CIA Structures and the Semantics of Recursion*

Stefan Milius[1,**], Lawrence S. Moss[2], and Daniel Schwencke[1]

[1] Institut für Theoretische Informatik, Technische Universität Braunschweig, Germany
mail@stefan-milius.eu, schwencke@iti.cs.tu-bs.de
[2] Department of Mathematics, Indiana University, Bloomington, IN, USA
lsm@cs.indiana.edu

Abstract. Final coalgebras for a functor serve as semantic domains for state based systems of various types. For example, formal languages, streams, non-well-founded sets and behaviors of CCS processes form final coalgebras. We present a uniform account of the semantics of recursive definitions in final coalgebras by combining two ideas: (1) final coalgebras are also initial *completely iterative algebras* (cia); (2) additional algebraic operations on final coalgebras may be presented in terms of a *distributive law* λ. We first show that a distributive law leads to new extended cia structures on the final coalgebra. Then we formalize recursive function definitions involving operations given by λ as recursive program schemes for λ, and we prove that unique solutions exist in the extended cias. We illustrate our results by the four concrete final coalgebras mentioned above, e. g., a finite stream circuit defines a unique stream function and we show how to define new process combinators from given ones by sos rules involving recursion.

Keywords: recursion, semantics, completely iterative algebra, coalgebra, distributive law.

1 Introduction

Recursive definitions are a useful tool to specify infinite system behavior. For example, Milner [21] proved that in his calculus CCS, one may specify a process uniquely by the equation $P = a.(P|c) + b$. More generally, such recursive equations have unique solutions whenever each recursion variable is in the scope of some action prefix. Another example is the shuffle product on streams of real numbers uniquely defined by $r.\sigma \otimes s.\tau = rs.(r.\sigma \otimes \tau + \sigma \otimes s.\tau)$. And as a third example consider non-well-founded sets [2, 10], a framework originating as a semantic basis for circular definitions. Here we can solve recursive function definitions such as $g(x) = \{g(\mathcal{P}(x)) \times x, x\}$ uniquely. It is the aim of this paper to develop abstract tools and results that explain why there exist unique solutions to all the aforementioned equations.

The key observation is that streams, non-well-founded sets and process behaviors constitute *final coalgebras* for certain functors on appropriate categories. Furthermore, the structure $c : C \to HC$ of a final coalgebra is an isomorphism [14], and the H-algebra (C, c^{-1}) is the initial *completely iterative algebra* (cia) for H [18]; cias are

* In this extended abstract all proofs are omitted. They can be found in the full version of our paper at www.stefan-milius.eu.

** Supported by the German Research Foundation (DFG) under the project MI 717/2–1.

L. Ong (Ed.): FOSSACS 2010, LNCS 6014, pp. 312–327, 2010.

algebras in which recursive (function) definitions involving the operations given by c^{-1} can be solved uniquely. However, cia structures for H are not sufficient to yield the existence and uniqueness of solutions in our motivating examples; these involve additional algebraic operations not captured by H. For example, $|, +$ in CCS, the stream addition $+$, and powerset and cartesian product $\mathcal{P}, \times$ in the example from non-well-founded set theory.

Additional algebraic operations are often presented by *distributive laws* in various guises. In process algebra one defines operations such as $|$ or $+$ by structural operational semantics (sos) [1]. Plotkin and Turi [22] showed how to capture sos rules as a distributive law of the functor (or monad) M describing the desired algebraic operations over the "behavior" functor H. This distributive law then induces an algebraic structure for M on the final H-coalgebra C. Other instances of distributive laws are behavioral differential equations in stream calculus, see [24], and definitions of operations on non-well-founded sets.

Bartels systematically studies definition formats giving rise to distributive laws in his thesis [9] (see also [8]) and shows how to solve parameter-free first order recursive equations involving operations presented by a distributive law.

After recalling his results in Section 2 we extend them in Section 3 by combining them with our previous work in [3, 18, 19]. We first prove (Theorems 3.2 and 3.3) that the final H-coalgebra carries the structure of a cia for HM and for MHM. These results show how to construct new structures of cias on C out of the initial one using a distributive law. This improves Bartels' result in the sense that first order recursive definitions may employ constant parameters in the final coalgebra. This also explains why recursively defined operations may be used in subsequent recursive definitions.

In Section 4 we obtain new ways to provide the semantics of recursive definitions by applying the existing solution theorems with the new cia structures, and in Section 5 we turn our attention to functional recursive definitions like the above shuffle product or the above function g on non-well-founded sets. We introduce for a distributive law λ the notion of a recursive program scheme (λ-rps, for short). Our main result is that any λ-rps has a unique solution in the final coalgebra C. Moreover, we show that these solutions extend the cia structure of C, which means that they can be used in subsequent recursive definitions. This compositionality of taking solutions of recursive equations does not appear in any previous work in this generality. In fact, we believe that our result is the first one that allows to obtain recursively defined operations directly as the unique solutions of their specifications.

Finally, in Section 6 we demonstrate the value of our results by instantiating them in four different concrete applications: (1) CCS-processes—we explain how Milner's solution theorem from [21] arises as a special case of Theorem 3.3, and we also show how to define new process combinators recursively from given ones; (2) streams of real numbers—here we prove that every finite stream circuit defines a unique stream function; (3) non-well-founded sets—we prove that operations on non-well-founded sets are uniquely determined as solutions of λ-rps's; (4) formal languages—here we show how operations on formal languages like union, concatenation, complement, etc. arise step-by-step using the compositionality of unique solutions of λ-rps's.

Related Work. The work in [22] was taken further by Lenisa, Power and Watanabe in [15, 16]. Jacobs [13] shows how to apply Bartels' result to obtain the (first order) solution theorems from [3, 18]. Capretta et al. [11] work in a dual setting and generalize the results of [8] beyond terminal coalgebras and they also obtain the (dual of) the solution theorem from [3, 18] by an application of their general results. Our Theorems 3.2 and 3.3 are similar to results in [11] but extend the ones of [8] in a different direction by considering parameters in recursive definitions. So our results in the present paper go beyond what can be accomplished with previous work. For example, while [16] gives an abstract explanation of adding operations to a process calculus, it gives no account of the kind of compositionality we have in our results.

2 Distributive Laws and Bialgebras

We shall assume some familiarity with basic notions from category theory such as functors, (initial) algebras and (final) coalgebras, monads, see e. g. [17, 23, 5].

Suppose we are given an endofunctor H on some category $\mathcal{A}$ describing the behavior type of a class of systems. In our work we shall be interested in additional algebraic operations on the final coalgebra C for H. The type of these algebraic operations is given by an endofunctor M on $\mathcal{A}$. Specification of algebraic operations by sos rules is abstractly captured by giving a distributive law of M over H, see e. g. [22, 8, 13]. Often M comes with the extra structure of a pointed endofunctor or a monad. Our goal is to provide a setting in which recursive equations involving the algebraic operations described by M can be uniquely solved.

Assumption 2.1. Throughout this section we assume that $H : \mathcal{A} \to \mathcal{A}$ is a functor, and that $c : C \to HC$ is a final coalgebra. In addition, we assume that M is a *pointed endofunctor* on $\mathcal{A}$, i. e., M comes equipped with a natural transformation $\eta : \mathrm{Id} \to M$.

Definition 2.2. *(1) An* algebra for (M, η) *is a pair* (A, a) *where* A *is an object of* $\mathcal{A}$ *and* $a : MA \to A$ *is a morphism satisfying the* unit law $a \cdot \eta_A = \mathrm{id}_A$.

(2) A distributive law *of* M *over* H *is a natural transformation* $\lambda : MH \to HM$ *such that we have*

$$H\eta = (\, H \xrightarrow{\eta H} MH \xrightarrow{\lambda} HM\,).$$

Remark 2.3. (1) In most concrete examples, M in Definition 2.2 is part of a monad (M, η, μ). Then any distributive law for the monad M over H is obviously also a distributive law for the pointed endofunctor M over H: in fact, recall that a distributive law of the monad M over H is a distributive law as in Definition 2.2(2) above additionally satisfying one obvious law concerning compatibility of λ and μ.

(2) Suppose that M is a free monad $\widehat{K}$ on an endofunctor $K : \mathcal{A} \to \mathcal{A}$ given objectwise by free K-algebras $\widehat{K}X$. Bartels [9] (Lemma 3.4.24) shows that distributive laws $\lambda : \widehat{K}H \to H\widehat{K}$ (of the monad $\widehat{K}$ over H) correspond to natural transformations $\ell : KH \to H\widehat{K}$. Indeed, given λ we get $\ell = \lambda \cdot \kappa H$, where $\kappa : K \to \widehat{K}$ is the universal natural transformation of the free monad. Conversely, let η and μ be the unit

and multiplication of the free monad $\widehat{K}$. Given ℓ, we see that $H\mu_X \cdot \ell_{\widehat{K}X} : KH\widehat{K}X \to H\widehat{K}\widehat{K}X \to H\widehat{K}X$ is an algebra for K. Thus, by the freeness of the algebra $\widehat{K}HX$, there exists a unique K-algebra homomorphism $\lambda_X : \widehat{K}HX \to H\widehat{K}X$ extending $H\eta_X$, i. e., such that $\lambda_X \cdot \eta_{HX} = H\eta_X$. One readily shows that λ is a distributive law of $\widehat{K}$ over H.

Construction 2.4. [8, 9] Let $\lambda : MH \to HM$ be a distributive law of the pointed endofunctor M over H. By using coinduction (i. e., the universal property of the final coalgebra) we define an M-algebra structure on C as the unique coalgebra homomorphism b from the coalgebra $\lambda_C \cdot Mc : MC \to HMC$ to the final coalgebra, i. e., $b : MC \to C$ is such that

$$c \cdot b = Hb \cdot \lambda_C \cdot Mc\,. \tag{2.1}$$

Bartels [8] showed that (C, b) is indeed an algebra for M. Moreover, if M is a monad and λ a distributive law of M over H, then (C, b) is an Eilenberg-Moore algebra for the monad M.

Definition 2.5. *For every distributive law λ we call $b : MC \to C$ from Construction 2.4 the* λ-interpretation *in C. In the case described in Remark 2.3(2), where λ is induced by ℓ, we shall also say that b is the* ℓ-interpretation *in C.*

Examples 2.6. We review a couple of examples of interest in this paper where $\mathcal{A} = \mathsf{Set}$. We shall elaborate these examples in Section 6.

(1) Formal languages. Consider the endofunctor $HX = X^A \times 2$, where $2 = \{0, 1\}$. Coalgebras for H are precisely the (possibly infinite) deterministic automata over the set A (as an alphabet). The final coalgebra $c : C \to HC$ consists of all formal languages with $c(L) = (\lambda a.D_a(L), i)$ with $i = 1$ iff the empty word ε is in L and where $D_a(L) = \{w \mid aw \in L\}$.

To specify e. g. the intersection of formal languages via a distributive law, let $KX = X \times X$ and let $\ell : KH \to HK$ be given by $\ell_X((f, i), (g, j)) = (\langle f, g\rangle, i \wedge j)$ where $\wedge$ denotes the "and"-operation on $\{0, 1\}$. Take $M = K + \mathrm{Id}$ and $\lambda = \mathsf{can} \cdot (\ell + \mathrm{id}_H) : MH \to HM$, where $\mathsf{can} = [H\mathsf{inl}, H\mathsf{inr}] : HK + H \to H(K + \mathrm{Id})$. Then the λ-interpretation $b : C \times C + C \to C$ has its left-hand component given by intersection of formal languages.

(2) Streams have been studied in a coalgebraic setting by Rutten [24]. Here we take the functor $HX = \mathbb{R} \times X$ whose final coalgebra (C, c) is carried by the set $\mathbb{R}^\omega$ of all streams over $\mathbb{R}$ and $c = \langle \mathsf{hd}, \mathsf{tl}\rangle : \mathbb{R}^\omega \to \mathbb{R} \times \mathbb{R}^\omega$ is given by the usual head and tail functions on streams.

Consider the functor $KX = \mathbb{R} \times X + X \times X$ corresponding to the signature Σ with a unary operation symbol $r.(-)$ for every real number r and with a binary operation symbol zip. Now zip is defined by the behavioral differential equation $\mathsf{zip}(r.x, s.y) = (r, \mathsf{zip}(s.y, x))$. This can conveniently be expressed as a natural transformation $\ell : KH \to H\widehat{K}$, where $\widehat{K}$ is the free monad on K (recall that $\widehat{K}X$ is the set of all Σ-terms on the set X of variables). Indeed, define ℓ for the two coproduct components of $KHX = \mathbb{R} \times HX + HX \times HX$ by $\ell_X(r, (s, x)) = (r, s.x)$ and $\ell_X((r, x), (s, y)) = (r, \mathsf{zip}(s.y, x))$, and note that the right-hand sides are pairs in $\mathbb{R} \times \widehat{K}X$. The ℓ-interpretation $b : \widehat{K}C \to C$ interprets Σ-terms built from the operation symbols as expected.

(3) Processes. We shall be interested in Milner's CCS [21]. Let κ be a regular cardinal and $\mathcal{P}_\kappa$ be the functor assigning to the set X the set of all subsets Y with $|Y| < \kappa$. Here we consider the functor $HX = \mathcal{P}_\kappa(A \times X)$ where A is some fixed alphabet of actions. Following Milner [21], we assume that for every $a \in A$ we also have a complement $\bar{a} \in A$ (so that $\bar{\bar{a}} = a$) and a special silent action $\tau \in A$.

Recall that the final coalgebra for the finite power set functor $\mathcal{P}_{\mathsf{fin}}$ was described by Worrell [27]: it is carried by the set of all strongly extensional finitely branching trees, where an unordered tree t is called *strongly extensional* if two subtrees rooted at distinct children of some node of t are never bisimilar. Similarly, the final coalgebra for the countable power set functor $\mathcal{P}_{\mathsf{c}}$ is carried by the set of all strongly extensional countably branching trees, see [25]. The technique by which this result is obtained in loc. cit. generalizes to the functor $\mathcal{P}_\kappa(A \times X)$ from above: its final coalgebra C turns out to consist of all strongly extensional κ-branching trees with edges labelled in A; strongly extensional has the analogous meaning as above: two subtrees rooted at distinct children of some node are never bisimilar if both edges to the children carry the same label. The elements of C can be considered as (denotations of) CCS-agents modulo strong bisimilarity.

Notice that the inverse $c^{-1} : \mathcal{P}_\kappa(A \times C) \to C$ assigns to a set $\{(a_i, E_i) \mid i < \kappa\}$ of pairs of actions and agents the agent $\sum_{i<\kappa} a_i.E_i$. The usual process combinators "$a.-$" (prefixing), "$|$" (composition), "$\sum_{i<\kappa}$" (sum), $-[f]$ (relabelling) and "$-\backslash L$" (restriction) are given by sos rules. Let E, E', F, F' be agents and $a \in A$ some action, then these rules are:

$$\frac{}{a.E \xrightarrow{a} E} \qquad \frac{E_j \xrightarrow{a} E_j'}{(\sum_{i<\kappa} E_i) \xrightarrow{a} E_j'}\ (j<\kappa) \qquad \frac{E \xrightarrow{a} E'}{E|F \xrightarrow{a} E'|F} \qquad \frac{F \xrightarrow{a} F'}{E|F \xrightarrow{a} E|F'} \qquad \frac{E \xrightarrow{a} E' \quad F \xrightarrow{\bar{a}} F'}{E|F \xrightarrow{\tau} E'|F'}\ (a \neq \tau)$$

$$\frac{E \xrightarrow{a} E'}{E[f] \xrightarrow{f(a)} E'[f]} \qquad \frac{E \xrightarrow{a} E'}{E\backslash L \xrightarrow{a} E'\backslash L}\ (a, \bar{a} \notin L)$$

Now let K be the polynomial functor for the signature given by taking the combinators as operation symbols. Then the above rules are easily seen to give a natural transformation $\ell : KH \to H\widehat{K}$, and the ℓ-interpretation $b : \widehat{K}C \to C$ in C evaluates all terms built from the considered combinators in C. Further details are presented in Section 6.1.

Our first result (Theorem 3.2) improves a result from [8, 9] that we now recall. We are interested in *λ-bialgebras* for a distributive law $\lambda : MH \to HM$. We shall not recall the formal definition here since it is not needed for this paper. We only note that for the λ-interpretation $b : MC \to C$ the triple (C, b, c) is a λ-bialgebra; in fact, it is the final one, see [9], Corollary 3.4.12).

Definition 2.7. *A* λ-equation *is an HM-coalgebra; that is, a morphism of the form $e : X \to HMX$. A* solution *of e in the λ-bialgebra (C, b, c) is a morphism $e^\dagger : X \to C$ such that the following equation holds: $c \cdot e^\dagger = Hb \cdot HMe^\dagger \cdot e$.*

Theorem 2.8. [8, 9] *Let $\mathcal{A}$ be a cocomplete category, and let λ be a distributive law of the pointed endofunctor M over H. Then every λ-equation has a unique solution in (C, b, c).*

Bartels proved a version of this theorem in the setting where the functor M is not necessarily pointed. So this result is just a slight variation on Bartels' Theorem 4.2.2.

It is worthwhile to note that Bartels' work also contains a similar result for the case where M is a monad and where λ is a distributive law of M over H, see [9], Corollary 4.3.6. In that case the assumption that $\mathcal{A}$ is cocomplete is not necessary. This last result is the dual of a result obtained independently and at the same time by Uustalu, Vene and Pardo (see [26], Theorem 1), and in [11] Capretta, Uustalu and Vene generalize this work further obtaining Theorem 2.8 as a special case.

3 Completely Iterative Algebras

In addition to our assumptions in 2.1 we assume from now on that our base category $\mathcal{A}$ is cocomplete.

It is our aim in this section to extend Theorem 2.8 so as to obtain several new structures of completely iterative algebras (for functors other than H) on C. We briefly recall the basic definitions and one example; more details and examples can be found in [18, 7, 19].

Definition 3.1. *[18] A* flat equation morphism *in an object A (of parameters) is a morphism $e : X \to HX + A$. An H-algebra $a : HA \to A$ is called* completely iterative *(or a* cia*, for short) if every flat equation morphism in A has a unique solution, i. e., for every $e : X \to HX + A$ there exists a unique morphism $e^\dagger : X \to A$ such that*

$$e^\dagger = (X \xrightarrow{e} HX + A \xrightarrow{He^\dagger + A} HA + A \xrightarrow{[a,A]} A).$$

Recall that the inverse of the structure $c : C \to HC$ of the final coalgebra is, equivalently, an initial cia for H, see [18]. The following show that the distributive law λ induces further structures of completely iterative algebras on C.

Theorem 3.2. *Consider the algebra $k = (HMC \xrightarrow{Hb} HC \xrightarrow{c^{-1}} C)$, where $b : MC \to C$ is the λ-interpretation in C. Then (C, k) is a cia for the composite functor HM.*

Theorem 3.3. (Sandwich Theorem) *Consider the algebra $k' = MHMC \xrightarrow{Mk} MC \xrightarrow{b} C$, where $b : MC \to C$ is the λ-interpretation in C and $k = c^{-1} \cdot Hb$ as in Theorem 3.2. Then (C, k') is a cia for the composite functor MHM.*

These two results extend Theorem 2.8 in two important ways. Firstly, the structure of a cia allows one to reuse solutions of given recursive specification by using constants in C on the right-hand sides of recursive equations, i. e., in a cia we can solve open recursive equations not just closed ones. This gives a clear explanation of why it is possible to use recursively defined objects (streams, processes, etc.) in subsequent recursive definitions. This kind of compositionality of the unique solutions is a useful and desired property often employed in specifications.

Secondly, Theorem 3.3 permits the right-hand sides of recursive specifications to be from a wider class. For example, Milner's solution theorem for CCS (see [21], Chapter 4, Proposition 14) allows recursion over process terms E in which the recursion variables occur within the scope of some prefixing combinator $a.-$. This combinator can occur *anywhere within* E, not necessarily at the head of that term. Hence, Theorem 3.3 allows us to obtain Milner's result as a special case, directly. This will be explained in detail in Section 6.1.

4 Solution Theorems for Free

Using the new cia structures obtained from Theorems 3.2 and 3.3, the existing body of results on the semantics of recursion in cias [3, 18, 19] now gives us further theorems.

We begin with a terse review of some terminology from the area [3, 18, 19]. We assume that, in addition to the final H-coalgebra C, for every object X the final coalgebra $T^H X$ for $H(-) + X$ exists, i. e., in the terminology of loc. cit., H is *iteratable*. Our examples in 2.6 are all iteratable endofunctors of Set.

The inverse of the final coalgebra structure $T^H X \to HT^H X + X$ gives morphisms $\eta^H_X : X \to T^H X$ and $\tau^H_X : HT^H X \to T^H X$. It is proved in [18] that $(T^H X, \tau^H_X)$ is a free cia on X with the universal arrow η^H_X. From this it easily follows that T^H is the object assignment of a monad and that η^H and τ^H are natural transformations. Denote by κ^H the natural transformation $\tau^H \cdot H\eta^H : H \to T^H$.

Let (A, a) be a cia for H. Then there is a unique homomorphism $\widetilde{a} : T^H A \to A$ of H-algebras such that $\widetilde{a} \cdot \eta^H_A = \mathrm{id}_A$. We call $\widetilde{a}$ the *evaluation morphism* associated to A. Notice that $\widetilde{a} \cdot \kappa^H_A = a$.

In our previous work we have shown how to obtain unique solutions of more general (first-order) recursive equations than the flat ones appearing in the definition of a cia:

Definition 4.1. [3, 18] *An* equation morphism *is a morphism of the form* $e : X \to T^H(X + A)$. *It is called* guarded *if there exists a factorization* $f : X \to HT^H(X + A) + A$ *such that* $e = [\tau^H_{X+A}, \eta^H_{X+A} \cdot \mathsf{inr}] \cdot f$.

A solution *of an equation morphism* e *in a cia* (A, a) *is a morphism* $e^\dagger : X \to A$ *such that the following equation holds:* $e^\dagger = \widetilde{a} \cdot T^H([e^\dagger, \mathrm{id}_A]) \cdot e$.

Theorem 4.2. [18] *Let* (A, a) *be a cia for* H. *Then every guarded equation morphism has a unique solution in* A.

An even more general property of cias was proved in [19]; one can solve recursive function definitions uniquely in a cia. We recall the respective result.

Definition 4.3. *Let* V *be an endofunctor such that* $H + V$ *is iteratable. A* recursive program scheme *(rps, for short) is a natural transformation* $e : V \to T^{H+V}$. *It is called* guarded *if there exists a natural transformation* $f : V \to HT^{H+V}$ *such that* $e = \tau^{H+V} \cdot (\mathsf{inl} T^{H+V}) \cdot f$, *where* $\mathsf{inl} : H \to H + V$.

Now let (A, a) *be a cia for* H. *An* interpreted solution *of* e *in* A *is a* V*-algebra structure* $e^\ddagger_A : VA \to A$ *giving rise to an Eilenberg-Moore algebra structure* $\beta : T^{H+V}A \to A$ *(i. e.,* $\beta \cdot \kappa^{H+V}_A = [a, e^\ddagger_A]$*) such that we have* $e^\ddagger_A = \beta \cdot e_A$.

Theorem 4.4. [19] *In a cia, every guarded rps has a unique interpreted solution.*

We are now able to prove more:

Theorem 4.5. *Let* $e : V \to T^{H+V}$ *be a guarded rps, and let* $a : HA \to A$ *be a cia. Then the interpreted solution* $e^\ddagger_A : VA \to A$ *extends the cia structure on* A*; more precisely, the algebra* $[a, e^\ddagger_A] : (H + V)A \to A$ *is a cia for* $H + V$.

The last result implies that operations obtained as solutions of recursive program schemes can be used in subsequent recursive function definitions, which will still have unique solutions. For the special case of interpreted rps solutions in cias this strengthens the results in [20].

Now assume that the composite HM is iteratable. By applying the above two Theorems 4.2 and 4.4 to the cia $k : HMC \to C$ from Theorem 3.2 we get two more solutions theorems for free, and two similar theorems hold for the cia $k' : MHMC \to C$ obtained from Theorem 3.3:

Corollary 4.6. *Every guarded equation morphism $e : X \to T^{HM}(X + C)$ has a unique solution in the cia (C, k).*

Corollary 4.7. *Every guarded rps $e : V \to T^{HM+V}$ has a unique interpreted solution in the cia (C, k).*

5 Recursive Function Definitions over the Behavior

All results we have seen so far do not allow us to obtain functions such as the shuffle product on streams (see introduction) as a unique solution since its definition refers to the behavior of the arguments of the function. In this section we introduce a special form of recursive program schemes called *λ-rps's* which accommodate such examples. We prove that every λ-rps has a unique solution in the final coalgebra C, and this solution extends the cia structure for HM on C given by Theorem 3.2—this is a compositionality result similar to the one given in Theorem 4.5 for ordinary rps.

We show that every λ-rps easily gives rise to a distributive law, and so the results in this section are essentially an application of the work in [8] and our results in Section 3. However, to prove the uniqueness of solutions we need that free algebras can always be constructed as colimits of transfinite chains. Hence, the following

Assumption 5.1. We assume that our base category $\mathcal{A}$ is Set, and in this section we only consider endofunctors on Set that are *accessible*, i. e., they preserve α-filtered colimits for some regular cardinal α. This implies that they are iteratable.

We further assume that the functor $M = \widehat{K}$ is the free monad on some endofunctor K and that $\lambda : \widehat{K}H \to H\widehat{K}$ arises from some natural transformation $\ell : KH \to H\widehat{K}$ as explained in Remark 2.3(2).

Notation 5.2. (1) Recall that for an accessible endofunctor $F : \mathsf{Set} \to \mathsf{Set}$ the free monad $\widehat{F}$ does indeed exist and is given objectwise by the free F-algebras on X, see [4]. We denote by $\varphi_X : F\widehat{F}X \to \widehat{F}X$ the structure of this algebra and by $u_X : X \to \widehat{F}X$ the universal morphism. We abuse notation and write φ_X and u_X for different functors F.
(2) For any F-algebra $a : FA \to A$ we denote the corresponding Eilenberg-Moore algebra for $\widehat{F}$ by $\widehat{a} : \widehat{F}A \to A$. Observe that $\widehat{a}$ is the unique F-algebra homomorphism with $\widehat{a} \cdot u_A = \mathrm{id}_A$.
(3) Let $b : \widehat{K}C \to C$ be the λ-interpretation in C. We denote by b_0 the corresponding K-algebra structure $b_0 = b \cdot \varphi_C \cdot Ku_C$. Then clearly we have $b = \widehat{b_0}$.

Definition 5.3. *A recursive program scheme w. r. t. λ (shortly, λ-*rps*) is a natural transformation $e : VH \to H\widehat{K+V}$.*

An interpreted solution *of e in C is a V-algebra structure $s : VC \to C$ such that*

$$s \cdot Vc^{-1} = c^{-1} \cdot H\widehat{[b_0, s]} \cdot e_C : VHC \to C.$$

Theorem 5.4. *For every λ-rps there exists a unique interpreted solution s in C. In addition, s extends the cia structure on C, i. e., the following is the structure of a cia for $H\widehat{K+V}$ on C:*

$$H\widehat{K+V}(C) \xrightarrow{H\widehat{[b_0,s]}} HC \xrightarrow{c^{-1}} C\,. \tag{5.1}$$

Notice again that the fact that the unique solution s of a λ-rps extends the cia structure on C means that the operations on C defined in this way may be part of recursive definitions according to the Corollaries 4.6, 4.7 (where $M = \widehat{K+V}$) and also Theorem 5.4 (where K is now replaced by $K+V$). We will make use of this feature in our applications below.

Theorem 5.5. *Let $e_i : V_iH \to H\widehat{K+V_i}$, $i = 1, 2$, be two λ-rps's. Then the cia structure on C extended by the unique solutions $s_i : V_iC \to C$ of the e_i is independent of the order of extension.*

More precisely, we may first take $s_1 : V_1C \to C$ to obtain an extended cia structure as in (5.1), and then take the solution of s_2 in the new cia, or vice versa. Either way, the resulting extended cia structure is $c^{-1} \cdot H\widehat{[b_0, s_1, s_2]} : H(K + \widehat{V_1 + V_2})(C) \to C$.

6 Applications

6.1 Process Algebras

Recall Example 2.6(3) where $HX = \mathcal{P}_\kappa(A \times X)$. We shall first explain more in detail how the natural transformation ℓ is obtained. Recall that K is the polynomial functor corresponding to the types of the CCS combinators, i. e., KX is a coproduct of the following components: $A \times X$ for agent expressions $a.x$, $\coprod_{n<\kappa} X^n$ for agent expressions $\sum_{i=1}^n x_i$, $X \times X$ for $x_1|x_2$, $\coprod_f X$, where f ranges over functions on the action set $A \setminus \{\tau\}$ with $\overline{f(a)} = f(\bar{a})$, for renaming $x[f]$, and $\coprod_{L \subseteq A\setminus\{\tau\}} X$, for restriction $x\backslash L$. The natural transformation $\ell : KH \to H\widehat{K}$ is given by the sos rules in 2.6(3) in terms of the components of the coproduct KH, i. e., for each combinator separately. The first component $\ell_X(a, S)$, for $S \subseteq A \times X$, is $(a, \sum_{(a_i,x_i)\in S}(a_i.x_i))$, the second one is for every $n < \kappa$ and every family $(S_i)_{i<n}$ with $S_i \subseteq A \times X$ given by $\ell_X((S_i)_{i<\kappa}) = \bigcup_{i<\kappa} S_i$. The third one $\ell_X(S_1, S_2)$, where $S_1, S_2 \in HX$, is the union of three sets: (i) all $(a, x|(\sum_{(a_i,x_i)\in S_2}(a_i.x_i)))$, where (a, x) ranges through S_1, (ii) all $(a, (\sum_{(a_i,x_i)\in S_1}(a_i.x_i))|x)$, where (a, x) ranges through S_2, and (iii) all $(\tau, x|y)$ where for some $a \in A \setminus \{\tau\}$ we have $(a, x) \in S_1$ and $(\bar{a}, y) \in S_2$. The remaining two components are $\ell_X(S) = \{(f(a), x[f]) \mid (a, x) \in S\}$ (here we mean $f(\tau) = \tau$, of course) and $\ell_X(S) = \{(a, x\backslash L) \mid (a, x) \in S, a, \bar{a} \notin L\}$. The form of these definitions is very

similar to the ones given by Aczel [2] in the setting of non-well-founded set theory. We already mentioned the ℓ-interpretation $b : \widehat{K}C \to C$ giving the desired operations on agents, and this gives the two new cia structures for $H\widehat{K}$ and $\widehat{K}H\widehat{K}$ as in Theorems 3.2 and 3.3.

Remark 6.1. If we replaced the second component $\coprod_{n<\kappa} X^n$ of K by $\mathcal{P}_\kappa X$ we still have a distributive law. Furthermore, in both cases the induced (binary) operation of summation is automatically commutative, associative and idempotent: these three laws that have to be proved in process theory come "for free" by encoding them in the distributive law using the union operation.

Now let us recall Milner's solution theorem for CCS agents from [21]. Suppose that E_i, $i \in I$, are agent expressions with the free variables x_i, $i \in I$. Suppose further that each variable x_j in each E_i, $i, j \in I$ is *weakly guarded*, i. e., it only occurs within the scope of some prefix combinator $a.-$. Then there is a unique solution of the recursive system $x_i = E_i$ of equations. More precisely, let $\sim$ denote strong bisimilarity, and let $E_i[\boldsymbol{P}/\boldsymbol{x}]$ denote simultaneous substitution of P_j for x_j for every j. Then we have

Theorem 6.2. [21] *There exist unique CCS agents P_i such that $P_i \sim E_i[\boldsymbol{P}/\boldsymbol{x}]$ holds for each $i \in I$.*

It is easy to see that this theorem is a consequence of our Theorem 3.3; to give a system $x_i = E_i$ where each variable is weakly guarded is the same as to give a map $X \to \widehat{K}H\widehat{K}X$, where $X = \{x_i \mid i \in I\}$. This map can be extended to a flat equation morphism $X \to \widehat{K}H\widehat{K}X + C$, which has a unique solution in C. Actually, the extra summand C allows us to use constant agents in recursive specifications. So, for example, we can obtain the agent P as the unique solution of $x = a.(x|c) + b$ in the introduction and then use it in a system like $x = b.(x + y), y = P$ which has a unique solution by Theorem 3.3.

Finally, suppose we want to define the new combinators op_1 and op_2 by the rule

$$\frac{E \xrightarrow{a} F}{\mathsf{op}_1(E) \xrightarrow{a} F|\mathsf{op}_2(F+E) \quad \mathsf{op}_2(E) \xrightarrow{a} F + \mathsf{op}_1(F|E)}.$$

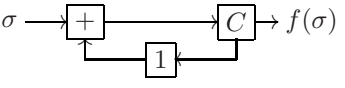
Then Theorem 5.4 tells us that this rule uniquely determines the two combinators. Indeed, we translate the rule into a λ-rps: let $V = \mathrm{Id} + \mathrm{Id}$ (two unary combinators are defined) and let $e : VH \to \widehat{HK + V}$ be given by $e(S) = \{(a, x|\mathsf{op}_2(x + \sum_{(a,x)\in S} a.x)) \mid (a, x) \in S\}$ on the first component and $e(S) = \{(a, x + \mathsf{op}_1(x|\sum_{(a,x)\in S} a.x)) \mid (a, x) \in S\}$ on the second one. The unique solution of e gives us two new unary combinators on C extending its cia structure. This means that Theorem 6.2 remains true for the extended calculus, without further work.

6.2 Streams

Recall from Example 2.6(2) that here we take $HX = \mathbb{R} \times X$ and we have $C = \mathbb{R}^\omega$ with the structure given by $\langle \mathsf{hd}, \mathsf{tl} \rangle : C \to \mathbb{R} \times C$. Recursive function definitions in this realm are often given in terms of stream circuits, and we show how this arises as a

special case of our results. Stream circuits are usually defined as pictorial compositions of the following basic stream circuits

The r-multiplier multiplies all elements in a stream by $r \in \mathbb{R}$, the adder performs componentwise addition, the copier yields two copies of a stream, and the register prepends $r \in \mathbb{R}$ to a stream σ to yield $r.\sigma$. The stream circuits are then built from the basic circuits by plugging wires together, and there may also be feedback (loops). For example the following picture shows a simple stream circuit:



(6.1)

Now let K be the signature functor associated to the signature Σ given by r-multiplication, the adder and the register operations (copying will be implicit via variable sharing). In symbols, $KX = \mathbb{R} \times X + X \times X + \mathbb{R} \times X$. These operations are defined by so-called *behavioral differential equations* [24] with $\sigma_0 = \mathsf{hd}(\sigma)$ and $\sigma' = \mathsf{tl}(\sigma)$:

$$\begin{array}{cc} (r\sigma)_0 = r\sigma_0 & (r\sigma)' = r\sigma' \\ (\sigma+\tau)_0 = \sigma_0 + \tau_0 & (\sigma+\tau)' = \sigma' + \tau' \\ (r.\sigma)_0 = r & (r.\sigma)' = \sigma \end{array}$$

These definitions are easily seen to give rise to a natural transformation $\ell : KH \to H\widehat{K}$, for example the middle component $(\mathbb{R} \times X)^2 \to \mathbb{R} \times \widehat{K}X$ is given by $\ell_X((r,x),(s,y)) = (r+s, x+y)$ where $r+s \in \mathbb{R}$ and $x+y$ is a Σ-term. We then get the ℓ-interpretation in C and the corresponding extended cia structures by Theorems 3.2 and 3.3. Recall that a stream circuit is called *valid* if every loop passes through at least one register. It is well-known that every *closed* valid stream circuit defines a unique stream at every output wire, see [24]. Our Theorem 5.4 now also implies that open circuits define unique stream functions. To our knowledge, this is a new result in coalgebraic stream calculus.

Theorem 6.3. *Every finite valid stream circuit defines a unique stream function.*

Moreover, the fact that the unique solution of a λ-rps extends the cia structure on C explains why stream circuits can be used as building blocks as if they were basic operations in subsequent stream circuits. And Theorem 6.3 remains valid for the extended circuits.

The proof of 6.3 essentially gives a translation of an arbitrary valid stream circuit into a λ-rps. Instead of giving the full proof here, we demonstrate this on the circuit given in (6.1) above. First we introduce for the output a function symbol f and for the register output the function symbol g. To determine their arity we count the number of input wires which have a (directed) path to the register and the output, respectively. In both cases the arity is one. Now we must give a definition of $f(r.x)$ and $g(r.x)$ for an abstract input stream with head $r \in \mathbb{R}$. These definitions are each given by a pair (s,t)

where $s \in \mathbb{R}$ and t is a term in the one variable x over operations corresponding to the basic circuits and f, g. We define

$$g(r.x) = (1, r.x + g(r.x)) \qquad f(r.x) = (r+1, x + (r.x + g(r.x)) \,.$$

For $g(r.x)$ we take the value 1 of the register as first component, and the right-hand term is obtained as follows: we follow all paths from the register backwards until we find an input or a register. Since the given circuit is valid, all such paths are finite and there are only finitely many of them. So we get a finite tree or, equivalently, the desired term. For $f(r.x)$ we first follow all paths to inputs and registers backwards to get the term $t' = x_I + x_R$, where x_I represents the input and x_R the register. For the first component of $f(r.x)$ we evaluate t' with the head r of the input and the initial value 1 of the register, and for the second component we replace in t' the input by x and the register by the second component of its function $g(r.x)$. The two equations above are easily seen to yield a λ-rps $e : VH \to H\widehat{K+V}$, where $V = \mathrm{Id} + \mathrm{Id}$ is the polynomial functor for the signature with two unary symbols f and g. The unique solution of e gives two unary operations (for f and g) on C, and the one for f is precisely the function computed by the circuit (6.1). Since these new unary operations on C extend the cia structure, we can use f (and also g) as "black-boxes" in subsequent recursive definitions or stream circuits.

6.3 Non-Well-Founded Sets

For background on non-well-founded sets, the antifoundation axiom (AFA), and classes, please see the books [2, 10]. We work here on the category $\mathcal{C}$ of classes. Observe first that even though we are working in a different category than Set, the results of Section 5 hold true for $\mathcal{C}$. This is because the construction of free algebras for a functor as colimits of transfinite chains works in $\mathcal{C}$, see [6].

Consider $\mathcal{P} : \mathcal{C} \to \mathcal{C}$ taking a class X to the class $\mathcal{P}X$ of sub*sets* of X. AFA is equivalent to the assertion that (V, c) is a final coalgebra, where V is the class of all sets, and $c : V \to \mathcal{P}V$ takes a set and considers it a set of sets. (That is, $c(s) = s$ for all s.) Let us note some natural transformations:

$$\begin{array}{lll} p : \mathcal{P} \to \mathcal{P}\mathcal{P} & op : \mathrm{Id} \times \mathrm{Id} \to \mathcal{P}\mathcal{P} & cp : \mathcal{P} \times \mathcal{P} \to \mathcal{P}(\mathrm{Id} \times \mathrm{Id}) \\ p_X(x) = \mathcal{P}(x) & op_X(x,y) = \{\{x\}, \{x,y\}\} & cp_X(x,y) = x \times y \end{array}$$

Also note that c^{-1} is the operation on V taking a family $x \subseteq V$ of sets to the set $\{\, y \mid y \in x \,\}$.

We will now define three additional operations on V: the powerset operation $b_1 : x \mapsto \{\, y \mid y \subseteq x \,\}$, the Kuratowski pair $b_2 : (x,y) \mapsto \{\{\, x \,\}, \{\, x, y\}\}$ and the cartesian product $b_3 : (x,y) \mapsto x \times y$. So let K be the functor $\mathrm{Id} + (\mathrm{Id} \times \mathrm{Id}) + (\mathrm{Id} \times \mathrm{Id}) + \mathcal{P} + \mathcal{P}^2$; its first three components represent (the type of) our three desired operations, the fourth component $\mathcal{P}$ represents c^{-1} and the fifth one represents $c^{-1} \cdot \mathcal{P}c^{-1}$—the latter two are needed for the definition of the former three. We write the coproduct injections of K as $\mathsf{inj}_1, \ldots, \mathsf{inj}_5$. We define a natural transformation $\ell : K\mathcal{P} \to \mathcal{P}K$ componentwise, using $\mathcal{P}\mathsf{inj}_4 \cdot p : \mathcal{P} \to \mathcal{P}K$, $\mathcal{P}\mathsf{inj}_5 \cdot op\mathcal{P} : \mathcal{P} \times \mathcal{P} \to \mathcal{P}K$, $\mathcal{P}\mathsf{inj}_2 \cdot cp : \mathcal{P} \times \mathcal{P} \to \mathcal{P}K$, $\mathcal{P}\mathsf{inj}_4 : \mathcal{P}\mathcal{P} \to \mathcal{P}K$ and $\mathcal{P}\mathsf{inj}_5 : \mathcal{P}\mathcal{P}\mathcal{P} \to \mathcal{P}K$. Then ℓ lifts to a distributive law $\lambda = \mathsf{can} \cdot (\ell + \mathrm{id}_{\mathcal{P}})$ of the free pointed endofunctor $M = K + \mathrm{Id}$ over $\mathcal{P}$. Let

$b : MV \to V$ be the λ-interpretation in V. Let us write $b_1, \dots, b_5$ for the components of b corresponding to the left-hand component K of M, so $b_i = b \cdot \mathsf{inl} \cdot (\mathsf{inj}_i)_V$. To obtain explicit formulas for these, we use equation (2.1) and the above definitions to write:

$$\begin{aligned} c \cdot b_1 &= \mathcal{P}b_4 \cdot p_V \cdot c & \qquad c \cdot b_4 &= \mathcal{P}b_4 \cdot \mathcal{P}c \\ c \cdot b_2 &= \mathcal{P}b_5 \cdot op_{\mathcal{P}V} \cdot (c \times c) & \qquad c \cdot b_5 &= \mathcal{P}b_5 \cdot \mathcal{P}^2 c \\ c \cdot b_3 &= \mathcal{P}b_2 \cdot cp_V \cdot (c \times c) \end{aligned}$$

We check easily that $b_4 = c^{-1}$ and $b_5 = c^{-1} \cdot \mathcal{P}c^{-1}$ satisfy the last two equations. From these we see that $b_1 = c^{-1} \cdot \mathcal{P}c^{-1} \cdot p_V \cdot c$, $b_2 = c^{-1} \cdot \mathcal{P}b_5 \cdot op_{\mathcal{P}V} \cdot (c \times c)$, and $b_3 = c^{-1} \cdot \mathcal{P}b_2 \cdot cp_V \cdot (c \times c)$. In words, b_4 and b_5 are the identity, and b_1, b_2 and b_3 are as desired.

By Theorem 3.2, we have a cia structure $(V, c^{-1} \cdot \mathcal{P}b)$ for the composite $\mathcal{P}M$.

Remark 6.4. We could have obtained the various operations on V in a step by step fashion starting with b_4 and b_5 and then defining b_1, b_2, b_3 by successive applications of Theorem 5.4. We decided against this, to keep the presentation short. But in the next section on formal languages we follow this approach.

Continuing our discussion of non-well-founded sets, we may solve systems of equations which go beyond what one finds in the standard literature on non-well-founded sets [2, 10]. For example, one may solve the system $x = \{\mathcal{P}(y)\}$, $y = \{y \times y, \emptyset\}$. Further, one may uniquely solve recursive function definitions such as $g(x) = \{g(\mathcal{P}(x)) \times x, x\}$ from the introduction. Indeed, for $W = \mathrm{Id}$ this equation yields a λ-rps $e : W\mathcal{P} \to \mathcal{P}\widehat{K + W}$ whose unique solution given by Theorem 5.4 is a function $g_V : V \to V$ behaving as specified.

6.4 Formal Languages

Recall Example 2.6(1); here we have $HX = X^A \times 2$ on Set. A coalgebra $x : X \to X^A \times 2$ for H is precisely a deterministic automaton with the (possibly infinite) state set X. Here $C = \mathcal{P}(A^*)$, and the unique homomorphism $h : (X, x) \to (C, c)$ assigns to each state the language it accepts. We shall now show how various operations on formal languages can be defined in a compositional way using Theorem 5.4. It is well-known that such operations can be defined as interpretations of one distributive law in C, see e. g. [12]. However, the previous bialgebraic account does not explain why one may define these operations in a step-by-step fashion by subsequent recursive definitions. This is the added value of Theorem 5.4.

We start with the functor $K_0 = C_\emptyset$ (that means, we start from scratch with no given operations) and with $\ell_0 : C_\emptyset H \to H\widehat{C_\emptyset} = H$ given by the empty maps. So the corresponding distributive law λ_0 is the identity on H, and its interpretation is the identity on C. Thus, the cia structure for $H\widehat{K_0}$ on C given by Theorem 3.2 is simply the initial cia (C, c^{-1}) for H. At each subsequent step we are given a functor K_i with $\ell_i : K_i H \to H\widehat{K_i}$ with its interpretation $b_i : \widehat{K_i}C \to C$. We then give a λ_i-rps $e_i : V_i H \to H\widehat{K_i + V_i}$ and its unique solution $s_i : V_i C \to C$ extends the cia structure as follows: let $K_{i+1} = K_i + V_i$ and let $\ell_{i+1} = [H\widehat{\mathsf{inl}} \cdot \ell_i, e_i] : K_{i+1}H \to H\widehat{K_{i+1}}$, where $\widehat{\mathsf{inl}} : \widehat{K_i} \to \widehat{K_{i+1}}$ is the monad morphism induced by $\mathsf{inl} : K_i \to K_{i+1}$. By induction

it is easy to see that the ℓ_{i+1}-interpretation is $b_{i+1} = \widehat{[s_j]_{j=0,\ldots,i}} : \widehat{K_{i+1}}C \to C$. And this gives an extended cia $c^{-1} \cdot Hb_{i+1} : H\widehat{K_{i+1}}(C) \to C$ by Theorem 3.2.

As a first step we define constants in C for $\emptyset$, $\{\varepsilon\}$, and $\{a\}$ for each $a \in A$, and the operation $I = c^{-1} : C^A \times 2 \to C$ as solutions of a λ_0-rps. (Note that $I((L_a)_{a\in A}, j)$ is the language $L = \bigcup_{a\in A}\{a\}L_a$ if $j = 0$ and $L \cup \{\varepsilon\}$ otherwise.) We express this as a λ_0-rps as follows: take the functor $V_0X = 1 + 1 + A + X^A \times 2$ expressing the above constants and the operation I. We define $e_0 : V_0H \to H\widehat{K_0 + V_0} = H\widehat{V_0}$ componentwise. We write for every set X, $\emptyset$ for $\mathsf{inj}_1(*) \in V_0X$, ε for $\mathsf{inj}_2(*) \in V_0X$. Then $e_0(\emptyset) = ((\emptyset)_{a\in A}, 0)$, $e_0(\varepsilon) = ((\emptyset)_{a\in A}, 1)$, $e_0(a) = ((t_b)_{b\in A}, 0)$ with $t_b = \varepsilon$ for $b = a$ and $t_b = \emptyset$ otherwise, and finally, $e_0((p_a)_{a\in A}, j) = ((Ip_a)_{a\in A}, j)$ where each $p_a \in X^A \times 2$. It is now straightforward to check that the unique solution s_0 of e_0 yields the desired operations on C extending the cia structure.

Next we add the operations of union, intersection and language complement to the cia structure. Let $K_1 = K_0 + V_0$ and let ℓ_1 as above with interpretation $b_1 = \widehat{s_0}$. Let $V_1X = X \times X + X \times X + X$ be the polynomial functor corresponding to two binary symbols $\cup$ and $\cap$ and one unary one $\overline{(-)}$. We give the λ_1-rps $e_1 : V_1H \to H\widehat{K_1 + V_1}$ componentwise in the form of the three assignments (where a ranges over A):

$$\begin{array}{ll} ((x_a), j) \cup ((y_a), k) \mapsto ((x_a \cup y_a), j \vee k) & \qquad \overline{((x_a), j)} \mapsto ((\overline{x_a}, \neg j) \\ ((x_a), j) \cap ((y_a), k) \mapsto ((x_a \cap y_a), j \wedge k) & \end{array}$$

where $\vee$, $\wedge$ and $\neg$ are the evident operations on $2 = \{0, 1\}$. The corresponding unique solution $s_1 : V_1C \to C$ is easily checked to provide the desired operations extending the cia structure on C.

The next step adds concatenation to the cia structure on C. For this let $V_2X = X \times X$ and e_2 is given by the assignment $((x_a), j) \cdot ((y_a), k) \mapsto ((t_a), j \wedge k)$ where $t_a = (x_a \cdot I((y_a), k)) \cup y_a$ if $j = 1$ and $t_a = x_a \cdot I((y_a), k)$ otherwise. Its unique solution $s_2 : C \times C \to C$ is the concatenation operation.

As the final step we add the Kleene star operation by taking $V_3X = X$ and e_3 given by $e_3((x_a), j) = ((x_a \cdot (I((x_a), j)^*), 1)$. Notice that this definition makes use of concatenation which was a solution at the previous stage and concatenation makes use of union which was a solution at stage 1.

7 Conclusions

In many areas of theoretical computer science, one is interested in recursive definitions of functions on final coalgebras C for various functors H. This paper provides a more comprehensive foundation for recursive definitions than had been presented up until now. The overall idea is to present operations in terms of a distributive law λ of a pointed endofunctor M over H. We proved that λ induces new completely iterative algebra structures for HM and MHM on C. As a result, we are able to define operations with useful algebraic properties such as commutativity or associativity "for free". We also introduced the notion of a λ-rps and showed how to uniquely solve recursive function definitions in C which are given by a λ-rps. Our results explain why taking unique

solutions of such equations is a compositional process. And we have seen that our results can be applied to provide the semantics of recursive specifications in a number of different areas of theoretical computer science.

References

[1] Aceto, L., Fokking, W., Verhoef, C.: Structural Operational Semantics. In: Handbook of Process Algebra. Elsevier, Amsterdam (2001)

[2] Aczel, P.: Non-Well-Founded Sets. CLSI Lecture Notes, vol. 14. CLSI Publications, Stanford (1988)

[3] Aczel, P., Adámek, J., Milius, S., Velebil, J.: Infinite trees and completely iterative theories: A coalgebraic view. Theoret. Comput. Sci. 300, 1–45 (2003)

[4] Adámek, J.: Free algebras and automata realizations in the language of categories. Comment. Math. Univ. Carolin. 15, 589–602 (1974)

[5] Adámek, J.: Introduction to coalgebra. Theory Appl. Categ. 14, 157–199 (2005)

[6] Adámek, J., Milius, S., Velebil, J.: On coalgebras based on classes. Theoret. Comput. Sci. 316, 3–23 (2004)

[7] Adámek, J., Milius, S., Velebil, J.: Elgot algebras. Log. Methods Comput. Sci. 2(5:4), 31 (2006)

[8] Bartels, F.: Generalized coinduction. Math. Structures Comput. Sci. 13(2), 321–348 (2003)

[9] Bartels, F.: On generalized coinduction and probabilistic specification formats. PhD thesis, Vrije Universiteit Amsterdam (2004)

[10] Barwise, J., Moss, L.S.: Vicious circles. CLSI Publications, Stanford (1996)

[11] Capretta, V., Uustalu, T., Vene, V.: Recursive coalgebras from comonads. Inform. and Comput. 204, 437–468 (2006)

[12] Jacobs, B.: A bialgebraic review of deterministic automata, regular expressions and languages. In: Futatsugi, K., Jouannaud, J.-P., Meseguer, J. (eds.) Algebra, Meaning, and Computation. LNCS, vol. 4060, pp. 375–404. Springer, Heidelberg (2006)

[13] Jacobs, B.: Distributive laws for the coinductive solution of recursive equations. Inform. and Comput. 204(4), 561–587 (2006)

[14] Lambek, J.: A fixpoint theorem for complete categories. Math. Z. 103, 151–161 (1968)

[15] Lenisa, M., Power, A.J., Watanabe, H.: Distributivity for endofunctors, pointed and copointed endofunctors, monads and comonads. In: Reichel, H. (ed.) Proc. Coalgebraic Methods in Computer Science. Electron. Notes Theor. Comput. Sci., vol. 33. Elsevier, Amsterdam (2000)

[16] Lenisa, M., Power, A.J., Watanabe, H.: Category theory for operational semantics. Theoret. Comput. Sci. 327, 135–154 (2004)

[17] MacLane, S.: Categories for the working mathematician, 2nd edn. Springer, Heidelberg (1998)

[18] Milius, S.: Completely iterative algebras and completely iterative monads. Inform. and Comput. 196, 1–41 (2005)

[19] Milius, S., Moss, L.S.: The category theoretic solution of recursive program schemes. Theoret. Comput. Sci. 366, 3–59 (2006) (fundamental study)

[20] Milius, S., Moss, L.S.: Equational properties of recursive program scheme solutions. Cah. Topol. Gèom. Diffèr. Catèg. 50, 23–66 (2009)

[21] Milner, R.: Communication and Concurrency. International Series in Computer Science. Prentice Hall, Englewood Cliffs (1989)

[22] Plotkin, G.D., Turi, D.: Towards a mathematical operational semantics. In: Proc. Logic in Computer Science, LICS (1997)

[23] Rutten, J.J.M.M.: Universal coalgebra: a theory of systems. Theoret. Comput. Sci. 249(1), 3–80 (2000)
[24] Rutten, J.J.M.M.: A coinductive calculus of streams. Math. Structures Comput. Sci. 15(1), 93–147 (2005)
[25] Schwencke, D.: Coequational logic for accessible functors. Accepted for publication in Inform. and Comput. (2009)
[26] Uustalu, T., Vene, V., Pardo, A.: Recursion schemes from comonads. Nordic J. Comput. 8(3), 366–390 (2001)
[27] Worrell, J.: On the final sequence of a finitary set functor. Theoret. Comput. Sci. 338, 184–199 (2005)