Math. Struct. in Comp. Science: page 1 of 21. © Cambridge University Press 2016
doi:10.1017/S0960129515000493

Practical coinduction

DEXTER KOZEN[†] and ALEXANDRA SILVA[‡]

[†]*Computer Science, Cornell University, Ithaca, New York 14853-7501, U.S.A.*
Email: `kozen@cs.cornell.edu`
[‡]*Intelligent Systems, Radboud University Nijmegen, Postbus 9010, 6500 GL Nijmegen, the Netherlands*
Email: `alexandra@cs.ru.nl`

Received 6 March 2013; revised 17 October 2014

Induction is a well-established proof principle that is taught in most undergraduate programs in mathematics and computer science. In computer science, it is used primarily to reason about inductively defined datatypes such as finite lists, finite trees and the natural numbers. Coinduction is the dual principle that can be used to reason about coinductive datatypes such as infinite streams or trees, but it is not as widespread or as well understood. In this paper, we illustrate through several examples the use of coinduction in informal mathematical arguments. Our aim is to promote the principle as a useful tool for the working mathematician and to bring it to a level of familiarity on par with induction. We show that coinduction is not only about bisimilarity and equality of behaviors, but also applicable to a variety of functions and relations defined on coinductive datatypes.

1. Introduction

Perhaps the single most important general proof principle in computer science, and arguably in all of mathematics, is *induction*. There is a valid induction principle corresponding to any well-founded relation, but in computer science, it is most often seen in the form known as *structural induction*, in which the domain of discourse is an inductively-defined datatype such as finite lists, finite trees, or the natural numbers.

For example, consider the type $\mathsf{List\ of\ } A$ of finite lists over an alphabet A, defined inductively by

— $\mathsf{nil} \in \mathsf{List\ of\ } A$
— if $a \in A$ and $\ell \in \mathsf{List\ of\ } A$, then $a :: \ell \in \mathsf{List\ of\ } A$.

The defined datatype is the least solution of the equation

$$\mathsf{List\ of\ } A = \mathsf{nil} + A \times \mathsf{List\ of\ } A. \tag{1.1}$$

It is the initial algebra for a signature consisting of one constant (nil) and one binary constructor ($::$). This means that one can define functions with domain $\mathsf{List\ of\ } A$ uniquely by structural induction. For example, the functions length, which computes the length of a finite list, and concat, which concatenates two finite lists, can be defined as follows:

$$\mathsf{length}(\mathsf{nil}) = 0 \qquad\qquad \mathsf{concat}(\mathsf{nil}, \ell) = \ell$$
$$\mathsf{length}(a :: \ell) = 1 + \mathsf{length}(\ell) \qquad\qquad \mathsf{concat}(a :: \ell_1, \ell_2) = a :: \mathsf{concat}(\ell_1, \ell_2).$$

No one would dispute that these functions are uniquely defined. Now, we can prove that $\mathsf{length}(\mathsf{concat}(\ell_1, \ell_2)) = \mathsf{length}(\ell_1) + \mathsf{length}(\ell_2)$ by structural induction on the first argument.

$$\begin{aligned}
\mathsf{length}(\mathsf{concat}(\mathsf{nil}, \ell_2)) &= \mathsf{length}(\ell_2),\\
&= 0 + \mathsf{length}(\ell_2) = \mathsf{length}(\mathsf{nil}) + \mathsf{length}(\ell_2),\\[1ex]
\mathsf{length}(\mathsf{concat}(a :: \ell_1, \ell_2)) &= \mathsf{length}(a :: \mathsf{concat}(\ell_1, \ell_2)),\\
&= 1 + \mathsf{length}(\mathsf{concat}(\ell_1, \ell_2)),\\
&= 1 + \mathsf{length}(\ell_1) + \mathsf{length}(\ell_2), \qquad \text{(inductive step)}\\
&= \mathsf{length}(a :: \ell_1) + \mathsf{length}(\ell_2).
\end{aligned}$$

This proof would raise no objections as to its correctness. The induction principle in play here is implicit and trusted; there is no need to reassert its validity every time it is used or whenever a new inductive datatype is introduced.

Coinduction, on the other hand, is still mysterious and unfamiliar to many. Coinduction is the dual principle to induction and is used to prove properties of coinductively-defined datatypes such as infinite streams, infinite trees, and coterms. These datatypes are typically final coalgebras for a signature. For example, the finite and infinite streams over A form the final coalgebra for the signature $(\mathsf{nil}, ::)$ and are the greatest solution of Equation (1.1).

Although coinduction has been around for decades, many proofs in the literature that rely on coinduction still end up essentially reasserting the principle every time it is used. It is clearly not as familiar as induction and not trusted in the same way. Quoting Rutten from his seminal paper on universal coalgebra:

Firstly, induction principles are well known and much used. The coinductive definition and proof principles for coalgebras are less well known by far, and often even not very clearly formulated.
– Rutten (2000)

Rutten's paper was the precursor of much work on coalgebra and coinduction, which included, among many others, extensions to modal logics (Kurz 2001; Schröder 2005, 2008; Schröder and Pattinson 2007) and structural operational semantics (Klin 2007; Turi and Plotkin 1997). However, most attention has been devoted to bisimulation proofs of equality between coinductively defined objects. With only a handful of exceptions, e.g. Brandt and Henglein (1998); Hermida and Jacobs (1998); Milner and Tofte (1991); Niqui and Rutten (2009), not much has been explored when it comes to properties of other relations on coinductive datatypes besides equality.

Our aim in this paper is to introduce an informal style of coinductive reasoning that can be quite useful in dealing with infinite data. We illustrate this style with a number of interesting examples. Our arguments may seem a bit magical at first, because they apply to infinite objects and look something like induction without a basis. Nevertheless, they are backed by sound formal proof principles. The reason they work is summed up in the following motto:

A property holds by induction if there is good reason for it to hold; whereas a property holds by coinduction if there is no good reason for it not to hold.

Although there is a *coinductive step* but no basis, any difficulty that would arise that would cause the property not to hold would manifest itself in the attempt to prove the coinductive step.

The examples we give in the paper demonstrate the versatility of the principle. We will prove properties of several kinds:

— *Classical bisimulation proofs.* For example, given two coinductively defined streams, are they equal?
— *Properties other than equality.* For example, given two streams σ and τ over $\mathbb{N}$, is σ lexicographically less than τ?
— *Properties of relations on coinductive datatypes.* For example, is the subtype order on recursive types transitive?
— *Properties of functions between coinductive datatypes.* For example, given two coinductively defined partial orders and a function between them, is the function monotone?

In all these examples, the proofs we give are quite short and involve establishing a *coinductive step* analogous to the inductive step in proofs by induction. What is missing is the final argument that the proof is a valid application of the coinduction principle; but it is not necessary to include this step for the same reason that it is not necessary to argue with every inductive proof that the proof is a valid application of the induction principle.

We emphasize that we are not claiming to introduce any new coinductive proof principles. The foundations of coinduction underlying our approach are well known. Rather, our purpose is only to present an informal style of coinductive reasoning that can be used in everyday mathematics, much as induction is used today.

We hope that this paper will be of interest both to experts in coalgebra and coinduction by pointing out nonstandard examples of proofs by coinduction and to nonexperts by showing how coinduction can be used in an informal way to prove interesting properties from the realm of functional and imperative programming.

2. Coinductive datatypes

Coinductive datatypes provide a wealth of examples from functional programming. Coinductive datatypes usually refer to possibly infinite structures. Prime examples include infinite streams, infinite trees, coterms (infinite terms), and finite and infinite words over an alphabet. In programming language semantics, coinductive types are often used to model traces (Ichiro *et al.* 2007), recursive types (Brandt and Henglein 1998), and program state (Jeannin and Kozen 2012).

Formally, coinductive datatypes can be defined as elements of a final coalgebra for a given polynomial endofunctor on Set. For instance, the set A^ω of infinite streams over an alphabet A is (the carrier of) the final coalgebra of the functor $FX = A \times X$, whereas the set A^∞ of finite and infinite words over an alphabet A is the final coalgebra of $FX = \mathbb{1} + A \times X$.

Many functional programming languages such as Haskell and OCaml support coinductive types; Standard ML and F# do not. The type of streams would be defined in

Haskell as

$$\mathsf{data\ Stream\ a = S\ a\ (Stream\ a)}.$$

Here data is a keyword and S is a constructor. The type Stream a is polymorphic, parameterized by the type variable a.

Coinductive datatypes are usually presented together with their *destructors*. For instance, streams admit two operations $\mathsf{hd} : A^\omega \to A$ and $\mathsf{tl} : A^\omega \to A^\omega$, which in Haskell would be defined as

$$\mathsf{hd\ (S\ a\ as) = a} \qquad\qquad \mathsf{tl\ (S\ a\ as) = as}.$$

The existence of destructors is a consequence of the fact that A^ω is a coalgebra for the functor $FX = A \times X$. All such coalgebras come equipped with a *structure map* $\langle \mathsf{obs}, \mathsf{cont} \rangle : X \to A \times X$; for A^ω, $\mathsf{obs} = \mathsf{hd}$ and $\mathsf{cont} = \mathsf{tl}$. Interestingly enough, the structure map of a final coalgebra is always an isomorphism, as is the structure map of an initial algebra. This is the content of *Lambek's lemma* (Lambek 1968). Thus, initial algebras and final coalgebras are always both algebras and coalgebras for the same functor. In the case of streams, the inverse of $\langle \mathsf{hd}, \mathsf{tl} \rangle$, usually referred to as the *constructor*, is cons, a function of type $A \times A^\omega \to A^\omega$. In Haskell, it would be defined as

$$\mathsf{cons\ (a,as) = S\ a\ as}.$$

3. Some motivating examples

3.1. *Lexicographic order on streams*

In this section, we give an informal proof that lexicographic order on streams is transitive. The argument illustrates an informal style of coinductive reasoning in a nonstandard setting. At first glance, this technique seems quite magical because it appears to involve induction on a non-well-founded relation.

Let $(A, \leqslant)$ be a partially ordered alphabet. An *A-stream* is an element of A^ω. The constructor :: (cons) of type $A \times A^\omega \to A^\omega$ and corresponding destructors $\mathsf{hd} : A^\omega \to A$ and $\mathsf{tl} : A^\omega \to A^\omega$ are defined as in Section 1. The ordering $\leqslant_{\text{lex}}$ on A-streams is defined to be the *maximum* relation $R \subseteq A^\omega \times A^\omega$ satisfying the following property:

Property 1. If $\sigma R \tau$, then

i. $\mathsf{hd}(\sigma) \leqslant \mathsf{hd}(\tau)$, and
ii. if $\mathsf{hd}(\sigma) = \mathsf{hd}(\tau)$, then $\mathsf{tl}(\sigma) R\, \mathsf{tl}(\tau)$.

The relation $\leqslant_{\text{lex}}$ exists and is unique, and any relation satisfying Property 1 is a subset. This is because if $\{R_\alpha\}$ is any indexed family of relations satisfying Property 1, then their union $\bigcup_\alpha R_\alpha$ also satisfies Property 1. The relation $\leqslant_{\text{lex}}$ is thus the union of all relations satisfying Property 1.

The relation $\leqslant_{\text{lex}}$ satisfies many desirable properties. For example, $\leqslant_{\text{lex}}$ is reflexive, that is, $\sigma \leqslant_{\text{lex}} \sigma$ holds for any A-stream σ, because the identity relation $\mathsf{id} = \{(\sigma, \sigma) \mid \sigma \in A^\omega\}$ satisfies Property 1, therefore $\mathsf{id} \subseteq \leqslant_{\text{lex}}$.

Moreover, because $\leqslant_{\text{lex}}$ is maximum, the converse of Property 1 holds for $\leqslant_{\text{lex}}$; that is, if

i. $\mathsf{hd}(\sigma) \leqslant \mathsf{hd}(\tau)$, and
ii. $\mathsf{hd}(\sigma) = \mathsf{hd}(\tau) \Rightarrow \mathsf{tl}(\sigma) \leqslant_{\text{lex}} \mathsf{tl}(\tau)$,

then $\sigma \leqslant_{\text{lex}} \tau$. If not, then $\leqslant_{\text{lex}}$ would not be maximal; one could add the pair (σ, τ) to $\leqslant_{\text{lex}}$ without violating Property 1.

To say that $\leqslant_{\text{lex}}$ is the maximum relation satisfying Property 1 says that it is the greatest fixpoint of the operator.

$$T_{\leqslant_{\text{lex}}}(R) = \{(\sigma, \tau) \mid \mathsf{hd}(\sigma) \leqslant \mathsf{hd}(\tau) \text{ and } \mathsf{hd}(\sigma) = \mathsf{hd}(\tau) \Rightarrow \mathsf{tl}(\sigma) R\, \mathsf{tl}(\tau)\}.$$

Formally, the relation $\leqslant_{\text{lex}}$ is defined as the greatest fixpoint of $T_{\leqslant_{\text{lex}}}$; in symbols, $\leqslant_{\text{lex}} = \nu X.T_{\leqslant_{\text{lex}}}(X)$.

Now we will show

Theorem 1. The relation $\leqslant_{\text{lex}}$ is transitive.

Proof. We want to show that if $\sigma \leqslant_{\text{lex}} \rho \leqslant_{\text{lex}} \tau$, then $\sigma \leqslant_{\text{lex}} \tau$. Suppose

$$\sigma \leqslant_{\text{lex}} \rho \leqslant_{\text{lex}} \tau. \tag{3.2}$$

By Property 1(i),

$$\mathsf{hd}(\sigma) \leqslant \mathsf{hd}(\rho) \leqslant \mathsf{hd}(\tau). \tag{3.3}$$

Since $\leqslant$ is transitive on A, $\mathsf{hd}(\sigma) \leqslant \mathsf{hd}(\tau)$. Thus, Property 1(i) holds for the pair σ, τ.

For Property 1(ii), if $\mathsf{hd}(\sigma) = \mathsf{hd}(\tau)$, then $\mathsf{hd}(\sigma) = \mathsf{hd}(\rho) = \mathsf{hd}(\tau)$ by Equation (3.3) and the antisymmetry of $\leqslant$ on A. By the assumption (3.2) and Property 1(ii), $\mathsf{tl}(\sigma) \leqslant_{\text{lex}} \mathsf{tl}(\rho) \leqslant_{\text{lex}} \mathsf{tl}(\tau)$. By the coinduction hypothesis, $\mathsf{tl}(\sigma) \leqslant_{\text{lex}} \mathsf{tl}(\tau)$. This establishes Property 1(ii) for σ, τ.

We have shown that under the assumption (3.2) and the coinduction hypotheses on the tails, both clauses (i) and (ii) of Property 1 hold for the pair σ, τ. By the converse of Property 1, $\sigma \leqslant_{\text{lex}} \tau$. □

The part of this proof that is unsettling is the appeal to the coinduction hypothesis on the tails of the two streams. Streams are infinite, and there is nothing like a basis. So the entire argument seems non-well-founded. But as we will show later, the argument is quite firmly grounded. Intuitively, one can appeal to the coinductive hypothesis as long as there has been progress in observing the elements of the stream (guardedness) and there is no further analysis of the tails (opacity). We will explain this formally in Section 4.

There are of course other ways to prove transitivity of $\leqslant_{\text{lex}}$. Here is an informal proof by induction that is dual to the proof presented above.

Proof of Theorem 1 (alternative). We show the contrapositive: For any σ, ρ, τ, if $\sigma \not\leqslant_{\text{lex}} \tau$, then either $\sigma \not\leqslant_{\text{lex}} \rho$ or $\rho \not\leqslant_{\text{lex}} \tau$. We proceed by induction on the inductive definition of $\not\leqslant_{\text{lex}}$.[†]

If $\sigma \not\leqslant_{\text{lex}} \tau$ because of (i), then $\mathsf{hd}(\sigma) \not\leqslant \mathsf{hd}(\tau)$, therefore either $\mathsf{hd}(\sigma) \not\leqslant \mathsf{hd}(\rho)$ or $\mathsf{hd}(\rho) \not\leqslant \mathsf{hd}(\tau)$, since $\leqslant$ is transitive on A. Then either $\sigma \not\leqslant_{\text{lex}} \rho$ or $\rho \not\leqslant_{\text{lex}} \tau$ by (i). This is the basis.

If $\sigma \not\leqslant_{\text{lex}} \tau$ because of (ii), then $\mathsf{hd}(\sigma) = \mathsf{hd}(\tau)$ and $\mathsf{tl}(\sigma) \not\leqslant_{\text{lex}} \mathsf{tl}(\tau)$, and $\mathsf{tl}(\sigma) \not\leqslant_{\text{lex}} \mathsf{tl}(\tau)$ was determined at an earlier stage in the inductive definition. By the induction hypothesis, either $\mathsf{tl}(\sigma) \not\leqslant_{\text{lex}} \mathsf{tl}(\rho)$ or $\mathsf{tl}(\rho) \not\leqslant_{\text{lex}} \mathsf{tl}(\tau)$, say the former without loss of generality. If either $\mathsf{hd}(\sigma) \not\leqslant \mathsf{hd}(\rho)$ or $\mathsf{hd}(\rho) \not\leqslant \mathsf{hd}(\tau)$, we are done as above. Otherwise $\mathsf{hd}(\sigma) \leqslant \mathsf{hd}(\rho) \leqslant \mathsf{hd}(\tau)$, and since $\mathsf{hd}(\sigma) = \mathsf{hd}(\tau)$, we have $\mathsf{hd}(\sigma) = \mathsf{hd}(\rho) = \mathsf{hd}(\tau)$. Since $\mathsf{tl}(\sigma) \not\leqslant_{\text{lex}} \mathsf{tl}(\rho)$, we have $\sigma \not\leqslant_{\text{lex}} \rho$ by (ii). □

In the latter proof, we are actually doing induction on the relation

$$\{((\mathsf{tl}(\sigma), \mathsf{tl}(\tau)), (\sigma, \tau)) \mid \mathsf{hd}(\sigma) = \mathsf{hd}(\tau)\},$$

which is well founded on the set $\not\leqslant_{\text{lex}}$. One can show that $\sigma \not\leqslant_{\text{lex}} \tau$ iff there exists $n \geqslant 0$ such that $\mathsf{hd}(\mathsf{tl}^m(\sigma)) = \mathsf{hd}(\mathsf{tl}^m(\tau))$ for $m < n$ and $\mathsf{hd}(\mathsf{tl}^n(\sigma)) \not\leqslant \mathsf{hd}(\mathsf{tl}^n(\tau))$. The smallest such n is the stage in the inductive definition of $\not\leqslant_{\text{lex}}$ at which $\sigma \not\leqslant_{\text{lex}} \tau$ is established.

A third alternative would show that the relation $\{(\sigma, \tau) \mid \exists \rho \ \sigma \leqslant_{\text{lex}} \rho \leqslant_{\text{lex}} \tau\}$ satisfies Property 1, therefore is contained in the maximal such relation $\leqslant_{\text{lex}}$. The details of this argument, written out, would contain all the same ingredients as our other two proofs.

Here is another example involving lexicographic order on streams.

Theorem 2. For streams over a commutative semigroup $(A, +)$, pointwise addition is monotone; that is,

$$\sigma \leqslant_{\text{lex}} \tau \text{ and } \rho \leqslant_{\text{lex}} \pi \quad \Rightarrow \quad \sigma + \rho \leqslant_{\text{lex}} \tau + \pi,$$

where $\sigma + \tau$ is the pointwise sum of the two streams.

Proof. First observe that the pointwise sum operation $+$ on streams satisfies the equations

$$\mathsf{hd}(\sigma + \tau) = \mathsf{hd}(\sigma) + \mathsf{hd}(\tau) \qquad \mathsf{tl}(\sigma + \tau) = \mathsf{tl}(\sigma) + \mathsf{tl}(\tau). \tag{3.4}$$

By Property 1(i),

$$\mathsf{hd}(\sigma + \rho) = \mathsf{hd}(\sigma) + \mathsf{hd}(\rho) \leqslant \mathsf{hd}(\tau) + \mathsf{hd}(\pi) = \mathsf{hd}(\tau + \pi).$$

Thus, Property 1(i) holds for the pair $(\sigma + \rho, \tau + \pi)$.

For Property 1(ii), if $\mathsf{hd}(\sigma + \rho) = \mathsf{hd}(\tau + \pi)$ and using the fact that, by hypothesis, $\mathsf{hd}(\sigma) \leqslant \mathsf{hd}(\tau)$ and $\mathsf{hd}(\rho) \leqslant \mathsf{hd}(\pi)$, then we can conclude that $\mathsf{hd}(\sigma) = \mathsf{hd}(\tau)$ and $\mathsf{hd}(\rho) = \mathsf{hd}(\pi)$. By the assumptions $\sigma \leqslant_{\text{lex}} \tau$ and $\rho \leqslant_{\text{lex}} \pi$ and Property 1(ii), $\mathsf{tl}(\sigma) \leqslant_{\text{lex}} \mathsf{tl}(\tau)$ and

[†] It is a well-known fact that a relation is coinductively defined as the greatest fixpoint of some monotone operator iff its complement is inductively defined as the least fixpoint of the dual operator; expressed in the language of the μ-calculus, $\neg \nu X.\tau(X) = \mu X.\neg\tau(\neg X)$.

$\mathsf{tl}(\rho) \leqslant_{\text{lex}} \mathsf{tl}(\pi)$. By the coinduction hypothesis, we have $\mathsf{tl}(\sigma) + \mathsf{tl}(\rho) \leqslant_{\text{lex}} \mathsf{tl}(\tau) + \mathsf{tl}(\pi)$. That is, $\mathsf{tl}(\sigma + \rho) \leqslant_{\text{lex}} \mathsf{tl}(\tau + \pi)$. This establishes Property 1(ii) for $(\sigma + \rho, \tau + \pi)$. $\square$

A subtle but important observation is that the equations (3.4) determine the operation $+$ on streams uniquely. Indeed, this would be the preferred way to *define* the operation $+$ coinductively for the purpose of formalization in an automated deduction system such as Coq or NuPrl, as the informal definition above using pointwise sum would require the extraneous notions of the natural numbers and indexing.

But how do we know from Equation (3.4) alone that $+$ exists and is unique? Ultimately, this comes from the fact that $(A^\omega, \mathsf{hd}, \mathsf{tl})$ is a final coalgebra (Aczel 1988; Barwise and Moss 1996). This means that for any coalgebra $(X, \mathsf{obs}, \mathsf{cont})$ with $\mathsf{obs} : X \to A$ and $\mathsf{cont} : X \to X$, there is a unique coalgebra morphism $X \to A^\omega$. If we make a coalgebra out of $A^\omega \times A^\omega$ by defining

$$\mathsf{obs}(\sigma, \tau) = \mathsf{hd}(\sigma) + \mathsf{hd}(\tau) \qquad\qquad \mathsf{cont}(\sigma, \tau) = (\mathsf{tl}(\sigma), \mathsf{tl}(\tau)),$$

then $+$ is the unique morphism to the final coalgebra A^ω, the equations (3.4) asserting exactly that $+$ is a coalgebra morphism.

3.2. *Recursive types*

Recursive types were introduced by Mendler (1988). The subtyping problem for recursive types was studied in Amadio and Cardelli (1993); Brandt and Henglein (1998); Kozen *et al.* (1995). In their simplest form, recursive types are constructed from the constants $\bot$ and $\top$ and the binary function space constructor $\to$. The set of *recursive types* C is the set of coterms of this signature. The subtype order $\leqslant$ is defined to be the greatest binary relation on C such that if $\sigma \leqslant \tau$, then either

— $\sigma = \bot$, or
— $\tau = \top$, or
— $\sigma = \sigma_1 \to \sigma_2$ and $\tau = \tau_1 \to \tau_2$ and $\tau_1 \leqslant \sigma_1$ and $\sigma_2 \leqslant \tau_2$.

In other words, $\leqslant$ is $\nu X.T(X)$, the greatest post-fixpoint of the monotone map

$$\begin{aligned} T(X) = \{(\bot, \tau) \mid \tau \in C\} \cup \{(\sigma, \top) \mid \sigma \in C\}, \\ \cup \{(\sigma_1 \to \sigma_2, \tau_1 \to \tau_2) \mid (\tau_1, \sigma_1) \in X,\ (\sigma_2, \tau_2) \in X\}. \end{aligned}$$

Theorem 3. $\leqslant$ is transitive.

Proof. Suppose $\sigma \leqslant \rho \leqslant \tau$. If $\sigma = \bot$ or $\tau = \top$, we are done. Otherwise, we cannot have $\rho = \top$ since $\rho \leqslant \tau$, and we cannot have $\rho = \bot$ since $\sigma \leqslant \rho$, therefore $\rho = \rho_1 \to \rho_2$ for some ρ_1, ρ_2. Then we must also have $\sigma = \sigma_1 \to \sigma_2$ since $\sigma \leqslant \rho$ and $\tau = \tau_1 \to \tau_2$ since $\rho \leqslant \tau$. Because $\sigma \leqslant \rho \leqslant \tau$, we must have $\tau_1 \leqslant \rho_1 \leqslant \sigma_1$ and $\sigma_2 \leqslant \rho_2 \leqslant \tau_2$. By the coinduction hypothesis, $\tau_1 \leqslant \sigma_1$ and $\sigma_2 \leqslant \tau_2$, therefore $\sigma \leqslant \tau$. $\square$

3.3. *Closure conversion*

Here is a more involved example from Jeannin and Kozen (2012). Consider the λ-calculus with variables Var and atomic constants Const. For a λ-term e, let $\mathsf{FV}(e)$ denote the set of its free variables. Let λ-Abs denote the set of λ-abstractions $\lambda x.e$.

Closures are the traditional representation of functions in functional languages with static scoping. A closure consists of a λ-abstraction paired with a copy of the environment in effect at the site of the function's definition. The environment is used to interpret the free variables in the body of the λ-abstraction. Closures can be defined coinductively with the recursive type definition

$$\begin{aligned} \mathsf{Val} &= \mathsf{Const} + \mathsf{Cl}, && \text{values} \\ \mathsf{Cl} &= \lambda\text{-}\mathsf{Abs} \times \mathsf{ClEnv}, && \text{closures} \\ \mathsf{ClEnv} &= \mathsf{Var} \rightharpoonup \mathsf{Val}. && \text{closure environments} \end{aligned}$$

Milner and Tofte (1991). Thus a closure is a pair $\{\lambda x.e, \Sigma\}$, where $\lambda x.e$ is a λ-abstraction and Σ is a closure environment. A closure $\{\lambda x.e, \Sigma\}$ must satisfy the additional requirements that $\mathsf{FV}(\lambda x.e) \subseteq \mathsf{dom}\,\Sigma$ and $\mathsf{dom}\,\Sigma$ is finite.

Capsules (Jeannin and Kozen 2012) are a simplified representation of the global state of a computation that achieve static scoping in a more direct way than with closures. A *capsule* is a pair $\langle e, \sigma\rangle$, where e is a λ-term and $\sigma : \mathsf{Var} \rightharpoonup \mathsf{Const} + \lambda\text{-}\mathsf{Abs}$ is a partial function with finite domain $\mathsf{dom}\,\sigma$, such that

i. $\mathsf{FV}(e) \subseteq \mathsf{dom}\,\sigma$,
ii. if $x \in \mathsf{dom}\,\sigma$, then $\mathsf{FV}(\sigma(x)) \subseteq \mathsf{dom}\,\sigma$.

The component σ is called a *capsule environment*. The set of capsule environments is denoted CapEnv. Note that capsule environments $\sigma : \mathsf{CapEnv}$ and closure environments $\Sigma : \mathsf{ClEnv}$ are very different things.

A capsule gives a coalgebraic representation of the global state of a computation. Capsules are essentially elements of a final coalgebra, and in Jeannin and Kozen (2012) informal coinductive reasoning was used extensively.

One result from Jeannin and Kozen (2012) is that capsule semantics and closure semantics are equivalent. Each capsule $\langle e, \sigma\rangle$ can be coinductively mapped to a closure $\langle e, \bar\sigma\rangle$ by

$$\bar\sigma(y) = \begin{cases} \{\sigma(y), \bar\sigma\}, & \text{if } \sigma(y) : \lambda\text{-}\mathsf{Abs}, \\ \sigma(y), & \text{if } \sigma(y) : \mathsf{Const}. \end{cases}$$

This definition may appear circular at first glance, since $\bar\sigma$ seems to be defined in terms of itself. But it actually defines $\bar\sigma$ uniquely for the same reason that $+$ was defined uniquely in Section 3.1. In pseudo-ML, the definition might look like

```
let rec $\bar\sigma = \lambda y$ . match $\sigma(y)$, with
    | Const($c$) $\to$ Const($c$),
    | $\lambda$-Abs($\lambda x.e$) $\to$ Cl($\{\lambda x.e, \bar\sigma\}$).
```

To state the relationship between capsules and closures, we define a binary relation $\sqsubseteq$ on capsule environments, closure environments, and values. For capsule environments, define $\sigma \sqsubseteq \tau$ if $\mathsf{dom}\,\sigma \subseteq \mathsf{dom}\,\tau$ and for all $y \in \mathsf{dom}\,\sigma$, $\sigma(y) = \tau(y)$. The definition for values and closure environments is by mutual coinduction: $\sqsubseteq$ is defined to be the largest relation such that

A. on closure environments, if $\Sigma \sqsubseteq \Gamma$ then
   i. $\mathsf{dom}\,\Sigma \subseteq \mathsf{dom}\,\Gamma$, and
   ii. for all $y \in \mathsf{dom}\,\Sigma$, $\Sigma(y) \sqsubseteq \Gamma(y)$; and

B. on values, if $u \sqsubseteq v$ then either
   i. u and v are constants and $u = v$; or
   ii. $u = \{\lambda x.e,\ \Sigma\}$, $v = \{\lambda x.e,\ \Gamma\}$, and $\Sigma \sqsubseteq \Gamma$.

Formally, $\sqsubseteq$ for closures consists of two relations defined by mutual coinduction, one on closure environments and one on values. More precisely, the relation $\sqsubseteq$ is defined to be the largest relation R on $(\mathsf{ClEnv} \times \mathsf{ClEnv}) + (\mathsf{Val} \times \mathsf{Val})$ such that $R \subseteq T(R)$ (symbolically, $\sqsubseteq = \nu X.T(X)$), where T is the monotone map

$$T : (\mathsf{ClEnv} \times \mathsf{ClEnv}) + (\mathsf{Val} \times \mathsf{Val}) \ \rightarrow\ (\mathsf{ClEnv} \times \mathsf{ClEnv}) + (\mathsf{Val} \times \mathsf{Val}),$$

defined as follows:

A. for closure environments, $(\Sigma, \Gamma) \in T(R)$ iff
   i. $\mathsf{dom}\,\Sigma \subseteq \mathsf{dom}\,\Gamma$, and
   ii. for all $y \in \mathsf{dom}\,\Sigma$, $(\Sigma(y), \Gamma(y)) \in R$; and

B. for values, $(u, v) \in T(R)$ iff either
   i. u and v are constants and $u = v$; or
   ii. $u = \{\lambda x.e,\ \Sigma\}$, $v = \{\lambda x.e,\ \Gamma\}$, and $(\Sigma, \Gamma) \in R$.

Theorem 4. The relation $\sqsubseteq$ is transitive.

Proof. Suppose $\Sigma \sqsubseteq \Delta \sqsubseteq \Gamma$. By A(i), $\mathsf{dom}\,\Sigma \subseteq \mathsf{dom}\,\Delta \subseteq \mathsf{dom}\,\Gamma$, so $\mathsf{dom}\,\Sigma \subseteq \mathsf{dom}\,\Gamma$, and A(i) holds for the pair Σ, Γ. Moreover, for all $y \in \mathsf{dom}\,\Sigma$, by A(ii), $\Sigma(y) \sqsubseteq \Delta(y) \sqsubseteq \Gamma(y)$, therefore $\Sigma(y) \sqsubseteq \Gamma(y)$ by the coinduction hypothesis on values. Thus A(ii) holds, and $\Sigma \sqsubseteq \Gamma$.

For values, suppose $u \sqsubseteq w \sqsubseteq v$. If u is a constant c, then $w = c$ and $v = c$, hence B(i) holds for the pair u, v. If $u = \{\lambda x.e,\ \Sigma\}$, then by B(ii), $w = \{\lambda x.e,\ \Delta\}$, $v = \{\lambda x.e,\ \Gamma\}$, and $\Sigma \sqsubseteq \Delta \sqsubseteq \Gamma$. By the coinduction hypothesis on closure environments, $\Sigma \sqsubseteq \Gamma$, thus $u \sqsubseteq v$. □

Theorem 5. Closure conversion is monotone with respect to $\sqsubseteq$. That is, if $\sigma \sqsubseteq \tau$, then $\bar{\sigma} \sqsubseteq \bar{\tau}$.

Proof. Let σ and τ be capsule environments and suppose that $\sigma \sqsubseteq \tau$. Then $\mathsf{dom}\,\sigma \subseteq \mathsf{dom}\,\tau$ and $\sigma(y) = \tau(y)$ for all $y \in \mathsf{dom}\,\sigma$. Note that $\mathsf{dom}\,\bar{\sigma} = \mathsf{dom}\,\sigma \subseteq \mathsf{dom}\,\tau = \mathsf{dom}\,\bar{\tau}$, which gives A(i) for $\bar{\sigma}$ and $\bar{\tau}$ immediately.

For any $y \in \mathsf{dom}\,\bar{\sigma}$, if $\sigma(y)$ is a constant c, then $\tau(y) = c$ because $\sigma \sqsubseteq \tau$, and $\bar{\sigma}(y) = \sigma(y) = \tau(y) = \bar{\tau}(y)$, thus A(ii) holds of $\bar{\sigma}$ and $\bar{\tau}$. If $\sigma(y)$ is a λ-abstraction, then so is $\tau(y)$ and they are equal, thus $\bar{\sigma}(y) = \{\sigma(y), \bar{\sigma}\} \sqsubseteq \{\tau(y), \bar{\tau}\} = \bar{\tau}(y)$, using the coinduction hypothesis B(ii). In both cases, A(ii) holds of $\bar{\sigma}$ and $\bar{\tau}$. □

3.4. *Bisimilarity*

For set-based coalgebras, one form of the classical coinduction principle states that if two elements are bisimilar, then their unique images in the final coalgebra are the same. In particular, bisimilar elements of a final coalgebra are equal. Traditional proofs involving this principle can be handled in a way similar to the applications of the previous section.

For example, in the case of A-streams, R is a bisimulation if for any σ, τ,

$$(\sigma, \tau) \in R \quad \Rightarrow \quad \mathsf{hd}(\sigma) = \mathsf{hd}(\tau) \text{ and } (\mathsf{tl}(\sigma), \mathsf{tl}(\tau)) \in R.$$

The relation of bisimilarity $\sim$ is the maximal bisimulation. This is the greatest postfixpoint of the monotone operator

$$T(R) = \{(\sigma, \tau) \mid \mathsf{hd}(\sigma) = \mathsf{hd}(\tau) \text{ and } (\mathsf{tl}(\sigma), \mathsf{tl}(\tau)) \in R\},$$

or in other words, the greatest relation $\sim$ such that $\sim \subseteq T(\sim)$. The greatest post-fixpoint is also the greatest fixpoint, therefore $\sim = T(\sim)$.

We can now prove coinductively that $\sim$ is an equivalence relation. Let us illustrate by proving that $\sim$ on streams is symmetric.

Theorem 6. $\sim$ on A-streams is symmetric relation. That is, $\sigma \sim \tau$ implies $\tau \sim \sigma$.

Proof. Assume $\sigma \sim \tau$. Then $\mathsf{hd}(\sigma) = \mathsf{hd}(\tau)$ and $\mathsf{tl}(\sigma) \sim \mathsf{tl}(\tau)$. By the symmetry of equality on A, $\mathsf{hd}(\tau) = \mathsf{hd}(\sigma)$. By the coinduction hypothesis on the tails, $\mathsf{tl}(\tau) \sim \mathsf{tl}(\sigma)$. As $\sim$ is maximal, $\tau \sim \sigma$. □

We can also use the principle to reason about properties of stream operations. For example, consider the two inverse stream operations

$$\begin{aligned} \mathsf{split}(\sigma_0\sigma_1\sigma_2\cdots) &= (\sigma_0\sigma_2\ldots, \sigma_1\sigma_3\cdots), \\ \mathsf{merge}(\sigma_0\sigma_1\cdots, \tau_0\tau_1\cdots) &= \sigma_0\tau_0\sigma_1\tau_1\cdots, \end{aligned}$$

characterized coinductively by the equations

$$\begin{aligned} \mathsf{merge}(a :: \sigma, \tau) &= a :: \mathsf{merge}(\tau, \sigma), \\ \mathsf{split}(a :: \sigma) &= \mathsf{let}\ (\rho, \tau) = \mathsf{split}(\sigma)\ \mathsf{in}\ (a :: \tau, \rho), \end{aligned}$$

or, expressed completely in terms of destructors,

$$\begin{aligned} \mathsf{hd}(\mathsf{merge}(\sigma, \tau)) &= \mathsf{hd}(\sigma) & \mathsf{hd}(\mathsf{split}(\sigma)_1) &= \mathsf{hd}(\sigma), \\ \mathsf{tl}(\mathsf{merge}(\sigma, \tau)) &= \mathsf{merge}(\tau, \mathsf{tl}(\sigma)) & \mathsf{tl}(\mathsf{split}(\sigma)_1) &= \mathsf{split}(\mathsf{tl}(\sigma))_2, \\ & & \mathsf{split}(\sigma)_2 &= \mathsf{split}(\mathsf{tl}(\sigma))_1. \end{aligned} \tag{3.5}$$

Let us argue that merge is a left inverse of split.

Theorem 7. For all streams σ, $\mathsf{merge}(\mathsf{split}(\sigma)) = \sigma$.

Proof. We will prove $\mathsf{merge}(\mathsf{split}(\sigma)) \sim \sigma$ by coinduction; since equality and bisimilarity coincide on the final coalgebra, we will have $\mathsf{merge}(\mathsf{split}(\sigma)) = \sigma$. We argue in terms of the characterization in Equation (3.5).

$$\begin{aligned}
\mathsf{hd}(\mathsf{merge}(\mathsf{split}(\sigma))) &= \mathsf{hd}(\mathsf{merge}(\mathsf{split}(\sigma)_1, \mathsf{split}(\sigma)_2)),\\
&= \mathsf{hd}(\mathsf{split}(\sigma)_1),\\
&= \mathsf{hd}(\sigma),\\
\mathsf{tl}(\mathsf{merge}(\mathsf{split}(\sigma))) &= \mathsf{tl}(\mathsf{merge}(\mathsf{split}(\sigma)_1, \mathsf{split}(\sigma)_2)),\\
&= \mathsf{merge}(\mathsf{split}(\sigma)_2, \mathsf{tl}(\mathsf{split}(\sigma)_1)),\\
&= \mathsf{merge}(\mathsf{split}(\mathsf{tl}(\sigma))_1, \mathsf{split}(\mathsf{tl}(\sigma))_2),\\
&= \mathsf{merge}(\mathsf{split}(\mathsf{tl}(\sigma))),\\
&\sim \mathsf{tl}(\sigma),
\end{aligned}$$

the last step by the coinduction hypothesis. As $\sim$ is maximal, we can conclude that $\mathsf{merge}(\mathsf{split}(\sigma)) \sim \sigma$. □

Why did we not argue in the last step that $\mathsf{merge}(\mathsf{split}(\mathsf{tl}(\sigma))) = \mathsf{tl}(\sigma)$ by the coinduction hypothesis, then conclude that $\mathsf{merge}(\mathsf{split}(\sigma)) = \sigma$ because the heads and tails were equal? We might have done so, but we wanted to emphasize that it is bisimilarity $\sim$, not equality $=$, that is the maximal fixpoint of the relevant monotone map

$$T(X) = \{(\sigma, \tau) \mid \mathsf{hd}(\sigma) = \mathsf{hd}(\tau) \text{ and } (\mathsf{tl}(\sigma), \mathsf{tl}(\tau)) \in X\}.$$

We may not use the technique with just any property, only with those defined as maximal fixpoints.

We could conclude equality because streams are the final coalgebra, for which bisimilarity and equality coincide. But except for this step, the argument works for any coalgebra for this signature. Consider coalgebras $(X, \mathsf{obs}, \mathsf{cont})$ with observations $\mathsf{obs} : X \to A$ and continuations $\mathsf{cont} : X \to X$. The equations (3.5) can be interpreted as implicit coinductive descriptions of maps $\mathsf{merge} : C \times C \to C$ and $\mathsf{split} : C \to C \times C$:

$$\begin{aligned}
\mathsf{obs}(\mathsf{merge}(x, y)) &= \mathsf{obs}(x) & \mathsf{obs}(\mathsf{split}_1(x)) &= \mathsf{obs}(x),\\
\mathsf{cont}(\mathsf{merge}(x, y)) &= \mathsf{merge}(y, \mathsf{cont}(x)) & \mathsf{cont}(\mathsf{split}_1(x)) &= \mathsf{split}_2(\mathsf{cont}(x)),\\
& & \mathsf{split}_2(x) &= \mathsf{split}_1(\mathsf{cont}(x)).
\end{aligned}$$

Note that these equations do not define merge and split uniquely, because they do not specify what $\mathsf{merge}(x, y)$ and $\mathsf{split}(x)$ actually are, but only describe their observable behavior. Nevertheless, whatever they are, they are inverses up to bisimulation:

Theorem 8. For all x, $\mathsf{merge}(\mathsf{split}(x)) \sim x$.

The proof is the same as that of Theorem 7, *mutatis mutandis*.

4. A coinductive proof principle

The proofs of Section 3, magical as they may seem, involve no magic – only a little sleight of hand! The rule we are using in these examples is a special form of a more general coinduction principle that is best explained in the language of dynamic logic (DL) and the modal μ-calculus; see Harel *et al.* (2000). Our examples typically involve

— coalgebras K_1, K_2 viewed as Kripke models with binary relations a and b, respectively, encoding coalgebraic destructors, and

— a kind of simulation relation $\pi : K_1 \times K_2$ between them, often a function $\pi : K_1 \to K_2$.

The relations a and b induce modalities $[a]$, $\langle a \rangle$ on K_1 and $[b]$, $\langle b \rangle$ on K_2. To have a common domain to work in, we form the coproduct $K = K_1 + K_2$ whose elements are the disjoint union of K_1 and K_2 with relations a and b inherited from K_1 and K_2.

We are typically trying to establish that a property of the form $Q \to [\pi]R$ holds universally in K, where Q is a precondition defined on K_1 and R is a property on K_2 defined as a greatest fixpoint of the form $R = \nu X.G \wedge [b]X$. The set R is the greatest set of states satisfying G and closed under the action of b; in the language of DL, $[b^*]G$.[†] The property $Q \to [\pi]R$ says that any state in K_1 satisfying Q must map under π to a state or states in K_2 satisfying R. The property G in the definition of R is typically a condition that can be checked locally on states of K_2, whereas the part of the definition involving $[b]$ encodes a recursive check of R on successor states. The binary relation a on K_1 does not appear here, but will appear in the coinductive proof rule to be presented in the next section.

For example, in the application of Section 3.1 involving the transitivity of $\leqslant_{\text{lex}}$ on A-streams, the statement we are trying to prove is

For all A-streams σ, ρ, τ, if $\sigma \leqslant_{\text{lex}} \rho \leqslant_{\text{lex}} \tau$, then $\sigma \leqslant_{\text{lex}} \tau$.

Here $K_1 = A^\omega \times A^\omega \times A^\omega$ and $K_2 = A^\omega \times A^\omega$, along with relations

$$(\sigma,\ \rho,\ \tau) \xrightarrow{a} (\mathsf{tl}(\sigma),\ \mathsf{tl}(\rho),\ \mathsf{tl}(\tau)),\ \text{ if } \mathsf{hd}(\sigma) = \mathsf{hd}(\rho) = \mathsf{hd}(\tau),$$

$$(\sigma,\ \tau) \xrightarrow{b} (\mathsf{tl}(\sigma),\ \mathsf{tl}(\tau)),\ \text{ if } \mathsf{hd}(\sigma) = \mathsf{hd}(\tau),$$

$$(\sigma,\ \rho,\ \tau) \xrightarrow{\pi} (\sigma,\ \tau).$$

In this case the relation π is a function $\pi : K_1 \to K_2$, the projection of a triple onto its first and third components.

The property Q is true of a triple $(\sigma,\ \rho,\ \tau)$ if $\sigma \leqslant_{\text{lex}} \rho \leqslant_{\text{lex}} \tau$, and the property R is true of a pair $(\sigma,\ \tau)$ if $\sigma \leqslant_{\text{lex}} \tau$. Transitivity states that $Q \to [\pi]R$ is universally valid in K. The definition of R is $R = \nu X.G \wedge [b]X = [b^*]G$, where

$$G = \{(\sigma,\ \tau) \mid \mathsf{hd}(\sigma) \leqslant \mathsf{hd}(\tau)\}.$$

Above, we could alternatively have used the isomorphism $A^\omega \times \cdots \times A^\omega \cong (A \times \cdots \times A)^\omega$ in the definition of K_1, K_2, slightly simplifying the definition of the transition relations[‡].

[†] Note that there is no explicit representation of infinite computations in the standard binary relation semantics of the modal μ-calculus or DL. One might imagine that $\nu X.G \wedge [b]X$ must involve infinite sequences of b, thus cannot be equal to $[b^*]G$, which is the meet of its finite approximants $[b^n]G$; but such infinite computations do not produce a result, thus have no bearing on $\nu X.G \wedge [b]X$.

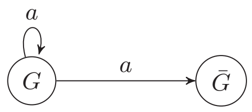
[‡] This was suggested to us by Horst Reichel.

4.1. *A Proof Rule*

It may seem that the informal rule we are using is

$$\frac{Q \to [\pi]G \qquad [a](Q \to [\pi]R) \to (Q \to [\pi]R)}{Q \to [\pi]R}\,. \tag{4.6}$$

However, this rule is unsound in general (this is the sleight of hand mentioned above). Here is a counterexample, in which $a = b$, $Q = G$ and π is the identity:



For this example, the rule (4.6) reduces to

$$\frac{G \to G \qquad [a](G \to [a^*]G) \to (G \to [a^*]G)}{G \to [a^*]G}\,.$$

The left-hand premise is trivially true in both states. The right-hand premise is also true in both states. It is true in the right-hand state, because G is false, therefore $G \to [a^*]G$ is true; and it is true in the left-hand state, because $G \to [a^*]G$ is false, therefore $[a](G \to [a^*]G)$ is false. But the conclusion $G \to [a^*]G$ is false in the left-hand state.

However, a careful look at the proofs of Section 3 reveals that we did not use any properties of R except $G \wedge [b]R \to R$ at the very last moment. Up to that point, the induction step actually established that

$$[a](Q \to [\pi]X) \to (Q \to [\pi b]X) \tag{4.7}$$

without any knowledge of X. Thus we are actually using the rule

$$\frac{Q \to [\pi]G \qquad [a](Q \to [\pi]X) \to (Q \to [\pi b]X)}{Q \to [\pi]R}\,, \tag{4.8}$$

where X is a fresh atomic symbol. We prove below (Theorem 9) that this rule is sound.

Rules similar to this appear in different forms in the literature (Brandt and Henglein 1998; Jaffar *et al.* 2008; Roşu and Lucanu 2009). In most cases, the rules are Gentzen-style with structural restrictions such as *progress* (aka *guardedness* or *contraction* (Brandt and Henglein 1998)) and *opacity* (aka *frozenness* Roşu and Lucanu (2009)). In our treatment, progress takes the form of the modalities $[a]$, $[b]$ and opacity is captured in the use of the atomic symbol X.

We have mentioned that we engaged in a little sleight-of-hand. This has to do with the use of R instead of X in the last step, which makes it seem as if we are using the unsound rule (4.6). To be completely honest, in the proof of Theorem 1 we should replace the sentence,

By the coinduction hypothesis, $\mathsf{tl}(\sigma) \leqslant_{\text{lex}} \mathsf{tl}(\tau)$.

with

By the coinduction hypothesis, $(\mathsf{tl}(\sigma), \mathsf{tl}(\tau)) \in X$, thus $(\sigma, \tau) \in [b]X$ and $(\sigma, \rho, \tau) \in [\pi b]X$.

4.2. *Soundness*

Theorem 9. The rule (4.8) is sound.

We give two proofs of this theorem.

Proof 1. For any P, if $K \vDash Q \to [\pi]P$, then $K \vDash [a](Q \to [\pi]P)$ by modal generalization. Substituting P for X in the second premise of (4.8), we have $K \vDash Q \to [\pi b]P$. Thus for any P,

$$K \vDash Q \to [\pi]P \quad \Rightarrow \quad K \vDash Q \to [\pi b]P.$$

Applying this construction inductively, we have that for all P and all $n \geqslant 0$,

$$K \vDash Q \to [\pi]P \quad \Rightarrow \quad K \vDash Q \to [\pi b^n]P,$$

therefore

$$K \vDash Q \to [\pi]P \quad \Rightarrow \quad K \vDash Q \to [\pi b^*]P.$$

In particular, for $P = G$, using the first premise of (4.8) and the definition $R = [b^*]G$, we conclude that $K \vDash Q \to [\pi]R$. □

Proof 2. From DL, we have the Galois connection

$$\vDash X \to [c]Y \quad \Leftrightarrow \quad \vDash \langle c^- \rangle X \to Y, \tag{4.9}$$

where $c^- = \{(s, t) \mid (t, s) \in c\}$. Specializing the second premise of Equation (4.8) at $X = \langle \pi^- \rangle Q$, we have

$$K \vDash [a](Q \to [\pi]\langle \pi^- \rangle Q) \quad \to \quad (Q \to [\pi b]\langle \pi^- \rangle Q).$$

But the left-hand side is a tautology of DL, therefore by modus ponens this reduces to $K \vDash Q \to [\pi b]\langle \pi^- \rangle Q$. Again by (4.9), we have

$$K \vDash \langle \pi^- \rangle Q \to [b]\langle \pi^- \rangle Q.$$

Similarly, applying (4.9) to the first premise of (4.8), we have $K \vDash \langle \pi^- \rangle Q \to G$. Combining these two facts,

$$K \vDash \langle \pi^- \rangle Q \to G \wedge [b]\langle \pi^- \rangle Q,$$

therefore $K \vDash \langle \pi^- \rangle Q \to R$, since $R = \nu X.G \wedge [b]X$. Applying (4.9) one more time, we obtain $K \vDash Q \to [\pi]R$, the conclusion of (4.8). □

4.3. *A More General Version*

The rule (4.8) only applies to monotone transformations of the form $T(X) = G \wedge [b]X$, for which $R = \nu X.T(X) = [b^*]G$. This is all we need for the examples in this paper. However, one can generalize the rule to arbitrary monotone T at the expense of some added complication in the proof system. The rule is

$$\frac{\Gamma, Q \to [\pi]X \;\vdash\; Q \to [\pi]T(X)}{\Gamma \;\vdash\; Q \to [\pi]R}, \tag{4.10}$$

for X a fresh atomic symbol not occurring in Γ, Q or π, where $R = \nu X.T(X)$. In other words, if it is possible to prove $Q \to [\pi]T(X)$ from the assumptions Γ and $Q \to [\pi]X$, where X is an atomic symbol not occurring elsewhere, then it is safe to conclude $Q \to [\pi]R$. Soundness would say that for any Kripke model K satisfying Γ, if $K \vDash Q \to [\pi]T(X)$ whenever $K \vDash Q \to [\pi]X$, then $K \vDash Q \to [\pi]R$. This rule now looks more like the rules of Brandt and Henglein (1998); Jaffar *et al.* (2008); Roşu and Lucanu (2009).

Theorem 10. The rule (4.10) is sound.

Proof. By induction on the lengths of proofs. Suppose it is possible to prove $Q \to [\pi]T(X)$ from the assumptions Γ and $Q \to [\pi]X$, where X is an atomic symbol not occurring in Γ, Q or π. By the induction hypothesis, that proof is sound. Thus, in any Kripke model K satisfying Γ, for any interpretation of X,

$$K \vDash Q \to [\pi]X \quad\Rightarrow\quad K \vDash Q \to [\pi]T(X).$$

In particular, for $X = \langle\pi^-\rangle Q$, we have

$$K \vDash Q \to [\pi]\langle\pi^-\rangle Q \quad\Rightarrow\quad K \vDash Q \to [\pi]T(\langle\pi^-\rangle Q).$$

The left-hand side is a tautology of DL, so we are left with the right-hand side, which reduces by (4.9) to $K \vDash \langle\pi^-\rangle Q \to T(\langle\pi^-\rangle Q)$. As $R = \nu X.T(X)$ is the greatest postfixpoint of T, we have $K \vDash \langle\pi^-\rangle Q \to R$. The conclusion $K \vDash Q \to [\pi]R$ follows from this and (4.9). □

4.4. *Examples*

We now describe how the other examples of Section 3 fit into this framework.

4.4.1. *Recursive Types*

In the example of Section 3.2, the statement we are trying to prove is

$$\text{For all types } \sigma,\rho,\tau, \text{ if } \sigma \leqslant \rho \leqslant \tau, \text{ then } \sigma \leqslant \tau.$$

Here $K_1 = C \times C \times C$ and $K_2 = C \times C$, where C is the set of recursive types, along with relations

$$\begin{tikzcd}
& (\sigma_1 \to \sigma_2,\ \rho_1 \to \rho_2,\ \tau_1 \to \tau_2) \arrow[dl, "a"'] \arrow[dr, "a"] & & & (\sigma_1 \to \sigma_2,\ \tau_1 \to \tau_2) \arrow[dl, "b"'] \arrow[dr, "b"] & \\
(\tau_1,\ \rho_1,\ \sigma_1) & & (\sigma_2,\ \rho_2,\ \tau_2) & (\tau_1,\ \sigma_1) & & (\sigma_2,\ \tau_2)
\end{tikzcd}$$

$$(\sigma,\ \rho,\ \tau) \xrightarrow{\pi} (\sigma,\ \tau).$$

The relation π is a function $\pi : K_1 \to K_2$, the projection of a triple onto its first and third components. Note the contravariance of the left-hand a- and b-successors.

The property Q is true of a triple (σ, ρ, τ) if $\sigma \leqslant \rho \leqslant \tau$, and the property R is true of a pair (σ, τ) if $\sigma \leqslant \tau$. Transitivity states that $Q \to [\pi]R$. The definition of R is

$R = \nu X.G \wedge [b]X = [b^*]G$, where

$$G = \{(\sigma, \tau) \mid [b]\mathsf{false} \to (\sigma = \bot \vee \tau = \top)\},$$

that is, G holds of a pair (σ, τ) with no b-successors in K_2 if either $\sigma = \bot$ or $\tau = \top$, thus $\sigma \leqslant \tau$ by local considerations.

Note that there can be an infinite b-path of pairs (σ, τ) such that $\sigma \not\leqslant \tau$. For example, if $\sigma = \bot \to \sigma$ and $\tau = \top \to \tau$, then $\sigma \not\leqslant \tau$ and $(\sigma, \tau) \xrightarrow{b} (\sigma, \tau)$.

The property $\langle \pi^- \rangle Q$ in the second proof of Theorem 9 is true of the pair (σ, τ) iff $\exists \rho\ \sigma \leqslant \rho \leqslant \tau$. The main part of the argument of Theorem 3 essentially shows that $\langle \pi^- \rangle Q \to [b]\langle \pi^- \rangle Q$ and that $\langle \pi^- \rangle Q \to G$, thereby establishing that $\langle \pi^- \rangle Q$ is a postfixpoint of $T(X) = G \wedge [b]X$.

4.4.2. *Closure Conversion* In the example of Section 3.3 involving the monotonicity of closure conversion, recall that the closure-converted form of a capsule $\langle e, \sigma \rangle$ is $\langle e, \bar{\sigma} \rangle$, where $\bar{\sigma}$ is defined as

$$\bar{\sigma}(y) = \begin{cases} \{\sigma(y), \bar{\sigma}\}, & \text{if } \sigma(y) : \lambda\text{-}\mathsf{Abs}, \\ \sigma(y), & \text{if } \sigma(y) : \mathsf{Const}. \end{cases}$$

Here we can take

$$K_1 = \mathsf{CapEnv} \times \mathsf{CapEnv} \qquad\qquad K_2 = \mathsf{ClEnv} \times \mathsf{ClEnv},$$

where CapEnv and ClEnv are the sets of capsule environments and closure environments, respectively, and

$$Q = \{(\sigma, \tau) \mid \sigma \sqsubseteq \tau\} \qquad\qquad R = \{(\Sigma, \Gamma) \mid \Sigma \sqsubseteq \Gamma\}.$$

The relation $\sqsubseteq$ on capsule environments can be defined without coinduction: $\sigma \sqsubseteq \tau$ if $\mathsf{dom}\,\sigma \subseteq \mathsf{dom}\,\tau$ and for all $y \in \mathsf{dom}\,\sigma$, $\sigma(y) = \tau(y)$. The definition for closure environments is by coinduction. In Section 3.3, it was defined by mutual coinduction on closure environments and values, but we can consolidate this into a definition just on closure environments: $\sqsubseteq$ is the largest binary relation on closure environments such that if $\Sigma \sqsubseteq \Gamma$, then $\mathsf{dom}\,\Sigma \subseteq \mathsf{dom}\,\Gamma$ and for all $y \in \mathsf{dom}\,\Sigma$, either

— $\Sigma(y)$ and $\Gamma(y)$ are constants and $\Sigma(y) = \Gamma(y)$; or
— $\Sigma(y) = \{\lambda x.e, \Delta\}$, $\Gamma(y) = \{\lambda x.e, \Pi\}$, and $\Delta \sqsubseteq \Pi$.

The relation R is defined as the greatest fixpoint $\nu X.G \wedge [b]X = [b^*]G$, where G is true of a pair (Σ, Γ) if $\mathsf{dom}\,\Sigma \subseteq \mathsf{dom}\,\Gamma$ and for all for all $y \in \mathsf{dom}\,\Sigma$, either

— $\Sigma(y)$ and $\Gamma(y)$ are constants and $\Sigma(y) = \Gamma(y)$; or
— $\Sigma(y) = \{\lambda x.e, \Delta\}$ and $\Gamma(y) = \{\lambda x.e, \Pi\}$ for some $\lambda x.e$, Δ, and Π,

and the relation b on K_2 is

$$(\Sigma, \Gamma) \xrightarrow{b} (\Delta, \Pi)$$

whenever $\Sigma(y) = \{d, \Delta\}$ and $\Gamma(y) = \{e, \Pi\}$ for some d, e, and y. The relation a on K_1 is simply $(\sigma, \tau) \xrightarrow{a} (\sigma, \tau)$.

The monotonicity theorem says

$$\forall\sigma\ \forall\tau\ \ \sigma \sqsubseteq \tau\ \rightarrow\ \bar{\sigma} \sqsubseteq \bar{\tau},$$

which is just $Q \rightarrow [\pi]R$, where π is the closure conversion function $\sigma \mapsto \bar{\sigma}$.

4.4.3. *Bisimilarity* In Section 3.4, we proved that bisimilarity is symmetric on streams and that merge is a left inverse of split.

In the first example, the statement we are trying to prove is

For all A-streams σ, τ, if $\sigma \sim \tau$, then $\tau \sim \sigma$.

Here we take $K_1 = K_2 = A^\omega \times A^\omega$ with relations

$$(\sigma,\,\tau) \xrightarrow{a,b} (\mathsf{tl}(\sigma),\,\mathsf{tl}(\tau)) \qquad\qquad (\sigma,\,\tau) \xrightarrow{\pi} (\tau,\,\sigma).$$

The properties Q and R are both $\sim$. The theorem states that $Q \rightarrow [\pi]R$. The definition of R is $R = \nu X.G \wedge [b]X = [b^*]G$, where

$$G = \{(\sigma,\,\tau) \mid \mathsf{hd}(\sigma) = \mathsf{hd}(\tau)\}.$$

In the second example, the statement we are trying to prove is

For all A-streams σ, $\mathsf{merge}(\mathsf{split}(\sigma)) = \sigma$.

Here we take $K_1 = A^\omega$ and $K_2 = A^\omega \times A^\omega$ with relations

$$\sigma \xrightarrow{a} \mathsf{tl}(\sigma) \qquad (\sigma,\,\tau) \xrightarrow{b} (\mathsf{tl}(\sigma),\,\mathsf{tl}(\tau)) \qquad \sigma \xrightarrow{\pi} (\mathsf{merge}(\mathsf{split}(\sigma)),\,\sigma).$$

The property R is still $\sim$ as above, but here $Q = \mathsf{true}$. In this case, the theorem $Q \rightarrow [\pi]R$ reduces to $[\pi]R$. It is interesting to note that the property $\langle\pi^-\rangle Q$ in the second proof of Theorem 9 here reduces to $\langle\pi^-\rangle\mathsf{true}$ and holds of a pair $(\sigma,\,\tau)$ iff $\sigma = \mathsf{merge}(\mathsf{split}(\tau))$.

4.5. *Discussion*

There are two sufficient conditions for the premise (4.7) of our proof rule that hold in many applications. These conditions can be expressed in the language of Kleene algebra with tests (KAT) Kozen (1997). They are

$$Q\pi b \leqslant Qa\pi \qquad\qquad Qa \leqslant aQ. \tag{4.11}$$

The condition on the left says that the relation π is a kind of simulation: under the enabling condition Q, the action a on the left-hand side of π simulates the action b on the right-hand side. It serves the same purpose as the DL formula $Q \rightarrow [a\pi]X \rightarrow [\pi b]X$ for atomic X, but is slightly stronger.

Lemma 1. In any Kripke model K, if $Q\pi b \leqslant Qa\pi$, then for any X, the DL formula $Q \rightarrow [a\pi]X \rightarrow [\pi b]X$ holds universally in K.

Proof. Suppose $Q\pi b \leqslant Qa\pi$ in K. Then for any X, $Q\pi b\bar{X} \leqslant a\pi\bar{X}$, where the overbar denotes Boolean negation. This implies the DL formula $Q \wedge \langle\pi b\rangle\bar{X} \rightarrow \langle a\pi\rangle\bar{X}$, which is equivalent to $Q \rightarrow [a\pi]X \rightarrow [\pi b]X$. □

The condition on the right of (4.11) says that the enabling condition Q is preserved by a on the left-hand side. It is equivalent to the KAT equations $Qa = QaQ$ and $Qa\bar{Q} = 0$, to the DL formula $Q \to [a]Q$, and to the Hoare partial correctness assertion $\{Q\}\, a\, \{Q\}$.

Theorem 11. If $Q\pi b \leqslant Qa\pi$, then the formula

$$(Q \to [\pi]G) \to (Q \to [a]Q) \to [a](Q \to [\pi]R) \to (Q \to [\pi]R)$$

is universally valid.

Proof. We show that any state satisfying $Q \to [\pi]G$, $Q \to [a]Q$, $[a](Q \to [\pi]R)$, and Q also satisfies $[\pi]R$. From Q and $Q \to [a]Q$ we have $[a]Q$. From $[a]Q$ and $[a](Q \to [\pi]R)$ we have $[a](Q \wedge (Q \to [\pi]R))$, whence $[a\pi]R$. From Q and $[a\pi]R$, by Lemma 1 we have $[\pi b]R$. From Q and $Q \to [\pi]G$ we have $[\pi]G$. From $[\pi b]R$ and $[\pi]G$ we have $[\pi](G \wedge [b]R)$, and since $G \wedge [b]R = R$ we have $[\pi]R$. □

It follows from Theorem 11 that if $Q\pi b \leqslant Qa\pi$, then the proof rule

$$\frac{Q \to [\pi]G \qquad Q \to [a]Q \qquad [a](Q \to [\pi]R)}{Q \to [\pi]R}$$

is sound, and this rule is similar to our unsound rule (4.6). However, in this case a stronger result holds.

Lemma 2. The following is a theorem of KAT:

$$Q\pi b \leqslant Qa\pi \ \wedge\ Qa \leqslant aQ \ \to\ Q\pi b^* \leqslant Qa^*\pi.$$

Proof. From $Qa \leqslant aQ$ we have

$$Q + a^*Qa \leqslant Q + a^*aQ = a^*Q,$$

therefore by a star rule of Kleene algebra,

$$Qa^* \leqslant a^*Q.$$

Using this and the first premise, we have

$$Q\pi + Qa^*\pi b = Q\pi + QQa^*\pi b \leqslant Q\pi + Qa^*Q\pi b \leqslant Q\pi + Qa^*a\pi = Qa^*\pi.$$

Again by a star rule, $Q\pi b^* \leqslant Qa^*\pi$. □

Theorem 12. Suppose $Q\pi b \leqslant Qa\pi$. The following rule is sound:

$$\frac{Q \to [\pi]G \qquad Q \to [a]Q}{Q \to [\pi]R}. \tag{4.12}$$

Proof. From the two premises, we have $Q \to [\pi]G \wedge [a]Q$, therefore

$$Q \to \nu X.[\pi]G \wedge [a]X,$$

and $\nu X.[\pi]G \wedge [a]X = [a^*\pi]G$, therefore

$$Q \to [a^*\pi]G.$$

By Lemmas 1 and 2, $Q \to [\pi b^*]G$, that is, $Q \to [\pi]R$. □

In most of our examples, the condition $Q\pi b \leqslant Qa\pi$ and the premises of the rule (4.12) are satisfied. For example, for recursive types, the first premise says that if $\sigma \leqslant \rho \leqslant \tau$, and if (σ, τ) has no b-successors, then either $\sigma = \bot$ or $\tau = \top$. The second premise says that if

$$\sigma_1 \to \sigma_2 \;\leqslant\; \rho_1 \to \rho_2 \;\leqslant\; \tau_1 \to \tau_2,$$

then $\tau_1 \leqslant \rho_1 \leqslant \sigma_1$ and $\sigma_2 \leqslant \rho_2 \leqslant \tau_2$. The condition $Q\pi b \leqslant Qa\pi$ says that if

$$\sigma_1 \to \sigma_2 \;\leqslant\; \rho \;\leqslant\; \tau_1 \to \tau_2,$$

then ρ is of the form $\rho_1 \to \rho_2$.

5. Conclusion

We have described a new style of informal coinductive reasoning and illustrated its use in mathematical arguments with several examples. The technique is like induction without a basis. We have shown that the approach is soundly based on classical coinductive principles.

An interesting research direction is to investigate whether a similar proof principle holds for properties and relations defined as least fixpoints. If this is indeed the case, can we also devise a mixed principle for induction and coinduction?

In the realm of metric coinduction, a similar proof principle has been proposed in Kozen and Ruozzi (2009). Studying connections and possible generalizations of both proof principles will possibly involve a change in category or a more categorical formulation. We would also like to explore whether we can incorporate other known proof techniques such as bisimulation up-to, as in Pous and Sangiorgi (2011). We leave these investigations for future work.

Acknowledgments

We thank Samson Abramsky, Mark Bickford, Marcello Bonsangue, Robert Constable, Helle Hvid Hansen, Bart Jacobs, Jean-Baptiste Jeannin, Horst Reichel, Jan Rutten, Ana Sokolova, and Hans Zantema for stimulating discussions. We thank the anonymous referees for their valuable suggestions for improvement of the presentation and for pointing out overlooked references.

Financial Support

The second author was partially supported by the Dutch Research Foundation (NWO), project numbers 639.021.334 and 612.001.113. The first author was supported by the National Security Agency.

Conflict of Interest

None.

References

Aczel, P. (1988). *Non-Well-Founded Sets*, Number 14 in CSLI Lecture Notes, Center for the Study of Language and Information, Stanford, CA.

Amadio, R. M. and Cardelli, L. (1993). Subtyping recursive types. *ACM Transactions on Programming Languages and Systems (TOPLAS)* **15** (4), 575–631.

Barwise, J. and Moss, L. (1996). *Vicious Circles: On the Mathematics of Non-Wellfounded Phenomena*, Number 60 in CSLI Lecture Notes, Center for the Study of Language and Information, Stanford, CA.

Brandt, M. and Henglein, F. (1998). Coinductive axiomatization of recursive type equality and subtyping. *Fundamenta Informaticae* **33** (4) 309–338.

Harel, D., Kozen, D. and Tiuryn, J. (2000). *Dynamic Logic*. MIT Press, Cambridge, MA.

Hermida, C. and Jacobs, B. (1998). Structural induction and coinduction in a fibrational setting. *Information and Computation* **145** (2) 107–152.

Ichiro, H., Jacobs, B. and Sokolova, A. (2007). Generic trace semantics via coinduction. *Logical Methods in Computer Science* **3** (4:11) 1–36.

Jaffar, J., Santosa, A. and Voicu, R. (September 2008). A coinduction rule for entailment of recursively-defined properties. In: Stuckey, P.J. (ed.) Proceedings of the 14th International Conference on Principles and Practice of Constraint Programming. *Lecture Notes in Computer Science* **5202**, Springer, Berlin, 493–508.

Jeannin, J.-B. and Kozen, D. (2012). Computing with capsules. *J. Automata, Languages and Combinatorics* **17** (2–4) 185–204.

Klin, B. (2007). Bialgebraic operational semantics and modal logic. In: *LICS*, IEEE Computer Society 336–345.

Kozen, D. (May 1997). Kleene algebra with tests. *Transactions on Programming Languages and Systems* **19** (3) 427–443.

Kozen, D., Palsberg, J. and Schwartzbach, M.I. (1995). Efficient recursive subtyping. *Mathematical Structures in Computer Science* **5** (1) 113–125.

Kozen, D. and Ruozzi, N. (2009). Applications of metric coinduction. *Logical Methods in Computer Science* **5** (3:10) 1–19.

Kurz, A. (2001). Specifying coalgebras with modal logic. *Theoretical Computer Science* **260** (1-2)119–138.

Lambek, J. (1968). A fixpoint theorem for complete categories. *Mathematische Zeitschrift* **103** 151–161.

Mendler, N.P. (1988). *Inductive Definition in Type Theory*. PhD thesis, Cornell University.

Milner, R. and Tofte, M. (1991). Co-induction in relational semantics. *Theoretical Computer Science* **87** (1) 209–220.

Niqui, M. and Rutten, J. (2009). Coinductive predicates as final coalgebras. In: *Proceedings of the 6th Workshop on Fixed Points in Computer Science (FICS 2009)* 79–85.

Paulson, L.C. (1997). Mechanizing coinduction and corecursion in higher-order logic. *Journal of Logic and Computation* **7** (2) 175–204.

Pous, D. and Sangiorgi, D. (2011). Enhancements of the coinductive proof method. In: *Advanced Topics in Bisimulation and Coinduction*, Cambridge University Press.

Roşu, G. and Lucanu, D. (September 2009). Circular coinduction: A proof theoretical foundation. In: Proceedings of the 3rd Conference on Algebra and Coalgebra in Computer Science (CALCO'09). *Lecture Notes in Computer Science* **5728**, Springer, Berlin, 127–144.

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

Schröder, L. (2005). Expressivity of coalgebraic modal logic: The limits and beyond. In: Sassone, V. (ed.) FoSSaCS. *Springer Lecture Notes in Computer Science* **3441** 440–454.

Schröder, L. (2008). Expressivity of coalgebraic modal logic: The limits and beyond. *Theoretical Computer Science* **390** (2-3) 230–247.

Schröder, L. and Pattinson, D. (2007). Rank-1 modal logics are coalgebraic. In: Thomas, W. and Weil, P. (eds.) STACS. *Springer Lecture Notes in Computer Science* **4393** 573–585.

Turi, D. and Plotkin, G.D. (1997). Towards a mathematical operational semantics. In: *LICS* 280–291.