A final coalgebra for k-regular sequences

Helle Hvid Hansen[1,3], Clemens Kupke[2],
Jan Rutten[1,3] and Joost Winter[3]

[1] Radboud University Nijmegen
[2] University of Strathclyde
[3] CWI Amsterdam

Abstract. We study k-regular sequences from a coalgebraic perspective. Building on the observation that the set of streams over a semiring S can be turned into a final coalgebra, we obtain characterizations of k-regular sequences in terms of finite weighted automata, finite systems of behavioral differential equations, and recognizable power series. The latter characterization is obtained via an isomorphism of final coalgebras based on the k-adic numeration system.

Dedication

It is our greatest pleasure to dedicate this article to Prakash Panangaden on the occasion of his 60th birthday. There are not many subjects in our own research that have not been influenced by his work and ideas. Before the notion of finality in semantics became prominent in the early nineties of the previous century, Prakash was already writing [14] about infinite objects requiring " ... a limit construction and a final object ... ". Another early reference that is of direct relevance for the present paper is [16], published as a report in 1985, in which streams and stream functions play a key role. For these and many other similar such inspiring examples, we are immensely grateful.

1 Introduction

Infinite sequences, or streams, are much studied in the fields of number theory, analysis, combinatorics, formal languages and many more. Streams are also one of the best known examples of a final coalgebra [18]. Of particular interest is the classification of streams in terms of certain finite automata, or alternatively, stream differential equations of a certain form. The simplest such class consists of all eventually periodic streams (over a set S). They are generated by finite automata in which each state is assigned an output in S and a unique next state. Let us call these deterministic 1-automata as they are deterministic automata on a one-letter alphabet (with output in S). In terms of stream differential equations (cf. [18]), eventually periodic streams are defined by simple systems of stream differential equations such as $x(0) = 0$, $x' = y$, $y(0) = 1$, $y' = y$.

We consider two ways of generalizing deterministic 1-automata. One is by going from deterministic to weighted transitions. In this setting we must assume that the output set S has the structure of a semiring. The class of sequences generated by finite weighted 1-automata is known to be the class of rational power series on a 1-letter alphabet, or in the case that S is a field, the class of rational streams (cf. [18]). In terms of stream differential equations, rational streams are defined by equations such as $x(0) = 0$, $x' = x + y$, $y(0) = 1$, $y' = 2x$.

The other generalization is by going from a one-letter alphabet to a k-letter alphabet, for $k \in \mathbb{N}$. Here, finite deterministic k-automata generate exactly the k-automatic sequences [2]. It was shown in [9, 12] that k-automatic sequences are defined by systems of equations involving the stream operation $\mathbf{zip}_k$, such as (for $k = 2$), $x(0) = 1$, $x = \mathbf{zip}_2(x, y)$, $y(0) = 2$, $y = \mathbf{zip}_2(y, y)$. (Note that the left-hand sides are x and y, and not x' and y'). These equations can be expressed using the **even** and **odd** stream operations, such as $x(0) = 1$, $\mathbf{even}(x) = y, \mathbf{odd}(x) = y$, $y(0) = 2$, $\mathbf{even}(y) = y, \mathbf{odd}(y) = x$. This approach generalizes easily to arbitrary $k \geq 2$.

In this paper we will show that (generalizing in both directions) finite weighted k-automata generate exactly the k-regular sequences. On the side of equational specification, we show that k-regular streams are defined by systems of linear $\mathbf{zip}_k$-behavioral differential equations, which are equations such as, e.g., $x(0) = 0$, $x' = \mathbf{zip}(x + y, 2y)$, $y(0) = 1$, $y' = \mathbf{zip}(2x, x + y)$. One way of summarising our insights in a slogan would be: k-regular sequences are to k-automatic sequences what rational streams are to eventually periodic ones. Our main characterization results are stated in Theorem 14.

Our approach is coalgebraic, although we use the more familiar terminology of automata. A seemingly small, technical difference with existing work is our use of the bijective k-adic numeration system as opposed to the non-bijective standard base k numeration. The advantage of using the bijective numeration system is that the automaton structure on streams obtained via the k-adic numeration yields immediately a final k-automaton, rather than a relatively final one for zero-consistent automata, as in [12]. Consequently, we obtain an isomorphism between the final k-automaton of sequences and the (classic) final k-automaton of formal power series, and this isomorphism restricts to one between k-regular sequences and rational formal power series. We also obtain a characterization of k-automatic sequences as those k-regular sequences that have finite output range. Another generalization with respect to [1] is the assumption that S is just a semiring, not a ring.

Finally, we provide a connection between our coalgebraic presentation of the k-regular sequences, and sequences attainable by so-called *divide-and-conquer recurrences* (see e.g. [10], [20]). We also note that linear $\mathbf{zip}_k$-behavioral differential equations give an easy way of specifying these sequences coinductively in the functional programming language Haskell.

Related work. The k-regular sequences were introduced in [1] as generalizations of k-automatic sequences, and are further treated in [3], Chapter 16 of [2], and Chapter 5 of [6]. Some open questions posed in the original paper [1] were solved

in [5, 15]. The work in this article builds on existing coalgebraic approaches to automatic sequences, which can be found in [12] and [9]. In particular, our systems of linear zip-behavioral differential equations can be seen as a linear generalization of the $\mathbf{zip}_k$-specifications in [9]. The isomorphism of final coalgebras presented in this paper is probably folklore, but we think its usefulness warrants an explicit inclusion in our paper. Finally, we remark that the definitions and results of section 3.2 on solutions to linear $\mathbf{zip}_k$-behavioral differential equations can be seen as instances of more general concepts in the theory of bialgebra (cf. [4]). Such an abstract presentation is, however, not necessary and would not improve the results of the paper.

Acknowledgements. We would like to thank Alexandra Silva for helpful discussions on weighted automata.

2 Preliminaries

2.1 Semirings and semimodules

Throughout this paper, S denotes a semiring. A semiring $S = (S, +, \cdot, 0, 1)$ consists of a commutative monoid $(S, +, 0)$ and a monoid $(S, \cdot, 1)$ such that the following identities hold for all s, t, u in S: $0 \cdot s = s \cdot 0 = 0$, $s \cdot (t + u) = s \cdot t + s \cdot u$, $(s + t) \cdot u = s \cdot u + t \cdot u$. A *left-semimodule over* S is a commutative monoid $(M, +, 0)$ together with a left-action by S, i.e., a map $S \times M \to M$, denoted as scalar multiplication $(s, m) \mapsto sm$ for all $s \in S, m \in M$ which satisfies: $(st)m = s(tm)$, $s(m + n) = sm + sn$, $(s + t)m = sm + tm$, $0m = 0$, $1m = m$. A *left-linear map* between left-semimodules is a map $f \colon M \to N$ which respects scalar multiplication and sum: $f(sm) = sf(m)$ and $f(m_1 + m_2) = f(m_1) + f(m_2)$. Right-semimodules over S are defined similarly via a right action. If S is commutative, i.e., the multiplicative monoid $(S, \cdot, 1)$ is commutative, then left and right semimodules are the same.

We will work in the setting of left-semimodules over S and left-linear maps, which for simplicity we refer to as *S-semimodules* and *linear maps*. Note that S is itself an S-semimodule with the left action given by multiplication in S.

2.2 Stream operations, zip and unzip

We will use some notation and terminology from coinductive stream calculus (see e.g. [18]). A *stream* (over the semiring S) is an (infinite) sequence $(\sigma(0), \sigma(1), \sigma(2), \ldots)$ of elements from S, or more formally, a map $\sigma \colon \mathbb{N} \to S$, also written $\sigma \in S^{\mathbb{N}}$. We will use the terminology of *streams* and *sequences* interchangeably, and the notions can be regarded as synonymous.

The *initial value* and *derivative* of a stream $\sigma \in S^{\mathbb{N}}$ are $\sigma(0)$ and σ', respectively, where $\sigma'(n) = \sigma(n + 1)$ for all $n \in \mathbb{N}$. The initial value and derivative of σ are also known as $\mathbf{head}(\sigma)$ and $\mathbf{tail}(\sigma)$.

The streams $S^{\mathbb{N}}$ form an S-semimodule under pointwise addition and scalar multiplication which are the unique stream operations that satisfy the following *stream differential equations* (cf. [18]):

$$\begin{array}{ll} (\sigma+\tau)(0) = \sigma(0)+\tau(0), & (\sigma+\tau)' = \sigma'+\tau', \\ (a\sigma)(0) = a\sigma(0), & (a\sigma)' = a\sigma'. \end{array} \tag{1}$$

for all $\sigma, \tau \in S^{\mathbb{N}}$ and $a \in S$. Note that from the above equations it follows immediately that $\mathbf{head}\colon S^{\mathbb{N}} \to S$ and $\mathbf{tail}\colon S^{\mathbb{N}} \to S^{\mathbb{N}}$ are linear.

The *shift* operation $\mathcal{X}$ is defined as $\mathcal{X}\sigma = (0, \sigma(0), \sigma(1), \ldots)$, or equivalently, by the stream differential equation:

$$(\mathcal{X}\sigma)(0) = 0, \qquad (\mathcal{X}\sigma)' = \sigma.$$

We will use the so-called *fundamental theorem of stream calculus*(cf. [18])[4]:

$$\text{for all } \sigma \in S^{\mathbb{N}}\colon \quad \sigma = \sigma(0) + \mathcal{X}\sigma' \tag{2}$$

Of central importance to us are the k-ary operations $\mathbf{zip}_k$. For $k \in \mathbb{N}$, $\mathbf{zip}_k$ zips together k streams $\sigma_0, \ldots, \sigma_{k-1}$ into one by taking elements in turn from its arguments. Formally, for $k \in \mathbb{N}$ and streams $\sigma_0, \ldots, \sigma_{k-1}$ the stream operation $\mathbf{zip}_k$ is defined by

$$\mathbf{zip}_k(\sigma_0, \ldots, \sigma_{k-1})(i+nk) = \sigma_i(n) \qquad \forall n, i \in \mathbb{N}, 0 \leq i < k \tag{3}$$

or equivalently, by the stream differential equation:

$$\begin{array}{ll} \mathbf{zip}_k(\sigma_0, \ldots, \sigma_{k-1})(0) = \sigma_0(0) \\ \mathbf{zip}_k(\sigma_0, \ldots, \sigma_{k-1})' & = \mathbf{zip}_k(\sigma_1, \ldots, \sigma_{k-1}, \sigma_0'). \end{array} \tag{4}$$

For example, for $k = 2$, we have $\mathbf{zip}_2(\sigma, \tau) = (\sigma(0), \tau(0), \sigma(1), \tau(1), \ldots)$.

Conversely, the unzipping operations are defined as follows for $k, j \in \mathbb{N}$ with $j < k$:

$$\mathbf{unzip}_{j,k}(\sigma)(n) = \sigma(j+nk) \qquad \forall n \in \mathbb{N} \tag{5}$$

For $k = 2$, $\mathbf{unzip}_{0,2}$ and $\mathbf{unzip}_{1,2}$ are also known as **even** and **odd**:

$$\begin{array}{ll} \mathbf{unzip}_{0,2}(\sigma) = \mathbf{even}(\sigma) = (\sigma(0), \sigma(2), \sigma(4), \ldots) \\ \mathbf{unzip}_{1,2}(\sigma) = \mathbf{odd}(\sigma) & = (\sigma(1), \sigma(3), \sigma(5), \ldots) \end{array}$$

It can easily be verified that

$$\mathbf{zip}_k(\mathbf{unzip}_{0,k}(\sigma), \ldots, \mathbf{unzip}_{k-1,k}(\sigma)) = \sigma \tag{6}$$

and conversely that (for i with $0 \leq i < k$)

$$\mathbf{unzip}_{i,k}(\mathbf{zip}_k(\sigma_0, \ldots, \sigma_{k-1})) = \sigma_i. \tag{7}$$

In other words, $\mathbf{zip}_k\colon (S^{\mathbb{N}})^k \to S^{\mathbb{N}}$ is a bijection with inverse

$$\mathbf{unzip}_k = (\mathbf{unzip}_{0,k}, \ldots, \mathbf{unzip}_{k-1,k})\colon S^{\mathbb{N}} \to (S^{\mathbb{N}})^k.$$

The **unzip**-operations relate to the more familiar notion of a k-kernel. The *k-kernel* of a stream σ can be defined as the closure of the set $\{\sigma\}$ under the operations $\mathbf{unzip}_{j,k}$ for $0 \leq j < k$.

[4] Here $\sigma(0)$ is overloaded as the stream $(\sigma(0), 0, 0, \ldots)$

2.3 Automata as coalgebras

We briefly recall some basic definitions on (weighted) automata (with weights in a semiring), and how these are modelled as coalgebras [17].

Automata and formal power series. Given a finite alphabet A and a semiring S, a *(linear) A-automaton (with output in S)* is a triple (Q, o, δ), where Q is an S-semimodule, $o\colon Q \to S$ is a linear map assigning an output value $o(q)$ to each $q \in Q$, and $\delta\colon Q \to Q^A$ is a linear map assigning to each $q \in Q$ and $a \in A$ a next state $\delta(q)(a)$ which we will also denote by q_a and refer to as the *a-derivative of q*. Note the absence of initial states or state vectors in this presentation. The transition function δ can be extended to a map $\delta^*\colon Q \to Q^{A^*}$ in the usual inductive manner: $\delta^*(q)(\epsilon) = q$ and $\delta^*(q)(wa) = \delta(\delta^*(q)(w))(a)$.

The set $S\langle\langle A\rangle\rangle$ of *formal power series over noncommuting variables from A with outputs in S* is the function space $S\langle\langle A\rangle\rangle = \{\rho\colon A^* \to S\}$ equipped with pointwise S-semimodule structure. We note that a formal power series $\rho\colon A^* \to S$ can also be seen as an S-weighted language.[5] The formal power series *generated by a state $q \in Q$* in an A-automaton (Q, o, δ) is the map $[\![q]\!]_L\colon A^* \to S$ defined by $[\![q]\!]_L(w) = o(\delta^*(q)(w))$.

An A-automaton is a coalgebra for the functor $S \times (-)^A$ on the category of S-semimodules and linear maps. The theory of universal coalgebra [17] now directly yields an associated notion of homomorphism. Diagrammatically, given A-automata (Q, o_Q, δ_Q) and (R, o_R, δ_R), a linear map $h : Q \to R$ is a *homomorphism* iff it makes the diagram

$$\begin{array}{ccc} Q & \xrightarrow{\quad h \quad} & R \\ {\scriptstyle (o_Q,\delta_Q)}\downarrow & & \downarrow{\scriptstyle (o_R,\delta_R)} \\ S \times Q^A & \xrightarrow{1_S \times h^A} & S \times R^A \end{array}$$

commute, or equivalently, iff for all $q \in Q$, $o_Q(q) = o_R(h(q))$ and $h(q_a) = h(q)_a$ for all $a \in A$. An *isomorphism of A-automata* is a bijective homomorphism.

The set $S\langle\langle A\rangle\rangle$ of formal power series is itself an A-automaton

$$\mathcal{L} = (S\langle\langle A\rangle\rangle, O, \Delta)$$

with O and Δ defined by

$$O(\rho) = \rho(\epsilon) \qquad \text{and} \qquad \Delta(\rho)(a)(w) = \rho(aw).$$

In fact, $(S\langle\langle A\rangle\rangle, O, \Delta)$ is known to be *final* in the category of A-automata (this follows from [18, Theorem 9.1] combined with the fact that all final mappings are linear). This means that given any A-automaton (Q, o, δ), there is a unique homomorphism $(Q, o, \delta) \to (S\langle\langle A\rangle\rangle, O, \Delta)$. This unique homomorphism is, in fact, the function $[\![-]\!]_L : Q \to S\langle\langle A\rangle\rangle$ which maps $q \in Q$ to the formal power series generated by q. We note that final coalgebras are unique only up to isomorphism.

[5] This explains our later use of L as subscript to indicate formal power series.

Recognizable formal power series are characterized in terms of weighted automata. We first introduce some notation. For a set X, we denote by S_ω^X the set of all functions $\phi\colon X \to S$ with finite support, i.e., $\phi(x) \neq 0$ for only finitely many $x \in X$. Equivalently, such a ϕ can be seen as a linear combination $a_1x_1 + \cdots + a_nx_n$ where $a_i = \phi(x_i)$, $i = 1, \ldots, n$ and $\phi(x) = 0$ for all $x \notin \{x_1, \ldots, x_n\}$. The set X is included into S_ω^X via the map $\eta\colon X \hookrightarrow S_\omega^X$ defined as $\eta(x) = 1x$. Importantly, by taking pointwise S-semimodule structure, S_ω^X is the *free S-semimodule* over the set X, which means that for each function $f\colon X \to M$ into some S-semimodule M, there is a unique linear map $\hat{f}\colon S_\omega^X \to M$ extending f, i.e., $\hat{f} \circ \eta = f$.

Weighted automata. An *S-weighted A-automaton* is a triple (X, o, δ) where X is a set (of states), $o\colon X \to S$ is an output function, and $\delta\colon X \to (S_\omega^X)^A$ is a transition function. In terms of weighted transitions, $\delta(x)(a)(y) \in S$ is the weight of the a-transition from x to y. We say that (X, o, δ) is *finite* if X is finite. An S-weighted A-automaton is a coalgebra for the functor $S \times (S_\omega^-)^A$ on the category of sets and functions. We note that a nondeterministic automaton is a $\mathbf{2}$-weighted automaton where $\mathbf{2} = (\{\bot, \top\}, \vee, \wedge, \bot, \top)$ is the Boolean semiring.

Determinization. Any S-weighted A-automaton (X, o, δ) can be *determinized* to an A-automaton, by constructing an A automaton $(S_\omega^X, \hat{o}, \hat{\delta})$, where $\hat{o}$ and $\hat{\delta}$ are the unique linear extensions of o and δ to the free semimodule S_ω^X, i.e. the unique linear mappings satisfying $\hat{o}(\eta(x)) = o(x)$ and $\hat{\delta}(\eta(x)) = \delta(x)$.

This construction can be summarized in the following diagram:

$$\begin{array}{ccccc} X & \xrightarrow{\eta} & S_\omega^X & \xrightarrow{[\![-]\!]_L} & S\langle\!\langle A \rangle\!\rangle \\ {\scriptstyle (o,\delta)}\downarrow & \swarrow{\scriptstyle (\hat{o},\hat{\delta})} & & & \downarrow{\scriptstyle (O,\Delta)} \\ S \times (S_\omega^X)^A & & \xrightarrow{1_S \times [\![-]\!]_L{}^A} & & S \times S\langle\!\langle A \rangle\!\rangle^A \end{array} \tag{8}$$

We say that a state x in an S-weighted A-automaton *generates* $\rho \in S\langle\!\langle A \rangle\!\rangle$ if $[\![\eta(x)]\!]_L = \rho$. When X is finite, the determinization has a finitely generated S-semimodule S_ω^X as its state space, but as a set S_ω^X is generally infinite. For further details on determinization and its categorical/coalgebraic setting we refer to [19].

The above now yields the following definition of the recognizable power series.

Definition 1. *A formal power series $\rho \in S\langle\!\langle A \rangle\!\rangle$ is recognizable if and only if there is a finite S-weighted A-automaton (X, o, δ) such that $\rho = [\![x]\!]_L$ for some $x \in X$.*

In other words, ρ is recognizable if and only if it is generated by some state in a finite S-weighted A-automaton. This definition is easily seen to correspond to the classic definition in e.g. [6].

2.4 Numeration systems

For any natural number $k \in \mathbb{N}$ with $k \geq 1$, let A_k denote the alphabet of digits

$$A_k = \{\mathtt{1}, \ldots, \mathtt{k}\}$$

We emphasize the use of the digits as alphabet symbols by writing them in a fixed-width font. The map $\nu_k \colon A_k^* \to \mathbb{N}$, which assigns to each string of digits the natural number it represents, is defined inductively by:

$$\nu_k(\epsilon) = 0 \qquad \text{and} \qquad \nu_k(\mathtt{i} \cdot w) = i + k \cdot \nu_k(w).$$

It is well-known and easy to see that ν_k is a bijection between natural numbers and their representation in the *k-adic numeration system*[6], with the least significant digit on the left. For example, the 2-adic numeration of the natural numbers starts as follows: $\epsilon, 1, 2, 11, 21, 12, 22, 111, \ldots$

We contrast the bijective k-adic numeration system with the familiar *(standard) base k numeration system* which is defined as follows. The alphabet of digits is

$$B_k = \{\mathtt{0}, \ldots, \mathtt{k-1}\}$$

and, whenever $k \geq 2$, we can define a mapping $\xi_k \colon B_k^* \to \mathbb{N}$ inductively by

$$\xi_k(\epsilon) = 0 \qquad \text{and} \qquad \xi_k(\mathtt{i} \cdot w) = i + k \cdot \xi_k(w)$$

This again gives us a presentation with the least significant digit on the left.[7] For example, standard base 2 is the reverse binary notation with zero represented by ϵ, i.e., starting as $\epsilon, 1, 01, 10, 11, 001, 101, 011, 111, \ldots$ The map ξ_k has the property that for all $w \in B_k^*$, $\xi(w) = \xi(w \cdot \mathtt{0})$, and hence ξ_k is not bijective.

Finally, observe that, from the inductive definitions of the k-adic and standard base k numeration, we obtain that

$$\nu_k(\mathtt{a_0} \ldots \mathtt{a_n}) = \sum_{i=0}^{n} a_i k^i \qquad \text{and} \qquad \xi_k(\mathtt{b_0} \ldots \mathtt{b_n}) = \sum_{i=0}^{n} b_i k^i$$

for all $\mathtt{a_i} \in A_k$ and $\mathtt{b_i} \in B_k$, which can be taken as alternative definitions of the two numeration systems.

In most literature on k-automatic and k-regular sequences, the standard base k numeration system is employed. However, we prefer the bijective k-adic numeration system since it yields a bijective correspondence at the level of final coalgebras. Moreover, unlike in the standard base k numeration, there is a straightforward and intuitive bijective numeration for the case $k = 1$ given by $\epsilon, 1, 11, 111, \ldots$

[6] Unrelated to and not to be confused with the p-adic numbers

[7] For a more standard presentation with the most significant digit on the left, switch the inductive definition to $\xi_k(w \cdot \mathtt{i}) = i + k \cdot \xi_k(w)$.

3 Characterizations of k-regular sequences

The notion of a k-regular sequence with values in a ring was introduced in [1]. The following definition is (roughly) a direct generalization to sequences with values in a semiring. We discuss the more precise relationship in the remark below.

Definition 2. *A sequence $\sigma \in S^{\mathbb{N}}$ is* k-regular *iff the k-kernel of σ is contained in a finitely generated S-subsemimodule of $S^{\mathbb{N}}$, or equivalently, iff there is a finite set of generators $\Sigma = \{\sigma_0, \dots \sigma_{n-1}\}$ with $\sigma \in \Sigma$, and an indexed family $a_{h,i,j}$ for all $h, i, j \in \mathbb{N}$ with $h < n$, $i < n$, $j < k$, such that for all $h < n$ and $j < k$*

$$\mathbf{unzip}_{j,k}(\sigma_h) = \sum_{i<n} a_{h,i,j}\sigma_i$$

or equivalently, for all $h < n$:

$$\sigma_h = \mathbf{zip}_k\left(\sum_{i<n} a_{h,i,0}\sigma_i, \dots, \sum_{i<n} a_{h,i,k}\sigma_i\right). \tag{9}$$

Remark 3. In [1], the definition of a k-regular sequence is as follows: *Let R be a ring and R' a (commutative) Noetherian ring contained in R. A sequence $\sigma \in R^{\mathbb{N}}$ is (R', k)-*regular *if each sequence in the k-kernel of σ is an R'-linear combination of some finite set of sequences $\sigma_1, \dots, \sigma_n \in R^{\mathbb{N}}$.* In terms of modules, this is equivalent with saying that the k-kernel of σ is contained in a finitely generated R'-submodule of $R^{\mathbb{N}}$ (viewed as an R'-module). Since R' is assumed to be Noetherian, this in turn is equivalent with the k-kernel itself being a finitely generated R'-submodule of $R^{\mathbb{N}}$. For simplicity, we do not distinguish between the semiring S of values and a subsemiring $S' \subseteq S$ from which linear coefficients may be taken. Hence in the terminology of [1], our definition of k-regular could be phrased as (S, k)-regular. If we assume that S is a Noetherian semiring (cf. [8]), then our definition is equivalent to requiring that the k-kernel of σ is a finitely generated S-subsemimodule of $S^{\mathbb{N}}$.

In this section we will give characterizations of k-regular sequences in terms of finite weighted automata, finite systems of (certain) behavioral differential equations, and recognizable formal power series.

3.1 An isomorphism between final A_k-automata

We start by defining an A_k-automaton $\mathcal{S}$ with state space $S^{\mathbb{N}}$ as the composition of bijections:

$$S^{\mathbb{N}} \xrightarrow[\cong]{(\mathbf{head},\mathbf{tail})} S \times S^{\mathbb{N}} \xrightarrow[\cong]{1_S \times \mathbf{unzip}_k} S \times (S^{\mathbb{N}})^{A_k}$$

That is,

$$\mathcal{S} := (S^{\mathbb{N}}, \mathbf{head}, \mathbf{unzip}_k \circ \mathbf{tail})$$

In [9, Proposition 26] it was observed that $\mathcal{S}$ is a final A_k-automaton. This result will also follow from our Proposition 5 below, and the finality of $\mathcal{L}$.

Given an A_k-automaton (Q, o, δ), we let $[\![-]\!]_S \colon Q \to S^{\mathbb{N}}$ denote the unique mapping into the final A_k-automaton on $S^{\mathbb{N}}$:

$$\begin{array}{ccc} Q & \xrightarrow{\quad [\![-]\!]_S \quad} & S^{\mathbb{N}} \\ {\scriptstyle (o,\delta)}\downarrow & & \downarrow{\scriptstyle (\mathbf{head}, \mathbf{unzip}_k \circ \mathbf{tail})} \\ S \times Q^{A_k} & \xrightarrow{1_S \times [\![-]\!]_S^{A_k}} & S \times (S^{\mathbb{N}})^{A_k} \end{array} \tag{10}$$

The commutativity of the above diagram means that for all $q \in Q$:

$$\begin{array}{ll} o(q) & = \mathbf{head}([\![q]\!]_S) \\ [\![\delta(q)(\mathtt{i})]\!]_S = [\![q_{\mathtt{i}}]\!]_S & = \mathbf{unzip}_{i-1,k}([\![q]\!]'_S) \qquad \text{for all } \mathtt{i} \in A_k \end{array} \tag{11}$$

or, equivalently, using the $\mathbf{zip}_k$-$\mathbf{unzip}_k$ isomorphism (6),

$$[\![q]\!]_S(0) = o(q), \qquad [\![q]\!]'_S = \mathbf{zip}_k([\![\delta(q)(\mathtt{1})]\!]_S, \ldots, [\![\delta(q)(\mathtt{k})]\!]_S). \tag{12}$$

We refer to $[\![q]\!]_S$ as the *stream semantics of* q. Conversely, we will say that q *generates the stream* $[\![q]\!]_S$.

As final coalgebras are unique up to isomorphism, it follows that $\mathcal{S}$ and $\mathcal{L} = (S\langle\!\langle A_k \rangle\!\rangle, O, \Delta)$ are isomorphic. We will show that the unique isomorphism between $\mathcal{S}$ and $\mathcal{L}$ is concretely given by k-adic numeration. First, we define a map from sequences to formal power series.

Definition 4. *We define the map* $h_L \colon S^{\mathbb{N}} \to S\langle\!\langle A_k \rangle\!\rangle$ *by*

$$h_L(\sigma)(w) = \sigma(\nu_k(w)) \qquad \textit{for all } w \in A_k^* \tag{13}$$

where $\nu_k \colon A_k^* \to \mathbb{N}$ *is the bijective k-adic numeration given in Section 2.4. For* $\sigma \in S^{\mathbb{N}}$ *we refer to* $h_L(\sigma)$ *as* the formal power series corresponding to σ via k-adic numeration.

Proposition 5. *The map* $h_L \colon S^{\mathbb{N}} \to S\langle\!\langle A_k \rangle\!\rangle$ *is an isomorphism of* A_k*-automata from* $\mathcal{S}$ *to* $\mathcal{L}$*, i.e., the following diagram commutes (where* $h_S = h_L^{-1}$*):*

$$\begin{array}{ccc} S^{\mathbb{N}} & \underset{h_S}{\overset{h_L}{\rightleftarrows}} & S\langle\!\langle A_k \rangle\!\rangle \\ {\scriptstyle (\mathbf{head}, \mathbf{unzip}_k \circ \mathbf{tail})}\downarrow & & \downarrow{\scriptstyle (O,\Delta)} \\ S \times (S^{\mathbb{N}})^{A_k} & \underset{1_S \times h_S^k}{\overset{1_S \times h_L^k}{\rightleftarrows}} & S \times S\langle\!\langle A_k \rangle\!\rangle^{A_k} \end{array}$$

Proof. We must show that h_L is a bijective homomorphism. From the fact that ν_k is a bijection it directly follows that h_L is a bijection. It remains to show that h_L is a homomorphism of A_k-automata.

For this, we first have to prove that $\mathbf{head}(\sigma) = O(h_L(\sigma))$, which holds because

$$\mathbf{head}(\sigma) = \sigma(0) = \sigma(\nu_k(\epsilon)) = h_L(\sigma)(\epsilon) = O(h_L(\sigma)).$$

Now, we have to show that $(h_L(\sigma))_{\mathtt{i}} = h_L(\sigma_{\mathtt{i}})$. This holds, because given any $w \in A_k^*$ and $\mathtt{i} \in A_k$, we have:

$$\begin{aligned}
(h_L(\sigma))_{\mathtt{i}}(w) &= (h_L(\sigma))(\mathtt{i} \cdot w) \\
&= \sigma(\nu_k(\mathtt{i} \cdot w)) \\
&= \sigma(i + k \cdot \nu_k(w)) \\
&= \sigma'((i-1) + k \cdot \nu_k(w)) \\
&= \mathbf{unzip}_{i-1,k}(\sigma')(\nu_k(w)) \\
&= h_L((\mathbf{unzip}_{i-1,k} \circ \mathbf{tail})(\sigma))(w) \\
&= h_L(\sigma_{\mathtt{i}})(w)
\end{aligned}$$

Finally, it can easily be verified that h_L is linear using (1). □

In combination with the fact that homomorphisms to final automata are unique, this now directly leads to the following corollary:

Corollary 6. *For any A_k-automaton (Q, o, δ), we have*

$$h_L \circ [\![-]\!]_S = [\![-]\!]_L \quad \textit{and} \quad h_S \circ [\![-]\!]_L = [\![-]\!]_S.$$

3.2 Systems of linear zip-behavioral differential equations

The finality of $\mathcal{S}$ gives rise to a coinduction principle for weighted automata. Namely, by defining an S-weighted A_k-automaton (X, o, δ) we are defining the streams $[\![\eta(x)]\!]_S \in S^{\mathbb{N}}$ for each $x \in X$, via determinization and finality as described in the following diagram, which is the analogue of (8) only with $\mathcal{S}$ instead of $\mathcal{L}$.

$$\begin{array}{ccccc}
X & \xrightarrow{\eta} & S_\omega^X & \xrightarrow{[\![-]\!]_S} & S^{\mathbb{N}} \\
{\scriptstyle (o,\delta)}\downarrow & \swarrow{\scriptstyle (o,\delta)^\sharp} & & & \downarrow{\scriptstyle (\mathbf{head},\mathbf{unzip}_k \circ \mathbf{tail})} \\
S \times (S_\omega^X)^{A_k} & & \xrightarrow{1_S \times [\![-]\!]_S{}^{A_k}} & & S \times (S^{\mathbb{N}})^{A_k}
\end{array} \tag{14}$$

The existence of $[\![-]\!]_S$ and the commutativity of the above diagram immediately tells us the following fact:

Lemma 7. *A sequence σ is generated by a state in a finite weighted automaton if and only if there is a finite set of sequences $\Sigma = \{\sigma_0, \ldots, \sigma_{n-1}\}$ with $\sigma \in \Sigma$ such that for all $j < k$ and $i < n$, $\mathbf{unzip}_{j,k}(\sigma_i')$ is a linear combination of elements from Σ.*

We can formulate the uniqueness of $[\![-]\!]_S$ and the commutativity of (14) in terms of the $\mathbf{zip}_k$-operations by using the homomorphism condition (12).

Lemma 8. *Given an S-weighted A_k-automaton (X, o, δ), $[\![-]\!]_S$ is the unique linear mapping $S_\omega^X \to S^{\mathbb{N}}$ satisfying, for each $x \in X$, the equations*

$$[\![\eta(x)]\!]_S(0) = o(x) \qquad [\![\eta(x)]\!]'_S = \mathbf{zip}_k([\![\delta(x)(\mathtt{1})]\!]_S, \ldots, [\![\delta(x)(\mathtt{k})]\!]_S).$$

(Recall that $\eta\colon X \hookrightarrow S_\omega^X$ is the inclusion of states into the determinization.)

The above lemma justifies an alternative method of specifying streams via equations involving the $\mathbf{zip}_k$-operation. A *system of linear $\mathbf{zip}_k$-behavioral differential equations over a set (of variables) X* is a system of equations, one for each $x \in X$, of the form,

$$x(0) = a_x, \qquad x' = \mathbf{zip}_k(\alpha_{x,1}, ..., \alpha_{x,k}) \tag{15}$$

where $a_x \in S$ and $\alpha_{x,1}, \ldots, \alpha_{x,k}$ are S-linear combinations over X.

Linear $\mathbf{zip}_k$-behavioral differential equations and "plain" behavioral differential equations (using formal power series derivatives) both describe weighted automata, but the use of linear **zip**-behavioral differential equations makes it explicit that we intend to apply the finality of $\mathcal{S}$ (to obtain streams), rather than the finality of $\mathcal{L}$ (to obtain formal power series). Explicitly, given a system of linear $\mathbf{zip}_k$-behavioral differential equations as in (15), the corresponding S-weighted A_k-automaton (X, o, δ) is given by $o(x) = a_x$, and $\delta(x)(\mathtt{i}) = \alpha_{x,i}$ for all $x \in X$ and $\mathtt{i} \in A_k$.

We illustrate with a small example. The streams specified in derivative form by the behavioral differential equations:

$$\begin{array}{lll} o(x) = 1, & x_1 = x + y, & x_2 = 3x + y \\ o(y) = 2, & y_1 = y, & y_2 = x + 2y \end{array}$$

are equivalently specified in terms of $\mathbf{unzip}_{0,2} \circ \mathbf{tail}$ and $\mathbf{unzip}_{1,2} \circ \mathbf{tail}$, i.e., by $\mathbf{even} \circ \mathbf{tail}$ and $\mathbf{odd} \circ \mathbf{tail}$:

$$\begin{array}{lll} o(x) = 1, & \mathbf{even}(x') = x + y, & \mathbf{odd}(x') = 3x + y \\ o(y) = 2, & \mathbf{even}(y') = y, & \mathbf{odd}(y') = x + 2y \end{array}$$

and equivalently (via the zip-unzip isomorphism (6)), by the system of linear $\mathbf{zip}_2$-behavioral differential equations:

$$\begin{array}{ll} o(x) = 1, & x' = \mathbf{zip}_2(x + y, 3x + y) \\ o(y) = 2, & y' = \mathbf{zip}_2(y, x + 2y) \end{array}$$

A *solution* to a system of linear $\mathbf{zip}_k$-behavioral differential equations over X with components given as in (15), is a map $f\colon X \to S^{\mathbb{N}}$ such that for all $x \in X$,

$$f(x)(0) = a_x, \qquad f(x)' = \mathbf{zip}_k(\hat{f}(\alpha_{x,0}), ..., \hat{f}(\alpha_{x,k-1})) \tag{16}$$

where $\hat{f}\colon S_\omega^X \to S^{\mathbb{N}}$ is the linear extension of f with respect to the semimodule structure on $S^{\mathbb{N}}$.

The basic fact that justifies viewing systems of linear $\mathbf{zip}_k$-behavioral differential equations as defining streams, is stated in the following lemma.

Lemma 9. *Every system of linear* $\mathbf{zip}_k$*-behavioral differential equations has a unique solution given by the final stream semantics* $[\![-]\!]_S \circ \eta$ *of the corresponding weighted automaton* (X, o, δ)*.*

Proof. By Lemma 8, the map $[\![-]\!]_S \circ \eta \colon X \to S^{\mathbb{N}}$ is a solution, and it is unique by uniqueness of $[\![-]\!]_S$. □

We will say that a stream σ is *defined by system of linear* $\mathbf{zip}_k$*-behavioral differential equations over* X if $\sigma = [\![\eta(x)]\!]_S$ for some $x \in X$ in such a system. In what follows, we will suppress η and simply write $[\![x]\!]_S$ instead of $[\![\eta(x)]\!]_S$.

3.3 Characterizations of k-regular sequences

We will now show that k-regular sequences are obtained precisely by the stream semantics finite S-weighted A_k-automata. It will follow that k-regular sequences are in bijective correspondence with recognizable formal power series via k-adic numeration. This is an analogue of the result in [1] which shows that k-regular sequences over the integers $\mathbb{Z}$ correspond to *some* $\mathbb{Z}$-rational power series in noncommuting variables $\{\mathtt{0}, \ldots, \mathtt{k-1}\}$ via the standard base k numeration.

Proposition 10. *Given any* $k \geq 2$*, if* $\sigma \in S^{\mathbb{N}}$ *is a* k*-regular sequence, then there is a finite* S*-weighted* A_k*-automaton* (X, o, δ) *and an* $x \in X$*, such that* $[\![x]\!]_S = \sigma$*.*

Proof. If σ is k-regular, there is a finite set of sequences $\Sigma = \{\sigma_0, \ldots, \sigma_{n-1}\}$ with $\sigma \in \Sigma$, and values a_h and $b_{h,i,j}$ in S indexed over $h < n$, $i < n$, $j < k$, such that for all $h < n$:

$$\sigma_h = \mathbf{zip}_k \left(\sum_{i<n} a_{h,i,0}\sigma_i, \ldots, \sum_{i<n} a_{h,i,k-1}\sigma_i \right).$$

Taking the derivative and second derivative of each σ_h using (4), we obtain:

$$\sigma'_h = \mathbf{zip}_k \left(\sum_{i<n} a_{h,i,1}\sigma_i, \ldots, \sum_{i<n} a_{h,i,0}\sigma'_i \right)$$

$$\sigma''_h = \mathbf{zip}_k \left(\sum_{i<n} a_{h,i,2}\sigma_i, \ldots, \sum_{i<n} a_{h,i,0}\sigma'_i, \sum_{i\leq n} a_{h,i,1}\sigma'_i \right)$$

Hence, for each $\sigma \in \Sigma^+ := \Sigma \cup \{\sigma' \mid \sigma \in \Sigma\}$ and $j < k$, $\mathbf{unzip}_{j,k}(\sigma')$ is a linear combination of elements from Σ^+, and hence there is a finite S-weighted A_k-automaton (X, o, δ) and an $x \in X$, such that $[\![x]\!]_S = \sigma$ by Lemma 7. □

Example 11. We illustrate Proposition 10 with a well-known 2-regular sequence, which the composer Per Nørgård used in a variety of his compositions, and which

he called the *infinity sequence*[8] (A004718 on the Online Encyclopedia of Integer Sequences[9]):

$$\sigma = (0, 1, -1, 2, 1, 0, -2, 3, -1, 2, 0, 1, 2, \ldots) \in \mathbb{Z}^{\mathbb{N}}$$

This sequence can be characterized uniquely by the following equations:

$$\begin{aligned} o(x) &= 0 & x &= \mathbf{zip}_2(-x, x + y) \\ o(y) &= 1 & y &= \mathbf{zip}_2(y, y) \end{aligned}$$

(with x denoting σ). The **zip**-equations on the right-hand side are a system in the format of (9) and hence the sequence is 2-regular. Taking derivatives and second derivatives of the **zip**-equations, we now get using (4):

$$\begin{aligned} x' &= \mathbf{zip}_2(x + y, -x') & x'' &= \mathbf{zip}_2(-x', x' + y') \\ y' &= \mathbf{zip}_2(y, y') & y'' &= \mathbf{zip}_2(y', y') \end{aligned}$$

We can now compute the initial values of x' and y' as

$$\begin{aligned} o(x') &= o(\mathbf{zip}_2(x + y, -x')) = o(x + y) = o(x) + o(y) = 1 \\ o(y') &= o(\mathbf{zip}_2(y, y)) = o(y) = 1. \end{aligned}$$

Introducing new variables z and w representing x' and y' respectively, we now can specify a weighted automaton as the unique solution to the following system of **zip**-equations:

$$\begin{aligned} o(x) &= 0 & x' &= \mathbf{zip}_2(x + y, -z) \\ o(y) &= 1 & y' &= \mathbf{zip}_2(y, w) \\ o(z) &= 1 & z' &= \mathbf{zip}_2(-z, z + w) \\ o(w) &= 1 & w' &= \mathbf{zip}_2(w, w) \end{aligned}$$

The final homomorphism $[\![-]\!]_S$ maps x to Nørgård's infinity sequence:

$$[\![x]\!]_S = (0, 1, -1, 2, 1, 0, -2, 3, -1, 2, 0, 1, 2, \ldots)$$

We remark, however, that this weighted automaton is not minimal, as y and w both are mapped onto the constant sequence of ones.

Example 12. Another example, which can be constructed in the same manner as the previous example, is given by the following $\mathbb{N}$-weighted A_2-automaton:

$$\begin{aligned} o(x) &= 1 & x' &= \mathbf{zip}_2(x, x) \\ o(y) &= 1 & y' &= \mathbf{zip}_2(2y, 2y + x) \\ o(z) &= 1 & z' &= \mathbf{zip}_2(z, x + y) \end{aligned}$$

[8] `http://pernoergaard.dk/eng/strukturer/uendelig/uindhold.html`
[9] `http://oeis.org`

Here, the final homomorphism $[\![-]\!]_S$ maps x onto the constant stream of ones, y onto the stream of natural numbers, and z onto Kimberling's sequence (A003602 on OEIS):

$$[\![z]\!]_S = (1, 1, 2, 1, 3, 2, 4, 1, 5, 3, 6, 2, 7, 4, 8, \ldots)$$

We now prove the converse of Proposition 10.

Proposition 13. *Given any $k \geq 2$, a finite S-weighted A_k-automaton (X, o, δ), and a state $x \in X$, $[\![x]\!]_S$ is k-regular.*

Proof. By Lemma 8, we have

$$[\![x]\!]'_S = \mathbf{zip}_k([\![\delta(x)(\mathtt{1})]\!]_S, \ldots, [\![\delta(x)(\mathtt{k})]\!]_S)$$

and by (4) and (2) that

$$[\![x]\!]_S = \mathbf{zip}_k(o(x) + \mathcal{X}[\![\delta(x)(\mathtt{k})]\!]_S, [\![\delta(x)(\mathtt{1})]\!]_S, \ldots, [\![\delta(x)(\mathtt{k\text{-}1})]\!]_S). \tag{17}$$

Using the fact that $(\mathcal{X}[\![x]\!]_S)' = [\![x]\!]_S$, and applying again (4) and (2), we obtain:

$$\mathcal{X}[\![x]\!]_S = \mathbf{zip}_k(\mathcal{X}[\![\delta(x)(\mathtt{k\text{-}1})]\!]_S, o(x) + \mathcal{X}[\![\delta(x)(\mathtt{k})]\!]_S, [\![\delta(x)(\mathtt{1})]\!]_S, \ldots, [\![\delta(x)(\mathtt{k\text{-}2})]\!]_S) \tag{18}$$

By defining the set of generators

$$\Sigma = \{[\![x]\!]_S \mid x \in X\} \cup \{\mathcal{X}[\![x]\!]_S \mid x \in X\} \cup \{(1, 0, 0, \ldots)\}$$

the equations (17) and (18) show (via the zip-unzip isomorphism (6)) that for each generator $\sigma \in \Sigma$ and $j < k$, $\mathbf{unzip}_{j,k}(\sigma)$ is a linear combination of the generators. It follows from the definition that $[\![x]\!]_S$ is k-regular for all $x \in X$. □

We now can gather, from our previous results, the following equivalent characterizations of k-regular sequences, arriving at our main theorem:

Theorem 14. *The following are equivalent for any stream $\sigma \in S^{\mathbb{N}}$:*

1. *σ is k-regular.*
2. *σ is generated by a state in a finite weighted A_k-automaton.*
3. *σ is defined by a linear system of $\mathbf{zip}_k$-behavioral differential equations over a finite set of variables.*
4. *$h_L(\sigma) \in S\langle\langle A_k \rangle\rangle$ is a recognizable power series.*

Proof. 1 ⇒ 2 is Proposition 10. 2 ⇒ 1 is Proposition 13. 2 ⇔ 3 follows from Lemma 8 and Lemma 9. 2 ⇔ 4 follows from Proposition 5. □

The equivalence 1 ⇔ 4 in the above theorem combined with the fact that h_L is a bijection with inverse h_S directly yields the following corollary, establishing a bijective correspondence between k-regular power series and recognizable power series on the alphabet A_k:

Corollary 15. *For all formal power series $\rho \in S\langle\langle A_k \rangle\rangle$ over variables A_k, we have: ρ is recognizable if and only if the sequence $h_S(\rho) \in S^{\mathbb{N}}$ is k-regular.*

The equivalence $1 \Leftrightarrow 4$ of Theorem 14 is analogous to [1, Theorem 4.3], which says that $\sigma \in \mathbb{Z}^{\mathbb{N}}$ is k-regular if and only if the formal power series $\sum_{n<\omega} \sigma(n)\bar{\xi}(n)$ is rational (or equivalently, recognizable), where $\bar{\xi}\colon \mathbb{N} \to B_k^* \backslash B_k^* 0$ is the inverse of the bijection obtained by restricting the standard base k numeration $\xi\colon B_k^* \to \mathbb{N}$ to words not ending in `0`. In contrast with our results, [1, Theorem 4.3] cannot be extended to a bijective correspondence between the classes of k-regular sequences and rational power series over variables in B_k as there are rational power series $\rho \in S\langle\langle B_k \rangle\rangle$ that do not correspond to any k-regular sequence via standard base k numeration. In other words, there is no analogue of our Corollary 15 in the presentation using standard base k numeration from [1].

Zero-consistent automata. It is also possible to characterize k-regular sequences by a class of weighted automata that read the numeration used in [1, Theorem 4.3] (standard base k backwards). This class of automata is provided by the so-called zero-consistent S-weighted B_k-automata which are a mild generalisation of the zero-consistent automata that have been described in [12].

The defining feature for zero-consistent automata is that the (immediate) output of the automaton does not change when reading letter `0`. Intuitively, reading letter `0` corresponds to moving from a state generating stream σ to a state that generates the stream $\mathbf{unzip}_{0,k}(\sigma)$. Zero-consistency reflects the fact that the head of any stream σ is equal to the head of the stream $\mathbf{unzip}_{0,k}(\sigma)$. More generally, reading a letter `j` with $0 \leq j < k$ in this setting corresponds to moving from a state that represents a stream σ to a state that represents the stream $\mathbf{unzip}_{j,k}(\sigma)$. As Definition 2 of k-regular sequences is based on the $\mathbf{unzip}_k$-operations (and not on the $\mathbf{unzip}_k \circ \mathbf{tail}$-operations as used in the definition of 8) it is rather straightforward to prove that the k-regular sequences are precisely the ones that can be generated using zero-consistent automata.

3.4 Connections to automatic sequences

All of the results that were presented earlier in this section can be seen as generalizations of corresponding results about the k-automatic sequences. In this subsection, we will state (without proofs, but with some explanations about the relationships) the corresponding theorems. We remark that, unlike in the case of k-regular sequences, the results for automatic sequences are, in this form, well-known in the literature. The following definition is analogous to Definition 2, but uses the more restrictive condition that the k-kernel is finite, rather than finitely generated:

Definition 16. *A sequence $\sigma \in S^{\mathbb{N}}$ is* k-automatic *iff the k-kernel of σ is finite, or equivalently, iff there is a finite set of sequences $\Sigma = \{\sigma_0, \ldots \sigma_{n-1}\}$ with $\sigma \in \Sigma$, such that for all $h, j \in \mathbb{N}$ with $h < n$ and $j < k$, $\mathbf{unzip}_{j,k}(\sigma_h) \in \Sigma$.*

We can now obtain the following results, using essentially the same techniques have been used in this paper for k-regular sequences:

- *A sequence $\sigma \in S^{\mathbb{N}}$ is k-automatic if and only if there is a finite A_k-automaton (Q, o, δ) such that $\sigma = [\![q]\!]_S$ for some $q \in Q$.* (Equivalent to [2, Theorem 5.2.7])
- *A sequence σ is k-automatic iff it takes finitely many values $v_1, \ldots, v_n$, and for each v_i, the language $\{w \in A_k^* \mid h_L(\sigma)(w) = v_i\}$ is regular.* (Analogous to [2, Lemma 5.2.6], with change from standard to bijective numeration)

4 Relation to divide and conquer recurrences

Divide and conquer recurrences, which have been considered for example in [10] and [20], can somewhat informally be seen as—in the case of $k = 2$, to which we will restrict ourselves in this section—sequences σ where $\sigma(0)$ is given, and for each n, $\sigma(n)$ is defined in terms of $\sigma(\mathbf{floor}((n-1)/2))$, $\sigma(\mathbf{ceil}((n-1)/2))$, and polynomials in n.

In this section, we will establish a close link between divide and conquer recurrences satisfying a number of 'natural' conditions, and the k-regular sequences, by showing that their sequences occur as 2-regular sequences.

We will restrict ourselves to special (more precisely defined) restricted versions of divide and conquer recurrences. To be precise, we will consider recurrences of the form

$$\sigma(2n) = a_2\sigma(n-1) + a_3\sigma(n) + \tau_1(n) \qquad \sigma(2n+1) = a_1\sigma(n) + \tau_0(n)$$

where a_1, a_2, a_3 are scalars from the semiring S, and τ_1 and τ_0 are themselves 2-regular sequences. We furthermore hold the assumption that $\sigma(0) = 0$ (we will later see that this assumption can be relaxed).

Now observe that

$$\sigma(2n+2) = \sigma(2(n+1)) = a_2\sigma(n) + a_3\sigma(n+1) + \tau_1'(n).$$

As a result of the equalities $\sigma(2n+1) = (\mathbf{even}(\sigma'))(n)$ and $\sigma(2n+2) = (\mathbf{odd}(\sigma'))(n)$ we now derive

$$\begin{aligned} &(\mathbf{even}(\sigma'))(n) = (a_1\sigma + \tau_0)(n) \\ \text{and} \qquad &(\mathbf{odd}(\sigma'))(n) \ = (a_2\sigma + a_3\sigma' + \tau_1')(n) \end{aligned}$$

and hence also

$$\mathbf{even}(\sigma') = a_1\sigma + \tau_0 \qquad \mathbf{odd}(\sigma') = a_2\sigma + a_3\sigma' + \tau_1'.$$

A large number of combinatorial problems can be expressed by means of divide and conquer recurrences of this type, and can be transformed using this construction, including problems such as the Josephus problem, the sequence of all numbers whose ternary representation does not contain the digit 1, or

does not contain the digit 2, counting of quicksort insertions, and a variety of other combinatorial and graph-theoretic problems. An overview of many of these examples can be found at `http://oeis.org/somedcgf.html`. One of the questions asked here, is whether all the examples presented there are indeed 2-regular. We will soon see that this question can be answered positively.

Example 17. As an example illustrating the construction, the recurrence given by

$$\sigma(0) = 0 \qquad \sigma(2n) = \sigma(n) + \sigma(n-1) + 2n - 2 \qquad \sigma(2n+1) = 2\sigma(n) + 2n - 1$$

specifying the sorting numbers (OEIS A001855) can first be transformed into:

$$\sigma(0) = 0 \qquad \sigma(2n+1) = 2\sigma(n) + 2n - 1 \qquad \sigma(2n+2) = \sigma(n+1) + \sigma(n) + 2n$$

We can coinductively specify the streams **ones** and **nats** by

$$\begin{array}{lll} o(\mathbf{ones}) = 1, & \mathbf{even}(\mathbf{ones}') = \mathbf{ones}, & \mathbf{odd}(\mathbf{ones}') = \mathbf{ones} \\ o(\mathbf{nats}) = 1, & \mathbf{even}(\mathbf{nats}') = 2 \cdot \mathbf{nats}, & \mathbf{odd}(\mathbf{nats}') = 2 \cdot \mathbf{nats} + \mathbf{ones} \end{array}$$

and now we can transform the earlier recurrence into the behavioral differential equation:

$$o(\sigma) = 0 \qquad \mathbf{even}(\sigma') = 2\sigma + 2 \cdot \mathbf{nats} - \mathbf{ones} \qquad \mathbf{odd}(\sigma') = \sigma' + \sigma + 2 \cdot \mathbf{nats}$$

We can now establish that σ is again 2-regular:

Proposition 18. *Let τ_0 and τ_1 be 2-regular sequences over a semiring S, and let a_0, a_1, a_2, and a_3 be elements of S. Then there is a unique sequence σ satisfying*

$$\sigma(0) = a_0 \qquad \mathbf{even}(\sigma') = a_1\sigma + \tau_0 \qquad \mathbf{odd}(\sigma') = a_2\sigma + a_3\sigma' + \tau_1$$

which is again 2-regular.

Proof. If τ_0 and τ_1 are 2-regular, there are finite weighted automata (X_0, o_0, δ_0) and (X_1, o_1, δ_1) with elements $x_0 \in X_0$, $x_1 \in X_1$ such that $[\![x_0]\!]_S = \tau_0$ and $[\![x_1]\!]_S = \tau_1$.

Observe that, if σ satisfies the above equation, we can directly derive:

$$\sigma'(0) = a_1a_0 + o(\tau_0) \qquad \mathbf{even}(\sigma'') = a_2\sigma + a_3\sigma' + \tau_1 \quad \mathbf{odd}(\sigma'') = a_1\sigma' + \tau_0'$$

We thus specify a system $(X_0 \cup X_1 \cup \{y, z\}, o, \delta)$ satisfying the behavioral differential equations for X_0 and X_1 as before, and additionally:

$$\begin{array}{llll} o(y) = a_0 & \mathbf{even}(y') = a_1y + o(x_0) & \mathbf{odd}(y') = a_2y + a_3z + x_1 \\ o(z) = a_1a_0 + o(x_0) & \mathbf{even}(z') = a_2y + a_3z + x_1 & \mathbf{odd}(z') = a_1z + x_0' \end{array}$$

By Lemma 9, this system has a unique solution, in which $[\![y]\!]_S$ satisfies the equations for σ and $[\![z]\!]_S = [\![y]\!]_S'$. Because, given systems for τ_0 and τ_1, any solution to the original equation for σ has to satisfy all equations in the composite system, the solution for σ also is unique. □

This construction now leads to a large collection of examples, directly derivable from specifications of divide and conquer-recurrences. We have used the overview on `http://oeis.org/somedcgf.html` as a basis for the examples that follow. We will not explicitly specify the constructions used: however, all examples have been obtained by a combination of the constructions presented in Propositions 10 and 18. All the examples that follow are 2-regular over the ring $\mathbb{Z}$.

Example 19. We can now specify a large amount of sequences as 2-regular sequences. In most cases, we just need the three variables x (over the ring $\mathbb{Z}$, and in cases where no negative coefficients occur also over the semiring $\mathbb{N}$), **nats** and **ones**, where x represents the sequence itself; in some cases we need a fourth variable x', but these cases still fit in the format of Proposition 18.

$o(x)$	$\mathbf{even}(x')$	$\mathbf{odd}(x')$	OEIS
1	$4x$	$4x + \mathbf{ones}$	A000695
0	$2x + 2 \cdot \mathbf{nats} - \mathbf{ones}$	$x + x' + 2 \cdot \mathbf{nats}$	A001855
0	$2x + 2 \cdot \mathbf{nats}$	$x + x' + 2 \cdot \mathbf{nats} + \mathbf{ones}$	A003314
1	$-x$	$x + \mathbf{ones}$	A004718
1	$3x$	$3x + \mathbf{ones}$	A005836
1	$2x - 1$	$x + x'$	A006165
0	$x + \mathbf{ones}$	0	A007814
1	$-x$	$\mathbf{ones} - x$	A065359
0	$2(x + \mathbf{nats} + \mathbf{ones})$	$x + x' + 2(\mathbf{nats} + \mathbf{ones})$	A067699
1	$2x + \mathbf{ones}$	$2 \cdot \mathbf{nats} + \mathbf{ones}$	A086799

Here, for example, A000695 is the Moser-de Bruijn sequence which is the ordered sequence of all numbers that can be written as a sum of distinct powers of 4; the nth element of A001855 is the maximal number of comparisons needed to sort n elements by binary insertion; and A005836 is the ordered sequence of all numbers whose base 3 representation contains no 2. For specifications of the other examples, as well as a large amount of background information about these sequences, we refer to the entries at the OEIS.

Remark 20. All the examples in this section can be easily implemented in the functional programming language Haskell. Building on the existing work on stream calculus in Haskell (e.g. [13], [11], [21]), the **zip** operation of arbitrary arity can be specified in Haskell using the (behavioral differential) equation

```
xzip (s:t) = head s : xzip (t ++ [tail s])
```

allowing us to specify all of the above examples with a single line of Haskell code. For example, the (tail of the) Nørgård sequence can now be specified by:

```
n = 1 : xzip [-n, n + ones]
```

5 Conclusions and Future Work

We have given a coalgebraic (or automata-theoretic) as well as an algebraic characterization of k-regular sequences: the k-regular sequences are exactly the sequences that are generated by finite S-weighted automata over the k-letter alphabet A_k. They are also exactly the sequences that can be defined by a finite system of linear **zip**-behavioral differential equations. We also showed that there is an isomorphism between the final A_k-automaton of formal power series and the A_k-automaton of sequences (which is then also final). This isomorphism is given by bijective k-adic numeration, and we derived from it directly that k-regular sequences are in bijective correspondence with recognizable formal power series over A_k.

The following table gives an overview of the classes of sequences and formal power series that are generated by finite deterministic, respectively weighted, automata with respect to three different final automata:

S semiring	deterministic automata	weighted automata
1-letter $(S^{\mathbb{N}}, \mathbf{head}, \mathbf{tail})$	eventually periodic (= 1-automatic)	recognizable (= 1-regular)
k-letter $(S\langle\langle A_k \rangle\rangle, O, \Delta)$	S-simple power series	recognizable power series
k-letter $(S^{\mathbb{N}}, \mathbf{head}, \mathbf{unzip}_k \circ \mathbf{tail})$	k-automatic sequences	k-regular sequences

If S is finite, then the right-hand "weighted" column collapses and becomes equal to the left-hand "deterministic" column. Hence every finite weighted automaton with output in a finite S is equivalent to a finite deterministic automaton with output in S.

Generalization to other numeration systems. The k-adic numeration system appears to be a 'nice' choice because it is bijective and hence gives a bijective correspondence between k-regular sequences and recognizable series (which is not the case with standard base k numeration). It may be interesting to investigate whether corresponding results can be obtained with respect to other (bijective or not) numeration systems.

Relation to S-algebraic sequences. In [7], a coalgebraic characterization of algebraic power series, which generalizes context-free languages, was provided. Here, (constructively) algebraic power series can be described using systems of behavioral differential equations over a finite set X, where each derivative is given as a polynomial over X with coefficients in S. As with weighted automata, such a system can be determinized into an automaton whose states are polynomials over X with coefficients in S.

It would be interesting to see if we can connect this notion of (constructive) algebraicity to the existing notion of k-context-free sequences (see e.g. [15]),

using techniques analogous to the ones used in this paper to connect k-regular sequences, recognizable sequences and power series. As a final remark, we note that it is easily possible to use the isomorphism from Section 3.1 as the basis for a definition of k-context-freeness, however, we note that the product inherited on $S^{\mathbb{N}}$ from the convolution product on $S\langle\langle A_k \rangle\rangle$ differs from the standard convolution product on $S^{\mathbb{N}}$ except for the case where $k = 1$.

References

1. Jean-Paul Allouche and Jeffrey O. Shallit. The ring of k-regular sequences. *Theoretical Computer Science*, 98:163–197, 1992.
2. Jean-Paul Allouche and Jeffrey O. Shallit. *Automatic Sequences – Theory, Applications, Generalizations*. Cambridge University Press, 2003.
3. Jean-Paul Allouche and Jeffrey O. Shallit. The ring of k-regular sequences, II. *Theoretical Computer Science*, 307:3–29, 2003.
4. Falk Bartels. *On Generalized Coinduction and Probabilistic Specification Formats*. PhD thesis, Vrije Universiteit Amsterdam, 2004.
5. Jason P. Bell. On the values attained by a k-regular sequence. *Advances in Applied Mathematics*, 34:634–643, 2005.
6. Jean Berstel and Christophe Reutenauer. *Noncommutative Rational Series with Applications*. Cambridge University Press, 2011.
7. Marcello M. Bonsangue, Jan Rutten, and Joost Winter. Defining context-free power series coalgebraically. In Dirk Pattinson and Lutz Schröder, editors, *Proceedings of Coalgebraic Methods in Computer Science (CMCS 2012)*, volume 7399 of *LNCS*, pages 20–39. Springer, 2012.
8. Zoltán Ésik and Andreas Maletti. The category of simulations for weighted tree automata. *International Journal of Foundations of Computer Science (IJFCS)*, 22(8):1845–1859, 2011.
9. Clemens Grabmayer, Jörg Endrullis, Dimitri Hendriks, Jan Willem Klop, and Lawrence S. Moss. Automatic sequences and zip-specifications. In *Proceedings of Logic in Computer Science (LICS 2012)*, pages 335–344. IEEE Computer Society Press, 2012.
10. Ronald L. Graham, Donald E. Knuth, and Oren Patashnik. *Concrete Mathematics: A Foundation for Computer Science*. Addison-Wesley Longman Publishing Co., Inc., 2nd edition, 1994.
11. Ralf Hinze. Concrete stream calculus—an extended study. *Journal of Functional Programming*, 20(5-6):463–535, 2011.
12. Clemens Kupke and Jan Rutten. On the final coalgebra of automatic sequences. In Robert L. Constable and Alexandra Silva, editors, *Logic and Program Semantics*, volume 7230 of *LNCS*, pages 149–164. Springer, 2012.
13. M. Douglas McIlroy. The music of streams. *Information Processing Letters*, 77(2-4):189–195, 2001.
14. Nax P. Mendler, Prakash Panangaden, and Robert L. Constable. Infinite objects in type theory. In *Proceedings of Logic in Computer Science (LICS 1986)*, pages 249–255. IEEE Computer Society Press, 1986.
15. Yossi Moshe. On some questions regarding k-regular and k-context-free sequences. *Theoretical Computer Science*, 400:62–69, 2008.
16. Sanjay V. Rajopadhye and Prakash Panangaden. Verification of systolic arrays: A stream function approach. In *International Conference on Parallel Processing (ICPP'86)*, pages 773–775. IEEE Computer Society Press, 1986.

17. Jan Rutten. Universal coalgebra: a theory of systems. *Theoretical Computer Science*, 249(1):3–80, 2000.
18. Jan Rutten. Behavioural differential equations: a coinductive calculus of streams, automata, and power series. *Theoretical Computer Science*, 308(1-3):1–53, 2003.
19. Alexandra Silva, Filippo Bonchi, Marcello Bonsangue, and Jan Rutten. Generalizing determinization from automata to coalgebras. *Logical Methods in Computer Science*, 9(1), 2013.
20. Ralf Stephan. Divide-and-conquer generating functions. Part I. Elementary sequences. *ArXiv Mathematics e-prints*, July 2003.
21. Joost Winter. QStream: a suite of streams. In Reiko Heckel and Stefan Milius, editors, *Proceedings of Algebra and Coalgebra in Computer Science (CALCO 2013)*, volume 8089 of *LNCS*, 2013.