Generalised Coinduction

F. Bartels [1]

CWI, P.O. Box 94079, 1090 GB Amsterdam, The Netherlands
http://www.cwi.nl/~bartels

Abstract

We introduce the λ-*coiteration* schema for a distributive law λ of a functor T over a functor F. Under certain conditions it can be shown to uniquely characterise functions into the carrier of a final F-coalgebra, generalising the basic coiteration schema as given by finality. The duals of primitive recursion and course-of-value iteration, which are known extensions of coiteration, arise as instances of our framework. One can furthermore obtain schemata justifying recursive specifications that involve operators such as addition of power series, regular operators on languages, or parallel and sequential composition of processes.

Next, the same type of distributive law λ is used to generalise coinductive proof techniques. To this end, we introduce the notion of a λ-*bisimulation* relation. It specialises to what could be called bisimulation up-to-equality or bisimulation up-to-context for contexts built from operators of the type mentioned above. We state that every such relation is contained in some larger conventional bisimulation and demonstrate that this principle leads to simpler bisimilarity proofs using less complex relations.

1 Introduction

Around the early nineties, the categorical notion of a *final* F-*coalgebra* for a functor F on some category C was found useful for the abstract description of possibly infinite objects in computer science. Examples are datatypes such as infinite streams or dynamical systems like processes or automata (see e.g. the introductions by Jacobs and Rutten [JR96,Rut00b]). Treating these different entities uniformly as behavioural systems of some type F allowed for an abstract formulation of definition and proof principles that have been studied separately for various applications before.

States of a final coalgebra can be characterised using a principle which is given directly by finality. We call it *coiteration* here to distinguish it from

---

derived variants. The main tool for proving states equal is that of an F-*bisimulation* [AM89], a categorical generalisation of notions of bisimulation used for different concrete systems. But these basic principles are often too rigid to nicely cover given examples. Many functions into the carrier of a final coalgebra can be shown not to be coiterative and many statements about behavioural equivalence require bisimulations which are difficult to exhibit and check. Therefore several extensions for particular settings are studied. Examples of a more general type are the definition principles that arise as the duals of primitive recursion and course-of-value iteration. The specification of processes by systems of recursive equations forms another one. For proof purposes one often uses relations which instead of being bisimulations themselves satisfy sufficient conditions for being contained in some bisimulation. These are often called *bisimulations up-to* (see e.g. Milner [Mil89]). Sangiorgi [San98] has introduced a framework for deriving such principles in the context of labelled transition systems.

Categorical formulations of the schemata that form the duals of primitive recursion and course-of-value iteration are known, see e.g. Vene and Uustalu [UV99]. Following common practice, we will call the first one *primitive corecursion.*

A step towards a more general description of extended principles on this level has been made by Lenisa in the course of her comparison of set-theoretic and coalgebraic (categorical) formulations of coinduction [Len99a]. She has demonstrated that her framework captures the arrows obtainable by primitive corecursion and the same can be shown for the dual of course-of-value iteration, but in neither case does her theory directly yield the universal characterisation for these morphisms mentioned above. By modifying Lenisa's approach we arrived at a categorical description of generalised coinductive definition and proof principles based on such a universal characterisation which specialises directly to both principles from above.

Simply speaking, the conventional coiteration schema assigns infinite behaviours to the states of a set X by specifying for each element a direct observation and successor states. Since these successors are taken from X again, the same specification applied to them reveals the second layer of the behaviour, and so on. A different approach is taken by the λ-*coiteration schema* that we introduce, which is parameterised by another functor T and a distributive law λ of T over F: it allows the successor states to be taken from TX instead of X, which may increase the expressiveness of the format in case the former can be regarded as being "richer" than the latter. For the observations to be continued with these successors, the distributive law λ lifts the specification for X to TX. In the main theorems of this paper we show for two additional assumptions that each of them is sufficient for the λ-coiteration schema to uniquely define arrows into the final F-coalgebra.

Making similar use of T and λ, we define the notion of a λ-*bisimulation* and show that the same conditions as for the validity of the λ-coiteration schema

are sufficient to show that all states in such a relation are bisimilar. A small example demonstrates that the technique enables simpler proofs involving less complex relations.

The categorical formulation of primitive corecursion and the dual of course-of-value iteration mentioned above can be obtained from the λ-coiteration schema for suitable instantiations of T and λ, although we do not explain the details here. Instead, we briefly sketch how it can be used to justify specifications involving operators of a certain type. These are the operators that Turi and Plotkin [TP97] have shown to be closely related to those definable by structured transition rules in GSOS format [BIM95]. Presenting these schemata as instances of our framework at the same time produces the corresponding variants of the λ-bisimulation proof technique. For the case of definitions via operators one obtains a notion of bisimulation up-to-context [San98] for multivariate contexts (i.e. context with several "holes") built from operators of the type mentioned above. This provides an abstract justification for the usability of these operators for this purpose.

The theory presented is stated for some category C, but for easy reading our explanations and examples all refer to the category of sets and total functions, Set. This paper is a short version of a CWI technical report under the same name [Bar00]. The long version contains detailed proofs for most of the statements given here and further examples.
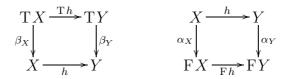
2   (Co)Algebras and (Co)Iteration

In this section we will briefly present a categorical view on algebras and coalgebras. More detailed expositions can be found e.g. among the papers of Jacobs and Rutten [JR96,Rut00b]. We take C and T, F : C → C to denote a category and two functors on it.

Definition 2.1 [T-algebra, F-coalgebra] A T-*algebra* is a pair $\langle X, \beta \rangle$ where X is an object of C and $\beta : TX \to X$ is an arrow in C. We will sometimes call X and β the *carrier* and *operation* of the algebra. Dually, an F-*coalgebra* is a pair $\langle X, \alpha \rangle$ where the operation $\alpha : X \to FX$ is an arrow going into the reversed direction.

Generally, algebra operations can be seen as a means for *constructing* elements of their carrier. The operation of a coalgebra – also called *destruction* or *unfolding* elsewhere – gives us information about its states, either in terms of attributes or (potential) successor states. In the first case we will talk about the *observation* a state allows, in the second about its *dynamics*.

Definition 2.2 [homomorphism] An arrow $h : X \to Y$ is a T-*algebra homomorphism* from one T-algebra $\langle X, \beta_X \rangle$ to another T-algebra $\langle Y, \beta_Y \rangle$ if it makes the left diagram below commute. Similarly, an F-*coalgebra homomorphism* from one F-coalgebra $\langle X, \alpha_X \rangle$ to another F-coalgebra $\langle Y, \alpha_Y \rangle$ is an arrow $h : X \to Y$ making the right diagram commute:

$$
\begin{array}{ccc}
\mathrm{T}X & \xrightarrow{\mathrm{T}h} & \mathrm{T}Y \\
\beta_X \downarrow & & \downarrow \beta_Y \\
X & \xrightarrow{h} & Y
\end{array}
\qquad\qquad
\begin{array}{ccc}
X & \xrightarrow{h} & Y \\
\alpha_X \downarrow & & \downarrow \alpha_Y \\
\mathrm{F}X & \xrightarrow{\mathrm{F}h} & \mathrm{F}Y
\end{array}
$$

We will often just talk about homomorphisms when their type is clear from the context. Algebras and coalgebras together with the corresponding homomorphisms form the categories $\mathsf{Alg}^{\mathrm{T}}$ and $\mathsf{Coalg}_{\mathrm{F}}$ respectively.

Definition 2.3 [final F-coalgebra] A *final* F-*coalgebra* is a final object in the category of F-coalgebras $\mathsf{Coalg}_{\mathrm{F}}$, i.e. a coalgebra – usually denote here by $\langle \Omega_{\mathrm{F}}, \omega_{\mathrm{F}} \rangle$ – such that there exists exactly one homomorphism from every other F-coalgebra to it.

Example 2.4 Some of our later examples will involve infinite streams of real numbers $\sigma \in I\!\!R^{\omega}$. These arise as states of the final coalgebra of the functor $\mathrm{S} : \mathsf{Set} \to \mathsf{Set}$ where $\mathrm{S}X := I\!\!R \times X$. Coalgebras of this functor have the shape $\langle X, \langle o, s \rangle \rangle$ for a set X and two functions $o : X \to I\!\!R$ and $s : X \to X$. That is, for each state we can observe a real number and move to a successor state. The infinite streams of real numbers form the final S-coalgebra $\langle I\!\!R^{\omega}, \langle \mathtt{head}, \mathtt{tail} \rangle \rangle$, where for each stream $\sigma = \langle \sigma_0, \sigma_1, \ldots \rangle \in I\!\!R^{\omega}$ the observation is given by its first element $\mathtt{head}(\sigma) := \sigma_0$ and the successor by the stream that remains after removing it, $\mathtt{tail}(\sigma) := \sigma' := \langle \sigma_1, \sigma_2, \ldots \rangle$. We will call S-coalgebras *stream systems* in the following.

The finality of an F-coalgebra $\langle \Omega_{\mathrm{F}}, \omega_{\mathrm{F}} \rangle$ yields an arrow $h : X \to \Omega_{\mathrm{F}}$ for every F-coalgebra operation α on X by assuming it to be the unique homomorphism from $\langle X, \alpha \rangle$ to $\langle \Omega_{\mathrm{F}}, \omega_{\mathrm{F}} \rangle$. Such an arrow is then called the *coiterative arrow defined (or induced) by* α:

$$
\begin{array}{ccc}
X & \xdashrightarrow{\exists! h} & \Omega_{\mathrm{F}} \\
\forall \alpha \downarrow & & \downarrow \omega_{\mathrm{F}} \\
\mathrm{F}X & \xdashrightarrow[\mathrm{F}h]{} & \mathrm{F}\Omega_{\mathrm{F}}
\end{array}
$$

Example 2.5 [Coiteration for Streams] The coiteration schema for stream systems from Example 2.4 states that for every pair of functions $o : X \to I\!\!R$ and $s : X \to X$ there is a unique function $f : X \to I\!\!R^{\omega}$ satisfying

$$\mathtt{head}(f(x)) = o(x),$$
$$\mathtt{tail}(f(x)) = f(s(x)).$$

As an example we take a look at the element-wise addition of two streams. By the coiteration schema there is a unique function $\oplus : I\!\!R^{\omega} \times I\!\!R^{\omega} \to I\!\!R^{\omega}$ satisfying the following two equations:

$$\mathtt{head}(\sigma \oplus \tau) = \mathtt{head}(\sigma) + \mathtt{head}(\tau)$$
$$\mathtt{tail}(\sigma \oplus \tau) = \mathtt{tail}(\sigma) \oplus \mathtt{tail}(\tau)$$

As a proof principle, one can show that for a final F-coalgebra $\langle \Omega_F, \omega_F \rangle$ two arrows $h_1, h_2 : X \to \Omega_F$ are equal by providing an F-coalgebra operation on X for which both arrows are homomorphisms. Most of the time one exploits this property for proving that two states show the same behaviour, i.e. that they are mapped onto the same state of the final coalgebra by the coiterative morphisms. Technically one uses a derived principle based on the notion of *bisimulation*. We work with a definition in terms of spans:

Definition 2.6 [span] In a category C, a *span* $\mathcal{R} = \langle R, r_1, r_2 \rangle$ between two C objects X and Y consists of an object R and two arrows $r_1 : R \to X$ and $r_2 : R \to Y$. A span between X and itself is called a span *on* X.

There is a preorder $\preceq$ of spans between the objects X and Y defined as $\langle R, r_1, r_2 \rangle \preceq \langle S, s_1, s_2 \rangle$ if and only if there is an arrow $f : R \to S$ such that both triangles in the following diagram commute:

$$
\begin{array}{ccc}
 & R & \\
r_1 \swarrow & \downarrow \exists f & \searrow r_2 \\
X & & Y \\
s_1 \nwarrow & \downarrow & \nearrow s_2 \\
 & S &
\end{array}
$$

Definition 2.7 [Bisimulation] For a functor $F : C \to C$, an F-*bisimulation* between two F-coalgebras $\langle X, \alpha_X \rangle$ and $\langle Y, \alpha_Y \rangle$ is a span $\mathcal{B} = \langle B, b_1, b_2 \rangle$ between their carriers X and Y, such that there is an F-coalgebra operation $\gamma : B \to FB$ turning b_1 and b_2 into homomorphisms:

$$
\begin{array}{ccc}
X & \xleftarrow{b_1} B \xrightarrow{b_2} & Y \\
\alpha_X \downarrow & \downarrow \exists \gamma & \downarrow \alpha_Y \\
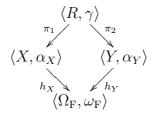FX & \xleftarrow[F b_1]{} FB \xrightarrow[F b_2]{} & FY
\end{array}
$$

A bisimulation between a coalgebra $\langle X, \alpha \rangle$ and itself is called a bisimulation *on* $\langle X, \alpha \rangle$.

In Set one often only considers bisimulations which are relations, i.e. spans $\langle B, \pi_1, \pi_2 \rangle$ for a relation $B \subseteq X \times Y$ (see e.g. Rutten [Rut00b]). We use the formulation based on spans because it generalises to other categories and is sometimes easier to work with. We will still often talk about bisimulation *relations*. This is justified by the observation that every span $\langle R, r_1, r_2 \rangle$ in Set can be regarded as representing the image $\langle r_1, r_2 \rangle[R] \subseteq X \times Y$. The order $\preceq$ of spans corresponds to relation inclusion of images. Furthermore the image of a (span) bisimulation is again a (relational) bisimulation.

One usually calls two states *bisimilar* if they are related by some bisimulation relation and takes bisimilarity to be the notion of behavioural equivalence. This is justified by the fact that in case a final F-coalgebra exists bisimilar states are identified by the coiterative morphisms. This can easily be seen as follows:

Let $B \subseteq X \times Y$ be a bisimulation between the F-coalgebras $\langle X, \alpha_X \rangle$ and $\langle Y, \alpha_Y \rangle$ containing the pair $\langle x, y \rangle$, where the bisimulation property is witnessed by $\gamma : B \to FB$. Let h_X and h_Y denote the coiterative arrows from $\langle X, \alpha_X \rangle$

and $\langle Y, \alpha_Y \rangle$ to the final coalgebra $\langle \Omega_F, \omega_F \rangle$, then one gets the diagram below in Coalg_F, commuting by finality. This yields $h_X(x) = h_X(\pi_1(\langle x, y \rangle)) = h_Y(\pi_2(\langle x, y \rangle)) = h_Y(y)$ as wanted.

$$
\begin{array}{ccc}
 & \langle R, \gamma \rangle & \\
{\scriptstyle \pi_1} \swarrow & & \searrow {\scriptstyle \pi_2} \\
\langle X, \alpha_X \rangle & & \langle Y, \alpha_Y \rangle \\
{\scriptstyle h_X} \searrow & & \swarrow {\scriptstyle h_Y} \\
 & \langle \Omega_F, \omega_F \rangle &
\end{array}
$$

3   Definition by λ-Coiteration

The states of a final F-coalgebra $\langle \Omega_F, \omega_F \rangle$ can be taken to represent abstract behaviours of the type F, as we did already in the explanations above. An arrow $f : X \to \Omega_F$ assigns such behaviours to the elements of X. The coiteration schema allows us to define this function such that it maps every element of X to the behaviour it exhibits as a state in a given F-coalgebra $\langle X, \alpha \rangle$. Unfortunately, for many functions $f : X \to \Omega_F$ that one wants to specify there is no F-coalgebra operation α on X itself making f the coiterative morphism or it may not be obvious from the given specification of f.

In this section we are first going to present a specification of this kind. Then we will formulate the encountered pattern more abstractly and – as the main theorems of this paper – state sufficient conditions for it to uniquely characterise arrows into the carrier of a final coalgebra. This yields a definition schema that we call λ-*coiteration*.

3.1   Example: Multiplication of Formal Power Series

Like Rutten [Rut00a], we will consider infinite streams of real numbers $\sigma = \langle \sigma_0, \sigma_1, \ldots \rangle \in \mathbb{R}^\omega$ as representations of formal power series $\sum_{i=0}^{\infty} \sigma_i X^i$. The operation $\oplus$ from Example 2.5 turns out to be such that $\sigma \oplus \tau$ represents the sum of the two series represented by σ and τ. Similarly, we would like to define a binary operation $\otimes$ on $\mathbb{R}^\omega$ such that $\sigma \otimes \tau$ represents the convolution product of the two series represented by σ and τ. After some computation (c.f. [Rut00a]) one arrives at the following specification, where the constant series $r \in \mathbb{R}$ is represented by $[r] = \langle r, 0, 0, \ldots \rangle \in \mathbb{R}^\omega$: [2]

$$\mathtt{head}(\sigma \otimes \tau) = \mathtt{head}(\sigma) \cdot \mathtt{head}(\tau),$$
$$\mathtt{tail}(\sigma \otimes \tau) = ([\mathtt{head}(\sigma)] \otimes \mathtt{tail}(\tau)) \oplus (\mathtt{tail}(\sigma) \otimes \tau).$$

These two equations do not form a coiterative definition as in Example 2.5 because of the use of $\oplus$ in the expression for the tail. To get a better picture of the type of definition we have here, we set $X := \mathbb{R}^\omega \times \mathbb{R}^\omega$ and $o : X \to \mathbb{R}$, $s_1, s_2 : X \to X$ with

---

[2] Coiteratively: $\mathtt{head}([r]) = r$ and $\mathtt{tail}([r]) = [0]$.

$$o(\sigma, \tau) := \texttt{head}(\sigma) \cdot \texttt{head}(\tau),$$
$$s_1(\sigma, \tau) := \langle [\texttt{head}(\sigma)], \texttt{tail}(\tau) \rangle,$$
$$s_2(\sigma, \tau) := \langle \texttt{tail}(\sigma), \tau \rangle.$$

Using this, the equations above can be stated alternatively by asking $\otimes$ to fit into the diagram below:

$$
\begin{array}{ccc}
X & \xdashrightarrow{\otimes} & I\!\!R^\omega \\
{\scriptstyle \langle o, \langle s_1, s_2 \rangle \rangle} \downarrow & & \downarrow {\scriptstyle \langle \texttt{head}, \texttt{tail} \rangle} \\
I\!\!R \times (X \times X) & \xdashrightarrow[{\texttt{Id}_R \times (\oplus \circ (\otimes \times \otimes))}]{} & I\!\!R \times I\!\!R^\omega
\end{array}
$$

In the following the question to be addressed more generally amounts here to asking whether this condition uniquely defines the arrow $\otimes$.

3.2 The general Pattern

To describe the format of the specification from above more abstractly, we make the following two definitions:

Definition 3.1 [T-extension] Let $\beta : \mathrm{T}Y \to Y$ be a T-algebra. For a given arrow $f : X \to Y$, we call $f|^\beta := \beta \circ \mathrm{T}f : \mathrm{T}X \to Y$ the T-*extension of f along β*:

$$
\begin{array}{ccc}
\mathrm{T}X & \xrightarrow{\mathrm{T}f} & \mathrm{T}Y \\
 & \searrow {\scriptstyle f|^\beta} & \downarrow {\scriptstyle \beta} \\
X & \xrightarrow{f} & Y
\end{array}
$$

Definition 3.2 [homomorphism up-to-β] Let $\langle X, \phi \rangle$ be an FT-coalgebra and $\langle Y, \alpha \rangle$ be an F-coalgebra with a T-algebra operation $\beta : \mathrm{T}Y \to Y$ on its carrier. An arrow $f : X \to Y$ is called a *homomorphism up-to-β* from $\langle X, \phi \rangle$ to $\langle Y, \alpha \rangle$, if it makes the following diagram commute (note that in the picture we added an arrow for β to visualise its typing, though it does not contribute to the commutativity expressed):

$$
\begin{array}{ccc}
 & & \mathrm{T}Y \\
 & & \downarrow {\scriptstyle \beta} \\
X & \xrightarrow{f} & Y \\
{\scriptstyle \phi} \downarrow & & \downarrow {\scriptstyle \alpha} \\
\mathrm{F}\mathrm{T}X & \xrightarrow[\mathrm{F}f|^\beta]{} & \mathrm{F}Y
\end{array}
$$

By taking the functor $\mathrm{T} := \mathrm{Id} \times \mathrm{Id}$, $\otimes$ becomes a homomorphism up-to-$\oplus$ from the $\mathrm{S}\mathrm{T}$-coalgebra $\langle X, \langle o, \langle s_1, s_2 \rangle \rangle \rangle$ to $\langle I\!\!R^\omega, \langle \texttt{head}, \texttt{tail} \rangle \rangle$, the final S-coalgebra, since we can rewrite the arrow at the bottom of the diagram intended to specify $\otimes$ as follows:

$$\texttt{Id}_R \times (\oplus \circ (\otimes \times \otimes)) = \mathrm{S}(\oplus \circ (\otimes \times \otimes)) = \mathrm{S}(\oplus \circ \mathrm{T}\otimes) = \mathrm{S}\otimes|^\oplus.$$

3.3 Distributive Laws

We are looking for a framework in which a homomorphism up-to-β into a final F-coalgebra $\langle \Omega_F, \omega_F \rangle$ uniquely exists for a given T-algebra operation β on its carrier. Since it is easy to see that this is not the case for all β, one needs to restrict the class of operations allowed. The unique existence of a homomorphism up-to-β means that an FT-operation on an object X specifies an arrow $h : X \to \Omega_F$, i.e. an assignment of abstract F-behaviours to the elements in X.

This suggest that the way β evaluates elements of $T\Omega_F$ can be described more globally via a specification of the interaction of T and F. In our approach we assume this to be given by a *distributive law* λ of T over F defined below, which uniquely determines the T-algebra operation on the final F-coalgebra. In the case of our example, the operation $\otimes$ can be seen to arise from such a distributive law λ.

Definition 3.3 [distributive law] Let $T, F : C \to C$ be two functors. A natural transformation $\lambda : TF \Rightarrow FT$ is called a *distributive law* of T over F. We will sometimes alternatively use the phrase that T *distributes over* F *via* λ.

A major application of the notion of a distributive law in computer science has been given by Turi and Plotkin [TP97] where the two functors were coming as a monad and a comonad respectively and additional coherence axions involving the extra structure were considered (see the work by Power and Watanabe [PW99] for a structured account of this setting). Subsequently, distributive laws were also used in situations where less structure was assumed for the functors and the corresponding coherence axioms were dropped (see [LPW00]). For now we only treat plain functors and no coherence axioms, but we will later encounter a case requiring more structure.

Regarding the elements from TX as structured entities containing elements from X as arguments, a distributive law tells us how to assign an F-step to such an entity given the steps the arguments can do. With this information, we can derive an F-coalgebra operation for TX from the F-coalgebra $\langle X, \alpha \rangle$:

Definition 3.4 [λ-lifting] Given a distributive law λ of a functor T over a functor F, we can lift $T : C \to C$ to the functor $T_\lambda : \mathsf{Coalg}_F \to \mathsf{Coalg}_F$ on the F-coalgebras by setting

$$T_\lambda \langle X, \alpha \rangle := \langle TX, lift_\alpha^\lambda \rangle \quad \text{and} \quad T_\lambda h := Th,$$

for an F-coalgebra $\langle X, \alpha \rangle$ and an F-coalgebra homomorphism h, where

$$lift_\alpha^\lambda := \lambda_X \circ T\alpha : TX \to FTX.$$

For this definition to make sense we need the following lemma which is easily proved using the assumption on λ being natural:

Lemma 3.5 (see also [Rut00b, Theorem 15.3]) *Let h be an F-coalgebra homomorphism from $\langle X, \alpha_X \rangle$ to $\langle Y, \alpha_Y \rangle$, then Th is a homomorphism from $T_\lambda \langle X, \alpha_X \rangle$ to $T_\lambda \langle Y, \alpha_Y \rangle$.*

In the case of our example the distributive law λ of T := Id $\times$ Id over S should give a global description of the addition of two states. For a set X we define $\lambda_X : (I\!R \times X) \times (I\!R \times X) \to I\!R \times (X \times X)$ as

$$(1) \qquad \lambda_X(\langle o_x, s_x \rangle, \langle o_y, s_y \rangle) := \langle o_x + o_y, \langle s_x, s_y \rangle \rangle.$$

It is easy to verify that we get a natural transformation indeed. Given a stream system $\langle X, \langle o, s \rangle \rangle$, the resulting system $T_\lambda \langle X, \langle o, s \rangle \rangle$ has pairs of states from X as states. Unfolding such a pair $\langle x, y \rangle$ yields the sum of the two observations $o(x) + o(y)$ and the pair of the successors $\langle s(x), s(y) \rangle$, that is the behaviour of $\langle x, y \rangle$ in $T_\lambda \langle X, \langle o, s \rangle \rangle$ is the (stream) sum of the behaviours of x and y in $\langle X, \langle o, s \rangle \rangle$ as wanted.

With the lifting from above, a distributive law λ assigns F-behaviours to the set TX, given the behaviours of the elements from X. The generalised coinduction schema we are about to state and justify involves T-algebra operations β which preserve these behaviours. More precisely, for a given F-coalgebra $\langle X, \alpha \rangle$ we consider homomorphisms β from $T_\lambda \langle X, \alpha \rangle$ to $\langle X, \alpha \rangle$. It turns out that this situation is captured by the notion of a λ-bialgebra occurring in the literature, thus we recall its definition here:

Definition 3.6 [T,F-bialgebra, T,F-bialgebra homomorphism, λ-bialgebra] A T,F-*bialgebra* is a triple $\langle X, \beta, \alpha \rangle$ of an object X and two arrows $\beta : TX \to X$ and $\alpha : X \to FX$, i.e. a T-algebra and an F-coalgebra operation on a common carrier. Given two T,F-bialgebras $\langle X, \beta_X, \alpha_X \rangle$ and $\langle Y, \beta_Y, \alpha_Y \rangle$, a T,F-*bialgebra homomorphism* from $\langle X, \beta_X, \alpha_X \rangle$ to $\langle Y, \beta_Y, \alpha_Y \rangle$ is an arrow $h : X \to Y$ which is both, a T-algebra homomorphism from $\langle X, \beta_X \rangle$ to $\langle Y, \beta_Y \rangle$ and an F-coalgebra homomorphism from $\langle X, \alpha_X \rangle$ to $\langle Y, \alpha_Y \rangle$. Like with T-algebras and F-coalgebras, T,F-bialgebras and their homomorphisms form a category, denoted by Bialg_F^T.
Given a distributive law λ of T over F, a λ-*bialgebra* is a T,F-bialgebra $\langle X, \beta, \alpha \rangle$ such that the following diagram commutes:

The full subcategory of Bialg_F^T containing all λ-bialgebras is denoted by λ-Bialg.

Note that indeed the definition of $\langle X, \beta, \alpha \rangle$ being a λ-bialgebra is equivalent to saying that β is an F-coalgebra homomorphism from $T_\lambda \langle X, \alpha \rangle$ to $\langle X, \alpha \rangle$ as wanted above. With this remark, for a final F-coalgebra $\langle \Omega_F, \omega_F \rangle$ there is exactly one choice for a T-algebra operation $\beta_\lambda : T\Omega_F \to \Omega_F$ such that $\langle \Omega_F, \beta_\lambda, \omega_F \rangle$ is a λ-bialgebra, namely the coiterative morphism from $T_\lambda \langle \Omega_F, \omega_F \rangle$ to $\langle \Omega_F, \omega_F \rangle$.

Bialgebras for a distributive law play an important role in the paper by

Turi and Plotkin [TP97] as well. As with the definition of the distributive law, our notion of a λ-bialgebra differs from theirs due to the fact that we do not assume T and F to come as a monad and a comonad: in their setting $\langle X, \beta \rangle$ and $\langle X, \alpha \rangle$ are assumed to be an algebra for the monad and a coalgebra for the comonad respectively.

In our example the bialgebra under consideration was $\langle I\!R^\omega, \oplus, \langle \texttt{head}, \texttt{tail} \rangle \rangle$. It is easily verified to be a λ-bialgebra for λ as in (1).

3.4   The λ-Coiteration Definition Schema

We are now ready to define our new schema called λ-*coiteration* for a distributive law λ of a functor T over F:

Definition 3.7 [λ-Coiterative Arrow] Assume we are given a functor F with a final coalgebra $\langle \Omega_F, \omega_F \rangle$ and a distributive law λ of another functor T over F. For an FT-coalgebra $\langle X, \phi \rangle$ we call an arrow $f : X \to \Omega_F$ a λ-*coiterative arrow induced by* ϕ if it is a homomorphism up-to-β_λ from $\langle X, \phi \rangle$ to $\langle \Omega_F, \omega_F \rangle$ for the unique arrow β_λ such that $\langle \Omega_F, \beta_\lambda, \omega_F \rangle$ is a λ-bialgebra (c.f. remark following Definition 3.6):

$$
\begin{array}{ccc}
 & & T\Omega_F \\
 & & \downarrow{\scriptstyle \beta_\lambda} \\
X & \dashrightarrow[f]{} & \Omega_F \\
{\scriptstyle \phi}\downarrow & & \downarrow{\scriptstyle \omega_F} \\
FTX & \xrightarrow[F f|^{\beta_\lambda}]{} & F\Omega_F
\end{array}
$$

We want to use the above diagram as a definition for f, calling it the λ-*coiteration schema*. To be able to do so, we need to prove the unique existence of a λ-coiterative arrow for any given FT-coalgebra $\langle X, \phi \rangle$. The two answers we are going to present work by establishing a one-to-one correspondence between the λ-coiterative arrows and the coiterative arrows from an F-coalgebra $\langle L_X, \alpha_\phi \rangle$ constructed from $\langle X, \phi \rangle$. The unique existence of the former then follows from that of the latter. The constructions require additional assumptions on C or T and λ respectively:
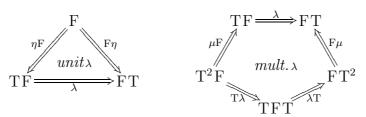
Theorem 3.8 (λ-Coiteration (1)) *Assume the category C has countable co-products and we are given a functor $F : C \to C$ with a final coalgebra $\langle \Omega_F, \omega_F \rangle$ and another functor $T : C \to C$ that distributes over F via λ. Then, for every FT-coalgebra $\langle X, \phi \rangle$ there exists a unique λ-coiterative arrow induced by ϕ.*

For this approach, one constructs an F-coalgebra on the countable coproduct $\sum_{i=0}^\infty T^i X$. The operation is a case analysis involving $\phi_i : T^i X \to F T^{i+1} X$ and appropriate coproduct injections where the ϕ_i are obtained as liftings of ϕ via λ. The correspondence of homomorphisms up-to-β_λ $f : X \to \Omega_F$ and homomorphisms $h : \sum_{i=0}^\infty T^i X \to \Omega_F$ is given by $f \mapsto [f_i]_{i=0}^\infty$ and $h \mapsto h \circ \texttt{in}_0$ where $f_0 := f$, $f_{i+1} := f_i|^{\beta_\lambda}$, $\texttt{in}_i$ is the ith coproduct injection, and $[.]_{i=0}^\infty$ is

the countable case analysis. A detailed proof can be found in [Bar00].

Using this theorem we can finally answer the question whether the specification in Section 3.1 uniquely defined the function $\otimes$ positively: The example was living in the category Set, which of course has countable coproducts.

Theorem 3.9 (λ-Coiteration (2)) *Assume we are given a functor* $F : C \to C$ *with a final coalgebra* $\langle \Omega_F, \omega_F \rangle$, *a monad* $\langle T, \eta, \mu \rangle$ *in* C *and a distributive law* λ *of this monad over* F, *i.e. a distributive law of* T *over* F *such that the following diagrams commute, which we will refer to as the* unit *and* multiplication law *of* λ:

$$
\begin{array}{ccc}
 & F & \\
\eta F \swarrow & & \searrow F\eta \\
 & unit\,\lambda & \\
TF & \xrightarrow{\ \ \lambda\ \ } & FT
\end{array}
\qquad
\begin{array}{ccc}
TF & \xrightarrow{\ \lambda\ } & FT \\
\mu F \Big\uparrow & mult.\,\lambda & \Big\uparrow F\mu \\
T^2 F & & FT^2 \\
T\lambda \searrow & & \nearrow \lambda T \\
 & TFT &
\end{array}
$$

Then, for every FT-*coalgebra* $\langle X, \phi \rangle$ *there exists a unique* λ-*coiterative arrow induced by* ϕ.

In case T is coming as a monad, for every $i \in \mathbb{N}$ the elements from $T^i X$ can be represented within TX, either using η or (possibly several times) μ. The additional assumptions on λ ensure that these mappings "preserve behaviours", such that it suffices to take TX as the carrier of the F-coalgebra to be constructed instead of $\sum_{i=0}^{\infty} T^i X$ as above. The correspondence of homomorphisms up-to-β_λ $f : X \to \Omega_F$ and homomorphisms $h : TX \to \Omega_F$ is given by $f \mapsto f|^{\beta_\lambda}$ and $h \mapsto h \circ \eta_X$.

As a remark, the above argument marks the λ-coiterative arrows obtained by Theorem 3.9 as being coiterative up-to-$\langle T, \eta \rangle$ in the framework of Lenisa [Len99a]. Actually the setting she uses to reason about these morphisms corresponds to yet another variation of the assumptions which lies in between those of the two theorems given here: it assumes the category C to have colimits of ω-chains and the functor T is supposed to come as a pointed functor for which λ satisfies the unit law.

3.5 Properties and an Extension

As an interesting though trivial observation note that the coiteration schema itself arises as the λ-coiteration schema for the identity functor distributing over F via the natural transformation $F \mathsf{Id} : \mathsf{Id}F \Rightarrow F\mathsf{Id}$.

Another observation in this direction is that when there is a unit natural transformation $\eta : \mathsf{Id} \Rightarrow T$ for which λ satisfies the unit law (which is assumed in Theorem 3.9 anyway), then the corresponding instance of the λ-coiteration schema individually generalises the coiteration schema: one can obtain a coiterative arrow induced by any F-coalgebra operation $\alpha : X \to FX$ as the λ-coiterative arrow induced by $F\eta_X \circ \alpha : X \to FTX$ (the crucial observation

is that the assumption on λ makes β_λ an algebra operation for the pointed functor $\langle \mathrm{T}, \eta \rangle$, which yields $f|^{\beta_\lambda} \circ \eta_X = f$).

One simple extension of the coiteration schema arises as the dual of primitive recursion and is therefore sometimes called *primitive corecursion* (see e.g. [UV99]). It states that in a category with binary coproducts every arrow $\phi : X \to \mathrm{F}(X + \Omega_\mathrm{F})$ uniquely determines a morphism $f : X \to \Omega_\mathrm{F}$ making the diagram below commute.
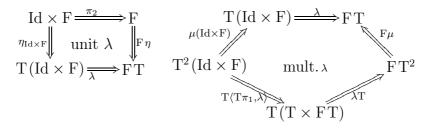
$$
\begin{array}{ccc}
X & \overset{\exists! \underline{h}}{- - - - - -} \!\!\!\!> & \Omega_\mathrm{F} \\
{\scriptstyle \forall \phi} \downarrow & & \downarrow {\scriptstyle \omega_\mathrm{F}} \\
\mathrm{F}(X + \Omega_\mathrm{F}) & \underset{\mathrm{F}[\underline{h},\mathtt{Id}]}{- - - - -}\!\!\!\!> & \mathrm{F}\,\Omega_\mathrm{F}
\end{array}
$$

This characterisation can be obtained as the λ-coiteration schema for the functor $\mathrm{T}X := X + \Omega_\mathrm{F}$ and the distributive law $\lambda : \mathrm{TF} \Rightarrow \mathrm{FT}$ defined as $\lambda_X := [\mathrm{F}\,\mathtt{in}_l, \mathrm{F}\,\mathtt{in}_r \circ \omega_\mathrm{F}] : \mathrm{F}X + \Omega_\mathrm{F} \to \mathrm{F}(X + \Omega_\mathrm{F})$. In loc. cit. the schema that arises as the dual of course-of-value iteration is also given categorically. It can be obtained as an instance of λ-coiteration too. We do not state it here since this requires quite some technicalities. We still would like to mention that for this second example λ-coiteration simplifies the justification of the schema considerably, compared to a proof "by hand".

Finally, we would like to mention that in case the category C has binary products the results from above can be extended to a setting where the distributive law λ is replaced by a natural transformation of the type $\lambda : \mathrm{T}(\mathrm{Id} \times \mathrm{F}) \Rightarrow \mathrm{FT}$. This increases the expressiveness in that one can use both, the F-step that some $x \in X$ appearing inside $t \in \mathrm{T}X$ can do and x itself, for the specification of the behaviour of t. The Theorems 3.8 and 3.9 can be generalised to this new setting with minor modifications: In the first case the construction further requires the existence of binary coproducts in the category C, in the second the unit and multiplication laws for λ have to be adapted as follows: [3]

$$
\begin{array}{ccc}
\mathrm{Id} \times \mathrm{F} & \overset{\pi_2}{\Longrightarrow} & \mathrm{F} \\
{\scriptstyle \eta_{\mathrm{Id} \times \mathrm{F}}} \| & \text{unit } \lambda & \| {\scriptstyle \mathrm{F}\eta} \\
\mathrm{T}(\mathrm{Id} \times \mathrm{F}) & \underset{\lambda}{\Longrightarrow} & \mathrm{FT}
\end{array}
\qquad\qquad
\begin{array}{ccc}
\mathrm{T}(\mathrm{Id} \times \mathrm{F}) & \overset{\lambda}{\Longrightarrow} & \mathrm{FT} \\
{\scriptstyle \mu(\mathrm{Id} \times \mathrm{F})} \nearrow & & \nwarrow {\scriptstyle \mathrm{F}\mu} \\
\mathrm{T}^2(\mathrm{Id} \times \mathrm{F}) & \text{mult. } \lambda & \mathrm{FT}^2 \\
{\scriptstyle \mathrm{T}\langle \mathrm{T}\pi_1, \lambda \rangle} \searrow & & \nearrow {\scriptstyle \lambda\mathrm{T}} \\
& \mathrm{T}(\mathrm{T} \times \mathrm{FT}) &
\end{array}
$$

---

[3] Following a suggestion by Lenisa, Power, and Watanabe [LPW00], one can elegantly show this by moving from the functor F to the cofree copointed functor generated by F, namely $\langle \mathrm{Id} \times \mathrm{F}, \pi_1 \rangle$.

4 Proof by λ-Coinduction

In this Section we will consider generalised proof principles for bisimilarity. We give an example of a statement for which the straightforward relation fails to be a bisimulation. But it satisfies a condition sufficient to conclude that it is contained in some larger bisimulation. Again the condition is expressed via the existence of an FT-coalgebra operation on the relation instead of an F-coalgebra structure and we will call such relations λ-*bisimulations*. The corresponding proof principle allows for simpler bisimilarity proofs involving less complex relations.

4.1 Example: Distributivity of $\oplus$ over $\otimes$

We would like to prove that the addition of streams of real numbers defined in the Example 2.5 distributes over the multiplication considered in Section 3.1:

$$\forall \sigma, \tau, \rho \in I\!\!R^\omega : \quad \sigma \otimes (\tau \oplus \rho) = (\sigma \otimes \tau) \oplus (\sigma \otimes \rho)$$

Since on the final coalgebra bisimilarity is contained in equality, it suffices to prove that both terms denote bisimilar streams. The bisimulation to be found needs to contain

$$(2) \qquad B := \{\langle \sigma \otimes (\tau \oplus \rho), (\sigma \otimes \tau) \oplus (\sigma \otimes \rho)\rangle \mid \sigma, \tau, \rho \in I\!\!R^\omega\} \subseteq I\!\!R^\omega \times I\!\!R^\omega$$

One can easily check that all pairs related by B have equal heads. Furthermore, the bisimulation to be found has to relate the tails of every two streams related by B. A computation yields [4]

$$\texttt{tail}(\sigma \otimes (\tau \oplus \rho)) = \underbrace{([\sigma_0] \otimes (\tau' \oplus \rho'))}_{=:x_1} \oplus \underbrace{(\sigma' \otimes (\tau \oplus \rho))}_{=:x_2}$$

$$\texttt{tail}((\sigma \otimes \tau) \oplus (\sigma \otimes \rho)) = \underbrace{(([\sigma_0] \otimes \tau') \oplus ([\sigma_0] \otimes \rho'))}_{=:y_1} \oplus \underbrace{((\sigma' \otimes \tau) \oplus (\sigma' \otimes \rho))}_{=:y_2}.$$

We obtain sums of streams related by B, namely $\langle x_1, y_1 \rangle$ and $\langle x_2, y_2 \rangle$. Thus, the bisimulation should also contain

$$T(B) := \{\langle x_1 \oplus x_2, y_1 \oplus y_2 \rangle \mid \langle x_1, y_1 \rangle, \langle x_2, y_2 \rangle \in B\} \subseteq I\!\!R^\omega \times I\!\!R^\omega.$$

But then we need to check on the pairs contained in $T(B)$ as well. The continuation of this procedure would lead to the relation

$$\tilde{B} := \bigcup_{i=0}^{\infty} T^i(B).$$

To show that it is a bisimulation one could prove by induction on $i \in I\!\!N$ that for $\langle x, y \rangle \in T^i(B)$ one finds

$$\texttt{head}(x) = \texttt{head}(y) \quad \text{and} \quad \langle \texttt{tail}(x), \texttt{tail}(y) \rangle \in T^{i+1}(B).$$

---

[4] The second one involves commutativity and associativity of $\oplus$. Both can easily be proved using standard bisimulations.

The calculations above constitutes the base case of this induction, the induction step is easy and independent of B.[5] It proves the following principle: Given a relation $B \subseteq \mathbb{R}^\omega \times \mathbb{R}^\omega$ such that for $\langle x, y \rangle \in B$ one has

$$\mathtt{head}(x) = \mathtt{head}(y) \quad \text{and} \quad \langle \mathtt{tail}(x), \mathtt{tail}(y) \rangle \in T(B)$$

then $\tilde{B} := \bigcup_{i=0}^\infty T^i(B)$ is a bisimulation containing B.

With this principle, one can work directly with the relation B as in (2) which only contained the pairs used in the statement itself. There is no need to find a description covering all the tails encountered while walking down the streams, as it is done e.g. by Rutten [Rut00a], who uses all pairs

$$\langle \sum_{i=0}^n (\sigma_i \otimes (\tau_i \oplus \rho_i)), \sum_{i=0}^n ((\sigma_i \otimes \tau_i) \oplus (\sigma_i \otimes \rho_i)) \rangle$$

for $n \in \mathbb{N}, \sigma_i, \tau_i, \rho_i \in \mathbb{R}^\omega$ $(0 \leq i \leq n)$.

4.2 λ-Coinduction

We are now going to capture the situation from above categorically. For a functor T that distributes over F via λ, it corresponds to asking for an FT-coalgebra operation χ on the relation $B \subseteq X \times Y$ which makes the projections homomorphisms up-to-β for suitable T-algebra operations on the F-coalgebras under consideration. We introduce the following notion:

Definition 4.1 [λ-bisimulation] Let F, T : $\mathsf{C} \to \mathsf{C}$ be functors. A span $\mathcal{B} = \langle B, b_1, b_2 \rangle$ is a λ-*bisimulation* between the F-coalgebras $\langle X, \alpha_X \rangle$ and $\langle Y, \alpha_Y \rangle$, if there exist an FT-operation χ on B and T-algebra operations β_X and β_Y on X and Y, such that $\langle X, \beta_X, \alpha_X \rangle$ and $\langle Y, \beta_Y, \alpha_Y \rangle$ are λ-bialgebras and b_1 and b_2 are homomorphisms up-to-β_X and -β_Y respectively:

If we want to make the T-algebra operations β_X and β_Y explicit, we talk about a λ-bisimulation *with respect to β_X and β_Y*. A λ-bisimulation between $\langle X, \alpha \rangle$ and itself will be called a λ-*bisimulation on $\langle X, \alpha \rangle$*.

It turns out that under the assumptions made in Theorem 3.8 (3.9) about C (T and λ), λ-bisimulations only relate bisimilar states. This can be proved by giving a construction that enlarges a λ-bisimulation such that a conventional bisimulation is encountered, similar to what we have done in the example. The

---

[5] The induction step basically states that T is a *respectful function* in the terminology of Sangiorgi [San98] (with a translation of this notion from labelled transition systems to stream systems).

argument does not assume a final F-coalgebra to exist, hence it can be used in any context where behavioural equivalence is defined via bisimulations.

Theorem 4.2 (λ-coinduction) *Let two functors* $T, F : C \to C$ *be given such that* T *distributes over* F *via* λ *and assume that*
(i) the category C *has countable coproducts or*
(ii) T *is taken from a monad* $\langle T, \eta, \mu \rangle$ *such that* λ *satisfies the unit and multiplication law from Theorem 3.9.*
If $\mathcal{B}$ *is a* λ-*bisimulation between two given* F-*coalgebras* $\langle X, \alpha_X \rangle$ *and* $\langle Y, \alpha_Y \rangle$ *then there exists a (conventional) bisimulation* $\tilde{\mathcal{B}}$ *between them with* $\mathcal{B} \preceq \tilde{\mathcal{B}}$.

The proof uses an F-coalgebra $\langle L_B, \alpha_\chi \rangle$ constructed from $\langle B, \chi \rangle$ as in the proof of Theorem 3.8 or 3.9 respectively. In the same way the homomorphisms up-to-β_X and -β_Y give rise to F-coalgebra homomorphisms from $\langle L_B, \alpha_\chi \rangle$ to $\langle X, \alpha_X \rangle$ and $\langle Y, \alpha_Y \rangle$, say $b_1 \mapsto \tilde{b}_1$ and $b_2 \mapsto \tilde{b}_2$. (Note that this direction only needs the assumption that the bialgebras involved are λ-bialgebras, finality is not needed.) This yields a bisimulation $\langle \langle L_B, \alpha_\chi \rangle, \tilde{b}_1, \tilde{b}_2 \rangle$. The order of spans claimed is witnessed by $\mathtt{in}_0$ in (i) and by η_B in (ii).

4.3 The Example revisited

Consider again the example from Section 4.1 where we had $T := \mathrm{Id} \times \mathrm{Id}$ and λ as given in equation (1) on page 9. In this setting the definition of a λ-bisimulation can be spelled out as follows:

Let $\langle X, \langle o_X, s_X \rangle \rangle$ and $\langle Y, \langle o_Y, s_Y \rangle \rangle$ be two stream systems, both coming with binary operations $\oplus_X$ and $\oplus_Y$ on their carrier turning them into λ-bialgebras. The condition for a relation $B \subseteq X \times Y$ to be a λ-bisimulation between $\langle X, \langle o_X, s_X \rangle \rangle$ and $\langle Y, \langle o_Y, s_Y \rangle \rangle$ with respect to $\oplus_X$ and $\oplus_Y$ can easily be seen to be equivalent to the following one: For all $\langle x, y \rangle \in B$ we have that $o_X(x) = o_Y(y)$ and $\langle s_X(x), s_Y(y) \rangle \in T(B)$ where

$$T(B) := \{ \langle x_1 \otimes_X x_2, y_1 \otimes_Y y_2 \rangle \mid \langle x_i, y_i \rangle \in B; i = 1, 2 \}.$$

By taking $\langle X, \langle o_X, s_X \rangle \rangle = \langle Y, \langle o_Y, s_Y \rangle \rangle = \langle I\!R^\omega, \langle \mathtt{head}, \mathtt{tail} \rangle \rangle$ and $\otimes_X = \otimes_Y = \otimes$ we find that the assumption on the relation B in the principle given in Section 4.1 is just to be a λ-bisimulation on the final stream system. Theorem 4.2 tells us that this is enough to conclude that there is a bisimulation $\tilde{B} \subseteq I\!R^\omega \times I\!R^\omega$ with $B \subseteq \tilde{B}$. The span constructed in the proof of Theorem 4.2 corresponds to $\tilde{B}$ from Section 4.1.

5 λ-Coiteration and Operators

In this section we will again consider specifications involving operators in the definition of the dynamics of a state, but this time in a more flexible manner. The definition of the convolution product of formal power series treated in the previous sections could only be handled by the simple functor $T := \mathrm{Id} \times \mathrm{Id}$ because of the very regular occurrence of the auxiliary operation inside the

specification: the tail of $\sigma \otimes \tau$ could always be characterised as the sum of two products of streams. Here we will give a schema that allows for arbitrary terms build of possibly several operators belonging to a certain class. Again we start with a concrete example.

5.1 The Stream of Hamming Numbers

Taking up an example from Dijkstra's [Dij81] (also treated e.g. by Sijtsma [Sij89]), we consider the stream $\mathtt{ham} \in I\!N^\omega$ containing exactly those natural numbers in increasing order whose only prime factors are 2 and 3. These numbers are usually referred to as the Hamming Numbers (admittedly we dropped the prime factor 5 for simplicity). Similar to the previous examples the carrier of the final $S_{I\!N}$-coalgebra, where $S_{I\!N} X := I\!N \times X$, is taken to model infinite streams of natural numbers. We will concentrate on the following specification using the auxiliary operators $\mathtt{merge}$ (binary) and $\mathtt{map}_g$ (unary) for $g : I\!N \to I\!N$:

$$\langle \mathtt{head}, \mathtt{tail} \rangle(\mathtt{ham}) = \langle 1, \mathtt{merge}(\mathtt{map}_{*2}(\mathtt{ham}), \mathtt{map}_{*3}(\mathtt{ham})) \rangle$$

$$\langle \mathtt{head}, \mathtt{tail} \rangle(\mathtt{merge}(\sigma, \tau)) = \begin{cases} \langle \sigma_0, \mathtt{merge}(\sigma', \tau) \rangle & \text{if } \sigma_0 < \tau_0 \\ \langle \sigma_0, \mathtt{merge}(\sigma', \tau') \rangle & \text{if } \sigma_0 = \tau_0 \\ \langle \tau_0, \mathtt{merge}(\sigma, \tau') \rangle & \text{if } \sigma_0 > \tau_0 \end{cases}$$

$$\langle \mathtt{head}, \mathtt{tail} \rangle(\mathtt{map}_g(\sigma)) = \langle g(\sigma_0), \mathtt{map}_g(\sigma') \rangle$$

for $\sigma = \langle \sigma_0 : \sigma' \rangle, \tau = \langle \tau_0 : \tau' \rangle \in I\!N^\omega$ and the functions $*2, *3 : I\!N \to I\!N$ that double and triple their arguments.

Viewing the stream $\mathtt{ham}$ as a function $\mathtt{ham} : 1 \to I\!N^\omega$, where $1 = \{*\}$ is a one element set, one could try to define it by λ-coiteration. The idea is to employ a functor T mapping a set X to the set of terms freely generated by corresponding operator symbols $\underline{\mathtt{merge}}$ and $\underline{\mathtt{map}}_g$ ($g : I\!N \to I\!N$) over X and a distributive law λ that equips these terms with a behaviour according to the definition of $\mathtt{merge}$ and $\mathtt{map}_g$ from above. The stream of Hamming Numbers should then arise as the λ-coiterative arrow induced by

$$\phi := * \mapsto \langle 1, \underline{\mathtt{merge}}(\underline{\mathtt{map}}_{*2}(*), \underline{\mathtt{map}}_{*3}(*)) \rangle : 1 \to S_{I\!N} \mathrm{T} 1.$$

We are going to study this schema in a more general setting in the remainder of this section.

5.2 The λ-Coiteration Schema on Terms

Assume we are given a signature $\langle \Sigma, ar \rangle$, i.e. a set Σ of operator symbols coming with an arity assignment $ar : \Sigma \to I\!N$. Let $\langle \mathrm{T}, \eta, \mu \rangle$ denote the term monad generated by this signature. That is, $\mathrm{T}X$ is the set of terms freely generated by $\langle \Sigma, ar \rangle$ over X (we will sometimes call the set X a set of *variables* in this context), $\mathrm{T}f$ renames all variables inside a term by applying f, η_X converts a variable $x \in X$ into a term (we will usually leave its application implicit), and $\mu_X : \mathrm{T}^2 X \to \mathrm{T}X$ flattens two-level terms.

To use the functor T within λ-coiteration, we need to specify its interaction with F, i.e. a distributive law $\lambda : \mathrm{TF} \Rightarrow \mathrm{FT}$ or, using the generalisation mentioned at the end of Section 3.4, $\lambda : \mathrm{T}(\mathrm{Id} \times \mathrm{F}) \Rightarrow \mathrm{FT}$. We will concentrate on the latter type, since many examples for this approach require the greater expressiveness it offers.

Assume we are given such a distributive law λ. Let $\langle \Omega_{\mathrm{F}}, \omega_{\mathrm{F}} \rangle$ be a final F-coalgebra and let $[\![.]\!]$ denote the coiterative morphism from $\mathrm{T}_\lambda \langle \Omega_{\mathrm{F}}, \omega_{\mathrm{F}} \rangle$ to $\langle \Omega_{\mathrm{F}}, \omega_{\mathrm{F}} \rangle$. By applying Theorem 3.8 we get that for every operation $\phi : X \to \mathrm{FT}X$ there is a unique arrow $f : X \to \Omega_{\mathrm{F}}$ satisfying $\omega_{\mathrm{F}} \circ f = \mathrm{F}f|^{[\![.]\!]} \circ \phi$. In the above example this means that `ham` is a unique arrow satisfying `head(ham)` $= 1$ and `tail(ham)` $= [\![\underline{\mathtt{merge}}(\underline{\mathtt{map}}_{*2}(*), \underline{\mathtt{map}}_{*3}(*))]\!]$. For this to be a solution for the above specification we need that $[\![.]\!]$ evaluates terms according to the (semantic) operators `merge` and $\mathtt{map}_g$. In particular, $[\![.]\!]$ should be compositional, i.e. for all $\sigma \in \Omega_{\mathrm{F}}$, $t_i \in \mathrm{T}\Omega_{\mathrm{F}}$ we ask

$$[\![\sigma]\!] = \sigma \quad \text{and} \quad [\![op(t_1, \ldots, t_n)]\!] = [\![op([\![t_1]\!], \ldots, [\![t_n]\!])]\!].$$

It can be shown that we obtain a compositional T-algebra operation in case λ is compositional itself, which means it satisfies the modified unit and multiplication laws shown in Section 3.5. This is equivalent to saying that λ is constructed inductively from natural transformations

$$(3) \qquad \qquad \rho^{op} : (\mathrm{Id} \times \mathrm{F})^n \Rightarrow \mathrm{FT},$$

for each operator symbol $op \in \Sigma$, $ar(op) = n$. Then $[\![.]\!]$ can be uniquely characterised as being compositional and satisfying

$$\omega_{\mathrm{F}}([\![op(\sigma_1, \ldots, \sigma_n)]\!]) = (\mathrm{F}[\![.]\!])(\rho^{op}(\langle \sigma_1, \omega_{\mathrm{F}}(\sigma_1) \rangle, \ldots, \langle \sigma_n, \omega_{\mathrm{F}}(\sigma_n) \rangle)).$$

In our concrete example it is straightforward to come up with transformations $\rho^{\underline{\mathtt{merge}}}$ and $\rho^{\underline{\mathtt{map}}_g}$ such that the evaluation of terms using `merge` and $\mathtt{map}_g$ satisfies this characterisation. Using this we can easily compute that the λ-coiterative arrow above is the unique one satisfying the original specification.

Generally, the approach yields unique solutions for specifications involving a set of auxiliary operators that can be captured by natural transformations as in (3). This boils down to saying that for every operator op the first F-step of $op(\sigma_1, \ldots, \sigma_n)$ can be determined by only looking at the direct observations for $\sigma_i \in \Omega_{\mathrm{F}}$ and the successor states contained are among the σ_i, their immediate successors, or any state that can be reached from those by applying the operators themselves (these restrictions are due to the naturality condition in (3)). Turi and Plotkin [TP97] show that this type of operator specification is closely related to transition rules in GSOS format [BIM95].

The fact that $[\![.]\!]$ evaluates terms by applying corresponding semantic operators can also be used to simplify the definition of λ-bisimulations on the final F-coalgebra: Let $\tilde{B}$ denote the congruence closure of a given relation $B \subseteq \Omega_{\mathrm{F}} \times \Omega_{\mathrm{F}}$ under all operators, i.e. the smallest relation containing B such that for all operators op we have that $\langle op(\sigma_1, \ldots, \sigma_n), op(\tau_1, \ldots, \tau_n) \rangle \in \tilde{B}$ if $\langle \sigma_i, \tau_i \rangle \in \tilde{B}$ for $1 \leq i \leq n$. We get that B is a λ-bisimulation on $\langle \Omega_{\mathrm{F}}, \omega_{\mathrm{F}} \rangle$ if and

only if there is an operation $\chi : B \to F\tilde{B}$ making the diagram below commute (note that the projections π_i in the upper and lower part of the diagram are taken w.r.t. different relations):

$$
\begin{array}{ccccc}
\Omega_F & \xleftarrow{\ \pi_1\ } & B & \xrightarrow{\ \pi_2\ } & \Omega_F \\
{\scriptstyle\omega_F}\Big\downarrow & & \Big\downarrow{\scriptstyle\exists\chi} & & \Big\downarrow{\scriptstyle\omega_F} \\
F\Omega_F & \xleftarrow[F\pi_1]{} & F\tilde{B} & \xrightarrow[F\pi_2]{} & F\Omega_F
\end{array}
$$

A relation satisfying this condition is sometimes called a *bisimulation up-to-context* [San98]. Our treatment abstractly justifies that all states related by a bisimulation up-to-context are bisimilar, if the contexts are built from operators of the type mentioned above. As an example, the operators on formal power series Rutten considers for his stream calculus [Rut00a], of which we have seen the sum and convolution product already, all fit into this framework. The resulting proof principle up-to-context yields simpler proofs for many of the equations involving these operators he states in Theorem 4.1 in loc. cit.

6 Related and Future Work

As mentioned in the introduction, we developed our framework as an advancement of Lenisa's schemata [Len99a]. She defines the definition principle of *coiteration up-to-$\mathcal{T}$* for a pointed functor $\mathcal{T} = \langle T, \eta \rangle$ – i.e. a functor $T : C \to C$ with a natural transformation $\eta : \mathrm{Id} \Rightarrow T$ – and a proof principle based on the notion of *bisimulation up-to-$\mathcal{T}$*, which additionally assumes that the pointed functor distributes over the behaviour functor F via some λ and that it is actually taken from a monad $\langle T, \eta, \mu \rangle$. In this case one can argue that there is a close relationship between our λ-coiterative arrows and her arrows coiterative up-to-$\mathcal{T}$.

Our definition principle improves Lenisa's in that it comes as a direct characterisation of the arrows to be defined instead of a construction. For the proof principles, λ-coiteration generalises the one based on bisimulations up-to-$\mathcal{T}$ in that it is no longer limited to bialgebras of the shape $\langle T X, \mu_X, \alpha \rangle$. These are central in Lenisa's setting since the coalgebras involved appear within coiteration up-to-$\mathcal{T}$. The main impact of this generalisation is presumably the fact that it allows one to work directly on the final coalgebra, as we have demonstrated in our example. This is particularly helpful since it obviates the need to reason up-to-bisimilarity, which arises in many examples but is possible with neither of the two principles yet. Generally, in relation to Lenisa's theory we have emphasised and clarified the role of the distributive law, paying less attention to the monad structure.

Lenisa has used bisimulations up-to-$\mathcal{T}$ to derive a sound principle for proving equivalences induced by arrows coiterative up-to-$\mathcal{T}$. In our case of a λ-coiterative arrow f for $\phi : X \to FTX$ her statement simplifies to the immediate observation that all FT-bisimilar states in $\langle X, \phi \rangle$ are equated by f. There

are clearly stronger statements to be found about the equivalence induced by λ-coiterative arrows, which we leave for future work.

Another paper that our work on bisimilarity proofs is related to is that by Sangiorgi [San98]. It is about relational bisimulations for labelled transition systems in Set, but a reformulation in terms of spans between arbitrary F-coalgebras in an abstract category C should be possible (certainly requiring extra assumptions to be made for particular aspects of the theory to carry over). We expect that our notion of λ-bisimulation can be made to fit into such a framework (as argued in [Bar00]). This may open our technique for further examples covered by Sangiorgi's method, like *bisimulation up-to-bisimilarity* (see e.g. [Mil89]).

Our use of a distributive law λ and λ-bialgebras follows that of Turi and Plotkin [TP97]. Starting from a signature functor Σ and a behavioural functor F, they describe program syntax using the term monad $\langle T, \eta, \mu \rangle$ generated by Σ and global program behaviour by means of the behavioural comonad $\langle D, \varepsilon, \delta \rangle$ generated by F. The specification of the semantics takes the shape of a distributive law λ of the monad over the comonad and its models are the λ-bialgebras.

In the example of corecursive definitions using operators from Section 5 our theory gets related to their setting, in that the functor T is taken from the term monad as well and that the class of well behaved operators here coincides with the main class considered by Turi and Plotkin. We add to their approach a separate treatment of specifications of the type $\phi : X \rightarrow \mathrm{FT}X$, which are similar to what is sometimes called *guarded recursive definitions*. One of the advantages of treating the operators and solutions of the recursive specifications separately is that we obtain a notion of bisimulation up-to-context to reason about these solutions.

The setting of Turi and Plotkin is more general in the sense that they consider distributive laws of the term monad over the behavioural comonad $\langle D, \varepsilon, \delta \rangle$ generated by F instead of F alone. This allows them to also treat a second major class of operators corresponding to *tree rules*. We cannot handle this class in our setting, which is in agreement with the fact that its operators may not safely be used for bisimulation up-to-context. Sangiorgi has already pointed out that this condition is stronger than only asking that bisimulation be a congruence for the operators under consideration (see the example at the end of Section 2 in [San98]). The latter condition is the decisive one for Turi and Plotkin's approach.

We left for future work the quest for further interesting instances of our framework. Since we found sufficient conditions for our schema to work that do not assume the functor T to come as a pointed functor or monad, it would be particularly nice to come up with examples exploiting this generality. In all the examples we have so far this structure can be added immediately or at least after a straightforward reformulation of the problem.

Another interesting aspect to study would be the interplay of λ-coiterative

arrows and invariant properties.

Acknowledgements

References

[AM89] Peter Aczel and Nax Mendler. A final coalgebra theorem. In D.H. Pitt, D.E. Rydeheard, P. Dybjer, A.M. Pitts, and A. Poigné, editors, *Proceedings 3rd Conf. on Category Theory and Computer Science, CTCS'89, Manchester, UK, 5–8 Sept 1989*, volume 389 of *Lecture Notes in Computer Science*, pages 357–365. Springer-Verlag, Berlin, 1989.

[Bar00] F. Bartels. Generalised coinduction. Technical Report SEN-R0043, CWI - Centrum voor Wiskunde en Informatica, December 2000.

[BIM95] Bard Bloom, Sorin Istrail, and Albert R. Meyer. Bisimulation can't be traced. *Journal of the ACM*, 42(1):232–268, January 1995.

[Dij81] E.W. Dijkstra. Hamming's exercise in sasl. Personal Note EWD792, see `http://www.cs.utexas.edu/users/EWD/`, 1981.

[JR96] Bart Jacobs and Jan Rutten. A tutorial on (co)algebras and (co)induction. *Bulletin of the EATCS*, 62:222–259, 1996.

[JR99] B. Jacobs and J. Rutten, editors. *Proc. Coalgebraic Methods in Computer Science (CMCS 1999)*, volume 19 of *Electronic Notes in Theoretical Computer Science*, 1999.

[Len99a] M. Lenisa. From set-theoretic coinduction to coalgebraic coinduction: Some results, some problems. Unpublished extension of [Len99b] (available through the author's homepage at `http://www.dimi.uniud.it/~lenisa/`), December 1999.

[Len99b] M. Lenisa. From set-theoretic coinduction to coalgebraic coinduction: some results, some problems. In Jacobs and Rutten [JR99], pages 1–21.

[LPW00] M. Lenisa, J. Power, and H. Watanabe. Distributivity for endofunctors, pointed and co-pointed endofunctors, monads and comonads. In H. Reichel, editor, *Proc. Coalgebraic Methods in Computer Science (CMCS 2000)*, volume 33 of *Electronic Notes in Theoretical Computer Science*, pages 233–263, 2000.

[Mil89] R. Milner. *Communication and Concurrency*. International Series in Computer Science. Prentice Hall, 1989.

[PW99] J. Power and H. Watanabe. Distributivity for a monad and a comonad. In Jacobs and Rutten [JR99], pages 119–132.

[Rut00a] J.J.M.M. Rutten. Behavioural differential equations: a coinductive calculus of streams, automata, and power series. Technical Report SEN-R0023, CWI - Centrum voor Wiskunde en Informatica, 2000.

[Rut00b] J.J.M.M. Rutten. Universal coalgebra: A theory of systems. *Theoretical Computer Science*, 249(1):3–80, October 2000.

[San98] D. Sangiorgi. On the bisimulation proof method. *Journal of Mathematical Structures in Computer Science*, 8:447–479, 1998.

[Sij89] Ben A. Sijtsma. On the Productivity of Recursive List Definitions. *ACM Transactions on Programming Languages and Systems*, 11(4):633–649, October 1989.

[TP97] D. Turi and G.D. Plotkin. Towards a mathematical operational semantics. In *Proc. 12*th *LICS Conf.*, pages 280–291. IEEE, Computer Society Press, 1997.

[UV99] Tarmo Uustalu and Varmo Vene. Primitive (co)recursion and course-of-value (co)iteration, categorically. *INFORMATICA (IMI, Lithuania)*, 10(1):5–26, 1999.