Context-free coalgebras

Joost Winter, Marcello M. Bonsangue, Jan J. M. M. Rutten

January 20, 2013

Abstract

In this article, we provide a coalgebraic account of parts of the mathematical theory underlying context-free languages. We characterize context-free languages, and power series and streams generalizing or corresponding to the context-free languages, by means of systems of behavioural differential equations; and prove a number of results, some of which are new, and some of which are new proofs of existing theorems, using the techniques of bisimulation and bisimulation up to linear combinations. Furthermore, we establish a link between automatic sequences and these systems of equations, allowing us to, given an automaton generating an automatic sequence, easily construct a system of behavioural differential equations yielding this sequence as a context-free stream.

Contents

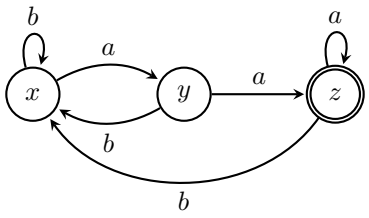
1 Introduction

During the last 15 years, a coalgebraic picture of the theory of automata and formal languages has emerged and been further developed. The earliest traces of this picture date back to work by Janusz A. Brzozowski who introduced the idea of *derivatives* of regular expressions in [Brz64]. This idea of input derivatives was linked to the more abstract notion of a F-coalgebra for a functor F by the third author of the present article in [Rut98], and further developed in e.g. [Rut03], [Rut08], and [SBBR10].

In [WBR11], we extended this coalgebraic picture further, giving a characterization of the context-free languages as solutions to systems of *behavioural differential equations*, in which all Brzozowski derivatives are presented as polynomials over the set of variables, and in [BRW12], we have generalized the theory to formal power series in noncommuting variables, providing a new characterization of so-called *algebraic* or *context-free* power series, coinciding with the familiar generalization from context-free languages to algebraic power series.

In the present article, we will provide a more systematic account of this framework. It turns out that important classes of formal languages and formal power series—finitary, rational, and algebraic (or context-free) series, respectively—can each be characterized as solutions to some format of systems of behavioural differential equations.

The simplest class of systems, which will be recalled in Section 2, consists of equations in which each derivative is presented simply as another state (or, equivalently, as a *variable* or *nonterminal*): these systems can be seen as describing deterministic automata and generalizations thereof. As an example of such a system, consider the deterministic automaton (presented without initial state)

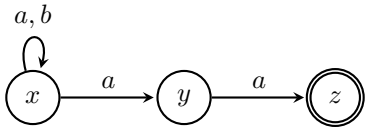


which corresponds to the following system of behavioural differential equations:

$$\begin{array}{lll} o(x) = 0 & x_a = y & x_b = x \\ o(y) = 0 & y_a = z & y_b = x \\ o(z) = 1 & z_a = z & z_b = x \end{array}$$

Here, the equation $o(x) = 0$ can be read as 'x is a non-accepting state', $o(z) = 1$ as 'z is an accepting state', and $x_a = y$ as 'x makes an a-transition to y'.

In Section 3, we will combine the notions of automata and modules yielding the notion of a linear automaton, and recall some of the main ideas and results regarding rational power series and linear automata. This class of series can be characterized by systems of behavioural differential equations, in which every derivative is presented as a linear combination of states. For example, the nondeterministic automaton



is in direct correspondence with the system of behavioural differential equations

$$\begin{array}{lll} o(x) = 0 & x_a = x + y & x_b = x \\ o(y) = 0 & y_a = z & y_b = 0 \\ o(z) = 1 & z_a = 0 & z_b = 0. \end{array}$$

In this system, for example, the equation $x_a = x + y$ can be read as 'x makes a-transitions to x and y', and $y_b = 0$ can be read as 'y makes no b-transitions'.

In Section 4 we will consider a class of systems which yields a class of automata with an even richer structure than that of linear automata, and establish in Section 4.1 its equivalence with existing notions of *context-freeness* or *algebraicity*. Here, the derivatives of all nonterminals are presented as *polynomials* with coefficients in the underlying semiring.

An example of such a system is the following

$$\begin{array}{lll} o(x) = 1 & x_a = xy & x_b = 0 \\ o(y) = 0 & y_a = 0 & y_b = 1 \end{array}$$

and we will see in Section 4 that in this system x can be interpreted as the language $\{a^n b^n \mid n \in \mathbb{N}\}$. Although closely related to pushdown automata (with a single state), these systems correspond more directly to context-free grammars in Greibach normal form. For example, the system above is in direct correspondence with the following grammar:

$$\begin{array}{c} x \to 1 \mid axy \\ y \to b \end{array}$$

Next, we will consider of a number of preservation results, and concrete constructions of new context-free systems of behavioural differential equations from old systems, providing some deeper links with the classical theory of context-free languages and automatic sequences. In Sections 6, we will introduce the **zip** operation and a number of related operations, and show that these operations all preserve context-freeness. Finally, in 7, we will connect our notion of context-free power series to the theory of automatic sequences by showing that streams over finite fields are context-free if and only if they are algebraic, and furthermore present a simple method to extract systems of behavioural differential equations from automatic sequences.

The subject matter of this article can perhaps best be characterized as *coalgebraic automata theory*. Although closely related to the traditional, algebraic, approach to automata theory, and benefitting greatly from usage of traditional algebraic structures, our approach differs by making extensive usage of *bisimulations* (and bisimulations up to linear combinations), *systems of behavioural differential equations*, and *final coalgebra semantics*, and by altogether omitting the usage of matrices. Combined, these techniques enable us to provide proofs that are often more concise than the traditional ones, and different in style, often establishing close connections between the behaviour of automata, and algebraic properties characterizable by simple equations. Our approach also relates to the more abstract world of *universal coalgebra* [Rut00], in the sense that many of the notions that will be used in this article can be directly seen as instances of more abstract, categorically defined, notions from universal coalgebra.

The present article is an extended and thoroughly revised version of [BRW12]. Other than providing a more systematic account of context-free power series and streams, many of the propositions and theorems are presented with new, coinductive, proofs, many of which use the technique of bisimulations up to linear combinations. Most of the propositions and theorems will be accompanied with

full proofs, partly because this article has a strong focus on the proof technique of bisimulation and bisimulation up to linear combinations. Compared to [BRW12], the present article also makes a shift, away from term algebras and towards the usage of polynomials as fundamental structures; and more broadly speaking, a step away from category theory and towards classical algebra.

Acknowledgements: We would like to express our gratitude to Christophe Reutenauer, for pointing us towards the article by Michel Fliess [Fli74], in which two notions of algebraicity are related, as well as to Jean Berstel, Jeffrey Shallit, Jean-Paul Allouche and Arto Salomaa. We thank the referees of the original CMCS paper for providing constructive comments, and we thank Jurriaan Rot, Helle Hansen, and Alexandra Silva for discussions and criticism. Finally, we would like to thank the participants in the Representing Streams workshop that was held in Leiden in December 2012, for discussion, criticism, and exploration of new ideas.

2 Preliminaries

We will first start by covering some of the background material: some parts of the material covered in this section can be considered 'standard' or classical (monoids, semirings, formal power series); the part on automata as coalgebras, although nonstandard in a sense, has been covered extensively in earlier papers (see e.g. [Rut98], [BBB$^+$12]). In order to both increase accessibility, and to make this article as self-containing as possible, we include all definitions and the most important results from these underlying frameworks in this section.

2.1 Monoids and automata

2.1.1 Monoids

A *monoid* $(M, \cdot, 1)$ consists of a set M, together with a distinguished element $1 \in M$, and a multiplication operator $\cdot : M \times M \to M$, such that 1 is a *unit* for M, that is, for all $m \in M$ $1 \cdot m = m = m \cdot 1$, and the operator $\cdot$ is *associative*, that is, for all $m, n, p \in M$, $m \cdot (n \cdot p) = (m \cdot n) \cdot p$

A monoid is called *commutative* if for all $m, n \in M$, $mn = nm$. (We will follow the usual convention of omitting the multiplication operator $\cdot$ whenever possible and convenient.) Often commutative monoids will be represented in additive notation, using the symbols 0 and $+$ rather than 1 and $\cdot$.

Given two monoids $(M, \cdot_M, 1_M)$ and $(N, \cdot_N, 1_N)$, we say a function $f : M \to N$ is a monoid morphism whenever $f(m \cdot_M n) = f(m) \cdot_N f(n)$ and $f(1_M) = 1_N$.

Given a finite alphabet A, let A^* denote the set of all *words* over A, that is, lists of finite sequences of elements from A. We use the symbol 1 to denote the empty word—that is, the sole word of length zero, and given words

$$v = a_1 \dots a_m \qquad \text{and} \qquad w = b_1 \dots b_n,$$

let the product $v \cdot w$ represent the concatenation

$$vw = a_1 \dots a_m b_1 \dots b_n.$$

It is easy to see that the unit and associativity laws for monoids hold here, turning the structure $(A^*, \cdot, 1)$ into a monoid. Moreover, from a categorical point of view this monoid is the *free* monoid over the set A: given a monoid $(M, \cdot_M, 1_M)$ and a function $f : A \to M$, there is a unique monoid morphism $\hat{f} : A^* \to M$ such that $\hat{f} \circ \eta_A = f$. Here η_A is a mapping from A to A^*, mapping each alphabet symbol $a \in A$ onto the word a of length 1, which moreover can be seen as the *unit* of the star *monad*.

2.1.2 Automata as coalgebras

In this article, an *automaton* over a finite alphabet A with output in a semiring S, or an S-*automaton*, consists of a triple

$$(Q, o, \delta)$$

where

1. Q is a set of states;
2. $o : Q \to S$ is a function assigning an *output value* in S to each state; and
3. $\delta : Q \to Q^A$ assigns a mapping from alphabet symbols to states to each state. We call δ the *transition function* of the automaton.

We call an automaton (Q, o, δ) *finite* whenever Q is finite, and often simply refer to such an automaton as Q, when no confusion about the interpretation of the o and δ operations is likely to arise. Compared to the classical presentation, the main difference is the absence of an initial state, but we lose—at least, not as far as this paper is concerned—nothing in doing so, as we can always assume a state from the automaton as 'initial' if so desired. On the other hand, by omitting initial states, as we will see soon, we gain the advantage of the existence of a unique automaton morphism from any given automaton into a final automaton, which turns out to be a highly useful property.

Throughout this article, we will usually write q_a instead of $\delta(q)(a)$, and call q_a the a-*derivative* of q. We can extend the transition function δ to a transition function δ^* over words $w \in A^*$ inductively by setting

$$\delta^*(q)(1) = q \qquad \text{and} \qquad \delta^*(q)(aw) = \delta^*(\delta(q)(a))(w). \tag{1}$$

Since $\delta^*(q)(a) = \delta^*(\delta(q)(a))(1) = \delta(q)(a)$ for all $q \in Q$ and $a \in A$, the functions δ and δ^* are compatible, and (1) can be reformulated in terms of word derivatives by the equations

$$q_1 = q \qquad \text{and} \qquad q_{aw} = (q_a)_w$$

which hold for all $q \in Q$, $a \in A$, and $w \in A^*$. The second of these equations can be generalized using a simple lemma:

Lemma 1. *For all $v, w \in A^*$, $q_{vw} = (q_v)_w$.*

Proof. Induction on the length of v. If $v = 1$, then $q_{1w} = q_w = (q_1)_w$. If $v = au$ for some $a \in A$ and $u \in A^*$, use the inductive hypothesis that $q_{uw} = (q_u)_w$ holds. Now observe

$$q_{vw} = q_{(au)w} = q_{a(uw)} = (q_a)_{uw} = ((q_a)_u)_w = (q_{au})_w = (q_v)_w$$

and the proof is complete. □

A *morphism* between two automata Q and R is a mapping $h : Q \to R$ such that for all $q \in Q$ and $a \in A$, we have $o(q) = o(h(q))$ and $h(q_a) = h(q)_a$.

Lemma 2. *For any morphism $h : Q \to R$, any $q \in Q$ and any $w \in A^*$, $h(q_w) = h(q)_w$.*

Proof. Induction on the length of w. Base case: $h(q_1) = h(q) = h(q)_1$. Inductive case: assume $h(q_v) = h(q)_v$ for all $q \in Q$, then also $h(q_{av}) = h((q_a)_v) = h(q_a)_v = h(q)_{av}$. □

As an important example, consider the automaton $(S\langle\langle A\rangle\rangle, o, \delta)$, where $S\langle\langle A\rangle\rangle$ is defined as the function space

$$\{f \mid f : A^* \to S\}$$

from A^* to S. For now, S can be any set, but once we start adding more structure to our automata, we will assume S to be a semiring. We let the Greek letters σ and τ denote elements from $S\langle\langle A\rangle\rangle$, and, following classical notation we write

$$(\sigma, w)$$

for $\sigma(w)$. We define the behaviour of $S\langle\langle A\rangle\rangle$, for any $\sigma \in S\langle\langle A\rangle\rangle$, $a \in A$ and $w \in A^*$, by

$$o(\sigma) = (\sigma, 1) \qquad \text{and} \qquad (\sigma_a, w) = (\sigma, aw).$$

Observe that this definition completely describes the functions o and δ, so that we have defined, completely and unambiguously, the automaton structure of $S\langle\langle A\rangle\rangle$. We again can easily generalize the equality $(\sigma_a, w) = (\sigma, aw)$ inductively:

Lemma 3. *For all $v, w \in A^*$, we have $(\sigma_v, w) = (\sigma, vw)$, and hence $o(\sigma_w) = (\sigma_w, 1) = (\sigma, w)$.*

Proof. Induction on the length of v. If $v = 1$, then $(\sigma_v, w) = (\sigma, w) = (\sigma, vw)$. If $v = au$ for some $a \in A$ and $u \in A^*$, use the inductive hypothesis that for all $\tau \in S\langle\langle A\rangle\rangle$, $o(\tau_u, w) = (\tau_u, w)$ holds, and observe $(\sigma_v, w) = ((\sigma_a)_u, w) = (\sigma_a, uw) = (\sigma, auw) = (\sigma, vw)$. □

The following proposition establishes the fact that $S\langle\langle A\rangle\rangle$ is a final automaton or, in more general terminology, a final coalgebra (for the functor $S \times (-)^A$):

Proposition 4. *For any S-automaton Q over an alphabet A, there exists a unique morphism from Q to the automaton $S\langle\langle A\rangle\rangle$.*

Proof. Consider the mapping $[\![\ \ ]\!] : Q \to S\langle\langle A\rangle\rangle$ defined by

$$([\![q]\!], w) = o(q_w).$$

To see that this mapping is a morphism, we have to show $[\![q_a]\!] = [\![q]\!]_a$ for all $q \in Q$ and $a \in A$. But this is the case, since for any $w \in A^*$, we have

$$([\![q_a]\!], w) = o((q_a)_w) = o(q_{aw}) = ([\![q]\!], aw) = ([\![q]\!]_a, w).$$

For unicity, assume that h is a morphism from Q to $S\langle\langle A\rangle\rangle$. But this gives

$$(h(q), w) = o(h(q)_w) = o(h(q_w)) = o(q_w) = o([\![q]\!]_w) = ([\![q]\!], w)$$

for arbitrary $q \in Q$ and $w \in A^*$, so we must have $h = [\![\ \ ]\!]$. □

We will next introduce bisimulations, which can be seen as a relational generalization of morphisms. Bisimulations are, in the coalgebraic framework, an important technique to establish behavioural equivalence between different automata. Given automata (P, o_P, δ_P) and (Q, o_Q, δ_Q), we say a relation $R \subseteq P \times Q$ is a *bisimulation* if and only if, whenever $(p, q) \in R$, we have

1. $o_P(p) = o_Q(q)$, and
2. for all $a \in A$, $(p_a, q_a) \in R$.

Proposition 5. *If $R \subseteq P \times Q$ is a bisimulation, and $(p, q) \in R$, then $[\![p]\!] = [\![q]\!]$.*

Proof. We can define an automaton structure on R by defining, for $(p, q) \in R$, $o((p, q)) = o(p)(= o(q))$, and $(p, q)_a = (p_a, q_a)$. Now observe that the projection functions $\pi_1 : R \to P$ and $\pi_2 : R \to Q$ are morphisms as $o(\pi_1(p, q)) = o(p) = o((p, q))$, and $\pi_1((p, q)_a) = \pi_1(p_a, q_a) = p_a = (\pi_1(p, q))_a$, and similarly for π_2.

We now obtain morphisms $[\![\ \ ]\!] \circ \pi_1$ and $[\![\ \ ]\!] \circ \pi_2$ from R into the final coalgebra, but by Proposition 4, it follows that these morphisms must be identical, and hence, given $(p, q) \in R$,

$$[\![p]\!] = [\![\pi_1(p, q)]\!] = [\![\pi_2(p, q)]\!] = [\![q]\!]$$

completing the proof. □

2.2 Semirings and beyond

2.2.1 Semirings

A *semiring* $(S, \cdot, +, 1, 0)$ consists of a set S, such that $(S, \cdot, 1)$ is a monoid, and $(S, +, 0)$ is a commutative monoid, where addition distributes over multiplication, that is, for all $r, s, t \in S$, $r(s+t) = rs+rt$ and $(r+s)t = rt+st$, and 0 is a multiplicative annihilator, that is, for all $r \in S$, $0r = 0 = r0$. Given semirings S and T, a function $f : S \to T$ is called a semiring morphism whenever it is a monoid morphism with respect to both the additive and multiplicative monoid underlying the semiring.

We call a semiring $(S, \cdot, +, 1, 0)$ *commutative* whenever the multiplicative monoid $(S, \cdot, 1)$ is commutative, and *idempotent* whenever for all $s \in S$, the equality $s + s = s$ holds.

Two semirings that will be of importance in this article, both commutative, are the natural numbers $(\mathbb{N}, \cdot, +, 1, 0)$ with the familiar addition and multiplication, and the Boolean algebra $\mathbb{B} = \{0, 1\}$ of two elements $(\mathbb{B}, \cdot, +, 1, 0)$ with $+$ and $\cdot$ representing the functions max and min, respectively. These two semirings $\mathbb{N}$ and $\mathbb{B}$ are furthermore *initial* objects in the category of semirings and the category of idempotent semirings, respectively: given any semiring (respectively, idempotent semiring) S, there exists a unique semiring morphism from $\mathbb{N}$ (respectively, $\mathbb{B}$) to S.

2.2.2 Modules

The next step will be the move from semirings to *modules* which can be regarded as semiring-valued spaces in the same way as vector spaces can be regarded as field-valued spaces for fields F.

Given a semiring S, a *left S-module*[1] or a left module over S, consists of a commutative monoid $(M, +, 0)$ and an operation $\cdot : S \times M \to M$, such that for all $r, s \in S$ and $m, n \in M$

$$\begin{aligned} r(m+n) &= rm + rn \\ (r+s)m &= rm + sm \\ (rs)m &= r(sm) \\ 1m &= m. \end{aligned}$$

Analogous to this definition, a *right S-module* consists of a commutative monoid $(M, +, 0)$ and an operation $\cdot : M \times S \to M$, such that for all $r, s \in S$ and $m, n \in M$

$$\begin{aligned} (m+n)r &= mr + nr \\ m(r+s) &= mr + ms \\ m(rs) &= (mr)s \\ m1 &= m. \end{aligned}$$

When S is a commutative semiring, the notions of left and right S-modules are equivalent, and are simply called *S-modules*.

Given two S-modules M and N, we call a function $f : M \to N$ an S-module morphism if and only if f is linear, i.e. if $f(0_M) = 0_N$, $f(m_1 + m_2) = f(m_1) + f(m_2)$ for all $m_1, m_2 \in M$, and if $f(sm) = sf(m)$ for all $s \in S$ and $m \in M$.

As an importance instance of an S-module, given a set X, which can be regarded as a set of *variables*, and a semiring S, consider the set of finite linear combinations

$$S_\omega^X = \{f : X \to S \,|\, \{x \in X \,|f(x) \neq 0\} \text{ is finite}\}$$

of elements of X, with coefficients in S. We will, for the sake of consistency, write (f, x) for $f(x)$ for some $f \in S_\omega^X$ and $x \in X$. Defining zero and sum by

$$(0, x) = 0_S$$

[1] We will not use the term *semimodule*. When needed, we can always distinguish between modules over a ring and modules over a semiring.

and

$$(s+t,x) = (s,x) + (t,x),$$

and scalar multiplication by

$$(rs,x) = r \cdot (s,x)$$

it is clear that this structure is a S-module: moreover, it is the *free* S-module over X, which plays an important role in the theory of rational power series. Also note that S itself also can be regarded as an S-module, isomorphic to $S_\omega^{\{1\}}$.

We can represent elements of S_ω^X using (formal) sums, writing elements $s \in S_\omega^X$ as

$$\sum_{x \in X} x(s,x).$$

It is easy to see that this summation operation is compatible with the addition operator $+$ defined on the module S_ω^X.

We can extend functions $f : X \to Y$ to functions $S_\omega^f : S_\omega^X \to S_\omega^Y$ by defining, for arbitrary $s \in S_\omega^X$

$$S_\omega^f(s) = S_\omega^f\left(\sum_{x \in X} x(s,x)\right) = \sum_{x \in X} f(x)(s,x) = \sum_{y \in Y} y\left(\sum_{x \in X : f(x) = y} (s,x)\right).$$

Note that we can only be certain that the last sum is properly defined because we know there can only be finitely many $x \in X$ such that $(s,x) \neq 0$. This last step, however, can easily be understood when we regard it as a simple regrouping of the term: the key idea is to transform finite S-linear combinations of elements of X into finite S-linear combinations of elements to Y by simply applying f to the occurrences of elements of X in the linear sum, and then regrouping the resulting expression as a summation over elements of y. It is easy to see that the equalities $S_\omega^{1_X} = 1_{S_\omega^X}$ and $S_\omega^{g \circ f} = S_\omega^g \circ S_\omega^f$ hold, which is equivalent to the categorical fact that the operation S_ω^- can be seen as a (covariant!) functor.[2]

Given a S-module M, we moreover always have a unique morphism $\alpha_M : S_\omega^M \to M$, satisfying, for all $m \in M$, $\alpha_M \circ \eta_M(m) = m$ (here $\eta_M : M \to S_\omega^M$ is the function mapping each $m \in M$ to $\lambda n(\textbf{if } n = m \textbf{ then } 1 \textbf{ else } 0)$), $\alpha_M(0_{S_\omega^M}) = 0_M$, $\alpha_M(n_1 + n_2) = \alpha_M(n_1) + \alpha_M(n_2)$, and $\alpha_M(sn) = s\alpha_M(n)$. This morphism α_M, together with these concrete conditions can be seen as an instance of the more general and abstract categorical notion of an *algebra for a monad* T. Another way to understand this morphism, is to see it as a witness of a fact that, for any S-module M, linear combinations of elements of M (that is, elements of S_ω^M) can again be seen as elements of M.

As an important example, let us consider the case where S is the Boolean semiring $\mathbb{B}$. Here S_ω^X is equivalent to $\mathcal{P}_\omega(X)$, the set of finite subsets of X; and, given a $f : X \to Y$, the function S_ω^f will correspond to $\mathcal{P}_(f)$, defined by $\mathcal{P}_\omega(f)(Z) = \{f(z) \,|\, z \in Z\}$ for finite $Z \subseteq X$.

2.2.3 Formal power series

In a similar manner, we can assign a $\mathbb{B}$-module structure to the set $\mathcal{P}(A^*)$ of languages over an alphabet A. However, $\mathcal{P}(A^*)$ can also be assigned a semiring structure: given languages $L, M \in \mathcal{P}(A^*)$, by defining the multiplication as $LM := \{vw \,|\, v \in L, w \in M\}$. In this section, we will introduce the notion of a formal power series over a set of noncommuting variables, and see that there is a much more general class of structures satisfying this property of being a semiring as well as a semimodule (in the literature often called a *S-algebra*).

Formal power series, over commuting variables, were originally introduced as a generalization of classical power series, abstracting away from the traditional interpretation as Taylor series of a function. Their noncommuting variants turn out to be of major importance in the theory of formal

[2] This in contrast with the ordinary exponentiation functor S^X, which is *contravariant*.

languages. For a more comprehensive treatment of formal power series over noncommuting variables, see e.g. [BR11], or, for a coalgebraic view, [Rut03].

In Section 2.1.2, we already introduced the notation $S\langle\langle A\rangle\rangle$ for the function space from A^* to S. From now on, we will assume S to be a semiring, and call elements of $S\langle\langle A\rangle\rangle$ *formal power series* with coefficients in S and (noncommuting) variables A, and use this semiring structure of S to define an embedding

$$i : S \to S\langle\langle A\rangle\rangle$$

and operations

$$+ : S\langle\langle A\rangle\rangle \times S\langle\langle A\rangle\rangle \to S\langle\langle A\rangle\rangle$$

and

$$\cdot : S\langle\langle A\rangle\rangle \times S\langle\langle A\rangle\rangle \to S\langle\langle A\rangle\rangle$$

by setting, for all $w \in A^*$, $s \in S$, and $\sigma, \tau \in S\langle\langle A\rangle\rangle$:

$$(i(s), w) = (\textbf{if } w = 1 \textbf{ then } s \textbf{ else } 0),$$
$$(\sigma + \tau, w) = (\sigma, w) + (\tau, w), \text{ and}$$
$$(\sigma\tau, w) = \sum_{uv=w} (\sigma, u)(\tau, v).$$

It is now easy to see that the structure

$$(S\langle\langle A\rangle\rangle, \cdot, +, i(1_S), i(0_S))$$

is indeed a semiring. Combining this semiring structure on $S\langle\langle A\rangle\rangle$ with the automaton structure from Section 2.1.2, we now obtain the following result:

Proposition 6. *For all $\sigma, \tau \in S\langle\langle A\rangle\rangle$, we have*

$$o(\sigma\tau) = o(\sigma)o(\tau)$$

and for all $\sigma, \tau \in S\langle\langle A\rangle\rangle$ and $a \in A$,

$$(\sigma\tau)_a = \sigma_a\tau + o(\sigma)\tau_a.$$

Proof. First we have

$$o(\sigma\tau) = (\sigma\tau, 1) = (\sigma, 1)(\tau, 1) = o(\sigma)o(\tau)$$

and then, for any $w \in A^*$

$$\begin{aligned}
(\sigma\tau_a, w) &= (\sigma\tau, aw) \\
&= \sum_{u'v'=aw} (\sigma, u')(\tau, v') \\
&= \sum_{uv=w} (\sigma, au)(\tau, v) + (\sigma, 1)(\tau, aw) \\
&= \sum_{uv=w} (\sigma_a, u)(\tau, v) + (\sigma, 1)(\tau_a, w) \\
&= (\sigma_a\tau, w) + o(\sigma)(\tau_a, w) \\
&= (\sigma_a\tau + o(\sigma)\tau_a, w).
\end{aligned}$$

Note that, in the last step, we have tacitly identified $o(\sigma)$ with its value under the embedding i_S. □

Given a formal power series σ, we say its *support* is the set of words $w \in A^*$ such that $(\sigma, w) \neq 0$. We call a formal power series σ *polynomial* whenever its support is finite, or in other words, there are only finitely many words $w \in A^*$ such that $(\sigma, w) \neq 0$. The set of all polynomial power series with coefficients in S and variables in A is denoted by $S\langle A\rangle$. It is easy to see that $S\langle A\rangle$ is closed under the operations $\cdot$ and $+$, or, in other words, if σ and τ have finite support, then so do $\sigma \cdot \tau$ and $\sigma + \tau$. As a result, $S\langle A\rangle$ has a semiring structure in addition to its S-module structure.

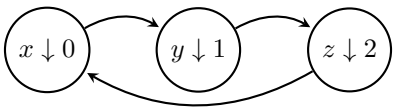
2.2.4 Streams

A special instance of formal power series arises when we restrict ourselves to a single alphabet symbol (which we will, by convention, denote with the symbol $\mathcal{X}$). The set of power series with output in a semiring S, is then equivalent to the set $S^{\mathbb{N}}$ of functions from $\mathbb{N}$ to S, or as we will commonly call them, *streams* over S, identifying $(\sigma, \mathcal{X}^n)$ with (σ, n) under this new representation.

Furthermore, in this case where we have just a single alphabet symbol $\mathcal{X}$, we simply write σ' and $\delta(\sigma)$ instead of $\sigma_{\mathcal{X}}$ and $\delta(\sigma)(a)$. We also use the familiar notation of writing $\sigma^{(n)}$ for the nth derivative of σ, that is, $\sigma_{\mathcal{X}^n}$. We can summarize the relationship between our notation for the specific case of streams, and the notation for the more general case of formal power series, in the following diagram:

	Formal power series (in noncommuting variables)	Streams
Alphabet	A	$\{\mathcal{X}\}$
Solution space	$S\langle\langle A\rangle\rangle$	$S^{\mathbb{N}} = S\langle\langle\{\mathcal{X}\}\rangle\rangle$
Derivative of x	x_a $(a \in A)$	x'

Example 7. Consider the automaton



over a single alphabet symbol ($\mathcal{X}$, not shown in the drawing of the automaton), and with outputs in $\mathbb{N}$. This automaton corresponds to the system of behavioural differential equations

$$\begin{aligned} o(x) &= 0 \quad x' = y \\ o(y) &= 1 \quad y' = z \\ o(z) &= 2 \quad z' = x \end{aligned}$$

and it is easily seen that the final homomorphism maps x onto the stream

$$[\![x]\!] = 0, 1, 2, 0, 1, 2, 0, 1, 2, \ldots$$

For a comprehensive treatment of streams and the coinductive stream calculus, we refer to [Rut03] and [Rut05].

3 Rational systems

In this section, we will present and recall some results about *weighted automata*, which will be needed in the remainder of this article. The class of languages and power series characterized by these automata is called *rational*, and, whenever the underlying semiring S is a field, this notion of rationality coincides with the classical notion. For a more comprehensive background on rational systems, power series, and weighted automata, see for example [BR11] or [Sak09] for a classical account, or [Rut08] or [BBB$^+$12] for a coalgebraic approach.

Fixing a finite alphabet A, a weighted automaton with weights in a semiring S consists of a triple (X, o, δ), where

1. X is a finite set, to be regarded as a set of *variables* or *nonterminals*;
2. $o : X \to S$ is, as in the case of automata, an output function

3. $\delta : X \to (S_\omega^X)^A$ is the transition function, describing each of the possible derivatives of the set X as a S-weighted linear combination of elements of X.

Closely related to weighted automata are *linear automata*: we call an S-automaton (Q, o, δ) *linear* whenever

1. Q is an S-module; and
2. $o : Q \to S$ and $\delta : Q \to Q^A$ are S-module morphisms.

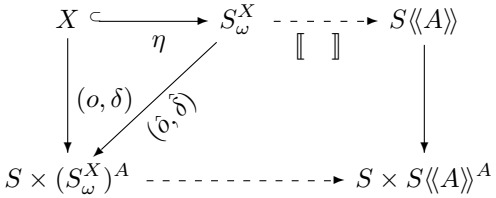
Using the method of determinization ([RS59], [SBBR10]), we can transform each weighted automaton (X, o, δ) into a linear automaton $(S_\omega^X, \hat{o}, \hat{\delta})$. To ensure compatibility between o and δ on one side, and $\hat{o}$ and $\hat{\delta}$ on the other side, we must have $\hat{o}(\eta(x)) = o(x)$ for all $x \in X$, and $\hat{\delta}(\eta(x))(a) = \delta(x)(a)$ for all $x \in X$ and $a \in A$. Moreover, $\hat{o}$ and $\hat{\delta}$ have to be linear mappings as well. Because S_ω^X is the free S-module over X, however, we know that there must be a unique mapping satisfying these properties: given any $\sigma \in S_\omega^X$, we have

$$\hat{o}(\sigma) = \hat{o}\left(\sum_{x \in X} (\sigma, x)x\right) = \sum_{x \in X} (\sigma, x)o(x)$$

and for all $\sigma \in S_\omega^X$ and $a \in A$, we have

$$\sigma_a = \left(\sum_{x \in X} (\sigma, x)x\right)_a = \sum_{x \in X} (\sigma, x)x_a.$$

This construction, which can be seen as an instance of a more general categorical framework presented in [SBBR10], can be summarized in the following diagram, together with the final homomorphism from the linear automaton S_ω^X into the final automaton:



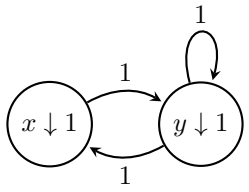
We will call a formal power series σ *rational* whenever there is a weighted automaton X, and an $x \in X$, such that $[\![\eta(x)]\!] = \sigma$.

Because $\hat{o}$ and $\hat{\delta}$ are compatible with o and δ, we will from now on simply use the symbols o and δ to refer to both functions.

Example 8. Consider the following system over the semiring $\mathbb{N}$, with a set $X = \{x, y\}$ of two variables, and a single alphabet symbol $\mathfrak{X}$:

$$\begin{array}{ll} o(x) = 1 & x' = y \\ o(y) = 1 & y' = x + y \end{array}$$

This system corresponds to the weighted automaton[3]



[3]The 1s labelling the arrows here are the weights of the transitions, rather than alphabet symbols.

and we now obtain

$$[\![\eta_X(x)]\!] = 1, 1, 2, 3, 5, 8, 13, 21, 34, \ldots,$$

or in other words, x is mapped by the final homomorphism to the stream of Fibonacci numbers, easily establishing that this sequence is a $\mathbb{N}$-rational stream.

3.1 Bisimulations up to linear combinations

In the case of linear automata, we can establish equality under the final homomorphism using the notion of a bisimulation up to linear combinations. The general idea here is that, given two linear automata X and Y, whenever two elements (s, t) are related by a bisimulation up to linear combinations $R \subseteq X \times Y$, their outputs are identical, and moreover, the derivatives $s_a = t_a$ are linear combinations of projections of related pairs again. The idea of bisimulations up to linear combinations is closely related to the more abstract, categorically defined, notion of coalgebraic bisimulation up-to: for more background on this topic, we refer to [RBR13].

In order to make this notion precise, we will first need to define the relation ΣR, which can, in a way, be regarded as a linear extension of R. We can formally specify ΣR as

$$\Sigma R := \{(\alpha_X \circ S_\omega^{\pi_1}, \alpha_Y \circ S_\omega^{\pi_2})(r) \mid r \in S_\omega^R\};$$

S_ω^R here is simply the set of S-linear combinations of elements of R; however, because X and Y themselves have a S-module structure, we can canonically transform elements of S_ω^R into elements of $X \times Y$: this is done by the function

$$(\alpha_X \circ S_\omega^{\pi_1}, \alpha_Y \circ S_\omega^{\pi_2}).$$

Here mappings $\alpha_X : S_\omega^X \to X$ and $\alpha_Y : S_\omega^Y \to Y$ are as defined in Section 2.2.2; and $S_\omega^{\pi_1}$ and $S_\omega^{\pi_2}$ are the liftings of the projection morphisms π_1 and π_2 over the functor S_ω^-, going from S_ω^R to S_ω^X and S_ω^Y, respectively. As a result, ΣR is again a subset of $X \times Y$.

Given linear automata X and Y, we will now call a relation $R \subseteq X \times Y$ a *bisimulation up to linear combinations* whenever the following conditions are satisfied:

1. for all $(x, y) \in R$, $o(x) \in o(y)$, and
2. for all $(x, y) \in R$ and $a \in A$, $x_a \; \Sigma R \; y_a$.

We will next show that a relation R is a bisimulation up to linear combinations whenever ΣR is a bisimulation. In order to show this, hovever, we first to prove a few auxiliary results.

Lemma 9. *Given any $(p, q) \in X \times Y$, we have $(p, q) \in \Sigma R$ iff there exists an index set I, and three mappings assigning elements $i \in I$ to coefficients $s_i \in S$ and elements $p_i \in X$ and $q_i \in Y$, such that*

$$p = \sum_{i \in I} s_i p_i \qquad \text{and} \qquad q = \sum_{i \in I} s_i q_i,$$

and for each $i \in I$, $(p_i, q_i) \in R$.

Proof. Assume $(p, q) \in \Sigma R$. Then, by the definition of ΣR, there has to be a $r \in S_\omega^R$, such that

$$p = \alpha_X \circ S_\omega^{\pi_1}(r) \qquad \text{and} \qquad q = \alpha_Y \circ S_\omega^{\pi_2}(r).$$

However, because of the definition of S_ω^-, it is clear that there must be some index set I and mappings assigning elements $i \in I$ to coefficients $s_i \in S$ and elements $r_i \in R$, such that

$$r = \sum_{i \in I} s_i r_i.$$

Writing each r_i as (p_i, q_i), we now obtain

$$p = \alpha_X \circ S^{\pi_1}_\omega \left(\sum_{i \in I} s_i (p_i, q_i) \right) = \alpha_X \left(\sum_{i \in I} s_i p_i \right) = \sum_{i \in I} s_i p_i$$

and similarly for q. The vanishing of the α here can be understood as representing a move from elements of S^X_ω, represented as formal sums, to the corresponding real sums in X. For the other direction, we can simply define

$$r = \sum_{i \in I} s_i (p_i, q_i),$$

observe that $r \in S^R_\omega$, and verify that $p = \alpha_X \circ S^{\pi_1}_\omega(r)$ and $q = \alpha_Y \circ S^{\pi_2}_\omega(r)$. □

From this lemma we directly obtain the result $R \subseteq \Sigma R$:

Corollary 10. *For all $R \subseteq X \times Y$, $R \subseteq \Sigma R$.*

Proof. Given a $(p, q) \in R$, consider the singleton index set $I = \{1\}$, together with the mappings $s_1 = 1$, $p_1 = p$, and $q_1 = q$, and apply Lemma 9 to obtain $(p, q) \in \Sigma R$. □

We now are equipped with the prerequisites needed to establish the desired equivalence:

Proposition 11. *A relation $R \subseteq X \times Y$ is a bisimulation up to linear combinations if and only if ΣR is a bisimulation.*

Proof. First assume that ΣR is a bisimulation. If $(p, q) \in R$, then $(p, q) \in \Sigma R$, and hence we obtain both $o(p) = o(q)$ and $p_a \, \Sigma R \, q_a$ directly.

For the other direction, assume that $R \subseteq X \times Y$ is a bisimulation up to linear combinations. Now, take any $(p, q) \in \Sigma R$. By Lemma 9, we have an index set I and mappings assigning s_i, p_i and q_i to each $i \in I$, such that

$$p = \sum_{i \in I} s_i p_i \qquad \text{and} \qquad q = \sum_{i \in I} s_i q_i.$$

and for each $i \in I$, $(p_i, q_i) \in R$. We now have

$$o(p) = o\left(\sum_{i \in I} s_i p_i \right) = \sum_{i \in I} s_i o(p_i) = \sum_{i \in I} s_i o(q_i) = o\left(\sum_{i \in I} s_i q_i \right) = o(q)$$

and

$$p_a = \left(\sum_{i \in I} s_i p_i \right)_a = \sum_{i \in I} s_i (p_i)_a \quad \Sigma R \quad \sum_{i \in I} s_i (q_i)_a = \left(\sum_{i \in I} s_i q_i \right)_a = q_a,$$

and the proof is complete. □

We will conclude this section with an elementary but important result: just like in the case of ordinary bisimulation, elements related by a bisimulation up to linear combinations have the same semantics in the final automaton.

Proposition 12. *If $R \subseteq X \times Y$ is a bisimulation up to linear combinations, and $(p, q) \in R$, then $[\![p]\!] = [\![q]\!]$.*

Proof. Use the fact that $R \subseteq \Sigma R$, the fact that ΣR is a bisimulation, and Proposition 5. □

4 Context-free systems

In this section, we will present a format for systems of behavioural differential equations, which turns out to characterize precisely the context-free languages when $\mathbb{B}$ is chosen as underlying semiring. This format is an adaptation and extension of the formats presented in [WBR11] and [BRW12]. As we will show later in this article, the formal power series characterizable in this way correspond exactly to various notions of *algebraic* power series.

Fixing an alphabet A, a polynomial, or context-free, system of behavioural differential equations over a semiring S consists of a triple (X, o, δ), where

1. X is a finite set, to be regarded as a set of *variables* or *nonterminals*;
2. $o : X \to S$ is, as in the case of automata, an output function; and
3. $\delta : X \to S\langle X\rangle^A$ is the transition function, describing each of the possible derivatives of the set X as a polynomial.

We can, also, regard these systems of equations as coalgebras of the type

$$X \overset{(o,\delta)}{\to} S \times S\langle X\rangle^A.$$

In the introduction, we already saw an example of a context-free system of equations. In order to be able to give meaning to these systems of behavioural differential equations, we first require a method of transforming such a system into an automaton[4] (or, equivalently, a coalgebra for the functor $S \times -^A$). We will do this using a method related to, but more complicated than, the determinization method presented in Section 3.

To start, we extend (o, δ) into a mapping

$$(\bar{o}, \bar{\delta}) : X^* \to S \times S\langle X\rangle^A$$

specifying output values and derivatives of *words* over X, by means of the inductive definition

$$\begin{array}{ll} \bar{o}(1) = 1 & 1_a = 0 \\ \bar{o}(xw) = o(x)\bar{o}(w) & (xw)_a = x_a w + o(x)w_a \end{array}$$

for all $x \in X$, $w \in X^*$, and $a \in A$.

We can see this inductive definition as an instance of a *product rule*, relating to Brzozowski derivatives in a similar manner as the familiar (Leibniz) product rule relates to ordinary function derivatives. We can now prove that product rule can easily be extended from products of an alphabet symbol and a word, to products of arbitrary words:

Proposition 13. *For all $v, w \in X^*$, the equations*

$$\bar{o}(vw) = \bar{o}(v)\bar{o}(w) \qquad \text{and} \qquad (vw)_a = v_a w + \bar{o}(v)w_a$$

hold in any system defined as above.

Proof. Induction on the length of v.

If $v = 1$, then

$$\bar{o}(vw) = \bar{o}(1w) = \bar{o}(w) = \bar{o}(1)\bar{o}(w)$$

and

$$(vw)_a = (1w)_a = w_a = 0w + 1w_a = v_a w + \bar{o}(v)w_a.$$

[4] In most cases, an infinite automaton

If $v = xu$ for $x \in X$ and $u \in X^*$, use the inductive hypothesis that

$$\bar{o}(uw) = \bar{o}(u)\bar{o}(w) \qquad \text{and} \qquad (uw)_a = u_a w + \bar{o}(u)w_a$$

and now observe

$$\bar{o}(vw) = \bar{o}(xuw) = o(x)\bar{o}(uw) = o(x)\bar{o}(u)\bar{o}(w) = \bar{o}(xu)\bar{o}(w) = \bar{o}(v)\bar{o}(w)$$

and

$$\begin{aligned}
(vw)_a &= (xuw)_a \\
&= x_a(uw) + o(x)(uw)_a \\
&= x_a uw + o(x)(u_a w + \bar{o}(u)w_a) \\
&= x_a uw + o(x)u_a w + o(x)\bar{o}(u)w_a \\
&= (xu)_a w + \bar{o}(xu)w_a \\
&= v_a w + \bar{o}(v)w_a,
\end{aligned}$$

completing the proof. □

Now, because $S\langle X\rangle \simeq S_\omega^{X^*}$ (or, in other words: because polynomials over X are the same thing as finite linear combinations of words over X), the inductive extension presented above simply gives a nondeterministic system

$$(\bar{o}, \bar{\delta}) : X^* \to (S_\omega^{X^*})^A.$$

As a result, we can at this stage simply apply the determinization method from Section 3, obtaining a deterministic automaton $(\hat{o}, \hat{\delta})$. This automaton again satisfies (a more general version of) the product rule, this time defined on polynomials:

Proposition 14. *For any polynomials $s, t \in S\langle X\rangle$ and any $a \in A$, the equations*

$$\hat{o}(st) = \hat{o}(s)\hat{o}(t) \qquad \textit{and} \qquad (st)_a = s_a t + \hat{o}(s)t_a$$

hold in any system $(S\langle X\rangle, \hat{o}, \hat{\delta})$ as defined above.

Proof. First note that we have

$$s = \sum_{w \in A^*} (s, w)w \qquad \text{and} \qquad t = \sum_{w \in A^*} (t, w)w.$$

Now observe

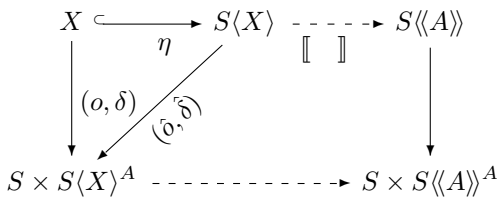
$$\begin{aligned}
\hat{o}(st) &= \hat{o}\left(\sum_{v \in X^*} (s, v)v \sum_{w \in X^*} (t, w)w\right) \\
&= \hat{o}\left(\sum_{v,w \in X^*} (s, v)(t, w)vw\right) \\
&= \sum_{v,w \in X^*} (s, v)(t, w)o(vw) \\
&= \sum_{v,w \in X^*} (s, v)(t, w)o(v)o(w) \\
&= \sum_{v \in X^*} (s, v)o(v) \sum_{w \in X^*} (t, w)o(w) \\
&= \hat{o}(s)\hat{o}(t)
\end{aligned}$$

and

$$\begin{aligned}
(st)_a &= \left(\sum_{v\in X^*}(s,v)v \sum_{w\in X^*}(t,w)w\right)_a \\
&= \sum_{v,w\in X^*}(s,v)(t,w)(vw)_a \\
&= \sum_{v,w\in X^*}(s,v)(t,w)(v_a w + o(v)w_a) \\
&= \sum_{v,w\in X^*}(s,v)(t,w)v_a w + \sum_{w\in X^*}(s,v)(t,w)o(v)w_a \\
&= \sum_{v\in X^*}(s,v)v_a \sum_{w\in X^*}(t,w)w + \sum_{v\in X^*}(s,v)o(s)\sum_{w\in X^*}(t,w)w_a \\
&= \sum_{v\in X^*}(s_a,v)v \sum_{w\in X^*}(t,w)w + \sum_{v\in X^*}(s,v)o(s)\sum_{w\in X^*}(t_a,w)w \\
&= s_a t + \hat{o}(s)t_a,
\end{aligned}$$

and the proof is complete. □

Now we can combine any system of equations of the form $(X,o,\delta): X \to S\times S\langle X\rangle^A$, its extension $(X,\hat{o},\hat{\delta})$, and the mapping $[\![\ \ ]\!]$ into the final coalgebra $S\langle\!\langle A\rangle\!\rangle$, in the following diagram:



We will henceforth, given a context-free system of behavioural differential equations, call the composition of η (which is the unit of the monad $S\langle -\rangle$) and the final homomorphism $[\![-]\!]$ the *solution* to this system.

Moreover, this final homomorphism preserves products:

Proposition 15. *For any $s,t \in S\langle X\rangle$, we have $[\![s]\!][\![t]\!] = [\![st]\!]$.*

Proof. Consider the relation

$$R = \{(st, [\![s]\!][\![t]\!]) \mid s,t \in S\langle X\rangle\}.$$

R is a bisimulation up to linear combinations between $S\langle X\rangle$ and $S\langle\!\langle A\rangle\!\rangle$, because

$$o(st) = o(s)o(t) = o([\![s]\!])o([\![t]\!]) = o([\![s]\!][\![t]\!])$$

and if $(st, [\![s]\!][\![t]\!]) \in R$, we get

$$\begin{aligned}
(st)_a &= s_a t + o(s)t_a \\
\Sigma R &\quad [\![s_a]\!][\![t]\!] + [\![o(s)]\!][\![t_a]\!] \\
&= [\![s]\!]_a[\![t]\!] + o([\![s]\!])[\![t]\!]_a \\
&= ([\![s]\!][\![t]\!])_a,
\end{aligned}$$

and hence $[\![st]\!] = [\![[\![s]\!][\![t]\!]]\!] = [\![s]\!][\![t]\!]$. □

We will call a formal power series σ over a semiring S and an alphabet A *context-free* whenever there is a context-free system of behavioural differential equations (X, o, δ), and an $x \in X$, such that $[\![\eta_X(x)]\!] = \sigma$.

An equivalent characterization of context-free power series is the following:

Lemma 16. *A formal power series σ is context-free if and only if there is a context-free system of behavioural differential equations, and a polynomial $s \in S\langle X\rangle$, such that $[\![s]\!] = \sigma$.*

Proof. If σ is context-free and (X, o, δ) is a system of behavioural differential equations with $x \in X$ such that $[\![\eta_X(x)]\!] = \sigma$, then the polynomial $\eta_X(x)$ satisfies the above condition.

Conversely, assume that $[\![s]\!] = \sigma$ for some polynomial s. Now construct a new context-free system over the set $\bar{X} = \{\bar{x} \mid x \in X\} \cup \{\bar{s}\}$, and consider the mapping $f : X \to \bar{X}$ defined by $f(x) = \bar{x}$ for all $x \in X$, which extends to a mapping $S\langle f\rangle : S\langle X\rangle \to S\langle \bar{X}\rangle$, and define, for all $\bar{x}$ such that $x \in X$, $o(\bar{x}) = o(x)$, and for all $a \in A$, $\bar{x}_a = f(x_a)$, and furthermore $o(\bar{s}) = o(s)$ and $\bar{s}_a = f(s_a)$. It is now easy to see that the relation

$$R = \{(f(t), [\![t]\!]) \mid t \in S\langle X\rangle\} \cup \{(\bar{s}, \sigma)\}$$

is a bisimulation between $S\langle \bar{X}\rangle$ and $S\langle\!\langle A\rangle\!\rangle$. Hence, $[\![\eta_X(\bar{s})]\!] = \sigma$, so σ is context-free. □

Example 17. Let us return now to the system from the introduction, which was given by the following system of behavioural differential equations:

$$\begin{array}{lll} o(x) = 1 & x_a = xy & x_b = 0 \\ o(y) = 0 & y_a = 0 & y_b = 1 \end{array}$$

We will now make the earlier claim, that x can be interpreted as the language

$$\{a^n b^n \mid n \in \mathbb{N}\},$$

precise. In order to do so, consider the following relation

$$R = \{(xy^k, \{a^n b^{n+k}\, n \in \mathbb{N}\}) \mid k \in \mathbb{N}\} \cup \{(y^k, \{b^k\}) \mid k \in \mathbb{N}\}$$

between $\mathbb{B}\langle X\rangle$ (or, equivalently, $\mathcal{P}_\omega(X)$) and $\mathbb{B}\langle\!\langle A\rangle\!\rangle$ (or, equivalently, $\mathcal{P}(A)$). To see that R is a bisimulation, take an arbitrary $(r_1, r_2) \in R$:

- If (r_1, r_2) is of the form $(xy^k, \{a^n b^{n+k} \mid n \in \mathbb{N}\})$, then either $k = 0$, giving

  $$o(r_1) = o(xy^k) = o(x) = 1 = o(\{a^n b^n \mid n \in \mathbb{N}\}) = o(r_2),$$

  or $k > 0$, giving

  $$o(r_1) = o(xy^k) = 0 = o(\{a^n b^{n+k} \mid n \in \mathbb{N}\}) = o(r_2).$$

  Furthermore,

  $$(r_1)_a = (xy^k)_a = xy^{k+1}\ R\ \{a^n b^{n+k+1} \mid n \in \mathbb{N}\} = \{a^n b^{n+k} \mid n \in \mathbb{N}\}_a = (r_2)_a,$$

  and

  $$(r_1)_b = (xy^k)_b = y^{k-1}\ R\ \{b^{k-1}\} = \{a^n b^{n+k} \mid n \in \mathbb{N}\}_b = (r_2)_b,$$

  completing the case.

- If (r_1, r_2) is of the form $(y^k, \{b^k\})$, then

$$o(r_1) = \textbf{if } k = 0 \textbf{ then } 1 \textbf{ else } 0 = o(r_2),$$

$$(r_1)_a = 0 = (r_2)_1,$$

and

$$(r_1)_b = (y^k)_b = y^{k-1} \; R \; \{b^{k-1}\} = \{b^k\}_b,$$

completing this case, too.

From the fact that R is a bisimulation, it now follows directly that $[\![x]\!] = \{a^n b^n \mid n \in \mathbb{N}\}$, as previously claimed.

Example 18. As an example of a context-free power series that is not a language (over the semiring of the natural numbers and over a singleton alphabet), taken from [Rut02], consider the stream defined by the following equation:

$$o(x) = 1 \qquad x' = x^2$$

Its solution is the stream of Catalan numbers

$$1, 1, 2, 5, 14, 42, 132, 429, 1430, \ldots \,^{5}$$

The nth element of this stream counts the number of well-bracketed words consisting of n pairs of opening and closing brackets. In Section 5.2, we will show how to derive the above equation from a context-free grammar representing pairs of brackets.

Example 19. Another example of a context-free stream, using the finite field $\mathbb{F}_2$ (characterized by $1 + 1 = 0$) as underlying semiring, is defined by the following system of equations (where now $X = \{w, x, y, z\}$):

$$\begin{array}{ll} o(w) = 0 & w' = x \\ o(x) = 1 & x' = x^2 + zy^2 \\ o(y) = 0 & y' = y^2 + zx^2 \\ o(z) = 0 & z' = 1 \end{array}$$

One can show that w is mapped by the final homomorphism onto the stream

$$0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, \ldots \,^{6}$$

which is the so-called Thue-Morse sequence. This stream can also be characterized as the morphic sequence generated by the morphism with $p(0) = 01$ and $p(1) = 10$. In Section 7, we will see a way to derive this sequence directly from a *2-automaton* generating this sequence.

We will finish this section with some basic, but important closure properties of classes of context-free power series.

Proposition 20. *If σ and τ are context-free, then so are $\sigma + \tau$ and $\sigma\tau$.*

Proof. If σ and τ are context-free, then there must be systems of behavioural differential equations (X, o_X, δ_X) and (Y, o_Y, δ_Y) and elements $p \in X$ and $q \in Y$ such that $[\![p]\!] = \sigma$ and $[\![q]\!] = \tau$. Now consider the system $(X + Y, o_{X+Y}, \delta_{X+Y})$, where $X + Y$ is the disjoint union of the two earlier systems. For $x \in X$, we now set $o_{X+Y}(x) = o_X(x)$ and $\delta_{X+Y}(x) = \delta_X(x)$, and similarly for Y.

It is now immediately clear that the relations $\{(x, x) \mid x \in X\}$ and $\{(y, y) \mid y \in Y\}$ are bisimulations between X and $X + Y$, and Y and $X + Y$ respectively. Hence $[\![p]\!] = \sigma$ and $[\![q]\!] = \tau$ also hold with respect to the new system. However, we now directly obtain $[\![p + q]\!] = \sigma + \tau$ because of linearity, and $[\![pq]\!] = \sigma\tau$ because of Proposition 15, so $\sigma + \tau$ and $\sigma\tau$ must be context-free as well. □

[5] This sequence appears in the The On-Line Encyclopedia of Integer Sequences as http://oeis.org/A000108

[6] http://oeis.org/A010060

4.1 Systems of equations and the Greibach normal form

We will now establish a connection between the context-free power series we just defined in terms of behavioural differential equations, and the presentation, common in the theory of weighted automata, of algebraic power series over a semiring S as solutions to certain classes of systems of equations. In the current context, a *system of equations* over an alphabet A will be a pair (X, p), where X is a finite set of nonterminals, and

$$p : X \to S\langle A + X\rangle$$

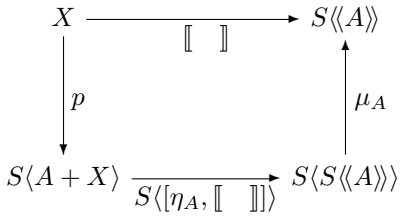
is a mapping assigning a polynomial over the disjoint union $A + X$ to each $x \in X$. This definition corresponds exactly to S-algebraic systems as presented in e.g. [PS09] and originally defined in [Fli74]. Note that we can also regard such systems as weighted grammars: the mapping p, in this case, can be seen as the set of (weighted) production rules of the grammar.

A system of equations is called *proper* iff, for all $x \in X$, $(p(x), 1) = 0$, and for all $x, y \in X$, $(p(x), y) = 0$. If we would regard such systems as weighted grammars, the notion of being proper corresponds to the absence of ε-productions and unit productions in the grammar. Furthermore, such a system is said to be in *Greibach normal form* if its support is contained in the set AX^*.

Our definition of a solution to such systems is, again, equivalent to the classical situation, but presented in a more categorical fashion: to be precise, a *solution* to a system of equations is a mapping

$$[\![\ \ ]\!] : X \to S\langle\langle A\rangle\rangle,$$

such that the diagram



commutes. In this diagram, the arrow labelled $S\langle[\eta_A, [\![\ \ ]\!]]\rangle$ maps polynomials over the coproduct $A + X$ into polynomials over $S\langle\langle A\rangle\rangle$ using the copair operation $[\ ,\ ]$, applying the injection η_A to elements of A, and the mapping $[\![\ \ ]\!]$ to elements of X. These polynomials over $S\langle\langle A\rangle\rangle$, in turn, are also elements of $S\langle\langle S\langle\langle A\rangle\rangle\rangle\rangle$, and now applying the multiplication μ_A for the monad $S\langle\langle -\rangle\rangle$ yields elements of $S\langle\langle A\rangle\rangle$. Alternately, we can see the composition $\mu_A \circ S\langle[\eta_A, [\![\ \ ]\!]]\rangle$ as a lifting of the mapping $[\![\ \ ]\!]$ from elements of X to polynomials over $X + A$ with coefficients in S.

We call such a solution *strong* iff, for all $x \in X$, we have

$$([\![x]\!], 1) = 0,$$

and call a formal power series $\sigma \in S\langle\langle A\rangle\rangle$ *constructively S-algebraic* iff there exists a proper system of equations (X, p), a scalar $s \in S$, a nonterminal $x \in X$, and a strong solution $[\![\ \ ]\!]$, such that

$$\sigma = s + [\![x]\!].$$

It should be noted that, in the case where the underlying semiring is the Boolean semiring $\mathbb{B}$, such systems of equations are equivalent to context-free grammars. Furthermore, we note that, without the additional condition of properness, these systems of equations do not necessarily have solutions: for example, the (non-proper) system consisting of the single equation $x = x + 1$, over the semiring $\mathbb{N}$ does not have any solutions.

We recall the following results from [PS09] without proving them:

Proposition 21. *Every proper S-weighted system of equations has exactly one strong solution [PS09, Theorem 3.2].*

Other than the strong solution, proper systems also may have other solutions. For example, the system $x = x^2$ has (over an arbitrary semiring) solutions $x = 0$ and $x = 1$; but only the first of these solutions is strong. This can be contrasted with context-free systems of behavioural differential equations, which are guaranteed to always have a unique solution.

Proposition 22. *Any component of a strong solution to a proper S-weighted system of equations also occurs as a component of a strong solution to such a system in Greibach normal form [PS09, Theorem 3.15].*

The last proposition now enables us to establish the equivalence between context-free and constructively S-algebraic power series:

Theorem 23. *A formal power series over a semiring S is context-free iff it is constructively S-algebraic.*

Proof. First assume $\sigma \in S\langle\langle A \rangle\rangle$ is constructively S-algebraic. Then, there exists a system of equations in Greibach normal form (X, p), a $x \in X$ and a $s \in S$ such that $\sigma = s + [\![x]\!]$. Now construct a system of behavioural differential equations (X, o, δ) by setting

$$o(x) = 0 \qquad \text{for all } x \in X$$

and

$$(x_a, w) = (p(x), aw) \qquad \text{for all } x \in X,\ a \in A, \text{ and } w \in X^*.$$

We can now verify that $\sigma = s + [\![x]\!]$ again holds with respect to the system of behavioural differential equations. The other direction goes similarly, assuming that we start from a system with $o(x) = 0$ for all $x \in X$. The crucial insight here is that there is a one-to-one correspondence between systems of behavioural differential equations where $o(x) = 0$ for all $x \in X$, and proper systems of equations in Greibach normal form. □

Hence, we can, from now on, regard context-free systems of behavioural differential equations as a defining format for constructively algebraic power series. Compared to the traditional systems of equations, these behavioural differential equations provide the advantage of a more elegant description: no additional conditions are needed to guarantee the existence of unique solutions. Also, they enable us to use bisimulation techniques to reason about context-free languages and context-free systems.

5 Coinductive counting and combinatorics

We will now turn to some applications of the behavioural differential equations, and the proof technique of bisimulation, to a number of combinatorial counting problems. It turns out that a number of familiar sequences, including e.g. the Catalan and Schroeder numbers, can easily be described and understood using systems of behavioural differential equations, characterizing these sequences as context-free streams.

The results in this section can be related to [Rut02], in which a number of counting problems are presented using (both finite and infinite) weighted automata. Compared to the work in that article, we present a more systematic account, giving a uniform technique of obtaining systems of behavioural differential equations directly from a description of the combinatorial structure of a sequence. For example, we can start from the characterization of the Catalan numbers as the number of matching pairs of parentheses of a certain length, and from this characterization directly obtain a system of behavioural differential equations having the Catalan numbers as a solution. Because context-free systems of behavioural differential equations can be regarded as infinite weighted automata, we can regard this method as a more systematic approach, extending the more ad hoc approach from [Rut02].

In order to establish these results, however, we first need to introduce some terminology and notation allowing us to talk about grammars, and derivations of words from such a grammar.

5.1 Grammars and derivations

For now, we will restrict ourselves to the case where the S is the Boolean semiring $\mathbb{B}$. In this case, our context-free series will simply be context-free languages, and the systems of equations correspond to context-free grammars (rather than the weighted grammars we would obtain in the more general case). Moreover, systems of equations in Greibach normal form are in direct correspondence to systems of behavioural differential equations.

We will use the familiar syntax of grammar rules

$$x \to u$$

to denote $(p(x), u) = 1$. In the case of grammars in Greibach normal form, we have $x \to u$ if and only if either

1. $u = 1$ and $o(x) = 1$; or
2. $u = av$ for some $a \in A$ and $v \in X^*$, and $(u_a, v) = 1$

with respect to the system of behavioural differential equations corresponding to the grammar.

We can now define single-step derivations by writing

$$vxw \Rightarrow vuw$$

whenever $x \to u$, for arbitrary $v, w \in (X + A)^*$, and arbitrary derivations by writing

$$v \Rightarrow^* w$$

whenever there is a natural number n, together with a function $f : \{n \in \mathbb{N} \mid n \leq n\} \to (X + A)^*$ such that for all $m \in \mathbb{N}$ with $m < n$, $f(m) \Rightarrow f(m + 1)$, and moreover $f(0) = v$ and $f(n) = w$.

Given $v, w \in (X + A)^*$, we let $\mathbf{derivs}(v, w)$ denote the set of all distinct leftmost derivations $v \Rightarrow^* w$, that is, the set of all such pairs (n, f) witnessing $v \Rightarrow^* w$.

5.2 Counting problems

In this section, we will establish that certain power series and streams, representing the degrees of ambiguity of derivations of context-free grammars in Greibach normal form, are again context-free. This will, again, be done using the technique of bisimulation up to linear combinations. The first proposition will show that, given a context-free grammar in Greibach normal form, presented as a context-free system of differential equations (X, o, δ) over the Boolean semiring $\mathbb{B}$, the power series

$$\sum_{w \in A^*} |\mathbf{derivs}(v, w)|$$

is context-free (over $\mathbb{N}$) for all words $v \in X^*$. The results in this section were originally proven by Chomsky and Schützenberger in [CS63]; we will here provide coinductive proofs of these classical results.

In order to show this, we first transform (X, o, δ) into another system, which is almost identical to the original system, except that the underlying semiring now is $\mathbb{N}$. The underlying set of variables of this new system will be a set of notational variants

$$\bar{X} = \{\bar{x} \mid x \in X\}$$

of the original system. Again, we have a function $f : X \to \bar{X}$, sending each variable to its notational variant, which can be lifted over the star monad yielding $f^* : X^* \to \bar{X}^*$; as well as a function $e : \mathbb{B} \to \mathbb{N}$ defined by $e(0) = 0$ and $e(1) = 1$. We now define a function $g : \mathbb{B}\langle X\rangle \to \mathbb{N}\langle\bar{X}\rangle$ by

$$(g(s), f^*(w)) = e(s, w),$$

which is a proper definition simply because f (and hence, f^* too) is a bijection.

First note we have, for $v \in X^*$, $a \in A$, and $z \in A^*$,

$$|\mathbf{derivs}(v, 1)| = o(v)$$

and

$$|\mathbf{derivs}(v, az)| = \sum_{u \in X^*} e(v_a, u)|\mathbf{derivs}(u, z)|.$$

Proposition 24. *The relation*

$$R = \left\{ \left(\bar{v}, \sum_{w \in A^*} |\mathbf{derivs}(v, w)|w \right) \middle| \, v \in X^* \right\}$$

is a bisimulation up to linear combinations between $\mathbb{N}\langle\bar{X}\rangle$ and $\mathbb{N}\langle\langle A\rangle\rangle$.

Proof. We have

$$o(\bar{v}) = o(v) = |\mathbf{derivs}(v, 1)| = o\left(\sum_{w \in A^*} |\mathbf{derivs}(v, w)|w \right)$$

and, if $\left(\bar{v}, \sum_{w \in A^*} |\mathbf{derivs}(v, w)|w\right) \in R$, then

$$\begin{aligned}
\bar{v}_a &= \sum_{u \in X^*} (\bar{v}_a, \bar{u})\bar{u} \\
&= \sum_{u \in X^*} (v_a, u)\bar{u} \\
\Sigma R \quad &\sum_{u \in X^*} (v_a, u) \sum_{w \in A^*} |\mathbf{derivs}(u, w)|w \\
&= \sum_{z \in A^*} \sum_{u \in X^*} (v_a, u)|\mathbf{derivs}(u, z)|z \\
&= \sum_{z \in A^*} |\mathbf{derivs}(v, az)|z \\
&= \sum_{b \in A} \sum_{z \in A^*} |\mathbf{derivs}(v, bz)|(bz)_a + |\mathbf{derivs}(v, 1)|1_a \\
&= \sum_{w \in A^*} |\mathbf{derivs}(v, w)|w_a \\
&= \left(\sum_{w \in A^*} |\mathbf{derivs}(v, w)|w \right)_a .
\end{aligned}$$

□

We can now also, given a system of behavioural differential equations (X, o, δ) over an alphabet A and some $v \in X^*$, create a context-free stream σ, such that for every number $n \in \mathbb{N}$, $\sigma(n)$ is equal to

$$\sum_{w \in A^*, |w| = n} = |\mathbf{derivs}(v, w)|.$$

In order to do this, we will construct a new system $(\bar{X}, o, \delta)$ over the set

$$\bar{X} = \{\bar{x} \,|\, x \in X\};$$

however, first, we define a morphism $f : X \to \bar{X}$ simply by $f(x) = \bar{x}$. This morphism can be lifted over the functor $S\langle -\rangle$, giving a morphism $\hat{f} : S\langle X\rangle \to S\langle \bar{X}\rangle$, and enabling us to specify the new system with the equations

$$o(\bar{x}) = o(x) \qquad \text{and} \qquad \bar{x}' = \sum_{a\in A} \hat{f}(x_a)$$

from which the equality $(\bar{x}', \bar{u}) = \sum_{a\in A}(x_a, u)$ easily follows.

The following proposition establishes that this new system indeed has the intended behaviour:

Proposition 25. *The relation*

$$R = \left\{ \left(\bar{v}, \sum_{w\in A^*} |\mathbf{derivs}(v, w)|\mathcal{X}^{|w|} \right) \,\middle|\, v \in X^* \right\}$$

is a bisimulation up to linear combinations between $\mathbb{N}\langle\bar{X}\rangle$ (as just defined) and $\mathbb{N}^{\mathbb{N}}$.

Proof. We have

$$o(\bar{v}) = o(v) = |\mathbf{derivs}(v, 1)| = o\left(\sum_{w\in A^*} |\mathbf{derivs}(v, w)|\mathcal{X}^{|w|} \right)$$

and, if

$$\left(\bar{v}, \sum_{w\in A^*} |\mathbf{derivs}(v, w)|\mathcal{X}^{|w|} \right) \in R,$$

then also

$$\begin{aligned}
\bar{v}' &= \sum_{u\in X^*} (\bar{v}', \bar{u})\bar{u} \\
&= \sum_{u\in X^*} \sum_{a\in A} (v_a, u)\bar{u} \\
\Sigma R \quad &\sum_{u\in X^*} \sum_{a\in A} (v_a, u) \sum_{w\in A^*} |\mathbf{derivs}(u, w)|\mathcal{X}^{|w|} \\
&= \sum_{a\in A} \sum_{z\in A^*} \sum_{u\in X^*} (v_a, u)|\mathbf{derivs}(u, z)|\mathcal{X}^{|w|} \\
&= \sum_{a\in A} \sum_{z\in A^*} |\mathbf{derivs}(v, az)|\mathcal{X}^{|z|} \\
&= \sum_{w\in A^*} |\mathbf{derivs}(v, w)|\mathcal{X}^{|w|'} \\
&= \left(\sum_{w\in A^*} |\mathbf{derivs}(v, w)|\mathcal{X}^{|w|} \right)'.
\end{aligned}$$

□

This construction now enables us to derive specifications of some well-known number sequences as context-free streams.

Example 26. As a first example, consider the Catalan numbers. It is well-known that the nth Catalan number corresponds to the number of ways to combine n pairs of matching brackets. An unambiguous CFG in (weak) Greibach normal form representing matching pairs of brackets is

$$x \to axbx \,|\, 1$$

which corresponds to the system of equations

$$\begin{array}{llll} o(x) = 1 & x_a = xyx & x_b = 0 \\ o(y) = 0 & y_a = 0 & y_b = 1. \end{array}$$

Using the transformation on which Proposition 25 was based, we now obtain another system of equations

$$\begin{array}{ll} o(\bar{x}) = 1 & \bar{x}' = \bar{x}\bar{y}\bar{x} \\ o(\bar{y}) = 0 & \bar{y}' = 1. \end{array}$$

yielding the stream σ such that for all $n \in \mathbb{N}$, $\sigma(2n)$ is equal to the nth Catalan number, and $\sigma(2n+1) = 0$.

Now, because multiplication of streams over $\mathbb{N}$ is commutative, observe that

$$\bar{x}'' = (\bar{x}\bar{y}\bar{x})' = (\bar{y}\bar{x}^2)' = \bar{y}'\bar{x}^2 + o(\bar{y})(\bar{x}^2)' = \bar{y}'\bar{x}^2 = \bar{x}^2,$$

giving us a new system, over a single variable z, defined by

$$o(z) = 1 \qquad z' = z^2.$$

Because it clearly holds that $[\![z]\!](n) = [\![\bar{x}]\!](2n)$, it follows immediately that the final homomorphism $[\![-]\!]$ maps z onto the stream of Catalan numbers: thus we have established that the Catalan numbers are a context-free stream.

Example 27. Another, closely related, example arises from the following problem: given a $n \times n$ grid, how many different ways are there to go from $(0,0)$ to (n,n) by making steps of the types $(0,1)$, $(1,0)$ and $(1,1)$ that stay below the diagonal? The sequence mapping each $n \in \mathbb{N}$ to the number of such paths from $(0,0)$ to (n,n) is called the sequence of (large) Schroeder numbers, and starts with

$$1, 2, 6, 22, 90, 394, 1806, 8558, 41586, 206098, \ldots\,[7]$$

It is easy to see that the nth Schroeder number corresponds to the number of derivations of words of length $2n$ given by the following system of behavioural differential equations:

$$\begin{array}{llll} o(x) = 1 & x_a = xyx & x_b = 0 & x_c = zx \\ o(y) = 0 & y_a = 0 & y_b = 1 & y_c = 0 \\ o(z) = 0 & z_a = 0 & z_b = 0 & z_c = 1. \end{array}$$

Here a represents a step of the type $(0,1)$, b represents a step of the type $(1,0)$, and cc (cs can only occur in pairs in the language $[\![x]\!]$) represents steps of the type $(1,1)$.

Using the transformation from Proposition 25, we now obtain a new system of equations

$$\begin{array}{ll} o(\bar{x}) = 1 & \bar{x}' = \bar{x}\bar{y}\bar{x} + \bar{z}\bar{x} \\ o(\bar{y}) = 0 & \bar{y}' = 1 \\ o(\bar{z}) = 0 & \bar{z}' = 1 \end{array}$$

and again, by commutativity of multiplication, we now get

$$\bar{x}'' = (\bar{x}\bar{y}\bar{x} + \bar{z}\bar{x})' = (\bar{y}\bar{x}^2)' + (\bar{z}\bar{x})' = \bar{y}'\bar{x}^2 + o(\bar{y})(\bar{x}^2)' + \bar{z}'\bar{x} + o(\bar{z})\bar{x}' = \bar{x}^2 + \bar{x}$$

yielding a new system over a single variable u:

$$o(u) = 1 \qquad u' = u^2 + u$$

which is mapped by $[\![-]\!]$ onto the Schroeder numbers.

[7] http://oeis.org/A006318

6 The zip-, even- and odd-operations

We will now turn to another case, where bisimulation turns out to be a useful proof technique. In particular, we will consider the **zip** operation on streams, where $\mathbf{zip}(\sigma, \tau)$ is the stream that alternately takes an element from σ and an element from τ, and easily see that it preserves context-freeness, or algebraicity. In [NR10], this result was established for cases where the underlying semiring is a field; the main result of this section generalizes the earlier result to arbitrary semirings. Finally, we will generalize the result to a more general zip_k operation, representing a stream alternatingly taking elements from k streams $\sigma_0, \ldots, \sigma_{k-1}$, and see that this operation, too, preserves context-freeness.

Given two streams $\sigma, \tau \in S^{\mathbb{N}}$ over a semiring S, we formally define the new stream $\mathbf{zip}(\sigma, \tau)$ by setting

$$\mathbf{zip}(\sigma, \tau)(2n) = \sigma(n) \qquad \text{and} \qquad \mathbf{zip}(\sigma, \tau)(2n+1) = \tau(n)$$

for all $n \in \mathbb{N}$. Intuitively, we can see $\mathbf{zip}(\sigma, \tau)$ as a stream that alternately takes an element from σ and an element from τ.

Equivalently, we can also define **zip** coinductively with the following system of behavioural differential equations:

$$o(\mathbf{zip}(\sigma, \tau)) = o(\sigma) \qquad \text{and} \qquad (\mathbf{zip}(\sigma, \tau))' = \mathbf{zip}(\tau, \sigma')$$

We will, in the remainder of this section, make use of the properties

$$\sum_{i \in I} s_i \cdot \mathbf{zip}(0, \sigma_i) = \mathbf{zip}\left(0, \sum_{i \in I} s_i \sigma_i\right)$$

and

$$\mathbf{zip}(\sigma, \tau) = \mathbf{zip}(\sigma, 0) + \mathbf{zip}(0, \tau)$$

without proof.

In order to prove that, whenever σ and τ are context-free, the same goes for $\mathbf{zip}(\sigma, \tau)$, we first prove that $\mathbf{zip}(\sigma, 0)$ and $\mathbf{zip}(0, \tau)$ are context-free:

Proposition 28. *If $\sigma \in S^{\mathbb{N}}$ is context-free, then so are $\mathbf{zip}(\sigma, 0)$ and $\mathbf{zip}(0, \sigma)$.*

Proof. If σ is context-free, there must exist a system of polynomial behavioural differential equations (X, o, δ) and a $x \in X$, such that $[\![x]\!] = \sigma$.

We now will construct a new system of polynomial differential equations, over the set

$$\bar{X} = \{\bar{x} \mid x \in X\} \cup \{\mathcal{X}\}$$

where $\mathcal{X}$ is a new variable, representing the stream

$$0, 1, 0, 0, 0, \ldots$$

Now consider the morphism $f : X \to \bar{X}$, defined by $f(x) = \bar{x}$ for all $x \in X$, which easily extends to a morphism on words $\hat{f} : S\langle X \rangle \to S\langle \bar{X} \rangle$.

The new system $(\bar{X}, \hat{o}, \hat{\delta})$ is now defined by

$$\hat{o}(\bar{x}) = o(x), \quad \hat{o}(\mathcal{X}) = 0, \quad \bar{x}' = \mathcal{X} \cdot f(x'), \quad \text{and} \quad \mathcal{X}' = 1.$$

Now consider the relation $R \subseteq S\langle \bar{X} \rangle \times S^{\mathbb{N}}$, defined by

$$R = \{(f(v), \mathbf{zip}([\![v]\!], 0)) \mid v \in X^*\} \cup \{(\mathcal{X} \cdot f(v), \mathbf{zip}(0, [\![v]\!])) \mid v \in X^*\}.$$

In order to show that R is a bisimulation up to linear combinations, we will first establish an auxiliary result, namely that, for all $v \in X^*$,

$$f(v)' = \mathcal{X} f(v').$$

We prove this by induction on the length of v. If $v = 1$, then

$$f(v)' = f(1)' = 1' = 0 = \mathcal{X} \cdot 0 = \mathcal{X} f(0) = \mathcal{X} f(1')$$

and if $v = xw$ for some $x \in X$ and $w \in X^*$, using the inductive hypothesis that $f(w)' = \mathcal{X} f(w')$, we obtain

$$\begin{aligned} f(xw)' &= (\bar{x} f(w))' \\ &= \bar{x}' f(w) + o(\bar{x}) f(w)' \\ &= \bar{x}' f(w) + o(x) f(w)' \\ &= \bar{x}' f(w) + o(x) \mathcal{X} f(w') \\ &= \mathcal{X} f(x') f(w) + o(x) \mathcal{X} f(w') \\ &= \mathcal{X}(f(x'w) + o(x) f(w')) \\ &= \mathcal{X} f(x'w + o(x) w') \\ &= \mathcal{X} f((xw)'). \end{aligned}$$

Note that we have, here, established true equality in the polynomial semiring, rather than some weaker notion of equivalence.

Another auxiliary result that we need, is that for all $v \in X^*$, we have $o(f(v)) = o(v)$. We omit the proof, which is again by induction on the length of v.

We can now show that R is, indeed, a bisimulation up to linear combinations. If $(t, \sigma) \in R$, then either:

1. $t = f(v)$ and $\sigma = \mathbf{zip}(\llbracket v \rrbracket, 0)$ for some $v \in X^*$. We then get

   $$o(t) = o(f(v)) = o(v) = o(\llbracket v \rrbracket) = o(\mathbf{zip}(\llbracket v \rrbracket, 0))$$

   and

   $$\begin{aligned} f(v)' &= \mathcal{X} f(v') \\ &= \mathcal{X} f\left(\sum_{w \in X^*} (v', w) w \right) \\ &= \sum_{w \in X^*} (v', w) \mathcal{X} f(w) \\ \Sigma R \quad & \sum_{w \in X^*} (v', w) \mathbf{zip}(0, \llbracket w \rrbracket) \\ &= \mathbf{zip}\left(0, \sum_{w \in X^*} (v', w) \llbracket w \rrbracket \right) \\ &= \mathbf{zip}(0, \llbracket v' \rrbracket) \\ &= \mathbf{zip}(\llbracket v \rrbracket, 0)' \end{aligned}$$

2. $t = \mathcal{X} \cdot f(v)$ and $\sigma = \mathbf{zip}(0, \llbracket v \rrbracket, 0)$ for some $v \in X^*$. We then get

   $$o(t) = o(\mathcal{X}) \cdot o(f(v)) = 0 \cdot o(f(v)) = 0 = o(\mathbf{zip}(0, \llbracket v \rrbracket))$$

   and

   $$t' = \mathcal{X}' \cdot f(v) + o(\mathcal{X}) f(v)' = f(v) \quad \Sigma R \quad \mathbf{zip}(\llbracket v \rrbracket, 0) = \mathbf{zip}(0, \llbracket v \rrbracket)'.$$

So R is, indeed, a bisimulation up to linear combinations, and it follows that $\mathbf{zip}(\sigma, 0)$ and $\mathbf{zip}(0, \sigma)$ are context-free. □

The desired result can now be obtained directly:

Theorem 29. *If σ and τ are context-free, then so is $\mathbf{zip}(\sigma, \tau)$.*

Proof. By Proposition 28, we obtain that $\mathbf{zip}(\sigma, 0)$ and $\mathbf{zip}(0, \tau)$ are context-free. Now use

$$\mathbf{zip}(\sigma, \tau) = \mathbf{zip}(\sigma, 0) + \mathbf{zip}(0, \tau)$$

and the fact that context-freeness is preserved under finite linear combinations. □

Going in the other direction, given a stream σ, we can define streams $\mathbf{even}(\sigma)$ and $\mathbf{odd}(\sigma)$ consisting of the even and odd elements of σ, respectively, as follows:

$$(\mathbf{even}(\sigma), n) = (\sigma, 2n) \qquad \text{and} \qquad (\mathbf{odd}(\sigma), n) = (\sigma, 2n + 1)$$

We can relate **zip**, **even**, and **odd** by the equation

$$\sigma = \mathbf{zip}(\mathbf{even}(\sigma), \mathbf{odd}(\sigma))$$

which holds for all streams σ.

So far, it remains an open question whether the **even** and **odd** operators, too, preserve context-freeness. The natural approach here would be to transform the given system into a new system, by setting the derivative of the nonterminals in the new system equal to the second derivatives in the old system. For example, we could try to transform the system

$$o(x) = 1 \qquad \text{and} \qquad x' = x^2$$

yielding the Catalan numbers, into the new system

$$o(\bar{x}) = 1 \qquad \text{and} \qquad \bar{x}' = \bar{x}^3 + \bar{x}.$$

However, the latter system yields the stream

$$[\![\bar{x}]\!] = 1, 2, 10, 66, 498, 4066, 34970, \ldots$$ [8]

rather than the even-indexed Catalan numbers, and the suggested construction does not work. This mismatch can be contrasted with the case of Example 18, where the process of taking the second derivative *does* work, because of the occurrence of a variable which can be seen as doing nothing but playing the role of a delay of one step, being defined by $o(y) = 0$ and $y' = 1$.

6.1 Generalizing to $\mathbf{zip}_k$ and $\mathbf{unzip}_{i,k}$

We can generalize the operation **zip** to the operation $\mathbf{zip}_n$, defined for all $n \leq 2$ and streams $\sigma_1, \ldots, \sigma_n$ as follows:

$$\begin{aligned} o(\mathbf{zip}_n(\sigma_1, \ldots, \sigma_n)) &= o(\sigma_1) \\ \mathbf{zip}_n(\sigma_1, \ldots, \sigma_n)' &= \mathbf{zip}_n(\sigma_2, \ldots, \sigma_n, \sigma_1) \end{aligned}$$

Similarly, we can generalize the **even** and **odd** operations to the operation $\mathbf{unzip}_{i,k}$, defined for all $i, k \in \mathbb{N}$ with $i < k$ by

$$\mathbf{unzip}_{i,k}(\sigma)(n) = \sigma(kn + i)$$

[8] http://oeis.org/A027307

yielding $\mathbf{even} = \mathbf{unzip}_{0,2}$ and $\mathbf{odd} = \mathbf{unzip}_{1,2}$.

The $\mathbf{zip}_n$ and $\mathbf{unzip}_{i,n}$ functions are again related by the equation

$$\sigma = \mathbf{zip}_k(\mathbf{unzip}_{1,k}(\sigma), \ldots, \mathbf{unzip}_{k,k}(\sigma)).$$

The following theorem can now be proved in a similar way as Theorem 29:

Theorem 30. *Given streams $\sigma_1, \ldots, \sigma_n$, n-$\mathbf{zip}(\sigma_1, \ldots, \sigma_n)$ is context-free if all σ_i are context-free.*

7 Fields and automatic sequences

In this section, we will be concerned with streams over fields, and relate the earlier notions of constructive algebraicity and context-freeness to the classical notion of objects that are algebraic over a field. In particular, we will be interested in finite fields, and their relation to so-called automatic sequences. The main result from this section is that p-automatic sequences can precisely be characterized by context-free systems of behavioural equations, obtainable via a straightforward and direct construction from the presentation as a p-automaton. We can see this result as an instance of a more general result due to Michel Fliess from [Fli74], establishing the equivalence between classical algebraicity and constructive algebraicity, which we prove in a coalgebraic and much more direct manner than the traditional proof.

7.1 Preliminaries

First we recall that a *ring* is a semiring S where, for every element $s \in S$, there is an element $-s \in S$ such that $s + -s = 0 = -s + s$. A *field* F is a commutative ring such that for every $f \in F$ with $f \neq 0$, there is an element f^{-1} such that $f \cdot f^{-1} = 1 = f^{-1} \cdot f$. For a comprehensive treatment of these notions, we refer to any textbook on algebra, such as [LB99]. Examples of fields include the familiar structures $\mathbb{Q}$, $\mathbb{R}$, and $\mathbb{C}$ of rational, real and complex numbers; an additional example of a ring that is not a field is the structure $\mathbb{Z}$ of integers.

For every field $(F, +, \cdot)$, there is a unique semiring homomorphism h from the semiring $(\mathbb{N}, +, \cdot)$ to F, as a direct result of the fact that $\mathbb{N}$ is the initial semiring. If there are elements $n \in \mathbb{N}$ with $n \neq 0$ such that $h(n) = 0_F$, we say F has *characteristic* p if p is the smallest number with this condition; if there are no such elements, we say that F has characteristic 0. Whenever F has characteristic $p \neq 0$, p is a prime number.

We say a field F is *perfect* when F has either characteristic 0, or when F has characteristic p and, for every $f \in F$, there is a $g \in F$ such that $f = g^p$. (Here we inductively define $g^0 = 1$ and $g^{n+1} = g \cdot g^n$.)

Of special importance in the theory of automatic sequences are finite fields: for each prime number p and each natural $k > 0$, there is exactly one finite field of size p^k, denoted by $\mathbb{F}_{p^k}$; the characteristic of such a field is equal to p.

We will now recall the usual notion of algebraicity over a field, which applies, in our framework, only to streams over fields: an F-stream σ is called *F-algebraic* if and only if there exist polynomial F-streams $p_1, \ldots, p_n$, with at least one p_i not equal to zero, such that $\sum_{i \leq n} p_i \sigma^i = 0$.

7.2 Automatic sequences

Automatic sequences, of which a comprehensive treatment is given in [AS03], can be characterized as streams that can be generated by a class of automata called q-automata. Both the input alphabet and the output of these automata consists of a set of digits $\{0, \ldots, q\}$. Whenever a stream σ is q-automatic for some $q \in \mathbb{N}$, there exists a q-automaton such that the nth element of the stream can be obtained by using the base q representation of the number n as input; the output thus obtained—again a natural number smaller than q—is then the value of $\sigma(n)$. We will, in this article, restrict

ourselves to the case where q is a prime power, i.e. $q = p^k$ for some prime number p and positive natural number k.

Formally, given a $q \in \mathbb{N}$, such that there is a prime number p and a natural number $k > 0$ with $q = p^n$, a *q-automaton* is an automaton with output in the field $\mathbb{F}_q$, and input alphabet

$$A_q := \{\bar{n} \mid n < q\}.$$

We let the input alphabet consist of notational variants, in order to be able to distinguish between the symbol $\bar{1}$ and the unit element 1, which plays both the role of empty word and the role of multiplicative unit element of the underlying field.

We regard the inputs as natural numbers given in a base q representation, and define a class of functions $[-]_q : A_q^* \to \mathbb{N}$ mapping words of digits to the corresponding natural number, by setting:

$$\begin{array}{rcll} [1]_q & = & 0 & \textit{Beware! This is } 1\textit{, i.e. the empty word, not } \bar{1}. \\ [\bar{i} \cdot w]_q & = & q \cdot [w] + i. & \end{array}$$

Note that, in this presentation, the least significant digit occurs first, and hence, we have e.g.

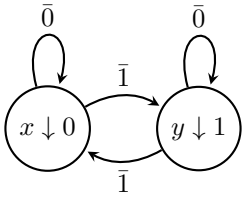
$$[\bar{0}\bar{0}\bar{1}]_2 = 4.$$

We call a stream $\sigma \in \mathbb{F}_q^{\mathbb{N}}$ *q-automatic*, whenever there exists a finite q-automaton (X, o, δ) over the alphabet A_q, together with a state $x \in X$, such that for all $w \in A_q^*$

$$\sigma([w]_q) = o(x_w).$$

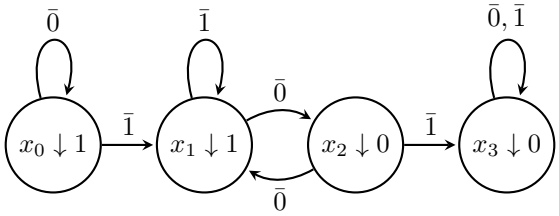
Note that the existence of such a σ is not guaranteed for an arbitrary q-automaton (X, o, δ) and each $x \in X$: we call a state $x \in X$ for which such a σ exists *zero-consistent*, and say that x *generates* σ. It is, however, easy to see that, whenever x is zero-consistent, then so is x_w for all $w \in A_q^*$.

Example 31. As an example of a 2-automatic sequence, consider the 2-automaton:



It is well-known from the literature (see e.g. [AS03]) that x generates the Thue-Morse sequence **tm**, which we already encountered in Example 19.

Example 32. For a second example of a 2-automaton, consider



In this automaton, x_0 generates the so-called Baum-Sweet sequence **bs** (again, see [AS03]), of which the nth element is equal to 1 if and only if the binary representation of n contains no block of consecutive 0s of odd length, and which starts with

$$1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, \ldots$$

7.3 Automatic sequences are context-free

We will start by showing that all 2-automatic sequences are context-free over the field $\mathbb{F}_2$, and after this, we will generalize this to arbitrary q-automatic sequences. First note that, for all streams $\sigma \in \mathbb{F}_2^{\mathbb{N}}$, the equality

$$\mathbf{zip}(\sigma, \tau) = \sigma^2 + \mathcal{X}\tau^2 \tag{2}$$

holds.

Proposition 33. *Given a finite automaton (X, o, δ) over A_2 and a consistent $x \in X$, we have*

$$\mathbf{str}(x_{\bar{0}}) = \mathbf{even}(\mathbf{str}(x))$$

and

$$\mathbf{str}(x_{\bar{1}}) = \mathbf{odd}(\mathbf{str}(x)).$$

Proof. This follows from the equations

$$\begin{aligned} \mathbf{str}(x_{\bar{0}})([w]) &= o((x_{\bar{0}})_w) \\ &= o(x_{\bar{0}\cdot w}) \\ &= \mathbf{str}(x)([\bar{0} \cdot w]) \\ &= \mathbf{str}(x)(2 \cdot [w]) \\ &= \mathbf{even}(\mathbf{str}(x))([w]) \end{aligned}$$

and

$$\begin{aligned} \mathbf{str}(x_{\bar{1}})([w]) &= o((x_{\bar{1}})_w) \\ &= o(x_{\bar{1}\cdot w}) \\ &= \mathbf{str}(x)([\bar{1} \cdot w]) \\ &= \mathbf{str}(x)(2 \cdot [w] + 1) \\ &= \mathbf{odd}(\mathbf{str}(x))([w]) \end{aligned}$$

respectively. □

Proposition 34. *All 2-automatic sequences are context-free over the field $\mathbb{F}_2$.*

Proof. Observe that, as a result of the equality $\mathbf{zip}(\mathbf{even}(\sigma), \mathbf{odd}(\sigma)) = \sigma$ and Proposition 37 we have

$$\mathbf{str}(x) = \mathbf{zip}(\mathbf{str}(x_{\bar{0}}), \mathbf{str}(x_{\bar{1}})),$$

and from there we get

$$\begin{aligned} \mathbf{str}(x)' &= \mathbf{zip}(\mathbf{str}(x_{\bar{0}}), \mathbf{str}(x_{\bar{1}}))' \\ &= \mathbf{zip}(\mathbf{str}(x_{\bar{1}}), \mathbf{str}(x_{\bar{0}})') \\ &= o(x_{\bar{1}}) + \mathcal{X} \cdot \mathbf{zip}(\mathbf{str}(x_{\bar{0}})', \mathbf{str}(x_{\bar{1}})') \\ &= o(x_{\bar{1}}) + \mathcal{X} \cdot (\mathbf{str}(x_{\bar{0}})')^2 + \mathcal{X}^2 \cdot (\mathbf{str}(x_{\bar{1}})')^2. \end{aligned} \tag{3}$$

Now consider the following context-free system of behavioural differential equations, over the set $\bar{X} = \{\bar{x} \,|\, x \in X\} \cup \{\mathcal{X}\}$. Whenever $x_{\bar{0}} = y$ and $x_{\bar{1}} = z$, we set:

$$o(\bar{x}) = o(z) \qquad \text{and} \qquad \bar{x}' = \bar{y}^2 + \mathcal{X}\bar{z}^2$$

and furthermore we set

$$o(\mathcal{X}) = 0 \qquad \text{and} \qquad \mathcal{X}' = 1.$$

We know that this system must have a unique solution. But from (3), it follows that this solution is given by

$$[\![\bar{x}]\!] = \mathbf{str}(x)',$$

for all $x \in X$. So $\mathbf{str}(x)'$ is context-free for any $x \in X$, and it now follows directly that $\mathbf{str}(x)$ is, too, completing the proof. □

Example 35. Returning to the automaton from Example 31, the construction from Proposition 34 now yields the system of behavioural differential equations

$$\begin{array}{cc} o(\bar{x}) = o(y) = 1 & \bar{x}' = \bar{x}^2 + \mathcal{X}\bar{y}^2 \\ o(\bar{y}) = o(x) = 0 & \bar{y}' = \bar{y}^2 + \mathcal{X}\bar{x}^2 \\ o(\mathcal{X}) = 0 & \mathcal{X}' = 1 \end{array}$$

with $[\![\bar{x}]\!] = \mathbf{tm}'$. Adding another variable z to this system with $o(z) = 0$ and $z' = x$, we obtain a new system with $[\![z]\!] = \mathbf{tm}$. Observe that this system is isomorphic to the system presented in Example 19: we have hereby showed that this system indeed generates the Thue Morse-sequence.

Example 36. Likewise, the automaton from Example 32 now yields the system:

$$\begin{array}{cc} o(\bar{x}_0) = 1 & \bar{x}_0' = \bar{x}_0^2 + \mathcal{X}\bar{x}_1^2 \\ o(\bar{x}_1) = 1 & \bar{x}_1' = \bar{x}_2^2 + \mathcal{X}\bar{x}_1^2 \\ o(\bar{x}_2) = 0 & \bar{x}_2' = \bar{x}_1^2 + \mathcal{X}\bar{x}_3^2 \\ o(\bar{x}_3) = 0 & \bar{x}_3' = \bar{x}_3^2 + \mathcal{X}\bar{x}_3^2 \\ o(\mathcal{X}) = 0 & \mathcal{X}' = 1, \end{array}$$

with $[\![\bar{x}_0]\!] = \mathbf{bs}$.

Generalizing from the previous result to q-automatic sequences for prime powers q, we have to make a move from the operations **zip**, **even**, **odd** to the operations $\mathbf{zip}_q$ and the family of operations $\mathbf{unzip}_{i,q}$ for $i \in \mathbb{N}$ with $i < q$.

In the case of streams over $\mathbb{F}_q$, we can generalize (2) to

$$\mathbf{zip}_q(\sigma_0, \ldots \sigma_{q-1}) = \sum_{i \in \mathbb{N}<q} \mathcal{X}^i {\sigma_i}^q.$$

Proposition 37. *Given a finite q-automaton (X, o, δ) and a consistent $x \in X$, we have*

$$\mathbf{str}(x_{\bar{i}}) = \mathbf{unzip}_{i,q}(\mathbf{str}(x)).$$

for all i with $0 \leq i \leq q$.

Proof. This follows from the equation

$$\begin{array}{rcl} \mathbf{str}(x_{\bar{i}})([w]) & = & o((x_{\bar{i}})_w) \\ & = & o(x_{\overline{i \cdot w}}) \\ & = & \mathbf{str}(x)([\bar{i} \cdot w]) \\ & = & \mathbf{str}(x)(k \cdot [w] + i) \\ & = & \mathbf{unzip}_i(\mathbf{str}(x))([w]). \end{array}$$

□

We can now generalize Proposition 34 to the following result:

Proposition 38. *For any prime power q, all q-automatic sequences are context-free over the field $\mathbb{F}_q$.*

Proof. Observe that, as a result of the equality $\sigma = \mathbf{zip}_q(\mathbf{unzip}_{1,q}(\sigma), \ldots, \mathbf{unzip}_{q,q}(\sigma))$ and Proposition 37 we have

$$\mathbf{str}(x) = \mathbf{zip}_q(\mathbf{str}(x_{\bar{1}}), \ldots, \mathbf{str}(x_{\bar{q}})),$$

and from there we get

$$\begin{aligned}
\mathbf{str}(x)' &= \mathbf{zip}_q(\mathbf{str}(x_{\bar{0}}), \ldots, \mathbf{str}(x_{\overline{q-1}}))' \\
&= \mathbf{zip}_q(\mathbf{str}(x_{\bar{1}}), \ldots \mathbf{str}(x_{\overline{q-1}}), \mathbf{str}(x_{\bar{0}})') \\
&= \sum_{1 \leq i < q-1} \mathcal{X}^{i-1} o(x_{\bar{i}}) + \mathcal{X}^q \cdot \mathbf{zip}_q(\mathbf{str}(x_{\bar{0}})', \ldots, \mathbf{str}(x_{\overline{q-1}})') \\
&= \sum_{1 \leq i < q-1} \mathcal{X}^{i-1} o(x_{\bar{i}}) + \mathcal{X}^q \left(\sum_{i<q} \mathcal{X}^i (\mathbf{str}(x_{\bar{i}})')^q \right) \\
&= o(x_{\bar{1}}) + \mathcal{X} \left(\sum_{2 \leq i < q-1} \mathcal{X}^{i-2} o(x_{\bar{i}}) + \sum_{i<q} \mathcal{X}^{q+i-1} (\mathbf{str}(x_{\bar{i}})')^q \right) \qquad (4)
\end{aligned}$$

We again construct a context-free system of behavioural differential equations, over the set $\bar{X} = \{\bar{x} \mid x \in X\} \cup \{\mathcal{X}\}$, as follows: whenever $x_{\bar{i}} = y[i]$ for all $i < q$, we set:

$$o(\bar{x}) = o(y[1]) \qquad \text{and} \qquad \bar{x}' = \sum_{2 \leq i < q-1} \mathcal{X}^{i-2} o(y[i]) + \sum_{i<q} \mathcal{X}^{q+i-1} \overline{(y[i])}^q$$

and furthermore we set

$$o(\mathcal{X}) = 0 \qquad \text{and} \qquad \mathcal{X}' = 1.$$

Again, we know that this system must have a unique solution, and it follows from (4) that this solution is given by

$$[\![\bar{x}]\!] = \mathbf{str}(x)',$$

for all $x \in X$. Again, it follows that $\mathbf{str}(x)'$ and $\mathbf{str}(x)$ are context-free for all $x \in X$. □

7.4 Generalizing to arbitrary fields

The next proposition shows that under very mild conditions, we can easily show that F-algebraic streams are context-free. We, however, remark that the proof of this theorem is neither coinductive, nor offering a lot of insight. For the definition and discussion of the inverse operator on streams, see e.g. [Rut03] or [Rut05].

Proposition 39. *Given any field F, let σ be an F-algebraic stream that is a solution to the equation $\sum_{i,j \leq m,n} f_{ij} \mathcal{X}^j \sigma^i = 0$ (recall that a polynomial p is of the form $\sum_{i \leq n} f_i \mathcal{X}^i$). Whenever $\sum_{i \leq m} f_{i0} (\sum_{k<i} o(\sigma)^k \sigma^{i-k-1}) \neq 0$, then σ is context-free (and hence also constructively F-algebraic).*

Proof. Note that

$$\sum_{i,j \leq m,n} f_{ij} \sigma^i \mathcal{X}^j = \sum_{i \leq m} f_{i0} \sigma^i + \sum_{i \leq m, 1 \leq j \leq n} f_{ij} \sigma^i \mathcal{X}^j = 0$$

where all $f_{ij} \in F$. Taking derivatives, we now obtain

$$\sigma' \sum_{i \leq m} f_{i0} \left(\sum_{k<i} o(\sigma)^k \sigma^{i-k-1} \right) + \sum_{i \leq m, 1 \leq j \leq n} f_{ij} \sigma^i \mathcal{X}^{j-1} = 0$$

and hence

$$\sigma' \sum_{i \leq m} f_{i0} \left(\sum_{k<i} o(\sigma)^k \sigma^{i-k-1} \right) = - \sum_{i \leq m, 1 \leq j \leq n} f_{ij} \sigma^i \mathcal{X}^{j-1}$$

When $\sum_{i \leq m} f_{i0} (\sum_{k<i} o(\sigma)^k \sigma^{i-k-1} \neq) 0$, we can rewrite this as

$$\sigma' = - \frac{\sum_{i \leq m, 1 \leq j \leq n} f_{ij} \sigma^i \mathcal{X}^{j-1}}{\sum_{i \leq m} f_{i0} (\sum_{k<i} o(\sigma)^k \sigma^{i-k-1})}$$

from which we can easily eliminate the fraction by introducing a new variable τ representing

$$\left(\sum_{i \leq m} f_{i0} \left(\sum_{k<i} o(\sigma)^k \sigma^{i-k-1} \right) \right)^{-1}.$$

Leaving out some further intermediate steps, we obtain the following system of behavioural differential equations in two variables, with $o(\sigma)$ given:

$$\begin{aligned}
\sigma' &= - \left(\sum_{i \leq m, 1 \leq j \leq n} f_{ij} \sigma^i \mathcal{X}^{j-1} \right) \cdot \tau \\
o(\tau) &= \left(\sum_{i \leq m} f_{i0} \cdot (i-1) \cdot o(\sigma)^i \right)^{-1} \\
\tau' &= -o(\tau) \cdot \sigma' \cdot \left(\sum_{i \leq m} f_{i0} \left(\sum_{k \leq i-1} o(\sigma)^k \left(\sum_{j \leq i-k-2} o(\sigma)^j \sigma^{i-j-k-2} \right) \right) \right) \cdot \tau
\end{aligned}$$

□

Finally, there is the following, general result by Fliess [Fli74], stating that the notions of algebraic and constructively algebraic (and hence context-free) coincide for perfect fields:

Proposition 40. *Let F be a perfect field. Then a stream σ over F is F-algebraic if and only if it is constructively F-algebraic.*

This result is not very well-known: its proof [Fli74, Proposition 7] relies on first using Furstenberg's theorem to transform algebraic streams into diagonals of rational power series in two commuting variables, which in turn can be transformed into a systems of equations using the construction given by Fliess. Combining this with the results from Section 4.1, we obtain the result that streams σ over perfect fields F are context-free if and only if they are F-algebraic. In other words, context-free systems of behavioural differential equations characterize precisely the algebraic streams or power series (in a single variable).

8 Conclusions

In this article, we have provided a generalized coalgebraic account of the notion of *algebraicity* or *context-freeness*, through polynomial or context-free systems of behavioural differential equations. Here every derivative is given as a polynomial over the set of nonterminals, with coefficients in the underlying semiring.

This new contribution extends earlier work on coalgebraic presentations of finite deterministic and weighted automata, and can be linked up with these earlier account in the following hierarchy, showing the types of derivatives:

Derivative given as	Type of system	Automaton/Grammar	Associated class of series
Single elements	$X \to S \times X^A$	Deterministic	Finitary
Linear combinations	$X \to S \times (S^X_\omega)^A$	Nondeterministic/weighted	Rational
Polynomials	$X \to S \times S\langle X\rangle^A$	CFG in GNF	Context-free

The main contributions of this article include the following results:

- A correspondence between this notion of context-freeness and (constructive) algebraicity, which can in turn be linked up with classical algebraicity thanks to a result from [Fli74].
- A proof that context-freeness of streams is preserved by the **zip** operation: a result which, as far as the authors are aware, has not been proven before at this level of generality.
- A direct construction yielding systems of behavioural differential equations from q-automata for prime powers q.
- A coalgebraic rephrasing of a classical result due to Chomsky and Schützenberger, which can be used to construct systems of behavioural differential equations, and thus describe a large number of sequences as context-free streams.

It remains interesting to see how much further we can carry this process of finding coalgebraic proofs of language-theoretic theorems. A possible new direction is to see what can be done by varying the coinductively defined operators: for example, by adding the Hadamard product as a primitive, it is known that we will reach a class of power series beyond the algebraic/context-free class described in this paper. A further direction would be to try to give coalgebraic characterizations of the higher levels in the Chomsky hierarchy, namely the context-sensitive and recursively enumerable languages: it remains interesting to see whether we can describe these, too, using systems of behavioural differential equations.

References

[AS03] Jean-Paul Allouche and Jeffrey O. Shallit. *Automatic Sequences – Theory, Applications, Generalizations*. Cambridge University Press, 2003.

[BBB+12] Filippo Bonchi, Marcello M. Bonsangue, Michele Boreale, Jan J. M. M. Rutten, and Alexandra Silva. A coalgebraic perspective on linear weighted automata. *Information and Computation*, 211:77–105, 2012.

[BR11] Jean Berstel and Christophe Reutenauer. *Noncommutative Rational Series with Applications*. Cambridge University Press, 2011.

[BRW12] Marcello M. Bonsangue, Jan J. M. M. Rutten, and Joost Winter. Defining context-free power series coalgebraically. In Dirk Pattinson and Lutz Schröder, editors, *CMCS*, volume 7399 of *Lecture Notes in Computer Science*, pages 20–39. Springer, 2012.

[Brz64] Janusz A. Brzozowski. Derivatives of regular expressions. *Journal of the ACM*, 11:481–494, 1964.

[CS63] Noam Chomsky and Marcel-Paul Schützenberger. *Computer Programming and Formal Systems*, chapter The Algebraic Theory of Context-Free Languages, pages 118–161. North-Holland, 1963.

[Fli74] Michel Fliess. Sur divers produits de sries formelles. *Bulletin de la S.M.F.*, 102:181–191, 1974.

[LB99] Saunders Mac Lane and Garrett B. Birkhoff. *Algebra*. AMS Chelsea Pub., 1999.

[NR10] Milad Niqui and Jan J. M. M. Rutten. Sampling, splitting and merging in coinductive stream calculus. In Claude Bolduc, Jules Desharnais, and Béchir Ktari, editors, *Mathematics of Program Construction*, volume 6120 of *Lecture Notes in Computer Science*, pages 310–330. Springer, 2010.

[PS09] Ion Petre and Arto Salomaa. *Handbook of Weighted Automata*, chapter Algebraic systems and pushdown automata, pages 257–289. Springer, 2009.

[RBR13] Jurriaan Rot, Marcello Bonsangue, and Jan Rutten. Coalgebraic bisimulation-up-to. *to appear in SOFSEM '13*, 2013.

[RS59] Michael O. Rabin and Dana Scott. Finite automata and their decision problems. *IBM J. Res. Dev.*, 3(2):114–125, April 1959.

[Rut98] Jan J. M. M. Rutten. Automata and coinduction (an exercise in coalgebra). In Davide Sangiorgi and Robert de Simone, editors, *CONCUR*, volume 1466 of *Lecture Notes in Computer Science*, pages 194–218. Springer, 1998.

[Rut00] Jan J. M. M. Rutten. Universal coalgebra: a theory of systems. *Theoretical Computer Science*, 249(1):3–80, 2000.

[Rut02] Jan J. M. M. Rutten. Coinductive counting: bisimulation in enumerative combinatorics. *Electr. Notes Theor. Comput. Sci.*, 65(1):286–304, 2002.

[Rut03] Jan J. M. M. Rutten. Behavioural differential equations: a coinductive calculus of streams, automata, and power series. *Theoretical Computer Science*, 308(1-3):1–53, 2003.

[Rut05] Jan J. M. M. Rutten. A coinductive calculus of streams. *Mathematical Structures in Computer Science*, 15(1):93–147, 2005.

[Rut08] Jan J. M. M. Rutten. Rational streams coalgebraically. *Logical Methods in Computer Science*, 4(3), 2008.

[Sak09] Jacques Sakarovitch. *Elements of Automata Theory*. Cambridge University Press, 2009.

[SBBR10] Alexandra Silva, Filippo Bonchi, Marcello M. Bonsangue, and Jan J. M. M. Rutten. Generalizing the powerset construction, coalgebraically. In Kamal Lodaya and Meena Mahajan, editors, *FSTTCS*, volume 8 of *LIPIcs*, pages 272–283. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2010.

[WBR11] Joost Winter, Marcello M. Bonsangue, and Jan J. M. M. Rutten. Context-free languages, coalgebraically. In Andrea Corradini, Bartek Klin, and Corina Cîrstea, editors, *CALCO*, volume 6859 of *Lecture Notes in Computer Science*, pages 359–376. Springer, 2011.