How to Reason Coinductively Informally

Anton Setzer[*]

20 November 2015

Dedicated to Gerhard Jäger on occasion of his 60th Birthday

Abstract

We start by giving an overview of the theory of indexed inductively and coinductively defined sets. We consider the theory of strictly positive indexed inductive definitions in a set theoretic setting. We show the equivalence between the definition as an indexed initial algebra, the definition via an induction principle, and the set theoretic definition of indexed inductive definitions. We review as well the equivalence of unique iteration, unique primitive recursion, and induction. Then we review the theory of indexed coinductively defined sets or final coalgebras. We construct indexed coinductively defined sets set theoretically, and show the equivalence between the category theoretic definition, the principle of unique coiteration, of unique corecursion, and of iteration together with bisimulation as equality. Bisimulation will be defined as an indexed coinductively defined set. Therefore proofs of bisimulation can be carried out corecursively. This fact can be considered together with bisimulation implying equality as the coinduction principle for the underlying coinductively defined set. Finally we introduce various schemata for reasoning about coinductively defined sets in an informal way: the schemata of corecursion, of indexed corecursion, of coinduction, and of corecursion for coinductively defined relations. This allows to reason about coinductively defined sets similarly as one does when reasoning about inductively defined sets using schemata of induction. We obtain the notion of a coinduction hypothesis, which is the dual of an induction hypothesis.

1 Introduction

When reasoning about inductive defined sets such as the natural numbers, we are used to argue informally while referring to the induction hypothesis. When for instance showing $\forall x, y, z \in \mathbb{N}.(x + y) + z = x + (y + z)$, we do not define

[*]Department of Computer Science, Swansea University, Singleton Park, Swansea SA2 8PP, UK, Email: `a.g.setzer@swan.ac.uk`, `http://www.cs.swan.ac.uk/∼csetzer/`

first a relation $R(z) \Leftrightarrow \forall x, y.(x + y) + z = x + (y + z)$ and then argue that R is closed under 0 and successor S. Instead one proves $(x + y) + 0 = x + (y + 0)$ and proves $(x + y) + \mathrm{S}(z) = x + (y + \mathrm{S}(z)$ by using the induction hypothesis $(x + y) + z = x + (y + z)$.

Although these two versions are obviously equivalent, the version using the induction hypothesis is much more lightweight, and easier to teach to students.

When referring to coinductively defined sets, i.e. final coalgebras, we are currently usually following principles which are similar to referring to the closure of the relation R under $0, \mathrm{S}$ in inductive definitions. For instance when showing that two elements of a labelled transition system are bisimilar, one defines a relation on pairs of states of the transition system and shows that it is a bisimulation relation.

In this article we will discuss how to argue about coinductively defined sets in a similar way as we argue about inductive sets. This is made easier by following the approach in [1, 2, 27, 28] of introducing final coalgebras by their elimination rules rather than their introduction rules. For instance, instead of defining the set of streams of natural numbers as a set closed under $\mathrm{cons} : \mathbb{N} \to \mathrm{Stream} \to \mathrm{Stream}$ (and allowing infinite sequences of cons applications), we define Stream as a set such that we have $\mathrm{head} : \mathrm{Stream} \to \mathbb{N}$, and $\mathrm{tail} : \mathrm{Stream} \to \mathrm{Stream}$. This makes it easier to describe what the correct use of the corecursion hypothesis is: we can define $s : A \to \mathrm{Stream}$ by defining $\mathrm{head}(s(a)) \in \mathbb{N}$ and $\mathrm{tail}(s(a)) \in \mathrm{Stream}$. For defining tail we can use the corecursion hypothesis, i.e. define $\mathrm{tail}(s(a)) = s(a')$ for some a' (depending on a).

Coinduction is the dual of induction. In Sect. 3 we will review the well-known fact that the principle of induction is equivalent to the fact that there is only one solution for the equations defining a function by the principle of iteration. Therefore the principle of induction is just one way of expressing the fact that the principle of iteration has a unique solution. Dually, coinduction is a principle expressing that the principle of coiteration has a unique solution. In Sect. 8, Theorem 8.7 we will show that this principle is equivalent to the fact that bisimulation on coalgebras implies equality. Bisimulation can be defined coinductively. Therefore we can give proofs of bisimilarity by corecursion. Therefore coinduction can be considered as the principle that we can give proofs of equality by corecursion over the coinductive definition of bisimulation. The coinduction hypothesis is essentially the corecursion hypothesis in defining elements of the bisimilarity corecursively.

We hope that such schemata will make arguing about coinductively defined sets easier and less technical than it is at the moment.

We will in this article often use "*coinductively defined set*" for final coalgebra. The reason is that we want to use a terminology which suggests the use of corecursion and coinduction principles like those developed in this article, and which makes it clear that coinductively defined sets are the dual of inductively defined sets.

Content of this article We will start by introducing some notations in Sect. 2, where we will transfer notations from dependent type theory into set theory. Then we review in Sect. 3 the theory of indexed inductive definitions, and prove the equivalence between the category theoretic definition and the definition by induction. We use here restricted indexed inductive definitions as introduced in Peter Dybjer's and the author's articles [11, 13]. In Sect. 4 the notions of iteration and primitive recursion and their equivalence, if uniqueness is added, are reviewed. The main purpose of Sect. 3 and 4 is to motivate analogous definitions for coinductively defined sets, and make clear how they are obtained by dualising the concepts related to inductively defined sets. Our set theoretic definition of inductive definitions is based on defining its elements as terms, which are well-founded (in most standard examples therefore finitary) objects, and which can be represented in set theory in a straightforward way. Defining the elements of coinductively defined sets is more complicated, since the naïve interpretation using constructors would result in non-well-founded sets [3], whereas in ZF set theory all sets are well-founded. In Sect. 5 we give one way of introducing elements of coinductive sets set theoretically. Our construction is defined in such a way that it reflects the fact that coinductively defined sets are formed by giving their elimination rules or observations. In Sect. 6 we introduce the notions of coiteration, corecursion, and show the equivalence of those principles. In Sect. 7 we discuss a more convenient way of introducing elements of coinductively defined sets corecursively without having for each index to define a function. In Sect. 8 we introduce bisimulation, a principle of coinduction, and show that this principle of coinduction is equivalent to unique coiteration/corecursion. Finally in Sect. 9 we introduce various schemata for reasoning about coinductively defined sets informally. The schemata we introduce are corecursion, indexed corecursion, coinduction, and coinduction for bisimulation relations. We finish with a conclusion in Sect. 10.

We want to note that most of the material in Sect. 3 – 4 is well known in the theory of initial algebras and inductively defined sets. The purpose of those sections is to give an overview over the theory of indexed inductive definitions, so that it is easier to see in later sections how coinductively defined sets are the dual of inductively defined sets. Sect. 5 is the adaption of a well known categorical construction to the indexed case. We hope the fact that it is rather concrete and reflects the fact that coinductively defined sets are formed by their elimination rules or observations helps to get a better understanding of coinductively defined sets. The main contribution of this article are in Sects. 6 – 9, where the last section demonstrates, how to reason informally about coinductively defined sets.

We will work in this article set theoretically. The main reason for this is that the goal of this article is that ordinary mathematicians, who not necessarily work in type theory, should be able to use the schemata introduced in this article for reasoning about coalgebras. We believe that the reasoning principles can be transferred to extensional type theory, although further work is needed in order to make sure that all principles type-check. A transfer to intensional type theory, and therefore proof assistants such as Agda, would require further modifications. The main problem is that in order to obtain decidable type checking in intensional type theory one needs to replace final coalgebras by weakly final coalgebras. So coinduction can only be used to prove that elements are bisimilar rather than equal.

Related Work The equivalence between induction principles and category theoretic definition of initial algebras is well known, in case of inductive-recursive definitions it has for instance been worked out in [12], although the equivalence of inductive definitions has been known much longer. The reduction of indexed inductive-definitions to Petersson-Synek Trees has been developed in container theory, see esp. [19, 14] but as well [6, 21]. There are various set theoretic models of final coalgebras, examples are de Bruin [8], Barr [7] or Aczel [4]. The equivalence between final coalgebras and bisimulation as equality and iteration is well known in the theory of coalgebras, see for instance the articles and textbooks by Rutten and Sangiori [24, 25, 26] (the theory is much older). The notion of bisimulation of processes was initially defined by Park [22] and Milner [20] as a greatest fixed point, and therefore as a coinductively defined relation. Dybjer has defined a set theoretic interpretation of type theory in [9] and with the author in [10].

In our previous article [27] we introduced coalgebras into type theory by giving formation-, elimination-, introduction- and equality-rules. There we argued, that coalgebras are formed by giving their elimination rules, and that the introduction rules and equality rules are derived. We didn't explore the principle of coinduction in that article. The current article elaborates on this, however not in the context of type theory but in a general set theoretic setting. The difficulty is that in intensional type theory we obtain only weakly final coalgebras.

Acknowledgements. The author wants to thank the anonymous referee for valuable comments which greatly have improved this article. The diagrams in this article were typeset using the diagrams package by Paul Taylor.

2 Notations

In the following, we will work mainly set theoretically, using for simplicity the theory of Zermelo Fraenkel set theory with the axiom of choice. Since our inspiration comes from Martin-Löf type theory, we will simulate basic constructions in type theory in set theory.

We will work in this article in the set theoretical model of type theory, as introduced for instance in Sect. 6 of [10]. In this model inductively defined sets are modelled as sets of terms, introduced by constructors, and function types are modelled as set theoretic functions. Since the idea of this article is to work directly in set theory, we will identify inductively defined sets with the least set introduced by constructors, and function types with the set theoretic function set.

Assumption 2.1 *(a) We assume a finite set of constructor symbols* $\mathrm{C}_1, , \ldots,$ C_n *together with an arity* $\mathrm{arity}(\mathrm{C}_i) \in \mathbb{N}$ *associated with each of them.*

(b) We assume a Gödel number $\lceil \mathrm{C}_i \rceil \in \mathbb{N}$ *associated with each* C_i *such that* $\lceil \mathrm{C}_i \rceil \neq \lceil \mathrm{C}_j \rceil$ *for* $i \neq j$*.*

(c) We assume some standard encoding of sequences of sets $a_1, \ldots, a_n$ *as a set* $\langle a_1, \ldots, a_n \rangle$*, including the case* $n = 0$*. We assume this is done in such a way that there are functions which obtain from a code* $\langle a_1, \ldots, a_n \rangle$ *its length* n *and the* i*th element* a_i*.*

Definition 2.2 *(a) Let* Set *be the collection of sets.*

(b) We will in the following use set theoretic notation for function application, i.e. we will write $f(a)$ *for the application of* f *to* a*.*

(c) If C *is an* n*-ary constructor we define*

$$\begin{array}{l} \mathrm{C} : \mathrm{Set}^n \to \mathrm{Set} \\ \mathrm{C}(t_1, \ldots, t_n) := \langle \lceil \mathrm{C} \rceil, t_1, \ldots, t_n \rangle \end{array}$$

Definition 2.3 *(a) Let* $A \in \mathrm{Set}$ *and* $B(x) \in \mathrm{Set}$ *depending on* $x \in A$*. We define the dependent function set as*

$$(a \in A) \to B(a) := \{f \in A \to \bigcup_{a \in A} B(a) \mid \forall a \in A. f(a) \in B(a)\}$$

and the dependent product as

$$(a \in A) \times B(a) := \{\langle a, b \rangle \mid a \in A, b \in B(a)\}$$

Let π_0 *and* π_1 *be the first and second projections, i.e.* $\pi_0(\langle a, b \rangle) = a$*,* $\pi_1(\langle a, b \rangle) = b$*.*

(b)

$$\begin{array}{l} (x_1 \in A_1) \to (x_2 \in A_2) \to \cdots \to (x_n \in A_n) \to B \\ \quad := (x_1 \in A_1) \to ((x_2 \in A_2) \to (\cdots \to ((x_n \in A_n) \to B) \cdots)) \end{array}$$

(c)

$$\begin{array}{l} (x_1 \in A_1) \times (x_2 \in A_2) \times \cdots \times (x_n \in A_n) \\ \quad := (x_1 \in A_1) \times ((x_2 \in A_2) \times (\cdots \times (x_n \in A_n) \cdots)) \end{array}$$

(d) For $A, B \in \mathrm{Set}$ *let* $A + B := \{\mathrm{inl}(a) \mid a \in A\} \cup \{\mathrm{inr}(b) \mid b \in B\}$*, where* $\mathrm{inl}, \mathrm{inr}$ *are unary constructors.*

(e) $\times$ *binds stronger than* $+$ *and* $+$ *binds stronger than* $\to$*.*

(f) Let $\mathbf{1} := \{*\}$ *where* $*$ *is a 0-ary constructor.*

(g) *Let for a relation* $R(x_1, \ldots, x_n)$

$$\widehat{R}(x_1, \ldots, x_n) := \begin{cases} \mathbf{1} & \textit{if } R(x_1, \ldots, x_n) \\ \emptyset & \textit{otherwise} \end{cases}$$

When writing an argument of a function as being an element of a relation, we write $R(x_1, \ldots, x_n)$ *instead of* $\widehat{R}(x_1, \ldots, x_n)$. *For instance* $(n \in \mathbb{N}) \to (n > 0) \to \cdots$ *means more precisely* $(n \in \mathbb{N}) \to (n \mathbin{\widehat{>}} 0) \to \cdots$.

(h) *When having functions* $f : (x \in A) \to (y \in B(x)) \to C(x, y)$ *we write* $f(x, y)$ *for* $f(x)(y)$, *similarly for functions with more arguments.*

(i) *When referring to a function* $f : (x \in A) \to (y \in B(x)) \to C(x, y)$ *in a diagram we sometimes need its uncurried form* $\widehat{\mathrm{f}} : (x \in A) \times (y \in B(x)) \to C(x, y)$. *In order to reduce notational overhead we will usually write* f *instead of* $\widehat{\mathrm{f}}$.

(j) *When defining* $f : (x \in (A \times B)) \to (c \in C(x)) \to D(x, c)$ *we write* $f(a, b, c)$ *instead of* $f(\langle a, c \rangle, c)$, *similarly for longer products or functions with more arguments.*

Definition 2.4 (a) *For* $\mathrm{I} \in \mathrm{Set}$ *let* $\mathrm{Set}^{\mathrm{I}}$ *be the category of* I*-indexed sets with objects* $A \in \mathrm{I} \to \mathrm{Set}$ *and morphisms* $f : A \to B$ *being set theoretic functions* $f : (i \in \mathrm{I}) \to A(i) \to B(i)$.

(b) *For* $A, B \in \mathrm{Set}^{\mathrm{I}}$, *Let* $A +_{\mathrm{Set}^{\mathrm{I}}} B := \lambda i.A(i) + B(i)$, $A \times_{\mathrm{Set}^{\mathrm{I}}} B = \lambda i.A(i) \times B(i)$. *Furthermore, let*

$$\mathrm{inl}_{\mathrm{Set}^{\mathrm{I}}} := \lambda i, x.\mathrm{inl}(x) : A \to A + B$$

similarly for $\mathrm{inr}, \pi_0, \pi_1$.

(c) *For* $X \subseteq \mathrm{Set}^{\mathrm{I}}$ *let*

$$\begin{array}{lcl} \bigcap_{\mathrm{Set}^{\mathrm{I}}} X & := & \lambda i. \bigcap \{y(i) \mid y \in X\} \\ \bigcup_{\mathrm{Set}^{\mathrm{I}}} X & := & \lambda i. \bigcup \{y(i) \mid y \in X\} \end{array}$$

(d) *For* $X, Y \in \mathrm{Set}^{\mathrm{I}}$ *let* $X \subseteq_{\mathrm{Set}^{\mathrm{I}}} Y :\Leftrightarrow \forall i \in \mathrm{I}.X(i) \subseteq Y(i)$.

(e) *We will usually omit the index* $\mathrm{Set}^{\mathrm{I}}$ *in the notations introduced above.*

3 Initial Algebras and Inductively Defined Sets

We consider in the following the theory of simultaneous inductive definitions of sets $\mathrm{D}(i)$ for $i \in \mathrm{I}$. We fix $\mathrm{I} \in \mathrm{Set}$.

In [11, 13] Dybjer and the author introduced indexed inductive-recursive definitions. We defined an indexed inductively defined set $\mathrm{U} : \mathrm{I} \to \mathrm{Set}$ while simultaneously recursively defining a function $\mathrm{T} : (i \in \mathrm{I}) \to \mathrm{U}(i) \to E[i]$ for some type $E[i]$. $\mathrm{U}(i)$ was a universe of codes for elements of a type, and $\mathrm{T}(i,u)$ was the type corresponding to code u. The special case of indexed inductively defined sets (more precisely strictly positive indexed inductively defined sets) is obtained by taking $E[i] = \mathbf{1}$. Therefore T is equal to $\lambda i, x.*$. T becomes trivial and can be omitted. We call the set defined inductively in the following D instead of U and omit in the following T.

In [11, 13] we considered two versions of indexed inductive(-recursive) definitions, restricted and generalised ones. Generalised inductive definitions have constructors of the form

$$\begin{aligned}&\mathrm{C} : (x_1 \in A_1) \to (x_2 \in A_2(x_1)) \to \cdots \to (x_n \in A_n(x_1, \ldots, x_{n-1})) \\ &\quad \to \mathrm{D}(\mathrm{i}(x_1, \ldots, x_n))\end{aligned}$$

whereas in restricted ones the index of the result of C is given by the first argument, so

$$\begin{aligned}&\mathrm{C} : (i \in \mathrm{I}) \to (x_1 \in A_1(i)) \to (x_2 \in A_2(i, x_1)) \to \cdots \to (x_n \in A_n(i, x_1, \ldots, x_{n-1})) \\ &\quad \to \mathrm{D}(i)\end{aligned}$$

Restricted indexed inductive definitions allow decidable case distinction on elements of the set D defined inductively: an element of $\mathrm{D}(i)$ must be of the form $\mathrm{C}(i, x_1, \ldots, x_n)$ for one of the constructors of D. In case of general indexed inductive definitions we can in general not decide whether $\mathrm{C}(x_1, \ldots, x_n)$ forms an element of $\mathrm{D}(i)$, since we can in general not decide whether $\mathrm{i}(x_1, \ldots, x_n) = \mathrm{i}$.

We consider in the following only restricted indexed inductive definitions, since indexed inductive definitions are here mainly treated in order to motivate coinductively defined sets later, for which restricted ones are the natural choice.

Strictly positive restricted indexed inductive definitions are the least sets closed under constructors like C as before. In a notation borrowed from the type theoretic theorem prover Agda we write for the fact that Tree is this least set:

$$\begin{array}{l}\mathrm{data}\ \mathrm{D} : \mathrm{I} \to \mathrm{Set}\ \mathrm{where} \\ \quad \begin{array}{lcl}\mathrm{C} & : & (i \in \mathrm{I}) \to (x_1 \in A_1(i)) \to \cdots \to (x_n \in A_n(i, x_1, \ldots, x_{n-1})) \to \mathrm{D}(i) \\ \mathrm{C}' & : & (i \in \mathrm{I}) \to (y_1 \in A'_1(i)) \to \cdots \to (y_m \in A'_m(i, y_1, \ldots, y_{m-1})) \to \mathrm{D}(i) \\ \cdots & & \end{array}\end{array}$$

Strict positivity means that $A_k(i, \vec{x})$ are either sets which were defined before $\mathrm{D}(i)$ was introduced (non-inductive arguments), or are of the form $(b \in B(i, \vec{x})) \to \mathrm{D}(j(i, \vec{x}, b))$ (inductive arguments). Since we do not know anything about $\mathrm{D}(i)$, later arguments cannot depend on previous inductive arguments.[1]

Therefore we obtain an equivalent inductive definition by moving all inductive arguments to the end. Now we can replace all non-inductive arguments by one single one by forming a product (and letting the later arguments depend on the projections). The inductive arguments $((b \in B_1(i, \vec{x})) \to$

[1]This holds only in indexed inductive-definitions; in indexed inductive-recursive definitions arguments can depend on T applied to previous inductive arguments.

$\mathrm{D}(j_1(i,\vec{x},b))) \to \cdots \to ((b \in B_k(i,\vec{x})) \to \mathrm{D}(j_k(i,\vec{x},b)) \to$ can be replaced by $((b \in (B_1(i,\vec{x}) \times \cdots \times B_k(i,\vec{x}))) \to \mathrm{D}(j'(i,\vec{x},b))$ for some suitable j' (in the special case where there is no inductive argument, we obtain an inductive argument $\emptyset \to \mathrm{D}$). Therefore an inductive definition can be replaced by one having constructors of the form

$$\mathrm{C}_k : (i \in \mathrm{I}) \to (a \in A_k(i)) \to ((b \in B_k(i,a)) \to \mathrm{D}(j(i,a,b))) \to \mathrm{D}(i)$$

Assuming we have constructors $\mathrm{C}_0, \ldots, \mathrm{C}_{n-1}$ we can replace all constructors by one single one of type

$$\begin{aligned}\mathrm{C} : {} & (i \in \mathrm{I}) \\ & \to (k \in \{0, \ldots, n-1\}) \\ & \to (a \in A_k(i)) \\ & \to ((b \in B_k(i,a)) \to \mathrm{D}(j(i,a,b))) \\ & \to \mathrm{D}(i)\end{aligned}$$

which after merging the two non-inductive arguments into one becomes

$$\mathrm{C} : (i \in \mathrm{I}) \to (a \in \mathrm{A}(i)) \to ((b \in \mathrm{B}(i,a)) \to \mathrm{D}(\mathrm{j}(i,a,b))) \to \mathrm{D}(i)$$

This is the Petersson-Synek Tree ([23]), which is an indexed version of Martin-Löf's W-type. The Petersson-Synek trees subsume all strictly positive inductive definitions. They are initial algebras of indexed containers in the theory of containers, see [6, 21]. In [14, 19] a formal proof that initial algebras of indexed containers and therefore Petersson-Synek trees subsume all indexed inductive definitions is given.

We write in the following Tree instead of D and tree for the constructor C. Let us fix in the following $\mathrm{A}, \mathrm{B}, \mathrm{j}$:

Assumption 3.1 *(a) In the following assume*

$$\begin{array}{lcl}\mathrm{I} & \in & \mathrm{Set} \\ \mathrm{A} & : & \mathrm{I} \to \mathrm{Set} \\ \mathrm{B} & : & (i \in \mathrm{I}) \to \mathrm{A}(i) \to \mathrm{Set} \\ \mathrm{j} & : & (i \in \mathrm{I}) \to (a \in \mathrm{A}(i)) \to \mathrm{B}(i,a) \to \mathrm{I}\end{array}$$

(b) Let tree *be a constructor of arity* 3.

In the above we have

$$\begin{aligned}\mathrm{tree} : {} & (i \in \mathrm{I}) \to (a \in \mathrm{A}(i)) \to ((b \in \mathrm{B}(i,a)) \to \mathrm{Tree}(\mathrm{j}(i,a,b))) \\ & \to \mathrm{Tree}(i)\end{aligned}$$

In the data-notation introduced above we denote this by:

$$\begin{array}{l}\mathrm{data\ Tree} : \mathrm{I} \to \mathrm{Set\ where} \\ \quad \mathrm{tree} \quad : \quad (i \in \mathrm{I}) \to (a \in \mathrm{A}(i)) \to ((b \in \mathrm{B}(i,a)) \to \mathrm{Tree}(\mathrm{j}(i,a,b))) \\ \qquad\qquad\quad \to \mathrm{Tree}(i)\end{array}$$

We will now repeat the well-known argument, that the categorical definition of inductive definitions is equivalent to the induction principle. The dual of this argument will then be used to determine the equivalence between the categorical definition of coalgebras and the corresponding coinduction principle.

Definition 3.2 *(a) Let the functor* $\mathrm{F} : \mathrm{Set}^{\mathrm{I}} \to \mathrm{Set}^{\mathrm{I}}$ *be given by*

$$\begin{array}{l} \mathrm{F}(X, i) := (a \in \mathrm{A}(i)) \times ((b \in \mathrm{B}(i, a)) \to X(\mathrm{j}(i, a, b))) \\ \text{and for } f : X \to Y \\ \mathrm{F}(f) : \mathrm{F}(X) \to \mathrm{F}(Y) \\ \mathrm{F}(f, i, \langle a, g\rangle) := \langle a, \lambda b. f(\mathrm{j}(i, a, b), g(b))\rangle \end{array}$$

(b) An F*-algebra, where* F *is as above, is a pair* (X, f) *such that* $X \in \mathrm{Set}^{\mathrm{I}}$ *and* $f : \mathrm{F}(X) \to X$*.*

(c) The categorical definition[2] *of* Tree *is that* (Tree, tree) *is an initial* F*-algebra*[3]*, which means:*

- (Tree, tree) *is an* F*-algebra.*
- *For any other* F*-algebra* (X, f) *there exists a unique* $g : \mathrm{Tree} \to X$ *s.t. the following diagram commutes*

$$\begin{tikzcd} \mathrm{F}(\mathrm{Tree}) \arrow[r, "\mathrm{tree}"] \arrow[d, "\mathrm{F}(g)"'] & \mathrm{Tree} \arrow[d, "\exists! g"] \\ \mathrm{F}(X) \arrow[r, "f"] & X \end{tikzcd}$$

We call g *the unique* F*-algebra homomorphism into* (X, f)*.*

(d) The inductive definition of Tree *is given by*[4]

- (Tree, tree) *is an* F*-algebra*
- *for any formula* $\varphi(i, x)$ *depending on* $i \in \mathrm{I}$ *and* $x \in \mathrm{Tree}(i)$ *we have that if*

$$\begin{array}{l} \forall i \in \mathrm{I}. \forall a \in \mathrm{A}(i). \forall f \in (b \in \mathrm{B}(i, a)) \to \mathrm{Tree}(\mathrm{j}(i, a, b)). \\ \quad (\forall b \in \mathrm{B}(i, a). \varphi(\mathrm{j}(i, a, b), f(b))) \\ \quad \to \varphi(i, \mathrm{tree}(i, a, f)) \end{array} \qquad (\mathrm{Prog}(\varphi))$$

[2]Note that we deviate from standard category theory in so far as we fix the function tree: tree is the curried version of the constructor, which we introduced before. In standard category theory both the set Tree and the function tree can be arbitrary, and therefore the initial algebra is only unique up to isomorphism. Note as well that above we had the convention that we identify tree with its uncurried form $\widehat{\mathrm{tree}}$. Without this convention one would say that $(\mathrm{Tree}, \widehat{\mathrm{tree}})$ is an F-algebra.

[3]Here, F is as above, i.e. strictly positive.

[4]Again tree is the curried version of the constructor defined before.

then

$$\forall i \in \mathrm{I}.\forall x \in \mathrm{Tree}(i).\varphi(i,x)$$

We call the assumption $\mathrm{Prog}(\varphi)$ *that "φ is progressive".*

(e) *The set theoretic definition of* Tree *is given by*

$$\mathrm{Tree} = [\![\,\mathrm{Tree}\,]\!]$$

where

$$[\![\,\mathrm{Tree}\,]\!] := \bigcap\{X \in \mathrm{Set}^{\mathrm{I}} \mid (X, \mathrm{tree}) \textit{ is an } \mathrm{F}\textit{-algebra}\}$$

Lemma 3.3 $[\![\,\mathrm{Tree}\,]\!]$ *is a set.*

Proof: We repeat the standard argument. Define by induction on the ordinals $\mathrm{F}^{\alpha}, \mathrm{F}^{<\alpha} \in \mathrm{Set}^{\mathrm{I}}$,

$$\begin{array}{lcl} \mathrm{F}^{\alpha}(i) & := & \{\mathrm{tree}(i,a,f) \mid \langle a,f\rangle \in \mathrm{F}(\mathrm{F}^{<\alpha}, i)\} \\ \mathrm{F}^{<\alpha} & := & \bigcup_{\beta<\alpha} \mathrm{F}^{\beta} \end{array}$$

F is monotone, and therefore $\mathrm{F}^{\alpha} \subseteq \mathrm{F}^{\beta}$ for $\alpha < \beta$. Let κ be a regular infinite cardinal, $\kappa > \mathrm{card}(B(i,a))$ for $i \in \mathrm{I}$ and $a \in \mathrm{A}(i)$ (where $\mathrm{card}(x)$ is the cardinality of x).

We show that $(\mathrm{F}^{<\kappa}, \mathrm{tree})$ is an F-algebra. Assume $\langle a,f\rangle \in \mathrm{F}(\mathrm{F}^{<\kappa}, i)$. Then $a \in \mathrm{A}(i)$, $f \in (b \in \mathrm{B}(i,a)) \to \mathrm{F}^{<\kappa}(\mathrm{j}(i,a,b))$. Therefore, for $b \in \mathrm{B}(i,a)$ there exist $\beta < \kappa$ s.t. $f(b) \in \mathrm{F}^{\beta}(\mathrm{j}(i,a,b))$. By the regularity of κ and $\kappa > \mathrm{card}(\mathrm{B}(i,a))$ there exists a $\gamma < \kappa$ s.t. for all $b \in \mathrm{B}(i,a)$ we have $f(b) \in \mathrm{F}^{\gamma}(\mathrm{j}(i,a,b))$. Therefore $\mathrm{tree}(i,a,f) \in \mathrm{F}^{\gamma+1}(i) \subseteq \mathrm{F}^{<\kappa}(i)$.

It follows $[\![\,\mathrm{Tree}\,]\!] \subseteq \mathrm{F}^{<\kappa}$ which is a set.

In fact $[\![\,\mathrm{Tree}\,]\!] = \mathrm{F}^{<\kappa}$, since one can show by induction on α that for any F-algebra (X, tree) we have $\mathrm{F}^{\alpha} \subseteq X$, and therefore $\mathrm{F}^{<\kappa} \subseteq X$, so $(\mathrm{F}^{\kappa}, \mathrm{tree})$ is the initial algebra.

The following theorem is well known. We show it since it provides the key idea for the coinduction principle introduced later.

Theorem 3.4 *The following is equivalent:*

(a) *The categorical definition of* Tree.

(b) *The inductive definition of* Tree.

(c) *The set theoretic definition of* Tree.

Proof: (a) implies (b): Let $\varphi(i,x)$ be progressive. Define $E \in \mathrm{Set}^{\mathrm{I}}$, $E(i) := \{x \in \mathrm{Tree}(i) \mid \varphi(i,x)\}$. By progressivity of φ we obtain $\mathrm{tree} : \mathrm{F}(E) \to E$, therefore (E, tree) is an F-algebra. Let $h := \lambda i.x.x : E \to \mathrm{Tree}$ be the embedding function, g the unique F-algebra homomorphism $E \to \mathrm{Tree}$, and consider

$$\begin{tikzcd}
\mathrm{F}(\mathrm{Tree}) \arrow[r, "\mathrm{tree}"] \arrow[d, "\mathrm{F}(g)"'] & \mathrm{Tree} \arrow[d, "\exists g"] \\
\mathrm{F}(E) \arrow[r, "\mathrm{tree}"] \arrow[d, "\mathrm{F}(h)"'] & E \arrow[d, "h"] \\
\mathrm{F}(\mathrm{Tree}) \arrow[r, "\mathrm{tree}"] & \mathrm{Tree}
\end{tikzcd}$$

The upper diagram commutes by definition of g. The lower diagram obviously commutes. $h \circ g : \mathrm{Tree} \to \mathrm{Tree}$ and the identity function $\mathrm{id} : \mathrm{Tree} \to \mathrm{Tree}$ are two functions which make the outer diagram commute. By uniqueness of this function we get that $h \circ g = \mathrm{id}$, i.e. $\forall i \in \mathrm{I}.\forall x \in \mathrm{Tree}(i).g(i, x) = x$, and therefore $\forall i \in \mathrm{I}.\forall x \in \mathrm{Tree}(i).x \in E(i)$, $\forall i \in \mathrm{I}.\forall x \in \mathrm{Tree}(i).\varphi(i, x)$.

Proof of (b) implies (a): Let (X, f) be an F-homomorphism. The existence of a unique g follows as for the recursion theorem in set theory: One first defines for $i \in \mathrm{I}$ and $t \in \mathrm{Tree}(i)$ $\mathrm{TC}(i, t)$ as the least set such that,

- if $t = \mathrm{tree}(i, a, g)$, $b \in \mathrm{B}(i, a)$ then $\langle \mathrm{j}(i, a, b), g(b)\rangle \in \mathrm{TC}(i, t)$,
- if $\langle i', \mathrm{tree}(i', a, g)\rangle \in \mathrm{TC}(i, t)$ and $b \in \mathrm{B}(i', a)$ then $\langle \mathrm{j}(i', a, b), g(b)\rangle \in \mathrm{TC}(i, t)$.

So $\mathrm{Tree}(i, t)$ contains all proper subtrees of t and contains for every tree its subtrees.

Then it follows that we have course of value induction on Tree, i.e. if φ is course of value progressive, written $\mathrm{Prog}_{\mathrm{coursevalue}}(\varphi)$, i.e.

$$\forall i \in \mathrm{I}.\forall t \in \mathrm{Tree}(i).(\forall \langle i', t'\rangle \in \mathrm{TC}(i, t).\varphi(i', t')) \to \varphi(i, t)$$

then $\forall i \in \mathrm{I}.\forall t \in \mathrm{Tree}(i).\varphi(i, t)$. When showing this one shows first by induction on $i \in \mathrm{I}$, $t \in \mathrm{Tree}(i)$ that $\forall i \in \mathrm{I}.\forall t \in \mathrm{Tree}(i).\forall \langle i', y\rangle \in \mathrm{TC}(i, t).\varphi(i', y)$, which implies $\forall i \in \mathrm{I}.\forall t \in \mathrm{Tree}(i).\varphi(i, t)$. Let $\mathrm{TC}'(i, t) \in \mathrm{Set}^{\mathrm{I}}$, $\mathrm{TC}'(i, t, i') = \{t' \mid \langle i', t'\rangle \in \mathrm{TC}(i, t)\}$. Then one shows by course of value induction that for every $i \in \mathrm{I}$, $t \in \mathrm{Tree}(i)$ there exists a unique function $g : \mathrm{TC}'(i, t) \to X$ which fulfils the condition of the iteration principle given by the categorical diagram, restricted to $\mathrm{TC}'(i, t)$. We now obtain a function $g : \mathrm{Tree} \to X$ fulfilling the same equations, and show easily its uniqueness.

Proof of (c) implies (b): Assume φ is progressive. Define E as in the direction "(a) implies (b)". (E, tree) is an F-algebra, therefore $\mathrm{Tree} \subseteq E$.

Proof of (b) implies (c): We show first by induction on Tree that $\forall i \in \mathrm{I}.\forall x \in \mathrm{Tree}(i).x \in [\![\mathrm{Tree}]\!](i)$, therefore $\mathrm{Tree} \subseteq [\![\mathrm{Tree}]\!]$. Furthermore, (b) implies that $(\mathrm{Tree}, \mathrm{tree})$ is an F-algebra, and therefore $[\![\mathrm{Tree}]\!] \subseteq \mathrm{Tree}$.

4 Iteration, Recursion, Induction

In Sect. 6 we will introduce the principles of coiteration and corecursion. In order to see that these principles are the dual of iteration and primitive recursion, we repeat in this section the definition of those principles as well as the principle of type theoretic induction. We will give as well the (well-known) proof that the principles of being an initial F-algebra, of unique iteration, of unique primitive recursion, and of type theoretic induction are equivalent, which will as well be dualised in Sect. 6.

Definition 4.1 *Assume* $\mathrm{Tree} : \mathrm{I} \to \mathrm{Set}$ *and* $\mathrm{tree} : \mathrm{F}(\mathrm{Tree}) \to \mathrm{Tree}$.[5]

(a) By "$(\mathrm{Tree}, \mathrm{tree})$ *satisfies the principle of unique iteration" we mean the following: Assume*

$$\begin{array}{lll} X & : & \mathrm{I} \to \mathrm{Set} \\ f & : & (i \in \mathrm{I}) \to ((a \in \mathrm{A}(i)) \times ((b \in \mathrm{B}(i,a)) \to X(\mathrm{j}(i,a,b)))) \\ & & \qquad\qquad \to X(i) \end{array}$$

Then there exists a unique $g : \mathrm{Tree} \to X$ *such that*

$$g(i, \mathrm{tree}(i,a,h)) = f(i, \langle a, \lambda b.g(\mathrm{j}(i,a,b), h(b))\rangle)$$

(b) By "$(\mathrm{Tree}, \mathrm{tree})$ *satisfies the principle of unique primitive recursion" we mean the following: Assume*

$$\begin{array}{lll} X & : & \mathrm{I} \to \mathrm{Set} \\ f & : & (i \in \mathrm{I}) \to ((a \in \mathrm{A}(i)) \times \\ & & \qquad\qquad ((b \in \mathrm{B}(i,a)) \to (\mathrm{Tree}(\mathrm{j}(i,a,b)) \times X(\mathrm{j}(i,a,b))))) \\ & & \qquad\quad \to X(i) \end{array}$$

Then there exists a unique $g : \mathrm{Tree} \to X$ *such that*

$$g(i, \mathrm{tree}(i,a,h)) = f(i, \langle a, \lambda b.\langle h(b), g(\mathrm{j}(i,a,b), h(b))\rangle\rangle)$$

(c) By "$(\mathrm{Tree}, \mathrm{tree})$ *satisfies the principle of unique type theoretic induction" we mean the following: Assume*

$$\begin{array}{lll} X & : & (i \in \mathrm{I}) \to \mathrm{Tree}(i) \to \mathrm{Set} \\ f & : & (i \in \mathrm{I}) \to ((a \in \mathrm{A}(i)) \times \\ & & \qquad\qquad (h : (b \in \mathrm{B}(i,a)) \to ((t \in \mathrm{Tree}(\mathrm{j}(i,a,b))) \times X(\mathrm{j}(i,a,b),t)))) \\ & & \qquad\quad \to X(i, \mathrm{tree}(i,a,\pi_0 \circ h)) \end{array}$$

Then there exists a unique $g : (i \in \mathrm{I}) \to (t \in \mathrm{Tree}(i)) \to X(i,t)$ *such that*

$$g(i, \mathrm{tree}(i,a,h)) = f(i, \langle a, \lambda b.\langle h(b), g(\mathrm{j}(i,a,b), h(b))\rangle\rangle)$$

[5] Note that in contrast to other sections, tree can be an arbitrary function of this type, and Tree is assumed just to be an element of $\mathrm{Set}^{\mathrm{I}}$.

(d) By "(Tree, tree) *satisfies the principle of iteration, primitive recursion or type theoretic induction" we mean that it satisfies the corresponding principle as above, but omitting the condition that g is unique.*

Theorem 4.2 *Assume* Tree *is a set s.t.* tree : F(Tree) → Tree.
The following are equivalent

(a) (Tree, tree) *is an initial* F*-algebra.*

(b) (Tree, tree) *satisfies the principle of unique iteration.*

(c) (Tree, tree) *satisfies the principle of unique primitive recursion.*

(d) (Tree, tree) *satisfies the principle of type theoretic induction.*

(e) (Tree, tree) *satisfies the principle of unique type theoretic induction.*

Proof: (a) and (b) are equivalent since the principle of iteration is nothing but the commutativity of the diagram spelt out.

(a) implies (e): Define for X, f as in the definition of type-theoretic induction

$$X' : \mathrm{I} \to \mathrm{Set}$$
$$X'(i) = (t \in \mathrm{Tree}(i)) \times X(i,t)$$

$$h : \mathrm{F}(X') \to X'$$
$$h(i, \langle a, k\rangle) \quad = \quad \langle \mathrm{tree}(i, a, \pi_0 \circ k), f(i, \langle a, k\rangle)\rangle$$

Consider the diagram

$$\begin{tikzcd}
\mathrm{F}(\mathrm{Tree}) \arrow[r, "\mathrm{tree}"] \arrow[d, "\mathrm{F}(g')"] \arrow[dd, bend right, "{\mathrm{F}(\pi_0 \circ g'), \mathrm{F}(\mathrm{id})}"'] & \mathrm{Tree} \arrow[d, "\exists! g'"'] \arrow[dd, bend left, "{\pi_0 \circ g', \mathrm{id}}"] \\
\mathrm{F}(X') \arrow[r, "h"] \arrow[d, "\mathrm{F}(\pi_0)"] & X' \arrow[d, "\pi_0"'] \\
\mathrm{F}(\mathrm{Tree}) \arrow[r, "\mathrm{tree}"] & \mathrm{Tree}
\end{tikzcd}$$

There exists a unique g' such that the upper part of the diagram commutes. The lower part of the diagram commutes trivially. Both $\pi_0 \circ g'$ and id : Tree → Tree make the outer diagram commute. By uniqueness we get $\pi_0 \circ g' = \mathrm{id}$. Therefore $g'(i,t) = \langle t, g(i,t)\rangle$ for some $g : (i : \mathrm{I}) \to \mathrm{Tree}(i) \to X'(i,t)$, and we see immediately that g satisfies the equations for type-theoretic induction.

Assume g_0 is another solution for the equations for type theoretic induction in the theorem. Let $g'_0 : \mathrm{Tree} \to X'$, $g'_0(i,t) = \langle t, g_0(i,t)\rangle$. Then the upper diagram above with g' replaced by g'_0 commutes as well. By uniqueness of g' it follows $g'_0 = g'$ and therefore $g_0 = g$.

Obviously, (e) implies (d).

(d) implies (c). We immediately obtain (d) implies the principle of primitive recursion, since it is a special case of type theoretic induction. We get as well unique primitive recursion: Assume X, f as in the definition of unique primitive recursion, and let $g, g' : \mathrm{Tree} \to X$ be two solutions for the primitive recursion equation. Let

$$X' : (i \in I) \to \mathrm{Tree}(i) \to \mathrm{Set}$$
$$X'(i,t) = g(i,t) \mathrel{\widehat{=}} g'(i,t)$$

Let f' be of the type of the function underlying the principle of type-theoretic induction w.r.t. X', $f'(i, \langle a, h\rangle) = * \in X'(i,t)$, where $t := \mathrm{tree}(i, a, \pi_0 \circ h)$. $f'(i, \langle a, h\rangle) \in X'(i,t)$, since for $b \in \mathrm{B}(i,a)$ we have, with $j' := \mathrm{j}(i,a,b)$, that $\pi_1(h(b)) \in X'(j', \pi_0(h(b)))$, therefore $g(j', \pi_0(h(b))) = g'(j', \pi_0(h(b)))$, and therefore $g(i,t) = g'(i,t)$. Let g'' be defined by the principle of induction w.r.t. X' and f'. Then we have $g'' : (i \in \mathrm{I}) \to (t \in \mathrm{Tree}(i)) \to X'(i,t)$, and therefore for $i \in \mathrm{I}$, $t \in \mathrm{Tree}(i)$ we have $g(i,t) = g'(i,t)$.

(c) implies (b) since iteration is a special case of primitive recursion.

When dualising inductive definitions, we will not obtain a direct dual of type theoretic induction. We obtain only duals of iteration and recursion. So when dualising the current theorem, we need to omit (d) and (e) and therefore dualise a proof that (a) implies (c). But a proof that (a) implies (c) is essentially the same as the proof that (a) implies (e), where one omits the dependencies of X' on $\mathrm{Tree}(i)$.

5 Modelling Coinductive Sets in Set Theory

In case of inductive definitions it was easy to model inductively defined sets set-theoretically, since we could simply model the well-founded trees set theoretically. When defining coinductively defined sets (or final coalgebras) we obtain non-well-founded trees. If we define the elements as terms introduced by the constructor tree as used before (which was fixed function), then the coinductively defined set would need to be defined as a non-well-founded sets [3]. This can be overcome by introducing coinductively defined sets by their eliminators, and in the following we will give one concrete way of defining them. We can then define a constructor for introducing elements, this constructor is not the function tree defined before. We note that there are many different ways known for defining non-well-founded trees in set theory, our approach here is inspired by Aczel [4]. It can be considered as an indexed explicit version of the standard limit construction of coalgebras. This construction is a category theoretic construction, it is essentially the ω-limit of F^n. One of the earliest versions of such a construction seems to be [5], which is an extension of [3].

One advantage of this concrete representation of coinductively defined sets over other more abstract constructions is that because it is very concrete it is easy to have a feeling of what the elements of the coinductively defined sets are. As one can see the elements of coalgebras are descriptions of the result of applying the eliminators to them (several times in case of the eliminator E_2 which returns an element of the coalgebra). So this construction follows the slogan "an element of a coalgebra is determined by the result of applying the eliminators to it".

Assume $\mathrm{I}, \mathrm{A}, \mathrm{B}, \mathrm{j}$, F as before.

Definition 5.1
- *An* F*-coalgebra* (X, f) *is given by* $X \in \mathrm{Set}^{\mathrm{I}}$ *and* $f : X \to \mathrm{F}(X)$.
- *An* F*-coalgebra* (X, f) *is a final* F*-coalgebra if for any* F*-coalgebra* (Y, g) *there exists a unique* $h : Y \to X$ *s.t.*

$$\begin{tikzcd}
Y \arrow[r, "g"] \arrow[d, "\exists ! h"'] & \mathrm{F}(Y) \arrow[d, "\mathrm{F}(h)"] \\
X \arrow[r, "f"] & \mathrm{F}(X)
\end{tikzcd}$$

We will in the following construct a final F-coalgebra $([\![\,\mathrm{Tree}^{\infty}\,]\!], \mathrm{E})$. So we have

$$\begin{aligned}
\mathrm{E} : {} & (i \in \mathrm{I}) \to [\![\,\mathrm{Tree}^{\infty}\,]\!](i) \\
& \to ((a \in \mathrm{A}(i)) \times ((b \in \mathrm{B}(i, a)) \to [\![\,\mathrm{Tree}^{\infty}\,]\!](\mathrm{j}(i, a, b))))
\end{aligned}$$

We can replace the eliminator (or case distinction) E by two eliminators

$$\begin{aligned}
\mathrm{E}_1 &: (i \in \mathrm{I}) \to [\![\,\mathrm{Tree}^{\infty}\,]\!](i) \to \mathrm{A}(i) \\
\mathrm{E}_2 &: (i \in \mathrm{I}) \to (t \in [\![\,\mathrm{Tree}^{\infty}\,]\!](i)) \to (b \in \mathrm{B}(i, \mathrm{E}_1(i, t))) \to [\![\,\mathrm{Tree}^{\infty}\,]\!](\mathrm{j}(i, \mathrm{E}_1(i, t), b))
\end{aligned}$$

E_1 returns the label of the tree and E_2 its subtrees. Since $\mathrm{E}_1, \mathrm{E}_2$ are the two components of E we will in the following freely switch between E and $\mathrm{E}_1, \mathrm{E}_2$.

We summarise that $[\![\,\mathrm{Tree}^{\infty}\,]\!]$ is an F-coalgebra as follows:

$$\begin{aligned}
[\![\,\mathrm{Tree}^{\infty}\,]\!] &: \mathrm{I} \to \mathrm{Set} \\
\mathrm{E}_1 &: (i \in \mathrm{I}) \to [\![\,\mathrm{Tree}^{\infty}\,]\!](i) \to \mathrm{A}(i) \\
\mathrm{E}_2 &: (i \in \mathrm{I}) \to (b \in \mathrm{B}(i, \mathrm{E}_1(i, t))) \to [\![\,\mathrm{Tree}^{\infty}\,]\!](\mathrm{j}(i, \mathrm{E}_1(i, t), b))
\end{aligned}$$

The idea for defining $[\![\,\mathrm{Tree}^{\infty}\,]\!] : \mathrm{I} \to \mathrm{Set}$ and E_i as follows: An element of $[\![\,\mathrm{Tree}^{\infty}\,]\!]$ is anything which, when applying E_1 and E_2 to it, returns meaningful results. When applying E_1 we obtain an element of $A(i)$, which we can observe directly. When applying E_2 (with an argument in $B(i, a)$) we obtain an element of $[\![\,\mathrm{Tree}^{\infty}\,]\!](j)$ for some j, which we cannot observe directly. However we can continue applying E_2 several times and then E_1 to obtain an observable result. The observations we have are therefore that we apply several times E_2 to it and then E_1 to it and obtain an element of $A(i)$ for some i.

This means that the observations from an element of $[\![\,\mathrm{Tree}^{\infty}\,]\!](i)$ are if we set $i_0 = i$, an element $a_0 \in \mathrm{A}(i_0)$ which would be the result of applying E_1; we can then continue by choosing an arbitrary $b_0 \in \mathrm{B}(i_0, a_0)$, have now a new index $i_1 = \mathrm{j}(i_0, a_0, b_0)$. For this index we could apply E_1 to it and obtain an element $a_1 \in \mathrm{A}(i_1)$, or apply for an arbitrary $b_1 \in \mathrm{B}(i_0, a_0)$, an element corresponding to index $i_2 = \mathrm{j}(i_0, a_0, b_0)$ and so on.

The observations are therefore a set of sequences $\langle i_0, a_0, b_0, i_1, a_1, b_1, \ldots, i_n, a_n\rangle$, where $i_0 = i$, $a_k \in \mathrm{A}(i_k)$, $b_k \in \mathrm{B}(i_k, a_k)$ and $i_{k+1} = \mathrm{j}(i_k, a_k, b_k)$. Here b_k can be chosen freely, whereas a_k is defined uniquely depending on previous occurrences of $b_{k'}$. An element of $[\![\,\mathrm{Tree}^\infty\,]\!]$ is determined by those observations and therefore identified with those observations. This gives rise to the following definition of $[\![\,\mathrm{Tree}^\infty\,]\!]$ as a set of such sequences:

Definition 5.2 *(a) Let for $i \in \mathrm{I}$*

$$\begin{aligned}\mathrm{Seq}_{[\![\,\mathrm{Tree}^\infty\,]\!]}(i) := \{&\langle i_0, a_0, b_0, i_1, a_1, b_1, \ldots, i_n, a_n\rangle \mid \\ &n \geq 0, i_0 = i, \\ &(\forall k \in \{0, \ldots, n-1\}.b_k \in \mathrm{B}(i_k, a_k) \wedge i_{k+1} = \mathrm{j}(i_k, a_k, b_k)), \\ &\forall k \in \{0, \ldots, n\}.a_k \in \mathrm{A}(i_k)\}\end{aligned}$$

(b) Let $[\![\,\mathrm{Tree}^\infty\,]\!](i)$ be the set of $t \subseteq \mathrm{Seq}_{[\![\,\mathrm{Tree}^\infty\,]\!]}(i)$ such that the following holds:

- $\langle i_0, a_0, b_0, \ldots, i_{n+1}, a_{n+1}\rangle \in t \to \langle i_0, a_0, b_0, \ldots, i_n, a_n\rangle \in t$
- $\exists! a.\langle i, a\rangle \in t$*,*
- $\langle i_0, a_0, b_0, \ldots, i_n, a_n\rangle \in t \wedge b_n \in \mathrm{B}(i_n, a_n) \wedge i_{n+1} = \mathrm{j}(i_n, a_n, b_n)$
  $\to \exists! a_{n+1}.\langle i_0, a_0, b_0, \ldots, i_n, a_n, b_n, i_{n+1}, a_{n+1}\rangle \in t$

(c) Define

$$\begin{aligned}&\mathrm{E}_1 : (i \in \mathrm{I}) \to [\![\,\mathrm{Tree}^\infty\,]\!](i) \to \mathrm{A}(i) \\ &\mathrm{E}_1(i, t) := a \qquad \textit{if } \langle i, a\rangle \in t\end{aligned}$$

(d) Define

$$\begin{aligned}&\mathrm{E}_2 : ((i \in \mathrm{I}) \to (t \in [\![\,\mathrm{Tree}^\infty\,]\!](i)) \to (b \in \mathrm{B}(i, \mathrm{E}_1(i, t))) \\ &\quad \to [\![\,\mathrm{Tree}^\infty\,]\!](\mathrm{j}(i, \mathrm{E}_1(i, t), b)) \\ &\mathrm{E}_2(i, t, b) := \{\langle i_1, a_1, b_1, \ldots, i_{n+1}, a_{n+1}\rangle \\ &\qquad\qquad \mid \langle i, \mathrm{E}_1(i, t), b, i_1, a_1, b_1, \ldots, i_{n+1}, a_{n+1}\rangle \in t\}\end{aligned}$$

(e) Define

$$\begin{aligned}&\mathrm{E} : (i \in \mathrm{I}) \to (t \in [\![\,\mathrm{Tree}^\infty\,]\!](i)) \\ &\quad \to ((a \in \mathrm{A}(i)) \times ((b \in \mathrm{B}(i, a)) \to [\![\,\mathrm{Tree}^\infty\,]\!](\mathrm{j}(i, a, b)))) \\ &\mathrm{E}(i, t) = \langle \mathrm{E}_1(i, t), \lambda b \in \mathrm{B}(i, \mathrm{E}_1(i, t)).\mathrm{E}_2(i, t, b)\rangle\end{aligned}$$

Lemma 5.3 $\mathrm{E}_1, \mathrm{E}_2, \mathrm{E}$ *in the previous definition are well defined.*

Proof: Straightforward by definition.

Lemma 5.4 *Assume*

$$\begin{array}{lcl} G & : & \mathrm{I} \to \mathrm{Set} \\ \widehat{a} & : & (i \in \mathrm{I}) \to G(i) \to \mathrm{A}(i) \\ \widehat{g} & : & (i \in \mathrm{I}) \to (g \in G(i)) \to (b \in \mathrm{B}(i, \widehat{a}(i,g))) \to G(\mathrm{j}(i, \widehat{a}(i,g), b)) \end{array}$$

Then there exists a unique $f : (i \in \mathrm{I}) \to G(i) \to [\![\, \mathrm{Tree}^\infty \,]\!](i)$ *such that for all* $i \in \mathrm{I}$*,* $g \in G(i)$*,* $b \in \mathrm{B}(i, \widehat{a}(i,g))$ *we have*

$$\begin{array}{lcl} \mathrm{E}_1(i, f(i,g)) & = & \widehat{a}(i,g) \\ \mathrm{E}_2(i, f(i,g), b) & = & f(\mathrm{j}(i, \mathrm{E}_1(i, f(i,g)), b), \widehat{g}(i,g,b)) \end{array}$$

Proof: Define for $i \in \mathrm{I}$, $g \in G(i)$,

$$\begin{array}{lcl} f(i,g) & = & \{\langle i_0, a_0, b_0, i_1, a_1, b_1, \ldots, i_n, a_n \rangle \mid \\ & & \quad \langle i_0, g_0, a_0, b_0, i_1, a_1, g_1, b_1, \ldots, i_n, g_n, a_n \rangle \in Y(i,g)\} \quad \text{where} \\ Y(i,g) & = & \{\langle i_0, g_0, a_0, b_0, i_1, a_1, g_1, b_1, \ldots, i_n, g_n, a_n \rangle \mid \\ & & \quad g_0 = g, i_0 = i \\ & & \quad (\forall j \in \{0, \ldots, n\}. a_j = \widehat{a}(i_j, g_j)), \\ & & \quad \forall j \in \{0, \ldots, n-1\}. b_j \in B(i_j, a_j) \wedge i_{j+1} = \mathrm{j}(i_j, a_j, b_j) \\ & & \qquad\qquad\qquad\qquad \wedge g_{j+1} = \widehat{g}(i_j, g_j, b_j)\} \end{array}$$

One easily sees that $f(i,g) \in [\![\, \mathrm{Tree}^\infty \,]\!](i)$.
$\langle i, g, \widehat{a}(i,g) \rangle \in Y(i,g)$, therefore $\langle i, \widehat{a}(i,g) \rangle \in f(i,g)$, therefore
$\mathrm{E}_1(i, f(i,g)) = \widehat{a}(i,g)$.
Furthermore,

$$\begin{array}{lcl} \mathrm{E}_2(i, f(i,g), b) & = & \{\langle i_1, a_1, b_1, \ldots, i_n, a_n \rangle \mid \\ & & \quad \langle i, \mathrm{E}_1(i, f(i,g)), b, i_1, a_1, b_1, \ldots, i_n, a_n \rangle \in f(i,g)\} \\ & = & \{\langle i_1, a_1, b_1, \ldots, i_n, a_n \rangle \mid \\ & & \quad \langle i, g, \mathrm{E}_1(i, f(i,g)), b, i_1, g_1, a_1, b_1, \ldots, i_n, g_n, a_n \rangle \in Y(i,g)\} \\ & = & f(\mathrm{j}(i, \mathrm{E}_1(i, f(i,g)), b), \widehat{g}(i,g,b)) \end{array}$$

Therefore f fulfils the required equations.
Assume now

$$\begin{array}{lcl} f' : (i \in \mathrm{I}) \to G(i) \to [\![\, \mathrm{Tree}^\infty \,]\!](i) & & \text{s.t.} \\ \mathrm{E}_1(i, f'(i,g)) & = & \widehat{a}(i,g) \\ \mathrm{E}_2(i, f'(i,g), b) & = & f'(\mathrm{j}(i, \mathrm{E}_1(i, f'(i,g)), b), \widehat{g}(i,g,b)) \end{array}$$

Then

$$\langle i', a' \rangle \in f'(i,g) \quad \Leftrightarrow \quad i' = i \wedge a' = \widehat{a}(i,g) \Leftrightarrow \langle i', a' \rangle \in f(i,g)$$

Therefore sequences of length 2 in $f'(i,g)$ and $f(i,g)$ coincide. Furthermore,

$$\mathrm{E}_2(i, f'(i,g), b) \quad = \quad f'(\mathrm{j}(i, \widehat{a}(i,g), b), \widehat{g}(i,g,b))$$

Therefore

$$\begin{array}{rcl} \mathrm{E}_2(i, f'(i,g), b) & = & \{\langle i_1, a_1, b_1, \ldots, i_n, a_n\rangle \mid \\ & & \quad \langle i, \widehat{a}(i,g), b, i_1, a_1, b_1, \ldots, i_n, a_n\rangle \in f'(i,g)\} \\ & = & f'(\mathrm{j}(i, \widehat{a}(i,g), b), \widehat{g}(i,g,b)) \end{array}$$

which is the same equation as fulfilled by $f(i,g)$. This equation reduces sequences in $f'(i,g)$ of length > 2 to sequences of shorter length in some $f'(i',g')$ for some i', g', similarly for f. Together with the statement about sequences of length 2 above it follows by induction on length(σ)

$$\forall \sigma. \forall i, g. \sigma \in f'(i,g) \Leftrightarrow \sigma \in f(i,g)$$

therefore $\forall i, g. f(i,g) = f'(i,g)$, $f = f'$.

Main Theorem 5.5 ($[\![\,\mathrm{Tree}^\infty\,]\!], \mathrm{E}$) *is a final* F*-coalgebra.*

Proof: ($[\![\,\mathrm{Tree}^\infty\,]\!], \mathrm{E}$) is an F-coalgebra. Assume (G, g) is an F-coalgebra. Let $g(i,x) = \langle \widehat{a}(i,x), \widehat{g}(i,x)\rangle$. Lemma 5.4 implies that there exists a unique $f : G \to [\![\,\mathrm{Tree}^\infty\,]\!]$ s.t. the following diagram commutes:

$$\begin{tikzcd} G \arrow[r, "g"] \arrow[d, "\exists ! f"'] & \mathrm{F}(G) \arrow[d, "\mathrm{F}(f)"] \\ {[\![\,\mathrm{Tree}^\infty\,]\!]} \arrow[r, "\mathrm{tree}"] & \mathrm{F}([\![\,\mathrm{Tree}^\infty\,]\!]) \end{tikzcd}$$

Above we used the notation with keyword data for denoting an inductively defined set as given by its constructors. A similar notation expressing that $[\![\,\mathrm{Tree}^\infty\,]\!]$ is a final coalgebra with eliminators E_1, E_2, would be:

$$\begin{array}{l} \text{coalg } [\![\,\mathrm{Tree}^\infty\,]\!] : \mathrm{I} \to \mathrm{Set} \text{ where} \\ \quad \mathrm{E}_1 \;:\; (i \in \mathrm{I}) \to [\![\,\mathrm{Tree}^\infty\,]\!](i) \to \mathrm{A}(i) \\ \quad \mathrm{E}_2 \;:\; (i \in \mathrm{I}) \to (b \in \mathrm{B}(i, \mathrm{E}_1(i,t))) \to [\![\,\mathrm{Tree}^\infty\,]\!](\mathrm{j}(i, \mathrm{E}_1(i,t), b)) \end{array}$$

6 Coiteration and Corecursion

We will in this section introduce the dual of iteration and primitive recursion, namely coiteration and corecursion. We do not know how to directly formulate the dual of type theoretic induction (or dependent primitive recursion), since one cannot directly invert the arrow in a dependent function type. In Sect. 8 we will introduce a principle of coinduction, which can be considered as the dual of induction, although it is not its direct dual.

We show as well that the principles of being a final F-coalgebra, of unique coiteration, and of unique corecursion are equivalent. The definitions and the proof in this section are the exact dual of Sect. 4 (omitting type theoretic induction). Note that the dual of the product $\times$ is the disjoint union $+$. In the principle of primitive recursion we can make use of both the inductive argument and the recursion hypothesis, corresponding to the product $(\times)$. In the principle of corecursion we can either return a given element from Tree^∞ or recursively call the function in question, which is a call to the corecursion hypothesis, corresponding to the disjoint union $(+)$.

Definition 6.1 *Assume* Tree^∞ *is a set,* $\mathrm{E} : \mathrm{Tree}^\infty \to \mathrm{F}(\mathrm{Tree}^\infty)$*, and let* $\mathrm{E}_1, \mathrm{E}_2$ *be the two components of* E *as defined before.*

(a) By "$(\mathrm{Tree}^\infty, \mathrm{E})$ *satisfies the principle of unique coiteration" we mean the following: Assume*

$$\begin{array}{lll} X & : & \mathrm{I} \to \mathrm{Set} \\ \widehat{a} & : & (i \in \mathrm{I}) \to X(i) \to \mathrm{A}(i) \\ \widehat{x} & : & (i \in \mathrm{I}) \to (x \in X(i)) \to (b \in \mathrm{B}(i, \widehat{a}(i,x))) \\ & & \to X(\mathrm{j}(i, \widehat{a}(i,x), b)) \end{array}$$

Then there exists a unique $f : X \to \mathrm{Tree}^\infty$ *such that*

$$\begin{array}{lll} \mathrm{E}_1(i, f(i,x)) & = & \widehat{\mathrm{a}}(i,x) \\ \mathrm{E}_2(i, f(i,x), b) & = & f(\mathrm{j}(i, \mathrm{E}_1(i, f(i,x)), b), \widehat{\mathrm{x}}(i,x,b)) \end{array}$$

(b) By "$(\mathrm{Tree}^\infty, \mathrm{E})$ *satisfies the principle of unique corecursion" we mean the following: Assume*

$$\begin{array}{lll} \widehat{a} & : & (i \in \mathrm{I}) \to X(i) \to \mathrm{A}(i) \\ \widehat{x} & : & (i \in \mathrm{I}) \to (x \in X(i)) \to (b \in \mathrm{B}(i, \widehat{a}(i,x))) \\ & & \to X(\mathrm{j}(i, \widehat{a}(i,x), b)) + \mathrm{Tree}^\infty(\mathrm{j}(i, \widehat{a}(i,x), b)) \end{array}$$

Then there exists a unique $f : X \to \mathrm{Tree}^\infty$ *such that*

$$\begin{array}{lll} \mathrm{E}_1(i, f(i,x)) & = & \widehat{\mathrm{a}}(i,x) \\ \mathrm{E}_2(i, f(i,x), b) & = & \left\{ \begin{array}{ll} f(\mathrm{j}(i, \mathrm{E}_1(i, f(i,x)), b), x') & \textit{if } \widehat{\mathrm{x}}(i,x,b) = \mathrm{inl}(x') \\ x' & \textit{if } \widehat{\mathrm{x}}(i,x,b) = \mathrm{inr}(x') \end{array} \right. \end{array}$$

(c) By "$(\mathrm{Tree}^\infty, \mathrm{E})$ *satisfies the principle of corecursion or coiteration" we mean that it fills the corresponding principle as above, but omitting the condition that* f *is unique.*

Lemma 6.2 *Assume* Tree^∞ *is a set,* $\mathrm{E} : \mathrm{Tree}^\infty \to \mathrm{F}(\mathrm{Tree}^\infty)$*, and let* $\mathrm{E}_1, \mathrm{E}_2$ *be the two components of* E *as defined before.*

The following are equivalent

(a) $(\mathrm{Tree}^\infty, \mathrm{E})$ *is a final* F*-coalgebra.*

(b) $(\mathrm{Tree}^\infty, \mathrm{E})$ *satisfies the principle of unique coiteration.*

(c) $(\mathrm{Tree}^\infty, \mathrm{E})$ *satisfies the principle of unique corecursion.*

Proof: (a) and (b) are equivalent since $\widehat{a}, \widehat{x}$ are the two components of a morphism $g : X \to \mathrm{F}(X)$, so the unique existence of f as in (b) is equivalent to the unique existence of f in the diagram for defining final coalgebras.

Obviously, (c) implies (b), since coiteration is a special case of corecursion.

(a) implies (c): Define for X, $\widehat{a}$, $\widehat{x}$ as in the definition of corecursion

$$\begin{array}{l}
X' : \mathrm{I} \to \mathrm{Set} \\
X'(i) = X(i) + \mathrm{Tree}^\infty(i)
\end{array}$$

$$\begin{array}{lcl}
\multicolumn{3}{l}{h : (i \in \mathrm{I}) \to X'(i) \to \mathrm{F}(X', i)} \\
h(i, \mathrm{inl}(x)) & = & \langle \widehat{a}(i, x), \lambda b.\widehat{x}(i, x, b)\rangle \\
h(i, \mathrm{inr}(t)) & = & \langle \mathrm{E}_0(i, t), \lambda b.\mathrm{inr}(\mathrm{E}_1(i, t, b))\rangle
\end{array}$$

Consider the diagram

$$\begin{tikzcd}
\mathrm{Tree}^\infty \arrow[r, "\mathrm{E}"] \arrow[d, "\mathrm{inr}"] \arrow[dd, bend right=60, "g' \circ \mathrm{inr}, \mathrm{id}"'] & \mathrm{F}(\mathrm{Tree}^\infty) \arrow[d, "\mathrm{F}(\mathrm{inr})"'] \arrow[dd, bend left=60, "\mathrm{F}(g' \circ \mathrm{inr}), \mathrm{F}(\mathrm{id})"] \\
X' \arrow[r, "h"] \arrow[d, "\exists ! g'"] & \mathrm{F}(X') \arrow[d, "\mathrm{F}(g')"'] \\
\mathrm{Tree}^\infty \arrow[r, "\mathrm{E}"] & \mathrm{F}(\mathrm{Tree}^\infty)
\end{tikzcd}$$

There exists a unique g' such that the lower part of the diagram commutes. The upper part of the diagram commutes trivially. Both $g' \circ \mathrm{inr}$ and $\mathrm{id} : \mathrm{Tree}^\infty \to \mathrm{Tree}^\infty$ make the outer diagram commute. By uniqueness we get $g' \circ \mathrm{inr} = \mathrm{id}$. Let $g := g' \circ \mathrm{inl} : X \to \mathrm{Tree}^\infty$. By $g'(\mathrm{inr}(x)) = x$ we have that g satisfies the desired equation. Assume g_0 is another solution for the corecursion equation in the lemma. Let $g_0' : X' \to \mathrm{Tree}^\infty$, $g_0'(i, \mathrm{inl}(x)) = g_0(x)$, $g_0'(i, \mathrm{inr}(x)) = x$. Then the lower diagram above with g' replaced by g_0' commutes as well. By uniqueness of g' follows $g_0' = g'$ and therefore $g_0 = g$.

7 Indexed Corecursion

When defining elements of coinductively defined sets, we often want to define for some $X \in \mathrm{Set}$ and $\widehat{i} : X \to \mathrm{I}$ a function $f : (x \in X) \to \mathrm{Tree}^\infty(\widehat{i}(x))$ corecursively. This can be reduced to corecursion as follows:

Lemma 7.1 *Let* $(\mathrm{Tree}^\infty, \mathrm{E})$ *be a final* F*-coalgebra, where* F *is as before. Assume*

$$\begin{array}{lll} X \in \mathrm{Set} & & \\ \widehat{i} & : & X \to \mathrm{I} \\ \widehat{a} & : & (x \in X) \to \mathrm{A}(\widehat{i}(x)) \\ \widehat{x} & : & (x \in X) \to (b \in \mathrm{B}(\widehat{i}(x), \widehat{a}(x))) \\ & & \to \{x \in X \mid \widehat{i}(x) = \mathrm{j}(\widehat{i}(x), \widehat{a}(x), b)\} + \mathrm{Tree}^\infty(\mathrm{j}(\widehat{i}(x), \widehat{a}(x), b))) \end{array}$$

Then there exists a unique $f : (x \in X) \to \mathrm{Tree}^\infty(\widehat{i}(x))$*, such that*

$$\begin{array}{l} \mathrm{E}_1(\widehat{i}(x), f(x)) = \widehat{a}(x) \\ \mathrm{E}_2(\widehat{i}(x), f(x), b) = \begin{cases} f(y) & \textit{if } \widehat{x}(x,b) = \mathrm{inl}(y) \\ t & \textit{if } \widehat{x}(x,b) = \mathrm{inr}(t) \end{cases} \end{array}$$

Proof: Let $Y : \mathrm{I} \to \mathrm{Set}$, $Y(i) := \{x \in X \mid \widehat{i}(x) = i\}$. f satisfying the equations as stated in the lemma is equivalent to the function

$$\begin{array}{l} f' : (i \in \mathrm{I}) \to Y(i) \to \mathrm{Tree}^\infty(i) \\ f'(i, x) = f(x) \end{array}$$

satisfying the equations.

$$\begin{array}{l} \mathrm{E}_1(i, f'(i,x)) = \widehat{a}(x) \\ \mathrm{E}_2(i, f'(i,x), b) = \begin{cases} f'(\mathrm{j}(i, \widehat{a}(x), b), y) & \text{if } \widehat{x}(x,b) = \mathrm{inl}(y) \\ t & \text{if } \widehat{x}(x,b) = \mathrm{inr}(t) \end{cases} \end{array}$$

By existence and uniqueness of f' satisfying those equations follows existence and uniqueness of f.

8 Bisimulation and Coinduction

Definition 8.1 *Assume* Tree^∞ *is a set,* $\mathrm{E} : \mathrm{Tree}^\infty \to \mathrm{F}(\mathrm{Tree}^\infty)$*, and let* $\mathrm{E}_1, \mathrm{E}_2$ *be the two components of* E *as defined before.*

(a) Let for $i \in \mathrm{I}$*,* $t, t' \in \mathrm{Tree}^\infty(i)$

$\mathrm{I}^{\mathrm{Bisim}} := (i \in \mathrm{I}) \times \mathrm{Tree}^\infty(i) \times \mathrm{Tree}^\infty(i)$

$\mathrm{A}^{\mathrm{Bisim}} : \mathrm{I}^{\mathrm{Bisim}} \to \mathrm{Set}$
$\mathrm{A}^{\mathrm{Bisim}}(i, t, t') := (\mathrm{E}_0(i,t) = \mathrm{E}_0(i,t'))$ *(more precisely* $(\mathrm{E}_0(i,t) \mathrel{\widehat{=}} \mathrm{E}_0(i,t'))$*)*

$\mathrm{B}^{\mathrm{Bisim}} : (i \in \mathrm{I}^{\mathrm{Bisim}}) \to \mathrm{A}^{\mathrm{Bisim}}(i) \to \mathrm{Set}$
$\mathrm{B}^{\mathrm{Bisim}}(i, t, t', a) := \mathrm{B}(i, a)$

$\mathrm{j}^{\mathrm{Bisim}} : (i \in \mathrm{I}^{\mathrm{Bisim}}) \to (a \in \mathrm{A}^{\mathrm{Bisim}}(i)) \to (b \in \mathrm{B}^{\mathrm{Bisim}}(i, a)) \to \mathrm{I}^{\mathrm{Bisim}}$
$\mathrm{j}^{\mathrm{Bisim}}(i, t, t', *, b) = \langle \mathrm{j}(i, \mathrm{E}_0(i,t), b), \mathrm{E}_1(i,t,b), \mathrm{E}_1(i,t',b) \rangle$

$\mathrm{F}^{\mathrm{Bisim}} : \mathrm{Set}^{\mathrm{I}^{\mathrm{Bisim}}} \to \mathrm{Set}^{\mathrm{I}^{\mathrm{Bisim}}}$
$\mathrm{F}^{\mathrm{Bisim}}(X, i) = (a \in \mathrm{A}^{\mathrm{Bisim}}(i)) \times ((b \in \mathrm{B}^{\mathrm{Bisim}}(i,a)) \to X(\mathrm{j}^{\mathrm{Bisim}}(i, a, b))$

We note that, if $(\mathrm{Bisim}, \mathrm{E}^{\mathrm{Bisim}})$ is an $\mathrm{F}^{\mathrm{Bisim}}$-coalgebra, and $\mathrm{E}_0^{\mathrm{Bisim}}, \mathrm{E}_1^{\mathrm{Bisim}}$ are the two components of $\mathrm{E}^{\mathrm{Bisim}}$, then

$\mathrm{E}_0^{\mathrm{Bisim}}(i,t,t') \in (\mathrm{E}_0(i,t) = \mathrm{E}_0(i,t'))$
i.e. the existence of $\mathrm{E}_0^{\mathrm{Bisim}}(i,t,t')$ is equivalent to $\mathrm{E}_0(i,t) = \mathrm{E}_0(i,t')$
and for $a \in \mathrm{A}^{\mathrm{Bisim}}(i,t,t')$, $b \in \mathrm{B}^{\mathrm{Bisim}}(i,t,t',a) = \mathrm{B}(i,a)$
$\mathrm{E}_1^{\mathrm{Bisim}}(i,t,t',b) \in \mathrm{Bisim}(\mathrm{j}(i, \mathrm{E}_0(i,t), b), \mathrm{E}_1(i,t,b), \mathrm{E}_1(i,t',b))$

Definition 8.2 *For $X \in \mathrm{Set}$ which is considered as a relation we will in formulae write X instead of $(\exists x. x \in X)$*

Lemma 8.3 *Assume the axiom of choice. Assume $X : \mathrm{I}^{\mathrm{Bisim}} \to \mathrm{Set}$.*
There exists a g s.t. (X, g) is an $\mathrm{F}^{\mathrm{Bisim}}$-coalgebra iff

$$\begin{aligned}&\forall i,t,t'.X(i,t,t')\\ &\qquad \to \mathrm{E}_0(i,t) = \mathrm{E}_0(i,t')\\ &\qquad\quad \wedge \forall b \in \mathrm{B}(i, \mathrm{E}_0(i,t)).X(\mathrm{j}(i,\mathrm{E}_0(i,t),b), \mathrm{E}_1(i,t,b), \mathrm{E}_1(i,t',b))\end{aligned}$$

Proof "$\Rightarrow$" is obvious. For "$\Leftarrow$" define $g : X \to \mathrm{F}^{\mathrm{Bisim}}(X)$, $g(i,t,t',x) = \langle *, h\rangle$ where $h(b) = \text{some } y \in X(\mathrm{j}(i,\mathrm{E}_0(i,t),b), \mathrm{E}_1(i,t,b), \mathrm{E}_1(i,t',b))$.

Induction is a proof principle which is equivalent to the principle that an F-algebra is an initial F-algebra, or, as we have seen, the principle of unique iteration or unique primitive recursion. Dually coinduction should be a proof principle which is equivalent to the principle that an F-coalgebra is a final F-coalgebra, or equivalently, that it satisfies the principle of unique coiteration or the principle of unique corecursion.

We will see below that the principle of being a final F-coalgebra is equivalent to the fact that bisimulation implies equality. The latter is a proof principle. As it stands it does not seem to be of the same character as the principle of induction as a proof principle. However, bisimulation is a coalgebra, and proofs of bisimulation can therefore be carried out corecursively, and that will give rise to the dual of an induction hypothesis, namely a coinduction hypothesis. This way we obtain proof principle which we believe is of similar character as induction. We will elaborate this in Subsect. 9.3 where we will introduce schemata for coinduction.

We therefore call the fact that bisimulation implies equality the principle of coinduction:

Definition 8.4 *Let $(\mathrm{Tree}^\infty, \mathrm{E})$ be an F-coalgebra.*
By "$(\mathrm{Tree}^\infty, \mathrm{E})$ satisfies the principle of coinduction" we mean that it satisfies the principle of corecursion and for the final $\mathrm{F}^{\mathrm{Bisim}}$-coalgebra $(\mathrm{Bisim}, \mathrm{E}')$ we have

$$\forall i,t,t'.\mathrm{Bisim}(i,t,t') \to t = t$$

Remark 8.5 *Note that since proofs by bisimulation can be carried out by corecursion on $\mathrm{Bisim}(i,t,t')$ the principle of coinduction becomes a proper proof principle.*

Lemma 8.6 *Let* $(\mathrm{Tree}^\infty, \mathrm{E})$ *be an* F*-coalgebra. The following is equivalent*

(i) $(\mathrm{Tree}^\infty, \mathrm{E})$ *is a final* F*-coalgebra.*

(ii) $(\mathrm{Tree}^\infty, \mathrm{E})$ *satisfies the principle of corecursion and for any* $\mathrm{F}^{\mathrm{Bisim}}$*-coalgebra* (X, h) *we have*
$\forall i, t, t'.X(i, t, t') \to t = t'$.

(iii) $(\mathrm{Tree}^\infty, \mathrm{E})$ *satisfies the principle of coinduction.*

Proof: By Lemma 6.2, in (i) - (iii) the principle of corecursion is satisfied.

(i) implies (ii): Assume (X, h) is a $\mathrm{F}^{\mathrm{Bisim}}$-coalgebra.
Let

$$G : \mathrm{I} \to \mathrm{Set}$$
$$G(i) := \{\langle t, t'\rangle \in \mathrm{Tree}^\infty \times \mathrm{Tree}^\infty \mid X(i, t, t')\}$$

Define

$$g : G \to \mathrm{F}(G)$$
$$g(i, \langle t, t'\rangle) = \langle \mathrm{E}_1(i, t), \lambda b.\langle \mathrm{E}_1(i, t, b), \mathrm{E}_1(i, t', b)\rangle\rangle\}$$

Consider

$$\begin{tikzcd}
G \arrow[r, "g"] \arrow[d, "\exists! h"'] & \mathrm{F}(G) \arrow[d, "\mathrm{F}(h)"] \\
\mathrm{Tree}^\infty \arrow[r, "\mathrm{E}"] & \mathrm{F}(\mathrm{Tree}^\infty)
\end{tikzcd}$$

There exists a unique h which makes this diagram commute. Both the first and second projection (lifted to $\mathrm{Set}^{\mathrm{I}}$) make this diagram commute. By uniqueness follows they are equal and therefore the assertion follows.

(ii) implies (iii) is obvious since by the previous section there exist such a coalgebra.

(iii) implies (ii) since for any $\mathrm{F}^{\mathrm{Bisim}}$-coalgebra (X, h) we obtain a function $f : X \to \mathrm{Bisim}$. Therefore that $X(i, t, t')$ is inhabited implies that $\mathrm{Bisim}(i, t, t')$ is inhabited.

(ii) implies (i): Let (X, g) be an F-coalgebra and assume h, h' are two solutions which make the following diagram commute:

$$\begin{tikzcd}
X \arrow[r, "g"] \arrow[d, shift right, "h"'] \arrow[d, shift left, "h'"] & \mathrm{F}(X) \arrow[d, shift right, "\mathrm{F}(h)"'] \arrow[d, shift left, "\mathrm{F}(h')"] \\
\mathrm{Tree}^\infty \arrow[r, "\mathrm{E}"] & \mathrm{F}(\mathrm{Tree}^\infty)
\end{tikzcd}$$

Let $g(i,x) = \langle \widehat{a}(i,x), \lambda b.\widehat{x}(i,x,b)\rangle$.
Let

$$\begin{array}{l} H : (i \in \mathrm{I}) \to \mathrm{Tree}^{\infty}(i) \to \mathrm{Tree}^{\infty}(i) \to \mathrm{Set} \\ H(i,t,t') = \{x \in X(i) \mid t = h(i,x), t' = h'(i,x)\} \end{array}$$

It follows for $i \in \mathrm{I}$, $x \in X(i)$ and therefore $x \in H(i, h(i,x), h'(i,x))$ that

$$\mathrm{E}_0(i, h(i,x)) = \pi_0(\mathrm{F}(h)(i, g(i,x))) = \pi_0(g(i,x)) = \mathrm{E}_0(i, h'(i,x))$$

and for $b \in \mathrm{B}(i, \mathrm{E}_0(i, h(i,x)))$

$$\begin{array}{rl} \mathrm{E}_1(i, h(i,x), b) = & \pi_1(\mathrm{F}(h)(i, g(i,x)))(b) \\ = & h(\mathrm{j}(i, \pi_0(g(i,x)), b), \pi_1(g(i,x))(b)) \\ = & h(\mathrm{j}(\cdots), \widehat{x}(i,x,b)) \\ \multicolumn{2}{l}{\mathrm{E}_1(i, h'(i,x), b) = h'(\mathrm{j}(\cdots), \widehat{x}(i,x,b))} \\ \multicolumn{2}{l}{\widehat{\mathrm{x}}(i,x,b) \in H(\mathrm{j}(\cdots), \mathrm{E}_1(i, h(i,x), b), \mathrm{E}_1(i, h'(i,x), b))} \\ \multicolumn{2}{l}{\langle *, \lambda b.\widehat{\mathrm{x}}(i,x,b)\rangle \in \mathrm{F}^{\mathrm{Bisim}}(H)(i,x,b)} \\ \multicolumn{2}{l}{\lambda i, x.\langle *, \lambda b.\widehat{\mathrm{x}}(i,x,b)\rangle \in H \to \mathrm{F}^{\mathrm{Bisim}}(H)} \end{array}$$

Therefore (H, h) is an $\mathrm{F}^{\mathrm{Bisim}}$-coalgebra, $H(i,t,t')$ inhabited implies $t = t'$, and therefore $\forall x \in X(i).h(i,x) = h'(i,x)$.

Main Theorem 8.7 *Assume* Tree^{∞} *is a set,* $\mathrm{E} : \mathrm{Tree}^{\infty} \to \mathrm{F}(\mathrm{Tree}^{\infty})$*, and let* $\mathrm{E}_1, \mathrm{E}_2$ *be the two components of* E *as defined before.*
The following are equivalent

(a) $(\mathrm{Tree}^{\infty}, \mathrm{E})$ *is a final* F*-coalgebra.*

(b) $(\mathrm{Tree}^{\infty}, \mathrm{E})$ *satisfies the principle of unique coiteration.*

(c) $(\mathrm{Tree}^{\infty}, \mathrm{E})$ *satisfies the principle of unique corecursion.*

(d) $(\mathrm{Tree}^{\infty}, \mathrm{E})$ *satisfies the principle of coinduction.*

Proof: By Lemmata 6.2 and 8.6.

9 Schemata for Corecursive Definitions and Coinductive Proofs

9.1 Schema for Corecursion

By Lemma 6.2 we can introduce elements of the coinductively defined set (final F-coalgebra) $([\![\mathrm{Tree}^{\infty}]\!], \mathrm{E})$ as follows:

Assume $A : \mathrm{I} \to \mathrm{Set}$, $[\![\, \mathrm{Tree}^\infty \,]\!], \mathrm{E}_1, \mathrm{E}_2$ as before. We can define a function

$$f : (i \in \mathrm{I}) \to X(i) \to [\![\, \mathrm{Tree}^\infty \,]\!](i)$$

corecursively by defining for $i \in \mathrm{I}$, $x \in X(i)$

- a value $a' := \mathrm{E}_1(i, f(i,x)) \in \mathrm{A}(i)$
- and for $b \in \mathrm{B}(i,a)$ a value $\mathrm{E}_2(i, f(i,x), b) \in [\![\, \mathrm{Tree}^\infty \,]\!](i', b)$
  where $i' := \mathrm{j}(i, a', b)$
  and we can define $\mathrm{E}_2(i, f(i,x), b)$
  - as an element of $[\![\, \mathrm{Tree}^\infty \,]\!](i')$ defined before
  - or corecursively define $\mathrm{E}_2(i, f(i,x), b) = f(i', x')$
    for some $x' \in X(i')$.
    Here, $f(i', x')$ will be called the corecursion hypothesis.

As a simple example we consider Streams. Streams are the final F-coalgebra on Set with $\mathrm{F}(X) = \mathbb{N} \times X$. So we have $\mathrm{I} = \mathbf{1}$, $\mathrm{A}(*) = \mathbb{N}$, $\mathrm{B}(*, x) = \mathbf{1}$. Omitting the arguments in $\mathbf{1}$ we obtain $\mathrm{F}(X)$ as above. Let $(\mathrm{Stream}, \mathrm{E})$ be the final F-coalgebra, and let head, tail be the two components of E. Then we get

$$\begin{array}{lcl} \mathrm{head} & : & \mathrm{Stream} \to \mathbb{N} \\ \mathrm{tail} & : & \mathrm{Stream} \to \mathrm{Stream} \end{array}$$

The above schema is instantiated as follows:

Let $A \in \mathrm{Set}$. We can define

$$f : A \to \mathrm{Stream}$$

corecursively by defining for $a \in A$

- $\mathrm{head}(f(a)) \in \mathbb{N}$ and
- $\mathrm{tail}(f(a)) \in \mathrm{Stream}$,
  where for defining $\mathrm{tail}(f(a))$ we can
  - either return an element of Stream defined before or
  - corecursively define $\mathrm{tail}(f(a)) = f(a')$ for some $a' \in A$.
    Here, $f(a')$ will be called the corecursion hypothesis.

So we can for instance define by corecursion

$$\begin{array}{lcl} s \in \mathrm{Stream} & & \text{s.t.} \\ \mathrm{head}(s) & = & 0 \\ \mathrm{tail}(s) & = & s \end{array}$$

(Here, $A = \mathbf{1}$, and we omit the argument in A.) Or we define

$$\begin{array}{lcl} s' : \mathbb{N} \to \text{Stream} & & \text{s.t.} \\ \text{head}(s'(n)) & = & 0 \\ \text{tail}(s'(n)) & = & s'(n+1) \end{array}$$

or define

$$\begin{array}{lcl} \text{cons} : \mathbb{N} \to \text{Stream} \to \text{Stream} & & \text{s.t.} \\ \text{head}(\text{cons}(n,s)) & = & n \\ \text{tail}(\text{cons}(n,s)) & = & s \end{array}$$

(Here, $A = \mathbb{N} \times \text{Stream}$, and we curried the function.)

9.2 Schema for corecursively defined indexed functions

By Lemma 7.1 we have the following schema:

Assume $X \in \text{Set}$, $\widehat{j} : X \to \text{I}$.
We can define

$$f : (x \in X) \to [\![\text{Tree}^\infty]\!](\widehat{i}(x))$$

corecursively by determining for $x \in X$ with $i := \widehat{j}(x)$,

- $a := \text{E}_1(i, f(x)) \in \text{A}(i)$
- and for $b \in \text{B}(i,a)$ with $i' := \text{j}(i,a,b)$ the value
  $\text{E}_2(i, f(x), b) \in [\![\text{Tree}^\infty]\!](i')$
  where we can define $\text{E}_2(i, f(x), b)$ as
  - a previously defined value of $[\![\text{Tree}^\infty]\!](i')$
  - or corecursively define $\text{E}_2(i, f(x), b) = f(x')$ for some x' such that $\widehat{i}(x') = i'$.
    $f(x')$ will be called the corecursion hypothesis.

As an example consider the coinductively defined set of stacks of a certain height, $\text{Stack} : \mathbb{N} \to \text{Set}$ with destructors

$$\begin{array}{lcl} \text{top} & : & (n \in \mathbb{N}) \to (n > 0) \to \text{Stack}(n) \to \mathbb{N} \\ \text{pop} & : & (n \in \mathbb{N}) \to (n > 0) \to \text{Stack}(n) \to \text{Stack}(n-1) \end{array}$$

We can define empty : Stack(0), where we do not need to define anything since $(0 \widehat{>} 0) = \emptyset$. Furthermore, we can define

$$\begin{array}{lcl} \text{push} : (n \in \mathbb{N}) \to \text{Stack}(n) \to \text{Stack}(n+1) & & \text{s.t.} \\ \text{top}(n+1, *, \text{push}(n,s)) & = & n \\ \text{pop}(n+1, *, \text{push}(n,s)) & = & s \end{array}$$

More complicated examples of indexed coinductively defined sets are state-dependent interactive programs, see [17, 16, 15, 18], or bisimulation relations as defined below.

9.3 Schema for Coinduction

When proving that elements of a coinductively defined set are bisimilar one usually defines certain elements which should be shown to be bisimilar simultaneously. This amounts to having

$$\begin{array}{lcl} J & : & \mathrm{Set} \\ \widehat{i} & : & J \to \mathrm{I} \\ x_0, x_1 & : & (j \in J) \to [\![\,\mathrm{Tree}^{\infty}\,]\!](\widehat{i}(j)) \end{array}$$

and showing $\forall j \in J.x_0(j) = x_1(j)$ by proving $\forall j \in J.\mathrm{Bisim}(\widehat{i}(j), x_0(j), x_1(j))$.

Using the same method as in the previous subsection, and using the fact that $b \in \mathrm{Bisim}(i, x, x')$ implies equality, we can show this statement by coinduction by showing the following:

$$\begin{array}{l} \forall j \in J.\mathrm{E}_0(\widehat{i}(j), x_0(j)) = \mathrm{E}_0(\widehat{i}(j), x_1(j)) \wedge \\ \qquad \forall b \in \mathrm{B}(i, \mathrm{E}_0(\widehat{i}(j), x_0(j))).\mathrm{E}_1(\widehat{i}(j), x_0(j), b) = \mathrm{E}_1(\widehat{i}(j), x_1(j), b) \vee \\ \qquad\qquad\qquad\qquad \exists j'.\widehat{i}(j') = \mathrm{j}(\widehat{i}(j), \mathrm{E}_0(\widehat{i}(j), x_0(j), b)) \wedge \\ \qquad\qquad\qquad\qquad\quad x_0(j') = \mathrm{E}_0(\widehat{i}(j), x_0(j), b) \wedge \\ \qquad\qquad\qquad\qquad\quad x_1(j') = \mathrm{E}_0(\widehat{i}(j), x_1(j), b) \end{array}$$

This means that we have the following principle of coinductive proofs:

Assume

$$\begin{array}{lcl} J & : & \mathrm{Set} \\ \widehat{i} & : & J \to \mathrm{I} \\ x_0, x_1 & : & (j \in J) \to [\![\,\mathrm{Tree}^{\infty}\,]\!](\widehat{i}(j)) \end{array}$$

We can show $\forall j \in J.x_0(j) = x_0(j')$ coinductively by showing

- $\mathrm{E}_0(\widehat{i}(j), x_0(j))$ and $\mathrm{E}_0(\widehat{i}(j), x_1(j))$ are equal
- and for all b that
  $\mathrm{E}_1(\widehat{i}(j), x_0(j), b)$ and $\mathrm{E}_1(\widehat{i}(j), x_0(j), b)$ are equal,
  where we can use either the fact that
  - this was shown before,
  - or we can use the coinduction-hypothesis, which means using the fact
    $\mathrm{E}_1(\widehat{i}(j), x_0(j), b) = x_0(j')$ and $\mathrm{E}_1(\widehat{i}(j), x_1(j), b) = x_1(j')$ for some $j' \in J$.

Examples of proofs by coinduction in the example of streams with s, s', cons are as follows:

- We show $\forall n \in \mathbb{N}.s = s'(n)$ by coinduction: For using the schema above we have $J = \mathbb{N}$, $x_0(j) = s$, $x_1(j) = s'(n)$. The argument is as follows:

We show $\forall n \in \mathbb{N}.s = s'(n)$. Assume $n \in \mathbb{N}$. $\mathrm{head}(s) = \mathrm{head}(s'(n))$ and $\mathrm{tail}(s) = s = s'(n+1) = \mathrm{tail}(s'(n))$, where $s = s'(n+1)$ follows by the coinduction hypothesis.

- We show $\mathrm{cons}(0, s) = s$ by coinduction:
  $\mathrm{head}(\mathrm{cons}(0, s)) = 0 = \mathrm{head}(s)$ and $\mathrm{tail}(\mathrm{cons}(0, s)) = s = \mathrm{tail}(s)$, where we did not use the coinduction hypothesis.

9.4 Schema for Coinductively defined Relations

The previous example can be generalised to arbitrary coinductively defined sets relating elements of an indexed set. A typical example would be the bisimulation relation on a labelled transition system, which we consider below. Assume $\mathrm{I} \in \mathrm{Set}$, $D : \mathrm{I} \to \mathrm{Set}$ (not necessarily a coinductively defined set). Let

$$\mathrm{I}^+ := (i \in \mathrm{I}) \times D(i) \times D(i)$$

Assume

$$\begin{array}{lll}
A & : & (i \in \mathrm{I}) \to D(i) \to D(i) \to \mathrm{Set} \\
B & : & (i \in \mathrm{I}) \to (d \in D(i)) \to (d' \in D(i)) \to A(i, d, d') \to \mathrm{Set} \\
j & : & (i \in \mathrm{I}) \to (d \in D(i)) \to (d' \in D(i)) \to (a \in A(i, d, d')) \to B(i, d, d', a) \\
 & & \to \mathrm{I} \\
d_0, d_1 & : & (i \in \mathrm{I}) \to (d \in D(i)) \to (d' \in D(i)) \to (a \in A(i, d, d')) \to B(i, d, d', a) \\
 & & \to D(j(i, d, d', a))
\end{array}$$

Define

$$\begin{array}{l}
\mathrm{F} : \mathrm{Set}^{\mathrm{I}^+} \to \mathrm{Set}^{\mathrm{I}^+} \\
\mathrm{F}(X, i, d, d') = (a \in A(i, d, d')) \\
\qquad\qquad\qquad \times((b \in B(i, d, d', a)) \\
\qquad\qquad\qquad\quad \to X(j(i, d, d', a, b), d_0(i, d, d', a, b), d_1(i, d, d', a, b)))
\end{array}$$

Let $(\widehat{\mathrm{B}}, \mathrm{E})$ be the final F-coalgebra, $\mathrm{E}_1, \mathrm{E}_2$ be the two components of E. Assume

$$\begin{array}{lll}
\widehat{J} & : & \mathrm{Set} \\
\widehat{j} & : & \widehat{J} \to \mathrm{I} \\
\widehat{d}_0, \widehat{d}_1 & : & (j \in \widehat{J}) \to D(\widehat{j}(j))
\end{array}$$

A schema of corecursion (which may be called, if $\widehat{\mathrm{B}}$ is a bisimulation relation and therefore an equality like relation as a coinduction principle) is as follows:

In the above situation we can define a function

$$\widehat{\mathrm{b}} : (j \in J) \to \widehat{\mathrm{B}}(\widehat{j}(j), \widehat{d_0}(j), \widehat{d_1}(j))$$

coinductively by determining for $j \in J$

- an element $\widehat{\mathrm{a}}(j) \in A(\widehat{j}(j), \widehat{d_0}(j), \widehat{d_1}(j))$,
- and for $b \in B(\widehat{j}(j), \widehat{d_0}(j), \widehat{d_1}(j), \widehat{\mathrm{a}}(j))$
  with $i' := j(\widehat{\mathrm{j}}(j), \widehat{d_0}(j), \widehat{d_1}(j), \widehat{\mathrm{a}}(j), b)$,
  $d_i' := d_i(\widehat{\mathrm{j}}(j), \widehat{d_0}(j), \widehat{d_1}(j), \widehat{\mathrm{a}}(j), b)$,
  an element $\widehat{\mathrm{b}}' \in \widehat{\mathrm{B}}(i', d_0', d_1')$,
  where for defining $\widehat{\mathrm{b}}'$ we can use
  - an existing element of $\widehat{\mathrm{B}}(i', d_0', d_1')$
  - or corecursively define $\widehat{\mathrm{b}}' = \widehat{\mathrm{b}}(j')$ for some j'
    such that $\widehat{j}(j') = i'$, $\widehat{d_0}(j') = d_0'$, $\widehat{d_1}(j') = d_1'$.
    $\widehat{\mathrm{b}}(j')$ will be called the corecursion-hypothesis.

As an example we consider bisimulation for a labelled transition system. A labelled transition system is given by set of states S, a set of labels L a relation $\longrightarrow \subseteq \mathrm{S} \times \mathrm{L} \times \mathrm{S}$ where we write $s \xrightarrow{l} s'$ for $\langle s, l, s' \rangle \in \longrightarrow$. Bisimulation $\mathrm{Bisim}(s, s')$ in a transition system can be given by the coalgebraically defined relation $\mathrm{Bisim}(s, s')$ for the eliminators

$$\begin{array}{lcl}
\mathrm{E}_1 & : & (s, s' \in \mathrm{S}) \to \mathrm{Bisim}(s, s') \to (l \in \mathrm{L}) \to (s_0 \in \{s_0 \in \mathrm{S} \mid s \xrightarrow{l} s_0\}) \\
 & & \to ((s_0' \in \{s_0' \in \mathrm{S} \mid s' \xrightarrow{l} s_0'\}) \times \mathrm{Bisim}(s_0, s_0')) \\
\mathrm{E}_2 & : & (s, s' \in \mathrm{S}) \to \mathrm{Bisim}(s, s') \to (l \in \mathrm{L}) \to (s_0' \in \{s_0' \in \mathrm{S} \mid s' \xrightarrow{l} s_0'\}) \\
 & & \to ((s_0 \in \{s_0 \in \mathrm{S} \mid s \xrightarrow{l} s_0\}) \times \mathrm{Bisim}(s_0, s_0'))
\end{array}$$

The existence of E_1 and E_2 is equivalent to

$$\begin{array}{l}
\forall s, s' \in \mathrm{S}.\mathrm{Bisim}(s, s') \to \forall l \in \mathrm{L}.\forall s_0 \in \mathrm{S}.(s \xrightarrow{l} s_0) \\
\qquad \to \exists s_0' \in \mathrm{S}.s' \xrightarrow{l} s_0' \land \mathrm{Bisim}(s_0, s_0') \\
\forall s, s' \in \mathrm{S}.\mathrm{Bisim}(s, s') \to \forall l \in \mathrm{L}.\forall s_0' \in \mathrm{S}.(s' \xrightarrow{l} s_0') \\
\qquad \to \exists s_0 \in \mathrm{S}.s \xrightarrow{l} s_0 \land \mathrm{Bisim}(s_0, s_0')
\end{array}$$

Note that the type of E_1 is equivalent to

$$\begin{array}{l}
\mathrm{E}_1' : (s, s' \in \mathrm{S}) \to \mathrm{Bisim}(s, s') \\
\quad \to (s_0' \in (l \in \mathrm{L}) \to \{s_0 \in \{s_0 \in \mathrm{S} \mid s \xrightarrow{l} s_0\} \to \mathrm{S}) \\
\qquad \times((\langle l, s_0 \rangle \in ((l \in \mathrm{L}) \times \{s_0 \in \mathrm{S} \mid s \xrightarrow{l} s_0\})) \to \mathrm{Bisim}(s_0, s_0'(l, s_0)))
\end{array}$$

similarly for E_2 and both constructors can be unified into one. Therefore this relation is an instance of a strictly positive indexed coinductively defined set as defined in this article.

A proof of bisimulation by corecursion can be done by using the following schema: Let $I \in \mathrm{Set}$, $s, s' : I \to \mathrm{S}$.

We can prove $\forall i \in \mathrm{I}.\mathrm{Bisim}(s(i), s'(i))$ coinductively by defining for any $i \in \mathrm{I}$

- for any $l \in \mathrm{L}$, $s_0 \in \mathrm{S}$ s.t. $s(i) \xrightarrow{l} s_0$ an $s'_0 \in \mathrm{S}$ s.t.
  - $s'(i) \xrightarrow{l} s'_0$
  - and s.t. $\mathrm{Bisim}(s_0, s'_0)$
    where one can for prove the latter by invoking the Coinduction Hypothesis
    $\mathrm{Bisim}(s(i'), s'(i'))$ for some i' such that $s(i') = s_0$, $s'(i') = s'_0$.
- for any $l \in \mathrm{L}$, $s'_0 \in \mathrm{S}$ s.t. $s'(i) \xrightarrow{l} s'_0$ an $s_0 \in \mathrm{S}$ s.t.
  - $s(i) \xrightarrow{l} s_0$
  - and s.t. $\mathrm{Bisim}(s_0, s'_0)$
    where one can prove the latter by invoking the Coinduction Hypothesis
    $\mathrm{Bisim}(s(i'), s'(i'))$ for some i' such that $s(i') = s_0$, $s'(i') = s'_0$.

As an example consider $\mathrm{S} = \mathbf{1} \times \mathbb{N}$, $\mathrm{L} = \mathrm{tick}$ with transitions $* \xrightarrow{\mathrm{tick}} *$ and $n \xrightarrow{\mathrm{tick}} (n+1)$. We show $\forall n \in \mathbb{N}.\mathrm{Bisim}(*, n)$ by coinduction on Bisim. Assume $n \in \mathbb{N}$.
Assume $* \xrightarrow{l} s$. Then $l = \mathrm{tick}$, $s = *$, $n \xrightarrow{\mathrm{tick}} (n+1)$ and by co-IH $\mathrm{Bisim}(*, n+1)$.
Assume $n \xrightarrow{l} s$. Then $l = \mathrm{tick}$, $s = n+1$, $* \xrightarrow{\mathrm{tick}} *$ and by co-IH $\mathrm{Bisim}(*, n+1)$.

10 Conclusion

We have investigated indexed inductive and coinductively defined sets and shown that induction is equivalent to the initial algebra definition and coinduction, corecursion and coinduction are equivalent. We have developed schemata for defining informally elements of coinductively defined sets corecursively and for proving equality of elements by coinduction. We have seen that examples how to actually carry out such definitions and proofs informally.

We belief that carrying out such arguments about coinductively defined sets by corecursion and coinduction informally while referring to the coinduction and corecursion hypothesis makes it more intuitive to carry out such arguments. We hope that it will in the future become as natural to carry out such arguments as it has become natural to define functions into inductively defined sets by primitive recursion and to prove properties by induction in an intuitive way.

References

[1] A. Abel and B. Pientka. Wellfounded recursion with copatterns: a unified approach to termination and productivity. In G. Morrisett and T. Uustalu, editors, *ACM SIGPLAN International Conference on Functional Programming, ICFP'13, Boston, MA, USA - September 25 - 27, 2013*, pages 185–196. ACM, 2013.

[2] A. Abel, B. Pientka, D. Thibodeau, and A. Setzer. Copatterns: Programming infinite structures by observations. In R. Giacobazzi and R. Cousot, editors, *Proceedings of the 40th annual ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, POPL '13, pages 27–38, New York, NY, USA, 2013. ACM.

[3] P. Aczel. *Non-wellfounded set theory*, volume 14. CSLI Lecture notes, Stanford, CA: Stanford University, Center for the Study of Language and Information, 1988.

[4] P. Aczel. Algebras and coalgebras. In R. Backhouse, R. Crole, and J. Gibbons, editors, *Algebraic and Coalgebraic Methods in the Mathematics of Program Construction*, volume 2297 of *Lecture Notes in Computer Science*, pages 79–88. Springer, 2002.

[5] P. Aczel and N. Mendler. A final coalgebra theorem. In D. H. Pitt, D. E. Rydeheard, P. Dybjer, A. M. Pitts, and A. Poigné, editors, *Category Theory and Computer Science*, volume 389 of *Lecture Notes in Computer Science*, pages 357–365. Springer Berlin / Heidelberg, 1989. 10.1007/BFb0018361.

[6] T. Altenkirch and P. Morris. Indexed containers. In *Logic In Computer Science, 2009. LICS '09. 24th Annual IEEE Symposium on*, pages 277–285, 2009.

[7] M. Barr. Terminal coalgebras in well-founded set theory. *Theor. Comput. Sci.*, 114(2):299–315, 1993.

[8] P. J. de Bruin. *Inductive Types in constructive languages.* PhD thesis, Faculty of Mathematics and Natural Sciences, University of Groningen, Groningen, The Netherlands, March 1995. Available from https://www.rug.nl/research/portal/publications/pub%2887db58af-1fd6-4030-a862-98b5651d6be8%29.html and http://www.peterdebruin.net/.

[9] P. Dybjer. Inductive sets and families in Martin-Löf's type theory and their set-theoretic semantics. In G. Huet and G. Plotkin, editors, *Logical frameworks*, pages 280 – 306. Cambridge University Press, 1991.

[10] P. Dybjer and A. Setzer. A finite axiomatization of inductive-recursive definitions. In J.-Y. Girard, editor, *Typed Lambda Calculi and Applications*, volume 1581 of *Lecture Notes in Computer Science*, pages 129–146. Springer, April 1999.

[11] P. Dybjer and A. Setzer. Indexed induction-recursion. In R. Kahle, P. Schroeder-Heister, and R. Stärk, editors, *Proof Theory in Computer Science*, volume 2183 of *Lecture Notes in Computer Science*, pages 93–113. Springer, 2001.

[12] P. Dybjer and A. Setzer. Induction-recursion and initial algebras. *Annals of Pure and Applied Logic*, 124:1 – 47, 2003.

[13] P. Dybjer and A. Setzer. Indexed induction-recursion. *Journal of Logic and Algebraic Programming*, 66:1 – 49, 2006.

[14] P. Hancock, C. McBride, N. Ghani, L. Malatesta, and T. Altenkirch. Small induction recursion. In M. Hasegawa, editor, *Typed Lambda Calculi and Applications*, volume 7941 of *Lecture Notes in Computer Science*, pages 156–172. Springer, 2013.

[15] P. Hancock and A. Setzer. Interactive programs in dependent type theory. In P. Clote and H. Schwichtenberg, editors, *Computer Science Logic*, volume 1862 of *Lecture Notes in Computer Science*, pages 317–331. Springer, 2000.

[16] P. Hancock and A. Setzer. Specifying interactions with dependent types. In *Workshop on subtyping and dependent types in programming, Portugal, 7 July 2000*, 2000. Electronic proceedings, available via http://www-sop.inria.fr/oasis/DTP00/Proceedings/proceedings.html.

[17] P. Hancock and A. Setzer. Interactive programs and weakly final coalgebras (extended version). In T. Altenkirch, M. Hofmann, and J. Hughes, editors, *Dependently typed programming*, number 04381 in Dagstuhl Seminar Proceedings. Internationales Begegnungs- und Forschungszentrum (IBFI), Schloss Dagstuhl, Germany, 2004. Available via http://drops.dagstuhl.de/opus/volltexte/2005/176/.

[18] P. Hancock and A. Setzer. Interactive programs and weakly final coalgebras in dependent type theory. In L. Crosilla and P. Schuster, editors, *From Sets and Types to Topology and Analysis. Towards Practicable Foundations for Constructive Mathematics*, pages 115 – 134, Oxford, 2005. Clarendon Press.

[19] L. Malatesta, T. Altenkirch, N. Ghani, P. Hancock, and C. McBride. Small induction recursion, indexed containers and dependent polynomials are equivalent. Submitted for publication. Available from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.366.3934 &rep=rep1&type=pdf, 2012.

[20] R. Milner. Calculi for synchrony and asynchrony. *Theoretical Computer Science*, 25(3):267 – 310, 1983.

[21] P. Morris, T. Altenkirch, and N. Ghani. Constructing strictly positive families. In *Proceedings of the Thirteenth Australasian Symposium on Theory of Computing - Volume 65*, CATS '07, pages 111–121, Darlinghurst, Australia, Australia, 2007. Australian Computer Society, Inc.

[22] D. Park. Concurrency and automata on infinite sequences. In P. Deussen, editor, *Theoretical Computer Science*, volume 104 of *Lecture Notes in Computer Science*, pages 167–183. Springer Berlin Heidelberg, 1981.

[23] K. Petersson and D. Synek. A set constructor for inductive sets in Martin-Löf's Type Theory. In D. H. Pitt, D. E. Rydeheard, P. Dybjer, A. M. Pitts, and A. Poigné, editors, *Category Theory and Computer Science*, volume 389 of *Lecture Notes in Computer Science*, pages 128–140, London, UK, UK, 1989. Springer.

[24] J. Rutten. Universal coalgebra: A theory of systems. *Theoretical Computer Science*, 249(1):3 – 80, 2000.

[25] D. Sangiorgi. *Introduction to bisimulation and coinduction.* Cambridge University Press, 2011.

[26] D. Sangiorgi and J. Rutten. *Advanced topics in bisimulation and coinduction*, volume 52. Cambridge University Press, 2011.

[27] A. Setzer. Coalgebras as types determined by their elimination rules. In P. Dybjer, S. Lindström, E. Palmgren, and G. Sundholm, editors, *Epistemology versus Ontology*, volume 27 of *Logic, Epistemology, and the Unity of Science*, pages 351–369. Springer Netherlands, 2012. 10.1007/978-94-007-4435-6_16.

[28] A. Setzer, A. Abel, B. Pientka, and D. Thibodeau. Unnesting of copatterns. In G. Dowek, editor, *Rewriting and Typed Lambda Calculi. Proceedings RTA-TLCA 2014*, volume 8560 of *Lecture Notes in Computer Science*, pages 31–45. Springer, 2014.