INFORMATICA, 1999 Vol. 10, No. 1, 1–0

Primitive (Co)Recursion and Course-of-Value (Co)Iteration, Categorically

Tarmo Uustalu

Dept. of Teleinformatics, Royal Inst. of Technology, Electrum 204, SE-164 40 Kista, Sweden
E-mail: tarmo@it.kth.se

Varmo Vene

Inst. of Computer Science, Univ. of Tartu, J. Liivi 2, EE-2484 Tartu, Estonia
E-mail: varmo@cs.ut.ee

Abstract In the mainstream categorical approach to typed (total) functional programming, datatypes are modelled as initial algebras and codatatypes as terminal coalgebras. The basic function definition schemes of iteration and coiteration are modelled by constructions known as catamorphisms and anamorphisms. Primitive recursion has been captured by a construction called paramorphisms. We draw attention to the dual construction of apomorphisms, and show on examples that primitive corecursion is a useful function definition scheme. We also put forward and study two novel constructions, viz., histomorphisms and futumorphisms, that capture the powerful schemes of course-of-value iteration and its dual, respectively, and argue that even these are helpful.

Key words: typed (total) functional programming, category theory, program calculation, (co)datatypes, forms of (co)recursion

1. Introduction This paper is about a well-known categorical approach to typed (total) functional programming popular in the program calculation community. In this approach, datatypes and codatatypes are modelled as initial algebras and terminal coalgebras, and iteration and coiteration are modelled by constructions known as catamorphisms and anamorphisms. Primitive recursion, which universally recognized as an important generalization of iteration, is nicely captured by Meertens' (1992) paramorphisms. With this paper, we aim to draw attention to the dual construction of apomorphisms, described in (Vos, 95) and (Vene and Uustalu, 1998), which models an undeservedly little appreciated scheme that we like to call primitive corecursion. We also put forward and study two novel constructions, viz., histomorphisms and futumorphisms, that capture course-of-value iteration and its dual, respectively, and argue that even these schemes are helpful for a declaratively thinking programmer and program reasoner who loves languages of programming and program reasoning where programs and proofs of properties of programs are easy to write and read.

The tradition of categorical functional programming that we follow in this paper started off with the Bird-Meertens formalism (Bird, 1987), which is a theory of the datatype of lists. Malcolm (1990) and Fokkinga (1992), inspired by (Hagino, 1987), generalized the approach for arbitrary datatypes and codatatypes.

(Geuvers, 1992) contains a thorough category-theoretic analysis of primitive recursion versus iteration and a demonstration that this readily dualizes into an analysis of primitive corecursion versus coiteration. In general, however, it appears that primitive corecursion has largely been overlooked in the theoretical literature, e.g. (Fokkinga, 1992) ignores it. Apart from (Vos, 95) and (Vene and Uustalu, 1998), the sole discussion on primitive corecursion in a programming context that we know about is the laconic report in (Vesely, 1997) on a recent not very clean extension to the categorical functional language Charity in which it is possible to define functions by primitive recursion and primitive corecursion. The most probable reason for this situation is the young age of programming with codatatypes. We are not aware of any work whatsoever on categorical constructions modelling course-of-value (co)iteration.

This paper is an intermediate report of an ongoing research project on program construction for typed functional languages in which all programs are guaranteed to terminate (only total functions can be programmed). For a recent defence of the merits of typed total functional programming, see (Turner, 1995). Part of the material of the present paper originates from (Vene and Uustalu, 1998). In another earlier paper, (Uustalu and Vene, 1997), we discussed programming and program construction with (co)datatypes in a different setting of intuitionistic natural deduction.

We shall proceed as follows. In section 2, we present the setting and notation. In section 3, we give a short introduction to the standard categorical approach to (co)datatypes and (co)iteration. In section 4, we introduce a categorical construction for primitive recursion and prove several laws about it. Afterwards, we dualize this development, introducing a categorical primitive corecursor and stating the laws it obeys, and show the usefulness of primitive corecursion on several examples. In section 5, we introduce a novel construction for course-of-value iteration, prove the crucial laws about it, and hint at programming examples where course-of-value iteration is useful. Again, we also dualize the whole development, arriving at a categorical treatment of the dual of course-of-value iteration. In section 6, we indicate some possible directions for further work and conclude.

Proofs in this paper are carried out in the structured calculational proof style (Grundy, 1996).

2. Preliminaries Throughout the paper $\mathcal{C}$ is the default category, which we require to be *distributive*, i.e., it must have finite products $(\times, 1)$ and co-products $(+, 0)$, and the products must distribute over the coproducts. The typical example of a distributive category is $\mathbf{Sets}$ — the category of sets and total functions.

We make use the following quite standard notation. Given two objects A, B, we write $\mathtt{fst} : A \times B \to A$ and $\mathtt{snd} : A \times B \to B$ for the left and right projections for the product $A \times B$. For $g : C \to A$ and $h : C \to B$, $\langle\, g, h \,\rangle$ denotes the unique morphism $f : C \to A \times B$, such that $\mathtt{fst} \circ f = g$ and $\mathtt{snd} \circ f = h$ (pairing). The notation $g \times h$ is a shorthand for $\langle\, g \circ \mathtt{fst}, h \circ \mathtt{snd} \,\rangle$. The left and right injections for the coproduct $A + B$ are denoted by $\mathtt{inl} : A \to A + B$ and $\mathtt{inr} : B \to A + B$. For $g : A \to C$ and $h : B \to C$, $[\, g, h \,]$ denotes the unique morphism $f : A + B \to C$, such that $f \circ \mathtt{inl} = g$ and $f \circ \mathtt{inr} = h$ (case analysis). The notation $g + h$ abbreviates $[\, \mathtt{inl} \circ g, \mathtt{inr} \circ h \,]$. Also, ! and ¡ denote the unique morphisms $f : C \to 1$ and $f' : 0 \to C$. The inverse of the canonical map $[\, \mathtt{inl} \times \mathtt{id}_C, \mathtt{inr} \times \mathtt{id}_C \,] : (A \times C) + (B \times C) \to (A + B) \times C$ is denoted by $\mathtt{distr} : (A + B) \times C \to (A \times C) + (B \times C)$, and the inverse of the canonical map $[\, \mathtt{id}_A \times \mathtt{inl}, \mathtt{id}_A \times \mathtt{inr} \,] : (A \times B) + (A \times C) \to A \times (B + C)$ is denoted by $\mathtt{distl} : A \times (B + C) \to (A \times B) + (A \times C)$. Finally, given a predicate $p : A \to 1 + 1$, the guard $p? : A \to A + A$ is defined as $(\mathtt{snd} + \mathtt{snd}) \circ \mathtt{distr} \circ \langle\, p, \mathtt{id}_A \,\rangle$.

3. (Co)datatypes and (co)iteration Categorically, datatypes (of natural numbers, lists, etc.) are traditionally modelled by initial algebras and codatatypes (of conatural numbers, colists, streams, etc.) by terminal coalgebras. The function definition schemes of iteration and coiteration are modelled by constructions termed catamorphisms and anamorphisms. The theory presented in this section has become a bit of folklore; a good reference source is (Fokkinga, 1992).

Initial algebras and catamorphisms Let F be a functor from $\mathcal{C}$ to $\mathcal{C}$. An *algebra* with the *signature* F (F-algebra, for short) is a pair (A, φ), where A (called the *carrier*) is an object and $\varphi : F\,A \to A$ (called the *structure*) is a morphism in $\mathcal{C}$. For any two F-algebras (A, φ) and (C, ψ), a morphism $f : A \to C$ is said to be a *homomorphism (between F-algebras)* from (A, φ) to (C, ψ) if $f \circ \varphi = \psi \circ F\,f$.

It is easy to verify that the morphism composition of two homomorphisms is also a homomorphism. Moreover, for any F-algebra (A, φ), the identity morphism $\mathtt{id} : A \to A$ is a homomorphism from (A, φ) to (A, φ). It follows that the F-algebras and the homomorphisms between them form a category. The category $\mathcal{A}lg(F)$ is the category whose objects are the F-algebras and morphisms are the homomorphisms between them. Composition and identities are inherited from $\mathcal{C}$.

A F-algebra is said to be an *initial* F-algebra if it is initial (as an object) in the category $\mathcal{A}lg(F)$. The initial F-algebra may or may not exist. It is guaranteed to exist if F is ω-cocontinuous (i.e., it preserves the colimits of ω-chains). All polynomial functors (i.e., functors built up from products, sums, the identity functor, and constant functors) are ω-cocontinuous and, hence, the initial algebras for them exist.

Given a functor F, the existence of the initial F-algebra $(\mu F, \mathtt{in}_F)$ means that, for any F-algebra (C, φ), there exists a unique F-homomorphism of algebras from $(\mu F, \mathtt{in}_F)$ to (C, φ). Following Fokkinga (1992), we denote this homomorphism by $\llparenthesis\, \varphi \,\rrparenthesis_F$, so $\llparenthesis\, \varphi \,\rrparenthesis_F : \mu F \to C$ is characterized by the universal property

$$f \circ \mathtt{in}_F = \varphi \circ F\, f \quad \Leftrightarrow \quad f = \llparenthesis\, \varphi \,\rrparenthesis_F \qquad\qquad \text{cata-C{\small HARN}}$$

The type information is summarized in the following diagram:

$$\begin{tikzcd}
F\,\mu F \arrow[r, "\mathtt{in}_F"] \arrow[d, "F\,f"'] & \mu F \arrow[d, "f"] \\
F\,C \arrow[r, "\varphi"] & C
\end{tikzcd}$$

Morphisms of the form $\llparenthesis\, \varphi \,\rrparenthesis$ are called *catamorphisms* (derived from the Greek preposition $\kappa\alpha\tau\alpha$ meaning 'downwards'); the construction $\llparenthesis\, \cdot \,\rrparenthesis$ is an iterator.

EXAMPLE 1. The datatype of natural numbers can be represented as the initial algebra $(\mu N, \mathtt{in}_N)$ of the functor N defined by $N\,X = 1 + X$ and $N\,f = \mathtt{id}_1 + f$. Write Nat for μN. The function $\mathit{zero} : 1 \to \mathit{Nat}$ equals $\mathtt{in}_N \circ \mathtt{inl}$, the function $\mathit{succ} : \mathit{Nat} : \mathit{Nat} \to \mathit{Nat}$ equals $\mathtt{in}_N \circ \mathtt{inr}$. Given any two functions $c : 1 \to C$ and $h : C \to C$, the catamorphism $f = \llparenthesis\, [\, c, h\,] \,\rrparenthesis_N : \mathit{Nat} \to C$ is the unique solution of the equation system

$$\begin{aligned} & f \circ \mathit{zero} = c \\ \wedge\ & f \circ \mathit{succ} = h \circ f. \end{aligned}$$

Given a function $n : 1 \to \mathit{Nat}$, the function $\mathit{add}(n) : \mathit{Nat} \to \mathit{Nat}$ (which adds n to its argument), for instance, can be defined as the catamorphism $\llparenthesis\, [\, n, \mathit{succ}\,] \,\rrparenthesis_N$.

EXAMPLE 2. The datatype of lists over a given set A can be represented as the initial algebra $(\mu L_A, \mathtt{in}_{L_A})$ of the functor L_A defined by $L_A\,X = 1 + (A \times X)$ and $L_A\,f = \mathtt{id}_1 + (\mathtt{id}_A \times f)$. Denote μL_A by List_A. The function

$\mathit{nil} : 1 \to \mathit{List}_A$ equals $\mathtt{in}_{L_A} \circ \mathtt{inl}$, the function $\mathit{cons} : A \times \mathit{List}_A \to \mathit{List}_A$ equals $\mathtt{in}_{L_A} \circ \mathtt{inr}$. Given any two functions $c : 1 \to C$ and $h : A \times C \to C$, the catamorphism $f = \llparenthesis\, [\, c, h\,]\, \rrparenthesis_{L_A} : \mathit{List}_A \to C$ is the unique solution of the equation system

$$\begin{array}{rl} & f \circ \mathit{nil} = c \\ \wedge & f \circ \mathit{cons} = h \circ (\mathtt{id}_A \times f), \end{array}$$

i.e., $\mathit{foldr}(c, h)$ from functional programming. Given a function $h : A \to B$, the function $\mathit{map}(h) : \mathit{List}_A \to \mathit{List}_B$ (which applies h to every element of the argument), for instance, can be defined as the catamorphism

$$\llparenthesis\, [\, \mathit{nil}, \mathit{cons} \circ (h \times \mathtt{id}_{\mathit{List}_A})\,]\, \rrparenthesis_{L_A} .$$

Catamorphisms obey several nice laws, of which the *cancellation* law, the *reflection* law (also known as the identity law) and the *fusion* (or, promotion) law are especially important for program calculation (the cancellation law describing a certain program execution step):

$$\begin{array}{lr} \llparenthesis\, \varphi\, \rrparenthesis_F \circ \mathtt{in}_F = \varphi \circ F\, \llparenthesis\, \varphi\, \rrparenthesis_F & \text{cata-C}\textsc{ancel} \\ \mathtt{id}_{\mu F} = \llparenthesis\, \mathtt{in}_F\, \rrparenthesis_F & \text{cata-R}\textsc{efl} \\ f \circ \varphi = \psi \circ F\, f \;\Rightarrow\; f \circ \llparenthesis\, \varphi\, \rrparenthesis_F = \llparenthesis\, \psi\, \rrparenthesis_F & \text{cata-F}\textsc{usion} \end{array}$$

The reflection and fusion laws are proved as follows:

$$\left[\begin{array}{ll} & \mathtt{id}_{\mu F} \\ = & \quad \text{– cata-C}\textsc{harn}\text{ –} \\ & \left[\begin{array}{ll} & \mathtt{in}_F \\ = & \quad \text{– } F \text{ functor –} \\ & \mathtt{in}_F \circ F\, \mathtt{id}_{\mu F} \end{array}\right. \\ & \llparenthesis\, \mathtt{in}_F\, \rrparenthesis_F \end{array}\right.$$

$$\left[\begin{array}{ll} \multicolumn{2}{l}{\rhd\ f \circ \varphi = \psi \circ F\, f} \\ \hline & f \circ \llparenthesis\, \varphi\, \rrparenthesis_F \\ = & \quad \text{– cata-C}\textsc{harn}\text{ –} \\ & \left[\begin{array}{ll} & f \circ \llparenthesis\, \varphi\, \rrparenthesis_F \circ \mathtt{in}_F \\ = & \quad \text{– cata-C}\textsc{harn}\text{ –} \\ & f \circ \varphi \circ F\, \llparenthesis\, \varphi\, \rrparenthesis_F \\ = & \quad \text{– } \lhd \text{ –} \\ & \psi \circ F\, f \circ F\, \llparenthesis\, \varphi\, \rrparenthesis_F \\ = & \quad \text{– } F \text{ functor –} \\ & \psi \circ F\, (f \circ \llparenthesis\, \varphi\, \rrparenthesis_F) \end{array}\right. \\ & \llparenthesis\, \psi\, \rrparenthesis_F \end{array}\right.$$

The important simple result that $\mathtt{in}_F : F\, \mu F \to \mu F$ is an *isomorphism* with the inverse $\llparenthesis\, F\, \mathtt{in}_F\, \rrparenthesis_F : \mu F \to F\, \mu F$ was first recorded by Lambek (1968).

$$\mathtt{in}_F \circ \llparenthesis\, F\, \mathtt{in}_F\, \rrparenthesis_F = \mathtt{id}_{\mu F} \;\wedge\; \llparenthesis\, F\, \mathtt{in}_F\, \rrparenthesis_F \circ \mathtt{in}_F = \mathtt{id}_{F\, \mu F} \qquad \text{L}\textsc{emma}\text{-1}$$

It is proved by the following two calculations:

$$\left[\begin{array}{ll} & \mathtt{in}_F \circ (\!|\, F\,\mathtt{in}_F \,|\!)_F \\ = & \quad -\ \text{cata-}\textsc{Fusion}\ - \\ & (\!|\, \mathtt{in}_F \,|\!)_F \\ = & \quad -\ \text{cata-}\textsc{Refl}\ - \\ & \mathtt{id}_{\mu F} \end{array}\right.
\qquad
\left[\begin{array}{ll} & (\!|\, F\,\mathtt{in}_F \,|\!)_F \circ \mathtt{in}_F \\ = & \quad -\ \text{cata-}\textsc{Charn}\ - \\ & F\,\mathtt{in}_F \circ F\,(\!|\, F\,\mathtt{in}_F \,|\!)_F) \\ = & \quad -\ F\ \text{functor}\ - \\ & F\,(\mathtt{in}_F \circ (\!|\, F\,\mathtt{in}_F \,|\!)_F) \\ = & \quad -\ \text{above}\ - \\ & F\,\mathtt{id}_{\mu F} \\ = & \quad -\ F\ \text{functor}\ - \\ & \mathtt{id}_{F\,\mu F} \end{array}\right.$$

Define

$$\mathtt{in}_F^{-1} = (\!|\, F\,\mathtt{in}_F \,|\!)_F \qquad\qquad \text{in-inv-}\textsc{Def}$$

Lemma 1 gives us the following characterization for $\mathtt{in}_F^{-1}$:

$$\mathtt{in}_F \circ \mathtt{in}_F^{-1} = \mathtt{id}_{\mu F} \ \wedge\ \mathtt{in}_F^{-1} \circ \mathtt{in}_F = \mathtt{id}_{F\,\mu F} \qquad\qquad \text{in-inv-}\textsc{Charn}$$

EXAMPLE 3. Using $\mathtt{in}_F^{-1}$, the predecessor function $\mathit{pred} : \mathit{Nat} \to \mathit{Nat}$ can be defined as $[\,\mathit{zero}, \mathtt{id}_{\mathit{Nat}}\,] \circ \mathtt{in}_N^{-1}$.

Terminal coalgebras and anamorphisms We now dualize the material about initial algebras and catamorphisms.

Let F be a functor from $\mathcal{C}$ to $\mathcal{C}$. A *coalgebra* with the *signature* F (F-coalgebra, for short) is a pair (A, φ), where A (called the *carrier*) is an object and $\varphi : A \to F\,A$ (called the *structure*) is a morphism in $\mathcal{C}$. For any two F-coalgebras (C, ψ) and (A, φ), a morphism $f : C \to A$ is said to be a *homomorphism (between F-coalgebras)* from (C, ψ) to (A, φ) if $\varphi \circ f = F\,f \circ \psi$.

The F-coalgebras and the homomorphisms between them form a category. The category $\mathcal{C}o\mathcal{A}lg(F)$ is the category whose objects are the F-coalgebras and morphisms are the homomorphisms between them. Composition and identities are inherited from $\mathcal{C}$.

A F-coalgebra is said to be a *terminal* F-coalgebra if it is terminal (as an object) in the category $\mathcal{C}o\mathcal{A}lg(F)$. The terminal F-coalgebra may or may not exist. It is guaranteed to exist if F is ω-continuous (i.e., it preserves the limits of ω-chains). All polynomial functors (i.e., functors built up from products, sums, the identity functor, and constant functors) are ω-continuous and, hence, the terminal coalgebras for them exist.

The existence of the terminal F-coalgebra $(\nu F, \mathtt{out}_F)$ means that for any F-coalgebra (C, φ), there exists a unique F-homomorphism of coalgebras from

(C, φ) to $(\nu F, \mathtt{out}_F)$. This homomorphism is usually denoted by $[\!(\, \varphi \,)\!]_F$, so $[\!(\, \varphi \,)\!]_F : C \to \nu F$ is characterized by the universal property

$$\mathtt{out}_F \circ f = F\, f \circ \varphi \;\Leftrightarrow\; f = [\!(\, \varphi \,)\!]_F \qquad \text{ana-CHARN}$$

The type information is summarized in the following diagram:

$$\begin{tikzcd}
C \arrow[r, "\varphi"] \arrow[d, "f"'] & F\, C \arrow[d, "F\, f"] \\
\nu F \arrow[r, "\mathtt{out}_F"] & F\, \nu F
\end{tikzcd}$$

Morphisms of the form $[\!(\, \varphi \,)\!]$ are called *anamorphisms* (derived from the Greek preposition $\alpha\nu\alpha$ meaning 'upwards'; the name is due to Meijer), and the construction $[\!(\, \cdot \,)\!]$ is a coiterator. Like catamorphisms, anamorphisms enjoy various properties including the following cancellation, reflection and fusion laws:

$$\begin{array}{lr}
\mathtt{out}_F \circ [\!(\, \varphi \,)\!]_F = F\, [\!(\, \varphi \,)\!]_F \circ \varphi & \text{ana-CANCEL} \\
\mathtt{id}_{\nu F} = [\!(\, \mathtt{out}_F \,)\!]_F & \text{ana-REFL} \\
\psi \circ f = F\, f \circ \varphi \;\Rightarrow\; [\!(\, \psi \,)\!]_F \circ f = [\!(\, \varphi \,)\!]_F & \text{ana-FUSION}
\end{array}$$

Also, $\mathtt{out}_F : \nu F \to F\, \nu F$ is an isomorphism with the inverse $[\!(\, F\, \mathtt{out}_F \,)\!]_F : F\, \nu F \to \nu F$.

$$\begin{array}{lr}
\mathtt{out}_F^{-1} = [\!(\, F\, \mathtt{out}_F \,)\!]_F & \text{out-inv-DEF} \\
\mathtt{out}_F^{-1} \circ \mathtt{out}_F = \mathtt{id}_{\nu F} \wedge \mathtt{out}_F \circ \mathtt{out}_F^{-1} = \mathtt{id}_{F\, \nu F} & \text{out-inv-CHARN}
\end{array}$$

EXAMPLE 4. The codatatype of streams over a given set A is nicely represented by the terminal coalgebra $(\nu S_A, \mathtt{out}_{S_A})$ of the functor S_A defined by $S_A\, X = A \times X$ and $S_A\, f = \mathtt{id}_A \times f$. Write Stream_A for νS_A. The functions $\mathit{head} : \mathit{Stream}_A \to A$ and $\mathit{tail} : \mathit{Stream}_A \to \mathit{Stream}_A$ equal $\mathtt{fst} \circ \mathtt{out}_{S_A}$ and $\mathtt{snd} \circ \mathtt{out}_{S_A}$, respectively. Given any two functions $c : C \to A$ and $h : C \to C$, the anamorphism $[\!(\, \langle c, h \rangle \,)\!]_{S_A}$ is the unique solution $f : C \to \mathit{Stream}_A$ of the equation system

$$\begin{array}{c}
\mathit{head} \circ f = c \\
\wedge\; \mathit{tail} \circ f = f \circ h.
\end{array}$$

The function $\mathit{nats} : \mathit{Nat} \to \mathit{Stream}_{\mathit{Nat}}$, which returns the stream of all natural numbers starting with the natural number given as the argument, is the unique solution of the equation system

$$\begin{aligned} & \mathit{head} \circ \mathit{nats} = \mathtt{id}_{\mathit{Nat}} \\ \wedge\ & \mathit{tail} \circ \mathit{nats} = \mathit{nats} \circ \mathit{succ}, \end{aligned}$$

and is thus definable as the anamorphism $[\![\langle \mathtt{id}_{\mathit{Nat}}, \mathit{succ} \rangle]\!]$.

The function $\mathit{zip} : \mathit{Stream}_A \times \mathit{Stream}_A \to \mathit{Stream}_{A \times A}$ that zips the argument streams together is characterized as follows:

$$\begin{aligned} & \mathit{head} \circ \mathit{zip} = (\mathtt{fst} \times \mathtt{fst}) \circ (\mathtt{out}_{S_A} \times \mathtt{out}_{S_A}) \\ \wedge\ & \mathit{tail} \circ \mathit{zip} = \mathit{zip} \circ (\mathtt{snd} \times \mathtt{snd}) \circ (\mathtt{out}_{S_A} \times \mathtt{out}_{S_A}) \end{aligned}$$

This function can, therefore, be defined as

$$[\![\langle \mathtt{fst} \times \mathtt{fst}, \mathtt{snd} \times \mathtt{snd} \rangle \circ (\mathtt{out}_{S_A} \times \mathtt{out}_{S_A})]\!]_{S_A} \ .$$

EXAMPLE 5. The codatatype of colists over a given set A can be represented as the terminal coalgebra $(\nu L_A, \mathtt{out}_{L_A})$ of the functor L_A. Write List'_A for νL_A. Given any function $g : C \to 1 + (A \times C)$, the anamorphism $[\![g]\!]_{L_A}$ is the unique solution $f : C \to \mathit{List}'_A$ of the equation $\mathtt{out}_{L_A} \circ f = (\mathtt{id}_1 + (\mathtt{id}_A \times f)) \circ g$, i.e. the function $\mathit{unfold}(g)$ from functional programming.

4. Primitive (co-)recursion

Paramorphisms

EXAMPLE 6. The factorial function $\mathit{fact} : \mathit{Nat} \to \mathit{Nat}$ is most neatly characterized as the unique solution of the equation system

$$\begin{aligned} & \mathit{fact} \circ \mathit{zero} = \mathit{succ} \circ \mathit{zero} \\ \wedge\ & \mathit{fact} \circ \mathit{succ} = \mathit{mult} \circ \langle \mathit{fact}, \mathit{succ} \rangle. \end{aligned}$$

This function is not a catamorphism. The problem is that its value for a given argument depends not only on the values for the immediate subparts of the argument, but also on these immediate subparts directly.

Meertens (1992) showed that any function which can be characterized similarly to the factorial function is definable as the composition of the left projection and a catamorphism.

The relevant result is the following:

For any two morphisms $f : \mu F \to C$ and $\varphi : F\,(C \times \mu F) \to C$, we have

$$f \circ \mathtt{in}_F = \varphi \circ F\,\langle\, f, \mathtt{id}_{\mu F}\,\rangle \;\Leftrightarrow\; f = \mathtt{fst} \circ (\!|\, \langle\, \varphi, \mathtt{in}_F \circ F\,\mathtt{snd}\,\rangle \,|\!)_F \qquad \text{L}\textsc{emma}\text{-2}$$

It is proved by the following calculations:

$$\left[\begin{array}{ll}
\multicolumn{2}{l}{\rhd\; f \circ \mathtt{in}_F = \varphi \circ F\,\langle\, f, \mathtt{id}_{\mu F}\,\rangle} \\ \hline
 & f \\
= & \quad \text{– pairing –} \\
 & \mathtt{fst} \circ \langle\, f, \mathtt{id}_{\mu F}\,\rangle \\
= & \quad \text{– cata-C}\textsc{harn}\text{ –} \\
 & \left[\begin{array}{ll}
 & \langle\, f, \mathtt{id}_{\mu F}\,\rangle \circ \mathtt{in}_F \\
 = & \quad \text{– pairing –} \\
 & \langle\, f \circ \mathtt{in}_F, \mathtt{in}_F\,\rangle \\
 = & \quad \text{– } F \text{ functor –} \\
 & \langle\, f \circ \mathtt{in}_F, \mathtt{in}_F \circ F\,\mathtt{id}_{\mu F}\,\rangle \\
 = & \quad \text{– pairing –} \\
 & \langle\, f \circ \mathtt{in}_F, \mathtt{in}_F \circ F\,(\mathtt{snd} \circ \langle\, f, \mathtt{id}_{\mu F}\,\rangle)\,\rangle \\
 = & \quad \text{– } \lhd, F \text{ functor –} \\
 & \langle\, \varphi \circ F\,\langle\, f, \mathtt{id}_{\mu F}\,\rangle, \mathtt{in}_F \circ F\,\mathtt{snd} \circ F\,\langle\, f, \mathtt{id}_{\mu F}\,\rangle\,\rangle \\
 = & \quad \text{– pairing –} \\
 & \langle\, \varphi, \mathtt{in}_F \circ F\,\mathtt{snd}\,\rangle \circ F\,\langle\, f, \mathtt{id}_{\mu F}\,\rangle
 \end{array}\right. \\
 & \mathtt{fst} \circ (\!|\, \langle\, \varphi, \mathtt{in}_F \circ F\,\mathtt{snd}\,\rangle \,|\!)_F
\end{array}\right.$$

$$\left[\begin{array}{ll}
\multicolumn{2}{l}{\rhd\; f = \mathtt{fst} \circ (\!|\, \langle\, \varphi, \mathtt{in}_F \circ F\,\mathtt{snd}\,\rangle \,|\!)} \\ \hline
 & f \circ \mathtt{in}_F \\
= & \quad \text{– } \lhd \text{ –} \\
 & \mathtt{fst} \circ (\!|\, \langle\, \varphi, \mathtt{in}_F \circ F\,\mathtt{snd}\,\rangle \,|\!) \circ \mathtt{in}_F \\
= & \quad \text{– cata-C}\textsc{harn}\text{ –} \\
 & \mathtt{fst} \circ \langle\, \varphi, \mathtt{in}_F \circ F\,\mathtt{snd}\,\rangle \circ F\,(\!|\, \langle\, \varphi, \mathtt{in}_F \circ F\,\mathtt{snd}\,\rangle \,|\!)_F \\
= & \quad \text{– pairing, 2x –} \\
 & \varphi \circ F\,\langle\, \mathtt{fst} \circ (\!|\, \langle\, \varphi, \mathtt{in}_F \circ F\,\mathtt{snd}\,\rangle \,|\!)_F, \mathtt{snd} \circ (\!|\, \langle\, \varphi, \mathtt{in}_F \circ F\,\mathtt{snd}\,\rangle \,|\!)_F\,\rangle \\
= & \quad \text{– } \lhd \text{, cata-F}\textsc{usion}\text{ –} \\
 & \left[\begin{array}{ll}
 & \mathtt{snd} \circ \langle\, \varphi, \mathtt{in}_F \circ F\,\mathtt{snd}\,\rangle \\
 = & \quad \text{– pairing –} \\
 & \mathtt{in}_F \circ F\,\mathtt{snd}
 \end{array}\right. \\
 & \varphi \circ F\,\langle\, f, (\!|\, \mathtt{in}_F \,|\!)_F\,\rangle \\
= & \quad \text{– cata-R}\textsc{efl}\text{ –} \\
 & \varphi \circ F\,\langle\, f, \mathtt{id}_{\mu F}\,\rangle
\end{array}\right.$$

EXAMPLE 7. For the factorial function, we have that

$$\begin{aligned}&\mathit{fact} \circ \mathtt{in}_F\\ &\quad = [\, \mathit{succ} \circ \mathit{zero}, \mathit{mult} \circ \langle\, \mathit{fact}, \mathit{succ}\,\rangle\,]\\ &\quad = [\, \mathit{succ} \circ \mathit{zero}, \mathit{mult} \circ (\mathtt{id}_{\mathit{Nat}} \times \mathit{succ})\,] \circ N\,\langle\, \mathit{fact}, \mathtt{id}_{\mathit{Nat}}\,\rangle.\end{aligned}$$

and consequently we get the definition of factorial

$$\begin{aligned}&\mathit{fact}\\ &\quad = \mathtt{fst} \circ (\!|\, \langle\, [\, \mathit{succ} \circ \mathit{zero}, \mathit{mult} \circ (\mathtt{id}_{\mathit{Nat}} \times \mathit{succ})\,], \mathtt{in}_F \circ N\,\mathtt{snd}\,\rangle\, |\!)_N\\ &\quad = \mathtt{fst} \circ (\!|\, \langle\, [\, \mathit{succ} \circ \mathit{zero}, \mathit{mult} \circ (\mathtt{id}_{\mathit{Nat}} \times \mathit{succ})\,], [\, \mathit{zero}, \mathit{succ} \circ \mathtt{snd}\,]\,\rangle\, |\!)_N\end{aligned}$$

To make programming and program reasoning easier, let us introduce a new construction and study its properties. For any morphism $\varphi : F\,(C \times \mu F) \to C$, define the morphism $\langle\!|\, \varphi\, |\!\rangle_F : \mu F \to C$ by

$$\langle\!|\, \varphi\, |\!\rangle_F = \mathtt{fst} \circ (\!|\, \langle\, \varphi, \mathtt{in}_F \circ F\,\mathtt{snd}\,\rangle\, |\!)_F \qquad \text{para-DEF}$$

Morphisms of the form $\langle\!|\, \varphi\, |\!\rangle_F$ are called *paramorphisms* ($\pi\alpha\rho\alpha$ — Greek preposition meaning 'near to', 'at the side of', 'towards'; the name is due to Meertens). The construction $\langle\!|\, \cdot\, |\!\rangle_F$ is a *primitive recursor.*

From Lemma 2, we get the characterization of paramorphisms by the following universal property:

$$f \circ \mathtt{in}_F = \varphi \circ F\,\langle\, f, \mathtt{id}_{\mu F}\,\rangle \;\Leftrightarrow\; f = \langle\!|\, \varphi\, |\!\rangle_F \qquad \text{para-CHARN}$$

The type information is summarized in the following diagram:

$$\begin{tikzcd}
F\,\mu F \arrow[r, "\mathtt{in}_F"] \arrow[d, "F\,\langle\, f{,}\mathtt{id}_{\mu F}\,\rangle"'] & \mu F \arrow[d, "f"] \\
F\,(C \times \mu F) \arrow[r, "\varphi"] & C
\end{tikzcd}$$

EXAMPLE 8. The paramorphism $\langle\!|\, [\, \mathit{succ} \circ \mathit{zero}, \mathit{mult} \circ (\mathtt{id}_{\mathit{Nat}} \times \mathit{succ})\,]\, |\!\rangle_N$ defines the factorial function.

More generally, given any two functions $c : 1 \to C$ and $h : C \times \mathit{Nat} \to C$, the paramorphism $f = \langle\!|\, [\, c, h\,]\, |\!\rangle_N : \mathit{Nat} \to C$ is the unique solution of the equation system

$$\begin{aligned} & f \circ \mathit{zero} = c \\ \wedge\ & f \circ \mathit{succ} = h \circ \langle\, f, \mathtt{id}_{\mathit{Nat}} \,\rangle. \end{aligned}$$

The calculational properties of paramorphisms are similar to those of catamorphisms. In particular, we have the following "paramorphic" cancellation, reflection and fusion laws:

$\llparenthesis\, \varphi \,\rrparenthesis_F \circ \mathtt{in}_F = \varphi \circ F\,\langle\, \llparenthesis\, \varphi \,\rrparenthesis_F, \mathtt{id}_{\mu F} \,\rangle$	para-CANCEL
$\mathtt{id}_{\mu F} = \llparenthesis\, \mathtt{in}_F \circ F\,\mathtt{fst} \,\rrparenthesis$	para-REFL
$f \circ \varphi = \psi \circ F\,(f \times \mathtt{id}_{\mu F}) \;\Rightarrow\; f \circ \llparenthesis\, \varphi \,\rrparenthesis_F = \llparenthesis\, \psi \,\rrparenthesis_F$	para-FUSION

The reflection and fusion laws are proved as follows:

$$\left[\begin{array}{ll}
 & \mathtt{id}_{\mu F} \\
= & \quad -\ \text{para-}\textsc{Charn}\ - \\
 & \left[\begin{array}{ll}
 & \mathtt{in}_F \\
 = & \quad -\ F \text{ functor}\ - \\
 & \mathtt{in}_F \circ F\,\mathtt{id}_{\mu F} \\
 = & \quad -\ \text{pairing}\ - \\
 & \mathtt{in}_F \circ F\,(\mathtt{fst} \circ \langle\, \mathtt{id}_{\mu F}, \mathtt{id}_{\mu F} \,\rangle) \\
 = & \quad -\ F \text{ functor}\ - \\
 & \mathtt{in}_F \circ F\,\mathtt{fst} \circ F\,\langle\, \mathtt{id}_{\mu F}, \mathtt{id}_{\mu F} \,\rangle
 \end{array}\right. \\
 & \llparenthesis\, \mathtt{in}_F \circ F\,\mathtt{fst} \,\rrparenthesis_F
\end{array}\right.$$

$$\left[\begin{array}{ll}
\multicolumn{2}{l}{\triangleright\ f \circ \varphi = \psi \circ F\,(f \times \mathtt{id}_{\mu F})} \\ \hline
 & f \circ \llparenthesis\, \varphi \,\rrparenthesis_F \\
= & \quad -\ \text{para-}\textsc{Charn}\ - \\
 & \left[\begin{array}{ll}
 & f \circ \llparenthesis\, \varphi \,\rrparenthesis_F \circ \mathtt{in}_F \\
 = & \quad -\ \text{para-}\textsc{Charn}\ - \\
 & f \circ \varphi \circ F\,\langle\, \llparenthesis\, \varphi \,\rrparenthesis_F, \mathtt{id}_{\mu F} \,\rangle \\
 = & \quad -\ \triangleleft\ - \\
 & \psi \circ F\,(f \times \mathtt{id}_{\mu F}) \circ F\,\langle\, \llparenthesis\, \varphi \,\rrparenthesis_F, \mathtt{id}_{\mu F} \,\rangle \\
 = & \quad -\ F \text{ functor}\ - \\
 & \psi \circ F\,((f \times \mathtt{id}_{\mu F}) \circ \langle\, \llparenthesis\, \varphi \,\rrparenthesis_F, \mathtt{id}_{\mu F} \,\rangle) \\
 = & \quad -\ \text{pairing}\ - \\
 & \psi \circ F\,\langle\, f \circ \llparenthesis\, \varphi \,\rrparenthesis_F, \mathtt{id}_{\mu F} \,\rangle
 \end{array}\right. \\
 & \llparenthesis\, \psi \,\rrparenthesis_F
\end{array}\right.$$

By definition, every paramorphism is the composition of the left projection and a catamorphism. In converse, paramorphisms can be viewed as a generalization of catamorphisms, in the sense that every catamorphism is definable as a certain paramorphism:

$$(\!|\,\varphi\,|\!)_F = \langle\!|\,\varphi \circ F\,\mathtt{fst}\,|\!\rangle_F \qquad\qquad \text{para-}\textsc{Cata}$$

This law is verified by the following calculation:

$$\left[\begin{array}{ll}
 & (\!|\,\varphi\,|\!)_F \\
= & \quad \text{– para-}\textsc{Charn}\text{ –} \\
 & \left[\begin{array}{ll}
 & (\!|\,\varphi\,|\!)_F \circ \mathtt{in}_F \\
= & \quad \text{– cata-}\textsc{Charn}\text{ –} \\
 & \varphi \circ F\,(\!|\,\varphi\,|\!)_F \\
= & \quad \text{– pairing –} \\
 & \varphi \circ F\,(\mathtt{fst} \circ \langle\,(\!|\,\varphi\,|\!)_F, \mathtt{id}_{\mu F}\,\rangle) \\
= & \quad \text{– } F \text{ functor –} \\
 & \varphi \circ F\,\mathtt{fst} \circ F\,\langle\,(\!|\,\varphi\,|\!)_F, \mathtt{id}_{\mu F}\,\rangle
\end{array}\right. \\
 & \langle\!|\,\varphi \circ F\,\mathtt{fst}\,|\!\rangle_F
\end{array}\right.$$

Actually, every morphism whose source is the carrier of an initial algebra is a paramorphism:

$$f = \langle\!|\, f \circ \mathtt{in}_F \circ F\,\mathtt{snd}\,|\!\rangle_F \qquad\qquad \text{para-}\textsc{Any}$$

This observation, first recorded in (Meertens, 1992), is proved as follows:

$$\left[\begin{array}{ll}
 & f \\
= & \quad \text{– para-}\textsc{Charn}\text{ –} \\
 & \left[\begin{array}{ll}
 & f \circ \mathtt{in}_F \\
= & \quad \text{– } F \text{ functor –} \\
 & f \circ \mathtt{in}_F \circ F\,\mathtt{id}_{\mu F} \\
= & \quad \text{– pairing –} \\
 & f \circ \mathtt{in}_F \circ F\,(\mathtt{snd} \circ \langle\, f, \mathtt{id}_{\mu F}\,\rangle) \\
= & \quad \text{– } F \text{ functor –} \\
 & f \circ \mathtt{in}_F \circ F\,\mathtt{snd} \circ F\,\langle\, f, \mathtt{id}_{\mu F}\,\rangle
\end{array}\right. \\
 & \langle\!|\, f \circ \mathtt{in}_F \circ F\,\mathtt{snd}\,|\!\rangle_F
\end{array}\right.$$

From this law, it is a very short step to the following definition of the morphism $\mathtt{in}_F^{-1}$ as a paramorphism:

$$\mathtt{in}_F^{-1} = \llparenthesis\, F\,\mathtt{snd} \,\rrparenthesis_F \qquad\qquad \text{para-}\textsc{In-Inv}$$

The proof proceeds as follows:

$$\left[\begin{array}{ll} & \mathtt{in}_F^{-1} \\ = & \quad \text{– para-}\textsc{Any}\text{ –} \\ & \llparenthesis\, \mathtt{in}_F^{-1} \circ \mathtt{in}_F \circ F\,\mathtt{snd} \,\rrparenthesis_F \\ = & \quad \text{– in-inv-}\textsc{Charn}\text{ –} \\ & \llparenthesis\, F\,\mathtt{snd} \,\rrparenthesis_F \end{array}\right.$$

Apomorphisms Let us now dualize everything we know about paramorphisms. For any morphism $\varphi : C \to F\,(C + \nu F)$, define the morphism $\llbracket\!\langle \varphi \rangle\!\rrbracket_F : C \to \nu F$ as a composition of a certain anamorphism with the left injection:

$$\llbracket\!\langle \varphi \rangle\!\rrbracket_F = \llbracket\, [\,\varphi, F\,\mathtt{inr} \circ \mathtt{out}_F\,]\,\rrbracket_F \circ \mathtt{inl} \qquad\qquad \text{apo-}\textsc{Def}$$

Morphisms of the form $\llbracket\!\langle \varphi \rangle\!\rrbracket_F$ are the dual of paramorphisms. Both in (Vos, 95) and (Vene and Uustalu, 1998), these morphisms are called *apomorphisms* ($\alpha\pi o$ — Greek preposition meaning ‘apart from’, ‘far from’, ‘away from’)[1]. The construction $\llbracket\!\langle \cdot \rangle\!\rrbracket_F$ is, of course, a *primitive corecursor*. The characterizing universal property for apomorphisms is following:

$$\mathtt{out}_F \circ f = F\,[\,f, \mathtt{id}_{\nu F}\,] \circ \varphi \;\Leftrightarrow\; f = \llbracket\!\langle \varphi \rangle\!\rrbracket_F \qquad\qquad \text{apo-}\textsc{Charn}$$

The type information is summarized in the following diagram:

$$\begin{tikzcd} C \arrow[r, "\varphi"] \arrow[d, "f"'] & F\,(C + \nu F) \arrow[d, "F\,[\,f, \mathtt{id}_{\nu F}\,]"] \\ \nu F \arrow[r, "\mathtt{out}_F"] & F\,\nu F \end{tikzcd}$$

---

[1] A curious thing is that we wrote (Vene and Uustalu, 1998) unaware of the earlier existence of (Vos, 95), but happened to come up with the exactly the same new name

The laws for apomorphisms are just the duals of those for paramorphisms. The cancellation, reflection and fusion laws are these:

$$\begin{array}{lr}
\mathtt{out}_F \circ \llbracket \varphi \rrbracket_F = F\,[\,\llbracket \varphi \rrbracket_F, \mathtt{id}_{\nu F}\,] \circ \varphi & \text{apo-}\textsc{Cancel} \\
\mathtt{id}_{\nu F} = \llbracket F\,\mathtt{inl} \circ \mathtt{out}_F \rrbracket_F & \text{apo-}\textsc{Refl} \\
\psi \circ f = F\,(f + \mathtt{id}_{\nu F}) \circ \varphi \;\Rightarrow\; \llbracket \psi \rrbracket_F \circ f = \llbracket \varphi \rrbracket_F & \text{apo-}\textsc{Fusion}
\end{array}$$

As paramorphisms generalized catamorphisms, apomorphisms are a generalization of anamorphisms.

$$[\!(\, \varphi \,)\!]_F = \llbracket F\,\mathtt{inl} \circ \varphi \rrbracket_F \qquad \text{apo-}\textsc{Ana}$$

Also, any morphism whose target is the carrier of a terminal coalgebra, is an apomorphism.

$$f = \llbracket F\,\mathtt{inr} \circ \mathtt{out}_F \circ f \rrbracket_F \qquad \text{apo-}\textsc{Any}$$

The morphism $\mathtt{out}_F^{-1}$ has this nice “apomorphic” definition:

$$\mathtt{out}_F^{-1} = \llbracket F\,\mathtt{inr} \rrbracket_F \qquad \text{apo-}\textsc{Out-Inv}$$

EXAMPLE 9. The function $\mathit{maphd}(h) : \mathit{Stream}_A \to \mathit{Stream}_A$, which modifies any input stream by applying a function $h : A \to A$ to its head while leaving the tail unchanged, is definable as the composition of a certain anamorphism with the left injection that constructs the value for a given argument from the value of h for its head and from the elements of its tail.

$$\begin{array}{l}
\mathit{maphd}(h) \\
\;= [\!(\, [\,\langle h \circ \mathit{head}, \mathtt{inr} \circ \mathit{tail} \rangle, \langle \mathit{head}, \mathtt{inr} \circ \mathit{tail} \rangle\,] \,)\!]_{S_A} \circ \mathtt{inl} \\
\;= [\!(\, [\,\langle h \circ \mathit{head}, \mathtt{inr} \circ \mathit{tail} \rangle, S_A\,\mathtt{inr} \circ \mathtt{out}_{S_A}\,] \,)\!]_{S_A} \circ \mathtt{inl}
\end{array}$$

A much more natural definition of this function is given by an apomorphism that constructs the value for a given argument from the value of h for its head and from the whole of its tail.

$$\mathit{maphd}(h) = \llbracket \langle h \circ \mathit{head}, \mathtt{inr} \circ \mathit{tail} \rangle \rrbracket_{S_A}$$

EXAMPLE 10. Assume that a given set A is linearly ordered by a predicate $(\preceq) : A \times A \to 1 + 1$. For any function $x : 1 \to A$, the function $\mathit{insert}(x) : \mathit{Stream}_A \to \mathit{Stream}_A$, that inserts the element x into an input strean immediately before the first element that x is less than or equal to (so that the returned stream will be sorted, if the given stream is), is characterized by the equation

$$\langle\, head, tail \,\rangle \circ insert(x) \circ ys \;=\; \begin{cases} \langle\, x, ys \,\rangle & \text{if } x \preceq head \circ ys \\ \langle\, head, insert(x) \circ tail \,\rangle \circ ys & \text{otherwise} \end{cases}$$

The function $insert(x)$ is most naturally defined as an apomorphism that constructs the value for a given argument from its certain initial segment elementwise and the remainder as one whole:

$$insert(x) = [\!(\, [\, \langle\, x \circ !, \mathtt{inr} \,\rangle, (\mathtt{id}_A \times \mathtt{inl}) \,] \circ test(x) \,)\!]_{S_A}$$

where $test(x) : Stream_A \to Stream_A + S_A\, Stream_A$ is

$$test(x) = (\mathtt{fst}{+}\mathtt{snd}) \circ ((\preceq) \circ \langle\, x \circ !, \mathtt{fst} \circ \mathtt{snd} \,\rangle)? \circ \langle\, \mathtt{id}_{Stream_A}, \langle\, head, tail \,\rangle \,\rangle$$

EXAMPLE 11. For any function $\ell : 1 \to List'_A$, the function $append(\ell) : List'_A \to List'_A$, which appends the colist ℓ to an input colist, is naturally definable as an apomorphism which constructs the value for a given argument from the elements of the argument and from the whole of ℓ:

$$\begin{aligned} & append(\ell) \\ &= [\!(\, [\, [\, \mathtt{inl}, \mathtt{inr} \circ (\mathtt{id}_A \times \mathtt{inr}) \,] \circ \mathtt{out}_{L_A} \circ \ell, \mathtt{inr} \circ (\mathtt{id}_A \times \mathtt{inl}) \,] \circ \mathtt{out}_{L_A} \,)\!]_{L_A} \\ &= [\!(\, [\, L_A\, \mathtt{inr} \circ \mathtt{out}_{L_A} \circ \ell, \mathtt{inr} \,] \circ L_A\, \mathtt{inl} \circ \mathtt{out}_{L_A} \,)\!]_{L_A} \end{aligned}$$

5. Course-of-value (co)iteration

Histomorphisms

EXAMPLE 12. The famous Fibonacci function $fibo : Nat \to Nat$ is most smoothly characterized as the unique solution of the equation system

$$\begin{aligned} & fibo \circ zero = one \\ \wedge\ & fibo \circ one = one \\ \wedge\ & fibo \circ succ \circ succ = add \circ \langle\, fibo \circ succ, fibo \,\rangle, \end{aligned}$$

where $one = succ \circ zero$. This very nice characterization does not give us any definition of *fibo* in terms of catamorphisms. The problem is that the value of *fibo* for a given argument is defined not via the values for the immediate subparts of the argument, but via the values for its subparts of depth 2. But the characterization of *fibo* together with the function $\mathit{fibo}' : \mathit{Nat} \to \mathit{Nat} \times \mathit{Nat}$ (which, for any argument n, returns the pair formed of the value of *fibo* for n and either zero or the value of *fibo* for the predecessor of n) by a much trickier equation system, viz.,

$$\begin{aligned} & \mathit{fibo} = \mathtt{fst} \circ \mathit{fibo}' \\ \wedge\ & \mathit{fibo}' \circ \mathit{zero} = \langle\, \mathit{one}, \mathit{zero} \,\rangle \\ \wedge\ & \mathit{fibo}' \circ \mathit{succ} = \langle\, \mathit{add}, \mathtt{fst} \,\rangle \circ \mathit{fibo}', \end{aligned}$$

leads to a definition of *fibo* as the composition of the left projection and a catamorphism:

$$\mathit{fibo} = \mathtt{fst} \circ \llparenthesis\, \langle\, [\, \mathit{one}, \mathit{add} \,], [\, \mathit{zero}, \mathtt{fst} \,] \,\rangle \,\rrparenthesis_{\mathit{Nat}}.$$

To make programming and program reasoning easier, we could introduce a new construction that would capture the natural definition scheme of the Fibonacci function and closely similar functions, and start studying its properties. But this would only provide us with a partial solution to the problem manifested by the Fibonacci example, as one can imagine functions whose value for a given argument is naturally defined via the values for its subparts of depth three, four, etc. Instead of this, we introduce a construction that captures course-of-value iteration, which is a function definition scheme where the value of a function for a given argument may depend on the values for just any of its subparts.

We start with the following result, which, to our knowledge, is novel:

Let $F^{\times}_A$ denote the functor $F^{\times}_A\, X = A \times F\, X$, $F^{\times}_A\, h = \mathtt{id}_A \times F\, h$. Then, for any two morphisms $f : \mu F \to C$ and $\varphi : F(\nu F^{\times}_C) \to C$, we have

$$\begin{aligned} & f \circ \mathtt{in}_F = \varphi \circ F \llbracket\, \langle\, f, \mathtt{in}_F^{-1} \,\rangle \,\rrbracket_{F^{\times}} \\ & \qquad \Leftrightarrow \quad f = \mathtt{fst} \circ \mathtt{out}_{F^{\times}} \circ \llparenthesis\, \mathtt{out}_{F^{\times}}^{-1} \circ \langle\, \varphi, \mathtt{id} \,\rangle \,\rrparenthesis_F \qquad \text{LEMMA-3} \end{aligned}$$

Proving it is quite tricky.

$$\left[\begin{array}{ll}
\triangleright & f \circ \mathtt{in}_F = \varphi \circ F\,[\!(\,\langle f, \mathtt{in}_F^{-1} \rangle\,)\!]_{F^\times} \\ \hline
& f \\
= & \quad \text{– pairing –} \\
& \mathtt{fst} \circ \langle f, \mathtt{in}_F^{-1} \rangle \\
= & \quad \text{– pairing –} \\
& \mathtt{fst} \circ (\mathtt{id} \times F\,[\!(\,\langle f, \mathtt{in}_F^{-1} \rangle\,)\!]_{F^\times}) \circ \langle f, \mathtt{in}_F^{-1} \rangle \\
= & \quad \text{– ana-}\textsc{Charn}\text{ –} \\
& \mathtt{fst} \circ \mathtt{out}_{F^\times} \circ [\!(\,\langle f, \mathtt{in}_F^{-1} \rangle\,)\!]_{F^\times} \\
= & \quad \text{– cata-}\textsc{Charn}\text{ –} \\
& \left[\begin{array}{ll}
& [\!(\,\langle f, \mathtt{in}_F^{-1} \rangle\,)\!] \circ \mathtt{in}_F \\
= & \quad \text{– out-inv-}\textsc{Charn}\text{ –} \\
& \mathtt{out}_{F^\times}^{-1} \circ \mathtt{out}_{F^\times} \circ [\!(\,\langle f, \mathtt{in}_F^{-1} \rangle\,)\!]_{F^\times} \circ \mathtt{in}_F \\
= & \quad \text{– ana-}\textsc{Charn}\text{ –} \\
& \mathtt{out}_{F^\times}^{-1} \circ (\mathtt{id} \times F\,[\!(\,\langle f, \mathtt{in}_F^{-1} \rangle\,)\!]_{F^\times}) \circ \langle f, \mathtt{in}_F^{-1} \rangle \circ \mathtt{in}_F \\
= & \quad \text{– pairing –} \\
& \mathtt{out}_{F^\times}^{-1} \circ \langle f, F\,[\!(\,\langle f, \mathtt{in}_F^{-1} \rangle\,)\!]_{F^\times} \circ \mathtt{in}_F^{-1} \rangle \circ \mathtt{in}_F \\
= & \quad \text{– pairing –} \\
& \mathtt{out}_{F^\times}^{-1} \circ \langle f \circ \mathtt{in}_F, F\,[\!(\,\langle f, \mathtt{in}_F^{-1} \rangle\,)\!]_{F^\times} \circ \mathtt{in}_F^{-1} \circ \mathtt{in}_F \rangle \\
= & \quad \text{– } \triangleleft\text{, in-inv-}\textsc{Charn}\text{ –} \\
& \mathtt{out}_{F^\times}^{-1} \circ \langle \varphi \circ F\,[\!(\,\langle f, \mathtt{in}_F^{-1} \rangle\,)\!]_{F^\times}, F\,[\!(\,\langle f, \mathtt{in}_F^{-1} \rangle\,)\!]_{F^\times} \rangle \\
= & \quad \text{– pairing –} \\
& \mathtt{out}_{F^\times}^{-1} \circ \langle \varphi, \mathtt{id} \rangle \circ F\,[\!(\,\langle f, \mathtt{in}_F^{-1} \rangle\,)\!]_{F^\times}
\end{array}\right. \\
& \mathtt{fst} \circ \mathtt{out}_{F^\times} \circ (\!|\, \mathtt{out}_{F^\times}^{-1} \circ \langle \varphi, \mathtt{id} \rangle \,|\!)_F
\end{array}\right.$$

$$
\begin{array}{ll}
\multicolumn{2}{l}{\triangleright\ f = \mathtt{fst}\circ\mathtt{out}_{F^\times}\circ\llparenthesis\,\mathtt{out}^{-1}_{F^\times}\circ\langle\,\varphi,\mathtt{id}\,\rangle\,\rrparenthesis_F} \\ \hline
 & f\circ\mathtt{in}_F \\
= & \quad -\triangleleft- \\
 & \mathtt{fst}\circ\mathtt{out}_{F^\times}\circ\llparenthesis\,\mathtt{out}^{-1}_{F^\times}\circ\langle\,\varphi,\mathtt{id}\,\rangle\,\rrparenthesis\circ\mathtt{in}_F \\
= & \quad -\text{cata-}\textsc{Charn}- \\
 & \mathtt{fst}\circ\mathtt{out}_{F^\times}\circ\mathtt{out}^{-1}_{F^\times}\circ\langle\,\varphi,\mathtt{id}\,\rangle\circ F\,\llparenthesis\,\mathtt{out}^{-1}_{F^\times}\circ\langle\,\varphi,\mathtt{id}\,\rangle\,\rrparenthesis_F \\
= & \quad -\text{out-inv-}\textsc{Charn}- \\
 & \mathtt{fst}\circ\langle\,\varphi,\mathtt{id}\,\rangle\circ F\,\llparenthesis\,\mathtt{out}^{-1}_{F^\times}\circ\langle\,\varphi,\mathtt{id}\,\rangle\,\rrparenthesis_F \\
= & \quad -\text{pairing}- \\
 & \varphi\circ F\,\llparenthesis\,\mathtt{out}^{-1}_{F^\times}\circ\langle\,\varphi,\mathtt{id}\,\rangle\,\rrparenthesis_F \\
= & \quad -\text{ana-}\textsc{Charn}- \\
 & \left[\begin{array}{ll}
 & \mathtt{out}_{F^\times}\circ\llparenthesis\,\mathtt{out}^{-1}_{F^\times}\circ\langle\,\varphi,\mathtt{id}\,\rangle\,\rrparenthesis_F \\
= & \quad -\text{pairing}- \\
 & \langle\mathtt{fst}\circ\mathtt{out}_{F^\times}\circ\llparenthesis\,\mathtt{out}^{-1}_{F^\times}\circ\langle\,\varphi,\mathtt{id}\,\rangle\,\rrparenthesis_F, \\
 & \qquad\qquad \mathtt{snd}\circ\mathtt{out}_{F^\times}\circ\llparenthesis\,\mathtt{out}^{-1}_{F^\times}\circ\langle\,\varphi,\mathtt{id}\,\rangle\,\rrparenthesis_F\rangle \\
= & \quad -\triangleleft,\ \text{in-inv-}\textsc{Charn}- \\
 & \langle\,f,\mathtt{snd}\circ\mathtt{out}_{F^\times}\circ\llparenthesis\,\mathtt{out}^{-1}_{F^\times}\circ\langle\,\varphi,\mathtt{id}\,\rangle\,\rrparenthesis_F\circ\mathtt{in}_F\circ\mathtt{in}_F^{-1}\,\rangle \\
= & \quad -\text{cata-}\textsc{Charn}- \\
 & \langle f,\mathtt{snd}\circ\mathtt{out}_{F^\times}\circ\mathtt{out}^{-1}_{F^\times}\circ\langle\,\varphi,\mathtt{id}\,\rangle\circ \\
 & \qquad\qquad \circ F\,\llparenthesis\,\mathtt{out}^{-1}_{F^\times}\circ\langle\,\varphi,\mathtt{id}\,\rangle\,\rrparenthesis_F\circ\mathtt{in}_F^{-1} \\
= & \quad -\text{out-inv-}\textsc{Charn}- \\
 & \langle\,f,\mathtt{snd}\circ\langle\,\varphi,\mathtt{id}\,\rangle\circ F\,\llparenthesis\,\mathtt{out}^{-1}_{F^\times}\circ\langle\,\varphi,\mathtt{id}\,\rangle\,\rrparenthesis_F\circ\mathtt{in}_F^{-1}\,\rangle \\
= & \quad -\text{pairing}- \\
 & \langle\,f,F\,\llparenthesis\,\mathtt{out}^{-1}_{F^\times}\circ\langle\,\varphi,\mathtt{id}\,\rangle\,\rrparenthesis_F\circ\mathtt{in}_F^{-1}\,\rangle \\
= & \quad -\text{pairing}- \\
 & (\mathtt{id}\times F\,\llparenthesis\,\mathtt{out}^{-1}_{F^\times}\circ\langle\,\varphi,\mathtt{id}\,\rangle\,\rrparenthesis_F)\circ\langle\,f,\mathtt{in}_F^{-1}\,\rangle
 \end{array}\right. \\
 & \varphi\circ F\,\llbracket\,\langle\,f,\mathtt{in}_F^{-1}\,\rangle\,\rrbracket_{F^\times}
\end{array}
$$

For any morphism $\varphi\ :\ F(\nu F^\times_C)\ \to\ C$, let us now define the morphism $\{\!|\,\varphi\,|\!\}_F : \mu F \to C$ by letting

$$\{\!|\,\varphi\,|\!\}_F = \mathtt{fst}\circ\mathtt{out}_{F^\times}\circ\llparenthesis\,\mathtt{out}^{-1}_{F^\times}\circ\langle\,\varphi,\mathtt{id}\,\rangle\,\rrparenthesis_F \qquad \text{histo-}\textsc{Def}$$

Morphisms of the form $\{\!|\,\varphi\,|\!\}_F$ are called *histomorphisms* in this paper. The construction $\{\!|\cdot|\!\}_F$ is easily seen to be a *course-of-value iterator*. By Lemma 3, the characterizing universal property for histomorphisms is the following:

$$f\circ\mathtt{in}_F = \varphi\circ F\,\llbracket\,\langle\,f,\mathtt{in}_F^{-1}\,\rangle\,\rrbracket_{F^\times}\ \Leftrightarrow\ f = \{\!|\,\varphi\,|\!\}_F \qquad \text{histo-}\textsc{Charn}$$

The type information is summarized in the following diagram:

$$\begin{tikzcd}
F\,\mu F \arrow[r, "\mathtt{in}_F"] \arrow[d, "F\,\llbracket\,\langle\, f, \mathtt{in}_F^{-1}\,\rangle\,\rrbracket_{F^\times}"'] & \mu F \arrow[d, "f"] \\
F(\nu F_C^\times) \arrow[r, "\varphi"] & C
\end{tikzcd}$$

EXAMPLE 13. The following is a neat "histomorphic" definition of the Fibonacci function:

$$\mathit{fibo} = \{\!|\,[\,\mathit{one}, [\,\mathit{one} \circ \mathtt{snd}, \mathit{add} \circ (\mathtt{id} \times (\mathtt{fst} \circ \mathtt{out}_{N^\times}))\,] \circ \mathtt{distl} \circ \mathtt{out}_{N^\times}\,]\,|\!\}_N$$

The cancellation, reflection and fusion laws for histomorphisms are these:

$$\begin{array}{lr}
\{\!|\,\varphi\,|\!\}_F \circ \mathtt{in}_F = \varphi \circ F\,\llbracket\,\langle\,\{\!|\,\varphi\,|\!\}_F, \mathtt{in}_F^{-1}\,\rangle\,\rrbracket_{F^\times} & \text{histo-CANCEL} \\
\mathtt{id}_{\mu F} = \{\!|\,\mathtt{in}_F \circ F\,(\mathtt{fst} \circ \mathtt{out}_{F^\times})\,|\!\}_F & \text{histo-REFL} \\
f \circ \varphi = \psi \circ F\,\llbracket\,(f \times \mathtt{id}) \circ \mathtt{out}_{F^\times}\,\rrbracket_{F^\times} \;\Rightarrow\; f \circ \{\!|\,\varphi\,|\!\}_F = \{\!|\,\psi\,|\!\}_F & \text{histo-FUSION}
\end{array}$$

The reflection law is proved by the following calculation:

$$\left\lceil\begin{array}{ll}
 & \mathtt{id}_{\mu F} \\
= & \quad \text{– histo-CHARN –} \\
 & \left\lceil\begin{array}{ll}
 & \mathtt{in}_F \\
= & \quad \text{– } F \text{ functor –} \\
 & \mathtt{in}_F \circ F\,\mathtt{id} \\
= & \quad \text{– pairing –} \\
 & \mathtt{in}_F \circ F\,(\mathtt{fst} \circ \langle\,\mathtt{id}, \mathtt{in}_F^{-1}\,\rangle) \\
= & \quad \text{– pairing –} \\
 & \mathtt{in}_F \circ F\,(\mathtt{fst} \circ (\mathtt{id} \times F\,\llbracket\,\langle\,\mathtt{id}, \mathtt{in}_F^{-1}\,\rangle\,\rrbracket_{F^\times}) \circ \langle\,\mathtt{id}, \mathtt{in}_F^{-1}\,\rangle) \\
= & \quad \text{– ana-CHARN –} \\
 & \mathtt{in}_F \circ F\,(\mathtt{fst} \circ \mathtt{out}_{F^\times} \circ \llbracket\,\langle\,\mathtt{id}, \mathtt{in}_F^{-1}\,\rangle\,\rrbracket_{F^\times}) \\
= & \quad \text{– } F \text{ functor –} \\
 & \mathtt{in}_F \circ F\,(\mathtt{fst} \circ \mathtt{out}_{F^\times}) \circ F\,\llbracket\,\langle\,\mathtt{id}, \mathtt{in}_F^{-1}\,\rangle\,\rrbracket_{F^\times}
\end{array}\right\rfloor \\
 & \{\!|\,\mathtt{in}_F \circ F(\mathtt{fst} \circ \mathtt{out}_{F^\times})\,|\!\}_F
\end{array}\right\rfloor$$

The fusion law is proved as follows:

$$
\begin{array}{ll}
\rhd & f \circ \varphi = \psi \circ F\,[\![\, (f \times \mathtt{id}) \circ \mathtt{out}_{F^\times} \,]\!]_{F^\times} \\
\hline
 & f \circ \{\!|\, \varphi \,|\!\}_F \\
= & \quad \text{– histo-}\textsc{Charn}\text{ –} \\
 & \quad\quad f \circ \{\!|\, \varphi \,|\!\}_F \circ \mathtt{in}_F \\
 & \quad = \quad\quad \text{– histo-}\textsc{Charn}\text{ –} \\
 & \quad\quad f \circ \varphi \circ F\,[\![\, \langle\, \{\!|\, \varphi \,|\!\}, \mathtt{in}_F^{-1} \,\rangle \,]\!]_{F^\times} \\
 & \quad = \quad\quad - \lhd - \\
 & \quad\quad \psi \circ F\,[\![\, (f \times \mathtt{id}) \circ \mathtt{out}_{F^\times} \,]\!]_{F^\times} \circ F\,[\![\, \langle\, \{\!|\, \varphi \,|\!\}_F, \mathtt{in}_F^{-1} \,\rangle \,]\!]_{F^\times} \\
 & \quad = \quad\quad \text{– } F \text{ functor –} \\
 & \quad\quad \psi \circ F\,([\![\, (f \times \mathtt{id}) \circ \mathtt{out}_{F^\times} \,]\!]_{F^\times} \circ [\![\, \langle\, \{\!|\, \varphi \,|\!\}_F, \mathtt{in}_F^{-1} \,\rangle \,]\!]_{F^\times}) \\
 & \quad = \quad\quad \text{– ana-}\textsc{Fusion}\text{ –} \\
 & \quad\quad\quad\quad (f \times \mathtt{id}) \circ \mathtt{out}_{F^\times} \circ [\![\, \langle\, \{\!|\, \varphi \,|\!\}_F, \mathtt{in}_F^{-1} \,\rangle \,]\!]_{F^\times} \\
 & \quad\quad\quad = \quad\quad \text{– ana-}\textsc{Charn}\text{ –} \\
 & \quad\quad\quad\quad (f \times \mathtt{id}) \circ (\mathtt{id} \times F\,[\![\, \langle\, \{\!|\, \varphi \,|\!\}_F, \mathtt{in}_F^{-1} \,\rangle \,]\!]_{F^\times}) \circ \\
 & \quad\quad\quad\quad\quad\quad\quad\quad \circ \langle\, \{\!|\, \varphi \,|\!\}_F, \mathtt{in}_F^{-1} \,\rangle \\
 & \quad\quad\quad = \quad\quad \text{– pairing –} \\
 & \quad\quad\quad\quad (f \times F\,[\![\, \langle\, \{\!|\, \varphi \,|\!\}_F, \mathtt{in}_F^{-1} \,\rangle \,]\!]_{F^\times}) \circ \langle\, \{\!|\, \varphi \,|\!\}_F, \mathtt{in}_F^{-1} \,\rangle \\
 & \quad\quad\quad = \quad\quad \text{– pairing –} \\
 & \quad\quad\quad\quad \langle\, f \circ \{\!|\, \varphi \,|\!\}_F, F\,[\![\, \langle\, \{\!|\, \varphi \,|\!\}_F, \mathtt{in}_F^{-1} \,\rangle \,]\!]_{F^\times} \circ \mathtt{in}_F^{-1} \,\rangle \\
 & \quad\quad\quad = \quad\quad \text{– pairing –} \\
 & \quad\quad\quad\quad (\mathtt{id} \times F\,[\![\, \langle\, \{\!|\, \varphi \,|\!\}_F, \mathtt{in}_F^{-1} \,\rangle \,]\!]_{F^\times}) \circ \langle\, f \circ \{\!|\, \varphi \,|\!\}_F, \mathtt{in}_F^{-1} \,\rangle \\
 & \quad\quad \psi \circ F\,[\![\, \langle\, f \circ \{\!|\, \phi \,|\!\}_F, \mathtt{in}_F^{-1} \,\rangle \,]\!]_{F^\times} \\
 & \{\!|\, \psi \,|\!\}_F
\end{array}
$$

Similarly to paramorphisms, histomorphisms can be viewed as a generalization of catamorphisms:

$$
(\!|\, \varphi \,|\!)_F = \{\!|\, \varphi \circ F\,(\mathtt{fst} \circ \mathtt{out}_{F^\times}) \,|\!\}_F \qquad\qquad \text{histo-}\textsc{Cata}
$$

This fact is verified by the following calculation:

$$
\left[\begin{array}{ll}
 & (\!|\,\varphi\,|\!)_F \\
= & \quad \text{– histo-}\textsc{Charn}\text{ –} \\
 & \left[\begin{array}{ll}
 & (\!|\,\varphi\,|\!)_F \circ \mathtt{in}_F \\
= & \quad \text{– cata-}\textsc{Charn}\text{ –} \\
 & \varphi \circ F\,(\!|\,\varphi\,|\!)_F \\
= & \quad \text{– pairing –} \\
 & \varphi \circ F\,(\mathtt{fst} \circ \langle\, (\!|\,\varphi\,|\!)_F, \mathtt{in}_F^{-1}\,\rangle) \\
= & \quad \text{– pairing –} \\
 & \varphi \circ F\,(\mathtt{fst} \circ (\mathtt{id} \times F\,[\!(\,\langle\, (\!|\,\varphi\,|\!)_F, \mathtt{in}_F^{-1}\,\rangle\,)\!]_{F^\times}) \circ \langle\, (\!|\,\varphi\,|\!)_F, \mathtt{in}_F^{-1}\,\rangle) \\
= & \quad \text{– ana-}\textsc{Charn}\text{ –} \\
 & \varphi \circ F\,(\mathtt{fst} \circ \mathtt{out}_{F^\times} \circ [\!(\,\langle\, (\!|\,\varphi\,|\!)_F, \mathtt{in}_F^{-1}\,\rangle\,)\!]_{F^\times}) \\
= & \quad \text{– } F \text{ functor –} \\
 & \varphi \circ F\,(\mathtt{fst} \circ \mathtt{out}_{F^\times}) \circ F\,[\!(\,\langle\, (\!|\,\varphi\,|\!)_F, \mathtt{in}_F^{-1}\,\rangle\,)\!]_{F^\times}
\end{array}\right. \\
 & \{\!|\,\varphi \circ F\,(\mathtt{fst} \circ \mathtt{out}_{F^\times})\,|\!\}_F
\end{array}\right.
$$

Futumorphisms We now introduce a construction dual to histomorphisms. Let F_A^+ denote the functor $F_A^+\,X = A + F\,X$, $F_A^+\,h = \mathtt{id}_A + F\,h$. Then, given a morphism $\varphi : C \to F\,(\mu F_C^+)$, we define the morphism $\llbracket\,\varphi\,\rrbracket_F : C \to \nu F$ by letting

$$\llbracket\,\varphi\,\rrbracket_F = [\!(\,[\,\varphi, \mathtt{id}\,] \circ \mathtt{in}_{F^+}^{-1}\,)\!]_F \circ \mathtt{in}_{F^+} \circ \mathtt{inl} \qquad \text{futu-}\textsc{Def}$$

Our name for morphisms of the form $\llbracket\,\varphi\,\rrbracket_F$ is *futumorphisms*. The construction $\llbracket\,\cdot\,\rrbracket_F$ is the dual of a course-of-value iterator (a "cocourse-of-argument" coiterator). The characterizing universal property for futumorphisms is the following:

$$\mathtt{out}_F \circ f = F\,(\!|\,[\,f, \mathtt{out}_F^{-1}\,]\,|\!)_{F^+} \circ \varphi \;\Leftrightarrow\; f = \llbracket\,\varphi\,\rrbracket_F \qquad \text{futu-}\textsc{Charn}$$

The type information is summarized in the following diagram:

$$\begin{tikzcd}
C \arrow[r, "\varphi"] \arrow[d, "f"'] & F\,(\mu F_C^+) \arrow[d, "F\,(\!|\,[\,f,\mathtt{out}_F^{-1}\,]\,|\!)_{F^+}"] \\
\nu F \arrow[r, "\mathtt{out}_F"] & F\,\nu F
\end{tikzcd}$$

A straightforward dualization of the laws for histomorphisms gives laws for futumorphisms:

$$\begin{array}{lr}
\mathtt{out}_F \circ [\![\varphi]\!]_F = F\,(\![\, [[\![\varphi]\!]_F, \mathtt{out}_F^{-1}] \,]\!)_{F^+} \circ \varphi & \text{futu-}\textsc{Cancel} \\
\mathtt{id}_{\nu F} = [\![F\,(\mathtt{in}_{F^+} \circ \mathtt{inl}) \circ \mathtt{out}_F]\!]_F & \text{futu-}\textsc{Refl} \\
\psi \circ f = F\,(\![\mathtt{in}_{F^+} \circ (f + \mathtt{id})]\!)_{F^+} \circ \varphi \;\Rightarrow\; [\![\psi]\!]_F \circ f = [\![\varphi]\!]_F & \text{futu-}\textsc{Fusion} \\
[\![\varphi]\!]_F = [\![F\,(\mathtt{in}_{F^+} \circ \mathtt{inl}) \circ \varphi]\!]_F & \text{futu-}\textsc{Ana}
\end{array}$$

EXAMPLE 14. The function $\mathit{exch} : \mathit{Stream}_A \to \mathit{Stream}_A$, which pairwise exchanges the elements of any given argument, is characterized by the equation system

$$\begin{array}{ll}
& \mathit{head} \circ \mathit{exch} = \mathit{head} \circ \mathit{tail} \\
\wedge & \mathit{head} \circ \mathit{tail} \circ \mathit{exch} = \mathit{head} \\
\wedge & \mathit{tail} \circ \mathit{tail} \circ \mathit{exch} = \mathit{exch} \circ \mathit{tail} \circ \mathit{tail}.
\end{array}$$

This function is nicely definable as a futumorphism:

$$\mathit{exch} = [\![\langle \mathit{head} \circ \mathit{tail}, \mathtt{in}_{(S_A)^+} \circ \mathtt{inr} \circ \langle \mathit{head}, \mathtt{in}_{(S_A)^+} \circ \mathtt{inl} \circ \mathit{tail} \circ \mathit{tail} \rangle \rangle]\!]_{S_A}.$$

6. Conclusion and further work We presented a categorical treatment of the function definition schemes of primitive recursion and course-of-value iteration and their duals. These schemes are generalizations of more basic schemes—iteration and coiteration. While the value of an iterative function for a given argument depends solely on the values for its immediate subparts, the value of a primitive recursive function may additionally depend on these immediate subparts directly; the value of a course-of-value iterative function depends on the values for any subparts of the argument. Dually, the argument of a coiterative function for a value may only determine the argument for the immediate subparts of the value, whereas the argument of a primitive corecursive function may alternatively determine these immediate subparts directly; the argument of a "cocourse-of-argument coiterative" function may determine the arguments for any subparts of the value. Primitive (co)recursion and course-of-value (co)iteration, being more "liberal" than (co)iteration, make it easier to write programs and to prove properties about programs.

Codatatypes and the associating notions such as coinduction and bisimulation are used widely in the analysis of processes specifiable by transition systems or state machines (Jacobs and Rutten, 1997). If processes are understood as functions from states (and, optionally, streams of input tokens) to behaviors, sequential composition of processes becomes a natural example of a function elegantly definable as an apomorphism, whereas any concrete process specified by a finite transition or state machine becomes a natural example of a function elegantly definable as a futumorphism. This leads us to believe that apomorphisms and futumorphisms may turn out viable constructions in the modelling of processes. To check out this conjecture is one possible direction for continuing the work reported here.

Acknowledgement The work reported here was partially supported by the Estonian Science Foundation grant no. 2976. The diagrams were produced using the Xy-pic macro package by Kristoffer H. Rose.

Bibliography

Bird, R. S. (1987). An introduction to the theory of lists. In M. Broy (Ed.), *Logic of Programming and Calculi of Discrete Design* (*NATO ASI Series F*, Vol. 36). Springer-Verlag, Berlin. pp. 5–42.

Fokkinga, M. M. (1992). *Law and Order in Algorithmics*. PhD thesis, University of Twente. iv+161 pp.

Geuvers, H. (1992). Inductive and coinductive types with iteration and recursion. In B. Nordström, K. Pettersson, and G. Plotkin (Eds.), *Informal Proceedings of the Workshop on Types for Proofs and Programs, Båstad, June 1992*. Dept. of Computer Sciences, Chalmers Univ. of Technology and Göteborg Univ. pp. 193–217.
URL `ftp://ftp.cs.chalmers.se/pub/cs-reports/baastad.92/`.

Grundy, J. (1996). A browsable format for proof presentation. *Mathesis Universalis*, **2**.
URL `http://saxon.pip.com.pl/MathUniversalis/2/`.

Hagino, T. (1987). *A Categorical Programming Language*. PhD thesis CST-47-87, Univ. of Edinburgh. vii+180 pp.

Jacobs, B., and J. Rutten. (1997). A tutorial on (co)algebras and (co)induction. *Bulletin of the EATCS*, **62**, 222–259.

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

Malcolm, G. (1990). Data structures and program transformation. *Science of Computer Programming*, **14**(2–3), 255–279.

Meertens, L. (1992). Paramorphisms. *Formal Aspects of Computing*, **4**(5), 413–424.

Turner, D. A. (1995). Elementary strong functional programming. In P. H. Hartel and R. Plasmeijer (Eds.), *Proceedings of the 1st Internat. Symp. on Functional Programming Languages in education, FPLE'95, Nijmegen, Dec. 1995* (*Lecture Notes in Computer Science*, Vol. 1022). Springer-Verlag, Berlin. pp. 1-13.

Uustalu, T., and V. Vene. (1997). A cube of proof systems for the intuitionistic predicate μ-,ν-logic. In M. Haveraaen and O. Owe (Eds.), *Selected Papers of the 8th Nordic Workshop on Programming Theory, NWPT'96, Oslo, Dec. 1996*, Research Report 248, Dept. of Informatics, Univ. of Oslo. pp. 237–246.

Vene, V., and T. Uustalu. (1998). Functional programming with apomorphisms (corecursion). *Proceedings of the Estonian Academy of Sciences: Physics, Mathematics*, **47**(3), 147–161.

Vesely, P. (1997). Typechecking the Charity term logic. Unpublished notes. 21 pp.
URL `http://www.cpsc.ucalgary.ca/projects/charity/home.html`.

Vos, T. (1995). *Program Construction and Generation Based on Recursive Types*. MSc thesis INF/SCR-95-12, Univ. of Utrecht. 235 pp.

Received December 1998

T. Uustalu received the MSc degree in system and computer sciences from the Tallinn Technical University in 1992 and the PhD degree in computer science from the Royal Institute of Technology, Stockholm, Sweden, in 1998. Presently, he is a senior lecturer at the Royal Institute of Technology. His scientific interests include structural proof theory and theorem proving, semantics of programming languages, program verification, transformations, and construction, philosophy of logic and computing.

V. Vene obtained the MSc degree in computer science from the University of Tartu, Estonia, in 1994. He is a researcher and PhD student in computer science at the University of Tartu. His scientific interests are programming language design and implementation, functional programming, type theory, semantics based program manipulation, category theory.