Electronic Notes in Theoretical Computer Science 106 (2004) 145–166 

[www.elsevier.com/locate/entcs](http://www.elsevier.com/locate/entcs)

Relating Two Approaches to Coinductive Solution of Recursive Equations

# Bart Jacobs

*Department of Computer Science, University of Nijmegen*

*P.O. Box 9010, 6500 GL Nijmegen, The Netherlands Email:* [*mailto: bart@ cs. kun. nl*](mailto:bart@cs.kun.nl) *URL:* [*http: // www. cs. kun. nl/ ~bart*](http://www.cs.kun.nl/~bart)

Abstract

This paper shows that the approach of [[2](#_bookmark22),[12](#_bookmark32)] for obtaining coinductive solutions of equations on infinite terms is a special case of a more general recent approach of [[4](#_bookmark23)] using distributive laws.

*Keywords:* Coinduction, coalgebra, recursive equation.

# Introduction

The finality principle in the theory of coalgebras is usually called coinduc- tion [[8](#_bookmark28)]. It involves the existence and uniqueness of suitable coalgebra ho- momorphisms to final coalgebras. It was realised early on (see [[1](#_bookmark21),[5](#_bookmark24)]) that such coinductively obtained homomorphisms can be understood as solutions to recursive (or corecursive, if you like) equations. The equation itself is incor- porated in the commuting square expressing that we have a homomorphism from a certain “source” coalgebra to the final coalgebra. Since this diagram arises from the the source coalgebra, this source can also be identified with the recursive equation.

A systematic investigation of the solution of such equations first appeared in [[12](#_bookmark32)], followed by [[2](#_bookmark22)]. Their coalgebraic approach simplifies results on re- cursive equations with infinite terms from [[6](#_bookmark25),[7](#_bookmark27)]. More recently, a general and abstract approach is proposed in [[4](#_bookmark23)], building on distributive laws. The con- tribution of this paper is that it shows how the approach of [[2](#_bookmark22)] for infinite

1571-0661 © 2004 Published by Elsevier B.V. Open access under [CC BY-NC-ND license.](http://creativecommons.org/licenses/by-nc-nd/3.0/)

doi:10.1016/j.entcs.2004.02.032

terms fits in the general approach of [[4](#_bookmark23)] with distributive laws. This involves the identification of suitable distributive laws of the monads of terms over the underlying interface functor.

This paper is organised as follows. Section [2](#_bookmark0) briefly reviews the approach of [[4](#_bookmark23)] based on distributive laws. Section [3](#_bookmark4) introduces two distributive laws for canonical monads associated with a functor *F* . The approach of [[2](#_bookmark22)] for solu- tions of equations with infinite terms is then explained in Section [4](#_bookmark15). Finally, Section [5](#_bookmark18) shows that this approach is an instance of the distribution-based approach.

# Distributive laws and solutions of equations

Distributive laws found their first serious application in the area of coalgebras in the work of Turi and Plotkin [[15](#_bookmark35)] (see also [[14](#_bookmark33)]), providing a joint treatment of operational and denotational semantics. In that setting a distributive laws provides a suitable form of compatibility between syntax and dynamics. It leads to results like: bisimilarity is a congruence, where, of course, bisimilarity is a coalgebraic notion of equivalence, and congruence and algebraic one. The claim of [[15](#_bookmark35)] that distributive laws correspond to suitable rule formats for operators is further substantiated in [[4](#_bookmark23)]. The idea of using a distributive law in extended forms of coinduction (and hence equation solving) comes from [[9](#_bookmark29)], and is further developed in [[4](#_bookmark23)]. In this section we present its essentials.

Distributive laws are natural transformations *FG* ⇒ *GF* between two en- dofunctors *F, G*: C → C on a category C. These *F* and *G* may have additional structure (of a point or copoint, or a monad or comonad, see [[10](#_bookmark30)]), that must then be preserved by the distributive law. We shall concentrate on the case of distribution of a monad over a functor, because it seems to be most common and natural—see the example in the next section. We shall recall what this means.

Definition 2.1 *Let* (*T, η, µ*) *be a monad on a category* C*, and F* : C → C *be an arbitrary functor. A distributive law of T over F is a natural transformation*

*TF*  *λ* z*F* *T*

*making for each X* ∈ C *the following two diagrams commute.*

*T* (*λX*)

*λTX*

*FX* ¸¸¸¸¸*F* (*η* )

¸¸ *X*

*T* 2*FX*  *T* *FTX*

*F* *T* 2*X*

*ηFX*

¸¸¸¸¸¸

J z ˛

*µFX*

*F* (*µX*)

J J

*TFX λX*

*FTX TFX λX*

*FTX*

The underlying idea is that the monad *T* describes the terms in some syntax, and that the functor *F* is the interface for transitions on a state space. Intuitively, the presence of the distributive law tells us that the terms and behaviours interact appropriately. The associated notion of model is a so- called *λ*-bialgebra.

Definition 2.2 *Let λ*: *TF* ⇒ *FT be a distributive law, like above. A λ- bialgebra consists of an object X* ∈ C *with a pair of maps:*

*TX*  *a*  *X* *b*  *F* *X*

*where:*

* *a is an Eilenberg-Moore algebra, meaning that it satisﬁes two equations, namely: a* ◦ *ηX* = *id and a* ◦ *µX* = *a* ◦ *T* (*a*)*.*
* *a and b are compatible via λ, which means that the following diagram com- mutes.*

*TX*  *a*  *X* *b*  *F* *X*,,

*T* (*b*) *F* (*a*)

J

*TFX λX*

*FTX*

*A map of λ-bialgebras, from* (*TX* −*a*→ *X* −→*b*

*FX*) *to* (*TY* −→*c*

*Y* −*d*→

*FY* ) *is a map f* : *X* → *Y in* C *that is both a map of algebras and of coalgebras:*

*f* ◦ *a* = *c* ◦ *T* (*f* ) *and d* ◦ *f* = *F* (*f* ) ◦ *b.*

The following result is standard.

Lemma 2.3 *Assume a distributive law λ*: *TF* ⇒ *FT, and let ζ*: *Z* −=→ *FZ be a ﬁnal coalgebra. It carries an Eilenberg-Moore algebra obtained by ﬁnality in:*

*FTZ* \_

,,

\_ \_*F*\_(*α*\_)\_ \_

\_ *F* ,*Z* ,

*λZ*

*T F*,*Z*,

*T* (*ζ*) ∼=

*TZ* \_ \_

\_ \_ \_ \_ \_

*α*

∼= *ζ*

\_ \_ *Z*

*The resulting pair* ( *α*

*ζ*

) *is then a ﬁnal λ-bialgebra.*

*TZ* −→ *Z* −→ *FZ*

Proof By uniqueness one obtains that *α* is an Eilenberg-Moore algebra. By construction, *α* and *ζ* are compatible via *λ*. Assume an arbitrary *λ*-bialgebra

(*TX* −*a*→ *X*

−→*b*

*FX*). It induces a unique coalgebra map *f* : *X* → *Z* with

*ζ* ◦ *f* = *F* (*f* ) ◦ *b*. One then obtains *f* ◦ *a* = *α* ◦ *T* (*f* ) by showing that both maps are homomorphisms from the coalgebra *λX* ◦ *T* (*b*): *TX* → *FTX* to the final coalgebra *ζ*.

The following notion of equation and solution comes from [[4](#_bookmark23)].

Definition 2.4 *Assume a distributive law λ*: *TF* ⇒ *FT. A guarded re- cursive equation is an FT-coalgebra e*: *X* → *FTX. A solution to such an*

*equation in a λ-bialgebra* (*TY*

−*a*→ *Y*

−→*b*

*FY* ) *is a map f* : *X* → *Y making*

*the following diagram commute.*

*FTX FT* (*f* ) *F* *TY*

(1)

,,

J*F* (*a*)

*e F Y*,,

*b*

*X f*  *Y*

In ordinary coinduction one obtains solutions for equations *X* → *FX*. The power of the above notion of equation *X* → *FTX* lies in the fact that it allows actions on terms. For convenience we shall often call these equations *X* → *FTX λ*-equations—even though their formulation does not involve a distributive law *λ*. But their intended use is in a context with distributive laws.

This notion of solution may seem a bit strange at first, but becomes more natural in light of the following result. It is implicit in [[4](#_bookmark23)].

Proposition 2.5 *There exists a bijective correspondence between λ-equations*

*e*: *X* → *FTX and λ-bialgebras* ( 2 *µ*X *d* ) *with free algebra*

*T X* −→ *TX* −→ *FTX*

*µX.*

*Moreover, let* (*TY*

−*a*→ *Y*

−→*b*

*FY* ) *be a λ-bialgebra. Then there is a*

*bijective correspondence between solutions f* : *X* → *Y as in (*[*1*](#_bookmark2)*) and bialgebra maps g*: *TX* → *Y —for the associated λ-equations and λ-bialgebras.*

Proof Given a *λ*-equation *e*: *X* → *FTX* we define

*F* *TX*

*e* = *TX T* (*e*)

*TFTX*

*λTX*

*FT* 2*X*

*F* (*µX*)

This yields, together with the free algebra *µX*: *T* 2*X* → *TX* a *λ*-bialgebra:

*F* (*µX*) ◦ *λTX* ◦ *T* (*e*) = *F* (*µX*) ◦ *λTX* ◦ *T* (*F* (*µX*) ◦ *λTX* ◦ *T* (*e*))

= *F* (*µX*) ◦ *FT* (*µX*) ◦ *λT* 2 *X* ◦ *T* (*λTX*) ◦ *T* (*e*)

2

= *F* (*µX*) ◦ *F* (*µTX*) ◦ *λT* 2 *X* ◦ *T* (*λTX*) ◦ *T* (*e*)

2

= *F* (*µX*) ◦ *λTX* ◦ *µFT X* ◦ *T* 2(*e*)

= *F* (*µX*) ◦ *λTX* ◦ *T* (*e*) ◦ *µX*

= *e* ◦ *µX.*

Conversely, given a *λ*-bialgebra ( 2 *µ*X *d* ), we define a *λ*-

*T X* −→ *TX* −→ *FTX*

equation:

*d* = *X*  *ηX*  *T* *X*  *d*  *F* *TX*

These operations *e* '→ *e* and *d* '→ *d* are each others inverses:

*e* = *e* ◦ *ηX*

= *F* (*µX*) ◦ *λTX* ◦ *T* (*e*) ◦ *ηX*

= *F* (*µX*) ◦ *λTX* ◦ *ηFT X* ◦ *e*

= *F* (*µX*) ◦ *F* (*ηTX*) ◦ *e*

= *e.*

*d* = *F* (*µX*) ◦ *λTX* ◦ *T* (*d*)

= *F* (*µX*) ◦ *λTX* ◦ *T* (*d* ◦ *ηX* )

= *d* ◦ *µX* ◦ *T* (*ηX*)

= *d.*

Assume now we have a solution *f* : *X* → *Y* for *e*: *X* → *FTX* like in ([1](#_bookmark2)). We take *f* = *a* ◦ *T* (*f* ): *TX* → *Y* . It forms a map of *λ*-bialgebras, from (*µX, e*)

to (*a, b*):

*a* ◦ *T* (*f*) = *a* ◦ *T* (*a* ◦ *T* (*f* ))

= *a* ◦ *µY* ◦ *T* 2(*f* )

= *a* ◦ *T* (*f* ) ◦ *µX*

= *f* ◦ *µX.*

*F* (*f*) ◦ *e* = *F* (*a* ◦ *T* (*f* )) ◦ *F* (*µX*) ◦ *λTX* ◦ *T* (*e*)

= *F* (*a*) ◦ *F* (*µX*) ◦ *FT* 2(*f* ) ◦ *λTX* ◦ *T* (*e*)

= *F* (*a*) ◦ *FT* (*a*) ◦ *FT* 2(*f* ) ◦ *λTX* ◦ *T* (*e*)

= *F* (*a*) ◦ *λY* ◦ *TF* (*a*) ◦ *TFT* (*f* ) ◦ *T* (*e*)

= *F* (*a*) ◦ *λY* ◦ *T* (*b*) ◦ *T* (*f* )

= *b* ◦ *a* ◦ *T* (*f* )

= *b* ◦ *f.*

Conversely, assume a *λ*-bialgebra map *g*: *TX* → *Y* from (*µX, d*) to (*a, b*). It yields a map *g* = *g* ◦ *ηX*: *X* → *Y* which is a solution of *d*, since:

*F* (*a*) ◦ *FT* (*g*) ◦ *d* = *F* (*a*) ◦ *FT* (*g* ◦ *ηX*) ◦ *d* ◦ *ηX*

= *F* (*g*) ◦ *F* (*µX*) ◦ *FT* (*ηX*) ◦ *d* ◦ *ηX*

= *F* (*g*) ◦ *d* ◦ *ηX*

= *b* ◦ *g* ◦ *ηX*

= *b* ◦ *g.*

Finally, it is obvious that *f* '→ *f* and *g* '→ *g* are each others inverses.

Now we can formulate the main result of this distribution-based approach to solving equations.

Theorem 2.6 *Let F* : C → C *be a functor with a ﬁnal coalgebra Z*

−=→

*FZ. For each monad T with distributive law λ*: *TF* ⇒ *FT there are unique*

*solutions to λ-equations in the ﬁnal λ-bialgebra* (*TZ* → *Z* → *FZ*) *from Lemma* [*2.3*](#_bookmark1)*.*

Proof For a *λ*-equation *e*: *X* → *FTX*, a solution in (*TZ* → *Z* → *FZ*) is by the previous proposition the same thing as a map of *λ*-bialgebras from the associated (*T* 2*X* → *TX* → *FTX*) to (*TZ* → *Z* → *FZ*). Since the latter is final, there is precisely one such solution.

In Example [3.3](#_bookmark7) in the next section we present an illustration.

# Free monads and their distributive laws

In this section we consider an endofunctor *F* : C → C with two canonical associated monads *F* ∗ and *F* ∞, together with distributive laws *λ*∗ and *λ*∞ over *F* . The first result is not used directly, but provides the setting the second one—which forms the basis for Lemma [5.1](#_bookmark19) later on.

* 1. *The free monad on a functor*

Let *F* : C → C be an arbitrary endofunctor on a category C with (binary) coproducts +. The only assumption we make at this stage is that for each object *X* ∈ C the functor *X* + *F* (−): C → C has an initial algebra. We shall use the following notation. The carrier of this initial algebra will be written as *F* ∗(*X*) with structure map given as:

*X* + *F* (*F* ∗(*X*)) *α*  *F* ∗(*X*)

~

=

Further, we shall write

*ηX* = *α* ◦ *κ*1 *τX* = *α* ◦ *κ*2*,*

so that *αX* = [*ηX, τX*].

The mapping *X* '→ *F* ∗(*X*) is functorial: for *f* : *X* → *Y* we get:

*X* + *F* (*F*

id + *F* (*F* ∗(*f* ))

∗(*X*)) *X* + *F* (*F*

∗(*Y* ))

*αX* ∼=

∗ J

[*ηY* ◦ *f, τY* ]

∗ J

*F* (*X*) *F* (*Y* )

*F* ∗(*f* )

This means that

*F* ∗(*f* ) ◦ *ηX* = *ηY* ◦ *f F* ∗(*f* ) ◦ *τX* = *τY* ◦ *F* (*F* ∗(*f* ))*,*

*i.e.* that *η*: id ⇒ *F* ∗ and *τ* : *FF* ∗ ⇒ *F* ∗ are natural transformations.

Next we establish that *F* ∗ is a monad. The multiplication *µ* is obtained

in:

*F* ∗(*X*)+ *F* (*F* ∗(*F* ∗(*X*))) \_ \_id\_+\_ *F*\_ (\_*µ*\_*X* )\_ \_ *F* ∗(*X*)+ *F* (*F* ∗(*X*))

*αF* ∗(*X*) ∼=

[id*, τX* ]

∗ ∗J

\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_

∗ J

*F* (*F*

(*X*))

*µX*  *F* (*X*)

This yields one of the monad equations, namely *µX* ◦ *ηF* ∗(*X*) = id. The related equation *µX* ◦ *F* ∗(*ηX*) = id follows from uniqueness of algebra maps *αX* → *αX*:

*µX* ◦ *F* (*ηX*) ◦ *αX* = *µX* ◦ [*ηF* ∗(*X*) ◦ *ηX, τF* ∗(*X*)] ◦ (id + *F* (*F* (*ηX* )))

∗ ∗

= [*ηX, τX* ◦ *F* (*µX*)] ◦ (id + *F* (*F* ∗(*ηX* )))

= *αX* ◦ (id + *F* (*µX* ◦ *F* ∗(*ηX* )))*.*

Similarly, the other requirements making *F* ∗ a monad are obtained.

The following standard result sums up the situation.

Proposition 3.1 *Let F* : C → C *with induced monad* (*F* ∗*, η, µ*) *be as described above.*

1. *The mapping X* '→ (*F* (*F* ∗(*X*) *τ*X

−→ *F*

*forgetful functor U* : Alg(*F* ) → C*.*

∗(*X*)) *forms a left adjoint to the*

*The monad induced by this adjunction is* (*F* ∗*, η, µ*)*.*

1. *The mapping σX* = *τX* ◦ *F* (*ηX*): *F* (*X*) → *F* ∗(*X*) *yields a natural trans- formation F* ⇒ *F* ∗ *that makes F* ∗ *the free monad on F.*

The next observation shows that the monad *F* ∗ of (finite) *F* -terms fits with the behaviour of *F* . It follows from a general observation (made for instance in [[4](#_bookmark23)]) that distributive laws *F* ∗*G* ⇒ *GF* ∗ correspond to ordinary natural transformations *FG* ⇒ *GF* . Hence by taking *G* = *F* and the identity *FF* ⇒ *FF* one gets *F* ∗*F* ⇒ *FF* ∗. But here we shall present the explicit construction.

Proposition 3.2 *Let F* : C → C *have free monad F* ∗*. Then there is a dis- tributive law λ*∗: *F* ∗*F* ⇒ *FF* ∗*.*

Proof We define *λ*∗ : *F* ∗(*FX*) → *F* (*F* ∗*X*) as follows.

*X*

*α*−1

*F* ∗(*FX*) *F X*  *F* *X* + *F* (*F* ∗(*FX*))

~

=

[*F* (*ηX*)*,F* (*µX* ◦ *F* ∗(*σX* ))]

*F* (*F* ∗*X*)

where *σX* = *τX* ◦ *F* (*ηX*): *F* (*X*) → *F* ∗(*X*) as introduced in Proposition [3.1](#_bookmark5) (ii).

Example 3.3 *Let Z* = RN *be the set of streams of real numbers. It is of course the ﬁnal coalgebra of the functor F* = R × (−)*, via the head and tail*

*operations* ⟨*hd, tl*⟩:

*Z* −=→ R × *Z*

*. It is shown in [*[*13*](#_bookmark34)*] that on such streams one*

*can coinductively deﬁne binary operators* ⊕ *for sum and* ⊗ *for shuﬄe product*

*satisfying the recursive equations:*

*x* ⊕ *y* = (*hd*(*x*)+ *hd*(*y*)) · (*tl*(*x*) ⊕ *tl*(*y*))

*x* ⊗ *y* = (*hd*(*x*) × *hd*(*y*)) · ((*tl*(*x*) ⊗ *y*) ⊕ (*x* ⊗ *tl*(*y*)))*,*

*where* · *is preﬁx.*

*It is easy to see that one deﬁnes* ⊕ *by ordinary coinduction, in:*

R × (*Z* × *Z*) \_ *i*\_*d*\_×\_⊕\_ \_ R × *Z*

,, ,,

*c*⊕ ∼= ⟨*hd, tl*⟩

*Z* × *Z* \_

\_ \_ \_

\_ \_ \_

\_ *Z*

⊕

*where the coalgebra c*⊕ *is deﬁned by:*

*c*⊕(*x, y*) = ⟨*hd*(*x*)+ *hd*(*y*)*,* ⟨*tl*(*x*)*, tl*(*y*)⟩ ⟩*.*

*Once we have* ⊕: *Z* × *Z* → *Z we show how to obtain x* ⊗ *y as a solution of a λ-equation. We start from the signature functor* Σ(*X*)= *X* × *X.There is an obvious distributive law* Σ*F* ⇒ *F* Σ *given by* (⟨*r, x*⟩*,* ⟨*s, y*⟩) '−→ ⟨*r* + *s,* (*x, y*)⟩*. By a result of [*[*4*](#_bookmark23)*] it lifts to a distributive law λ*: Σ∗*F* ⇒ *F* Σ∗ *involving the as- sociated free monad* Σ∗*. The algebra* ⊕: Σ(*Z*) → *Z yields an Eilenberg-Moore algebra* [[ − ]]: Σ∗(*Z*) → *Z, which is by the same result of [*[*4*](#_bookmark23)*] a λ-bialgebra. Now we obtain* ⊗ *as solution in:*

*id* × Σ∗(⊗)

R × Σ∗(*Z* × *Z*) R × Σ∗(*Z*)

,,

*id* × [[ − ]]

J

*d*⊗ R ×,*Z*,

∼= ⟨*hd, tl*⟩

*Z* × *Z* *Z*

⊗

*in which the λ-equation d*⊗ *is deﬁned by:*

*d*⊗(*x, y*) = ⟨*hd*(*x*) × *hd*(*y*)*,* (*tl*(*x*)*, y*) ⊕(*x, tl*(*y*))⟩*,*

*where* ⊕ *is a symbol for sum in the language of terms on pairs from Z* × *Z. Here we exploit the expressive power of the λ-approach, because we can now write terms as second component.*

*Clearly,*

*hd*(*x* ⊗ *y*) = *hd*(*x*) × *hd*(*y*)*.*

*And, as required:*

*tl*(*x* ⊗ *y*) = ( **[** − ]] ◦ Σ∗(⊗) ◦ *π*2 ◦ *d*⊗)(*x, y*)

= ( **[** − ]] ◦ Σ∗(⊗))(*tl*(*x*)*, y*) ⊕(*x, tl*(*y*))

= [[ (*tl*(*x*) ⊗ *y*) ⊕(*x* ⊗ *tl*(*y*)) **]**

= (*tl*(*x*) ⊗ *y*) ⊕ (*x* ⊗ *tl*(*y*))*.*

*This concludes the example.*

* 1. *The free iterative monad on a functor*

Let, like in the previous section, *F* : C → C be an arbitrary endofunctor on a category C with (binary) coproducts +. The assumption we now make is that for each object *X* ∈ C the functor *X* + *F* (−): C → C has an final coalgebra— instead of an initial algebra. We shall use the following notation. The carrier of this final calgebra will be written as *F* ∞(*X*) with structure map given as:

*F* ∞(*X*) *ζ*  *X* + *F* (*F* ∞(*X*))

~

=

The sets *F* ∗(*X*) in the previous section are understood as the set of finite terms of type *F* with free variables from *X*. Here we understand *F* ∞(*X*) as the set of both finite and infinite terms (or trees) with free variables in *X*.

Like before, we shall write:

*ηX* = *ζ*−1 ◦ *κ*1 *τX* = *ζ*−1 ◦ *κ*2*.*

Functoriality of *F* ∞ is obtained as follows. For *f* : *X* → *Y* in C we get:

id + *F* (*F* ∞(*f* ))

*Y* + *F* (*F* ∞(*X*)) \_

,,

*f* + id ◦ *ζX*

\_ \_ \_

\_ \_ \_

\_ *X* + *F* (*F* ∞(*Y* ))

,,

∼= *ζY*

*F* ∞(*X*) *F* ∞(*Y* )

*F* ∞(*f* )

This means that

*F* ∞(*f* ) ◦ *ηX* = *ηY* ◦ *f F* ∞(*f* ) ◦ *τX* = *τY* ◦ *F* (*F* ∞(*f* ))*,*

*i.e.* that *η*: id ⇒ *F* ∞ and *τ* : *FF* ∞ ⇒ *F* ∞ are natural transformations.

It is shown in [[11](#_bookmark31),[3](#_bookmark26)] that *F* ∞ is a monad [1](#_bookmark9) . The multiplication opera- tion *µ* is rather complicated, and can best be introduced via substitution *t*[*s/x*]. What we mean is replacing all occurrences (if any) of the variable *x* in the term *t* by the term *s*, but now for possibly infinite terms. In most general form, this substitution *t*[→−*s /*→−*x* ] replaces all occurrences of all vari- ables *x* ∈ *X* simultaneously. In this way, substitution may be described as an operation which tells how an *X*-indexed collection (*sx*)*x*∈*X* of terms *sx* ∈ *F* ∞(*Y* ) acts on a term *t* ∈ *F* ∞(*X*). More precisely, substitution becomes an operation subst(*s*): *F* ∞(*X*) → *F* ∞(*Y* ), for a function *s*: *X* → *F* ∞(*Y* ). As usual, such a substitution operation should respect the term structure—*i.e.* be a homomorphism—and be trivial on variables. Standardly, substitution is defined by induction on the structure of (finite) terms. But since we are deal- ing here with possibly infinite terms, we have to use coinduction. This makes the substitution more challenging. In general, it is done as follows.

Lemma 3.4 *Let X, Y be arbitrary sets. Each function s*: *X* → *F* ∞(*Y* ) *gives rise to a coalgebraic substitution operator subst*(*s*): *F* ∞(*X*) → *F* ∞(*Y* )*, namely the unique homomorphism of F-algebras:*

∞ *F* (*subst*(*s*)) ∞

*F* (*F*

(*X*))

*F* (*F*

(*Y* ))

*X* ¸¸¸¸¸

¸¸¸¸*s*

*τX τY*

J J

*with ηX*

J

¸¸¸¸¸

¸¸¸z

*F* ∞(*X*)

*subst*(*s*)

*F* ∞(*Y* ) *F* ∞(*X*)

*subst*(*s*)

*F* ∞(*Y* )

Proof We begin by defining a coalgebra structure on the coproduct *F* ∞(*Y* )+

*F* ∞(*X*) of terms, namely as the vertical composite on the left below.

*Y* + *F* (*F* ∞(*Y* )+ *F* ∞(*X*)) \_ \_id\_*Y* \_+\_*F*\_(\_*f* )\_ \_ *Y* + *F* (*F* ∞(*Y* ))

,,

[(id*Y* + *F* (*κ*1)) ◦ *ζY , κ*2 ◦ *F* (*κ*2)]

*F* ∞(*Y* )+ *F* (*F* ∞(*X*))

,,

[*κ*1*,s* + id]

*F* ∞(*Y* )+ (*X* + *F* (*F* ∞(*X*)))

,,

id*Y* + *ζX*

,,

∼= *ζY*

*F* ∞(*Y* )+ *F* ∞(*X*) *F* ∞(*Y* )

*f*

1 Similar results have been obtained earlier by [[12](#_bookmark32)], but for the functor *X* '→*F* (*X* + −).

One first proves that *f* ◦ *κ*1 is the identity, using uniqueness of coalgebra maps

*ζY* → *ζY* . Then, *f* ◦ *κ*2 is the required map subst(*s*).

In the remainder of this paper we shall make frequent use of this substitu- tion operator subst(−). Computations with substitution are made much easier with the following elementary results. Proofs are obtained via the uniqueness property of substitution.

Lemma 3.5 *For s*: *X* → *F* ∞(*Y* ) *we have:*

1. *subst*(*ηX*)= *idF* (*X*)*.*
2. *subst*(*s*) ◦ *F* ∞(*f* )= *subst*(*s* ◦ *f* )*, for f* : *Z* → *X.*
3. *subst*(*r*) ◦ *subst*(*s*)= *subst*(*subst*(*r*) ◦ *s*)*, for r*: *Y* → *F* ∞(*Z*)*.*
4. *F* ∞(*f* ) = *subst*(*ηZ* ◦ *f* )*, for f* : *Y* → *Z, and hence subst*(*F* ∞(*f* ) ◦ *s*) =

*F* ∞(*f* ) ◦ *subst*(*s*)*.*

1. *subst*(*s*)= [*s, τY* ◦ *F* (*subst*(*s*))] ◦ *ζX.*

Proposition 3.6 *The map µX* = *subst*(*idF* ∞(*X*)): *F* (*F* (*X*)) → *F* (*X*)

∞ ∞ ∞

*makes the triple* (*F* ∞*, η, µ*) *a monad.*

*This monad F* ∞ *is called the iterative monad on F, via the natural trans- formation σ* = *τ* ◦ *Fη*: *F* ⇒ *F* ∞*.*

In [[2](#_bookmark22)] it shown that *F* ∞ is in fact a *free* iterative monad, in a suitable sense. This freeness is not relevant here.

Proof We check the monad equations, using Lemma [3.5](#_bookmark10).

*µX* ◦ *ηF* ∞*X* = subst(id*F* ∞(*X*)) ◦ *ηF* ∞*X*

= id*F* ∞(*X*)*.*

*µX* ◦ *F* (*ηX* ) = subst(id*F* ∞(*X*)) ◦ *F* (*ηX*)

∞ ∞

= subst(id*F* ∞(*X*) ◦ *ηX* )

= id*F* ∞(*X*)*.*

*µX* ◦ *F* (*µX*) = subst(id*F* ∞(*X*)) ◦ *F* (*µX*)

∞ ∞

= subst(*µX*)

= subst(subst(id*F* ∞(*X*)) ◦ id*F* ∞(*F* ∞(*X*)))

= subst(id*F* ∞(*X*)) ◦ subst(id*F* ∞(*F* ∞(*X*)))

= *µX* ◦ *µF* ∞(*X*)*.*

The following is less standard.

Proposition 3.7 *Consider F* : C → C *with its iterative monad F* ∞*.*

1. *There is a distributive law λ*∞: *F* ∞*F* ⇒ *FF* ∞*.*
2. *The induced mediating map of monads F* ∗ ⇒ *F* ∞ *commutes with the distributive laws, in the sense that the following diagram commutes.*

*F* ∗*F*  *F* ∞*F*

*λ*∗ *λ*∞

J∗ J∞

*FF FF*

Proof Like for *λ*∗ we define *λ*∞: *F* ∞(*FX*) → *F* (*F* ∞*X*) as follows:

*X*

*F* ∞(*FX*)

*ζF X*

∼=

*F* *X* + *F* (*F* ∞(*FX*))

[*F* (*ηX*)*,F* (*µX* ◦ *F* ∞(*σX* ))]

*F* (*F* ∞*X*)

where *σX* = *τX* ◦ *F* (*ηX*): *F* (*X*) → *F* ∞(*X*) as introduced in Proposition [3.6](#_bookmark11). It satisfies, like in the proof of Proposition [3.2](#_bookmark6),

(2)

Then:

*µX* ◦ *σF* ∞*X* = subst(id*F* ∞*X* ) ◦ *τF* ∞*X* ◦ *F* (*ηF* ∞*X* )

= *τX* ◦ *F* (subst(id*F* ∞*X* )) ◦ *F* (*ηF* ∞*X* )

= *τX* ◦ *F* (id*F* ∞*X* )

= *τX.*

*λ*∞ ◦ *ηFX* = [*F* (*ηX*)*,F* (*µX* ◦ *F* ∞(*σX* ))] ◦ *ζ* ◦ *ηFX*

*X*

= [*F* (*ηX*)*,F* (*µX* ◦ *F* ∞(*σX* ))] ◦ *κ*1

= *F* (*ηX*)*.*

We shall use the following two auxiliary results:

(3)

∞

*µX* ◦ *σF* ∞*X* ◦ *λX* = *µX* ◦ *F* ∞(*σX* )

*F* (*τX*) ◦ *F* (*λ*∞) = *λ*∞ ◦ *τFX.*

*X* *X*

We first prove the first equation, and use it immediately to prove the second

one.

*µX* ◦ *σF* ∞*X* ◦ *λX*

∞

= [*µX* ◦ *σF* ∞*X* ◦ *F* (*ηX*)*, µX* ◦ *σF* ∞*X* ◦ *F* (*µX* ◦ *F* (*σX* ))] ◦ *ζFX*

∞

= [*µX* ◦ *F* (*ηX*) ◦ *σX, µX* ◦ *F* (*µX* ◦ *F* (*σX* )) ◦ *σF* ∞*FX* ] ◦ *ζFX*

∞ ∞ ∞

= [*µX* ◦ *ηF* ∞*X* ◦ *σX, µX* ◦ *µF* ∞*X* ◦ *F F* (*σX*) ◦ *σF* ∞*FX* ] ◦ *ζFX*

∞ ∞

= [*µX* ◦ *F* (*σX*) ◦ *ηFX, µX* ◦ *F* (*σX*) ◦ *µFX* ◦ *σF* ∞*FX* ] ◦ *ζFX*

∞ ∞

= *µX* ◦ *F* ∞(*σX* ) ◦ [*ηFX, τFX*] ◦ *ζFX*

= *µX* ◦ *F* ∞(*σX* )*. F* (*τX*) ◦ *F* (*λ*∞)

*X*

([2](#_bookmark13)) ∞

= *F* (*µX* ◦ *σF* ∞*X* ◦ *λX* )

= *F* (*µX* ◦ *F* ∞(*σX* ))

= [*F* (*ηX*)*,F* (*µX* ◦ *F* ∞(*σX* ))] ◦ *κ*2

= *λ*∞ ◦ *τFX.*

*X*

Now we are ready to prove that *λ*∞ commutes with multiplications.

*λ*∞ ◦ *µFX*

*X*

= *λX* ◦ [id*, τFX* ◦ *F* (*µFX*)] ◦ *ζF* ∞*FX* by Lemma [3.5](#_bookmark10) (v)

∞

∞ ∞

= [*λX , λX* ◦ *τFX* ◦ *F* (*µFX*)] ◦ *ζF* ∞*FX*

([3](#_bookmark14)) ∞ ∞

= [*λX ,F* (*τX* ◦ *λX* ◦ *µFX*)] ◦ *ζF* ∞*FX*

([2](#_bookmark13)) ∞ ∞

= [*λX ,F* (*µX* ◦ *σF* ∞*X* ◦ *λX* ◦ *µFX*)] ◦ *ζF* ∞*FX*

([3](#_bookmark14)) ∞ ∞

= [*λX ,F* (*µX* ◦ *F* (*σX* ) ◦ *µFX*)] ◦ *ζF* ∞*FX*

= [*λX ,F* (*µX* ◦ *µF* ∞*X* ◦ *F*

∞

∞*F* ∞

(*σX*))] ◦ *ζF* ∞*FX*

= [*λ*∞*,F* (*µX* ◦ *F*

*X*

∞(*µX* ◦ *F*

∞(*σX* )))] ◦ *ζF* ∞*FX*

([3](#_bookmark14)) ∞ ∞ ∞

= [*λX ,F* (*µX* ◦ *F* (*µX* ◦ *σF* ∞*X* ◦ *λX* ))] ◦ *ζF* ∞*FX*

∞

∞

= [id*,F* (*µX* ◦ *µF* ∞*X* ◦ *F*

∞(*σF* ∞*X* ))] ◦ (*λX* + *F* (*F*

∞*λX* )) ◦ *ζF* ∞*FX*

= *F* (*µX*) ◦ [*F* (*ηF* ∞*X* )*,F* (*µF* ∞*X* ◦ *F*

∞(*σF* ∞*X* ))] ◦ *ζFF* ∞*X* ◦ *F*

∞(*λ*∞)

= *F* (*µX*) ◦ *λ*∞ ◦ *F* ∞(*λ*∞)*.*

*X*

*F* ∞*X*

*X*

In order to prove the second point of the proposition we have to disambig- uate the notation. Let’s write the monad *F* ∗ as (*F* ∗*, η*∗*, µ*∗) with associated *τ* ∗ and *σ*∗, and *F* ∞ as (*F* ∞*, η*∞*, µ*∞) with *τ* ∞ and *σ*∞. The induced mediating

map *σ*∞: *F* ∗ ⇒ *F* ∞ is then given by:

*X* + *F* (*F*

id + *F* (*σ*∞*X* )

∗*X*) *X* + *F* (*F*

∞*X*)

*αX* ∼=

,,

∼= *ζX*

∗J ∞

*F X* ∞ *F X*

*σ*

*X*

We already know (from Proposition [3.1](#_bookmark5)) that *σ*∞ is a homomorphism of mon- ads satisfying *σ*∞ ◦ *σ*∗ = *σ*∞. Hence *σ*∞ commutes with the distributive laws:

*λ*∞ ◦ *σ*∞*FX* = [*F* (*η*∞)*,F* (*µ*∞ ◦ *F* ∞(*σ*∞))] ◦ *ζFX* ◦ *σ*∞*FX*

*X X X* *X*

= [*F* (*η*∞)*,F* (*µ*∞ ◦ *F* ∞(*σ*∞))] ◦ id + *F* (*σ*∞*FX* ) ◦ *α*−1

*X X X FX*

= [*F* (*η*∞)*,F* (*µ*∞ ◦ *F* ∞(*σ*∞) ◦ *σ*∞*FX* )] ◦ *α*−1

*X X X FX*

= [*F* (*η*∞)*,F* (*µ*∞ ◦ *σ*∞*F* ∞*X* ◦ *F* ∗(*σ*∞))] ◦ *α*−1

*X X X FX*

= [*F* (*η*∞)*,F* (*µ*∞ ◦ *σ*∞*F* ∞*X* ◦ *F* ∗(*σ*∞*X* ◦ *σ*∗ ))] ◦ *α*−1

*X X X FX*

= [*F* (*σ*∞*X* ◦ *η*∗ )*,F* (*σ*∞*X* ◦ *µ*∗ ◦ *F* ∗(*σ*∗ ))] ◦ *α*−1

*X X X FX*

= *F* (*σ*∞*X* ) ◦ [*F* (*η*∗ )*,F* (*µ*∗ ◦ *F* ∗(*σ*∗ ))] ◦ *α*−1

*X* *X*

= *F* (*σ*∞*X* ) ◦ *λ*∗ *.*

*X*

*X FX*

# Iteration and solutions of equations

The material in this section comes (again) from [[2](#_bookmark22)]. In Definition [2.4](#_bookmark3) we have seen an abstract notion of *λ*-equation and solution. A bit more concretely, for a functor *F* , a set of recursive equations—often simply called a recurs- ive equation—consists first of all of a set *X* of recursive variables. For each variable *x* ∈ *X* we have a corresponding term *t* in an equation *x* = *t*. We shall allow this term to be infinite. The term *t* may involve both variables from an already given set *Y* , and from our new set of recursive variables

*X*. Hence *t* ∈ *F* ∞(*Y* + *X*). Summarising, a recursive equation is a map *e*: *X* → *F* ∞(*Y* + *X*). We shall often call such an *e* a ∞-equation, in contrast to a *λ*-equation *X* → *FTX*—as in Definition [2.4](#_bookmark3).

Definition 4.1 *Let F* : C → C *be a functor, with for X* ∈ C *a ﬁnal coalgebra*

*F* ∞(*X*) −=→ *X* + *F* (*F* ∞(*X*))*.*

*A solution for an* ∞*-equation e*: *X* → *F* ∞(*Y* + *X*) *is a map sol*(*e*): *X* →

*F* ∞(*Y* ) *that produces an appropriate term sol*(*e*)(*x*) *for each recursive variable*

*x* ∈ *X. This means that substituting the cotuple* [*ηY , sol*(*e*)]: *Y* + *X* → *F* ∞(*Y* )

*in e yields the solution sol*(*e*)*, i.e.*

*X* ¸¸¸

*e*  *F* ∞(*Y* + *X*)

*sol*(*e*)

¸¸¸¸¸¸

*in sol*( ¸¸¸¸¸¸

*subst*([*ηY , sol*(*e*)])

= *subst*([*ηY , sol*(*e*)]) ◦ *e*

*e*) z˛J

*F* ∞(*Y* )

*This shows that the solution is a ﬁxed point of subst*([*ηY ,* −]) ◦ *e.*

Like for *λ*-equations, we are interested in unique solutions for ∞-equations. Do they always exist? Not in trivial equations, like *x* = *x*, where any term is a solution. Such equations are standardly excluded by requiring that the terms of the recursive equation are ‘guarded’, *i.e.* that its terms are not variables from *X*. This notion can also be formulated in a general categorical setting: an

∞-equation *e*: *X* → *F* ∞(*Y* +*X*) is called guarded if it factors (in a necessarily unique way) as:

(4)

*g*, ,

. .

*Y* + *F* (*F* ∞(*Y* + *X*))

, , , ˛

*κ*1 + id

(*Y* + *X*)+ *F* J ∞(*Y* + *X*))

(*F*

~ −1

= *ζ*

*Y* +*X*

J

*X*  *F* (*Y* )

∞

*e* + *X*

This says that if we decompose the terms of *e* using the final coalgebra map, then we do not get variables from *X*.

Theorem 4.2 ([[2](#_bookmark22)]) *Each guarded* ∞*-equation has a unique solution.*

Proof Assume that a guarded ∞-equation *e*: *X* → *F* ∞(*Y* + *X*) factors as

−1

*ζ*

*Y* +*X*

* (*κ*1 + id) ◦ *g*, for a map *g*: *X* → *Y* + *F* (*F* ∞(*Y* + *X*)) like in ([4](#_bookmark17)).

In order to find a solution one first defines, like in the proof of Lemma [3.4](#_bookmark8),

an auxiliary map *h*: *F* ∞(*Y* + *X*)+ *F* ∞(*Y* ) → *F* ∞(*Y* ) by coinduction, via an

appropriate structure map on the left-hand-side below.

*Y* + *F* (*F* ∞(*Y* + *X*)+ *F* ∞(*Y* )) \_ \_ i\_d*Y*\_ +\_ \_*F* (\_*h*\_) \_ \_ *Y* + *F* (*F* ∞(*Y* ))

,,

[id + *F* (*κ*1)*,* (id + *F* (*κ*2)) ◦ *ζY* ]

(*Y* + *F* (*F* ∞(*Y* + *X*))) + *F* ∞(*Y* )

,,

[[*κ*1*, g*]*, κ*2]+ id

((*Y* + *X*)+ *F* (*F* ∞(*Y* + *X*))) + *F* ∞(*Y* )

,,

*ζY* +*X* + id

,,

∼= *ζY*

*F* ∞(*Y* + *X*)+ *F* ∞(*Y* ) *F* ∞(*Y* )

*h*

The proof then proceeds by showing that *h* ◦ *κ*2 is the identity, and that *h* ◦ *κ*1 is of the form subst(*k*) for *k*: *Y* + *X* → *F* ∞(*Y* ). The unique solution is then obtained as sol(*e*)= *k* ◦ *κ*2.

# ∞-equations and solutions as λ-equations and solu- tions

In this section we put previous results together. We start by fixing an object

*Y* ∈ C, and definining the associated functors *GY ,T Y* : C → C given by

*GY* (*X*) = *Y* + *F* (*X*) *T Y* (*X*) = *F* ∞(*Y* + *X*)

Why do we choose these functors? Well, a guard *X* → *Y* + *F* (*F* ∞(*Y* + *X*)) like in ([4](#_bookmark17)) is now simply a *GY T Y* -coalgebra. We like to understand it as a *λ*- equation, in order to fit the ∞-equations in the framework of *λ*-equations. The first requirement is thus to establish the appropriate monad and distribution structure.

It is not hard to see that *T Y* is again a monad with unit and multiplication:

*Y* ∞

*η*

= *η*

*X Y* +*X*

* *κ*2 : *X* −→ *Y* + *X* −→ *F* ∞(*Y* + *X*)

*Y* = subst([*η*∞ ◦ *κ*1*,* id]) : *F* ∞(*Y* + *F* ∞(*Y* + *X*)) −→ *F* ∞(*Y* + *X*)*.*

*µ*

*Y* +*X*

*X*

For convenience we shall drop the superscript *Y* whenever confusion is unlikely.

Next we note that *T Y* is isomorphic to (*GY* )∞, since each (*GY* )∞(*X*) forms by construction the final coalgebra for the mapping

*X* '−→ *X* + *GY* (−) = *X* + (*Y* + *F* (−)) ∼= (*Y* + *X*)+ *F* (−)*.*

so that (*GY* )∞(*X*) ∼= *F* ∞(*Y* + *X*) = *T Y* (*X*). Proposition [3.7](#_bookmark12) then yields the required distributive law. The next lemma describes it concretely.

Lemma 5.1 *In the above situation Proposition* [*3.7*](#_bookmark12) *yields a distributive law*

*Y*

*T Y GY*

*λ* z*G* *Y T Y*

*for each Y* ∈ C*. Ommitting the superscript Y , its components are maps of the form:*

*F* ∞(*Y* + (*Y* + *F* (*X*))) *λX*  *Y* + *F* (*F* ∞(*Y* + *X*))

*Morever, via the two obvious natural transformations κ*2: *F* ⇒ *GY and*

*F* ∞(*κ*2): *F* ∞ ⇒ *T Y we get a commuting diagram of distributive laws:*

*F* ∞*F*  *T* *Y GY*

*λ*∞ *λ*

J∞ J

*FF*  *G* *Y T Y*

Proof The distributive law can be described as composite:

*T Y GY*

=∼ (*GY* )∞*GY*  Proposition [3.7](#_bookmark12) *G* *Y* (*GY* )∞ ∼= *GY T Y*

We shall construct this *λX* explicitly. By first applying the final coalgebra map we get:

*F* ∞(*Y* + (*Y* + *FX*)) *ζ*  (*Y* + (*Y* + *FX*)) + *FF* ∞(*Y* + (*Y* + *FX*))

~

=

The component on the left of the main + on the right-hand-side readily gives a map to the required target, namely:

*Y* + (*Y* + *FX*) [*κ*1*,* id + *F* (*ηX* )] *Y* + *F* (*F* ∞(*Y* + *X*))

For the component on the right we have to do more work. We are done if we can find a map *F* ∞(*Y* + (*Y* + *FX*)) → *F* ∞(*Y* + *X*). Such a map can be obtained via substitution from:

[*η*∞ ◦ *κ*1*,* [*η*∞ ◦ *κ*1*, σ*∞ ◦ *F* (*κ*2)]]

*Y* + (*Y* + *FX*) *Y* +*X Y* +*X Y* +*X*  *F* ∞(*Y* + *X*)

Putting everything together we have the following complicated expression.

*λX* = [ [*κ*1*,* id + *F* (*ηX*)]*,*

*κ*2 ◦ *F* (subst([*η*∞

*Y* +*X*

* + *κ*1*,* [*η*∞

*Y* +*X*

* + *κ*1*, σ*∞

*Y* +*X*

* + *F* (*κ*2)]])) ] ◦ *ζY* +(*Y* +*FX*)*.*

It is not hard to check that the distributive laws are preserved, as claimed at the end of the lemma.

Lemma 5.2 *For each Y* ∈ C*, the object F* ∞(*Y* ) *carries a ﬁnal λY -bialgebra structure:*

*T Y* (*F* ∞(*Y* )) *ξY*  *F* ∞(*Y* ) *ζY*  *G* *Y* (*F* ∞(*Y* ))

~

=

*F* ∞(*Y* + *F* ∞(*Y* )) *Y* + *F* (*F* ∞(*Y* ))

*where ξY* = *subst*([*η*∞*, id*])*.*

*Y*

Proof By Lemma [2.3](#_bookmark1) there is on *F* ∞(*Y* ) an Eilenberg-Moore algebra struc- ture *ξY* : *T Y* (*F* ∞(*Y* )) → *F* ∞(*Y* ) forming a final *λY* -bialgebra. We establish that it is of the form *ξY* = subst([*η*∞*,* id]) by checking that it satisfies the

*Y*

defining equation in Lemma [2.3](#_bookmark1). We shall drop superscripts as usual.

*G*(*ξY* ) ◦ *λF* ∞*Y* ◦ *T* (*ζY* )

= *G*(*ξY* ) ◦ [ *,* ] ◦ *ζY* +(*Y* +*FF* ∞*Y* ) ◦ *F* (id + *ζY* )

∞

= *G*(*ξY* ) ◦ [ *,* ] ◦ ((id + *ζY* )+ *FF* (id + *ζY* )) ◦ *ζY* +*F* ∞*Y*

∞

= (id + *F* (*ξY* )) ◦ [ [*κ*1*,* id + *F* (*ηF* ∞*Y* )] ◦ (id + *ζY* )*,*

*κ*2 ◦ *F* (subst( )) ◦ *FF* (id + *ζY* )] ◦ *ζY* +*F* ∞*Y*

∞

= [ [*κ*1*,* (id + *F* (*ξY* ◦ *ηF* ∞*Y* )) ◦ *ζY* ]*,*

*κ*2 ◦ *F* (*ξY* ◦ subst( ) ◦ *F* (id + *ζY* )) ] ◦ *ζY* +*F* ∞*Y*

∞

= [ [*κ*1*,* (id + *F* (*ξY* ◦ *η*∞ ∞*Y* ◦ *κ*2)) ◦ *ζY* ]*,*

*Y* +*F*

(∗)

*κ*2 ◦ *F* (subst(*ξY* ◦ ◦ (id + *ζY* ))) ] ◦ *ζY* +*F* ∞*Y*

= [ [*κ*1*,* (id + *F* (id) ◦ *ζY* ]*,*

∞ ∞ ∞

*κ*2 ◦ *F* (subst([*ηY ,* [*ηY , τY* ]] ◦ (id + *ζY* ))) ] ◦ *ζY* +*F* ∞*Y*

= [ [*κ*1*, ζY* ]*,*

*κ*2 ◦ *F* (subst([*ηY ,* id]) ] ◦ *ζY* +*F* ∞*Y*

∞

= [ *ζY* ◦ [*η*∞*,* id]*,*

*Y*

∞

*ζY* ◦ *τY* ◦ *F* (*ξY* )] ◦ *ζY* +*F* ∞*Y*

∞ ∞

= *ζY* ◦ [ [*ηY ,* id]*, τY* ◦ *F* (*ξY* )] ◦ *ζY* +*F* ∞*Y*

= *ζY* ◦ *ξY ,* by Lemma [3.5](#_bookmark10) (v)*.*

The marked step (∗) in this calculation is explained as follows.

*ξY* ◦ *σ*∞ ∞ ◦ *F* (*κ*2) = subst([*η*∞*,* id]) ◦ *τ* ∞ ∞

* + *F* (*η*∞ ∞ ) ◦ *F* (*κ*2)

*Y* +*F Y*

*Y Y* +*F Y*

*Y* +*F Y*

= *τ* ∞ ◦ *F* (subst([*η*∞*,* id])) ◦ *F* (*η*∞ ∞ ) ◦ *F* (*κ*2)

*Y Y Y* +*F Y*

= *τ* ∞ ◦ *F* ([*η*∞*,* id]) ◦ *F* (*κ*2)

*Y Y*

= *τ* ∞*.*

*Y*

We are finally in a position to see that ∞-equations and solutions are a special case of *λ*-equations and solutions. This is our main result.

Theorem 5.3 *Let F* : C → C *be a functor with ﬁnal coalgebra F* ∞(*X*) −=→

*X* + *F* (*F* ∞(*X*))*. Then:*

1. *A guard g*: *X* → *Y* + *F* (*F* ∞(*Y* + *X*)) *for an* ∞*-equation e*: *X* → *F* ∞(*Y* +

*X*) *is a λY -equation, for the distributive law λY from Lemma* [*5.1*](#_bookmark19)*.*

1. *A solution sol*(*e*): *X* → *F* ∞(*Y* ) *of a guarded* ∞*-equation e is the same thing as a solution of its guard g—as a λY -equation—in the ﬁnal λY - bialgebra of Lemma* [*5.2*](#_bookmark20)*.*

Proof The first point is obvious, so we concentrate on the second one. We assume that we can write the guarded ∞-equation *e*: *X* → *F* ∞(*Y* + *X*) as

−1

*e* = *ζ*

*Y* +*X*

* (*κ*1 + id) ◦ *g*, like in ([4](#_bookmark17)), where *g*: *X* → *Y* + *F* (*F* ∞(*Y* + *X*)) is the

guard (or *λ*-equation). We observe for a map *f* : *X* → *F* ∞(*Y* ),

*f* is a solution of the *λ*-equation *g* (see Definition [2.4](#_bookmark3))

⇐⇒ *ζY* ◦ *f* = *G*(*ξY* ) ◦ *GT* (*f* ) ◦ *g*

⇐⇒ *f* = *ζ*−1 ◦ *G*(*ξY* ) ◦ *GT* (*f* ) ◦ *g*

*Y*

= [*η*∞*,τ* ∞] ◦ (id + *F* (*ξY* )) ◦ (id + *FF* ∞(id + *f* )) ◦ *g*

*Y Y*

= [*η*∞*,τ* ∞ ◦ *F* (*ξY* ) ◦ *FF* ∞(id + *f* )] ◦ *g*

*Y Y*

= [*η*∞*,τ* ∞ ◦ *F* (subst([*η*∞*,* id]) ◦ *F* ∞(id + *f* ))] ◦ *g*

*Y Y Y*

= [*η*∞*,τ* ∞ ◦ *F* (subst([*η*∞*,* id] ◦ (id + *f* )))] ◦ *g*

*Y Y Y*

= [*η*∞*,* subst([*η*∞*,f* ]) ◦ *τ* ∞

] ◦ *g*

*Y Y Y* +*X*

= subst([*η*∞*,f* ]) ◦ [*η*∞ *,τ* ∞

] ◦ *g*

*Y Y* +*X Y* +*X*

= subst([*η*∞*,f* ]) ◦ *ζ*−1

◦ (*κ*1 + id) ◦ *g*

*Y Y* +*X*

= subst([*η*∞*,f* ]) ◦ *e*

*Y*

⇐⇒ *f* is a solution of the ∞-equation *e* (see Definition [4.1](#_bookmark16))*.*

# Conclusion

We have unified the area of coinductive solutions of equations by showing that one notion developed in [[2](#_bookmark22)] (following [[12](#_bookmark32)]) is an instance of a more general notion from [[4](#_bookmark23)] based on distributive laws.

*Acknowledgments*

Thanks to an anonymous referee for suggesting many improvements.

# References

1. P. Aczel. *Non-well-founded sets*. CSLI Lecture Notes 14, Stanford, 1988.
2. P. Aczel, J. Ad´amek, S. Milius, and J. Velebil. Infinite trees and completely iterative theories: a coalgebraic view. *Theor. Comp. Sci.*, 300 (1-3):1–45, 2003.
3. J. Ad´amek, S. Milius, and J. Velebil. On rational monads and free iterative theories. In R. Blute and P. Selinger, editors, *Category Theory and Computer Science 2002*, number 69 in Elect. Notes in Theor. Comp. Sci. Elsevier, Amsterdam, 2003. <http://www.elsevier.com/locate/entcs/volume69.html> .
4. F. Bartels. *On generalised coinduction and probabilistic speciﬁcation formats. Distributive laws in coalgebraic modelling*. PhD thesis, Free Univ. Amsterdam, 2004. <http://homepages.cwi.nl/~bartels/>.
5. J. Barwise and L.S. Moss. *Vicious Circles: On the Mathematics of Non-wellfounded Phenomena*. CSLI Lecture Notes 60, Stanford, 1996.
6. C.C. Elgot. Monadic computation and iterative algebraic theories. In H.E. Rose and J.C. Shepherson, editors, *Logic Colloquium ’73*, pages 175–230, Amsterdam, 1975. North-Holland.
7. C.C. Elgot, S.L. Bloom, and R. Tindell. The algebraic structure of rooted trees. *Journ. Comp. Syst. Sci*, 16:361–399, 1978.
8. B. Jacobs and J. Rutten. A tutorial on (co)algebras and (co)induction. *EATCS Bulletin*, 62:222–259, 1997.
9. M. Lenisa. From set-theoretic coinduction to coalgebraic coinduction: some results, some problems. In B. Jacobs and J. Rutten, editors, *Coalgebraic Methods in Computer* *Science*, number 19 in Elect. Notes in Theor. Comp. Sci. Elsevier, Amsterdam, 1999. <http://www.elsevier.com/locate/entcs/volume19.html> .
10. M. Lenisa, J. Power, and H. Watanabe. Distributivity for endofunctors, pointed and co- pointed endofunctors, monads and comonads. In H. Reichel, editor, *Coalgebraic Methods in* *Computer Science*, number 33 in Elect. Notes in Theor. Comp. Sci. Elsevier, Amsterdam, 2000. <http://www.elsevier.com/locate/entcs/volume33.html> .
11. S. Milius. On iterable endofunctors. In R. Blute and P. Selinger, editors, *Category Theory and Computer Science 2002*, number 69 in Elect. Notes in Theor. Comp. Sci. Elsevier, Amsterdam, 2003. <http://www.elsevier.com/locate/entcs/volume69.html> .
12. L.S. Moss. Parametric corecursion. *Theor. Comp. Sci.*, 260(1-2):139–163, 2001.
13. J. Rutten. Behavioural differential equations: a coinductive calculus of streams, automata, and power series. *Theor. Comp. Sci.*, 308:1–53, 2003.
14. D. Turi. *Functorial operational semantics and its denotational dual*. PhD thesis, Free Univ. Amsterdam, 1996.
15. D. Turi and G. Plotkin. Towards a mathematical operational semantics. In *Logic in Computer Science*, pages 280–291. IEEE, Computer Science Press, 1997.