ABSTRACT GSOS RULES AND A MODULAR TREATMENT OF RECURSIVE DEFINITIONS

STEFAN MILIUS [a], LAWRENCE S. MOSS [b], AND DANIEL SCHWENCKE [c]

[a] Lehrstuhl für Theoretische Informatik, Friedrich-Alexander Universität Erlangen-Nürnberg, Germany
e-mail address: mail@stefan-milius.eu

[b] Department of Mathematics, Indiana University, Bloomington, IN, USA
e-mail address: lsm@cs.indiana.edu

[c] Institute of Transportation Systems, German Aerospace Center (DLR), Braunschweig, Germany
e-mail address: daniel.schwencke@dlr.de

ABSTRACT. Terminal coalgebras for a functor serve as semantic domains for state-based systems of various types. For example, behaviors of CCS processes, streams, infinite trees, formal languages and non-well-founded sets form terminal coalgebras. We present a uniform account of the semantics of recursive definitions in terminal coalgebras by combining two ideas: (1) *abstract GSOS rules* ℓ specify additional algebraic operations on a terminal coalgebra; (2) terminal coalgebras are also initial *completely iterative algebras* (cias). We also show that an abstract GSOS rule leads to new extended cia structures on the terminal coalgebra. Then we formalize recursive function definitions involving given operations specified by ℓ as recursive program schemes for ℓ, and we prove that unique solutions exist in the extended cias. From our results it follows that the solutions of recursive (function) definitions in terminal coalgebras may be used in subsequent recursive definitions which still have unique solutions. We call this principle *modularity*. We illustrate our results by the five concrete terminal coalgebras mentioned above, e. g., a finite stream circuit defines a unique stream function.

1. INTRODUCTION

Recursive definitions are a useful tool to specify infinite system behavior. For example, Milner [38] proved that in his calculus CCS, one may specify a process uniquely by the equation

$$P = a.(P|c) + b.0\,. \tag{1.1}$$

In other words, the process P is the unique solution of the recursive equation $x = a.(x|c) + b.0$. Another example is the shuffle product on streams of real numbers defined uniquely by a *behavioral differential equation* [42]:

$$(\sigma \otimes \tau)_0 = \sigma_0 \cdot \tau_0 \qquad (\sigma \otimes \tau)' = \sigma \otimes \tau' + \sigma' \otimes \tau. \qquad (1.2)$$

Here the real number σ_0 is the head of σ, and σ' is the tail; the operation $+$ is the componentwise addition of infinite streams. Besides these and other examples in the theory of computation, we shall mention below recursive specifications which are also important in other realms; we shall consider non-well-founded sets [3, 14], a framework originating as a semantic basis for circular definitions. Operations on non-well-founded sets can be specified uniquely by recursive function definitions. For example, we prove that the equation

$$x\|y = \{\, x\|y' \mid y' \in y \,\} \cup \{\, x'\|y \mid x' \in x \,\} \cup \{\, x'\|y' \mid x' \in x, y' \in y \,\} \qquad (1.3)$$

has a unique solution viz. a binary operation $\|$ on the class of all non-well-founded sets, which is reminiscent of parallel composition in process calculi. 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.

Terminal coalgebras. The key observation is that process behaviors, streams and non-well-founded sets constitute *terminal coalgebras* $(C, c : C \to HC)$ for certain endofunctors H on appropriate categories. The functor H describes the type of behavior of a class of state-based systems—the H-coalgebras—and the terminal coalgebra serves as the semantic domain for the behavior of states of systems of type H. All of the theory and examples in this paper pertain to terminal coalgebras.

Abstract GSOS rules. Let us return to the case of CCS operations to situate the work in a historical context. In the case of CCS one obtains algebraic operations on processes using structural operational semantics (sos) [2]. Operations are specified by operational rules such as

$$\frac{x \xrightarrow{a} x' \qquad y \xrightarrow{a} y'}{x|y \xrightarrow{a} x'|y'}$$

exhibiting the interplay between the operation and the behavior type. Syntactic restrictions on the rule format then ensure nice algebraic properties of the specified operations, e.g., GSOS rules [15] ensure that bisimilarity is a congruence. Turi and Plotkin gave in their seminal paper [39] a categorical formulation of GSOS rules, and they show how an abstract GSOS rule gives rise to a *distributive law* of a monad over a comonad, where the monad describes the signature of the desired algebraic operations and the comonad arises from the "behavior" functor H. Later Lenisa et al. [31] proved that abstract GSOS rules correspond precisely to distributive laws of a free monad M over the cofree copointed functor on the behavior functor H. In this paper we shall not need the original formulation of GSOS rules, and so the reader who is unfamiliar with this formulation will not be at a loss. What will be important are the categorical generalizations which we call abstract GSOS rules, and we begin with that topic in the next section. The theme of this paper is that abstract GSOS rules, either in a form close to the original one or in a more general one that we introduce, account for many interesting recursive definitions on terminal coalgebras in a uniform way. These include all of the examples we mentioned above, and many more.

Modularity. An important methodological goal in this paper is that our results should be *modular*. What we mean is that we want results which, given a class of algebras for endofunctors, enable us to expand the algebraic structure of an algebra from that class by a recursively defined operation, and stay inside the class. Thus the results will iterate.

We mentioned above that our results all had to do with terminal coalgebras, and so the reader might wonder how this idea of modularity could possibly apply. The answer is that a classical result due to Lambek [29] implies that the structure $c : C \to HC$ of a terminal coalgebra is an isomorphism. The corresponding inverse $c^{-1} : HC \to C$ turns C into an H-algebra. Moreover, we are typically interested to start with an algebraic structure on C that is already extended by additional operations. For example, in the case of the shuffle product $\sigma \otimes \tau$ of streams, the definition of $\otimes$ uses the stream addition operation $+$ on the right-hand side. So the definition is made on the algebra

$$(C, \quad (\sigma_0, \sigma') \mapsto \sigma, \quad \sigma, \tau \mapsto \sigma + \tau)$$

consisting of the set of streams of reals, its structure c^{-1} as an H-algebra, and its structure as an algebra for $X \times X$ given by componentwise addition. Returning to modularity, we aim to isolate an appropriateness condition on algebra structures expanding the inverse c^{-1} of the terminal coalgebra structure with the property that given an appropriate structure and a definition in a certain format, the definition specifies a new operation in a unique way, and if we add this operation to the given algebra structure, the resulting algebra structure on C is again appropriate. This is what we will call *modularity*, and precise formulations of appropriateness may be found in the Summaries 5.10 and 5.20.

Completely iterative algebras. The desired class of algebras mentioned in the previous paragraph is formed by completely iterative algebras (cias). Complete iterativity means that recursive equations involving algebraic operations corresponding to the H-algebra structure c^{-1} can be solved uniquely (see Definition 3.1). For example, let H_Σ be a polynomial endofunctor on Set arising from a signature of operation symbols with prescribed arity. In this case a cia is a Σ-algebra A in which every system of recursive equations

$$x_i = t_i \qquad (i \in I), \tag{1.4}$$

where t_i is either a single operation symbol applied to recursion variables (i. e., $t_i = \sigma(x_{i_1}, \ldots, x_{i_n})$ for an n-ary σ from Σ) or $t_i = a \in A$, has a unique solution. It can then be proved that more general equation systems involving (almost) arbitrary Σ-terms on the right-hand side and even recursive function definitions have unique solutions in a cia [33, 35]. Continuing, it was shown in [33] that the inverse $c^{-1} : HC \to C$ of the structure of the terminal coalgebra turns C into the initial cia for H. However, as we mentioned previously, cia structures for the "behavior" functor 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. These operations are $|$ and $+$ in the CCS process definition above, the stream addition $+$ in (1.2), and union $\cup$ in the example (1.3) from non-well-founded set theory.

Extended cia structures. We will show how the abstract GSOS rules of Turi and Plotkin extend the structure of an initial cia for H (alias terminal H-coalgebra). This then allows us to equip the initial cia with the desired additional operations such that circular definitions of elements of the carrier admit a unique solution, e. g., the equation $\sigma = 1.(\sigma + \sigma)$ (for

streams) defining the stream of powers of 2 or our first example (1.1) for CCS processes above.

The first steps in this direction were taken in Bartels' thesis [13] (see also [12]); he systematically studies definition formats giving rise to distributive laws and shows how to solve parameter-free first order recursive equations involving operations presented by a distributive law; Uustalu et al. [47] present the dual of this result.

We review some basic material on abstract GSOS rules and the solution theorem of Bartels in Section 2. In Section 3 we extend these solution theorems to equations with parameters, thereby combining them with our previous work on cias in [4, 33]. More concretely, given an abstract GSOS rule (equivalently, a distributive law of the free monad M over the cofree copointed functor $H \times \mathrm{Id}$) we prove in Theorems 3.5 and 3.6 that the terminal H-coalgebra carries the structure of a cia for HM and for MHM, respectively. These results show how to construct new structures of cias on C using an abstract GSOS rule. This improves Bartels' result in the sense that recursive equations may employ constant parameters in the terminal coalgebra.

In Section 4 we obtain new ways to provide the semantics of recursive definitions by applying the existing solution theorems from [4, 33, 35] to the new cia structures. For example, we consider solutions of *recursive program schemes*. Classical recursive program schemes [20] are function definitions such as

$$f(x) = F(x, G(f(x))) \tag{1.5}$$

defining a new function f recursively in terms of given ones F, G. In [35] is was shown how recursive program schemes can be formalized categorically. It was proved that every *guarded*[1] recursive program scheme has a unique (interpreted) solution in every cia $HA \to A$, where the functor H captures the signature of given operations. This solution is an algebra structure $VA \to A$, where V captures the signature of recursively defined operations. As a new result we now prove in Theorem 4.8 that the unique solution extends the structure of the given cia for H to a cia for the sum $H + V$; this yields modularity of unique solutions of recursive program schemes.

Extended formats of function definition. The recursive program schemes from [35] do not capture recursive function definitions like the one of the shuffle product in (1.2) above or the one for the operation ∥ from (1.3) above on non-well-founded sets. So in Section 5 we provide results that do capture those examples. We introduce for an abstract GSOS rule ℓ two formats of recursive program schemes w. r. t. ℓ: ℓ-rps and a variant called sandwiched ℓ-rps (ℓ-srps, for short). They give a categorical formulation of recursive function definitions such as (1.2) and (1.3) that refer not only to the operations provided by the behavior functor H but also to additional given operations specified by the abstract GSOS rule ℓ. We will also see that ℓ-srps's allow specifications which go beyond the format of abstract GSOS rules. In Theorems 5.14 and 5.23 we prove that every ℓ-(s)rps has a unique solution in the terminal coalgebra C. Moreover, we show that this solution extends the cia structure on C. This again gives rise to a modularity principle: operations defined uniquely by ℓ-(s)rps's can be used as givens in subsequent (sandwiched) rps's (we make this precise in the Summary 5.20). This modularity of taking solutions of recursive specifications of operations does not appear in any previous work in this generality.

---

[1] Guardedness is a mild syntactic restriction stating that terms on right-hand sides of equations do not have a newly defined function at their head.

A set of examples. Finally, in Section 6 we demonstrate the value of our results by instantiating them in five different concrete applications: (1) CCS-processes—we explain how Milner's solution theorem from [38] arises as a special case of Theorem 3.6, 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, we obtain the result from [42] that behavioral differential equations specify operations on streams in a unique way as a special instance of our Theorem 5.14, and we show how to solve recursive equations uniquely that cannot be captured by behavioral differential equations by applying Theorem 5.23; (3) infinite trees—we obtain the result from [45] that behavioral differential equations have unique solutions as a special case of Theorem 5.14; (4) formal languages—here we show how operations on formal languages like union, concatenation, complement, etc. arise step-by-step using the modularity of unique solutions of ℓ-rps's, and how languages generated by grammars arise as the unique solutions of flat equation morphisms in cias; (5) non-well-founded sets—we prove that operations on non-well-founded sets are uniquely determined as solutions of ℓ-(s)rps's.

Related Work. As already mentioned, Turi's and Plotkin's work [39] was taken further by Lenisa, Power and Watanabe in [30, 31]. Fiore and Turi [19] applied the mathematical operational semantics to provide semantics of message-passing process calculi such as Milner's π-calculus. But these papers do not consider the semantics of recursive definitions.

Turi [46] gives a treatment of guarded recursion. He does not isolate the notion of a solution of a recursive specification and whence does not prove that a solution exists uniquely. In addition that paper does not deal with recursive function definitions as we do here.

Jacobs [25] shows how to apply Bartels' result to obtain the (first order) solution theorems from [4, 33]. Capretta et al. [17] work in a dual setting and generalize the results of [12] beyond terminal coalgebras and they also obtain the (dual of) the solution theorem from [4, 33] by an application of their general results. Our Theorems 3.5 and 3.6 are similar to results in [17], but in the later sections we extend the work in [12] 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.

Modularity in mathematical operational semantics has been studied before in [40, 31]. These papers show how to combine two different specifications of operations over the same behavior by performing constructions at the level of distributive laws. This gives an abstract explanation of adding operations to a process calculus. At the heart of our proof of Theorem 5.14 lies a construction similar to the combination of two distributive laws that arises by taking the coproduct of the corresponding monads. However, while in the coproduct construction of [40, 31] the two distributive laws are independent of each other, our case is different because the operations specified by the second distributive law interact with the operations specified by the first one.

Much less related to our work is the work of Kick and Power [27] who show how to combine distributive laws of one monad over two sorts of behavior possibly interacting with one another.

To the best of our knowledge the results on modularity of solutions of recursive specifications we present are new.

The present paper is a completely revised version containing full proofs of the conference paper [37].

2. Abstract GSOS Rules and Distributive Laws

We shall assume some familiarity with basic notions from category theory such as functors, (initial) algebras and (terminal) coalgebras, and monads, see e. g. [32, 41, 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 terminal coalgebra C for H. The type of these algebraic operations is given by an endofunctor K on $\mathcal{A}$, and the algebraic operations are given by an *abstract GSOS rule* (cf. [39, 30, 31]). Our goal is to provide a setting in which recursive equations with operations specified by abstract GSOS rules have unique solutions. We now review the necessary preliminaries.

Assumption 2.1. Throughout the rest of this paper we assume that $\mathcal{A}$ is a category with binary products and coproducts, $H : \mathcal{A} \to \mathcal{A}$ is a functor, and that $c : C \to HC$ is its terminal coalgebra. We also assume that $K : \mathcal{A} \to \mathcal{A}$ is a functor such that for every object X of $\mathcal{A}$ there exists a free K-algebra MX on X.

Remark 2.2. (1) We denote by $\varphi_X : KMX \to MX$ and $\eta_X : X \to MX$ the structure and universal morphism of the free K-algebra MX. Recall that the corresponding universal property states that for every K-algebra $a : KA \to A$ and every morphism $f : X \to A$ there exists a unique K-algebra homomorphism $h : (MX, \varphi_X) \to (A, a)$ such that $h \cdot \eta_X = f$.

(2) Free algebras for the functor K exist under mild assumptions on K. For example, whenever K is an accessible endofunctor on Set it has all free algebras MX (see e. g. [9]).

(3) As proved by Barr [11] (see also Kelly [26]), the existence of free K-algebras as stated in Assumption 2.1 implies that there is a free monad on K. Indeed, M is the object assignment of a monad with the unit given by η_X from item (1) and the multiplication $\mu_X : MMX \to MX$ given as the unique homomorphic extension of id_{MX}. Also $\varphi : KM \to M$ is a natural transformation and

$$\kappa = (\,K \xrightarrow{K\eta} KM \xrightarrow{\varphi} M\,)$$

is the universal natural transformation of the free monad M. Notice that for complete categories all free monads arise from free algebras in this way.

Notation 2.3. For any K-algebra $a : KA \to A$, let $\widehat{a} : \widehat{K}A \to A$ be the unique K-algebra homomorphism with $\widehat{a} \cdot \eta_A = \mathrm{id}_A$. Then $\widehat{a}$ is the structure of an Eilenberg-Moore algebra for M. Notice also that

$$a = (\,KA \xrightarrow{\kappa_A} MA \xrightarrow{\widehat{a}} A\,). \tag{2.1}$$

Example 2.4. In our applications the functor K will be a polynomial functor on Set most of the time. In more detail, let $\mathcal{A} =$ Set and let $\Sigma = (\Sigma_n)_{n \in \mathbb{N}}$ be a signature of operation symbols with prescribed arity. The polynomial functor K_Σ associated with Σ is given by

$$K_\Sigma X = \coprod_{n \in \mathbb{N}} \Sigma_n \times X^n.$$

Algebras for the functor K_Σ are precisely the usual Σ-algebras and the free monad M_Σ assigns to a set X the Σ-algebra $M_\Sigma X$ of Σ-terms (or finite Σ-trees) on X, where a Σ-tree on X is a rooted and ordered tree with leaves labeled by constant symbols from Σ or elements of X and with inner nodes with n children labeled in Σ_n.

Definition 2.5. [39] An *abstract GSOS rule* is a natural transformation

$$\ell : K(H \times \mathrm{Id}) \to HM \,.$$

Remark 2.6. (1) The name "abstract GSOS" is motivated by the fact that for H the set functor $\mathcal{P}_{\mathsf{fin}}(A \times -)$ (whose coalgebras are labeled transition systems) and K a polynomial set functor, a natural transformation ℓ as in Definition 2.5 corresponds precisely to a transition system specification with operational rules in the GSOS format of Bloom, Istrail and Meyer [15]; this was proved by Turi and Plotkin [39].

(2) As proved by Lenisa et al. [30, 31], abstract GSOS rules are in one-to-one correspondence with distributive laws of the free monad M over the copointed functor $H \times \mathrm{Id}$. More precisely, recall from loc. cit. that a *copointed functor* (D, ε) is a functor $D : \mathcal{A} \to \mathcal{A}$ equipped with a natural transformation $\varepsilon : D \to \mathrm{Id}$. Also, notice that $H \times \mathrm{Id}$ together with the projection $\pi_1 : H \times \mathrm{Id} \to \mathrm{Id}$ is the cofree copointed functor on H. Further recall that a distributive law of a monad (M, η, μ) over a copointed functor (D, ε) is a natural transformation $\lambda : MD \to DM$ such that the following diagrams commute:

$$
\begin{array}{ccc}
\begin{array}{c}
\xymatrix{ & D & \\ MD & & DM }
\end{array}
&
\xymatrix{ MD \ar[r]^{\lambda} & DM \\ & M }
&
\xymatrix{ MMD \ar[r]^{M\lambda} \ar[d]_{\mu D} & MDM \ar[r]^{\lambda M} & DMM \ar[d]^{D\mu} \\ MD \ar[rr]_{\lambda} & & DM }
\end{array}
\qquad (2.2)
$$

Then to give an abstract GSOS rule $\ell : K(H \times \mathrm{Id}) \to HM$ is equivalent to giving a distributive law λ of the monad M over the copointed functor $H \times \mathrm{Id}$.

Theorem 2.7. [13, 39] *Let ℓ be an abstract GSOS rule. There is a unique structure $b : KC \to C$ of a K-algebra on the terminal H-coalgebra C such that the square below commutes:*

$$
\xymatrix{
KC \ar[r]^{K\langle c,\mathrm{id}_C\rangle} \ar[d]_{b} & K(HC \times C) \ar[r]^{\ell_C} & HMC \ar[d]^{H\widehat{b}} \\
C \ar[rr]_{c} & & HC
}
\qquad (2.3)
$$

Remark 2.8. (1) In the terminology of [13, 39] the triple (C, b, c) is a *model* of the abstract GSOS rule ℓ; in fact, it is the terminal one.

(2) For the distributive law $\lambda : M(H \times \mathrm{Id}) \to (H \times \mathrm{Id})M$ from Remark 2.6(2) corresponding to the abstract GSOS rule ℓ we have the following commutative diagram (see e. g. [13], Lemma 3.5.2, where this is formulated for $\mathcal{A} = \mathsf{Set}$):

$$
\xymatrix{
MC \ar[r]^{M\langle c,\mathrm{id}_C\rangle} \ar[d]_{\widehat{b}} & M(HC \times C) \ar[r]^{\lambda_C} & (H \times \mathrm{Id})MC \ar[d]^{(H\times\mathrm{Id})\widehat{b}} \\
C \ar[rr]_{\langle c,\mathrm{id}_C\rangle} & & HC \times C
}
\qquad (2.4)
$$

(3) In some of the examples below an abstract GSOS rule ℓ will arise from a natural transformation $\ell' : K(H \times \mathrm{Id}) \to HK$ as

$$\ell = (\ K(H \times \mathrm{Id}) \xrightarrow{\ell'} HK \xrightarrow{H\kappa} HM\ )\,.$$

In this case Diagram (2.3) can be simplified; indeed, $b : KC \to C$ is then the unique morphism such that the diagram below commutes:

$$
\begin{array}{ccc}
KC & \xrightarrow{K\langle c,\mathrm{id}_C\rangle} K(HC \times C) \xrightarrow{\ell'_C} & HKC \\
b \downarrow & & \downarrow Hb \\
C & \xrightarrow{ c } & HC
\end{array}
$$

Definition 2.9. For every abstract GSOS rule ℓ we will call the algebra structure $b : KC \to C$ from Theorem 2.7 the ℓ-*interpretation in* C.

Examples 2.10. We review a couple of examples of interest in this paper where $\mathcal{A} = \mathsf{Set}$. We shall elaborate on these examples in Section 6 and present two more examples.

(1) Processes. We shall be interested in Milner's CCS [38]. Let κ be an infinite cardinal and $\mathcal{P}_\kappa$ be the functor assigning to the set X the set of all $Y \subseteq X$ 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 [38], 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 $\tau \in A$.

To describe the terminal coalgebra for $\mathcal{P}_\kappa(A \times X)$ we first recall the description of the terminal coalgebra for the finite power set functor $\mathcal{P}_{\mathsf{fin}}$ by Worrell [49]: 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 as trees. Similarly, the terminal coalgebra for the countable power set functor $\mathcal{P}_{\mathsf{c}}$ is carried by the set of all strongly extensional countably branching trees, see [44]. The technique by which this result is obtained in loc. cit. generalizes to the functor $\mathcal{P}_\kappa(A \times X)$ from above: its terminal coalgebra C turns out to consist of all strongly extensional κ-branching trees with edges labeled in A; *strong extensionality* has the analogous meaning as above: two subtrees rooted at distinct children of some node are never bisimilar as trees 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), "$|$" (parallel composition), "$\sum_{i<\kappa}$" (summation), "$-[f]$" (relabeling) 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:

$$
\frac{E \xrightarrow{a} E'}{E|F \xrightarrow{a} E'|F} \qquad \frac{F \xrightarrow{a} F'}{E|F \xrightarrow{a} E|F'} \qquad \frac{E \xrightarrow{a} E' \quad F \xrightarrow{\bar{a}} F'}{E|F \xrightarrow{\tau} E'|F'}\ (a \neq \tau)
$$

$$
\frac{}{a.E \xrightarrow{a} E} \qquad \frac{E_j \xrightarrow{a} E'_j}{(\sum_{i<\kappa} E_i) \xrightarrow{a} E'_j}\ (j < \kappa) \qquad \frac{E \xrightarrow{a} E'}{E[f] \xrightarrow{f(a)} E'[f]} \qquad \frac{E \xrightarrow{a} E'}{E\backslash L \xrightarrow{a} E'\backslash L}\ (a, \bar{a} \notin L)
$$

Now let K be the polynomial functor for the signature given by taking these combinators as operation symbols. It easily follows from the work in [13] and [31] that the rules above give an abstract GSOS rule $\ell : K(H \times \mathrm{Id}) \to HM$, and the ℓ-interpretation $b : KC \to C$ in C provides the desired operations on CCS-agents (modulo strong bisimilarity). Further details are presented in Section 6.1.

(2) Streams. Streams have been studied in a coalgebraic setting by Rutten [42]. Here we take the functor $HX = \mathbb{R} \times X$ whose terminal 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.

Operations on streams can be defined by so-called *behavioral differential equations* [42]. Here one uses for every stream σ the notation $\sigma_0 = \mathsf{hd}(\sigma)$ and $\sigma' = \mathsf{tl}(\sigma)$. Then, for example, the function zip merging two streams is specified by

$$\mathsf{zip}(x, y)_0 = x_0 \qquad\qquad (\mathsf{zip}(x, y))' = \mathsf{zip}(y, x')\,.$$

This gives rise to an abstract GSOS rule as follows. Let $KX = X \times X$ (representing the binary operation zip), and let $\ell : K(H \times \mathrm{Id}) \to HM$ be

$$\ell = (\, K(H \times \mathrm{Id}) \xrightarrow{\ell'} HK \xrightarrow{H\kappa} HM \,)$$

where ℓ' is given by

$$\ell'_X((r, x', x), (s, y', y)) = (r, (y, x'))\,.$$

Notice that in a triple $(r, x', x) \in HX \times X$, x is a variable for a stream with head r and tail referred to by the variable x'. It is now straightforward to work out that the ℓ-interpretation $b : KC \to C$ is the operation zip.

For another example, the componentwise addition of two streams σ and τ is specified by

$$(\sigma + \tau)_0 = \sigma_0 + \tau_0 \qquad\qquad (\sigma + \tau)' = (\sigma' + \tau')\,. \tag{2.5}$$

In this case, we let $KX = X \times X$, and

$$\ell = (\, K(H \times \mathrm{Id}) \xrightarrow{\ell'} HK \xrightarrow{H\kappa} HM \,)\,,$$

where ℓ' is given by

$$\ell'_X((r, x', x), (s, y', y)) = (r + s, (x', y'))\,.$$

Again, it is not hard to show that the ℓ-interpretation $b : C \times C \to C$ is componentwise addition. (For related details, see Example 5.11 below.)

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

To specify e. g. the intersection of formal languages by an abstract GSOS rule, let $KX = X \times X$ and let $\ell : K(H \times \mathrm{Id}) \to HM$ be

$$\ell = (\, K(H \times \mathrm{Id}) \xrightarrow{K\pi_0} KH \xrightarrow{\ell'} HK \xrightarrow{H\kappa} HM \,)$$

where $\ell' : KH \to HK$ is given by $\ell'_X((f, i), (g, j)) = (\langle f, g \rangle, i \wedge j)$ where $\wedge$ denotes the "and"-operation on $\{0, 1\}$. Then the ℓ-interpretation is easily verified to be the intersection of formal languages.

Our first result (Theorem 2.16) improves a result from [12, 13] that we now recall. For the rest of this section we assume that an abstract GSOS rule $\ell : K(H \times \mathrm{Id}) \to HM$ is given. Recall that the structure of the terminal coalgebra $c : C \to HC$ is an isomorphism by the Lambek Lemma [29]. From now on we will regard C as an H-algebra with the structure $c^{-1} : HC \cdot C$ most of the time.

Definition 2.11. An ℓ-*equation* is an HM-coalgebra; that is, a morphism of the form

$$e : X \to HMX.$$

A *solution* of e in the terminal coalgebra C is a morphism $e^\dagger : X \to C$ such that the diagram below commutes:

$$
\begin{array}{ccc}
X & \xrightarrow{\ e^\dagger\ } & C \\
\downarrow e & & \uparrow c^{-1} \\
 & & HC \\
 & & \uparrow H\widehat{b} \\
HMX & \xrightarrow{\ HMe^\dagger\ } & HMC
\end{array}
\qquad (2.6)
$$

Theorem 2.12. [12, 13] *For every ℓ-equation there exists a unique solution in C.*

This result follows from Corollaries 4.3.6 and 4.3.8 and Lemma 4.3.9 in [13]. The first of these results is the dual of a result obtained independently and at the same time by Uustalu, Vene and Pardo (see [47], Theorem 1). Notice that both [13] and [47] work at the level of generality provided by distributive laws rather than with (the dual of) abstract GSOS rules. And Capretta, Uustalu and Vene [17] generalize this further. In their Theorems 19 and 28 they replace the inverse c^{-1} of the terminal coalgebra by an algebra $a : HA \to A$ having the property that for every coalgebra $e : X \to HX$ there exists a unique coalgebra-to-algebra homomorphism $h : X \to A$, i.e., h exists uniquely such that $a \cdot Hh \cdot e = h$. Our generalization in this paper goes in a different direction. We keep the algebra $c^{-1} : HC \to C$ but generalize the format of equations having a unique solution in this algebra. But before we do this let us give an example of an ℓ-equation and its solutions for streams.

Examples 2.13. As in our discussion of streams in Example 2.10(2), let $HX = \mathbb{R} \times X$ so that that C is the set of streams over $\mathbb{R}$, let $KX = X \times X$, and let ℓ be the abstract GSOS rule for zipping streams so that $b : KC \to C$ is the operation of zipping. As an example of an ℓ-equation, let $X = \{x, y\}$, and let $e : X \to \mathbb{R} \times MX$ be given by

$$e(x) = (0, \mathsf{zip}(y, x)) \qquad \text{and} \qquad e(y) = (1, \mathsf{zip}(x, y)).$$

Then the streams $t = e^\dagger(x)$ and $u = e^\dagger(y)$ satisfy:

$$
\begin{aligned}
t &= 1.\mathsf{zip}(u, t) \\
u &= 0.\mathsf{zip}(t, u)
\end{aligned}
\qquad (2.7)
$$

To continue the discussion from streams just above, we shall solve equations such as (2.9) below which are more complicated than (2.7). To start with, for the proof of Theorem 3.6 we shall need a variant of Theorem 2.12 for equations of the form $e : X \to MHMX$:

Definition 2.14. A *sandwiched ℓ-equation* is a MHM-coalgebra; that is, a morphism of the form $e : X \to MHMX$. A *solution* of e in the terminal coalgebra C is a morphism

$e^\dagger : X \to C$ such that the diagram below commutes:

$$
\begin{array}{ccc}
X & \xrightarrow{\ e^\dagger\ } & C \\
 & & \uparrow{\scriptstyle \widehat{b}} \\
 & & MC \\
{\scriptstyle e}\Big\downarrow & & \uparrow{\scriptstyle Mc^{-1}} \\
 & & MHC \\
 & & \uparrow{\scriptstyle MH\widehat{b}} \\
MHMX & \xrightarrow{\ MHMe^\dagger\ } & MHMC
\end{array}
\tag{2.8}
$$

Examples 2.15. This is a variation on Example 2.13. Using a sandwiched ℓ-equation, we can solve

$$
\begin{aligned}
t &=& \mathsf{zip}(1.u, 0.t) \\
u &=& \mathsf{zip}(0.t, 1.u)
\end{aligned}
\tag{2.9}
$$

Note the difference between (2.7) and (2.9). The key point about a sandwiched system is that the "guards" by prefix operations $r.-$ need not occur at the head of the term on the right-hand sides. (In this example, they are applied to the variables inside zip.) Incidentally, the solution assigns to u the famous Thue-Morse stream and to t its dual; the solutions to (2.7) are different. A more complicated sandwiched system would be

$$
\begin{aligned}
t &=& \mathsf{zip}(0.u, \mathsf{zip}(1.u, 0.\mathsf{zip}(1.u, 0.u))) \\
u &=& \mathsf{zip}(\mathsf{zip}(0.t, 1.u), \mathsf{zip}(0.t, 1.u)).
\end{aligned}
$$

Theorem 2.16. *For every sandwiched ℓ-equation there exists a unique solution in C.*

Proof. Given a sandwiched ℓ-equation $e : X \to MHMX$, we form the following (ordinary) ℓ-equation:

$$
\overline{e} = (\ HMX \xrightarrow{\ HMe\ } HMMHMX \xrightarrow{\ H\mu_{HMX}\ } HMHMX\ ).
$$

From Theorem 2.12 we know that $\overline{e}$ has a unique solution $\overline{e}^\dagger : HMX \to C$. Thus, we are finished if we can show that solutions of e and $\overline{e}$ are in one-to-one correspondence.

Firstly, from the solution $\overline{e}^\dagger$ of $\overline{e}$ we obtain

$$
e^\dagger = (\ X \xrightarrow{\ e\ } MHMX \xrightarrow{\ M\overline{e}^\dagger\ } MC \xrightarrow{\ \widehat{b}\ } C\ ),
$$

and we will now verify that $e^\dagger$ is a solution of e. To this end, consider the diagram below:

All its inner parts commute: parts (i) and (iii) commute by the definition of $e^\dagger$, for part (ii) use that $\overline{e}^\dagger$ is a solution of $\overline{e}$ (i.e. apply M to Diagram (2.6) with $\overline{e}$ in lieu of e), and the remaining parts commute by the definition of $\overline{e}$, naturality of μ and the multiplication law for the Eilenberg-Moore algebra $\widehat{b}$. Thus, the outside commutes, proving $e^\dagger$ to be a solution of e.

Secondly, suppose we are given any solution $e^\dagger$ of e. Then we form

$$HMX \xrightarrow{HMe^\dagger} HMC \xrightarrow{H\widehat{b}} HC \xrightarrow{c^{-1}} C \,,$$

and we now prove that this is a solution of $\overline{e}$. Indeed, in the diagram

all inner parts commute: for the big left-hand square apply HM to Diagram (2.8), the left-hand part is the definition of $\overline{e}$, the two lower squares and the lower right-hand triangle commute due to naturality of μ, the upper right-hand triangle is trivial, and the remaining middle right-hand part commutes by the multiplication law for the Eilenberg-Moore algebra $\widehat{b}$. Thus, the outside commutes proving $c^{-1} \cdot H\widehat{b} \cdot HMe^\dagger$ to be a solution of $\overline{e}$. Since $\overline{e}$ has a unique solution, we have

$$c^{-1} \cdot H\widehat{b} \cdot HMe^\dagger = \overline{e}^\dagger \,.$$

Lastly, the two constructions are inverse to each other: starting with the unique solution $\overline{e}^\dagger$ of $\overline{e}$, it is clear that by applying the two constructions we obtain $\overline{e}^\dagger$ again. Starting with any solution $e^\dagger$ of e, the application of the second construction results in the solution

$c^{-1} \cdot H\widehat{b} \cdot HMe^{\dagger} = \overline{e}^{\dagger}$ of $\overline{e}$. The application of the first construction to that solution gives back the solution $e^{\dagger}$ of e:

$$\widehat{b} \cdot M\overline{e}^{\dagger} \cdot e = \widehat{b} \cdot M(c^{-1} \cdot H\widehat{b} \cdot HMe^{\dagger}) \cdot e = \widehat{b} \cdot Mc^{-1} \cdot MH\widehat{b} \cdot MHMe^{\dagger} \cdot e = e^{\dagger}$$

where the last equality uses Diagram (2.8). We conclude that e has a unique solution $e^{\dagger}$. $\square$

3. Completely Iterative Algebras

We already mentioned in the introduction that the H-algebra $c^{-1} : HC \to C$ is the initial completely iterative algebra for H. After recalling this results below, it is our aim in this section to extend Theorems 2.12 and 2.16 so as to obtain several new structures of completely iterative algebras (for functors other than H) on C. Our first main results in this paper, Theorems 3.5 and 3.6, go in this direction. We shall see in the next section that having these new cia structures allows us to apply the existing theorems on completely iterative algebras to uniquely solve much more general recursive equations than what we have seen up to now and including this section.

We now briefly recall the basic definitions and some examples; more details and examples can be found in [33, 7, 35].

Definition 3.1. [33] 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 the square below commutes:

$$
\begin{array}{ccc}
X & \xrightarrow{\ e^{\dagger}\ } & A \\
{\scriptstyle e}\downarrow & & \uparrow{\scriptstyle [a,A]} \\
HX + A & \xrightarrow[He^{\dagger}+A]{} & HA + A
\end{array}
$$

Examples 3.2. We recall some examples from previous work.

(1) Let TX denote a terminal coalgebra for $H(-) + X$. (We assume that TX exists for all X.) Its structure is an isomorphism by Lambek's Lemma [29], and so its inverse yields (by composing with the coproduct injections) an H-algebra $\tau_X : HTX \to TX$ and a morphism $\eta_X : X \to TX$. Then (TX, τ_X) is a free cia on X with the universal arrow η_X, see [33]. Conversely, for any free cia (TX, τ_X) with the universal morphism η_X, $[\tau_X, \eta_X]$ is an isomorphism and $[\tau_X, \eta_X]^{-1}$ is the structure of a terminal coalgebra for $H(-) + X$. So in particular, the inverse of the structure $c : C \to HC$ of the terminal coalgebra for H is, equivalently, an initial cia for H.

(2) Let H_{Σ} be a polynomial functor (cf. Example 2.4). The terminal coalgebra for $H_{\Sigma}(-)+X$ is carried by the set $T_{\Sigma}X$ of all (finite and infinite) Σ-trees on X. According to the previous item, this is a free cia for H_{Σ} on X. As already mentioned in the introduction, cias for H_{Σ} are Σ-algebras in which systems of recursive equations (1.4) have unique solutions. For example, let Σ be the signature with one binary operation symbol $*$ and one constant symbol c. Consider the free cia $A = T_{\Sigma}Y$ and let $t \in T_{\Sigma}Y$. Then the system

$$
\begin{array}{rclcrcl}
x_0 & = & x_1 * x_2 & \qquad & x_2 & = & c \\
x_1 & = & x_0 * x_3 & \qquad & x_3 & = & t
\end{array}
$$

has the unique solution $e^\dagger : X \to T_\Sigma Y$ given by

$$e^\dagger(x_0) = \begin{array}{c} * \\ \diagup \diagdown \\ * \quad c \\ \diagup \diagdown \\ \vdots \quad t \end{array} \qquad e^\dagger(x_1) = \begin{array}{c} * \\ \diagup \diagdown \\ * \quad t \\ \diagup \diagdown \\ \vdots \quad c \end{array} \qquad e^\dagger(x_2) = c \qquad e^\dagger(x_3) = t\,.$$

(3) The algebra of addition on $\bar{\mathbb{N}} = \{1, 2, 3, \dots\} \cup \{\infty\}$ is a cia for $HX = X \times X$, see [8].

(4) Let $\mathcal{A} = \mathsf{CMS}$ be the category of complete metric spaces with distances in $[0, 1]$ and with non-expanding maps as morphisms, and let H be a *contracting* endofunctor of CMS (see e. g. [10]). Then any non-empty algebra for H is a cia, see [33] for details. For example, let A be the set of non-empty compact subsets of the unit interval $[0, 1]$ equipped with the Hausdorff metric [22]. This complete metric space can be turned into a cia such that the Cantor set arises as the unique solution of a flat equation morphism (see [35, Example 3.3(v)]).

(5) Unary algebras over Set. Here we take $\mathcal{A} = \mathsf{Set}$ and $H = \mathrm{Id}$. An algebra $\alpha : A \to A$ is a cia iff α has a fixed point a_0 and there is no infinite sequence $a_1, a_2, a_3, \dots$ with $a_i = \alpha(a_{i+1})$, $i = 1, 2, 3, \dots$, except for the one all of whose members are a_0. The second part of this condition can be put more vividly as follows: the graph with node set $A \setminus \{a_0\}$ and with an edge from $\alpha(a) \neq a_0$ to a for all a is well-founded.

(6) Classical algebras are seldom cias. For example a group or a semilattice is a cia (for $HX = X \times X$) iff they contain one element only (consider the unique solutions of $x = x \cdot 1$ or $x = x \vee x$, respectively).

Remark 3.3. In [7] the following property—called *compositionality*—of taking unique solutions of flat equations in a cia $a : HA \to A$ was proved. Suppose we have two flat equation morphisms $e : X \to HX + Y$ and $f : Y \to HY + A$. We form

$$f^\dagger \bullet e = (\, X \xrightarrow{\;e\;} HX + Y \xrightarrow{\;HX + f^\dagger\;} HX + A \,)$$

and

$$f \blacksquare e = (\, X + Y \xrightarrow{\;[e,\mathsf{inr}]\;} HX + Y \xrightarrow{\;HX + f\;} HX + HY + A \xrightarrow{\;\mathsf{can} + A\;} H(X + Y) + A \,)\,.$$

Then

$$(f \blacksquare e)^\dagger = (\, X + Y \xrightarrow{\;[(f^\dagger \bullet e)^\dagger, f^\dagger]\;} A \,)\,.$$

This gives a first precise formulation of the modularity principle we mentioned in the introduction, albeit restricted to solutions of flat equation morphisms. Indeed, the above equation states that in order to obtain the simultaneous unique solution of e and f (i. e. $(f \blacksquare e)^\dagger$) one may first solve f in the cia A, then plug its solution $f^\dagger$ as constant parameters into e and finally solve the resulting equation (i. e., one takes $(f^\dagger \bullet e)^\dagger$). For $\mathcal{A} = \mathsf{Set}$ one can view this as using the elements $f^\dagger(y)$, $y \in Y$, as new constants in A in the subsequent recursive equation given by e. We shall see modularity principles for more general formats of recursive definitions in Sections 4 and 5.

The following two theorems show that abstract GSOS rules induce further structures of completely iterative algebras on the (carrier of the) terminal H-coalgebra C besides the structure of an initial cia for H.

Assumption 3.4. As in the previous section, we shall write (M, η, μ) for the free monad on K to simplify notation, and we assume that $\ell : K(H \times \mathrm{Id}) \to HM$ is an abstract GSOS rule.

Theorem 3.5. *Consider the algebra*

$$k = (\, HMC \xrightarrow{H\widehat{b}} HC \xrightarrow{c^{-1}} C \,),$$

where $b : KC \to C$ is the ℓ-interpretation in C. Then (C, k) is a cia for the functor HM.

Proof. Let $e : X \to HMX + C$ be a flat equation morphism. We must prove that there exists a unique morphism $e^\dagger : X \to C$ such that the following square commutes:

$$
\begin{array}{ccc}
X & \xrightarrow{\;\;e^\dagger\;\;} & C \\
{\scriptstyle e}\downarrow & & \uparrow{\scriptstyle [k,C]} \\
HMX + C & \xrightarrow[HMe^\dagger+C]{} & HMC + C
\end{array}
$$

We start by forming the ℓ-equation

$$\overline{e} = (\, X + C \xrightarrow{[e,\mathsf{inr}]} HMX + C \xrightarrow{HMX + H\eta_C \cdot c} HMX + HMC \xrightarrow{\mathsf{can}} HM(X + C) \,).$$

By Theorem 2.12, there exists a unique morphism $s : X + C \to C$ such that the square below commutes:

$$
\begin{array}{ccc}
X + C & \xrightarrow{\;\;s\;\;} & C \\
& & \downarrow{\scriptstyle c} \\
{\scriptstyle \overline{e}}\downarrow & & HC \\
& & \uparrow{\scriptstyle H\widehat{b}} \\
HM(X + C) & \xrightarrow[HMs]{} & HMC
\end{array}
\tag{3.1}
$$

We will now prove that the morphism $e^\dagger = s \cdot \mathsf{inl} : X \to C$ is the desired unique solution of e. We begin by proving that the equation $s \cdot \mathsf{inr} = \mathrm{id}_C$ holds. Indeed, consider the diagram below:

$$
\begin{array}{ccccc}
C & \xrightarrow{\;\;\mathsf{inr}\;\;} & X + C & \xrightarrow{\;\;s\;\;} & C \\
{\scriptstyle c}\downarrow & & \downarrow{\scriptstyle \overline{e}} & & \uparrow{\scriptstyle k} \\
& HMC \xrightarrow{HM\mathsf{inr}} & HM(X + C) & \xrightarrow{HMs} & HMC \\
& {\scriptstyle H\eta_C}\nearrow & & & \uparrow{\scriptstyle H\eta_C} \\
HC & \xrightarrow[H(s\cdot\mathsf{inr})]{} & & & HC
\end{array}
\tag{3.2}
$$

This diagram commutes: the upper right-hand square is Diagram (3.1) above, the upper left-hand part commutes by the definition of $\overline{e}$, the lower part commutes by the naturality of η, and the right-hand part follows from the definition of $k = c^{-1} \cdot H\widehat{b}$ and the unit law $\widehat{b} \cdot \eta_C = \mathrm{id}_C$ of the Eilenberg-Moore algebra $(C, \widehat{b})$, cf. Notation 2.3. Hence, we see that $s \cdot \mathsf{inr}$ is a coalgebra homomorphism from the terminal coalgebra (C, c) to itself. Thus, $s \cdot \mathsf{inr}$ must be the identity as desired.

Next we prove that $e^\dagger$ is a solution of e. To this end we verify that the following diagram commutes:

$$ (3.3) $$

The upper part is the definition of $e^\dagger$, the left-hand part commutes by the definition of $\overline{e}$, the upper right-hand part is Diagram (3.1), and that the inner triangle commutes follows from the definition of $e^\dagger$ and the fact that $s \cdot \mathsf{inr} = \mathrm{id}_C$. Finally, we consider the two coproduct components of the lower right-hand triangle separately; the left-hand component trivially commutes, and for the right-hand one we compute as follows:

$$
\begin{aligned}
k \cdot H\eta_C \cdot c &= c^{-1} \cdot H\widehat{b} \cdot H\eta_C \cdot c && \text{(by the definition of } k) \\
&= c^{-1} \cdot c && \text{(since } \widehat{b} \cdot \eta_C = \mathrm{id}_C) \\
&= \mathrm{id}_C .
\end{aligned}
$$

To complete our proof we show that $e^\dagger = s \cdot \mathsf{inl} : X \to C$ is the unique solution of e. So suppose that we are given any solution $e^\dagger$ of e. Now form the morphism $s = [e^\dagger, \mathrm{id}_C]$. We are finished if we show that Diagram (3.1) commutes for this morphism s. We verify the two coproduct components separately: the right-hand component is checked using Diagram (3.2): since $s \cdot \mathsf{inr} = \mathrm{id}_C$ the outside of (3.2) commutes, and commutativity of the desired upper right-hand square composed with inr follows since all other inner parts commute as described below (3.2). The commutativity of the left-hand component is established using Diagram (3.3); indeed, since the outside and all other parts of that diagram commute for our morphism s so does the desired upper right-hand part composed with inl. $\square$

Theorem 3.6. (Sandwich Theorem) *Consider the algebra*

$$ k' = (MHMC \xrightarrow{Mk} MC \xrightarrow{\widehat{b}} C), $$

where $b : KC \to C$ *is the ℓ-interpretation in C and* $k = c^{-1} \cdot H\widehat{b}$ *as in Theorem 3.5. Then* (C, k') *is a cia for the functor MHM.*

Proof. The proof is a slight modification of the proof of Theorem 3.5: now we are given a flat equation morphism $e : X \to MHMX + C$ and form the morphism

$$\overline{e} = (\ X + C \xrightarrow{\ [e,\mathsf{inr}]\ } MHMX + C$$

$$\Big\downarrow {}^{MHMX + MH\eta_C \cdot \eta_{HC} \cdot c}$$

$$MHMX + MHMC \xrightarrow{\ \mathsf{can}\ } MHM(X + C)\ ).$$

This morphism $\overline{e}$ is a sandwiched ℓ-equation and we invoke Theorem 2.16 to see that it has a unique solution s. The rest of this proof is left to the reader since it is very close to the one of Theorem 3.5. $\qquad\square$

Remark 3.7. We discuss a few generalizations of the results in this section obtained by weakening the assumptions on M or on (C, c^{-1}).

(1) In our statements of Theorems 3.5 and 3.6, M was the free monad on K. However, it is possible to abstract away from this, by considering an *arbitrary* monad M. In this setting, we take

$$\lambda : M(H \times \mathrm{Id}) \to (H \times \mathrm{Id})M$$

to be a distributive law of the monad M over the cofree copointed functor $H \times \mathrm{Id}$. This is all we need in order to define the λ-interpretation $\widehat{b} : MC \to C$ such that Diagram (2.4) commutes. cf. Theorem 2.7 and Remark 2.8(2). The version of Theorem 2.12 presented in [12, 13] and dually in [47] states that for every $e : X \to HMX$ there is a unique solution $e^\dagger : X \to C$, i.e., $e^\dagger$ is such that the diagram in (2.6) commutes. Inspection of the proofs reveals that all our results so far hold in this generality . So Theorem 3.5 shows that (C, k) is a cia for HM, and Theorem 3.6 shows that (C, k') is a cia for MHM.

(2) Going even further, not all of the monad structure has been used in our work. Observe that our proof of Theorem 3.5 only makes use of the unit $\eta : \mathrm{Id} \to M$ of the monad M, not the multiplication. In fact, there are versions of Theorem 3.5 and 3.6 that hold for a pointed functor M in lieu of a monad and for a given distributive law of M over the cofree copointed functor $H \times \mathrm{Id}$ or the functor H, respectively. However, in this case we need to assume that the category $\mathcal{A}$ is cocomplete. The technical details are somewhat different than what we have seen and we discuss them in detail in an appendix that is provided as a supplementary file with our paper.

(3) Capretta, Uustalu and Vene [17] extend (the dual) of Theorem 2.12 by replacing the algebra (C, c^{-1}) arising from the terminal coalgebra by an algebra $a : HA \to A$ having the property that for every coalgebra $e : X \to HX$ there exists a unique coalgebra-to-algebra homomorphism from (X, e) to (A, a). One may ask whether the two theorems above can be extended from the initial cia to an arbitrary cia for H. However, note that our proof makes use of the fact that $c : C \to HC$ is an isomorphism. So the desired extension of our results is not obvious, and we leave this as an open problem for further work.

Theorems 3.5 and 3.6 extend Theorems 2.12 and 2.16 in two important ways. Firstly, the structure of a cia allows one to reuse solutions of a given flat equation morphism by using constants in C on the right-hand sides of recursive equations (cf. Remark 3.3). This gives a clear explanation of why it is possible to use recursively defined objects (processes, streams, etc.) in subsequent recursive definitions. This kind of modularity of the unique

solutions is a useful and desired property often employed in specifications. We shall discuss a concrete instance of this in Example 6.2.

Secondly, Theorem 3.6 permits the right-hand sides of recursive specifications to be from a wider class. For example, Milner's solution theorem for CCS (see [38], 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 (cf. Example 2.15). Hence, Theorem 3.6 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 3.5 and 3.6, the existing body of results on the semantics of recursion in cias [4, 33, 35] now gives us further theorems.

We begin with a terse review of some terminology from the area. We assume that in addition to the terminal H-coalgebra C, for every object X the terminal coalgebra $T^H X$ for $H(-) + X$ exists, i.e., in the terminology of loc. cit., H is *iterable*. Our examples in 2.10 are all iterable endofunctors of Set.

As explained in Example 3.2(1), the structure of the terminal coalgebra $T^H X$ yields the free cia on X with its structure and universal arrow as displayed below:

$$HT^H X \xrightarrow{\tau_X^H} T^H X \qquad X \xrightarrow{\eta_X^H} T^H X\,.$$

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

$$\kappa^H = (\, H \xrightarrow{H\eta^H} HT^H \xrightarrow{\tau^H} T^H \,)\,.$$

It was proved in [4, 33] that the monad T^H is characterized as the free completely iterative monad on H (with the universal natural transformation κ^H). We shall not recall the concept of a completely iterative monad as it is not needed in the present paper. However, we shall need that the assignment $H \mapsto T^H$ readily extends to natural transformations. Let $h : H \to H'$ be a natural transformation between iterable endofunctors. Then by the universal property of T^H we have a unique monad morphism

$$T^h : T^H \to T^{H'} \qquad \text{such that} \qquad
\begin{array}{ccc}
H & \xrightarrow{\ \kappa^H\ } & T^H \\
{\scriptstyle h}\downarrow & & \downarrow{\scriptstyle T^h} \\
H' & \xrightarrow{\ \kappa^{H'}\ } & T^{H'}
\end{array}
\qquad (4.1)$$

commutes.

Finally, let (A, a) be a cia for H. Then there is a unique H-algebra homomorphism

$$\widetilde{a} : T^H A \to A \qquad \text{such that} \qquad \widetilde{a} \cdot \eta_A^H = \mathrm{id}_A\,.$$

We call $\widetilde{a}$ the *evaluation morphism* associated with A. It is easy to prove that $\widetilde{a} \cdot \kappa_A^H = a$.

Remark 4.1. In the case of a polynomial functor H_Σ on Set, the evaluation morphism $\widetilde{a}$ can be thought of as a map that takes a (*not* necessarily finite) Σ-tree t with variables in the cia A and computes the value of t in A using the algebraic operations on A given by the structure $a : H_\Sigma A \to A$.

In previous work it was 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 4.2. [4, 33] 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

$$
\begin{array}{ccc}
X & \xrightarrow{\ e\ } & T^H(X + A) \\
& {\scriptstyle f}\searrow & \big\uparrow {\scriptstyle [\tau^H_{X+A},\eta^H_{X+A}\cdot\mathsf{inr}]} \\
& & HT^H(X + A) + A
\end{array}
$$

A *solution* of an equation morphism e in a cia (A, a) is a morphism $e^\dagger : X \to A$ such that the following square commutes:

$$
\begin{array}{ccc}
X & \xrightarrow{\ e^\dagger\ } & A \\
{\scriptstyle e}\big\downarrow & & \big\uparrow {\scriptstyle \widetilde{a}} \\
T^H(X + A) & \xrightarrow[T^H[e^\dagger,\mathrm{id}_A]]{} & T^H A
\end{array}
$$

Example 4.3. For a polynomial functor H_Σ on Set an equation morphism $e : X \to T^{H_\Sigma}(X + A)$ corresponds to a system of equations (1.4), where each right-hand side t_i is a (finite or infinite) Σ-tree with leaves labeled by a variable x_j or elements $a \in A$. Guardedness is the syntactic restriction that no right-hand side tree t_i is simply a single node tree with a variable as a label. A solution assigns to every variable x_i an element $a_i \in A$ such that $a_i = \widetilde{a}(t_i[\vec{a_j}/\vec{x_j}])$ for every $i \in I$, i.e., if we substitute all the solutions a_j for the corresponding variables x_j in t_i and evaluate the resulting tree in A using $\widetilde{a}$, then we obtain a_i.

Theorem 4.4. [33] *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 [35]; one can solve recursive function definitions uniquely in a cia. We recall the respective result.

Definition 4.5. 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

$$
e = (\ V \xrightarrow{\ f\ } HT^{H+V} \xrightarrow{\ \mathsf{inl}T^{H+V}\ } (H + V)T^{H+V} \xrightarrow{\ \tau^{H+V}\ } T^{H+V}\ ),
$$

where $\mathsf{inl} : H \to H + V$ is the coproduct injection.

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 $\beta : T^{H+V} A \to A$ with $\beta \cdot \kappa^{H+V}_A = [a, e^\ddagger_A]$ and such that we have

$$
e^\ddagger_A = (\ VA \xrightarrow{\ e_A\ } T^{H+V} A \xrightarrow{\ \beta\ } A\ ). \tag{4.2}
$$

Example 4.6. As explained in [35], for polynomial set functors, recursive program schemes as defined above provide a categorical formulation of recursive function definitions such as (1.5) from the introduction. For example, let $VX = X$ be the polynomial functor

associated with the signature with one unary operation symbol f and let $HX = X \times X + X$ be the polynomial functor for the signature of the givens F and G. Then (1.5) uniquely determines the natural transformation

$$e : V \to T^{H+V} \qquad \text{with} \qquad e_X(x) = \begin{array}{c} F \\ \diagup \;\; \diagdown \\ x \quad\; G \\ | \\ f \\ | \\ x \end{array}$$

Here guardedness corresponds to the syntactic restriction that the right-hand side of (1.5) starts with a given operation symbol such as F. Any cia A for H provides interpretations of the given operation symbols F and G as actual operations $F_A : A \times A \to A$ and $G_A : A \to A$, and an interpreted solution in A is precisely a new unary operation $f_A : A \to A$ such that for all $a \in A$, $f_A(a) = F_A(a, G_A(f_A(a)))$.

Theorem 4.7. [35] *In a cia, every guarded rps has a unique interpreted solution.*

We are now able to prove more. The next theorem implies modularity of taking solutions of recursive program schemes: operations obtained as solutions of recursive program schemes can be used as givens in subsequent definitions of other recursive program schemes. These new schemes will still have unique solutions. For the special case of interpreted rps solutions in cias this strengthens the results in [36].

Theorem 4.8. *Let $e : V \to T^{H+V}$ be a guarded rps, and let $a : HA \to A$ be a cia. Then the interpreted solution $e_A^{\ddagger} : VA \to A$ extends the cia structure on A; more precisely, the algebra $[a, e_A^{\ddagger}] : (H+V)A \to A$ is a cia for $H + V$.*

Remark 4.9. For the proof we need to recall some technical details. Recall that any guarded rps $e : V \to T^{H+V}$ as in Definition 4.5 induces a natural transformation

$$\overline{e} : T^{H+V} \to HT^{H+V} + \text{Id}$$

(see [35, Lemma 6.9]). The component $\overline{e}_A$ of this natural transformation at A is a flat equation morphism with parameters in A. Its unique solution in the cia (A, a) is the Eilenberg-Moore algebra structure $\beta : T^{H+V}A \to A$ in (4.2) satisfying $[a, e_A^{\ddagger}] = \beta \cdot \kappa_A^{H+V}$ (this follows from [35], see Lemma 7.4 and the proof of Theorem 7.3).

Proof of Theorem 4.8. Let $m : X \to (H+V)X + A$ be a flat equation morphism. We need to prove that m has a unique solution s. As shortcut notations we shall write T' for T^{H+V}, $\tau'_X : (H+V)T'X \to T'X$ for the corresponding structure of a free cia for $H + V$ as well as η' and μ' for the unit and multiplication of the monad T' and $\kappa' = \tau' \cdot (H+V)\eta'$ (cf. the introduction to Section 4).

(1) Existence of a solution. Since $T'A$ is the terminal coalgebra for $(H+V)(-) + A$ we have a unique homomorphism $h : X \to T'A$. We show that

$$s = (\, X \xrightarrow{\;h\;} T'A \xrightarrow{\;\beta\;} A \,)$$

is a solution of m in the algebra $(A, [a, e_A^\ddagger])$. To see this, consider the following diagram:

$$
\begin{array}{ccccc}
X & \xrightarrow{\;\;h\;\;} & T'A & \xrightarrow{\;\;\beta\;\;} & A \\
{\scriptstyle m}\downarrow & & {\scriptstyle [\tau'_A,\eta'_A]}\uparrow & & \uparrow{\scriptstyle [a,e_A^\ddagger,A]} \\
(H+V)X + A & \xrightarrow[(H+V)h+A]{} & (H+V)T'A + A & \xrightarrow[(H+V)\beta+A]{} & (H+V)A + A
\end{array}
$$

The left-hand square commutes since h is a coalgebra homomorphism, and for the right-hand component of the right-hand square use the unit law $\beta \cdot \eta'_A = \mathrm{id}_A$ of the Eilenberg-Moore algebra β. It remains to prove the commutativity of the left-hand component. This is established by inspecting the diagram below:

$$
\begin{array}{ccccc}
 & & \xrightarrow{\qquad\qquad \tau'_A \qquad\qquad} & & \\
(H+V)T'A & \xrightarrow{\;\kappa'_{T'A}\;} & T'T'A & \xrightarrow{\;\mu'_A\;} & T'A \\
{\scriptstyle (H+V)\beta}\downarrow & & {\scriptstyle T'\beta}\downarrow & & \downarrow{\scriptstyle \beta} \\
(H+V)A & \xrightarrow{\;\kappa'_A\;} & T'A & \xrightarrow{\;\beta\;} & A \\
 & & \xrightarrow[\qquad\qquad [a,e_A^\ddagger] \qquad\qquad]{} & &
\end{array}
\tag{4.3}
$$

The commutativity of the upper part is standard (see Corollary 3.17 in [35]), for the lower one see Remark 4.9, the left-hand inner square commutes due to naturality of κ', and the right-hand inner square by one of the laws for the Eilenberg-Moore algebra β.

(2) Uniqueness of solutions. Since e is a guarded rps, it factors through some $f : V \to HT'$ (cf. Definition 4.5). From f and m we form a flat equation morphism

$$g : X + T'X \to H(X + T'X) + A$$

w. r. t. H as follows. The left-hand component of g is

$$g \cdot \mathsf{inl} = (X \xrightarrow{\;m\;} HX + VX + A \xrightarrow{\;HX+f_X+A\;} HX + HT'X + A \xrightarrow{\;\mathsf{can}+A\;} H(X + T'X) + A),$$

and the right-hand component of g is

$$g \cdot \mathsf{inr} = (T'X \xrightarrow{\;\bar{e}_X\;} HT'X + X \xrightarrow{\;[\mathsf{inl}\cdot H\mathsf{inr}, g\cdot\mathsf{inl}]\;} H(X + T'X) + A).$$

Since (A, a) is a cia for H there exists a unique solution $g^\dagger : X + T'X \to A$. Now let $s : X \to A$ be any solution of the flat equation morphism m in the algebra $[a, e_A^\ddagger] : (H+V)A \to A$. We will show below that $[s, \beta \cdot T's] : X + T'X \to A$ is a solution of g in the H-algebra (A, a). So since (A, a) is a cia we have the following equation:

$$g^\dagger = [s, \beta \cdot T's] : X + T'X \to A. \tag{4.4}$$

Then s is uniquely determined by $g^\dagger$.

In order to prove Equation (4.4) we need to verify that the following square commutes:

$$
\begin{array}{ccc}
X + T'X & \xrightarrow{\;[s,\beta\cdot T's]\;} & A \\
{\scriptstyle g}\downarrow & & \uparrow{\scriptstyle [a,A]} \\
H(X + T'X) + A & \xrightarrow[\;H[s,\beta\cdot T's]+A\;]{} & HA + A
\end{array}
\tag{4.5}
$$

We shall verify the commutativity of the two coproduct components separately. For the left-hand component we consider the diagram below:

$$
\begin{array}{c}
\xymatrix{
X \ar[r]^{s} & A \\
}
\end{array}
$$

(4.6)

The left-hand part commutes by the definition of g, the right-hand part commutes trivially, the upper square commutes since s is a solution of m and the lower triangle commutes trivially, again. It remains to verify that the middle part commutes. We check the commutativity of this part componentwise: the left-hand and right-hand components commute trivially. We do not claim that the middle component commutes. However, in order to prove that the overall outside of (4.6) commutes, we need only show that this middle component commutes when extended by $[a, A] : HA + A \to A$. To see this consider the next diagram:

This diagram commutes: the left-hand part commutes since e is a guarded rps; the upper and lower squares in the middle commute due to the naturality of e and of $\tau' : (H+V)T' \to T'$ and $\mathsf{inl}T' : HT' \to (H+V)T'$, respectively; the upper right-hand triangle commutes since $e_A^{\ddagger}$ is an interpreted solution of the rps e. Finally, to see that the lower right-hand part commutes recall from Diagram (4.3) that

$$[a, e_A^{\ddagger}] \cdot (H + V)\beta = \beta \cdot \tau_A'.$$

Compose this last equation with the coproduct injection $\mathsf{inl}_{T'A} : HT'A \to (H + V)T'A$ to obtain the desired commutativity.

Finally, we verify that the right-hand component of (4.5) commutes. Indeed, consider the diagram below:

$$
\begin{array}{ccccc}
T'X & \xrightarrow{\;T's\;} & T'A & \xrightarrow{\;\beta\;} & A \\
\downarrow{\scriptstyle \overline{e}_X} & & \downarrow{\scriptstyle \overline{e}_A} & & \\
HT'X + X & \xrightarrow{HT's+s} & HT'A + A & & \uparrow{\scriptstyle [a,A]} \\
\downarrow{\scriptstyle [\mathrm{inl}\cdot H\mathrm{inr},g\cdot\mathrm{inl}]} & & & \searrow{\scriptstyle H\beta+A} & \\
H(X + T'X) + A & \xrightarrow{H[s,\beta\cdot T's]+A} & & & HA + A
\end{array}
$$

with $g\cdot\mathrm{inr}$ on the left.

The left-hand part commutes by the definition of g; the upper middle square commutes by the naturality of $\overline{e}$, the right-hand part commutes since β is the solution of $\overline{e}_A$ in the cia (A,a) (see Remark 4.9); and for the lower middle part we consider the components separately: the left-hand component clearly commutes by the functoriality of H, and for the right-hand component observe that it commutes when extended by $[a,A] : HA+A \to A$ (see Diagram (4.6)). Thus, the outside of the diagram above commutes, and this completes the proof. □

Coming back to our setting in Section 3, let $\ell : K(H \times \mathrm{Id}) \to HM$ be an abstract GSOS rule, where M is the free monad on K (or, more generally, let λ be a distributive law of an arbitrary monad over the cofree copointed functor $H \times \mathrm{Id}$). Assume furthermore that the composite HM is iteratable. By applying the two Theorems 4.4 and 4.7 and also Theorem 4.8 to the cia $k : HMC \to C$ from Theorem 3.5 we get two more solutions theorems for free:

Corollary 4.10. *Every guarded equation morphism* $e : X \to T^{HM}(X + C)$ *has a unique solution in the cia* (C,k).

Corollary 4.11. *Every guarded rps* $e : V \to T^{HM+V}$ *has a unique interpreted solution in the cia* (C,k), *and this solution extends the cia structure on* C.

Assuming that MHM is iteratable two similar theorems hold for the cia $k' : MHMC \to C$ obtained from Theorem 3.6:

Corollary 4.12. *Every guarded equation morphism* $e : X \to T^{MHM}(X + C)$ *has a unique solution in the cia* (C,k').

Corollary 4.13. *Every guarded rps* $e : V \to T^{MHM+V}$ *has a unique interpreted solution in the cia* (C,k'), *and this solution extends the cia structure on* C.

4.1. **Summary: Equation Formats.** (1) Until now we have seen several formats of equation morphisms. These are related in the sense that some formats comprise others as a special case. In particular, guarded "sandwiched" equation morphisms comprise all other

formats as shown in the following picture:

$$
\begin{array}{ccc}
\text{guarded equation m.} & & \text{guarded equation m.} \\
X \to T^{HM}(X + C) & \xrightarrow{\ T^{\eta HM}_{X+C} \cdot e\ } & X \to T^{MHM}(X + C) \\
\text{Corollary 4.10} & & \text{Corollary 4.12} \\
\Big\uparrow {\scriptstyle\mathsf{can}\cdot(\kappa_X^{HM}+\eta_C^{HM})\cdot e} & & \Big\uparrow {\scriptstyle\mathsf{can}\cdot(\kappa_X^{MHM}+\eta_C^{MHM})\cdot e} \\
\text{flat equation m.} & & \text{flat equation m.} \\
X \to HMX + C & \xrightarrow{\ (\eta_{HMX}+C)\cdot e\ } & X \to MHMX + C \\
\text{Theorem 3.5} & & \text{Theorem 3.6} \\
\Big\uparrow {\scriptstyle\mathsf{inl}\cdot e} & & \Big\uparrow {\scriptstyle\mathsf{inl}\cdot e} \\
\ell\text{-equation} & & \text{sandwiched } \ell\text{-equation} \\
X \to HMX & \xrightarrow{\ \eta_{HMX}\cdot e\ } & X \to MHMX \\
\text{Theorem 2.12} & & \text{Theorem 2.16}
\end{array}
$$

The arrows point to more general formats, and their labels indicate how one forms an equation morphism of the more general format from a given equation morphism e of the simpler format. Natural transformations without a superscript refer to the free monad M, the ones with a superscript refer to T-monads. The monad morphism $T^{\eta HM} : T^{HM} \to T^{MHM}$ arises from the natural transformation $\eta HM : HM \to MHM$ as explained at the beginning of this section (cf. (4.1)). In all cases one readily proves that the solutions are preserved along the arrows. For example, solutions of $e : X \to HMX + C$ in the cia (C, k) are in one-to-one correspondence with solutions of $(\eta_{HFX} + C) \cdot e : X \to MHMX + C$ in the cia (C, k'), etc.

(2) Similarly, guarded rps's in Corollary 4.13 subsume those in Corollary 4.11 which in turn subsume the rps of the form $V \to T^{H+V}$ from Theorem 4.7. Pictorially, we have

$$
\begin{array}{ccccc}
\text{(guarded) rps} & \xrightarrow{\ T^{H\eta+V}\cdot e\ } & \text{(guarded) rps} & \xrightarrow{\ T^{\eta HM+V}\cdot e\ } & \text{(guarded) rps} \\
V \to T^{H+V} & & V \to T^{HM+V} & & V \to T^{MHM+V}
\end{array}
$$

Again, one readily proves that solutions are preserved along the arrows. For example, solutions of $e : V \to T^{H+V}$ in the cia (C, c^{-1}) are in one-to-one correspondence with solutions of $T^{H\eta+V} \cdot e$ in the cia (C, k).

(3) Finally, guarded rps's subsume guarded equation morphisms. To see this let $e : X \to T^H(X + C)$ be a guarded equation morphism. Consider the constant functors C_X and C_C and let $H' = H + \mathsf{C}_C$. Then e corresponds precisely to a natural transformation $\overline{e} : \mathsf{C}_X \to T^H(\mathsf{C}_X + \mathsf{C}_C)$, and this gives rise to an rps as follows

$$
\begin{array}{c}
\mathsf{C}_X \xrightarrow{\ \overline{e}\ } T^H(\mathsf{C}_X + \mathsf{C}_C) \xrightarrow{\ T^H \kappa^{\mathsf{C}_X + \mathsf{C}_C}\ } T^H T^{\mathsf{C}_X + \mathsf{C}_C} \\
\Big\downarrow {\scriptstyle T^{\mathsf{inl}} T^{[\mathsf{inr}, \mathsf{inm}]}} \\
T^{H+\mathsf{C}_C+\mathsf{C}_X} T^{H+\mathsf{C}_C+\mathsf{C}_X} \xrightarrow{\ \mu^{H+\mathsf{C}_C+\mathsf{C}_X}\ } T^{H'+\mathsf{C}_X}.
\end{array}
$$

It is straightforward but rather tedious to check that this rps is guarded and that its solutions are in one-to-one correspondence with the solutions of e.

5. Recursive Function Definitions over the Behavior

Even with all the results we have seen so far, we are still not able to obtain functions such as the shuffle product $\otimes$ on streams (see (1.2)) as a unique solution since its definition refers to the behavior of the arguments of the function. Notice also that the specification of $\otimes$ makes use of the stream addition $+$ operation, so this operation is assumed as given or previously specified and the specification of $\otimes$ is built on top of the specification of $+$. Our aim in this section is to prove results that yield unique solutions of such specifications in terminal coalgebras.

Notation 5.1. From now on we shall also need to consider free monads of other functors than K from Asumption 2.1. We follow the convention that whenever we write $\widehat{F}$ for a functor F we assume that a free monad $\widehat{F}$ exists and is given objectwise by free algebras for F (cf. Remark 2.2(2)).

Finally, to shorten notation, we usually abbreviate $K + V$ by F.

Outline 5.2. Here is the basic plan for the results in this section. We begin, as before, with a endofunctor H and its terminal coalgebra (C, c). We also have a separate functor K, and an abstract GSOS rule $\ell : K(H \times \mathrm{Id}) \to H\widehat{K}$ specifying a set of "given" operations. When we say "specify" here, we refer to the algebra structure $b : KC \to C$, the unique morphism so that

$$c \cdot b = H\widehat{b} \cdot \ell_C \cdot K\langle c, \mathrm{id}_C \rangle$$

(see Theorem 2.7). We also have another functor V corresponding to the "new" operation symbols that we wish to interpret. What our approach requires at this point is a natural transformation

$$e : V(H \times \mathrm{Id}) \to H\widehat{K + V}.$$

We shall call e a *recursive program scheme w. r. t.* ℓ (or, an *ℓ-rps*, for short). From e and ℓ we shall obtain a natural transformation

$$n : (K + V)(H \times \mathrm{Id}) \to H\widehat{K + V}$$

in a canonical way. Again, n is an abstract GSOS rule—but notice that the functor involved is $K + V$, not K as it is for ℓ. This natural transformation n has an interpretation in C, call it

$$a : (K + V)C \to C.$$

Then it will turn out that $a \cdot \mathrm{inl} = b$, so that the algebra structure a is an extension of b. The interpretation of the new operation symbols in C will correspond to the V-algebra $(C, a \cdot \mathrm{inr})$. We will prove in Theorem 5.14 that this algebra is uniquely determined as a *solution* of the ℓ-rps e.

The upshot is that we start and end with the same kind of data, but with a different functor. We start with K, ℓ, and b, and we end with $K + V$, n, and a. The point we are trying to make here is that the situation repeats, and so we can apply the result successively.

The rest of this section works out the details of this outline. To see that this approach actually accounts for a large number of interesting operations on terminal coalgebras, we present many examples in Section 6. Prior to this, we have two other contributions which extend the basic result in the same ways as the results we saw in Section 2. We have a "sandwiched" version of Outline 5.2, and we also have results about cia structures. And in our main result, Theorem 5.23, we prove that every sandwiched ℓ-rps (see Definition 5.21 has a unique solution in C which extends the cia structure for $\widehat{K}H\widehat{K}$ on C given by Theorem 3.6.

Assumption 5.3. We continue to work under Assumption 2.1, and in addition we fix an abstract GSOS rule $\ell : K(H \times \mathrm{Id}) \to H\widehat{K}$ and an endofunctor $V : \mathcal{A} \to \mathcal{A}$.

Notation 5.4. (1) We overload the notation from Remark 2.2(3) and write, for any functor F on $\mathcal{A}$, $\varphi : F\widehat{F} \to \widehat{F}$ and $\eta : \mathrm{Id} \to \widehat{F}$ for (the natural transformations given by) the structures and universal morphisms of the free F-algebras, as well as $\mu : \widehat{F}\widehat{F} \to \widehat{F}$ and $\kappa : F \to \widehat{F}$ for the multiplication and universal natural transformation of the free monad $\widehat{F}$.

(2) Let F and G be endofunctors of $\mathcal{A}$. The coproduct injections $\mathsf{inl} : F \to F + G \leftarrow G : \mathsf{inr}$ lift to monad morphisms on the corresponding free monads, and we denote those monad morphisms by $\widehat{\mathsf{inl}} : \widehat{F} \to \widehat{F + G} \leftarrow \widehat{G} : \widehat{\mathsf{inr}}$.

Remark 5.5. (1) Notice that the monad morphism $\widehat{\mathsf{inl}} : \widehat{F} \to \widehat{F + G}$ is uniquely determined by the commutativity of the following square of natural transformations:

$$
\begin{array}{ccc}
F & \xrightarrow{\ \kappa\ } & \widehat{F} \\
{\scriptstyle \mathsf{inl}}\big\downarrow & & \big\downarrow{\scriptstyle \widehat{\mathsf{inl}}} \\
F + G & \xrightarrow[\ \kappa\ ]{} & \widehat{F + G}
\end{array}
$$

Similarly for $\widehat{\mathsf{inr}}$.

(2) Recall from Notation 2.3 that for every F-algebra (A, a) we have the corresponding Eilenberg-Moore algebra $\widehat{a} : \widehat{F}A \to A$ and that $a = \widehat{a} \cdot \kappa_A$. Moreover, the category of F-algebras is isomorphic to the category of Eilenberg-Moore algebras for $\widehat{F}$. More precisely, $a \mapsto \widehat{a}$ and precomposition with κ_A extend to mutually inverse functors.

(3) Combining parts (1) and (2) of this remark, we see that for every algebra $a : (F+G)A \to A$ the equation $\widehat{a} \cdot \widehat{\mathsf{inl}}_A = \widehat{a \cdot \mathsf{inl}_A}$ holds. Indeed, both sides are equal when precomposed with κ_A:

$$\widehat{a} \cdot \widehat{\mathsf{inl}}_A \cdot \kappa_A = \widehat{a} \cdot \kappa_A \cdot \mathsf{inl}_A = a \cdot \mathsf{inl}_A = \widehat{a \cdot \mathsf{inl}_A} \cdot \kappa_A \,.$$

If we make the coproduct algebra structure explicit as in $a = [a_0, a_1]$, we obtain

$$\widehat{[a_0, a_1]} \cdot \widehat{\mathsf{inl}}_A = \widehat{a_0} \qquad \text{and} \qquad \widehat{[a_0, a_1]} \cdot \widehat{\mathsf{inr}}_A = \widehat{a_1} \,.$$

Definition 5.6. A *recursive program scheme w. r. t. ℓ* (shortly, *ℓ-rps*) is a natural transformation

$$e : V(H \times \mathrm{Id}) \to H\widehat{F} \,,$$

where (throughout this section) $F = K + V$.

Construction 5.7. Let $e : V(H \times \mathrm{Id}) \to H\widehat{F}$ be an ℓ-rps. This gives an abstract GSOS rule $n : F(H \times \mathrm{Id}) \to H\widehat{F}$ defined on its coproduct components as displayed below:

$$
\begin{array}{ccc}
V(H \times \mathrm{Id}) & & \\
{\scriptstyle \mathrm{inr}(H\times\mathrm{Id})}\Big\downarrow & \searrow^{e} & \\
F(H \times \mathrm{Id}) & \xrightarrow{\ n\ } & H\widehat{F} \\
{\scriptstyle \mathrm{inl}(H\times\mathrm{Id})}\Big\uparrow & & \Big\uparrow{\scriptstyle H\widehat{\mathrm{inl}}} \\
K(H \times \mathrm{Id}) & \xrightarrow[\ \ell\ ]{} & H\widehat{K}
\end{array}
$$

We write $a : FC \to C$ for the n-interpretation in C (cf. Definition 2.9).

Remark 5.8. The construction of the abstract GSOS rule n above is reminiscent of the composition of two distributive laws $\lambda : MD \to DM$ and $\lambda' : M'D \to DM'$ of the monads M and M' over the same copointed functor D using the coproduct of monads, see [40, 31]. For free monads $M = \widehat{G}$ and $M' = \widehat{G'}$ and for $D = H \times \mathrm{Id}$, λ and λ' are equivalently presented by abstract GSOS rules $\ell : G(H \times \mathrm{Id}) \to H\widehat{G}$ and $\ell' : G'(H \times \mathrm{Id}) \to H\widehat{G'}$ and the construction of loc. cit. amounts to forming $\widetilde{n}$ as shown in the diagram below:

$$
\begin{array}{ccc}
G(H \times \mathrm{Id}) & \xrightarrow{\ \ell\ } & H\widehat{G} \\
{\scriptstyle \mathrm{inl}(H\times\mathrm{Id})}\Big\downarrow & & \Big\downarrow{\scriptstyle H\widehat{\mathrm{inl}}} \\
(G + G')(H \times \mathrm{Id}) & \xrightarrow{\ \widetilde{n}\ } & H\widehat{G + G'} \\
{\scriptstyle \mathrm{inr}(H\times\mathrm{Id})}\Big\uparrow & & \Big\uparrow{\scriptstyle H\widehat{\mathrm{inr}}} \\
G'(H \times \mathrm{Id}) & \xrightarrow[\ \ell'\ ]{} & H\widehat{G'}
\end{array}
$$

Notice that there is no interplay between ℓ and ℓ'. So, as explained in loc. cit., in the case of operational rules of process combinators the formation of $\widetilde{n}$ corresponds precisely to forming the disjoint union of two transition system specifications with mutually independent operations. In contrast, in Construction 5.7 the left-hand component e depends on K. So in the case of operational rules for process combinators our construction corresponds to combining a given transition specification (modeled by ℓ) with another one (modeled by e) which makes use of the operators specified by the first specification.

Our next result substantiates the details presented in Outline 5.2.

Proposition 5.9. *Let* $b : KC \to C$ *be the interpretation of* $\ell : K(H \times \mathrm{Id}) \to H\widehat{K}$. *Then for* $a : FC \to C$ *from Construction 5.7 we have*

$$
b = (\, KC \xrightarrow{\ \mathrm{inl}_C\ } FC \xrightarrow{\ a\ } C \,).
$$

Proof. Consider the diagram below:

The lower square commutes since a is the n-interpretation in C (cf. Theorem 2.7) and the upper right-hand one by the definition of n (cf. Construction 5.7). The upper left-hand square commutes by the naturality of $\mathsf{inl} : K \to F$, and for the right-hand part we remove H and use Remark 5.5(3). Thus the outside commutes.

Now recall that b is uniquely determined by the commutativity of the diagram in Theorem 2.7. Thus, $a \cdot \mathsf{inl}_C = b$ holds, as desired. $\qquad\square$

Summary 5.10. Let us summarize and review our work in this section so far. We say that the tuple
$$(c : C \to HC, \quad \ell : K(H \times \mathrm{Id}) \to H\widehat{K}, \quad b : KC \to C)$$
is *appropriate* if c is a terminal coalgebra structure, ℓ is an abstract GSOS rule, and b is its interpretation. Given an appropriate tuple, the work in this section shows how to obtain another appropriate tuple: let V be a functor, let $F = K + V$, and let $e : V(H \times \mathrm{Id}) \to H\widehat{F}$ be an ℓ-rps. We consider
$$n = [H\widehat{\mathsf{inl}} \cdot \ell, e] : F(H \times \mathrm{Id}) \to H\widehat{F} \ .$$
This is an abstract GSOS rule involving F, so it has an interpretation $a : FC \to C$. So
$$(c : C \to HC, \quad n : F(H \times \mathrm{Id}) \to H\widehat{F}, \quad a : FC \to C) \tag{5.1}$$
is again appropriate. It extends the earlier appropriate tuple in the sense that $a \cdot \mathsf{inr} = b : VC \to C$.

Example 5.11. We continue our exploration of stream operations as defined by behavioral differential equations. We want to study the shuffle product on streams mentioned in the introduction. It is specified by
$$(\sigma \otimes \tau)_0 = \sigma_0 \cdot \tau_0 \qquad\qquad (\sigma \otimes \tau)' = \sigma \otimes \tau' + \sigma' \otimes \tau \ . \tag{5.2}$$
The behavior functor $H : \mathsf{Set} \to \mathsf{Set}$ here is $HX = \mathbb{R} \times X$. On the right side of (5.2), we see the stream addition operation $+$. This is a binary operation and so corresponds to a K-algebra structure on C, where $KX = X \times X$. Our work in this section shows how to define $\otimes$ "on top of" $(C, +)$. First observe that the given operation $+ : KC \to C$ is obtained as the interpretation of the abstract GSOS rule $\ell : K(H \times \mathrm{Id}) \to H\widehat{K}$ given by (2.5) in Example 2.10(2). Now we take $VX = X \times X$, the type functor of the binary operation $\otimes$. For $F = K + V$ we may then identify $\widehat{F}X$ with the set of all terms obtained by applying the operation symbols $\otimes$ and $+$ to variables in X. (Note that $\otimes$ and $+$ are regarded as uninterpreted symbols at this point, and for the given operation $+$ we have the

interpretation on C induced by ℓ.) To obtain an interpretation of $\otimes$ on C, we use the ℓ-rps e whose components

$$e_X : (\mathbb{R} \times X \times X) \times (\mathbb{R} \times X \times X) \to \mathbb{R} \times \widehat{F}X$$

are given by

$$e_X((r, x, x'), (s, y, y')) = (r \cdot s, (x \otimes y') + (x' \otimes y)).$$

The abstract GSOS rule $n : F(H \times \mathrm{Id}) \to H\widehat{F}$ from Construction 5.7 now induces the algebra structure $a : FC \to C$ whose left-hand component is, by Proposition 5.9, $a \cdot \mathsf{inl} = + : KC \to C$ and whose right-hand component is easily seen to be the desired shuffle product $a \cdot \mathsf{inr} = \otimes : VC \to C$. So Proposition 5.9 is a kind of sanity check: the operation induced by n for the left-hand component K of $F = K + V$ is the same operation $+$ we started with as given.

Example 5.12. We present another example related to the zipping of streams; see Examples 2.10(2) and 2.13. We have seen that there is an abstract GSOS rule $\ell : K(H \times \mathrm{Id}) \to H\widehat{K}$ with $KX = X \times X$ presenting the zip operation on streams. Using an ℓ-rps, we can show that there is a unique function $f : C \to C$ satisfying

$$f(\sigma) = \mathsf{zip}(\sigma, f(\sigma)).$$

We might mention a subtlety: consider the alternative definition

$$g(\sigma) = \mathsf{zip}(g(\sigma), \sigma).$$

The uniqueness assertion fails for this. The point is that each $(g(\sigma))_0$ is arbitrary. As for f, note that $(f(\sigma))_0 = \sigma_0$ for all streams σ. In fact, this observation is central to obtaining f via Theorem 5.14. The point is that we can write

$$f(\sigma) = \mathsf{zip}(\sigma_0.\sigma', f(\sigma)) = \sigma_0.\mathsf{zip}(f(\sigma), \sigma').$$

The last formulation makes it clear that we can obtain f using the method of this section: one takes $VX = X$, and then the last equation gives rise to

$$e : V(H \times \mathrm{Id}) \to H\widehat{F} \quad \text{with} \quad e_X : (r, x', x) \mapsto (r, \mathsf{zip}(f(x), x')).$$

Incidentally, functions like f play a role in the theory of paperfolding sequences; cf. [18], Observation 1.3.

5.1. **Interpreted solutions and cia structures.** At this point, the reader might wish to revisit Outline 5.2. The work we have done so far builds a V-algebra structure on C by taking an F-algebra structure and (in effect) throwing away the interpretation of the "givens", since we know them to be the same as in the original K-algebra structure. Our next result, Theorem 5.14, provides a different way to look at things. It provides a direct notion of an *interpreted solution*; this is a V-algebra. The theorem shows that ℓ-rps's have unique interpreted solutions. In addition, Theorem 5.14 shows that cia structures are propagated according to our development.

Definition 5.13. Let $e : V(H \times \mathrm{Id}) \to H\widehat{F}$ be an ℓ-rps. An *interpreted solution* of e in C is a V-algebra structure $s : VC \to C$ such that the triangle below commutes:

$$
\begin{array}{ccc}
VC & \xrightarrow{\quad s \quad} & C \\
{\scriptstyle V\langle x, \mathrm{id}_C\rangle}\downarrow & & \nearrow {\scriptstyle c^{-1}} \\
V(HC \times C) & & HC \\
{\scriptstyle e_C}\downarrow & \nearrow {\scriptstyle H[\widehat{b,s}]} & \\
H\widehat{F}C & &
\end{array}
\tag{5.3}
$$

where $b : KC \to C$ is the ℓ-interpretation in C (cf. Definition 2.9).

Theorem 5.14. *For every ℓ-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 $H\widehat{F}$ on C:*

$$
H\widehat{F}C \xrightarrow{\;H[\widehat{b,s}]\;} HC \xrightarrow{\;c^{-1}\;} C \,.
\tag{5.4}
$$

Proof. Given the abstract GSOS rule ℓ and the ℓ-rps e, form n as in Construction 5.7 and define

$$
s = (VC \xrightarrow{\;\mathrm{inr}\;} FC \xrightarrow{\;a\;} C)\,,
$$

where a is the n-interpretation in C. Observe immediately that, by Proposition 5.9,

$$
a = [b, s] : FC \to C,
$$

where $b : KC \to C$ is the ℓ-interpretation in C, and so we have

$$
\widehat{a} = \widehat{[b, s]} : \widehat{F}C \to C \,.
\tag{5.5}
$$

(1) We prove that s is a solution of e in C. Indeed, consider the commutative diagram

$$
\begin{array}{ccccc}
VC & \xrightarrow{V\langle c, \mathrm{id}_C\rangle} & V(H \times \mathrm{Id})C & & \\
\downarrow{\scriptstyle \mathrm{inr}_C} & & \downarrow{\scriptstyle \mathrm{inr}_{(H\times\mathrm{Id})C}} & \searrow{\scriptstyle e_C} & \\
FC & \xrightarrow{F\langle c, \mathrm{id}_C\rangle} & F(H \times \mathrm{Id})C & \xrightarrow{n_C} & H\widehat{F}C \\
\downarrow{\scriptstyle a} & & & & \downarrow{\scriptstyle H\widehat{a}} \\
C & \xrightarrow{\qquad\qquad c \qquad\qquad} & & & HC\,.
\end{array}
\tag{5.6}
$$

The lower square commutes since a is the n-interpretation in C, and the upper right-hand triangle by the definition of n (cf. Construction 5.7). The upper left-hand square commtes by the naturality of $\mathrm{inl} : V \to F$, and the left-hand part is the definition of s. Thus the outside commutes, and we see that s is a solution of e since $\widehat{a} = \widehat{[b, s]}$ holds by (5.5).

(2) We now prove that s is unique. Suppose that t is any solution of e. We will prove that

$$
[b, t] = a,
\tag{5.7}
$$

which implies the desired equation $s = t$.

In order to prove (5.7) we have to verify the commutativity of the following diagram (cf. Theorem 2.7):

$$
\begin{array}{ccccc}
FC & \xrightarrow{F\langle c,\mathrm{id}_C\rangle} & F(H \times \mathrm{Id})C & \xrightarrow{\ n_C\ } & H\widehat{F}C \\
{\scriptstyle [b,t]}\downarrow & & & & \downarrow{\scriptstyle H\widehat{[b,t]}} \\
C & & \xrightarrow{\qquad\qquad c \qquad\qquad} & & HC
\end{array}
$$

We verify this for the two coproduct components of $FC = KC + VC$ separately.

For the right-hand component we obtain diagram (5.6) above with s replaced by t and a by $[b,t]$, which commutes since t is a solution of e. For the left-hand component we obtain the diagram below:

$$
\begin{array}{ccccccc}
KC & \xrightarrow{K\langle c,\mathrm{id}_C\rangle} & K(H \times \mathrm{Id})C & \xrightarrow{\mathsf{inl}_{(H\times\mathrm{Id})C}} & F(H \times \mathrm{Id})C & \xrightarrow{\ n_C\ } & H\widehat{F}C \\
& & & {\scriptstyle \ell_C}\searrow & & \nearrow{\scriptstyle H\widehat{\mathsf{inl}}_C} & \\
{\scriptstyle b}\downarrow & & & & H\widehat{K}C & & \downarrow{\scriptstyle H\widehat{[b,t]}} \\
& & & & & \searrow{\scriptstyle H\widehat{b}} & \\
C & & & \xrightarrow{\qquad c \qquad} & & & HC
\end{array}
$$

The big left-hand part commutes since b is the ℓ-interpretation in C (cf. Theorem 2.7), the upper triangle commutes by the definition of n (see Construction 5.7), and for the right-hand triangle remove H and notice that

$$
\widehat{[b,t]} \cdot \widehat{\mathsf{inl}}_C = \widehat{b}
$$

by Remark 5.5(3).

(3) To complete the proof we will show that $c^{-1} \cdot H\widehat{[b,s]} : H\widehat{F}C \to C$ is the structure of a cia for $H\widehat{F}$. But this is a consequence of Theorem 3.5; indeed, recall that $[b,s]$ is the interpretation of the abstract GSOS rule $n : F(H \times \mathrm{Id}) \to H\widehat{F}$ in C, see (5.5).   □

Remark 5.15. Notice that the fact that the unique solution s of an ℓ-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 4.10 and 4.11 (where $M = \widehat{K+V}$).

As we shall see in Section 6, in conrete instances many operations are definable as unique solutions of ℓ-rps's. But there are operations that cannot be defined by any ℓ-rps (or abstract GSOS rule):

Example 5.16. The tail function $\mathsf{tl} : \mathbb{R}^\omega \to \mathbb{R}^\omega$ on streams cannot be defined as (part of) an interpretation of any abstract GSOS rule $\ell : K(\mathbb{R} \times \mathrm{Id} \times \mathrm{Id}) \to \mathbb{R} \times \widehat{K}$. For if it were possible to obtain the tail function in this way we would have the induced cia $c^{-1} \cdot H\widehat{b} : H\widehat{K}C \to C$ according to Theorem 3.5. Then, for $X = \{x\}$ the ℓ-equation (see Definition 2.11) $e : X \to \mathbb{R} \times \widehat{K}X$ given by $e(x) = (r, \mathsf{tl}(x))$ would have a unique solution $e^\dagger : X \to C$ for every $r \in \mathbb{R}$. But this is clearly not the case: every stream σ with r at its head yields a solution e, since $e^\dagger(\sigma) = \sigma$.

The next proposition shows that for two ℓ-rps's that do not interact the order in which the algebra $b : KC \to C$ is extended by their solutions does not matter.

Proposition 5.17. *Let $e_i : V_i(H \times \mathrm{Id}) \to H\widehat{K + V_i}$, $i = 1, 2$, be two ℓ-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.*

Remark 5.18. More precisely, we may first take $s_1 : V_1C \to C$ to obtain an extended cia structure as in (5.4), and then take the solution of $s_2 : V_2C \to C$ in the new cia, or vice versa. Either way, the resulting extended cia structure is

$$H(K \widehat{+ V_1 + V_2})C \xrightarrow{H[\widehat{b,s_1,s_2}]} HC \xrightarrow{c^{-1}} C \,. \tag{5.8}$$

Proof of Proposition 5.17. It is sufficient to prove that the cia structure on C obtained by extending $k : H\widehat{K}C \to C$ first by s_1 and then by s_2 is (5.8).

So take s_1 and extend the cia structure (C, k) to obtain the cia

$$c^{-1} \cdot H[\widehat{b, s_1}] : H\widehat{K + V_1}(C) \to C$$

(cf. (5.4)). Recall from the proof of Theorem 5.14 that this cia structure is obtained as follows: one first forms the abstract GSOS rule

$$n = [H\widehat{\mathrm{inl}} \cdot \ell, e_1] : (K + V_1)(H \times \mathrm{Id}) \to H\widehat{K + V_1}$$

whose interpretation is $b' = [b, s_1] : (K + V_1)C \to C$, cf. (5.5), and then one applies Theorem 3.5.

Now we form the following n-rps

$$V_2(H \times \mathrm{Id}) \xrightarrow{e_2} H\widehat{K + V_2} \xrightarrow{H[\widehat{\mathrm{inl},\mathrm{inr}}]} H(K \widehat{+ V_1 + V_2}) \,.$$

Its unique solution is easily seen to be s_2; indeed, consider the following diagram (and notice that the right-hand arrow is $H[\widehat{b', s_2}]$):

$$
\begin{array}{ccccccc}
V_2C & \xrightarrow{V_2\langle c,\mathrm{id}_C \rangle} & V_2(H \times \mathrm{Id})C & \xrightarrow{(e_2)_C} & H\widehat{K + V_2}C & \xrightarrow{H[\widehat{\mathrm{inl},\mathrm{inr}}]_C} & H(K \widehat{+ V_1 + V_2})C \\
{\scriptstyle s_2}\downarrow & & & & & & \downarrow{\scriptstyle H[\widehat{b,s_1,s_2}]} \\
C & & & & \xleftarrow{\hspace{3cm}} & {\scriptstyle H[\widehat{b,s_2}]} & HC \\
& & & & c^{-1} & &
\end{array}
$$

The left-hand part commutes since s_2 is the unique solution of e_2 and for the right-hand part we remove H and then precompose with κ_C to obtain

$$
\begin{array}{rcll}
[\widehat{b, s_1, s_2}] \cdot [\widehat{\mathrm{inl}, \mathrm{inr}}]_C \cdot \kappa_C & = & [\widehat{b, s_1, s_2}] \cdot \kappa_C \cdot [\mathrm{inl}, \mathrm{inr}]_C & \text{cf. Remark 5.5(1)} \\
& = & [b, s_1, s_2] \cdot [\mathrm{inl}, \mathrm{inr}]_C & \text{see Remark 5.5(2)} \\
& = & [b, s_2] & \\
& = & [\widehat{b, s_2}] \cdot \kappa_C & \text{see Remark 5.5(2)}
\end{array}
$$

The desired equality now follows since precomposition with κ yields an isomorphism of categories, see Remark 5.5(2). $\qquad\square$

Remark 5.19. Notice that we can always consider the algebraic operation provided by c^{-1} : $HC \to C$ as a given operation in any ℓ-rps for an abstract GSOS rule $\ell : K(H \times \mathrm{Id}) \to H\widehat{K}$. More precisely, we can assume that $K = K' + H$ and that the ℓ-interpretation $b : KC \to C$ has the form $b = [b', c^{-1}]$. Indeed, given any abstract GSOS rule $\ell' : K'(H \times \mathrm{Id}) \to H\widehat{K'}$ with ℓ'-interpretation b', we define the ℓ'-rps

$$e = \left(H(H \times \mathrm{Id}) \xrightarrow{H\pi_0} HH \xrightarrow{H\mathsf{inr}} H(K' + H) \xrightarrow{H\kappa} H\widehat{K' + H} \right).$$

It is easy to verify that c^{-1} is its solution; and, as we see from the proof of Theorem 5.14, we obtain a new abstract GSOS rule $n : (K' + H)(H \times \mathrm{Id}) \to H\widehat{K' + H}$ defined as in Construction 5.7 and having the n-interpretation $[b', c^{-1}]$. According to Proposition 5.17 we can do this construction at any step when defining operations, the result being always a GSOS rule $\ell : K(H \times \mathrm{Id}) \to H\widehat{K}$ with $K = K' + H$ and an ℓ-interpretation $[b', c^{-1}]$ containing the algebraic structure c^{-1}.

Summary 5.20. Again, we summarize and review our work in this section. Given an appropriate tuple

$$(c : C \to HC, \quad \ell : K(H \times \mathrm{Id}) \to H\widehat{K}, \quad b : KC \to C)$$

as in Summary 5.10 we know from Theorem 3.5 that $c^{-1} \cdot H\widehat{b}$ is a cia for the functor $H\widehat{K}$. The work in this section shows that the appropriate tuple in (5.1) is obtained as follows: Let V be a functor, let $F = K + V$, and let $e : V(H \times \mathrm{Id}) \to H\widehat{F}$ be an ℓ-rps. We take the unique interpreted solution $s : VC \to C$ of e, and then we obtain the appropriate tuple

$$(c : C \to HC, \quad n : F(H \times \mathrm{Id}) \to H\widehat{F}, \quad a = [b, s] : FC \to C).$$

extending the earlier appropriate tuple.

This provides the desired *modularity principle* for solutions of ℓ-rps's as discussed in the introduction and again (restricted to flat equations) in Remark 3.3. On the level of concrete syntactic specifications, the operations provided by the unique solution $s : VC \to C$ of any ℓ-rps e may occur as givens in any subsequent n-rps $f : W(H \times \mathrm{Id}) \to H\widehat{F + W}$, which in turn has a unique solution $t : WC \to C$ that yields another appropriate tuple etc. So our results iterate as desired.

5.2. Sandwiched ℓ-rps's.

We will now prove a version of the Sandwich Theorem 3.6 for ℓ-rps's. The goal is to be able to solve specifications from a wider class uniquely. Let us again explain the idea using the example of streams of reals. Here we have $HX = \mathbb{R} \times X$ on Set. Suppose that K and V both are polynomial functors associated with a signature of givens and newly defined operations on the terminal coalgebra $C = \mathbb{R}^\omega$. The inverse of the coalgebra structure $c = \langle \mathsf{hd}, \mathsf{tl} \rangle$ yields the family of prefix operations $r.(-)$ prepending the number r to a stream. The format of an ℓ-rps means that the new operations of type V are always defined by an equation with a prefix operation as a guard at its head, see e.g. the following specification of the shuffle product reformulated from the behavioral differential equation of (5.2):

$$r.x \otimes s.y = rs.((r.x \otimes y) + (x \otimes s.y)).$$

This guard $rs, -$ is sufficient to ensure a unique solution $\otimes$. However, in general it is not necessary that the guard appears at the head of the term on the right-hand side of an equation. We shall now define the more general format of *sandwiched* recursive program

schemes that allow the guard to occur further inside the term. A concrete example of this kind of specification is the "parallel composition" of non-well founded sets in (1.3); here the guarding operation is set-bracketing $(x_i)_{i \in I} \mapsto \{x_i \mid i \in I\}$, and this occurs inside the union operation (we shall come back to this example in Section 6.5).

Definition 5.21. A *sandwiched recursive program scheme w. r. t.* ℓ (shortly, ℓ-*srps*) is a natural transformation $e : V(H \times \mathrm{Id}) \to \widehat{K}H\widehat{F}$.

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

$$s = (\, VC \xrightarrow{V\langle c, \mathrm{id}_C \rangle} V(HC \times C) \xrightarrow{e_C} \widehat{K}H\widehat{F}C \xrightarrow{\widehat{K}H\widehat{[b,s]}} \widehat{K}HC \xrightarrow{\widehat{K}c^{-1}} \widehat{K}C \xrightarrow{\widehat{b}} C \,).$$

Remark 5.22. (1) The notion of an ℓ-srps subsumes the one of an ℓ-rps. Indeed, for a given ℓ-rps $e : V(H \times \mathrm{Id}) \to H\widehat{F}$ one forms the ℓ-srps

$$V(H \times \mathrm{Id}) \xrightarrow{e} H\widehat{K+V} \xrightarrow{\eta H\widehat{F}} \widehat{K}H\widehat{F}.$$

And one readily proves that solutions of this ℓ-srps in C are in one-to-one correspondence with solutions of e in C.

(2) The notion of an ℓ-(s)rps is incomparable to the three formats of rps's we saw in Section 4 (see Section 4.1(2)). Indeed, for polynomial functors on Set, the latter rps's are systems of recursive function equations that allow infinite trees on the right-hand sides while ℓ-(s)rps's correspond to recursive equations where right-hand sides are restricted to consist of terms (or finite trees) only.

However, notice that classical recursive program schemes as in [20] are restricted to allow only finite trees on right-hand sides of equations (as, e. g., in (1.5)). Such a restricted rps corresponds to a natural transformation

$$e = (V \longrightarrow \widehat{H+V} \xrightarrow{m} T^{H+V}),$$

where $m : \widehat{H+V} \to T^{H+V}$ is the unique monad morphism from the free monad on $H+V$ to T^{H+V} induced by $\kappa^{H+V} : H+V \to T^{H+V}$. This given recursive program scheme is guarded if we have $f : V \to H\widehat{H+V}$ satisfying a similar property as f in Definition 4.5. Now this clearly yields an ℓ-rps

$$e' = (\, V(H \times \mathrm{Id}) \xrightarrow{V\pi_1} V \xrightarrow{f} H\widehat{H+V} \,),$$

where $K = H$ and

$$\ell = (\, H(H \times \mathrm{Id}) \xrightarrow{H\pi_1} H \xrightarrow{H\eta} H\widehat{H} \,).$$

Again one readily proves that solutions of e' are in one-to-one correspondence with solutions of e, and so the restricted rps's are a special instance of ℓ-rps's.

We are now ready to state and prove the main result of this paper; it provides a unique solution theorem for ℓ-rps's, the most general specification format we consider in this paper. In order to simplify notation we will write (M, η, μ) for the free monad $\widehat{K+V}$ in the rest of this section.

Theorem 5.23. (Sandwich Theorem for ℓ-srps's) *For every ℓ-srps there exists a unique interpreted solution s in C. In addition, s extends the cia structure on C; more precisely,*

the following is the structure of a cia for MHM on C:

$$MHM(C) \xrightarrow{MH\widehat{[b,s]}} MH(C) \xrightarrow{Mc^{-1}} M(C) \xrightarrow{\widehat{[b,s]}} C \,. \tag{5.9}$$

Proof. Recall from Remark 5.19 that we can assume $K = K' + H$ and that the ℓ-interpretation has the form $[b', c^{-1}]$. Furthermore recall from Remark 2.6(2) that ℓ gives rise to a distributive law $\lambda : \widehat{K}(H \times \mathrm{Id}) \to (H \times \mathrm{Id})\widehat{K}$ of the free monad $\widehat{K}$ over the cofree copointed functor $H \times \mathrm{Id}$.

Given an ℓ-srps $e : V(H \times \mathrm{Id}) \to \widehat{K}HM$, we form the (ordinary) ℓ-rps $\overline{e} : V(H \times \mathrm{Id}) \to HM$ by defining

$$\overline{e} = (\ V(H \times \mathrm{Id}) \xrightarrow{\quad e \quad} \widehat{K}HM \xrightarrow[\widehat{K}(HM \times \varphi)]{\widehat{K}\langle HM, \mathrm{inm}M\rangle} \widehat{K}(HM \times (K' + H + V)M)$$

$$\begin{matrix}\widehat{K}(HM \times M) \\ = \widehat{K}(H \times \mathrm{Id})M\end{matrix} \xrightarrow{\lambda M} (H \times \mathrm{Id})\widehat{K}M \xrightarrow{\pi_0 \widehat{K}M} H\widehat{K}M \xrightarrow{H\widehat{\mathrm{inl}}M} HMM \xrightarrow{H\mu} HM\ )\,.$$

Then we verify that solutions of e and $\overline{e}$ are in one-to-one correspondence. Consider the following diagram (we drop the indices denoting components of natural transformations):

We first show that all inner parts except part (i) commute: for part (ii) use the definition of $\overline{e}$, part (iv) is the commutative diagram in (2.4), part (v) trivially commutes, the parts (vi), (vii), (viii) and (ix) commute by the naturality of λ, $\pi_0 : H \times \mathrm{Id} \to H$ and $\widehat{\mathrm{inl}} : \widehat{K} \to M$, respectively, for part (x) use Remark 5.5(3), and part (xi) commutes since $\widehat{[b, s]}$ is the

structure of an Eilenberg-Moore algebra for M. It remains to verify the commutativity of part (iii); we remove $\widehat{\widehat{K}}$ and consider the product components separately: the left-hand component commutes using $c \cdot c^{-1} = \mathrm{id}_C$ and for the right-hand one we have the diagram

$$
\begin{array}{ccc}
HMC & \xrightarrow{\ H\widehat{[b,s]}\ } & HC \\[2mm]
{\scriptstyle \mathrm{inm}M}\downarrow & \searrow{\scriptstyle \mathrm{inm}} & \downarrow{\scriptstyle c^{-1}} \\[2mm]
(K'+H+V)MC \xrightarrow{(K'+H+V)\widehat{[b,s]}} (K'+H+V)C & & \\[2mm]
{\scriptstyle \varphi}\downarrow & \searrow{\scriptstyle [b,s]} & \\[2mm]
MC & \xrightarrow[\ \widehat{[b,s]}\ ]{} & C\,.
\end{array}
$$

Its upper part commutes by the naturality of $\mathrm{inm} : H \to K'+H+V$, for the lower part recall from Notation 2.3 the definition of $\widehat{[b,s]}$ as a homomorphism of algebras for $K' + H + V$, and the right-hand triangle commutes since $b = [b', c^{-1}]$, see Remark 5.19.

Now if s is a solution of $\overline{e}$, then the outside of Diagram (5.10) commutes. Therefore part (i) commutes, and this proves that s is a solution of e. Conversely, if s is a solution of e part (i) commutes and then s is also a solution of $\overline{e}$. By Theorem 5.14, $\overline{e}$ has a unique solution; thus, e has a unique solution, too.

We still need to prove that (5.9) is the structure of a cia for MHM. But this follows from Theorem 3.6. Indeed, from the ℓ-rps $\overline{e}$ we form the abstract GSOS rule $n : (K+V)(H\times\mathrm{Id}) \to HM$ analogously as in Construction 5.7, and the n-interpretation is $[b,s]$ for the unique solution s of $\overline{e}$ (or, equivalently, of e). Now the morphism in (5.9) is the structure k' from the statement of Theorem 3.6. $\qquad\square$

Remark 5.24. Notice that every operation on the terminal coalgebra C definable by a sandwiched ℓ-rps is also definable by an ordinary ℓ-rps. In fact, the proof of the above theorem gives a reduction of a given sandwiched ℓ-rps to an ordinary one with the same solution. However, sandwiched ℓ-rps extend the syntactic format of recursive specifications uniquely specifying operations on C.

5.3. **Modularity, again.** We formulated modularity principles in the Summaries 5.10 and 5.20. The same principles apply in the "sandwiched" case, mutatis mutandis.

Furthermore, we have modularity at the level of cia's:

(1) For ℓ-srps's the same modularity principle as discussed in Remark 5.15(2) applies. Given an ℓ-srps $e : V(H \times \mathrm{Id}) \to \widehat{K}H\widehat{K+V}$, then its unique solution $s : VC \to C$ arises as the unique solution of the ℓ-rps $\overline{e} : V(H \times \mathrm{Id}) \to H\widehat{K+V}$. Thus, as before one can form the abstract GSOS rule $n : F(H \times \mathrm{Id}) \to H\widehat{F}$ for $F = K+V$ and use operations of type F as givens in subsequent n-(s)rps's.

(2) In addition, one can readily formulate and prove a version of Proposition 5.17 for sandwiched ℓ-rps's. We leave this straightforward task as an exercise for the reader.

6. Applications

In this section we present five applications illustrating how to use our results from Section 3–5 to obtain unique solutions of recursive definitions in five different areas of theoretical computer science.

6.1. Process Algebras. Recall Example 2.10(3) where $HX = \mathcal{P}_\kappa(A \times X)$. We shall first explain more in detail how the abstract GSOS rule 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 (we shall denote elements in each component by the corresponding flat terms):

(1) $A \times X$ with elements $a.x$, $a \in A$, for prefixing,
(2) $\coprod_{n<\kappa} X^n$ with the n-the component consisting of elements $\sum_{i=1}^n x_i$, for summation,[2]
(3) $X \times X$ with elements $x_1 | x_2$, for parallel composition,
(4) $\coprod_f X$, where f ranges over functions on the action set A with $\overline{f(a)} = f(\bar{a})$ and $f(\tau) = \tau$, with elements $x[f]$, for relabeling, and
(5) $\coprod_{L \subseteq A \setminus \{\tau\}} X$ with elements $x \backslash L$, for restriction.

The abstract GSOS rule $\ell : K(H \times \mathrm{Id}) \to H\widehat{K}$ is given by the sos rules in Example 2.10(3) in terms of the components of the coproduct $K(H \times \mathrm{Id})$, i.e., for each combinator separately (in the same order as above):

(1) $\ell_X(a, S, x) = \{(a, x)\}$ where $S \subseteq A \times X$,
(2) $\ell_X((S_i, x_i)_{i<n}) = \bigcup_{i<n} S_i$ for every $n < \kappa$, where $(S_i)_{i<n}$ is an n-tuple of sets $S_i \subseteq A \times X$,
(3) $\ell_X(S_1, x_1, S_2, x_2)$ is given by the union of the three sets

$$\{(a, x|x_2) \mid (a, x) \in S_1\}, \quad \{(a, x_1|x) \mid (a, x) \in S_2\} \quad \text{and}$$

$$\{(\tau, x|y) \mid (a, x) \in S_1, (\bar{a}, y) \in S_2 \text{ for some } a \in A \setminus \{\tau\}\},$$

where $S_1, S_2 \subseteq A \times X$,
(4) $\ell_X(S, x) = \{(f(a), y[f]) \mid (a, y) \in S\}$, and
(5) $\ell_X(S, x) = \{(a, y \backslash L) \mid (a, y) \in S, a, \bar{a} \notin L\}$.

The form of these definitions is very similar to the ones given by Aczel [3] in the setting of non-well-founded set theory. We already mentioned the ℓ-interpretation $b : KC \to C$ giving the desired operations on CCS agents, and this gives the two new cia structures for $H\widehat{K}$ and $\widehat{K}H\widehat{K}$ as in Theorems 3.5 and 3.6.

Now let us recall Milner's solution theorem for CCS agents from [38]. 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 system

$$x_i = E_i, \qquad i \in I,$$

of mutually recursive equations. More precisely, let $\sim$ denote strong bisimilarity, and let $E_i[\vec{P}/\vec{x}]$ denote simultaneous substitution of P_j for x_j for every j. Then we have

Theorem 6.1. [38] *There exist, up to $\sim$, unique CCS agents P_i such that $P_i \sim E_i[\vec{P}/\vec{x}]$ holds for each $i \in I$.*

---

[2]The empty sum (for $n = 0$) is denoted by 0 as usual.

It is easy to see that this theorem is a consequence of our Theorem 2.16 because a system $x_i = E_i$ where each variable is weakly guarded is essentially the same as a map $X \to \widehat{K}H\widehat{K}X$, where $X = \{x_i \mid i \in I\}$.

Furthermore, our Theorem 3.6 allows us to obtain unique solutions of flat equation morphisms $X \to \widehat{K}H\widehat{K}X + C$. The extra summand C allows us to use constant agents in recursive specifications.

Example 6.2. Consider the recursive equation

$$x = a.(x|c) + b.0$$

from the introduction, where 0 denotes the empty sum (i. e. the "inactive agent"). For any set X, we identify elements of $H\widehat{K}X = \mathcal{P}_\kappa(A \times \widehat{K}X)$ with terms of the form

$$\sum_{i < \kappa} a_i.t_i$$

(modulo associativity, commutativity, idempotency and the unit laws for the sum), where $a_i \in A$ and t_i is a process term on variables from X for every $i < \kappa$.

So the above equation clearly gives an ℓ-equation $f_0 : \{x\} \to H\widehat{K}\{x\}$, and so we have its unique solution $f_0^\dagger(x) = P$ in C from Theorem 2.12. Now consider the following system

$$y = b.(y + z)|a.z \qquad z = P. \tag{6.1}$$

This gives a flat equation morphism $\{y, z\} \to \widehat{K}H\widehat{K}\{y, z\} + C$, which has a unique solution in C by Theorem 3.6. Next recall Remark 3.3 and notice that the latter equation morphism has the form $f^\dagger \bullet e$, where $e : \{y, z\} \to \widehat{K}H\widehat{K}\{y, z\} + \{x\}$ is given by two equations: the first equation in (6.1) and $z = x$, and $f : \{x\} \to \widehat{K}H\widehat{K}\{x\} + C$ arises from f_0 by forming

$$f = \mathsf{inl} \cdot (\eta H\widehat{K})_{\{x\}}$$

as explained in Section 4.1(1). So we see that we obtain the same solution when we use the constant $P \in C$ in the recursive equation (6.1) in lieu of forming the composed system

$$x = a.(x|c) + b.0 \qquad y = b.(y + z)|a.z \qquad z = a.(x|c) + b.0$$

that corresponds to the equation morphism $f \blacksquare e$.

We now turn to an applications of Theorem 5.23 that shows how to uniquely define new process combinators. Suppose we want to define the binary combinator "alt" which performs alternation of two processes. For its definition we shall first need another binary combinator, sequential composition of two processes (denoted by the infix ";"). This is defined by the following operations rules in GSOS format:

$$\frac{E \xrightarrow{a} E'}{E; F \xrightarrow{a} E'; F} \qquad \frac{F \xrightarrow{a} F'}{0; F \xrightarrow{a} F'} \,.$$

Here we suppose that the latter combinator is already included in our basic calculus—more precisely, we add a sixth coproduct component $X \times X$ to K for sequential composition in the above definition and complete ℓ above by

$$\ell_X(S_1, x_1, S_2, x_2) = \begin{cases} \{(a, x; x_2) \mid (a, x) \in S_1\} & \text{if } S_1 \neq \emptyset \\ S_2 & \text{if } S_1 = \emptyset \end{cases}$$

for this coproduct component. The ℓ-interpretation then gives indeed the desired combinator for sequential composition. Moreover, as a consequence of Theorem 2.12 (now applied to

the extended K), we see that that Theorem 6.1 still holds for the calculus including this sixth combinator. Now for this extended ℓ we give a sandwiched ℓ-rps $e : V(H \times \mathrm{Id}) \to \widehat{K}H\widehat{K + V}$, where $VX = X \times X$, in order to define the combinator alt:

$$e_X(S_1, x_1, S_2, x_2) = \begin{cases} S_1; \{(a, x; \mathsf{alt}(x_1, x_2)) \mid (a, x) \in S_2\} & \text{if } S_2 \neq \emptyset \\ \{(a, x; \mathsf{alt}(x_2, x_1)) \mid (a, x) \in S_1\} & \text{if } S_1 \neq \emptyset, S_2 = \emptyset \\ \emptyset & \text{if } S_1 = S_2 = \emptyset. \end{cases}$$

Notice that the term in the first line of this definition does not lie in $H\widehat{K + V}(X)$, so Theorem 5.14 cannot be applied. Notice also that the above definition of e cannot be directly translated into an operational rule in GSOS format as for the sequential composition before. However, e could first be transformed into an ordinary ℓ-rps using the construction from the proof of Theorem 5.23 and then translated to a set of operational rules in GSOS format. More directly, observe that the term in the first line does lie in $\widehat{K}H\widehat{K + V}(\mathrm{X})$, so Theorem 5.23 tells us that e has a unique solution $s : C \times C \to C$. It is not difficult to see that $s = \mathsf{alt}$ is the desired alternation combinator. Furthermore, Theorem 5.23 tells us that s extends the cia structure on C from Theorem 3.6 applied to the abstract GSOS rule ℓ. And, as before, this means that Theorem 6.1 remains true for the calculus extended by sequential composition and alternation of processes, without further work.

6.2. Streams. Recall from Example 2.10(2) that here we take $HX = \mathbb{R} \times X$ and we have $C = \mathbb{R}^\omega$ with the structure given by $\langle \mathsf{hd}, \mathsf{tl} \rangle : C \to \mathbb{R} \times C$. Rutten gives in [42] a general theorem for the existence of the solution of systems of behavioral differential equations. We will now recall this result and show that it is a special instance of our Theorem 5.14. For a system of behavioral differential equations one starts with the signature Σ of all the operations to be specified. One uses an infinite supply of variables, and for each variable x there is also a variable x' and a variable $x(0)$ (also written as x_0). For each operation symbol f from Σ one specifies

$$f(x_1, \ldots, x_n)_0 = h_f(x_1(0), \ldots, x_n(0)) \qquad f(x_1, \ldots, x_n)' = t_f, \qquad (6.2)$$

where h_f denotes a function from $\mathbb{R}^n$ to $\mathbb{R}$ and t_f is a term built from operation symbols from Σ on variables x_i, x_i' and $x_i(0)$, $i = 1, \ldots, n$. Theorem A.1 of [42] asserts that for every f from Σ there exists a unique function $(\mathbb{R}^\omega)^n \to \mathbb{R}^\omega$ satisfying the equation (6.2) above.

   We shall now show that every system of behavioral differential equations (6.2) gives rise to an ℓ-rps for a suitable abstract GSOS rule ℓ. To this end let $KX = \mathbb{R}$ be the constant functor and let

$$\ell = (K(H \times \mathrm{Id}) \xrightarrow{\ell'} HK \xrightarrow{H\kappa} H\widehat{K})$$

with ℓ' given by $\ell'_X(r) = (r, 0)$. Then the ℓ-interpretation $b : \mathbb{R} \to C$ assigns to every $r \in \mathbb{R}$ the stream $b(r) = (r, 0, 0, \ldots)$.

   Now given the system (6.2) let V be the polynomial functor associated with Σ, see Example 2.4. Notice that $K + V$ is the polynomial functor of the signature Σ extended with a constant symbol r for every real number r. We translate the system (6.2) into an ℓ-rps $e : V(H \times \mathrm{Id}) \to H\widehat{K + V}$ as follows. For every f from Σ the corresponding component of e_X is defined by

$$e_X((r_1, x_1', x_1), \ldots, (r_n, x_n', x_n)) = (h_f(r_1, \ldots, r_n), \overline{t_f}),$$

where the term $\overline{t_f} \in \widehat{K + V}(X)$ is obtained by replacing in t_f all variables $x_i(0)$ by the constant r_i. Notice also, that here $h_f(r_1, \ldots, r_n)$ is a real number (the value of h_f at $(r_1, \ldots, r_n)$) whereas in (6.2) we have formal application of h_f to the variables $x_i(0)$.

It is now straightforward to verify that a solution of e in C corresponds precisely to a solution of the system (6.2). Thus, we obtain from Theorem 5.14 the

Theorem 6.3 ([42])**.** *Every system of behavioral differential equations has a unique solution.*

Example 6.4. For the system given by (2.5) and (5.2) we have $VX = X \times X + X \times X$ and e given componentwise as follows: for the $+$ component we have

$$e_X((r, x', x), (s, y', y)) = (r + s, x' + y') \tag{6.3}$$

and for the $\otimes$ component we have

$$e_X((r, x', x), (s, y', y)) = (r \cdot s, (x \otimes y') + (x' \otimes y)). \tag{6.4}$$

Observe that the systems of behavioral differential equations do not distinguish between given operations and newly defined ones. However, our result in Theorem 5.14 allows us to make this distinction, and the modularity principle for solutions of ℓ-rps's (cf. Summary 5.20) means that operations specified by behavioural differential equations may be used in subsequent behavioral differential equations as given operations in the terms t_f from (6.2). We believe this modularity of unique solutions for behavioral differential equations is a new result.

Example 6.5. Take $VX = X \times X$ and the ℓ-rps $e : V(H \times \mathrm{Id}) \to H\widehat{K + V}$ given by (6.3) whose solution is the operation of stream addition $+ : VC \cdot C$. As shown in Construction 5.7, ℓ and e yield the abstract GSOS rule $n : F(H \times \mathrm{Id}) \to H\widehat{F}$ for $F = K + V$. Now let $V_1 X = X \times X$. Then (6.4) yields an n-rps $e_1 : V_1(H \times \mathrm{Id}) \to H\widehat{F + V_1}$ whose solution is the shuffle product $\otimes : V_1 C \to C$.

Next, we present an example illustrating Proposition 5.17.

Example 6.6. Continuing the previous example, consider the convolution product of streams specified by
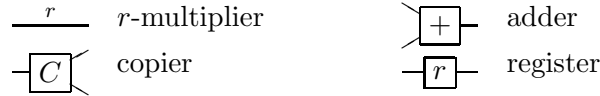
$$(\sigma \times \tau)_0 = \sigma_0 \cdot \tau_0 \qquad\qquad (\sigma \times \tau)' = (\sigma' \times \tau + \sigma_0 \times \tau'),$$

see [42]. Let $V_2 X = X \times X$ and let the n-rps $e_2 : V_2(H \times \mathrm{Id}) \to H\widehat{F + V_2}$ be given by

$$(e_2)_X((r, x', x), (s, y', y)) = (r \cdot s, (x' \times y) + (r \times y')). \tag{6.5}$$

Notice that this illustrates why we introduced the constants r; in this way we are able to deal with σ_0 in the equation for $(\sigma \times \tau)'$. Then the unique solution of e_2 is the convolution product as expected. Proposition 5.17 asserts that the extended cia structure for $H\widehat{F}$, $F = K + V + V_1 + V_2$ on C does not depend on the order of taking the solution of (6.4) and (6.5)—either way this is given by the constants coming from b and the operations of stream addition as well as convolution and shuffle product.

Stream circuits. We now turn to another method to define operations on streams—stream circuits [43], which are also called (signal) flow graphs in the literature. We shall demonstrate that specification of operations by stream circuits 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:

$$\sigma \longrightarrow \boxed{+} \longrightarrow \boxed{C} \rightarrow f(\sigma) \qquad (6.6)$$

It defines the following unary operation on stream:

$$f(\sigma) = (1 + \sigma_0, 1 + \sigma_0 + \sigma_1, 1 + \sigma_0 + \sigma_1 + \sigma_2, \dots).$$

For our treatment we shall consider the operations presented by r-multipliers, adders and registers as givens. So let K be the polynomial functor associated with the signature Σ given by these operations (copying will be implicit via variable sharing). In symbols, $KX = \mathbb{R} \times X + X \times X + \mathbb{R} \times X$. Our given operations are defined by the behavioral differential equations:

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

As explained above these definitions easily give rise to an abstract GSOS rule $\ell : K(H \times \mathrm{Id}) \to H\widehat{K}$ (see also [13, Section 3.5.1]). We then get the ℓ-interpretation in C and the corresponding extended cia structures by Theorems 3.5 and 3.6. A stream circuit is called *valid* if every loop passes through at least one register. It is well-known that every finite valid stream circuit with one input and one output defines a unique stream function (see [43]). Of course, a similar result holds for more than one input and output, and we present here a new proof of this result based on our Theorem 5.14.

Theorem 6.7. *Every finite valid stream circuit defines a unique stream function at every output.*

Proof. Let a finite valid stream circuit be given. We explain how to construct an ℓ-rps from the circuit. Notice first that the wires in a circuit can be regarded as directed edges (cf. (6.6)). We take for every register R in our circuit an operation symbol g_R and define its arity as the number of inputs that can be reached by following all possible paths from R backwards through the circuit. Similarly, we take for every output edge O of the circuit an operation symbol f_O with the arity obtained in the same way. Let Σ be the signature of all f_O and g_R, and let V be the corresponding polynomial functor for Σ. To give an ℓ-rps $e : V(H \times \mathrm{Id}) \to H\widehat{K + V}$ it suffices to give a natural transformation $e' : VH \to H\widehat{K + V}$

and to define $e = e' \cdot V\pi_0$, where $\pi_0 : H \times \mathrm{Id} \to H$ is the projection. To obtain e', we give for each n-ary symbol s from Σ an assignment

$$s(r_1.x_1, \ldots, r_n.x_n) \mapsto (r, t)$$

where $r_1, \ldots, r_n, r \in \mathbb{R}$ and t is a term built from symbols of Σ and of the signature Γ of basic circuit operations using the variables $x_1, \ldots, x_n$. Notice that the arguments of s stand for generic elements (r_i, x_i) from HX for some set X and that r may depend on r_i and t may contain operation symbols $r_i.-$. We now show how to define the above assignment for each operation symbol g_R. Suppose that the register R has the initial value r. Then

$$g_R(r_1.x_1, \ldots, r_n.x_n) \mapsto (r, t_R),$$

and we now explain how to obtain t_R: in order to construct the term t_R one starts in R and traverses from there every possible path in the circuit backwards (i.e., one follows edges from inputs to outputs of basic circuits) adding for every basic stream circuit the corresponding operation symbol to t_R until

(1) an input edge corresponding to some argument $r_i.x_i$ is met, or
(2) some register is met.

More precisely, we construct t_R as a $(\Sigma + \Gamma)$-tree: we follow the input edge of R backwards until we reach either the output wire of an r-multiplier, the output wire of an adder, an input wire of the whole circuit or the output wire of a register. For an r-multiplier or an adder we add a node to t_R labeled by the corresponding operation symbol and continue this process for each input node of the r-multiplier or adder constructing the corresponding subtrees of t_R. For an input wire corresponding to $r_i.x_i$ add a node labeled by the prefix operation $r_i.-$ and below that a leaf labeled by x_i; for a register S add the tree (of height 2) given by $g_S(r_{i_1}.x_{i_1}, \ldots, r_{i_k}.x_{i_k})$, where the $r_{i_j}.x_{i_j}$ correspond to those input wires of the circuit backwards reachable from the register S. (Notice that these arguments of g_S form a subset of the arguments $\{ r_1.x_1, \ldots, r_n.x_n \}$ of g_R since every input that is backwards reachable from S is also backwards reachable from R. Also notice that copiers are ignored while forming t_R.) Since the given circuit C is valid we have indeed constructed only a finite tree, whence a term t_R.

We still need to define the assignment corresponding to e' for output symbols f_O:

$$f_O(r_1.x_1, \ldots, r_n.x_n) \mapsto (r, t_O).$$

We first form the tree t'_O in essentially the same way as t_R for a register R with the difference that for every input wire and for every register we just insert an unlabeled leaf for the moment. To obtain r, label every leaf of t'_O corresponding to the input $r_i.x_i$ by r_i and every leaf corresponding to a register by its initial value; now evaluate the corresponding term to get r. In order to get t_O one replaces leaves of t'_O corresponding to inputs $r_i.x_i$ by x_i, and register leaves are replaced by the second components t_S from the right-hand sides of the equations for $g_S(r_{i_1}.x_{i_1}, \ldots, r_{i_k}.x_{i_k})$.

Finally, the unique solution of e yields a unique operation f_O on streams for every output O. By construction this is the operation computing the stream circuit. $\qquad\square$

The modularity principle for the unique solution of an ℓ-rps which we discussed in Summary 5.20 yields *modularity of stream circuits*: every stream circuit can be used as a building block as if it were a basic operation in subsequent stream circuits. And Theorem 6.7 remains valid for the extended circuits.

Example 6.8. The proof of Theorem 6.7 essentially gives a translation of an arbitrary finite valid stream circuit into an ℓ-rps. We demonstrate this on the circuit given in (6.6) 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. 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, i. e., one evaluates $t'[r/x_I, 1/x_R]$. And for the second component we replace in t' the input by x and the register by the second component of the right-hand side of the definition above of $g(r.x)$, i. e., one forms $t'[x/x_I, r.x + g(r.x)/x_R]$. The two equations above are easily seen to yield an ℓ-rps $e : V(H \times \mathrm{Id}) \to H\widehat{K + V}$, 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 $f_C, g_C : C \to C$, and f_C is precisely the function computed by the circuit (6.6) and g_C is the stream function output by the register in (6.6). By the modularity of stream circuits explained above, we can use f (and also g) as "black-boxes" in subsequent stream circuits.

6.3. Infinite Trees. Rutten and Silva [45] developed behavioral differential equations for infinite trees and proved a unique solution theorem for them. Here we shall show that we obtain their theorem as a special instance of our Theorem 5.14. The work is similar to what we saw in Theorem 6.3 for streams.

Let $HX = X \times \mathbb{R} \times X$. The terminal coalgebra C for H consists of all infinite binary trees with nodes labeled in $\mathbb{R}$, and the terminal coalgebra structure $c : C \to C \times \mathbb{R} \times C$ assigns to a tree the triple (t_L, r, t_R) where r is the node label of the root of t and t_L and t_R are the trees rooted at the left-hand right-hand child nodes of the root of t. Single trees (constants) and operations on trees can be specified by behavioral differential equations. For example,

$$\mathsf{pi}(\varepsilon) = \pi \qquad \begin{aligned} \mathsf{pi}_L &= \mathsf{pi} \\ \mathsf{pi}_R &= \mathsf{pi} \end{aligned}$$

specifies the tree with every node labeled by the number π. For every real number r we have the constant $[r]$ specified by

$$[r](\varepsilon) = r \qquad \begin{aligned}{} [r]_L &= [0] \\ [r]_R &= [0]. \end{aligned}$$

The nodewise addition of numbers stored in the nodes of the trees t and s is defined by

$$(t + s)(\varepsilon) = t(\varepsilon) + s(\varepsilon) \qquad \begin{aligned} (t + s)_L &= t_L + s_L \\ (t + s)_R &= t_R + s_R. \end{aligned}$$

See [45] for further and more exciting examples.

In general a system of behavioral differential equations is specified as follows. Again, we assume an infinite supply of syntactic variables. For every variable x we have the notational variants x_L, x_R and also $x(\varepsilon)$. Furthermore, let Σ be a signature of operations to be specified. For each operation symbol f from Σ of arity n we provide equations of the form

initial value	differential equations
$(f(x_1,\ldots,x_n))(\varepsilon) = c_f(x_1(\varepsilon),\ldots,x_n(\varepsilon))$	$f(x_1,\ldots,x_n)_L = t_1$ $f(x_1,\ldots,x_n)_R = t_2$

$$(6.7)$$

where c_f denotes a function $\mathbb{R}^n \to \mathbb{R}$ and t_1 and t_2 are Σ-terms on the variables $x_1,\ldots,x_n$ and their three notational variants.

Theorem 6.9. ([45], Theorem 2) *Every system (6.7) of behavioral differential equations has a unique solution, i.e., for every f from Σ there exists a unique function $f : C^n \to C$ satisfying (6.7) (denoted by the same symbol).*

We present a new, short proof of this result based on Theorem 5.14. Let $KX = \mathbb{R}$ be the constant functor, and let the abstract GSOS rule ℓ be

$$\ell = (\ K(H \times \mathrm{Id}) \xrightarrow{\ell'} HK \xrightarrow{H\kappa} H\widehat{K}\ )$$

where the natural transformation ℓ' is given by $\ell'_X(r) = (0, r, 0)$. Then the ℓ-interpretation is $b : \mathbb{R} \to C$ with $b(r) = [r]$. Let V be the polynomial functor associated with Σ. Every system (6.7) gives an ℓ-rps $e : V(H \times \mathrm{Id}) \to H\widehat{K + V}$ as follows: let e be given on each component corresponding to f from Σ by

$$e_X(((x_1)_L, r_1, (x_1)_R, x_1), \ldots, ((x_n)_L, r_n, (x_n)_R, x_n)) = (\overline{t_1}, c_f(r_1, \ldots, r_n), \overline{t_2})\,,$$

where $\overline{t_i}$ is obtained from t_i by replacing each $x_i(\varepsilon)$ by the corresponding constant r_i. The solutions of e in C correspond precisely to solutions of (6.7); thus, Theorem 6.9 follows from Theorem 5.14.

In addition, we have again a modularity principle (cf. Summary 5.20): operations specified by behavioral differential equations may be used as givens in subsequent behavioral differential equations.

6.4. Formal Languages. Recall Example 2.10(3); 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 standard operations on formal languages can be defined in a modular way using Theorem 5.14. Working in a bialgebraic setting, Jacobs [24] shows that these operations can be defined as interpretations of one abstract GSOS rule (or distributive law) in C. However, this account does not explain why one may define these operations in a step-by-step fashion by successive recursive definitions. This is the added value of Theorem 5.14.

We start with the functor $K_0 = C_\emptyset$ (that means, we start from scratch with no given operations) and with $\ell_0 : C_\emptyset(H \times \mathrm{Id}) \to H\widehat{C_\emptyset} = H$ given by the empty maps. The corresponding interpretation is the empty map $b : \emptyset \to C$, and $\widehat{b}$ is then the identity on C. Thus, the cia structure for $H\widehat{K_0}$ on C given by Theorem 3.5 is simply the initial cia (C, c^{-1}) for H. At each subsequent step we are given a functor K_i and an abstract GSOS rule

$$\ell_i : K_i(H \times \mathrm{Id}) \to H\widehat{K_i} \quad \text{with its interpretation} \quad b_i : K_i C \to C\,.$$

We then give an ℓ_i-rps
$$e_i : V_i(H \times \mathrm{Id}) \to H\widehat{K_i + V_i},$$
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
$$\ell_{i+1} = [H\widehat{\mathsf{inl}} \cdot \ell_i, e_i] : K_{i+1}(H \times \mathrm{Id}) \to H\widehat{K_{i+1}},$$
where $\widehat{\mathsf{inl}} : \widehat{K_i} \to \widehat{K_{i+1}}$ is the monad morphism induced by $\mathsf{inl} : K_i \to K_{i+1}$ (cf. Notation 5.4(1)); so the formation of ℓ_{i+1} from ℓ_i corresponds precisely to forming n from ℓ in Construction 5.7. By induction it is easy to see that the ℓ_{i+1}-interpretation is
$$b_{i+1} = [s_j]_{j=0,\dots,i} : K_{i+1}C \to C.$$
And this gives an extended cia
$$c^{-1} \cdot H\widehat{b}_{i+1} : H\widehat{K_{i+1}}(C) \to C$$
by Theorem 3.5.

As a first step we define constants in C for $\emptyset$, $\{\varepsilon\}$, and $\{a\}$, for each $a \in A$, as solutions of an ℓ_0-rps. We express this as an ℓ_0-rps as follows: take the functor $V_0 X = 1 + 1 + A$ corresponding to these languages. We define $e_0 : V_0(H \times \mathrm{Id}) \to H\widehat{K_0 + V_0} = H\widehat{V_0}$ componentwise. We write for every set X, $\emptyset$ for $\mathsf{inj}_1(*) \in V_0 X$ and ε for $\mathsf{inj}_2(*) \in V_0 X$. Then $(e_0)_X$ is given by the assignments

$$\begin{aligned}
\emptyset &\mapsto ((\emptyset)_{a \in A}, 0) \\
\varepsilon &\mapsto ((\emptyset)_{a \in A}, 1) \\
a &\mapsto ((t_b)_{b \in A}, 0), \qquad \text{where } t_b = \begin{cases} \varepsilon & \text{if } b = a \\ \emptyset & \text{else}. \end{cases}
\end{aligned} \tag{6.8}$$

It is now straightforward to check that the unique solution s_0 of e_0 yields the desired constants in 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 $\ell_1 = [H\widehat{\mathsf{inl}} \cdot \ell_0, e_0]$ as above with interpretation $b_1 = s_0$. Let
$$V_1 X = X \times X + X \times X + X$$
be the polynomial functor corresponding to two binary symbols $\cup$ and $\cap$ and one unary one $\overline{(-)}$. We give the ℓ_1-rps
$$e_1 : V_1(H \times \mathrm{Id}) \to H\widehat{K_1 + V_1}$$
componentwise in the form of the three assignments in (6.9) below. We write $((x_a), j, x)$ for elements of $HX \times X$, where (x_a) is an $|A|$-tuple, i.e., an element of X^A. We also write elements of $V_2 Z$, $Z = HX \times X$, as flat terms $z_1 \cup z_2$, $z_1 \cap z_2$ and $\overline{z}$ on the left-hand side for the three components of $(e_1)_Z$:

$$\begin{aligned}
((x_a), j, x) \cup ((y_a), k, y) &\mapsto ((x_a \cup y_a), j \vee k) & \overline{((x_a), j, x)} &\mapsto ((\overline{x_a}), \neg j) \\
((x_a), j, x) \cap ((y_a), k, y) &\mapsto ((x_a \cap y_a), j \wedge k)
\end{aligned} \tag{6.9}$$

where $\vee$, $\wedge$ and $\neg$ are the evident operations on $2 = \{0, 1\}$. The corresponding unique solution $s_1 : V_1 C \to C$ is easily checked to provide the desired operations extending the cia structure on C.

The next step adds (language) concatenation to the cia structure on C. For this let
$$V_2 X = X \times X \qquad \text{and let} \qquad e_2 : V_2(H \times \mathrm{Id}) \to H\widehat{K_2 + V_2}$$

be given by the assignment (where elements of $V_2 Z$ are written as $z_1 \cdot z_2$ on the left-hand side)

$$((x_a), j, x) \cdot ((y_a), k, y) \mapsto ((t_a), j \wedge k) \qquad \text{where } t_a = \begin{cases} (x_a \cdot y) \cup y_a & \text{if } j = 1 \\ x_a \cdot y & \text{else}. \end{cases} \qquad (6.10)$$

As the final step we add the Kleene star operation by taking

$$V_3 X = X \qquad \text{and} \qquad e_3 : V_3(H \times \text{Id}) \to H\widehat{K_3 + V_3}$$

to be given by

$$e_3((x_a), j, x) = ((x_a \cdot x^*), 1)$$

with the unique solution $s_3 = (-)^* : C \to C$. 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. This corresponds to the fact that for all languages L, $(L^*)^a = L^a \cdot L^*$. Its unique solution $s_2 : C \times C \to C$ is the concatenation operation.

Remark 6.10. There are many further operations on formal languages that are definable by ℓ-rps's, including the following ones:

- prefixing $a.L = \{aw \mid w \in L\}$ for any $a \in A$;
- the operation given by $c^{-1} : C^A \times 2 \to C$ (see Remark 5.19)

$$((L_a), j) \mapsto \begin{cases} \bigcup_{a \in A} a.L_a & \text{if } j = 0 \\ \bigcup_{a \in A} a.L_a \cup \{\varepsilon\} & \text{else}; \end{cases}$$

- $\mathsf{shuffle}(L_1, L_2) = \bigcup_{w_1 \in L_1, w_2 \in L_2} \mathsf{shuffle}(w_1, w_2)$ where $\mathsf{shuffle}(w_1, w_2)$ is the usual operation merging the words w_1 and w_2.

We leave it to the reader to work out the details. An example of an operation on languages not definable by any ℓ-rps (or abstract GSOS rule) is the language derivative $L \mapsto L^a = \{w \mid aw \in L\}$ for every $a \in A$; the argument is similar as for the non-definablity of the tail operations on streams in Example 5.16, and so we leave those details to the reader, too.

Contex-free grammars. Next we show how context-free grammars in Greibach Normal Form and their generated languages are special instances of flat equations of the form $e : X \to \widehat{K}H\widehat{K}$ and their unique solutions in C for a suitable functor K. (Note that these flat equations do not involve elements of C; that is, we do not need equations of the form $e : X \to \widehat{K}H\widehat{K} + C$.) A coalgebraic description of context-free grammars in Greibach normal form has recently been given by Bonsangue, Rutten and Winter [48], and previously, a coalgebraic approach to context-free grammars was given by Hasuo and Jacobs [21]. Our approach here is completely different.

Recall (e.g. from [23]) that a *context-free grammar* is a four-tuple $G = (A, N, P, S)$ where A is a non-empty finite set of *terminal* symbols, N a finite set of *non-terminal* symbols, $P \subseteq N \times (A + N)^*$ is a finite relation with elements called *production rules* of G, and $S \in N$ is the starting symbol. As usual we write $n \to w$ for $(n, w) \in P$. A context-free grammar G is in *Greibach Normal Form* (*GNF*, for short) if all its production rules are of the form $n \to aw$ with $a \in A$ and $w \in N^*$. The language *generated* by a context-free grammar G is the set of all words over A that arise by starting with the string S and repeatedly substituting substrings according to the production rules of the grammar, and eliminating ε from the string whenever it occurs.

To see that context-free grammars in GNF yield flat equation morphisms, we consider the constant $\emptyset$ and the operations of union and concatenation as given operations. More precisely, let

$$KX = 1 + X \times X + X \times X$$

be the polynomial functor corresponding to $\emptyset$, $\cup$ and $\cdot$, and let $\ell : K(H \times \mathrm{Id}) \to H\widehat{K}$ be the abstract GSOS rule given by the corresponding assignments in (6.8), (6.9) and (6.10) with the interpretation $b : KC \to C$ given as desired. By Theorem 3.6 we obtain the cia structure $k' : \widehat{K}H\widehat{K}C \to C$. Now observe that a flat equation morphism assigns to each $x \in X$ either an element $e(x) \in C$ or $e(x)$ corresponds to a term of given operations in $H\widehat{K}X$.

We now show that for every context-free grammar in GNF there is a flat equation morphism $e : X \to H\widehat{K}X$ with X finite whose solution $e^\dagger : X \to X$ has the property that $e^\dagger(S)$ is the language of the grammar. (Once again, note that each $e(x)$ will belong to $H\widehat{K}X$, not just to the larger set $H\widehat{K}X + C$.)

Construction 6.11. Let $G = (A, N, P, S)$ be a context-free grammar in GNF. We define the flat equation morphism $e_G : X \to \widehat{K}H\widehat{K}X$. The set X is simply the set N of nonterminals of G, and e_G is the following map: for $n \in N$ for which there is no production rule $n \to aw$ in P we take $e_G(n) = \emptyset$, the constant term in $\widehat{K}(H\widehat{K}X)$. Otherwise for each production rule $r = n \to aw$ in P with $n \in N$ on the left-hand side we define the term $t_r \in H\widehat{K}X$ as

$$t_r = \begin{cases} a.(n_1 \cdot n_2 \cdot \cdots \cdot n_k) & \text{if } w = n_1 n_2 \cdots n_k \text{ and } n_1, \ldots, n_k \in N, \\ a.\emptyset & \text{if } w = \varepsilon \end{cases}$$

using the concatenation operation. Recall from Remark 6.10 the notation $a.t$ to see that $t_r \in H\widehat{K}X$. We define $e_G(n)$ as (the term in $\widehat{K}(H\widehat{K}X)$ representing) the "union" of all right-hand sides of production rules $r_i = n \to aw_i$, $i = 1, \ldots, l$, in P; so in symbols we have

$$e_G(n) = t_{r_1} \cup t_{r_2} \cup \cdots \cup t_{r_l} .$$

The point of using grammars in GNF is that each t_r really belongs to $H\widehat{K}X$, and so each $e_G(n)$ belongs to $K(H\widehat{K}X)$, hence to $\widehat{K}(H\widehat{K}X)$.

It is not difficult to see that the language generated by the grammar G is precisely the language $e_G^\dagger(S)$, where S is the starting symbol of G. So as a consequence of Theorem 3.6 we see that the language generated by G arises as the unique solution of the flat equation morphism e_G.

Analogously, it is also possible to translate right-linear grammars (which are a special case of context-free grammars generating regular languages) into flat equation morphisms using the constant empty and empty-word languages as well as union as the given operations. Again Theorem 3.6 implies that there is a unique solution which yields the language generated by the given grammar by the translation.

Remark 6.12. We have seen that by defining operations via ℓ-rps's (or ℓ-srps's) we obtain cia structures for $H\widehat{K}$ (or $\widehat{K}H\widehat{K}$) on C. It is interesting to ask what formal languages can arise as solutions of ℓ-equations $e : X \to H\widehat{K}X$ (or sandwiched ones $e : X \to \widehat{K}H\widehat{K}X$) according to Theorems 2.12 and 2.16 if X is *finite*. Not surprisingly one obtains precisely the regular languages for stages $i = 0, 1, 2$ in our definition process above, i.e., when we add the constant languages $\emptyset$, $\{\varepsilon\}$ and $\{a\}$ for every $a \in A$, and the operations $\cup$, $\cap$ and

$\overline{(-)}$. But adding concatenation one obtains non-regular languages: if for $i = 3$ one restricts to using union and concatenation in the terms in $\widehat{K}_3 H \widehat{K}_3$, the flat equation morphisms essentially correspond to context-free grammars in GNF. However, using intersection and/or complement allows one to obtain non-context-free languages as solutions. Precisely what class of languages can be defined by (sandwiched) ℓ-equations using different combinations of operations remains the subject of further work.

6.5. Non-well-founded Sets. Finally, we come to an application not directly related to computation. The theory of non-well-founded sets originated as a framework for providing the semantics of general circular definition. For background on non-well-founded sets, the antifoundation axiom (AFA), and classes, please see the books [3, 14]. We work here on the category $\mathcal{A} = \mathsf{Class}$ of classes. The results of Section 5 hold true for Class since every endofunctor of Class has terminal coalgebras and free algebras (see [6]).

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

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

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

We will now define three additional operations on C:

- 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 given by

$$KX = X + (X \times X) + (X \times X) + \mathcal{P}X + \mathcal{PP}X;$$

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

$$
\begin{aligned}
\ell' \cdot \mathsf{inj}_1 \mathcal{P} &= (\mathcal{P} \xrightarrow{\;p\;} \mathcal{PP} \xrightarrow{\mathcal{P}\mathsf{inj}_4} \mathcal{P}K) \\
\ell' \cdot \mathsf{inj}_2 \mathcal{P} &= (\mathcal{P} \times \mathcal{P} \xrightarrow{op\mathcal{P}} \mathcal{PPP} \xrightarrow{\mathcal{P}\mathsf{inj}_5} \mathcal{P}K) \\
\ell' \cdot \mathsf{inj}_3 \mathcal{P} &= (\mathcal{P} \times \mathcal{P} \xrightarrow{cp} \mathcal{P}(\mathrm{Id} \times \mathrm{Id}) \xrightarrow{\mathcal{P}\mathsf{inj}_2} \mathcal{P}K)
\end{aligned}
\qquad
\begin{aligned}
\ell' \cdot \mathsf{inj}_4 \mathcal{P} &= (\mathcal{PP} \xrightarrow{\mathcal{P}\mathsf{inj}_4} \mathcal{P}K) \\
\ell' \cdot \mathsf{inj}_5 \mathcal{P} &= (\mathcal{PPP} \xrightarrow{\mathcal{P}\mathsf{inj}_5} \mathcal{P}K)
\end{aligned}
$$

Then ℓ' yields an abstract GSOS rule

$$\ell = (\, K(\mathcal{P} \times \mathrm{Id}) \xrightarrow{K\pi_0} K\mathcal{P} \xrightarrow{\ell'} \mathcal{P}K \xrightarrow{\mathcal{P}\kappa} \mathcal{P}\widehat{K} \,).$$

Let $b : KC \to C$ be the ℓ-interpretation in C. Let us write $b_1, \ldots, b_5$ for the components of b, so $b_i = b \cdot (\mathsf{inj}_i)_C$. To obtain explicit formulas for these, we use Diagram (2.3) and the above definitions to write:

$$
\begin{aligned}
c \cdot b_1 &= \mathcal{P}b_4 \cdot p_C \cdot c & c \cdot b_4 &= \mathcal{P}b_4 \cdot \mathcal{P}c \\
c \cdot b_2 &= \mathcal{P}b_5 \cdot op_{\mathcal{P}C} \cdot (c \times c) & c \cdot b_5 &= \mathcal{P}b_5 \cdot \mathcal{P}^2 c \\
c \cdot b_3 &= \mathcal{P}b_2 \cdot cp_C \cdot (c \times c) &
\end{aligned}
$$

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

$$b_1 = c^{-1} \cdot \mathcal{P}c^{-1} \cdot p_C \cdot c,$$
$$b_2 = c^{-1} \cdot \mathcal{P}b_5 \cdot op_{\mathcal{P}C} \cdot (c \times c), \quad \text{and}$$
$$b_3 = c^{-1} \cdot \mathcal{P}b_2 \cdot cp_C \cdot (c \times c).$$

In words, b_4 and b_5 are the identity, and b_1, b_2 and b_3 are as desired.

By Theorem 3.5, we have a cia structure $(C, c^{-1} \cdot \mathcal{P}\widehat{b})$ for the composite $\mathcal{P}\widehat{K}$.

Remark 6.13. We could have obtained the various operations on C 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 5.14 as in the previous section on formal languages. We decided against this, to keep the presentation short.

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 [3, 14]. For example, one may solve the system

$$x = \{\mathcal{P}(y)\} \qquad y = \{y \times y, z\} \qquad z = \emptyset,$$

which gives rise to a flat equation morphism $e : X \to \mathcal{P}\widehat{K}X + C$ where $X = \{x, y, z\}$. The unique solution of e satisfies $e^\dagger(z) = \emptyset$ and assigns to y the non-well-founded set $e^\dagger(y)$ containing two elements: $\emptyset$ and the cartesian product of $e^\dagger(y)$ with itself. And $e^\dagger(x)$ is the singleton non-well-founded set containing the powerset of $e^\dagger(y)$ as its only element.

Finally, let us show how to obtain the operation specified by (1.3) in the introduction as a unique solution of a sandwiched ℓ-rps according to Theorem 5.23. Here we have the union operation $\cup$ as a given operation. So let $KX = X \times X$ be the corresponding polynomial functor, and let us consider the natural transformation

$$u_X : K\mathcal{P}X \to \mathcal{P}X,$$
$$u_X(x, y) = x \cup y.$$

We have the abstract GSOS rule

$$\ell = (K(\mathcal{P} \times \text{Id}) \xrightarrow{K\pi_0} K\mathcal{P} \xrightarrow{u} \mathcal{P} \xrightarrow{\mathcal{P}\eta} \mathcal{P}\widehat{K})$$

whose interpretation is the union operation $\cup : C \times C \to C$. Now let $WX = X \times X$. Then the equation (1.3) corresponds to a natural transformation $e_0 : W(\mathcal{P} \times \text{Id}) \to \widehat{K}\mathcal{P}W$, where $(e_0)_X$ maps a pair

$$\big((\{x_i \mid i \in I\}, x), (\{y_j \mid j \in J\}, y)\big) \in W(\mathcal{P}X \times X)$$

to

$$\{(x, y_j) \mid j \in J\} \cup \{(x_i, y) \mid i \in I\} \cup \{(x_i, y_j) \mid i \in I, j \in J\} \in \widehat{K}\mathcal{P}WX.$$

From this we obtain an ℓ-srps

$$e = (W(\mathcal{P} \times \text{Id}) \xrightarrow{e_0} \widehat{K}\mathcal{P}W \xrightarrow{\widehat{K}\mathcal{P}\text{inr}} \widehat{K}\mathcal{P}(K + W) \xrightarrow{\widehat{K}\mathcal{P}\kappa^{K+W}} \widehat{K}\mathcal{P}\widehat{K + W}).$$

Its unique solution in C is the desired "parallel composition" $\| : C \times C \to C$ of non-well-founded sets.

7. Conclusions

In many areas of theoretical computer science, one is interested in recursive definitions of functions on terminal 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 an abstract GSOS rule $\ell : K(H \times \mathrm{Id}) \to H\widehat{K}$. We proved that ℓ induces new completely iterative algebra structures for $H\widehat{K}$ and $\widehat{K}H\widehat{K}$ on C. As a result, we are able to apply the existing body of solution theorems for cias to obtain new unique solution theorems for recursive equations of more general formats "for free".

Next we introduced the notion of an ℓ-rps and showed how to solve recursive function definitions uniquely in C which are given by an ℓ-rps. Our results explain why taking unique solutions of such equations is a modular 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.

We also generalized this point to *sandwiched* ℓ-rps's. The reason for doing this was not to gain expressive power: every operation on C definable by a sandwiched ℓ-rps is also definable by an ordinary ℓ-rps. The reason for the generalization was to have more usable syntactic specification formats.

In concrete applications ℓ-rps's (or abstract GSOS rules) are mostly given by finite sets of rules or equations. But the conversion from an abstract GSOS rule $\ell : K(H \times \mathrm{Id}) \to H\widehat{K}$ (for set functors) to a coalgebra for H involves a finite-to-infinite blow-up, i.e., one forms a H-coalgebra on $\widehat{K}X$, which is typically an infinite set. We leave as an open question the question to investigate exactly which operations are definable by *finite ℓ-(s)rps's*.

Another related question concerns defining operations on the rational fixpoint of a functor H [8, 34]; for a set functor H this is the subcoalgebra of the terminal H-coalgebra given by the behavior of all finite coalgebras (e.g., regular processes, rational trees or regular languages). The question is: when does an abstract GSOS rule induce operations on the rational fixpoint for H? The answer to this is discussed in the recent paper [16], which is inspired by the work of Aceto [1] who studied specification formats for operations on regular processes (see also [2]).

There remain a number of other topics for further work. Firstly, it should be interesting to identify in the various applications concrete syntactic formats of operational rules that correspond to ℓ-(s)rps's. For example, in the case of process algebras as discussed in Section 6.1, Turi and Plotkin [39] proved that abstract GSOS rules correspond precisely to transition system specifications with operational rules in the GSOS format of [15]. Bartels gave in his thesis [13] concrete syntactic rule formats for abstract GSOS rules in several other concrete cases and Klin [28] studied the instantiation of abstract GSOS rules for weighted transition systems. But concrete syntactic formats that are equivalently characterized by ℓ-srps's have not been studied yet.

Secondly, there remains the question whether our results can be generalized to other algebras than the initial cia (alias terminal coalgebra) along the lines of Capretta, Uustalu and Vene [17] who generalized Bartels' results from [12, 13]. A different kind of generalization to be investigated concerns the step from using free monads in the definition of ℓ-(s)rps's to arbitrary monads. Thirdly, it should be interesting to work out further applications. For example, it is clear that for weighted transition system specifications as considered by Klin [28], our results can be used to obtain unique solutions of recursive specifications. It

should also be interesting to investigate whether our results yield unique solution theorems for name and value passing process calculi as considered by Fiore and Turi [19]. Finally, it would be of interest to extend the results of [36] on properties of recursive program scheme solutions to the richer settings of this paper.

Acknowledgements. We thank the anonymous referees for their comments which helped us to improve the presentation of this paper.

References

[1] L. Aceto. GSOS and finite labelled transition systems. *Theoret. Comput. Sci.*, 131(1):181–195, 1994.
[2] L. Aceto, W. Fokking, and C. Verhoef. *Handbook of Process Algebra*, chapter Structural Operational Semantics. Elsevier, 2001.
[3] P. Aczel. *Non-Well-Founded Sets*, volume 14 of *CLSI Lecture Notes*. CLSI Publications, Stanford, 1988.
[4] P. Aczel, J. Adámek, S. Milius, and J. Velebil. Infinite trees and completely iterative theories: A coalgebraic view. *Theoret. Comput. Sci.*, 300:1–45, 2003.
[5] J. Adámek. Introduction to coalgebra. *Theory Appl. Categ.*, 14:157–199, 2005.
[6] J. Adámek, S. Milius, and J. Velebil. On coalgebras based on classes. *Theoret. Comput. Sci.*, 316:3–23, 2004.
[7] J. Adámek, S. Milius, and J. Velebil. Elgot algebras. *Log. Methods Comput. Sci.*, 2(5:4):31 pp., 2006.
[8] J. Adámek, S. Milius, and J. Velebil. Iterative algebras at work. *Math. Structures Comput. Sci.*, 16(6):1085–1131, 2006.
[9] J. Adámek and V. Trnková. *Automata and Algebras in Categories*, volume 37 of *Mathematics and its Applications*. Kluwer Academic Publishers, 1990.
[10] P. America and J. J. M. M. Rutten. Solving reflexive domain equations in a category of complete metric spaces. *J. Comput. System Sci.*, 39:343–375, 1989.
[11] M. Barr. Coequalizers and free triples. *Math. Z.*, 116:307–322, 1970.
[12] F. Bartels. Generalized coinduction. *Math. Structures Comput. Sci.*, 13(2):321–348, 2003.
[13] F. Bartels. *On generalized coinduction and probabilistic specification formats*. PhD thesis, Vrije Universiteit Amsterdam, 2004.
[14] J. Barwise and L. S. Moss. *Vicious circles*. CLSI publications, Stanford, 1996.
[15] B. Bloom, S. Istrail, and A. R. Meyer. Bisimulation can't be traced. *J. ACM*, 42(1):232–268, 1995.
[16] M. M. Bonsangue, S. Milius, and J. Rot. On the specification of operations on the rational behavior of systems. In B. Luttik and M. A. Reniers, editors, *Proc. Combined Workshop on Expressiveness in Concurrency and Structural Operational Semenatics (EXPRESS/SOS'12)*, Electron. Proc. Theoret. Comput. Sci., 2012. To appear.
[17] V. Capretta, T. Uustalu, and V. Vene. Recursive coalgebras from comonads. *Inform. and Comput.*, 204:437–468, 2006.
[18] M. Dekking, M. Mendès France, and A. J. van der Poorten. Folds! *Mathematical Intelligencer*, 4(3):130–138, 1982.
[19] M. Fiore and D. Turi. Semantics of name and value passing. In *Proc. 16th Annual Symposium on Logic in Computer Science (LICS'01)*. IEEE Computer Society, 2001.
[20] I. Guessarian. *Algebraic Semantics*, volume 99 of *Lecture Notes in Comput. Sci.* Springer, 1981.
[21] I. Hasuo and B. Jacobs. Context-free languages via coalgebraic trace semantics. In J. F. et al., editor, *Proc. Algebra and Coalgebra in Computer Science: First International Conference (CALCO'05)*, volume 3629 of *Lecture Notes in Comput. Sci.*, pages 213–231. Springer, 2005.
[22] F. Hausdorff. *Grundzüge der Mengenlehre*. Von Veit & Co., Leipzig, 1914.
[23] J. E. Hopcroft, R. Motwani, and J. D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley Longman, Amsterdam, 3rd edition, 2007.
[24] B. Jacobs. A bialgebraic review of deterministic automata, regular expressions and languages. In *Goguen Festschrift*, volume 4060 of *Lecture Notes Comput. Sci.*, 2006.
[25] B. Jacobs. Distributive laws for the coinductive solution of recursive equations. *Inform. and Comput.*, 204(4):561–587, 2006.

[26] G. M. Kelly. A unified treatment of transfinite constructions for free algebras, free monoids, colimits, associated sheaves, and so on. *Bull. Austral. Math. Soc.*, 22:1–83, 1980.

[27] M. Kick and A. J. Power. Modularity of behaviour of mathematical operational semantics. In *Proc. Coalgebraic Methods in Computer Science (CMCS'04)*, volume 106 of *Electron. Notes Theor. Comput. Sci.*, pages 185–200, 2004.

[28] B. Klin. Structural operational semantics for weighted transition systems. In J. Palsberg, editor, *Semantics and Algebraic Specification: Essays Dedicated to Peter D. Mosses on the Occasion of His 60th Birthday*, volume 5700 of *Lecture Notes Comput. Sci.*, pages 121–139. Springer, 2009.

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

[30] M. Lenisa, A. J. Power, and H. Watanabe. Distributivity for endofunctors, pointed and co-pointed endofunctors, monads and comonads. In H. Reichel, editor, *Proc. Coalgebraic Methods in Computer Science*, volume 33 of *Electron. Notes Theor. Comput. Sci.*, Amsterdam, 2000. Elsevier.

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

[32] S. MacLane. *Categories for the working mathematician.* Springer, 2nd edition, 1998.

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

[34] S. Milius. A sound and complete calculus for finite stream circuits. In *Proc. 25th Annual Symposium on Logic in Computer Science (LICS'10)*, pages 449–458. IEEE Computer Society, 2010.

[35] S. Milius and L. S. Moss. The category theoretic solution of recursive program schemes. *Theoret. Comput. Sci.*, 366:3–59, 2006.

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

[37] S. Milius, L. S. Moss, and D. Schwencke. CIA structures and the semantics of recursion. In C.-H. L. Ong, editor, *Proc. Foundations of Software Science and Computation Structures (FoSSaCS)*, volume 6014 of *Lecture Notes Comput. Sci.*, pages 312–327. Springer, 2010.

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

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

[40] A. J. Power. Towards a theory of mathematical operational semantics. In *Proc. Coalgebraic Methods in Computer Science (CMCS'03)*, volume 82 of *Electron. Notes Theor. Comput. Sci.*, page 16 pp., 2003.

[41] J. J. M. M. Rutten. Universal coalgebra: a theory of systems. *Theoret. Comput. Sci.*, 249(1):3–80, 2000.

[42] J. J. M. M. Rutten. A coinductive calculus of streams. *Math. Structures Comput. Sci.*, 15(1):93–147, 2005.

[43] J. J. M. M. Rutten. A tutorial on coinductive stream calculus and signal flow graphs. *Theor. Comput. Sci.*, 343(3):443–481, 2005.

[44] D. Schwencke. Coequational logic for accessible functors. *Inform. and Comput.*, 208:1469–1489, 2010.

[45] A. Silva and J. J. M. M. Rutten. A coinductive calculus of binary trees. *Inform. and Comput.*, 208(5):578–593, 2010.

[46] D. Turi. Categorical modelling of structural operational rules: Case studies. In E. Moggi and G. Rosolini, editors, *Proc. Category Theory and Computer Science*, volume 1290 of *Lecture Notes Comput. Sci.*, pages 127–146. Springer, 1997.

[47] T. Uustalu, V. Vene, and A. Pardo. Recursion schemes from comonads. *Nordic J. Comput.*, 8(3):366–390, 2001.

[48] J. Winter, M. M. Bonsangue, and J. J. M. M. Rutten. Context-free languages, coalgebraically. In *Proc. Algebra and Coalgebra in Computer Science (CALCO'11)*, volume 6859 of *Lecture Notes Comput. Sci.*, pages 359–376. Springer, 2011.

[49] J. Worrell. On the final sequence of a finitary set functor. *Theoret. Comput. Sci.*, 338:184–199, 2005.