Strong Induction Is an Up-To Technique

Filippo Bonchi
Universitá di Pisa, Italy

Elena Di Lavore
Universitá di Pisa, Italy

Anna Ricci
Universitá di Pisa, Italy

Abstract

Up-to techniques are enhancements of the coinduction proof principle which, in lattice theoretic terms, is the dual of induction. What is the dual of coinduction up-to? By means of duality, we illustrate a theory of induction up-to and we observe that an elementary proof technique, commonly known as strong induction, is an instance of induction up-to. We also show that, when generalising our theory from lattices to categories, one obtains an enhancement of the induction definition principle known in the literature as comonadic recursion.

2012 ACM Subject Classification Theory of computation → Proof theory; Theory of computation → Logic and verification

Keywords and phrases Induction, Coinduction, Up-to Techniques, Induction up-to, Lattices, Algebras

Digital Object Identifier 10.4230/LIPIcs.CSL.2025.28

Funding This study was carried out within the National Centre on HPC, Big Data and Quantum Computing - SPOKE 10 (Quantum Computing) and received funding from the European Union Next-GenerationEU - National Recovery and Resilience Plan (NRRP) – MISSION 4 COMPONENT 2, INVESTMENT N. 1.4 – CUP N. I53C22000690001. This research was partly funded by the Advanced Research + Invention Agency (ARIA) Safeguarded AI Programme.
Filippo Bonchi: Supported by the Ministero dell'Università e della Ricerca of Italy grant PRIN 2022 PNRR No. P2022HXNSC - RAP (Resource Awareness in Programming).

Acknowledgements The authors would like to thank Jurriaan Rot for the inspiring discussions.

1 Introduction

Induction is a fundamental tool frequently used by mathematicians, logicians, and computer scientists without much thought. It includes both *definition* and *proof* principles. The definition principle allows for the specification of data types, such as natural numbers, lists, or trees, and to define functions from them; the proof principle enables proving properties on such inductively defined structures.

The coinduction proof principle, which is formally the dual of induction, is less familiar. It first emerged in the 1970s [33] in three independent fields: set theory [14], modal logic [40], and concurrency theory [27]. Since then, it has been recognized as a fundamental principle in computer science and has been applied in various contexts [24, 1, 11, 16, 12, 31, 25, 17, 22].

Up-to techniques are enhancements of the coinduction proof principle, originally introduced by Milner in [23] to simplify coinductive arguments. Coinduction up-to has proven useful, if not essential, in numerous proofs about concurrent systems (see [30] for references). It has been used to establish decidability results [9], improve standard automata algorithms [6], and prove the completeness of domains in abstract interpretation [3].

Table 1 The lattices-to-categories correspondence. On the left, the proof and definition principles with their enhacement; On the right, the corresponding inductive invariants and algebras.

induction proof principle	induction definition principle	$f(x) \sqsubseteq x$	$a\colon FX \to X$
induction up-to	comonadic recursion	$fd(x) \sqsubseteq x$	$a\colon FDX \to X$
strong induction	course-of-value iteration	$ff^{\downarrow}(x) \sqsubseteq x$	$a\colon FF^{\downarrow}X \to X$

The theory of up-to techniques was initially developed by Sangiorgi [32] and later generalized to the abstract setting of complete lattices by Pous [28, 29]. In particular, Pous introduced the notion of *f-compatible* techniques for some monotone map f. Intuitively, these are are sound up-to techniques with advantageous composition properties.

A curiosity that may have occurred to many is the following:

Since coinduction is the dual of induction, what is the dual of coinduction up-to?

In this paper, we introduce a theory of induction up-to by simply dualizing the work of Pous [28]. Our main finding is that the well-known principle of *strong induction* over the set natural numbers $\mathbb{N}$, a principle familiar even to undergraduate students, is an example of an inductive up-to technique.

More precisely, we dualize the notion of f-compatible techniques from [28] to that of *f-cocompatible* (Definition 5) up-to techniques, which, as expected, are sound (Theorem 7) and enjoy good composition properties (Proposition 8). We show that, under mild conditions, any proof by coinduction up-to can equivalently be carried out by means of induction up-to, and vice versa (Proposition 11).

For any monotone map f, its down-closure, denoted by $f^{\downarrow}$, is always f-cocompatible (Corollary 9). We name induction up-to $f^{\downarrow}$ *strong induction* since, when f is the monotone map with the least fixed point $\mathbb{N}$, induction up-to $f^{\downarrow}$ coincides with the usual strong induction over $\mathbb{N}$ (Section 6.1). Unsurprisingly, the same approach can be applied to obtain strong induction on words (Section 6.2) and other inductive data types.

Overall, this shows that induction up-to, and in particular strong induction, provide enhancements for the induction proof principle. At this point, another curiosity arises:

What are enhancements of the induction definition principle?

Intuitively, one can think of *recursion schemes* as enhancements of the induction definition principle: they ensure that the specification of a recursive function is well-defined.

To formalize this intuition, we exploit the fact that Pous' theory [28] has a beautiful categorical meaning [5]: when generalizing this theory from lattices to categories, one obtains Bartel's λ-coinduction [2], an enhancement of the coinduction definition principle that generalizes several specification techniques common in computer science [36, 20], notably the abstract GSOS by Turi and Plotkin [37, 21].

We illustrate that, following the same pattern, the theory of induction up-to generalize to a certain recursion scheme known as *comonadic recursion* by Capretta, Uustalu, Vene and Pardo [39, 8] (Proposition 18). In particular, strong induction generalises to a scheme known as *course-of-value iteration* [38, 7] (Proposition 20). These correspondences are summarised in Table 1.

Related Work

An elegant theory of inductive enhancements has been recently introduced by Sangiorgi in [35]. This theory substantially differs from ours in its goals: Sangiorgi aims to enhance the proof methods for those behavioural equivalences and preorders [41, 42], such as trace, failure, and ready, that are defined inductively. These relations can usually be stratified, and the proposed inductive enhancements are functions on relations preserving such stratification. Like in the theory of coinductive enhancements [32, 28], the starting notion is the one of (semi-)progression and enhancements are up-closures, which is a crucial difference from our inductive invariants.

Another approach, similar in spirit to [35], is based on the techniques of unique solutions of equations [13, 34]. However, to the best of our understanding, its applicability seems to be strongly tailored to equivalence relations.

Synopsis

We begin our exposition in Section 2 with a simple example of proof by strong induction for the Fibonacci sequence. We recall the lattice-theoretic understanding of induction and coinduction in Section 3 and the theory of coinductive up-to techniques in Section 4; its dual, the theory of inductive up-to techniques, is illustrated in Section 5. In particular, Section 5.1 links coinduction up-to and induction up-to by means of an involution operator; Example 12 illustrates how, following this link, the coinductive technique of up-to equivalence becomes *up-to apartness*. Section 6.2 introduces strong induction as a certain up-to technique. The fact that this generalises strong induction on $\mathbb{N}$ is not obvious and its proof is detailed in Section 6.1. Strong induction on words is discussed in Section 6.2. In Section 7, we turn from the proof principle to the definition principle: Section 7.1 quickly recalls the induction definition principle by means of initial algebras; Section 7.2 recalls comonadic corecursion and Section 7.3 course-of-value iteration, a special case of comonadic corecursion. Finally, Section 7.4, revisits the Fibonacci, by recalling that its definition is by means of course of value iteration. The appendix contains the missing proofs and some additional material.

Until Section 7, the reader only requires some familiarity with lattice theory. Then, we expect the reader to be familiar with category theory.

2 Motivating Example: the Fibonacci sequence

Induction is a proof principle that applies to inductively defined structures. For instance, for proving that a predicate $P(n)$ hold for all natural numbers $n \in \mathbb{N}$, one has to find a predicate $Q(n)$ that implies $P(n)$, that is true for 0 and that, when true for n, then it is true for $n+1$.

$$\frac{Q(0) \quad \forall n \in \mathbb{N}. Q(n) \Rightarrow Q(n+1) \qquad Q \Rightarrow P}{\forall n \in \mathbb{N}. P(n)} \tag{1}$$

Sometimes, induction might be too weak to prove certain properties. As an example, consider the Fibonacci sequence defined as

$$\mathsf{fib}(0) \stackrel{\text{def}}{=} 1 \qquad \mathsf{fib}(1) \stackrel{\text{def}}{=} 1 \qquad \mathsf{fib}(n+2) \stackrel{\text{def}}{=} \mathsf{fib}(n+1) + \mathsf{fib}(n),$$

and suppose that one would like to prove that $\mathsf{fib}(n) \geq n$ for all $n \in \mathbb{N}$. One could start a proof by induction by checking the base case, $\mathsf{fib}(0) = 1 \geq 0$. For the inductive case, one would need to bound $\mathsf{fib}(n+1) = \mathsf{fib}(n) + \mathsf{fib}(n-1)$. At this point, one would be stuck because there is no information on $\mathsf{fib}(n-1)$ from the inductive hypothesis.

Strong induction comes to our rescue by allowing a stronger inductive hypothesis. We still need to find a predicate $Q(n)$ that implies $P(n)$ and that holds at 0, but when showing that it is true for $n+1$, we may assume that it holds for all $k \leq n$.

$$\frac{Q(0) \qquad \forall n(\forall n' \in [0,n]\,.\,Q(n')) \Rightarrow Q(n+1) \qquad Q \Rightarrow P}{\forall n \in \mathbb{N}.P(n)} \tag{2}$$

We conclude the proof of $\mathsf{fib}(n) \geq n$ for all $n \in \mathbb{N}$ by strong induction. For $n = 0$, $\mathsf{fib}(0) = 1 \geq 0$. For the inductive step, we assume that $\mathsf{fib}(k) \geq k$ for all $k \leq n+1$, and we seek to bound $\mathsf{fib}(n+1) = \mathsf{fib}(n) + \mathsf{fib}(n-1)$. If $n = 1$ then $\mathsf{fib}(2) = \mathsf{fib}(1) + \mathsf{fib}(0) = 1+1 \geq 2$. Otherwise, if $n > 1$, we use the strong inductive hypothesis:

$$\mathsf{fib}(n+2) = \mathsf{fib}(n+1) + \mathsf{fib}(n) \geq n+1+n \geq n+2\ .$$

We want to draw the reader's attention to the fact that, here, strong induction is necessary because fib is not–strictly speaking–inductively defined: the value of $\mathsf{fib}(n+1)$ does not depend only on $\mathsf{fib}(n)$. We will revisit the relationship between definitions and proofs in Section 7. Until then, we will focus only on the proof principles.

3 Preliminaries and notation

A complete lattice is a partially ordered set $(L, \sqsubseteq)$ with joins ($\sqcup$), meets ($\sqcap$), a top ($\top$) and a bottom ($\bot$) elements, least upper bounds ($\bigsqcup$) and greatest lower bounds ($\bigsqcap$). Henceforth, we use $(L, \sqsubseteq), (L_1, \sqsubseteq_1), (L_2, \sqsubseteq_2)$ to range over complete lattices and x, y, z to range over their elements. One lattice that we will often use is $\mathcal{P}(X)$, the power set of a set X, ordered by set inclusion.

Recall that a function $f\colon L_1 \to L_2$ is said to be a *monotone map* if it preserves the order: for all $x, y \in L_1$, if $x \sqsubseteq_1 y$ then $f(x) \sqsubseteq_2 f(y)$. The identity $\mathsf{id}_L\colon L \to L$ and the composition $f \circ g\colon L_1 \to L_3$ of two monotone maps $g : L_1 \to L_2$ and $f\colon L_2 \to L_3$ are monotone. Therefore, if $f\colon L \to L$ is a monotone map, then its powers f^n are also monotone, where the functions $f^n\colon L \to L$ are defined inductively as

$$f^0 \stackrel{\text{def}}{=} \mathsf{id}_L \qquad\qquad f^{n+1} \stackrel{\text{def}}{=} f \circ f^n. \tag{3}$$

We will implicitly use the fact that monotone maps form a complete lattice with their natural point-wise order: whenever $f, g\colon L_1 \to L_2$ we write $f \sqsubseteq g$ iff $f(x) \sqsubseteq g(x)$ for all $x \in L_1$.

A monotone map $f\colon L \to L$ is an *up-closure* operator if $x \sqsubseteq f(x)$ and $ff(x) \sqsubseteq f(x)$. It is a *down-closure* operator if $f(x) \sqsubseteq x$ and $f(x) \sqsubseteq ff(x)$. Particularly relevant to our exposition are the up-closures and the down-closure generated by a (co)continuous map $f\colon L \to L$, namely a monotone map preserving arbitrary least upper bounds and greatest lower bounds:

$$f^{\uparrow} \stackrel{\text{def}}{=} \bigsqcup_{i \in \mathbb{N}} f^i \qquad\qquad f^{\downarrow} \stackrel{\text{def}}{=} \bigsqcap_{i \in \mathbb{N}} f^i\ . \tag{4}$$

Given a monotone map $f\colon L \to L$, the element $x \in L$ is said to be a *post-fixed point* iff $x \sqsubseteq f(x)$; a *pre-fixed point* iff $f(x) \sqsubseteq x$; a *fixed point* iff $x = f(x)$. We write μf and νf for the *least* and *greatest fixed point*. For a monotone map f on a complete lattice L, the Knaster-Tarski fixed point theorem characterises μf as the least upper bound of all pre-fixed points of f and μf as the greatest lower bound of all its post-fixed points:

$$\mu f = \bigsqcap \{x \mid f(x) \sqsubseteq x\} \qquad \nu f = \bigsqcup \{x \mid x \sqsubseteq f(x)\}.$$

This immediately leads to the *induction* and *coinduction proof principles*, illustrated by the inference rules below, on the left and on the right, respectively [26].

$$\frac{f(y) \sqsubseteq y \quad y \sqsubseteq x}{\mu f \sqsubseteq x} \qquad \frac{y \sqsubseteq f(y) \quad x \sqsubseteq y}{x \sqsubseteq \nu f} \tag{5}$$

The induction proof principle states that in order to prove that $\mu f \sqsubseteq x$, one should provide an *inductive invariant* –namely, a pre-fixed point of f– that is below x; dually, the coinduction proof principle states that in order to prove that $x \sqsubseteq \nu f$, one should provide a *coinductive invariant*, i.e., a post-fixed point of f, that is above x.

▶ Remark 1. From this lattice theoretic perspective, it is easy to see that the coinduction proof principle is simply the *dual* of induction. Indeed, whenever $(L, \sqsubseteq)$ is a lattice, then so is $(L, \sqsupseteq)$. Similarly, if $f\colon (L, \sqsubseteq) \to (L, \sqsubseteq)$ is monotone, then so is $f\colon (L, \sqsupseteq) \to (L, \sqsupseteq)$, and the greatest fixed point of f over $(L, \sqsubseteq)$ becomes the least fixed point of f over $(L, \sqsupseteq)$.

We illustrate inductive and coinductive invariants with an example from automata theory.

▶ **Example 2** (cf. [6, Remark 2])**.** We denote by A^* the set of words over an alphabet A; ϵ denotes the empty word and $u \cdot w$ the word obtained by concatenating $u \in A^*$ with $w \in A^*$. For a word $w \in A^*$, we indicate its length with $|w|$.

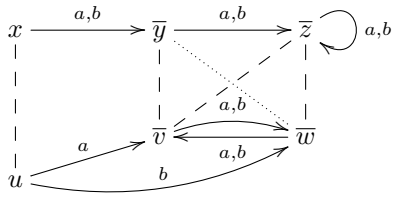
A deterministic automaton on the alphabet A is a triple (X, o, t), where X is a set of states, $o\colon X \to \{0,1\}$ is the output function, determining if a state x is accepting ($o(x) = 1$) or not ($o(x) = 0$) and $t\colon X \to X^A$ is the transition function which returns the next state, for each letter $a \in A$. Every automaton (X, o, t) induces a function $\mathsf{lan}_{\_}\colon X \to \{0,1\}^{A^*}$ defined inductively for all $x \in X$, $a \in A$ and $w \in A^*$ as $\mathsf{lan}_x(\varepsilon) = o(x)$ and $\mathsf{lan}_x(a \cdot w) = \mathsf{lan}_{t(x)(a)}(w)$. Two states $x, y \in X$ are said to be *language equivalent*, in symbols $x \sim y$, iff $\mathsf{lan}_x = \mathsf{lan}_y$.

Alternatively, $(\sim)$ can be defined as the greatest fixed point of some map on $\mathcal{P}(X \times X)$, the lattice of relations over X. The functions $l, q\colon \mathcal{P}(X \times X) \to \mathcal{P}(X \times X)$ are defined as

$$l(R) \stackrel{\text{def}}{=} \{(x,y) \mid \text{for all } a \in A,\ (t(x)(a), t(y)(a)) \in R\} \qquad q(R) \stackrel{\text{def}}{=} \{(x,y) \mid o(x) = o(y)\} \tag{6}$$

for all $R \subseteq X \times X$. One can easily check that both l and q are monotone and that $\nu(l \sqcap q) = (\sim)$. Thanks to this characterisation, one can prove that two states $x', y' \in X$ are language equivalent by means of the coinduction proof principle in (5): to show that $\{(x', y')\} \subseteq (\sim)$, it is enough to provide a relation R that is a post-fixed point of $l \sqcap q$ and such that $\{(x', y')\} \subseteq R$. Such coinductive invariants are often called *bisimulations*.

For an example, consider the following deterministic automaton, where final states are overlined and the transition function is represented by labelled arrows. The relation consisting of dashed and dotted lines is a bisimulation witnessing that $\{(x, u)\} \subseteq (\sim)$.



(7)

One can prove that $\{(x', y')\} \subseteq (\sim)$ by means of induction as well: for all $R \subseteq X \times X$, the functions $l^\dagger, p\colon \mathcal{P}(X \times X) \to \mathcal{P}(X \times X)$ are defined as follows.

$$l^\dagger(R) \stackrel{\text{def}}{=} \{(t(x)(a), t(y)(a)) \mid a \in A, (x,y) \in R\} \qquad p(R) \stackrel{\text{def}}{=} \{(x', y')\} \tag{8}$$

Note that p above, as well as q in (6), are constant functions: we will sometime take the freedom to identify them with the corresponding element in the lattice. Intuitively, $\mu(l^\dagger \sqcup p)$ represents the subset of all pairs of states that are reachable from the pair (x', y'). Thus, $\{(x', y')\} \subseteq (\sim)$ if and only if all those pairs of states are in q, i.e., if and only if $\mu(l^\dagger \sqcup p) \subseteq q$. The latter can be proved by exhibiting a relation R that is a pre-fixed point of $l^\dagger \sqcup p$ and such that $R \subseteq q$: the relation formed by the dashed and dotted lines in (7) satisfies this condition when taking (x', y') to be (x, u).

4 Coinduction up-to

Coinduction is a technique for proving $x \sqsubseteq \nu f$ for some map f on a lattice $(L, \sqsubseteq)$ by providing a coinductive invariant for f. In many situations, providing such an invariant is far too complicated. Motivated by this fact, Milner [23] introduced enhancements of the coinduction proof principle which are nowadays widely known as up-to techniques. In a nutshell, an *up-to technique* is an up-closure $d\colon (L, \sqsubseteq) \to (L, \sqsubseteq)$. An *$f$-coinductive invariant up-to d* is some $y \in L$ such that $y \sqsubseteq fd(y)$, namely a post-fixed point of fd. An up-to technique d is said to be *sound* w.r.t. f if the following *coinduction up-to principle* holds.

$$\frac{y \sqsubseteq f(d(y)) \quad x \sqsubseteq y}{x \sqsubseteq \nu f} \qquad \text{(Coinduction Up-To)}$$

In (5), one has to find an invariant y such that $y \sqsubseteq f(y)$. In (Coinduction Up-To), the search of such a y is simplified since it is enough that $y \sqsubseteq f(d(y))$. Since d is an up-closure, $f(y) \sqsubseteq f(d(y))$, which simplifies the task of finding coinductive invariants.

▶ **Example 3** (Up-to equivalence)**.** We continue Example 2 to illustrate a coinductive invariant up-to. We instantiate (Coinduction Up-To) by taking f to be $l \sqcap q$ and d to be the function $e\colon \mathcal{P}(X \times X) \to \mathcal{P}(X \times X)$ mapping any relation $R \subseteq X \times X$ into its equivalence closure. One can check $(\sim)$ by exhibiting a relation R such that $R \subseteq l \sqcap q(e(R))$.

Consider for instance the relation S consisting of only the dashed lines in (7). Note that $(y, w) \in e(S)$ but $(y, w) \notin S$. It is thus easy to see that $S \subseteq l \sqcap q(e(S))$, but S is not included in $l \sqcap q(S)$. In other words, S is a coinductive invariant up-to e but not a coinductive invariant. In Example 2, to prove that $x \sim u$ by means of coinduction, we need to take the relation consisting of both dashed and dotted lines in (7). With coinduction up-to e, it is thus enough to take only the dashed lines.

Of course, before using an up-to technique, one should prove it to be sound. Since this might be quite challenging, several theories [32, 28, 10, 29, 4] have been introduced for simplifying this task. In this paper we will consider the theory of Pous in [28] that focuses on the notion of compatible techniques: d is *f-compatible* iff $df \sqsubseteq fd$. The key results in [28] state that compatible techniques are sound and can be nicely composed.

5 Induction up-to

Recall that for applying the induction proof principle in (5), one has to find a $y \in L$ such that $f(y) \sqsubseteq y$. The idea of induction up-to is to simplify this task by weakening such constraint to $f(d(y)) \sqsubseteq y$ for some $d\colon L \to L$.

Note that if the map d is an up-closure, as it is the case of coinduction up-to, then this would only complicate our task by imposing additional constraints; indeed $f(y) \sqsubseteq f(d(y))$.

▶ **Example 4.** Recall from Example 2 that the relation R consisting of both dashed and dotted lines in (7) is an inductive invariant, i.e., $(l^\dagger \sqcup p)(R) \subseteq R$. Note that, instead $l^\dagger \sqcup p(e(R))$ is not included into R since, e.g., (v, w) is in $l^\dagger \sqcup p(e(R))$ but not in R.

As expected, the solution consists in considering down closures. An *(inductive) up-to technique* is a down closure $d\colon (L, \sqsubseteq) \to (L, \sqsubseteq)$. An *$f$-inductive invariant up-to d* is some $y \in L$ such that $fd(y) \sqsubseteq y$, namely a pre-fixed point of fd. An up-to technique d is said to be *sound* w.r.t. f if the following *induction up-to principle* holds.

$$\frac{f(d(y)) \sqsubseteq y \quad y \sqsubseteq x}{\mu f \sqsubseteq x} \qquad \text{(Induction Up-To)}$$

To prove the soundness of inductive up-to techniques, we consider the dual of the notion of *compatible functions* from [28].

▶ **Definition 5** (Cocompatible map). *Let $f, d\colon (X, \sqsubseteq) \to (X, \sqsubseteq)$ be two monotone maps. We say that d is* cocompatible with *f, shortly f-cocompatible, if $fd \sqsubseteq df$.*

When d is f-cocompatible, any inductive invariant up-to gives rise to an inductive invariant.

▶ **Proposition 6.** *Let d be a down closure that is cocompatible with some monotone map f. If y is a pre-fixed point for fd, then $d(y)$ is a pre-fixed point for f.*

Proof.

$$\begin{aligned} fd(y) &\sqsubseteq fdd(y) && (d \text{ down closure}) \\ &\sqsubseteq dfd(y) && (d \text{ is } f\text{-cocompatible}) \\ &\sqsubseteq d(y) && (y \text{ is a pre-fixed point of } fd) \end{aligned}$$

◀

▶ **Theorem 7.** *If d is f-cocompatible, then it is sound.*

Proof. We have to prove the conclusion of (Induction Up-To) assuming its premise. By $fd(y) \sqsubseteq y$ and Proposition 6, it holds that

$$fd(y) \sqsubseteq d(y).$$

Since $d(y) \sqsubseteq y$, as d is a downclosure, and $y \sqsubseteq x$ it holds that

$$d(y) \sqsubseteq x.$$

Thus by replacing y with $d(y)$ in (5), it holds that $\mu f \sqsubseteq x$. ◀

It is worth mentioning that the two results above also hold by dualising the theory in [28]. We have reported their proofs since they will be relevant in Section 7. The following result also follows easily from [28]. For convenience of the reader we report its proof in Appendix A.

▶ **Proposition 8** (The algebra of cocompatible maps). *Let $f, d, e\colon (X, \sqsubseteq) \to (X, \sqsubseteq)$ be monotone maps. Let $\{d_i\}_{i \in \mathbb{N}}$ be an $\mathbb{N}$-indexed family of monotone maps.*

1. *The identity id_X is f-cocompatible;*
2. *f is f-cocompatible;*
3. *If d and e are f-cocompatible, then $d \circ e$ is f-cocompatible;*
4. *If d is f-cocompatible then, for all $n \in \mathbb{N}$, d^n is f-cocompatible;*

5. *If d and e are f-cocompatible, then $d \sqcap e$ is f-cocompatible;*
6. *If, for all $i \in \mathbb{N}$, d_i is f-cocompatible, then $\bigsqcap_{i\in\mathbb{N}} d_i$ is f-cocompatible;*
7. *If d is f-cocompatible, then $d^{\downarrow}$ is f-cocompatible.*

▶ **Corollary 9.** *Let $f\colon (X,\sqsubseteq) \to (X,\sqsubseteq)$ be a continuous monotone map. Then, its down-closure $f^{\downarrow}$ is f-cocompatible.*

Proof. By point 2 and 7 in Proposition 8. ◀

▶ Remark 10. Note that we have defined up-to techniques to be down-closures, while compatible maps are defined as arbitrary monotone maps. This choice is justified by the fact that monotone maps compose nicely, while down-closures do not. This motivated Pous to introduce up-to techniques in the original theory in [28] as monotone maps rather than up-closures. Here, we preferred to stay with closures, as this simplifies the proofs of Proposition 6 and Theorem 7, which will be relevant in the categorical generalisation illustrated in Section 7.

Note also that restricting to down-closures does not limit the applicability of the theory: indeed, if d is an f-compatible monotone map, not necessarily a down-closure, then, by Proposition 8.7, $d^{\downarrow}$ is also f-compatible. Moreover, if $fd(y) \sqsubseteq x$, then

$$fd^{\downarrow}(y) \sqsubseteq fd(y) \sqsubseteq x$$

since $d^{\downarrow} \sqsubseteq d$. In other words, any proof up-to d is also a proof up-to $d^{\downarrow}$.

5.1 Relating Coinduction up-to and Induction Up-to via Involution

Coinduction and induction are equivalent whenever the lattice $(L,\sqsubseteq)$ comes with an involution operator $\neg\colon (L,\sqsubseteq) \to (L,\sqsupseteq)$, namely a function on L such that

$$\text{if } x \sqsubseteq y, \text{ then } \neg x \sqsupseteq \neg y \qquad \neg\neg x = x \tag{9}$$

for all $x, y \in L$. In this case, for any monotone map f on $(L,\sqsubseteq)$, one has that $\overline{f} \stackrel{\text{def}}{=} \neg f \neg$ is a monotone map. Moreover, assuming that f preserves $\bigsqcap$ of ω-chains, it holds that:

$$x \sqsubseteq \nu f \Leftrightarrow \neg(\nu f) \sqsubseteq \neg x \Leftrightarrow \mu\overline{f} \sqsubseteq \neg x.$$

Such correspondence lifts to up-to techniques: whenever one can prove the leftmost by coinduction up-to d, for some f-compatible technique d, one can equivalently prove the rightmost by induction up-to to $\overline{d}$. This is made formal by the following result.

▶ **Proposition 11.** *Let $f, d\colon (L,\sqsubseteq) \to (L,\sqsubseteq)$ be monotone maps and y be an element of L.*
1. *d is an up-closure iff $\overline{d}$ is a down-closure;*
2. *d is f-compatible iff $\overline{d}$ is $\overline{f}$-cocompatible;*
3. *y is an f-coinductive invariant up-to d iff $\neg y$ is an $\overline{f}$-inductive invariant up-to $\overline{d}$.*

▶ **Example 12** (Up-to apartness)**.** Following the above considerations, one can transform coinduction up-to equivalence in Example 3 into *induction up-to apartnesses*. Apartness relations are standard in constructive reals analysis and has been first axiomatised in [19]: $R \subseteq X \times X$ is a an *apartness relation* if it is

- irreflexive: $(x,x) \notin R$ for all $x \in X$;
- symmetric: if $(x,y) \in R$, then $(y,x) \in R$ for all $x, y \in X$;
- co-transitive: if $(x,y) \in R$, then $(x,z) \in R$ or $(z,y) \in R$, for all $x, y, z \in X$.

The reader can easily check that R is an apartness relation iff $\neg R$ is an equivalence relation, where $\neg$ indicates the complement of a relation. Recall from Example 3 that the up-closure $e\colon \mathcal{P}(X \times X) \to \mathcal{P}(X \times X)$ mapping any relation R into its equivalence closure. By Proposition 11.1, $\overline{e}$ is a down closure: it maps any R into the largest apartnesses relation contained in R. Since e is $(l \sqcap q)$-compatible, by Proposition 11.2, $\overline{e}$ is $(\overline{l \sqcap q})$-cocompatible. Since $\mathcal{P}(X \times X)$ is a boolean algebra, then $\overline{l \sqcap q} = \overline{l} \sqcup \overline{q}$; it is easy to check that $\overline{l}$ and $\overline{q}$ map any $R \subseteq X \times X$ into the following relations.

$$\overline{l}(R) = \{(x, y) \mid \text{exists } a \in A,\ (t(x)(a), t(y)(a)) \in R\} \qquad \overline{q}(R) = \{(x, y) \mid o(x) \neq o(y)\}.$$

With this characterisation, one can see that inductive invariants for $\overline{l} \sqcup \overline{q}$ are exactly those introduced [15, Definition 2.2]. Our work enhances this proof method with up-to apartness.

For an example of an inductive invariant up-to apartness, consider again the relation S consisting of the dashed lines in (7). Since S is a $(l \sqcap q)$-coinductive invariant up to e, then by Proposition 11.3, $\neg S$ is a $(\overline{l} \sqcup \overline{q})$-inductive invariant up-to $\overline{e}$.

6 Strong Induction is an up-to technique

This section studies the proof principle given by a particular f-cocompatible map: the down-closure of f. Indeed, by Corollary 9, $f^{\downarrow}$ is f-compatible and, by Theorem 7, the following proof principle is always sound.

$$\frac{f(f^{\downarrow}(y)) \sqsubseteq y \quad y \sqsubseteq x}{\mu f \sqsubseteq x} \qquad \text{(Strong Induction)}$$

We call such principle *strong induction*. Indeed, as we illustrate below, when instantiated to usual induction on natural numbers, the above proof principle coincides with the well-known strong induction illustrated in Section 2.

6.1 Strong Induction on natural numbers

We begin by illustrating how (5) generalises standard induction over natural numbers. Consider the lattice $\mathcal{P}(\mathbb{N})$ and the monotone map $b\colon \mathcal{P}(\mathbb{N}) \to \mathcal{P}(\mathbb{N})$ defined as

$$b(X) \stackrel{\text{def}}{=} \{0\} \cup \{x + 1 \mid x \in X\} \tag{10}$$

for all $X \in \mathcal{P}(\mathbb{N})$. The least fixed point of b is the set of natural numbers, $\mu b = \mathbb{N}$. We take the sets X and Y to be the sets of natural numbers on which $P(n)$ and $Q(n)$ are true, respectively.

$$X = \{n \in \mathbb{N} \mid P(n)\} \qquad\qquad Y = \{n \in \mathbb{N} \mid Q(n)\}$$

With these choices, set inclusion corresponds to predicate implication: $Y \subseteq X$ iff $Q \Rightarrow P$. The least fix point of b is contained in X iff all natural numbers are contained in X, which means that $P(n)$ holds for all $n \in \mathbb{N}$.

$$\mu b \subseteq X \quad \text{iff} \quad \mathbb{N} \subseteq X \quad \text{iff} \quad \forall n \in \mathbb{N}. P(n)$$

Similarly, Y is a pre-fixed point of b iff Y contains 0 and it contains $n + 1$ for each $n \in Y$, which means that Q holds at 0 and it holds at $n + 1$ whenever it holds at n.

$$b(Y) \subseteq Y \quad \text{iff} \quad \{0\} \cup \{n + 1 \mid n \in Y\} \subseteq Y \quad \text{iff} \quad Q(0) \text{ and } \forall n \in \mathbb{N}. Q(n) \Rightarrow Q(n + 1)$$

These considerations show that (1) is a particular instance of (5), when we instantiate it to $b\colon \mathcal{P}(\mathbb{N}) \to \mathcal{P}(\mathbb{N})$.

$$\frac{b(Y) \subseteq Y \quad Y \subseteq X}{\mu b \subseteq X} \qquad \text{iff} \qquad \frac{Q(0) \quad \forall n \in \mathbb{N}. Q(n) \Rightarrow Q(n+1) \qquad Q \Rightarrow P}{\forall n \in \mathbb{N}. P(n)}$$

We can now illustrate our main observation: when instantiating (Strong Induction) to b one obtains exactly the strong induction on natural numbers reported in (2). We start by computing the powers b^n of b.

▶ **Lemma 13.** *For all $n \in \mathbb{N}$, and all $X \in \mathcal{P}(\mathbb{N})$,*

$$b^n(X) = \{x \in \mathbb{N} \mid x < n\} \cup \{x + n \mid x \in X\}.$$

Proof. By induction. For the base case, we have the following derivation.

$$\begin{aligned} b^0(X) &= id_{\mathcal{P}(\mathbb{N})}(X) && (3)\\ &= X \\ &= \emptyset \cup X \\ &= \{x \in \mathbb{N} \mid x < 0\} \cup \{x + 0 \mid x \in X\} \end{aligned}$$

For the inductive case, we have the following derivation.

$$\begin{aligned} b^{n+1}(X) &= bb^n(X) && (3)\\ &= b(\,\{x \in \mathbb{N} \mid x < n\} \cup \{x + n \mid x \in X\}\,) && (\text{Ind. Hyp.})\\ &= \{0\} \cup \{x + 1 \mid x < n\} \cup \{x + n + 1 \mid x \in X\} && (10)\\ &= \{0\} \cup [1, n] \cup \{x + n + 1 \mid x \in X\} \\ &= \{x \in \mathbb{N} \mid x < n + 1\} \cup \{x + n + 1 \mid x \in X\} \end{aligned}$$ ◀

The core of our argument relies on the following result, stating that $b^{\downarrow}(X)$ is the largest closed interval from including 0 that is a subset of X.

▶ **Lemma 14.** *For any set $X \in \mathcal{P}(\mathbb{N})$, $b^{\downarrow}(X)$ is characterised as $b^{\downarrow}(X) = \{x \mid [0, x] \subseteq X\}$.*

Proof. By definition $b^{\downarrow} = \bigsqcap_{n=0}^{\infty} b^n$. Thus,

$$\begin{aligned} m \in b^{\downarrow}(X) &\Leftrightarrow \forall n \in \mathbb{N}.\, m \in b^n(X) \\ &\Leftrightarrow \forall n \in \mathbb{N}.\, (m \in \{x \in| \; x < n\} \vee m \in \{x + n \mid x \in X\}) && (\text{Lemma 13})\\ &\Leftrightarrow \forall n \in \mathbb{N}.\, (m < n \vee m \in \{x + n \mid x \in X\}) \\ &\Leftrightarrow \forall n \in \mathbb{N}.\, (\neg(m \geq n) \vee m \in \{x + n \mid x \in X\}) \\ &\Leftrightarrow \forall n \in \mathbb{N}.\, ((m \geq n) \Rightarrow m \in \{x + n \mid x \in X\}) \\ &\Leftrightarrow \forall n \in \mathbb{N}.\, ((n \leq m) \Rightarrow \exists x \in X.\, m = x + n) \\ &\Leftrightarrow \forall n \in \mathbb{N}.\, ((n \leq m) \Rightarrow \exists x \in X.\, x = m - n) \\ &\Leftrightarrow \forall n \in \mathbb{N}.\, ((n \leq m) \Rightarrow (m - n) \in X) \end{aligned}$$

In short,

$$m \in b^{\downarrow}(X) \Leftrightarrow \forall n \in \mathbb{N}.((n \leq m) \Rightarrow (m - n) \in X) \qquad (11)$$

We use (11) to prove the two inclusions of $b^{\downarrow}(X) = \{x \mid [0, x] \subseteq X\}$ separetely:

- $b^{\downarrow}(X) \subseteq \{x \mid [0,x] \subseteq X\}$. We assume that $m \in b^{\downarrow}(X)$ and we need to prove that $[0,m] \subseteq X$. Let us take an arbitrary $y \in [0,m]$. Since by (11), $(m-n) \in X$ for all $n \leq m$, then one can take n to be $m-y$ and have that $m-(m-y) \in X$, that is, $y \in X$.
- $b^{\downarrow}(X) \supseteq \{x \mid [0,x] \subseteq X\}$. We assume that $[0,m] \subseteq X$. Thus $\forall n \in \mathbb{N}.\,((n \leq m) \Rightarrow (m-n) \in X)$ that, by (11), means that $m \in b^{\downarrow}(X)$. ◀

▶ **Proposition 15.** *For any set $Y \in \mathcal{P}(\mathbb{N})$, the following are equivalent*

- $bb^{\downarrow}(Y) \subseteq Y$;
- $0 \in X$ *and* $(\forall n \in \mathbb{N}\,.\,[0,n] \subseteq Y \Rightarrow n+1 \in Y)$.

Proof.

$$
\begin{aligned}
b\,b^{\downarrow}(Y) \subseteq Y &\Leftrightarrow \{0\} \cup \{y+1 \mid y \in b^{\downarrow}(Y)\} \subseteq Y && (10)\\
&\Leftrightarrow 0 \in Y \text{ and } \{y+1 \mid y \in b^{\downarrow}(Y)\} \subseteq Y\\
&\Leftrightarrow 0 \in Y \text{ and } (\{y+1 \mid y \in \{n \in \mathbb{N} \mid [0,n] \subseteq Y\}\} \subseteq Y) && (\text{Lemma } 14)\\
&\Leftrightarrow 0 \in Y \text{ and } (\forall n \in \mathbb{N}\,.\,[0,n] \subseteq Y \Rightarrow n+1 \in Y)
\end{aligned}
$$

◀

The above proposition allows us to easily see that strong induction (2) is induction up-to $b^{\downarrow}$. The latter is illustrated below on the left. The former is reported on the right.

$$
\frac{bb^{\downarrow}(Y) \subseteq Y \quad Y \subseteq X}{\mu b \subseteq X} \quad \text{iff} \quad \frac{Q(0) \qquad \forall n(\forall n' \in [0,n]\,.\,Q(n')) \Rightarrow Q(n+1) \qquad Q \Rightarrow P}{\forall n \in \mathbb{N}.P(n)}
$$

The correspondence between the two rules mirrors that of induction. The conclusions of the two rules coincide in the same way that they did for induction. For the premise, observe that, by Proposition 15,

$$
b\,b^{\downarrow}(Y) \subseteq Y \qquad \text{iff} \qquad Q(0) \wedge (\,\forall n(\forall n' \in [0,n]\,.\,Q(n')) \Rightarrow Q(n+1)\,).
$$

6.2 Strong Induction on Words

As expected, one can use strong induction not only on $\mathbb{N}$ but on any inductive data type. Below, we illustrate strong induction on A^*, the set of words over an alphabet A.

As we did for natural numbers, we need to give a monotone map $c\colon \mathcal{P}(A^*) \to \mathcal{P}(A^*)$ that gives induction on words, i.e. whose least fixed point is A^*. The candidate monotone map c mimics the definition of the monotone map b for natural numbers: it maps a set X to the set containing the empty word and all the successors of words in X.

$$
c(X) \stackrel{\text{def}}{=} \{\epsilon\} \cup \{a \cdot w \mid w \in X,\ a \in A\} \tag{12}
$$

Since the least fixed point of c is A^*, the induction principle (5) instantiated to c give us the usual induction principle on words.

$$
\frac{c(Y) \subseteq Y \quad Y \subseteq X}{\mu c \subseteq X} \quad \text{iff} \quad \frac{Q(\epsilon) \qquad \forall a \in A.\forall w \in A^*.Q(w) \Rightarrow Q(a \cdot w) \qquad Q \Rightarrow P}{\forall w \in A^*.P(w)}
$$

We now turn our attention to (Strong Induction) instantiated with c:

$$
\frac{cc^{\downarrow}(Y) \subseteq Y \quad Y \subseteq X}{\mu c \subseteq X}
$$

What does this means in practice? To answer this question, the key is to have a handy characterisation of $c^{\downarrow}$. This is going to resemble that of $b^{\downarrow}$ but, instead of the ordering on natural numbers, we consider the suffix partial ordering of words:

$$v \sqsubseteq_{A^*} w \quad \text{iff} \quad \exists u \in A^*. w = u \cdot v \ .$$

The analogue of the interval $[0, n]$ for natural numbers is, then, the set of suffixes of a word, $\mathsf{Suf}(w) \stackrel{\text{def}}{=} \{u \in A^* \mid u \sqsubseteq w\}$. With this, we obtain that the down closure of c gives the biggest subset that is closed under suffixes.

$$c^{\downarrow}(X) = \{x \in A^* \mid \mathsf{Suf}(x) \subseteq X\}$$

With this result, we can explicit the strong induction principle on words.

$$\frac{Q(\epsilon) \qquad \forall a \in A. \forall w \in A^*. (\forall y \in \mathsf{Suf}(w) \,.\, Q(y)) \Rightarrow Q(a \cdot w) \qquad Q \Rightarrow P}{\forall w \in A^*. P(w)}$$

Compare this principle with strong induction on natural numbers: consider the singleton set $A = \{*\}$. Then, words on A are determined by their length, so A^* is in bijection with the natural numbers $\mathbb{N}$. Through this bijection, $\mathsf{Suf}(w)$ coincides with the interval $[0, |w|]$. This further justifies the name of strong induction.

7 From Lattice to Categories

Induction is both a proof principle and a *definition principle*. The latter can be obtained as generalisation of the former by moving from lattices to categories. As (inductive) up-to techniques are enhancements of the induction proof principle, by means of a similar generalisation, one can obtain enhancements of the induction definition principle. In this section, we illustrate that induction up-to generalises to a recursion scheme known as *comonadic recursion* [8] and strong induction generalises to *course-of-value iteration* [38, 7].

7.1 Initial agebras and the induction definition principle

Hereafter, we write $\times$ and $+$ for products and coproducts in some category $\mathbf{C}$, $\langle a, b\rangle \colon X \to Y \times Z$ for the pairing of $a \colon X \to Y$ and $b \colon X \to Z$ and $[c, d] \colon Y + Z \to X$ for the copairing of $c \colon Y \to X$ and $d \colon Z \to X$. The singleton set $\{\epsilon\}$ is denoted by 1.

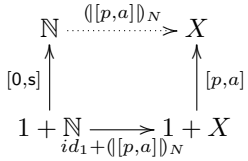
Given a functor $F \colon \mathbf{C} \to \mathbf{C}$ on some category $\mathbf{C}$, an *F-algebra* is a pair (X, a) where X is an object of $\mathbf{C}$ and $a \colon FX \to X$ is an arrow. Given two F-algebras (X, a) and (Y, b), an algebra morphism $h \colon (X, a) \to (Y, b)$ is an arrow $h \colon X \to Y$ of $\mathbf{C}$ making the diagram below commute. An F-algebra $(\mu F, \mathsf{i})$ is said to be *initial* if for any F-algebra (X, a), there exists a unique algebra morphism $(\!|a|\!)_F \colon (\mu F, \mathsf{i}) \to (X, a)$.

$$\begin{array}{ccc} X & \xrightarrow{h} & Y \\ {\scriptstyle a}\uparrow & & \uparrow{\scriptstyle b} \\ FX & \xrightarrow[Fh]{} & FY \end{array}$$

Initial algebras give the *induction definition principle*: in order to specify a morphism from μF to some object X it is enough to give an F-algebra on X.

▶ Remark 16. When the category $\mathbf{C}$ is a complete lattice, a functor F on $\mathbf{C}$ is simply a monotone map; an F-algebra is a pre-fixed point for F and an initial F-algebra is a least fixed-point. In this perspective, the induction definition principle collapses to the induction proof principle: specifying an arrow $\mu F \to X$ means exactly proving that $\mu F \sqsubseteq X$.

Consider **Set** –the category of sets and functions– and $N\colon \mathbf{Set} \to \mathbf{Set}$ the functor mapping a set X into $1 + X$ and a function a into $id_1 + a$. An initial algebra for N is provided by $(\mathbb{N}, [0, \mathsf{s}])$ where $0\colon 1 \to \mathbb{N}$ assigns to ϵ the number 0 and $\mathsf{s}\colon \mathbb{N} \to \mathbb{N}$ assigns to any $n \in \mathbb{N}$, its successor $n + 1$. Now, given an F-algebra $[p, a]\colon 1 + X \to X$, one obtains, by initiality, a function $(\!|[p, a]|\!)_N\colon \mathbb{N} \to X$, as illustrated below on the left.



$$\begin{aligned}(\!|[p,a]|\!)_N(0) &= p(\epsilon)\\ (\!|[p,a]|\!)_N(n+1) &= a((\!|[p,a]|\!)_N(n))\end{aligned}$$

The fact that the diagram on the left commutes is expressed by the conditions on the right. The first condition provides the base case of an inductive definition; the second one provides the inductive case. Note that, in order to define $(\!|[p, a]|\!)_N(n+1)$, one uses the value $(\!|[p, a]|\!)_N(n)$. Sometimes, as in the Fibonacci sequence in Section 2, functions are specified by using not just their value at n but also their values at some smaller numbers. This can be done by enhancing the induction definition principle by means of recursion schemes.

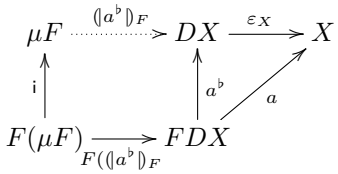
7.2 Comonadic Recursion

A *comonad* on a category $\mathbf{C}$ is a functor $D\colon \mathbf{C} \to \mathbf{C}$ together with two natural transformations, the counit $\varepsilon\colon D \Rightarrow Id$ and the comultiplication $\delta\colon D \Rightarrow DD$, such that $\varepsilon_{DX} \circ \delta_X = id_{DX} = D\varepsilon_X \circ \delta_X$ and $D\delta_X \circ \delta_X = \delta_{DX} \circ \delta_X$ for all objects X. A *distributive law* of a functor $F\colon \mathbf{C} \to \mathbf{C}$ over the comonad D is a natural transformation $\zeta\colon FD \Rightarrow DF$ such that $\varepsilon_{FX} \circ \zeta_X = F\varepsilon_X$ and $\delta_{FX} \circ \zeta_X = D\zeta_X \circ \zeta_{DX} \circ F\delta_X$.

Comonadic recursion exploits a comonad D and a distributive law $\zeta\colon FD \Rightarrow DF$ to enhance the induction definition principle. In order to define a morphism from μF to X, rather than specifying an F-algebra, one can specify an FD-algebra $a\colon FDX \to X$. Indeed, such a gives rise to

$$a^\flat \overset{\text{def}}{=} FDX \xrightarrow{F\delta_X} FDDX \xrightarrow{\zeta_{DX}} DFDX \xrightarrow{Da} DX$$

which is an F-algebra and thus, by initiality of μF, one obtains $(\!|a^\flat|\!)_F\colon \mu F \to DX$ that can be composed with the counit $\varepsilon_X\colon DX \to X$ to obtain the desired morphism from μF to X.



▶ Remark 17. Following Remark 16, when $\mathbf{C}$ is a complete lattice, a comonad D is simply a down-closure; a distributive law $\zeta\colon FD \Rightarrow DF$ witnesses that $FD \sqsubseteq DF$, namely that D is cocompatible with F. The algebra $a\colon FDX \to X$ is just an inductive invariant up-to D and $a^\flat\colon FDX \to DX$ is the corresponding inductive invariant provided by Proposition 6: observe that the three arrows in the definition of $a^\flat$ are exactly the three steps in the proof of Proposition 6. The morphism of $\varepsilon_X \circ (\!|a^\flat|\!)_F\colon \mu F \to X$ gives us the proof of Theorem 7. All this justifies the following statement.

▶ **Proposition 18.** *When* $\mathbf{C}$ *is a complete lattice, comonadic recursion is induction up-to.*

7.3 Course-of-Value Iteration

Course-of-value iteration is a recursion scheme that is obtained from comonadic recursion by taking D to be the *cofree comonad* generated by F. Below, we shortly recall this.

Coalgebras are the dual of algebras: a *coalgebra* for a functor $F\colon \mathbf{C} \to \mathbf{C}$ is a pair (X, a) with $a\colon X \to FX$. Given two F-coalgebras (X, a) and (Y, b), a coalgebra morphism $h\colon (X, a) \to (Y, b)$ is an arrow $h\colon X \to Y$ of $\mathbf{C}$ such that $Fh \circ a = b \circ h$. An F-coalgebra $(\nu F, \mathsf{a})$ is said to be *final* if for any F-coalgebra (X, a), there exists a unique coalgebra morphism $[\![a]\!]_F\colon (X, a) \to (\nu F, \mathsf{a})$.

For an object A of $\mathbf{C}$, we denote with $F_A\colon \mathbf{C} \to \mathbf{C}$ the functor mapping an object X into $FX \times A$ and arrow a into $Ff \times id_A$. Whenever F_A has a final coalgebra $\langle \mathsf{t}_A, \mathsf{o}_A\rangle\colon \nu F_A \to F(\nu F_A) \times A$ for all objects A, one can define a comonad $(F^{\downarrow}, \varepsilon^{\downarrow}, \delta^{\downarrow})$ on $\mathbf{C}$. The functor $F^{\downarrow}\colon \mathbf{C} \to \mathbf{C}$ maps an object X into the final coalgebra νF_X and an arrow $a\colon X \to Y$ into the unique final coalgebra morphism $[\![\langle \mathsf{t}_X, a \circ \mathsf{o}_X\rangle]\!]_{F_Y} : \nu F_X \to \nu F_Y$. For all objects X, the counit $\varepsilon^{\downarrow}\colon F^{\downarrow} \Rightarrow Id$ is defined as $\mathsf{o}_X\colon \nu F_X \to X$; the comultiplication $\delta^{\downarrow}\colon F^{\downarrow} \Rightarrow F^{\downarrow}F^{\downarrow}$ as $[\![\langle \mathsf{t}_X, id_{\nu F_X}\rangle]\!]_{F_{\nu F_X}} : \nu F_X \to \nu F_{\nu F_X}$.

Crucially, there exists a distributive law of F over $F^{\downarrow}$, denoted by $\zeta^{\downarrow}\colon FF^{\downarrow} \Rightarrow F^{\downarrow}F$, that is defined for all objects X as $[\![\langle F\pi_1, F\pi_2\rangle \circ F\langle \mathsf{t}_X, \mathsf{o}_X\rangle]\!]_{FFX} : F(\nu F_X) \to \nu F_{FX}$.

▶ Remark 19. Following Remarks 16 and 17, when $\mathbf{C}$ is a complete lattice, an F-coalgebra is a post-fixed point for F and a final F-algebra is a greatest fixed-point. The cofree comonad $F^{\downarrow}$ simplifies to the down-closure generated by the monotone map F, as defined in (4). The condition on the existence of a final F_A-coalgebra for all objects A trivially holds when $\mathbf{C}$ is a lattice. The existence of the distributive law $\zeta^{\downarrow}\colon FF^{\downarrow} \Rightarrow F^{\downarrow}F$ simplifies to Corollary 9. All this justifies the following statement.

▶ **Proposition 20.** *When* $\mathbf{C}$ *is a complete lattice, course of value iteration is strong induction.*

7.4 Back to Fibonacci

We conclude by illustrating course of value iteration in the case when F is the functor N introduced in Section 7.1. First, it is convenient to recall some auxiliary ingredients.

For a set A, we write A^{∞}_{ne} for the set of all finite and infinite sequences over A that are non-empty. For a sequence $\sigma \in A^{\infty}_{ne}$, we write $\sigma(0) \cdot \sigma(1) \cdot \ldots$ whenever σ is infinite and $\sigma(0) \cdot \sigma(1) \cdot \ldots \sigma(n) \cdot \epsilon$ whenever σ has length $n+1$. Appending ϵ at the end of the word is a convenient notation to avoid confusing an element $a \in A$ with the sequence $a \cdot \epsilon \in A^{\infty}_{ne}$. For all $\sigma \in A^{\infty}_{ne}$, we define $\mathsf{hd}\colon A^{\infty}_{ne} \to A$ and $\mathsf{tl}\colon A^{\infty}_{ne} \to 1 + A^{\infty}_{ne}$ as follows.

$$\mathsf{hd}(\sigma) \stackrel{\text{def}}{=} \sigma(0) \qquad \mathsf{tl}(\sigma) \stackrel{\text{def}}{=} \begin{cases} \sigma(1) \cdot \sigma(2) \cdot \cdots \cdot \sigma(n) \cdot \epsilon & \text{if } \sigma \text{ has length } n+1 \\ \sigma(1) \cdot \sigma(2) \cdot \cdots & \text{otherwise} \end{cases}$$

The pairing $\langle \mathsf{tl}, \mathsf{hd}\rangle\colon A^{\infty}_{ne} \to (1 + A^{\infty}_{ne}) \times A$ forms a coalgebra for the functor $N_A\colon \mathbf{Set} \to \mathbf{Set}$. Actually, $(A^{\infty}_{ne}, \langle \mathsf{tl}, \mathsf{hd}\rangle)$ is a final coalgebra for N_A: for all N_A-coalgebra $\langle t, o\rangle\colon X \to (1+X)\times A$ there exists a unique morphism making the following diagram on the left commute.

$$\begin{array}{ccc} X & \xrightarrow{\quad [\![\langle t,o\rangle]\!]_{N_A} \quad} & A^{\infty}_{ne} \\ {\scriptstyle \langle t,o\rangle}\downarrow & & \downarrow{\scriptstyle \langle \mathsf{tl},\mathsf{hd}\rangle} \\ (1+X)\times A & \xrightarrow[(id_1 + [\![\langle t,o\rangle]\!]_{N_A})\times id_A]{} & (1+A^{\infty}_{ne})\times A \end{array} \qquad \begin{array}{rcl} \mathsf{hd}(\,[\![\langle t,o\rangle]\!]_{N_A}(\sigma)\,) & = & o(\sigma) \\ \mathsf{tl}(\,[\![\langle t,o\rangle]\!]_{N_A}(\sigma)\,) & = & [\![\langle t,o\rangle]\!]_{N_A}(\,t(\sigma)\,) \end{array}$$

Commutation of the diagram is expressed by the conditions on the right: these provide a coinductive definition for $[\![\langle t, o\rangle]\!]_{N_A}$.

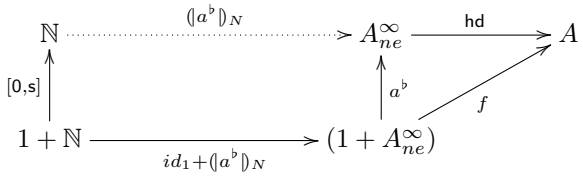
Since for all sets A there exists a final N_A-coalgebra, then there exists the cofree comonad $N^{\downarrow}$ and a distributive law $\zeta^{\downarrow}\colon NN^{\downarrow} \Rightarrow N^{\downarrow}N$ (more details in Appendix B). Most importantly, given a function $a\colon 1 + A^{\infty}_{ne} \to A$ (i.e, an $NN^{\downarrow}$-algebra), one can extend it to a function $a^{\flat}\colon 1 + A^{\infty}_{ne} \to A^{\infty}_{ne}$ (i.e, an N-algebra) coinductively defined for all $x \in 1 + A^{\infty}_{ne}$ as

$$\mathsf{hd}(\, a^{\flat}(x)\,) \stackrel{\text{def}}{=} a(x) \qquad \mathsf{tl}(\, a^{\flat}(x)\,) \stackrel{\text{def}}{=} \begin{cases} \epsilon & \text{if } x = \epsilon \\ a^{\flat}(\mathsf{tl}(x)) & \text{otherwise.} \end{cases} \tag{13}$$

By initiality of $(\mathbb{N}, [0, \mathsf{s}])$, one has a morphism $(\!|a^{\flat}|\!)_N$ inductively defined

$$\begin{array}{rcl} (\!|a^{\flat}|\!)_N(0) & = & a^{\flat}(\epsilon) \\ (\!|a^{\flat}|\!)_N(\mathsf{s}(n)) & = & a^{\flat}(\,(\!|a^{\flat}|\!)_N(n)\,) \end{array} \tag{14}$$

that can be composed with the counit hd to obtain the desired morphism $\mathbb{N} \to A$.



The inductive case in (14) can be conveniently rephrased (see Lemma 22 in Appendix B) as

$$(\!|a^{\flat}|\!)_N(\mathsf{s}(n)) = a(\,(\!|a^{\flat}|\!)_N(n)\,) \cdot (\!|a^{\flat}|\!)_N(n).$$

In summary, $(\!|a^{\flat}|\!)_N(0) = a(\epsilon) \cdot \epsilon$ and

$$(\!|a^{\flat}|\!)_N(\mathsf{s}(n)) = a((\!|a^{\flat}|\!)_N(n)) \cdot a((\!|a^{\flat}|\!)_N(n-1)) \cdot \ldots a((\!|a^{\flat}|\!)_N(0)) \cdot a(\epsilon) \cdot \epsilon.$$

Intuitively,$(\!|a^{\flat}|\!)_N(n+1)$ may depend on $(\!|a^{\flat}|\!)_N(n)$, $(\!|a^{\flat}|\!)_N(n-1)$ ...

For a concrete example take A to be $\mathbb{N}$ and a to be $fib\colon 1 + \mathbb{N}^{\infty}_{ne} \to \mathbb{N}$ defined as

$$fib(x) \stackrel{\text{def}}{=} \begin{cases} 1 & \text{if } x = \epsilon \text{ or } x \text{ has length } 1 \\ x(0) + x(1) & \text{otherwise} \end{cases}$$

for all $x \in 1 + \mathbb{N}^{\infty}_{ne}$. The reader can easily check that $(\!|fib^{\flat}|\!)_N(0) = 1 \cdot \epsilon$, $(\!|fib^{\flat}|\!)_N(1) = 1 \cdot 1 \cdot \epsilon$, $(\!|fib^{\flat}|\!)_N(2) = 2 \cdot 1 \cdot 1 \cdot \epsilon$ and so on. By composing $(\!|fib^{\flat}|\!)_N$ with $\mathsf{hd}\colon A^{\infty}_{ne} \to A$ is clear that one obtains the Fibonacci sequence recalled in Section 2.

8 Conclusions and future work

We have introduced induction up-to by dualising the framework of coinduction up-to from [28]. More precisely, we defined the notion of cocompatible functions (Definition 5) and proved that such functions provide sound up-to techniques (Theorem 7) that can be conveniently composed in various ways (Proposition 8). In particular, for any monotone function f, its downclosure $f^{\downarrow}$ is always f-cocompatible (Corollary 9). We refer to induction up-to $f^{\downarrow}$ as strong induction. Our main insight is that the well-known principle of strong induction over the natural numbers is induction up-to $b^{\downarrow}$ (Section 6.1), where b is the map having $\mathbb{N}$ as its least fixed point.

We then demonstrated that, by applying comonadic recursion [39] to lattices, we obtain exactly our theory of induction up-to (Proposition 18), while strong induction corresponds to the lattice-theoretic version of course-of-value iteration [38, 7] (Proposition 20).

We believe that these results shed light on the relationship between the schemes used to define programs and the techniques employed to prove their properties. It is no coincidence that, in Section 2, we required strong induction to prove properties of the Fibonacci sequence, which is defined by course-of-value iteration.

By dualising the results in [5], which enhance the fibrational framework of Hermida and Jacobs [18], we can distill compatible inductive up-to techniques from each comonad D. Verifying all the details is a substantial task that we leave for future work.

References

1 Roberto M Amadio and Luca Cardelli. Subtyping recursive types. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 15(4):575–631, 1993. doi:10.1145/155183.155231.

2 Falk Bartels. Generalised coinduction. *Mathematical Structures in Computer Science*, 13(2):321–348, 2003. doi:10.1017/S0960129502003900.

3 Filippo Bonchi, Pierre Ganty, Roberto Giacobazzi, and Dusko Pavlovic. Sound up-to techniques and complete abstract domains. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 175–184, 2018. doi:10.1145/3209108.3209169.

4 Filippo Bonchi, Barbara König, and Daniela Petrisan. Up-to techniques for behavioural metrics via fibrations. *Math. Struct. Comput. Sci.*, 33(4-5):182–221, 2023. doi:10.1017/s0960129523000166.

5 Filippo Bonchi, Daniela Petrisan, Damien Pous, and Jurriaan Rot. A general account of coinduction up-to. *Acta Informatica*, 54(2):127–190, 2017. doi:10.1007/S00236-016-0271-4.

6 Filippo Bonchi and Damien Pous. Checking NFA equivalence with bisimulations up to congruence. In Roberto Giacobazzi and Radhia Cousot, editors, *The 40th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL '13, Rome, Italy - January 23 - 25, 2013*, pages 457–468. ACM, 2013. doi:10.1145/2429069.2429124.

7 Daniela Cancila, Furio Honsell, and Marina Lenisa. Generalized coiteration schemata. *Electronic Notes in Theoretical Computer Science*, 82(1):76–93, 2003. doi:10.1016/S1571-0661(04)80633-9.

8 Venanzio Capretta, Tarmo Uustalu, and Varmo Vene. Recursive coalgebras from comonads. *Information and Computation*, 204(4):437–468, 2006. doi:10.1016/j.ic.2005.08.005.

9 Didier Caucal. Graphes canoniques de graphes algébriques. *RAIRO Theor. Informatics Appl.*, 24:339–352, 1990. doi:10.1051/ita/1990240403391.

10 Konstantinos Chatzikokolakis, Catuscia Palamidessi, and Valeria Vignudelli. Up-to techniques for generalized bisimulation metrics. In Josée Desharnais and Radha Jagadeesan, editors, *27th International Conference on Concurrency Theory, CONCUR 2016, August 23-26, 2016, Québec City, Canada*, volume 59 of *LIPIcs*, pages 35:1–35:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.CONCUR.2016.35.

11 Thierry Coquand. Infinite objects in type theory. In *International Workshop on Types for Proofs and Programs*, pages 62–78. Springer, 1993. doi:10.1007/3-540-58085-9_72.

12 Erik P. de Vink and Jan J.M.M. Rutten. Bisimulation for probabilistic transition systems: a coalgebraic approach. *Theoretical Computer Science*, 221(1):271–293, 1999. doi:10.1016/S0304-3975(99)00035-3.

13 Adrien Durier, Daniel Hirschkoff, and Davide Sangiorgi. Divergence and unique solution of equations. *Logical Methods in Computer Science*, 15, 2019. doi:10.23638/LMCS-15(3:12)2019.

14 Marco Forti and Furio Honsell. Set theory with free construction principles. *Annali della Scuola Normale Superiore di Pisa-Classe di Scienze*, 10(3):493–522, 1983.

15 Herman Geuvers and Bart Jacobs. Relating apartness and bisimulation. *Logical Methods in Computer Science*, 17, 2021. doi:10.46298/lmcs-17(3:15)2021.

16 Ichiro Hasuo, Bart Jacobs, and Ana Sokolova. Generic trace semantics via coinduction. *Logical Methods in Computer Science*, 3, 2007. doi:10.2168/LMCS-3(4:11)2007.

17 Ichiro Hasuo, Shunsuke Shimizu, and Corina Cîrstea. Lattice-theoretic progress measures and coalgebraic model checking. In *Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 718–732, 2016. doi:10.1145/2837614.2837673.

18 Claudio Hermida and Bart Jacobs. Structural induction and coinduction in a fibrational setting. *Information and computation*, 145(2):107–152, 1998. doi:10.1006/inco.1998.2725.

19 Arend Heyting. Zur intuitionistischen axiomatik der projektiven geometrie. *Mathematische Annalen*, 98(1):491–538, 1928.

20 Bart Jacobs, Alexandra Silva, and Ana Sokolova. Trace semantics via determinization. In *International Workshop on Coalgebraic Methods in Computer Science*, pages 109–129. Springer, 2012. doi:10.1007/978-3-642-32784-1_7.

21 Bartek Klin. Bialgebras for structural operational semantics: An introduction. *Theor. Comput. Sci.*, 412(38):5043–5069, 2011. doi:10.1016/j.tcs.2011.03.023.

22 Dexter Kozen and Alexandra Silva. Practical coinduction. *Mathematical Structures in Computer Science*, 27(7):1132–1152, 2017. doi:10.1017/S0960129515000493.

23 Robin Milner. *Communication and concurrency*, volume 84 of *PHI Series in computer science*. Prentice Hall, 1989.

24 Robin Milner and Mads Tofte. Co-induction in relational semantics. *Theoretical computer science*, 87(1):209–220, 1991. doi:10.1016/0304-3975(91)90033-X.

25 Keiko Nakata and Tarmo Uustalu. Resumptions, weak bisimilarity and big-step semantics for while with interactive I/O: An exercise in mixed induction-coinduction. In *In Proceedings of SOS 2010*, pages 57–75, 2010. doi:10.4204/EPTCS.32.5.

26 David Park. Fixpoint induction and proofs of program properties. *Machine intelligence*, 5, 1969.

27 David Park. Concurrency and automata on infinite sequences. In *Theoretical Computer Science: 5th GI-Conference Karlsruhe, March 23–25, 1981*, pages 167–183. Springer, 2005.

28 Damien Pous. Complete lattices and up-to techniques. In Zhong Shao, editor, *Programming Languages and Systems, 5th Asian Symposium, APLAS 2007, Singapore, November 29-December 1, 2007, Proceedings*, volume 4807 of *Lecture Notes in Computer Science*, pages 351–366. Springer, 2007. doi:10.1007/978-3-540-76637-7_24.

29 Damien Pous. Coinduction all the way up. In *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 307–316, 2016. doi:10.1145/2933575.2934564.

30 Damien Pous and Davide Sangiorgi. Enhancements of the bisimulation proof method. In Davide Sangiorgi and Jan J. M. M. Rutten, editors, *Advanced Topics in Bisimulation and Coinduction*, volume 52 of *Cambridge tracts in theoretical computer science*, pages 233–289. Cambridge University Press, UK, 2012.

31 Jan J.M.M. Rutten. Behavioural differential equations: a coinductive calculus of streams, automata, and power series. *Theoretical Computer Science*, 308(1):1–53, 2003. doi:10.1016/S0304-3975(02)00895-2.

32 Davide Sangiorgi. On the bisimulation proof method. *Mathematical Structures in Computer Science , Volume 8 , Issue 5*, 1998. doi:10.1017/S0960129598002527.

33 Davide Sangiorgi. On the origins of bisimulation and coinduction. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 31(4):1–41, 2009. doi:10.1145/1516507.1516510.

34 Davide Sangiorgi. Equations, contractions, and unique solutions. *ACM Transactions on Computational Logic (TOCL)*, 18(1):1–30, 2017. doi:10.1145/2971339.

35 Davide Sangiorgi. From enhanced coinduction towards enhanced induction. *Proceedings of the ACM on Programming Languages*, 6(POPL):1–29, 2022. doi:10.1145/3498679.

36 Alexandra Silva, Filippo Bonchi, Marcello M. Bonsangue, and Jan J. M. M. Rutten. Generalizing the powerset construction, coalgebraically. In Kamal Lodaya and Meena Mahajan, editors, *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer*

Science, FSTTCS 2010, December 15-18, 2010, Chennai, India, volume 8 of *LIPIcs*, pages 272–283. Schloss-Dagstuhl-Leibniz Zentrum für Informatik, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2010. doi:10.4230/LIPIcs.FSTTCS.2010.272.

37 Daniele Turi and Gordon Plotkin. Towards a mathematical operational semantics. In *Proceedings of Twelfth Annual IEEE Symposium on Logic in Computer Science*, pages 280–291. IEEE, 1997. doi:10.1109/LICS.1997.614955.

38 Tarmo Uustalu and Varmo Vene. Primitive (co) recursion and course-of-value (co) iteration, categorically. *Informatica*, 10(1):5–26, 1999. doi:10.3233/INF-1999-10102.

39 Tarmo Uustalu, Varmo Vene, and Alberto Pardo. Recursion schemes from comonads. *Nordic Journal of Computing*, 8(3):366–390, 2001. URL: http://www.cs.helsinki.fi/njc/References/uustaluvp2001:366.html.

40 Johan Van Benthem. *Modal logic and classical logic*. 1983.

41 Rob J van Glabbeek. The linear time - branching time spectrum II: The semantics of sequential systems with silent moves extended abstract. In *International Conference on Concurrency Theory*, pages 66–81. Springer, 1993.

42 Rob J Van Glabbeek. The linear time - branching time spectrum I. In Jan A. Bergstra, Alban Ponse, and Scott A. Smolka, editors, *Handbook of Process Algebra*, pages 3–99. North-Holland / Elsevier, 2001. doi:10.1016/b978-044482830-9/50019-9.

A Appendix to Section 5

Proof of Proposition 8. We prove in sequence each point.

1. $f \circ id_X = f = id_X \circ f$;
2. $f \circ f = f \circ f$;
3. By the following derivation

$$\begin{aligned} f \circ d \circ e &\sqsubseteq d \circ f \circ e && (d \text{ is } f-\textit{compatible}) \\ &\sqsubseteq d \circ e \circ f && (e \text{ is } f-\textit{compatible} \text{ and } d \text{ is monotone}) \end{aligned}$$

4. By induction.
   Base case: $n = 0$. By (3) and point 1.
   Inductive case: $n = n + 1$. By hypothesis d is f-compatible; by induction hypothesis d^n is f-compatible; by point 3, $d \circ d^n$ is f-compatible. By definition, see (3), $d^{n+1} = d \circ d^n$. Thus d^{n+1} is f-compatible.
5. Proving $f(d \sqcap e) \sqsubseteq (d \sqcap e)f$ means proving that: for all $x \in X$,

   $$f(d \sqcap e)(x) \sqsubseteq (d \sqcap e)f(x).$$

   Since, for all $x \in X$,

   $$d \sqcap e(x) = d(x) \sqcap e(x),$$

   it holds that:

   $$\begin{aligned} f(d \sqcap e)(x) &= f(d(x) \sqcap e(x)) \\ (d \sqcap e)f(x) &= df(x) \sqcap ef(x) \end{aligned}$$

   In summary, to prove $f(d \sqcap e) \sqsubseteq (d \sqcap e)f$, we need to prove that, for all $x \in X$, $f(d(x) \sqcap e(x)) \sqsubseteq df(x) \sqcap ef(x)$. We can proceed separately:

- First, $f(d(x) \sqcap e(x)) \sqsubseteq df(x)$:

$$\begin{aligned} f(d(x) \sqcap e(x)) &\sqsubseteq fd(x) && (d(x) \sqcap e(x) \sqsubseteq d(x)) \\ &\sqsubseteq df(x) && (d \text{ is } f-\textit{compatible}) \end{aligned}$$

- Similarly for $f(d(x) \sqcap e(x)) \sqsubseteq df(x)$.

Since $f(d(x) \sqcap e(x))$ is below both $df(x)$ and $ef(x)$, we have that

$$f(d(x) \sqcap e(x)) \sqsubseteq df(x) \sqcap ef(x).$$

6. The proof is analogous to the one of Point 5. We need to prove that, for all $x \in X$,

$$f \bigsqcap_{i \in \mathbb{N}} d_i(x) \sqsubseteq \bigsqcap_{i \in \mathbb{N}} d_i f(x)$$

Thus, it is enough to prove that $\forall i \in \mathbb{N}$, $f \bigsqcap_j d_j(x) \sqsubseteq d_i f(x)$. We proceed as follows:

$$\begin{aligned} f \bigsqcap_{j \in \mathbb{N}} d_j(x) &\sqsubseteq f d_i(x) && (\textstyle\bigsqcap_{j \in \mathbb{N}} d_j \sqsubseteq d_i) \\ &\sqsubseteq d_i f(x) && (d_i \text{ is } f-\textit{compatible}) \end{aligned}$$

7. By (4) and Lemmas 4 and 6. ◀

Proof of 11. We prove the two points separately.

1. Assume that d is an up-closure; Thus $\neg x \sqsubseteq d(\neg x)$ and $d(d(\neg x)) \sqsubseteq d(\neg x)$ for all $x \in L$; By (9), $\neg\neg x \sqsupseteq \neg d(\neg x)$ and $\neg d(d(\neg x)) \sqsupseteq \neg d(\neg x)$, i.e., $x \sqsupseteq \overline{d}(x)$ and $\overline{d}\,\overline{d}(x) \sqsupseteq \overline{d}(x)$. The reverse implication has the same proof.
2. Observe that for all monotone maps i, j on L, it holds that $i \sqsubseteq j$ iff $i\neg \sqsubseteq j\neg$ iff $\neg j \sqsubseteq \neg i$. Thus

$$\begin{aligned} df \sqsubseteq fd &\Leftrightarrow df\neg \sqsubseteq fd\neg \\ &\Leftrightarrow \neg fd\neg \sqsubseteq \neg df\neg \\ &\Leftrightarrow \neg f\neg\neg d\neg \sqsubseteq \neg d\neg\neg f\neg \\ &\Leftrightarrow \overline{f}\,\overline{d} \sqsubseteq \overline{d}\,\overline{f} \end{aligned}$$

3. By the following derivation

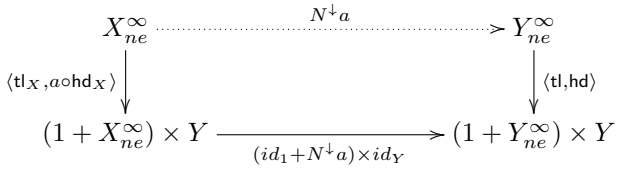
$$\begin{aligned} y \sqsubseteq fd(y) &\Leftrightarrow \neg fd(y) \sqsubseteq \neg y \\ &\Leftrightarrow \neg f\neg\, \neg d\neg(\neg y) \sqsubseteq \neg y \\ &\Leftrightarrow \overline{f}\,\overline{d}(\neg y) \sqsubseteq \neg y \end{aligned}$$

◀

B Details on Section 7.4

In Section 7.3, we give the recipe to compute the cofree comonad for an arbitrary functor F and in Section 7.4 we used the cofree comonad for the functor $N\colon \mathbf{Set} \to \mathbf{Set}$ in order to illustrate course-of-value Iteration for the natural numbers. For reader convenience, in this appendix we illustrate in details the cofree comonad $(N^{\downarrow}, \epsilon^{\downarrow}, \delta^{\downarrow})$ and the distributive law $\zeta\colon NN^{\downarrow} \Rightarrow N^{\downarrow}N$ by simply unfolding the definitions of Section 7.3.

First we illustrate the endofunctor $N^{\downarrow}\colon \mathbf{Set} \to \mathbf{Set}$. It maps any set X into the set X^{∞}_{ne} since, as discussed in the main text, $(X^{\infty}_{ne}, \langle \mathsf{tl}, \mathsf{hd} \rangle)$ is a final N_X coalgebra. For a function $a\colon X \to Y$, $N^{\downarrow}a\colon X^{\infty}_{ne} \to Y^{\infty}_{ne}$ is the unique map from the N_Y-coalgebra $(X^{\infty}_{ne}, \langle \mathsf{tl}_X, a \circ \mathsf{hd}_X \rangle)$ to the final N_Y-coalgebra $(Y^{\infty}_{ne}, \langle \mathsf{tl}, \mathsf{hd} \rangle)$ as illustrated below.



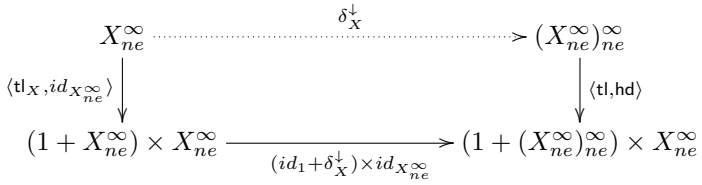
Thus, the function $N^\downarrow a \colon X^\infty_{ne} \to Y^\infty_{ne}$ can be defined conductively as follows.

$$\begin{aligned} \mathsf{hd}(\, N^\downarrow a(\sigma)\,) &= a(\mathsf{hd}_X(\sigma)) \\ \mathsf{tl}(\, N^\downarrow a(\sigma)\,) &= \begin{cases} \epsilon & \text{if } \mathsf{tl}_X(\sigma) = \epsilon; \\ N^\downarrow a(\mathsf{tl}_X(\sigma)) & \text{otherwise.} \end{cases} \end{aligned}$$

More explicitly, $N^\downarrow a$ maps $\sigma \in X^\infty_{ne}$ into

$$a(\sigma(0)) \cdot a(\sigma(1)) \cdot \ldots$$

We can now illustrate the natural transformations. The counit $\epsilon \colon N^\downarrow \Rightarrow Id$ is given, for all sets X, by $\mathsf{hd}_X \colon X^\infty_{ne} \to X$. The comultiplication $\delta^\downarrow \colon N^\downarrow \Rightarrow N^\downarrow N^\downarrow$ is given by the unique morphism from the $N_{X^\infty_{ne}}$-coalgebra $(X^\infty_{ne}, \langle \mathsf{tl}_X, id_{X^\infty_{ne}} \rangle)$ to the final $N_{X^\infty_{ne}}$-coalgebra $((X^\infty_{ne})^\infty_{ne}, \langle \mathsf{tl}, \mathsf{hd} \rangle)$, as illustrated below.



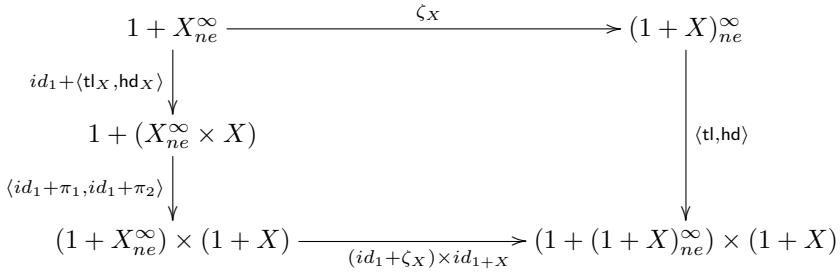
Thus, the function $\delta^\downarrow_X \colon X^\infty_{ne} \to (X^\infty_{ne})^\infty_{ne}$ can be coinductively defined as follows.

$$\begin{aligned} \mathsf{hd}(\, \delta^\downarrow_X(\sigma)\,) &= \sigma \\ \mathsf{tl}(\, \delta^\downarrow_X(\sigma)\,) &= \begin{cases} \epsilon & \text{if } \mathsf{tl}_X(\sigma) = \epsilon; \\ \delta^\downarrow_X(\mathsf{tl}_X(\sigma)) & \text{otherwise.} \end{cases} \end{aligned}$$

Intuitively, $\sigma \in X^\infty_{ne}$ is mapped into

$$\begin{array}{llll} \sigma(0) & \sigma(1) & \sigma(2) & \ldots \\ \sigma(1) & \sigma(2) & \ldots & \\ \sigma(2) & \ldots & & \\ \vdots & & & \end{array}$$

So far, we have described the comonad $(N^\downarrow, \epsilon^\downarrow, \delta^\downarrow)$. We can now move to the distributive law. For all sets X, the distributive law $\zeta \colon NN^\downarrow \Rightarrow N^\downarrow N$ is given by the unique morphism from the N_{1+X}-coalgebra illustrated below on the left to the final N_{1+X}-coalgebra $((1 + X)^\infty_{ne}, \langle \mathsf{tl}, \mathsf{hd} \rangle)$.



Thus, the function $\zeta_X \colon 1 + X^\infty_{ne} \to (1+X)^\infty_{ne}$ can be coinductively defined as follows.

$$\begin{aligned}
\mathsf{hd}(\,\zeta_X(\epsilon)\,) &= \epsilon \\
\mathsf{tl}(\,\zeta_X(\epsilon)\,) &= \epsilon \\
\mathsf{hd}(\,\zeta_X(\sigma)\,) &= \mathsf{hd}_X(\sigma) \\
\mathsf{tl}(\,\zeta_X(\sigma)\,) &= \begin{cases} \epsilon & \text{if } \mathsf{tl}_X(\sigma) = \epsilon; \\ \zeta_X(\mathsf{tl}_X(\sigma)) & \text{otherwise.} \end{cases}
\end{aligned}$$

Intuitively $[\![h]\!]_{1+X}$ maps ϵ into the sequence $\epsilon \cdot \epsilon$ and any $\sigma \in X^\infty_{ne}$ into the same sequence $\sigma \in (1+X)^\infty_{ne}$.

We conclude this appendix with some computation that allows for the handier characterisation of $(\!|a^\flat|\!)_N$ illustrated in the main text.

▶ **Lemma 21.** *For all $n \in \mathbb{N}$, $a^\flat(\mathsf{tl}(\,(\!|a^\flat|\!)_N(n)\,)) = (\!|a^\flat|\!)_N(n)$.*

Proof. By induction on $\mathbb{N}$.

For 0,

$$\begin{aligned}
a^\flat(\mathsf{tl}(\,(\!|a^\flat|\!)_N(0)\,)) &= a^\flat(\mathsf{tl}(\,a^\flat(\epsilon)\,)) && (14)\\
&= a^\flat(\epsilon) && (13)\\
&= (\!|a^\flat|\!)_N(0) && (14)
\end{aligned}$$

For $n+1$,

$$\begin{aligned}
a^\flat(\mathsf{tl}(\,(\!|a^\flat|\!)_N(n+1)\,)) &= a^\flat(\mathsf{tl}(\,a^\flat(\!|a^\flat|\!)_N(n)\,)) && (14)\\
&= a^\flat(a^\flat(\,\mathsf{tl}((\!|a^\flat|\!)_N(n)\,))) && (13)\\
&= a^\flat((\!|a^\flat|\!)_N(n)) && \text{(Induction Hypothesis)}\\
&= (\!|a^\flat|\!)_N(n+1) && (14)
\end{aligned}$$

◀

▶ **Lemma 22.** *For all $n \in \mathbb{N}$, $(\!|a^\flat|\!)_N(\mathsf{s}(n)) = a(\,(\!|a^\flat|\!)_N(n)\,) \cdot (\!|a^\flat|\!)_N(n)$.*

Proof.

$$\begin{aligned}
(\!|a^\flat|\!)_N(\mathsf{s}(n)) &= a^\flat(\,(\!|a^\flat|\!)_N(\mathsf{s}(n))\,) && (14)\\
&= \mathsf{hd}(a^\flat(\,(\!|a^\flat|\!)_N(\mathsf{s}(n))\,)) \cdot \mathsf{tl}(a^\flat(\,(\!|a^\flat|\!)_N(\mathsf{s}(n))\,)) \\
&= a(\,(\!|a^\flat|\!)_N(\mathsf{s}(n))\,) \cdot a^\flat(\mathsf{tl}(\,(\!|a^\flat|\!)_N(\mathsf{s}(n))\,)) && (13)\\
&= a(\,(\!|a^\flat|\!)_N(\mathsf{s}(n))\,) \cdot (\!|a^\flat|\!)_N(n) && \text{(Lemma 21)}
\end{aligned}$$

◀