type

We now abandon the setoid setting and work with types as primary object instead of setoids, as typically done in HoTT. Given an endofunctor $F : \mathsf{Type} \to \mathsf{Type}$ with action on functions $\mathsf{map_F}$, the types of F-coalgebras and F-coalgebra morphisms between two F-coalgebras (A, a) and (B, b) are defined as follows:

$$\begin{aligned} &\mathsf{Coalg}\ F =_{\mathrm{df}} \textstyle\sum A : \mathsf{Type}.\ A \to F\ A \\ &\mathsf{CoalgMor}\ F\ (A, a)\ (B, b) =_{\mathrm{df}} \textstyle\sum f : A \to B.\ (x : A) \to b\ (f\ x) = \mathsf{map_F}\ f\ (a\ x) \end{aligned} \tag{7}$$

In this setting, a coalgebra is final if there exists a unique (up to path equality) coalgebra morphism to any other coalgebra.

$$\mathsf{Final}\ F =_{\mathrm{df}} \textstyle\sum C : \mathsf{Coalg}\ F.\ (D : \mathsf{Coalg}\ F) \to \mathsf{isContr}\ (\mathsf{CoalgMor}\ F\ C\ D) \tag{8}$$

The definitions in (7) and (8) are the same of Ahrens et al. [5], which they only consider in the case of F being a polynomial functor specified by a signature. The coinductive type Tree of Section 2.2 is the final List-coalgebra, with the function anaTree of (2) as unique mediating coalgebra morphism.

4.1 The Type of Finite Subsets

The action of the finite powerset functor on a type A returns the set of all finite subsets of A. Following Frumin et al. [17], the finite powerset functor can be encoded as a higher inductive type in two equivalent ways: as a set quotient of lists or as the term algebra of the theory of join semilattices.

As a Set Quotient The set of finite subsets can be defined as a set quotient of the type of lists: $\mathsf{Pfin_q}\ A =_{\mathrm{df}} \mathsf{List}\ A/\mathsf{SameEls}$. The subscript q indicates that this type is a set quotient. The relation SameEls, as the name suggests, relates lists containing the same elements, and it is given by the relator $\widehat{\mathsf{List}}$ applied to path equality on A, i.e. $\mathsf{SameEls} =_{\mathrm{df}} \widehat{\mathsf{List}}\ (=)$.

As the Free Join Semilattice The set of finite subsets can also be defined as the *free join semilattice* on a given type A. A join semilattice is a partially ordered set $(X, \leq)$ with a bottom element and a binary join operation. Join semilattices admit an equational presentation as algebraic theories, from which the following higher inductive type can be extrapolated:

$$\frac{}{\varnothing : \mathsf{Pfin}\ A} \qquad \frac{a : A}{\eta\ a : \mathsf{Pfin}\ A} \qquad \frac{x, y : \mathsf{Pfin}\ A}{x \cup y : \mathsf{Pfin}\ A}$$

$$\frac{x : \mathsf{Pfin}\ A}{\mathsf{nr}\ x : x \cup \varnothing = x} \qquad \frac{x, y, z : \mathsf{Pfin}\ A}{\mathsf{assoc}\ x\ y\ z : (x \cup y) \cup z = x \cup (y \cup z)} \qquad \frac{x, y : \mathsf{Pfin}\ A}{\mathsf{comm}\ x\ y : x \cup y = y \cup x}$$

$$\frac{x : \mathsf{Pfin}\ A}{\mathsf{idem}\ x : x \cup x = x} \qquad \frac{x, y : \mathsf{Pfin}\ A \quad p, q : x = y}{\mathsf{squashPfin}\ p\ q : p = q}$$

The type $\mathsf{Pfin}\ A$ is a join semilattice, with empty subset $\varnothing$ as bottom element and binary union $\cup$ as join operation. The partial order can be recovered in the usual way: $x \leq y =_{\mathrm{df}} (x \cup y) = y$. The 1-path constructors mimic the equational theory of join semilattices, while the 2-path constructor $\mathsf{squashPfin}$ forces $\mathsf{Pfin}\ A$ to be a set. The constructor η embeds A into $\mathsf{Pfin}\ A$ and represents the singleton subset operation. The elimination principle of $\mathsf{Pfin}\ A$ corresponds to the universal property of $\mathsf{Pfin}\ A$ as the free join semilattice on A.

The membership relation $\in$ is defined by induction on the finite subset in input.

$$\begin{aligned}
&\in : A \to \mathsf{Pfin}\ A \to \mathsf{hProp} \\
&a \in \varnothing \quad =_{\mathrm{df}} \bot \\
&a \in \eta\ b \quad =_{\mathrm{df}} \|a = b\| \\
&a \in x \cup y =_{\mathrm{df}} \|a \in x + a \in y\|
\end{aligned}$$

The omitted cases for the higher constructors are dealt with using univalence. Moreover the right-hand-sides only contain the types underlying the propositions, the proof terms showing that these satisfy the predicate isProp have been omitted. The subset relation is given by $x \subseteq y =_{\mathrm{df}} (a : A) \to a \in x \to a \in y$, which is equivalent to the order relation $\leq$ defined above.

Given a type family $R : A \to B \to \mathsf{Type}$, its *lifting* to Pfin is the type family $\overline{\mathsf{Pfin}}\ R : \mathsf{Pfin}\ A \to \mathsf{Pfin}\ B \to \mathsf{Type}$ defined as

$$\begin{aligned}
\overline{\mathsf{Pfin}}\ R\ s\ t =_{\mathrm{df}} \quad & ((x : A) \to x \in s \to \exists y : B.\ y \in t \times R\ x\ y) \\
& \times \\
& ((y : B) \to y \in t \to \exists x : A.\ x \in s \times R\ y\ x)
\end{aligned}$$

For all relations R, it is possible to show that $\overline{\mathsf{Pfin}}\ R$ is a congruence, which means that we are able to construct elements of the following types:

$$\overline{\mathsf{Pfin}}\ R\ \varnothing\ \varnothing \qquad \overline{\mathsf{Pfin}}\ R\ s_1\ t_1 \to \overline{\mathsf{Pfin}}\ R\ s_2\ t_2 \to \overline{\mathsf{Pfin}}\ R\ (s_1 \cup s_2)\ (t_1 \cup t_2) \tag{9}$$

When R is path equality, $\overline{\mathsf{Pfin}}\ (=)$ corresponds to extensional equality of finite subsets, i.e. $\overline{\mathsf{Pfin}}\ (=)\ s\ t \simeq (s \subseteq t \times t \subseteq s)$. Since the subset relation is antisymmetric, we have that $\overline{\mathsf{Pfin}}\ (=)\ s\ t \simeq (s = t)$. We call $\mathsf{toPfinEq} : \overline{\mathsf{Pfin}}\ (=)\ s\ t \to (s = t)$ the left-to-right function underlying this equivalence.

The two types $\mathsf{Pfin}\ A$ and $\mathsf{Pfin}_{\mathsf{q}}\ A$ are provably equivalent [17].

4.2 The Final Coalgebra

As a Coinductive Type As mentioned in the last paragraph of Section 2.2, Cubical Agda allows the construction of coinductive types specified by functors using HITs in their definition.

```
record νPfin : Type where
  coinductive
  field
    subtreesP : Pfin νPfin
```

```
record νPfinB (t u : νPfin) : Type where
  coinductive
  field
    subtreesBP : Pfin‾ νPfinB (subtreesP t) (subtreesP u)
```

Figure 3 Agda definitions of coinductive final coalgebra of Pfin and its bisimilarity relation

When such functor is the finite powerset functor Pfin, this construction is given by the record type on the left in Figure 3. Notice that νPfin is a set: Pfin A is a set for all types A, so in particular Pfin νPfin is a set, and the latter is isomorphic to νPfin.

In Figure 3, the coinductive relation on the right is the notion of bisimilarity associated to the infinite trees in νPfin. The coinduction principle for νPfin is derivable by copattern matching and path introduction as follows:

```
bisimP : {t u : νPfin} → νPfinB t u → t = u
subtreesP (bisimP b i) =df
  toPfinEq (λx m. map‖(λ(x′, m′, b′). (x′, m′, bisimP b′)) (fst (subtreesBP b) x m),
            λx m. map‖(λ(x′, m′, b′). (x′, m′, bisimP b′)) (snd (subtreesBP b) x m))
           i
```

▶ **Theorem 2.** *νPfin is the final coalgebra of* Pfin *in the sense of (8).*

Proof. The construction of the mediating coalgebra morphism between a Pfin-coalgebra (A, a) and νPfin, whose coalgebra is the destructor subtrees$_P$, is analogous to the one in (2):

```
anaPfin : (c : X → Pfin X) → X → νPfin
subtreesP (anaPfin c x) =df mapPfin (anaPfin c) (c x)
```
(10)

Now assume given another coalgebra morphism $f : X \to \nu\mathsf{Pfin}$. We prove simultaneously the two following lemmata, and conclude uniqueness using the coinduction principle of νPfin.

```
anaPfinUniq : (x : X) → νPfinB (f x) (anaPfin c x)
anaPfinUniqR : (s : Pfin X) → Pfin‾ νPfinB (mapPfin f s) (mapPfin (anaPfin c) s)
```

The first lemma is proved by corecursion, so after an application of the destructor of νPfinB (and after unfolding the definition of anaPfin in (10)), we are left to construct an element of type Pfin‾ νPfinB (subtrees$_P$ (f x)) (map$_{Pfin}$ (anaPfin c) (c x)). Since f is a coalgebra morphism, we can substitute subtrees$_P$ (f x) for map$_{Pfin}$ f (c x) in the latter, and we return anaPfinUniqR (c x) as the inhabitant of the type resulting from the substitution.

The second lemma is proved by induction on s. Since the return type is a proposition, we only need to deal with the three cases of the point constructors.

- Case $s \equiv \varnothing$. We are done by the left result in (9).
- Case $s \equiv \eta\ z$. Our goal reduces to Pfin‾ νPfinB (η (f z)) (η (anaPfin c z)). We construct the first argument of this product type, the second argument is defined analogously. Assume given $x : \nu\mathsf{Pfin}$ and $p : x \in \eta\ (f\ z)$, i.e. a truncated equality proof $p : \|x = f\ z\|$. We need to show that there merely exists $y : \nu\mathsf{Pfin}$ such that $y \in \eta$ (anaPfin c z), i.e. $\|y =$ anaPfin c $z\|$, and νPfinB x y holds. Take $y =_{\mathrm{df}}$ anaPfin c z, and derive νPfinB x y by first rewriting x to f z using p (we can remove the propositional truncation in p since the return type is a proposition as well) and subsequently applying anaPfinUniq to z.
- Case $s \equiv s_1 \cup s_2$. We apply the right result in (9) and we conclude by invoking inductive hypotheses anaPfinUniqR s_1 and anaPfinUniqR s_2.

◀

As a Set Quotient Alternatively, we could quotient the type Tree of finitely ordered branching trees by the equivalence relation TreeR introduced in Figure 2. The resulting type is a fixpoint of $\mathsf{Pfin_q}$.

▶ **Theorem 3.** *The type* $\mathsf{Tree/TreeR}$ *is equivalent to* $\mathsf{Pfin_q}\ (\mathsf{Tree/TreeR})$

Proof. We only discuss the construction of the functions underlying the equivalence. A function $f : \mathsf{Tree/TreeR} \to \mathsf{Pfin_q}\ (\mathsf{Tree/TreeR})$ is defined by pattern matching (we only show the case of the point constructor): $f\ [t] =_{\mathrm{df}} [\mathsf{map_{List}}\ (\lambda x.\ [x])\ (\mathsf{subtrees_L}\ t)]$.

A function $g : \mathsf{Pfin_q}\ (\mathsf{Tree/TreeR}) \to \mathsf{Tree/TreeR}$ is also defined by pattern matching. Notice that this is equivalent to construct a function $g' : \mathsf{List}\ (\mathsf{Tree/TreeR}) \to \mathsf{Tree/TreeR}$ which is compatible with the relation $\mathsf{SameEls}$. Since the type $\mathsf{List}\ (\mathsf{Tree/TreeR})$ is equivalent to $\mathsf{List\ Tree}/\overline{\mathsf{List}}\ \mathsf{TreeR}$, it is sufficient to define a function $g'' : \mathsf{List\ Tree}/\overline{\mathsf{List}}\ \mathsf{TreeR} \to \mathsf{Tree/TreeR}$ satisfying an adjusted compatibility condition. This is given by pattern matching (again, we omit the cases of the path constructors): $g''\ [l] =_{\mathrm{df}} [\mathsf{subtrees_L^{-1}}\ l]$, where $\mathsf{subtrees_L^{-1}}$ is the inverse of the destructor $\mathsf{subtrees_L}$. ◀

Proving finality of the coalgebra underlying the equivalence of Theorem 3 seems to require the assumption of the full *axiom of choice*. This is constructively problematic, since in HoTT the axiom of choice implies the law of excluded middle [27]. We employ an alternative formulation of the axiom of choice, provably equivalent to the usual one. First, consider two types A, B and a type family $R : B \to B \to \mathsf{Type}$. Let $\overline{\mathsf{Fun}}\ R$ be the lifting of the type family R to the function space $A \to B$, i.e. given $f, g : A \to B$, define $\overline{\mathsf{Fun}}\ R\ f\ g =_{\mathrm{df}} (x : A) \to R\ (f\ x)\ (g\ x)$. It is possible to define a function $\theta_R : (A \to B)/\overline{\mathsf{Fun}}\ R \to A \to B/R$ by pattern matching on the first argument. The existence of a section for θ_R, for all type families R, is an equivalent phrasing of the axiom of choice (see e.g. [28] for a proof of this equivalence):

$$\begin{aligned}\mathsf{AC} =_{\mathrm{df}} &\{A, B : \mathsf{Type}\}\ (R : B \to B \to \mathsf{Type}) \\ &\to \exists \psi_R : (A \to B/R) \to (A \to B)/\overline{\mathsf{Fun}}\ R.\ (x : (A \to B)/\overline{\mathsf{Fun}}\ R) \to \theta_R\ (\psi_R\ x) = x\end{aligned} \tag{11}$$

▶ **Theorem 4.** *Assuming axiom of choice, the type* $\mathsf{Tree/TreeR}$ *is the final coalgebra of* $\mathsf{Pfin_q}$.

Proof. We only discuss the construction of the mediating coalgebra morphism. We are asked to construct a function $\mathsf{anaPfin_q} : (c : X \to \mathsf{Pfin_q}\ X) \to X \to \mathsf{Tree/TreeR}$. This can be obtained from the function $\mathsf{anaPfin'_q} : (c : X \to \mathsf{Pfin_q}\ X) \to \mathsf{Pfin_q}\ X \to \mathsf{Tree/TreeR}$ by precomposition with the coalgebra c. In turn, this can be obtained from the function $\mathsf{anaPfin''_q} : (X \to \mathsf{List}\ X)/\overline{\mathsf{Fun}}\ \mathsf{SameEls} \to \mathsf{List}\ X/\mathsf{SameEls} \to \mathsf{Tree/TreeR}$ by precomposition with the section ψ_{SameEls}. The latter is definable by pattern matching on both arguments: $\mathsf{anaPfin''_q}\ [c]\ [l] =_{\mathrm{df}} [\mathsf{anaTree}\ c\ l]$. The missing cases in the definition have been omitted. ◀

Without the assumption of the axiom of choice, one gets stuck in the construction of $\mathsf{anaPfin_q}$. In fact, the mediating coalgebra morphism may call the coalgebra $c : X \to \mathsf{Pfin_q}\ X$ an arbitrarily large number of times, and, since we are given no information on the cardinality of X, each application of c may happen on a different input $x : X$. This implies that generally the recursion principle of set quotients would need to be invoked the same large number of times, and this could only be achieved by assuming the full axiom of choice.

5 Analysis of Worrell's Classical Set-Theoretic Construction

In classical set theory there are many constructions of the final coalgebra of the finite powerset functor. See Adámek et al.'s collection and comparison of all these characterizations [4]. In this section we scrutinize a construction due to James Worrell as an $(\omega + \omega)$-limit [33]. Worrell's general construction of final coalgebras of finitary functors as $(\omega + \omega)$-limits can be seen as a generalization of the traditional construction of final coalgebras of polynomial functors as ω-limits. In the same spirit, one can consider our attempt at internalizing Worrell's construction in type theory, here in the special case of the final powerset functor, as a generalization of Ahrens et al.'s internalization of the construction of M-types in homotopy type theory [5]. We will see that a sprinkle of classical logic is needed for Worrell's construction to work in our constructive setting.

Worrell's construction starts by considering the ω-limit of the sequence

$$1 \xleftarrow{\;!\;} \mathsf{Pfin}\ 1 \xleftarrow{\;\mathsf{map}_{\mathsf{Pfin}}\ !\;} \mathsf{Pfin}^2\ 1 \xleftarrow{\;\mathsf{map}^2_{\mathsf{Pfin}}\ !\;} \dots \tag{12}$$

which in type theory can be encoded as the following dependent sum:

$$\mathsf{V}_\omega =_{\mathrm{df}} \sum x : (n : \mathbb{N}) \to \mathsf{Pfin}^n\ 1.\ (n : \mathbb{N}) \to \mathsf{map}^n_{\mathsf{Pfin}}\ !\ (x\ (\mathsf{suc}\ n)) = x\ n$$

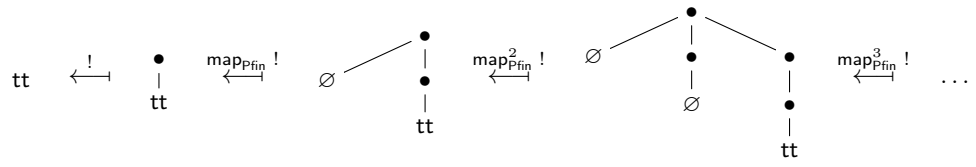
Here $\mathsf{Pfin}^n\ A$ is the n-iterated application of Pfin to type A, i.e. $\mathsf{Pfin}^{\mathsf{zero}}\ A =_{\mathrm{df}} A$ and $\mathsf{Pfin}^{\mathsf{suc}\ n} =_{\mathrm{df}} \mathsf{Pfin}\ (\mathsf{Pfin}^n\ A)$. Similarly, $\mathsf{map}^n_{\mathsf{Pfin}}$ is the n-iterated application of $\mathsf{map}_{\mathsf{Pfin}}$. Let ℓ_n be the function mapping an element of V_ω to its nth approximation in $\mathsf{Pfin}^n\ 1$, i.e. $\ell_n\ x =_{\mathrm{df}} \mathsf{fst}\ x\ n$. A function $\mathsf{alg}_{\mathsf{V}_\omega}$ from $\mathsf{Pfin}\ \mathsf{V}_\omega$ to V_ω can be constructed as follows, basically using the universal property of the ω-limit (we use copatterns and we only show the definition of the first projection): $\mathsf{fst}\ (\mathsf{alg}_{\mathsf{V}_\omega}\ s)\ n =_{\mathrm{df}} \mathsf{map}^n_{\mathsf{Pfin}}\ !\ (\mathsf{map}_{\mathsf{Pfin}}\ \ell_n\ s)$. As noticed by Adámek and Koubek [2], V_ω is not the final coalgebra of Pfin. This is because V_ω is not a fixpoint of Pfin, as the canonical algebra function $\mathsf{alg}_{\mathsf{V}_\omega}$ is not an isomorphism.

▶ **Proposition 5.** *The function* $\mathsf{alg}_{\mathsf{V}_\omega} : \mathsf{Pfin}\ \mathsf{V}_\omega \to \mathsf{V}_\omega$ *is not surjective.*

Proof. Consider the sequence

$$\begin{aligned}
&\mathsf{growing} : (n : \mathbb{N}) \to \mathsf{Pfin}^n 1\\
&\mathsf{growing}\ \mathsf{zero} &&=_{\mathrm{df}} \mathsf{tt}\\
&\mathsf{growing}\ (\mathsf{suc}\ \mathsf{zero}) &&=_{\mathrm{df}} \eta\ \mathsf{tt}\\
&\mathsf{growing}\ (\mathsf{suc}\ (\mathsf{suc}\ n)) &&=_{\mathrm{df}} \eta\ \varnothing \cup \mathsf{map}_{\mathsf{Pfin}}\ \eta\ (\mathsf{growing}\ (\mathsf{suc}\ n))
\end{aligned}$$

corresponding pictorially to the following element of V_ω:



The top-level branching of the sequence $\mathsf{growing}$ is, as the name suggests, growing. It is possible to show that it is absurd to assume that $\mathsf{growing}$ is in the image of $\mathsf{alg}_{\mathsf{V}_\omega}$. ◀

Elements of type V_ω represent non-wellfounded trees with unordered branching (as opposed to elements of type Tree, in which branching is ordered). The element $\mathsf{growing}$ introduced in the proof of Proposition 5 shows that these trees generally do not have finite

branching, even if all their finite approximations do. So V_ω cannot possibly be a fixpoint of Pfin, and, in particular, it cannot be its final coalgebra.

While the sequence in (12) does not stabilize in ω steps, Worrell shows that, in classical set theory, it stabilized after $\omega + \omega$ steps. To this end, he considers the ω-limit $\mathsf{V}_{\omega+\omega}$ of the sequence

$$\mathsf{V}_\omega \xleftarrow{\mathsf{alg}_{\mathsf{V}_\omega}} \mathsf{Pfin}\ \mathsf{V}_\omega \xleftarrow{\mathsf{map}_{\mathsf{Pfin}}\ \mathsf{alg}_{\mathsf{V}_\omega}} \mathsf{Pfin}^2\ \mathsf{V}_\omega \xleftarrow{\mathsf{map}^2_{\mathsf{Pfin}}\ \mathsf{alg}_{\mathsf{V}_\omega}} \dots \tag{13}$$

which in type theory corresponds to the dependent sum

$$\mathsf{V}_{\omega+\omega} =_{\mathrm{df}} \sum x : (n : \mathbb{N}) \to \mathsf{Pfin}^n\ \mathsf{V}_\omega.\ (n : \mathbb{N}) \to \mathsf{map}^n_{\mathsf{Pfin}}\ \mathsf{alg}_{\mathsf{V}_\omega}\ (x\ (\mathsf{suc}\ n)) = x\ n$$

and proves that $\mathsf{V}_{\omega+\omega}$ is the final coalgebra of Pfin. A fundamental ingredient in his proof is the fact that the function $\mathsf{alg}_{\mathsf{V}_\omega}$ is injective (even more, classically it is a split monomorphism), so that $\mathsf{Pfin}\ \mathsf{V}_\omega$ can be characterized as the subset of V_ω consisting of all the trees in which the top-level branching is finite. Consequently, $\mathsf{Pfin}^2\ \mathsf{V}_\omega$ consists of all trees in which the first two levels of branching are finite. The limit $\mathsf{V}_{\omega+\omega}$ can then be characterized as the subset of V_ω consisting of trees with finite branching at all levels.

In our constructive setting, the injectivity of $\mathsf{alg}_{\mathsf{V}_\omega}$ is not provable. In fact, under the assumption of the axiom of countable choice, injectivity of $\mathsf{alg}_{\mathsf{V}_\omega}$ is equivalent to the *lesser limited principle of omniscience (LLPO)*:

$$\begin{aligned}\mathsf{LLPO} =_{\mathrm{df}}\ & (a : \mathbb{N} \to \mathsf{Bool}) \to \mathsf{isProp}\ (\textstyle\sum n : \mathbb{N}.\ a\ n = \mathsf{true}) \\ & \to \|((n : \mathbb{N}) \to \mathsf{isEven}\ n \to a\ n = \mathsf{false}) + ((n : \mathbb{N}) \to \mathsf{isOdd}\ n \to a\ n = \mathsf{false})\|\end{aligned}$$

LLPO states that, if a Boolean stream a contains at most one occurrence of value true, then either all its even positions contain false or all its odd positions contain false. The axiom of countable choice is just AC in (11) with type A fixed to be $\mathbb{N}$, but we prefer to have a different (and more standard) equivalent formulation of countable choice that is directly applicable in the forthcoming constructions:

$$\mathsf{AC}_{\mathbb{N}} : (P : \mathbb{N} \to \mathsf{Type}) \to ((n : \mathbb{N}) \to \|P\ n\|) \to \|(n : \mathbb{N}) \to P\ n\|$$

Proving that the injectvity of $\mathsf{alg}_{\mathsf{V}_\omega}$ implies LLPO does not require countable choice. The proof is obtained as an adaptation of the proof of equivalence between LLPO and the completeness of finite sets of real numbers in Bishop-style constructive mathematics [22].

▶ **Theorem 6.** *From the injectivity of* $\mathsf{alg}_{\mathsf{V}_\omega}$ *we can construct the following term:*

$$\begin{aligned}&\mathsf{complete} : \{x\ y_1\ y_2 : \mathsf{V}_\omega\}\ (z : \mathbb{N} \to \mathsf{V}_\omega) \\ &\quad \to (p : (n : \mathbb{N}) \to z\ n = y_1 + z\ n = y_2)\ (q : (n : \mathbb{N}) \to \ell_n\ x = \ell_n\ (z\ n)) \\ &\quad \to x \in \eta\ y_1 \cup \eta\ y_2\end{aligned} \tag{14}$$

Proof. Assume given a sequence $z : \mathbb{N} \to \mathsf{V}_\omega$ with proof terms p and q as in the type above. Define two elements of $\mathsf{Pfin}\ \mathsf{V}_\omega$ as follows: $t =_{\mathrm{df}} \eta\ y_1 \cup \eta\ y_2$ and $s =_{\mathrm{df}} \eta\ x \cup t$. In order to prove $\mathsf{complete}\ z\ p\ q$, it is enough to show that $\mathsf{alg}_{\mathsf{V}_\omega}$ maps s and t to path equal elements, since then the injectivity of $\mathsf{alg}_{\mathsf{V}_\omega}$ would imply $s = t$ and therefore also $x \in t$. Proving $\mathsf{alg}_{\mathsf{V}_\omega}\ s = \mathsf{alg}_{\mathsf{V}_\omega}\ t$ is equivalent to show $\ell_n\ (\mathsf{alg}_{\mathsf{V}_\omega}\ s) = \ell_n\ (\mathsf{alg}_{\mathsf{V}_\omega}\ t)$ for all $n : \mathbb{N}$, which, unfolding the definition of $\mathsf{alg}_{\mathsf{V}_\omega}$, is also equivalent to show $\mathsf{map}_{\mathsf{Pfin}}\ \ell_n\ s = \mathsf{map}_{\mathsf{Pfin}}\ \ell_n\ t$ for all $n : \mathbb{N}$. Assuming $n : \mathbb{N}$, we invoke the antisymmetry of the subset relation and we are left to show $\mathsf{map}_{\mathsf{Pfin}}\ \ell_n\ s \subseteq \mathsf{map}_{\mathsf{Pfin}}\ \ell_n\ t$ (the other direction is trivial since $t \subseteq s$). Unfolding the definition of s, the only interesting case to prove is $\ell_n\ x \in \mathsf{map}_{\mathsf{Pfin}}\ \ell_n\ t$. The proof proceeds by case analysis on $p\ n$. If $z\ n = y_1$, then $\ell_n\ x = \ell_n\ (z\ n) = \ell_n\ y_1$, where the first path equality is given by $q\ n$, so $\ell_n\ x \in \mathsf{map}_{\mathsf{Pfin}}\ \ell_n\ t$. The case of $z\ n = y_2$ is analogous. ◀

Intuitively, complete corresponds to the completeness of two-element subsets of V_ω wrt. the pseudometric $d(x,y) =_{\mathrm{df}} \inf\{2^{-n} \mid n : \mathbb{N},\ \ell_n\ x = \ell_n\ y\}$ [4].

The proof of the next theorem requires the introduction of some auxiliary definitions. First, $\mathsf{long} : \mathsf{V}_\omega$ corresponds to the infinite tree in which each node has exactly one subtree. We only show the construction of the first projection (the definition uses copatterns).

$$\begin{aligned}&\mathsf{fst}\ \mathsf{long}\ \mathsf{zero} &&=_{\mathrm{df}} \mathsf{tt}\\ &\mathsf{fst}\ \mathsf{long}\ (\mathsf{suc}\ n) &&=_{\mathrm{df}} \eta\ (\mathsf{fst}\ \mathsf{long}\ n)\end{aligned}$$

Given a sequence $a : \mathbb{N} \to \mathsf{Bool}$, one can also define a variant $\mathsf{long?}\ a$ of long, which is the same as long if a contains only value false, but its height stop growing if there is $n : \mathbb{N}$ such that $a\ n$ is the first true in a. In the latter case, $\mathsf{long?}\ a$ is a finite tree with height n, so that $\mathsf{fst}\ \mathsf{long}\ (\mathsf{suc}\ n)$ is not equal to $\mathsf{fst}\ (\mathsf{long?}\ a)\ (\mathsf{suc}\ n)$.

$$\begin{aligned}&\mathsf{fst}\ (\mathsf{long?}\ a)\ \mathsf{zero} &&=_{\mathrm{df}} \mathsf{tt}\\ &\mathsf{fst}\ (\mathsf{long?}\ a)\ (\mathsf{suc}\ n) &&=_{\mathrm{df}} \mathsf{if}\ a\ \mathsf{zero}\ \mathsf{then}\ \varnothing\ \mathsf{else}\ \eta\ (\mathsf{fst}\ (\mathsf{long?}\ (a \circ \mathsf{suc}))\ n)\end{aligned}$$

Lastly, given a sequence $a : \mathbb{N} \to \mathsf{Bool}$, a type A with two elements $x, y : A$ and a Boolean b, we can define a sequence $\mathsf{seq}\ a\ x\ y\ b : \mathbb{N} \to A$ as follows:

$$\begin{aligned}&\mathsf{seq}\ a\ x\ y\ b\ \mathsf{zero} &&=_{\mathrm{df}} \mathsf{if}\ a\ \mathsf{zero}\ \mathsf{and}\ b\ \mathsf{then}\ y\ \mathsf{else}\ x\\ &\mathsf{seq}\ a\ x\ y\ b\ (\mathsf{suc}\ n) &&=_{\mathrm{df}} \mathsf{if}\ a\ \mathsf{zero}\ \mathsf{and}\ b\ \mathsf{then}\ y\ \mathsf{else}\ \mathsf{seq}\ (a \circ \mathsf{suc})\ x\ y\ (\mathsf{not}\ b)\ n\end{aligned}$$

The rationale behind the construction of the latter sequence, in the case when b is true, goes as follows: $\mathsf{seq}\ a\ x\ y\ \mathsf{true}\ n$ returns y if there exists an even number $k : \mathbb{N}$ with $k \leq n$ such that $a\ k = \mathsf{true}$ and $a\ j = \mathsf{false}$ for all $j < k$; in all other cases $\mathsf{seq}\ a\ x\ y\ \mathsf{true}\ n$ returns x.

▶ **Theorem 7.** *The existence of a term* complete *as in (14) implies* LLPO.

Proof. Let $a : \mathbb{N} \to \mathsf{Bool}$ be a sequence with at most one occurrence of value true. Define $y_1 =_{\mathrm{df}} \mathsf{long}$, $y_2 =_{\mathrm{df}} \mathsf{long?}\ a$ and $z =_{\mathrm{df}} \mathsf{seq}\ a\ y_1\ y_2\ \mathsf{true}$. Take x to be the diagonal of z, i.e. $\mathsf{fst}\ x\ n =_{\mathrm{df}} \ell_n\ z\ n$, which can in fact be proved to be an element of V_ω. Clearly each entry in z is either y_1 or y_2, therefore all the hypotheses in the type in (14) are satisfied. Applying complete to these hypotheses gives $x \in \eta\ y_1 \cup \eta\ y_2$. Invoking the recursion principle of propositional truncation on the resulting proof term, which we are allowed to use since the return type of LLPO is a proposition, gives us either $x = y_1$ or $x = y_2$. Assume $x = y_1$, we show $a\ n = \mathsf{false}$ for all even numbers $n : \mathbb{N}$. Suppose $a\ n = \mathsf{true}$ for a certain even number n. Since $a\ n = \mathsf{true}$, and this is the only true in a, we know that $z\ (\mathsf{suc}\ n) \equiv \mathsf{seq}\ a\ y_1\ y_2\ \mathsf{true}\ (\mathsf{suc}\ n) = y_2$ which in turn implies $\ell_{\mathsf{suc}\ n}\ x = \ell_{\mathsf{suc}\ n}\ (z\ (\mathsf{suc}\ n)) = \ell_{\mathsf{suc}\ n}\ y_2$. By assumption $\ell_{\mathsf{suc}\ n}\ x = \ell_{\mathsf{suc}\ n}\ y_1$, therefore by path composition and path inversion we get $\ell_{\mathsf{suc}\ n}\ y_1 = \ell_{\mathsf{suc}\ n}\ y_2$, i.e. $\mathsf{fst}\ \mathsf{long}\ (\mathsf{suc}\ n) = \mathsf{fst}\ (\mathsf{long?}\ a)\ (\mathsf{suc}\ n)$, which is impossible since $a\ n$ is true. So $a\ n$ must be false for all even n. Analogously one can prove that $x = y_2$ implies $a\ n = \mathsf{false}$ for all odd numbers $n : \mathbb{N}$, therefore concluding the derivation of LLPO. ◀

Patching together Theorems 6 and 7 shows that the injectivity of $\mathsf{alg}_{\mathsf{V}_\omega}$ implies LLPO.

▶ **Corollary 8.** *The injectivity of* $\mathsf{alg}_{\mathsf{V}_\omega}$ *implies* LLPO.

This displays the non-constructive nature of the injectivity of $\mathsf{alg}_{\mathsf{V}_\omega}$. The reverse implication also holds, which we have proved assuming the axiom of countable choice. We refrain from proving this in the paper, but the interested reader can find all the details in the Agda code.

▶ **Theorem 9.** *Assuming countable choice,* LLPO *implies the injectivity of* $\mathsf{alg}_{\mathsf{V}_\omega}$.

One can also modify the proofs of Corollary 8 and Theorem 9 to show that LLPO is also equivalent to the injectivity of the function $\ell_\omega : \mathsf{V}_{\omega+\omega} \to \mathsf{V}_\omega$ given by $\ell_\omega\ x =_{\mathrm{df}} \mathsf{fst}\ x\ \mathsf{zero}$. This demonstrates that Worrell's construction of the final coalgebra of Pfin as a subset of the limit V_ω is not achievable without the assumption of a certain amount of classical logic in the metatheory.

▶ **Theorem 10.** 1. *The injectivity of ℓ_ω implies* LLPO.
2. *Assuming countable choice,* LLPO *implies the injectivity of ℓ_ω.*

Having the injectivity of $\mathsf{alg}_{\mathsf{V}_\omega}$ at hand, the construction of a coalgebra structure on $\mathsf{V}_{\omega+\omega}$ and the proof of its finality morally follow Worrell's description [33].

▶ **Theorem 11.** *Assuming the axiom of countable choice and the injectvity of $\mathsf{alg}_{\mathsf{V}_\omega}$, $\mathsf{V}_{\omega+\omega}$ is a Pfin-coalgebra which is final.*

Proof. The meat of the proof lays in the construction of a function of type $\mathsf{V}_{\omega+\omega} \to \mathsf{Pfin}\ \mathsf{V}_{\omega+\omega}$. We show that the latter comes from an equivalence $\mathsf{V}_{\omega+\omega} \simeq \mathsf{Pfin}\ \mathsf{V}_{\omega+\omega}$ which is constructed in several steps. First define a family of functions $u : (n : \mathbb{N}) \to \mathsf{Pfin}^n\ \mathsf{V}_\omega \to \mathsf{V}_\omega$ by recursion: $u\ \mathsf{zero}\ x =_{\mathrm{df}} x$ and $u\ (\mathsf{suc}\ n)\ x =_{\mathrm{df}} u\ n\ (\mathsf{map}^n_{\mathsf{Pfin}}\ \mathsf{alg}_{\mathsf{V}_\omega}\ x)$. It is possible to prove that $\mathsf{V}_{\omega+\omega}$ is equivalent to the wide pullback $\bigcap u$ of the family of functions u. In general, given a family of functions $f : (n : \mathbb{N}) \to A\ n \to C$, its *wide pullback* is defined in type theory as

$$\bigcap f\ =_{\mathrm{df}} \sum x : (n : \mathbb{N}) \to A\ n.\ (n : \mathbb{N}) \to f\ (\mathsf{suc}\ n)\ (x\ (\mathsf{suc}\ n)) = f\ \mathsf{zero}\ (x\ \mathsf{zero})$$

We use the intersection symbol, and we refer to this pullback as *intersection*, since all the families of functions f that we consider have $f\ n$ injective, for all $n : \mathbb{N}$. In particular, each function $u\ n$ is injective, which can be proved by induction on n using the assumption that $\mathsf{alg}_{\mathsf{V}_\omega}$ is injective. This implies the existence of an equivalence $\mathsf{eqv}_1 : \mathsf{Pfin}\ \mathsf{V}_{\omega+\omega} \simeq \mathsf{Pfin}\ (\bigcap u)$.

In an analogous manner, one can prove that the intersection of the family $\mathsf{map}_{\mathsf{Pfin}} \circ u : (n : \mathbb{N}) \to \mathsf{Pfin}^{\mathsf{suc}\ n}\ \mathsf{V}_\omega \to \mathsf{Pfin}\ \mathsf{V}_\omega$ is equivalent to the ω-limit of the shifted sequence

$$\mathsf{Pfin}\ \mathsf{V}_\omega \xleftarrow{\mathsf{map}_{\mathsf{Pfin}}\ \mathsf{alg}_{\mathsf{V}_\omega}} \mathsf{Pfin}^2\ \mathsf{V}_\omega \xleftarrow{\mathsf{map}^2_{\mathsf{Pfin}}\ \mathsf{alg}_{\mathsf{V}_\omega}} \mathsf{Pfin}^3\ \mathsf{V}_\omega \xleftarrow{\mathsf{map}^3_{\mathsf{Pfin}}\ \mathsf{alg}_{\mathsf{V}_\omega}} \dots$$

It is well-known that the ω-limit of the shifted sequence is equivalent to the ω-limit of the original sequence in (13), i.e. $\mathsf{V}_{\omega+\omega}$. We obtain an equivalence $\mathsf{eqv}_3 : \bigcap(\mathsf{map}_{\mathsf{Pfin}} \circ u) \simeq \mathsf{V}_{\omega+\omega}$.

It is also possible to show, using the axiom of countable choice, that Pfin preserves intersections: given a generic family of injective functions $f : (n : \mathbb{N}) \to A\ n \to C$, the following equivalence exists: $\mathsf{eqv}_2 : \mathsf{Pfin}\ (\bigcap f) \simeq \bigcap(\mathsf{map}_{\mathsf{Pfin}} \circ f)$.

By composing equivalences $\mathsf{eqv}_1, \mathsf{eqv}_2$ and eqv_3, we obtain the desired equivalence showing that $\mathsf{V}_{\omega+\omega}$ is a fixpoint of Pfin. A Pfin-coalgebra for $\mathsf{V}_{\omega+\omega}$ is extracted as the function of type $\mathsf{V}_{\omega+\omega} \to \mathsf{Pfin}\ \mathsf{V}_{\omega+\omega}$ underlying this equivalence. It is possible to continue following Worrell's proof and show that this coalgebra is indeed final. ◀

6 Conclusions and Future Work

In this paper we discussed various presentations of the final coalgebra of the finite powerset functor in Cubical Agda: (i) as a setoid, (ii) as a coinductive type, (iii) as a set quotient and (iv) as a subset of an ω-limit. Construction (iii) requires the presence of the axiom of choice in the proof of finality, while construction (iv) corresponds to the classical construction of the final coalgebra as a $(\omega + \omega)$-limit by Worrell, which can be performed in our setting prior the assumption of countable choice and LLPO. For these reasons, we believe the best choice

to be number (*ii*), i.e. the coinductive type $\nu\mathsf{Pfin}$ of Section 4.2, since it does not require the assumption of classical principles such as choice or LLPO, and it does not force the user to employ setoids instead of types.

The work presented in this paper is motivated by our will to certify programming language semantics in proof assistants. We are specifically thinking about languages with nondeterministic or concurrent behavior. In previous work [29], we presented a fully abstract denotational model of the early π-calculus, mechanized in Guarded Cubical Agda. We believe possible to port these result to Cubical Agda using the presentations of the final Pfin-coalgebra of Section 4.2. Such an attempt would employ Cubical Agda's coinductive types instead of the guarded recursive types of Guarded Cubical Agda.

We wish to study the more general construction of final coalgebras of finitary functors in type theory. Frumin et al.'s functor Pfin captures a particular notion of finite type, known as Kuratowski finiteness: a type A is finite iff there exists a pair consisting of $x : \mathsf{Pfin}\ A$ and a proof that $(a : A) \to a \in x$. But in type theory, and more generally in constructive mathematics, there exist many more inequivalent formulations of finiteness [12, 25, 15, 16, 17]. We plan to investigate final coalgebras of finitary functors using these various formulations. In particular, we wonder if an alternative notion of finiteness in the specification of the finite powerset functor would make Worrell's proof go through without the assumption of additional classical principles. A large class of finitary functors should be definable via the syntax for set truncated HITs developed by Basold, Geuvers and van der Weide [8, 31].

The construction of the final coalgebra given in Section 4.2 used a higher inductive type in the domain of a coinductive type destructor. This definition is allowed in Cubical Agda, and it is intuitively justified by the treatment of HITs in cubical type theory as inductive types with constructors possibly depending on extra interval variables [11, 9]. We leave to future work a formal construction of the final coalgebra of the finite powerset and other finitary functors in the cubical set model [10]. Inspiration could be drawn from the recent model of clocked cubical type theory of Kristensen et al. [20], where HITs are shown to commute on the nose with limits modelling the notion of clock quantification.

References

1 Andreas Abel, Brigitte Pientka, David Thibodeau, and Anton Setzer. Copatterns: programming infinite structures by observations. In Roberto Giacobazzi and Radhia Cousot, editors, *Proc. of 40th Ann. ACM SIGPLAN-SIGACT Symp. on Principles of Programming Languages, POPL'13*, pages 27–38. ACM, 2013. doi:10.1145/2429069.2429075.

2 Jirí Adámek and Václav Koubek. On the greatest fixed point of a set functor. *Theoretical Computer Science*, 150(1):57–75, 1995. doi:10.1016/0304-3975(95)00011-K.

3 Jirí Adámek, Paul Blain Levy, Stefan Milius, Lawrence S. Moss, and Lurdes Sousa. On final coalgebras of power-set functors and saturated trees. *Applied Categorical Structures*, 23(4):609–641, 2015. doi:10.1007/s10485-014-9372-9.

4 Jiří Adámek, Stefan Milius, and Lawrence S. Moss. Initial algebras, terminal coalgebras, and the theory of fixed points of functors. Draft book, available from http://www.stefan-milius.eu, 2021.

5 Benedikt Ahrens, Paolo Capriotti, and Régis Spadotti. Non-wellfounded trees in homotopy type theory. In Thorsten Altenkirch, editor, *Proc. of 13th Int. Conf. on Typed Lambda Calculi and Applications, TLCA'15*, volume 38 of *Leibniz International Proceedings in Informatics*, pages 17–30. Schloss Dagstuhl, 2015. doi:10.4230/LIPIcs.TLCA.2015.17.

6 Michael Barr. Terminal coalgebras in well-founded set theory. *Theoretical Computer Science*, 114(2):299–315, 1993. doi:10.1016/0304-3975(93)90076-6.

7 Gilles Barthe, Venanzio Capretta, and Olivier Pons. Setoids in type theory. *Journal of Functional Programming*, 13(2):261–293, 2003. doi:10.1017/S0956796802004501.

8 Henning Basold, Herman Geuvers, and Niels van der Weide. Higher inductive types in programming. *Journal of Universal Computer Science*, 23(1):63–88, 2017. doi:10.3217/jucs-023-01-0063.

9 Evan Cavallo and Robert Harper. Higher inductive types in cubical computational type theory. *PACMPL*, 3(POPL):1:1–1:27, 2019. doi:10.1145/3290314.

10 Cyril Cohen, Thierry Coquand, Simon Huber, and Anders Mörtberg. Cubical type theory: A constructive interpretation of the univalence axiom. In Tarmo Uustalu, editor, *Proc. of 21st Int. Conf. on Types for Proofs and Programs, TYPES'15*, volume 69 of *Leibniz International Proceedings in Informatics*, pages 5:1–5:34. Schloss Dagstuhl, 2015. doi:10.4230/LIPIcs.TYPES.2015.5.

11 Thierry Coquand, Simon Huber, and Anders Mörtberg. On higher inductive types in cubical type theory. In Anuj Dawar and Erich Grädel, editors, *Proc. of the 33rd Ann. ACM/IEEE Symp. on Logic in Computer Science, LICS'18*, pages 255–264. ACM, 2018. doi:10.1145/3209108.3209197.

12 Thierry Coquand and Arnaud Spiwack. Constructively finite? In Laureano Lambàn, Ana Romero, and Julio Rubio, editors, *Scientific Contributions in Honor of Mirian Andrés Gómez*, pages 217–230. Universidad de La Rioja, 2010.

13 Nils Anders Danielsson. Bag equivalence via a proof-relevant membership relation. In Lennart Beringer and Amy P. Felty, editors, *Proc. of 3rd Int. Conf. on Interactive Theorem Proving, ITP'12*, volume 7406 of *Lecture Notes in Computer Science*, pages 149–165. Springer, 2012. doi:10.1007/978-3-642-32347-8_11.

14 Nils Anders Danielsson. Up-to techniques using sized types. *PACMPL*, 2(POPL):43:1–43:28, 2018. doi:10.1145/3158131.

15 Denis Firsov and Tarmo Uustalu. Dependently typed programming with finite sets. In Patrick Bahr and Sebastian Erdweg, editors, *Proc. of 11th ACM SIGPLAN Workshop on Generic Programming, WGP'15*, pages 33–44. ACM, 2015. doi:10.1145/2808098.2808102.

16 Denis Firsov, Tarmo Uustalu, and Niccolò Veltri. Variations on Noetherianness. In Robert Atkey and Neelakantan R. Krishnaswami, editors, *Proc. of 6th Wksh. on Mathematically Structured Functional Programming, MSFP'16*, volume 207 of *Electronic Proceedings in Theoretical Computer Science*, pages 76–88, 2016. doi:10.4204/EPTCS.207.4.

17 Dan Frumin, Herman Geuvers, Léon Gondelman, and Niels van der Weide. Finite sets in homotopy type theory. In *Proc. of 7th ACM SIGPLAN Int. Conf. on Certified Programs and Proofs, CPP'18*, pages 201–214. ACM, 2018. doi:10.1145/3167085.

18 John Hughes, Lars Pareto, and Amr Sabry. Proving the correctness of reactive systems using sized types. In *Proc. of 23rd ACM SIGPLAN-SIGACT Symp. on Principles of Programming Languages, POPL'96*, pages 410–423, 1996. doi:10.1145/237721.240882.

19 Yoshiki Kinoshita and John Power. Category theoretic structure of setoids. *Theoretical Computer Science*, 546:145–163, 2014. doi:10.1016/j.tcs.2014.03.006.

20 Magnus Baunsgaard Kristensen, Rasmus Ejlers Møgelberg, and Andrea Vezzosi. A model of clocked cubical type theory, 2021. arXiv:2102.01969.

21 Paul Blain Levy. Similarity quotients as final coalgebras. In Martin Hofmann, editor, *Proc. of 14th Int. Conf on Foundations of Software Science and Computational Structures, FoSSaCS'11*, volume 6604 of *Lecture Notes in Computer Science*, pages 27–41. Springer, 2011. doi:10.1007/978-3-642-19805-2_3.

22 Mark Mandelkern. Constructively complete finite sets. *Mathematical Logic Quarterly*, 34(2):97–103, 1988. doi:10.1002/malq.19880340202.

23 Robin Milner. *A calculus of communicating systems*, volume 92 of *Lecture Notes in Computer Science*. Springer, 1980. doi:10.1007/3-540-10235-3.

24 Robin Milner, Joachim Parrow, and David Walker. A calculus of mobile processes, I. *Information and Computation*, 100(1):1–40, 1992. doi:10.1016/0890-5401(92)90008-4.

25 Erik Parmann. Investigating streamless sets. In Hugo Herbelin, Pierre Letouzey, and Matthieu Sozeau, editors, *Proc. of 20th Int. Conf. on Types for Proofs and Programs, TYPES'14*, volume 39 of *Leibniz International Proceedings in Informatics*, pages 187–201. Schloss Dagstuhl, 2014. doi:10.4230/LIPIcs.TYPES.2014.187.

26 Jan J. M. M. Rutten. Universal coalgebra: a theory of systems. *Theoretical Computer Science*, 249(1):3–80, 2000. doi:10.1016/S0304-3975(00)00056-6.

27 The Univalent Foundations Program. *Homotopy type theory: Univalent foundations of mathematics*. https://homotopytypetheory.org/book, Institute for Advanced Study, 2013.

28 Niccolò Veltri. *A type-theoretical study of nontermination*. PhD thesis, Tallinn University of Technology, 2017. URL: https://digi.lib.ttu.ee/i/?7631.

29 Niccolò Veltri and Andrea Vezzosi. Formalizing π-calculus in Guarded Cubical Agda. In Jasmin Blanchette and Catalin Hritcu, editors, *Proc. of 9th ACM SIGPLAN Int. Conf. on Certified Programs and Proofs, CPP'20*, pages 270–283. ACM, 2020. doi:10.1145/3372885.3373814.

30 Andrea Vezzosi, Anders Mörtberg, and Andreas Abel. Cubical Agda: A dependently typed programming language with univalence and higher inductive types. *PACMPL*, 3(ICFP):87:1–87:29, 2019. doi:10.1145/3341691.

31 Niels van der Weide and Herman Geuvers. The construction of set-truncated higher inductive types. In Barbara König, editor, *Proc. of 35th Int. Conf. on Mathematical Foundations of Programming Semantics, MFPS'19*, volume 347 of *Electronic Notes in Theoretical Computer Science*, pages 261–280. Elsevier, 2019. doi:10.1016/j.entcs.2019.09.014.

32 James Worrell. Terminal sequences for accessible endofunctors. In Bart Jacobs and Jan J. M. M. Rutten, editors, *Proc. of 2nd Int. Wksh. on Coalgebraic Methods in Computer Science, CMCS'99*, volume 19 of *Electronic Notes in Theoretical Computer Science*, pages 24–38. Elsevier, 1999. doi:10.1016/S1571-0661(05)80267-1.

33 James Worrell. On the final sequence of a finitary set functor. *Theoretical Computer Science*, 338(1-3):184–199, 2005. doi:10.1016/j.tcs.2004.12.009.