Algebra and Coalgebra of Stream Products

Michele Boreale
University of Florence, Italy

Daniele Gorla
"Sapienza" University of Rome, Italy

Abstract

We study connections among polynomials, differential equations and streams over a field $\mathbb{K}$, in terms of algebra and coalgebra. We first introduce the class of (F, G)-products on streams, those where the stream derivative of a product can be expressed as a polynomial of the streams themselves and their derivatives. Our first result is that, for every (F, G)-product, there is a canonical way to construct a transition function on polynomials such that the induced unique final coalgebra morphism from polynomials into streams is the (unique) $\mathbb{K}$-algebra homomorphism – and vice-versa. This implies one can reason algebraically on streams, via their polynomial representation. We apply this result to obtain an algebraic-geometric decision algorithm for polynomial stream equivalence, for an underlying generic (F, G)-product. As an example of reasoning on streams, we focus on specific products (convolution, shuffle, Hadamard) and show how to obtain closed forms of algebraic generating functions of combinatorial sequences, as well as solutions of nonlinear ordinary differential equations.

2012 ACM Subject Classification Theory of computation → Operational semantics

Keywords and phrases Streams, coalgebras, polynomials, differential equations

Digital Object Identifier 10.4230/LIPIcs.CONCUR.2021.19

Related Version *Full Version*: https://arxiv.org/abs/2107.04455

Supplementary Material *Software (Source Code)*: https://local.disia.unifi.it/boreale/papers/streams.py

1 Introduction

We investigate a connection among polynomials, differential equations and *streams*, i.e., infinite sequences of elements from a set [20]. At a very informal level, this connection can be expressed by the following correspondences: polynomials = syntax; differential equations = operational semantics; streams = abstract (denotational) semantics. There are two important motivations behind this standpoint. (1) Diverse notions of *product* (convolution, shuffle,...) arise in streams, in relation to different models – discrete computations, combinatorial sequences, analytic functions, and more [4, 20]. There is also a close analogy between several forms of products and forms of parallelism arising in concurrency. Our aim is to uniformly accommodate such diverse notions, by automatically deriving an operational semantics for polynomials that is adequate for a given *generic* stream product. (2) Once adequate polynomial syntax and operational semantics have been obtained, one can apply powerful techniques both from algebraic geometry (Gröbner bases [12]) and from coalgebra (coinduction [20]) for reasoning on streams. This includes devising algorithms for deciding stream equivalence. Again, one would like to do so in a uniform fashion w.r.t. an underlying notion of stream product.

Technically, achieving these goals amounts to defining a fully abstract semantics from polynomials to streams, which is essential for algebraic-geometric reasoning on streams. Moreover, one wants the resulting construction to be as much as possible parametric with respect to the underlying notion of stream product.

As hinted above, we will pursue these goals relying on tools from algebra and coalgebra (Section 2). Indeed, it is well-known that, when polynomial coefficients and stream elements are drawn from a field $\mathbb{K}$, both polynomials and streams form $\mathbb{K}$-algebras, i.e., rings with an additional vector space structure over $\mathbb{K}$. Note that, while this algebra structure is fixed for polynomials, it varies with the underlying product for streams. On the other hand, streams also possess a *co*algebraic structure, arising from the operation of stream derivative. On the side of polynomials, it is also natural to interpret a differential equation $\dot{x}_i = p_i$ as a transition $x_i \to p_i$: thus one expects a transition structure, hence a coalgebra, over over polynomials as well. How to extend appropriately transitions from individual variables x_i to monomials and polynomials, though, depends nontrivially on the notion of stream product one wants to model.

Our first result (Section 3) is that the above outlined goals can be achieved for the class of (F, G)-products on streams, where, basically, the derivative of a product of two streams can be expressed as a polynomial of the streams themselves and their derivatives. One can then define a coalgebra structure on polynomials, depending on the given (F, G)-product and differential equations, such that the unique morphism from this coalgebra to the coalgebra of streams is also a $\mathbb{K}$-algebra *homo*morphism (and vice-versa: every homomorphism that satisfies the given differential equations is the unique morphism). Thus, full abstraction is achieved.

A major application of this result, which we view as our main contribution, is an algorithm based on an algebraic-geometric construction for deciding equivalence, i.e. if two polynomials denote the same stream (Section 4). Next, focusing on specific (F, G)-products (convolution, shuffle and Hadamard; Section 5), we show how establishing polynomial (algebraic) equalities on streams may lead to closed forms for generating functions of combinatorial sequences [13], and to solutions of nonlinear ordinary differential equations (ODEs). In the case of convolution product, we also show that the image of the coalgebra morphism is included in the set of algebraic sequences in the sense of [13].

To sum up, we make the following contributions. (1) A unifying treatment of stream products, implying that, under reasonable assumptions, coalgebra morphisms from polynomials to streams are also $\mathbb{K}$-algebra homomorphisms (full abstraction) – and viceversa. (2) An algorithm for deciding polynomial stream equivalence, that relies on the full abstraction result. (3) Based on that, methods for reasoning on generating functions and ordinary differential equations.

Due to space limitations, most proofs, as well as additional technical material, have been omitted and can be found in the full version of this paper [11].

Related work. Rutten's *stream calculus* [20, 21], a coinductive approach to the analysis of infinite sequences (streams), is a major source of inspiration for our work. [20] studies streams, automata, languages and formal power series in terms of coalgebra morphisms and bisimulation. In close analogy with classical analysis, [21] presents coinductive definitions and proofs for a calculus of *behavioural differential equations*, also called *stream differential equations* (SDEs) in later works. A number of applications to difference equations, analytical differential equations, continued fractions and problems from combinatorics, are presented. Convolution and shuffle product play a central role in the stream calculus; a duality between them, mediated by a variation of Laplace transform, exists. This duality also plays a role in our work in relation to generating functions and solutions of ODEs (Section 5). A coinductive treatment of analytic functions and Laplace transform is also presented by Escardo and Pavlovic [19]. Basold et al. [4] enrich the stream calculus with two types of products,

Hadamard and infiltration, and exhibit a duality between the two, mediated by a so-called *Newton transform*. Although these works form a conceptual prerequisite of our study, they do not offer a unifying treatment of the existing disparate notions of stream product, nor any algorithmic treatment of the induced stream equivalences. Bonchi et al. [5] consider an operational approach to streams and convolution product based on weighted automata, which correspond to linear expressions. They offer an equivalence checking algorithm for such automata, and the recognized streams, based on a linear-algebraic construction; however, the polynomial case is not addressed. A related work is Bonchi et al. [3], where it is shown how linear algebra and fractions can be used to decide the equality of streams specified by linear SDEs. Here, differently from them, we can also work with polynomial SDEs.

Most closely related to the present work is Hansen, Kupke and Rutten's [14]. There the authors prove that, when the SDEs defining given operations on streams obey a GSOS syntactic format, then the final coalgebra morphism is also a homomorphism from the free term algebra to the algebra (w.r.t. the given operations) of streams [14, Sect.8]. In contrast, we work with the algebra of polynomials, which besides being a ring and vector space over $\mathbb{K}$, possesses additional structure arising from monomials. All this structure is essential for algebraic-geometric reasoning, and sets our approach apart from those based on term algebras: for one thing, in term algebras there is no obvious analog of Hilbert's basis theorem, a result deeply related to the well-ordering of monomials (cf. Dickson's lemma, [12, Ch.2]), and a crucial ingredient in our decision algorithm. One might consider more complicated GSOS frameworks enriched with equational theories, but even so we doubt one could naturally capture the relevant polynomial structure, in particular as arising from monomials. Nevertheless, a thorough exploration of these issues is an interesting direction for future research.

The GSOS format has also been discussed in the framework of *bialgebras* [14, Sect.9]. Bialgebras are a unified categorical framework that encompass both algebras, viewed as a way of modeling syntax, and coalgebras, viewed as way of describing behaviours; see [16] for a general introduction. The theory of bialgebras is very abstract in spirit, and it is not immediate to pinpoint concrete relations to our results. Furthermore, it requires a substantial background in category theory, which we have preferred to avoid here so as to keep our approach as elementary and accessible as possible. In any case, we anticipate for bialgebras similar difficulties to those discussed above for term algebras. For these reasons, we have preferred to leave the exploration of connections with bialgebras for future work.

Somewhat related to ours is the work of Winters on coalgebra and polynomial systems: see e.g. [23, Ch.3]. Importantly, Winter considers polynomials in *noncommuting* variables: under suitable assumptions, this makes his systems of equations isomorphic to certain context-free grammars; see also [17]. The use of noncommuting variables sets Winter's treatment in a totally different mathematical realm, where the algebraic geometric concepts we rely on here, like ideals and Gröbner bases, are not applicable.

We also mention [7, 10], that adopt a coinductive approach to reason on polynomial ODEs. The ring of multivariate polynomials is employed as a syntax, with *Lie derivatives* inducing a transition structure. An algebraic-geometric algorithm to decide polynomial equivalence is presented. This algorithm as well has inspired our decision method: in particular, as Lie derivatives are precisely the transition structure induced in our framework by the shuffle product, the decision algorithms of [7, 10] are in essence a special case of our algorithm in Section 4. Furthermore, [8, 9] extend the framework of [7, 10] to polynomial partial differential equations, which pose significative additional challenges.

Relations with work in enumerative combinatorics [13, 22] are discussed in Section 5.

2 Background

2.1 Polynomials and differential equations

Let us fix a finite, non empty set of symbols or *variables* $X = \{x_1, \ldots, x_n\}$ and a distinct variable $x \notin X$. Informally, x will act as the independent variable, while $x_1, ..., x_n$ will act as dependent variables, or functions, defined by differential equations (see below). We fix a generic field $\mathbb{K}$ of characteristic 0; $\mathbb{K} = \mathbb{R}$ and $\mathbb{K} = \mathbb{C}$ are typical choices. We let $\mathcal{P} := \mathbb{K}[x, x_1, ..., x_n]$, ranged over by $p, q, ...$, be the set of polynomials with coefficients in $\mathbb{K}$ and indeterminates in $\{x\} \cup X$. We let $\mathcal{M}$, ranged over by $m, m', ...$, be the set of *monomials*, that is the free commutative monoid generated by $\{x\} \cup X$. As usual, we shall denote polynomials as formal finite sums of distinct monomials with nonzero coefficients in $\mathbb{K}$: $p = \sum_{i \in I} r_i m_i$, for $r_i \in \mathbb{K}$. By slight abuse of notation, we shall write the zero polynomial and the empty monomial as 0 and 1, respectively. Over $\mathcal{P}$, one can define the usual operations of sum $p + q$ and product $p \cdot q$, with 0 and 1 as identities, and enjoying commutativity, associativity and distributivity, which make $\mathcal{P}$ a ring; multiplication of $p \in \mathcal{P}$ by a scalar $r \in \mathbb{K}$, denoted rp, is also defined and makes $(\mathcal{P}, +, 0)$ a vector space over $\mathbb{K}$. Therefore, $(\mathcal{P}, +, \times, 0, 1)$ forms a $\mathbb{K}$-algebra.

We shall also fix a set $\mathcal{D} = \{\dot{x}_1 = p_1, ..., \dot{x}_n = p_n\}$ of *differential equations*, one for each $x_i \in X$, with $p_i \in \mathcal{P}$. An *initial condition* for $\mathcal{D}$ is a vector $\rho = (r_1, ..., r_n) \in \mathbb{K}^n$. The pair $(\mathcal{D}, \rho)$ forms an *initial value problem*. The vectors p_i on the right-hand side of the equations are called *drifts*, and $F = (p_1, ..., p_n)$ is a *vector field*. Informally, each $x_i \in X$ represents a placeholder for a function whose derivative is given by p_i, and whose value at the origin is $x_i(0) = r_i$. This terminology is borrowed from the theory of differential equations. Note, however, that depending on the semantics of polynomial product one adopts (see next section), $\mathcal{D}$ can be given diverse interpretations, including *stream* differential equations (SDE, for convolution, see next subsection) in the sense of Rutten [20], and of course *ordinary* differential equations (ODEs, for shuffle).

Notationally, it will be sometimes convenient to regard $\mathcal{D}$ and ρ as functions $\mathcal{D} : X \to \mathcal{P}$ and $\rho : X \to \mathbb{K}$, respectively, such that $\mathcal{D}(x_i) = p_i$ and $\rho(x_i) = r_i$. It is also convenient to extend $\mathcal{D}$ and ρ to x by letting $\mathcal{D}(x) = 1$ and $\rho(x) = 0$; note that, seen as an initial value problem, the last two equations define the identity function. Finally, we let x_0 denote x and, when using $\mathcal{D}$ and ρ as functions, use x_i as a metavariable on $\{x\} \cup X$: this makes $\mathcal{D}(x_i)$ and $\rho(x_i)$ well defined for $0 \leq i \leq n$.

2.2 Streams

We let $\Sigma\langle\mathbb{K}\rangle := \mathbb{K}^\omega$, ranged over by $\sigma, \tau, ...$, denote the set of *streams*, that is infinite sequences of elements from $\mathbb{K}$: $\sigma = (r_0, r_1, r_2, ...)$ with $r_i \in \mathbb{K}$. Often $\mathbb{K}$ is understood from the context and we shall simply write Σ rather than $\Sigma\langle\mathbb{K}\rangle$. When convenient, we shall explicitly consider a stream σ as a function from $\mathbb{N}$ to $\mathbb{K}$ and, e.g., write $\sigma(i)$ to denote the i-th element of σ. By slightly overloading the notation, and when the context is sufficient to disambiguate, the stream $(r, 0, 0, ...)$ $(r \in \mathbb{K})$ will be simply denoted by r, while the stream $(0, 1, 0, 0, ...)$ will be denoted by x; see [20] for motivations behind these notations[1]. Furthermore, a stream made up of all the same element $r \in \mathbb{K}$ will be denoted as $\underline{r} = (r, r, ...)$. One defines the *sum* of two streams σ and τ as the stream $\sigma + \tau$ defined by: $(\sigma + \tau)(i) := \sigma(i) + \tau(i)$ for each $i \geq 0$, where the $+$ on the right-hand side denotes the sum in $\mathbb{K}$. Sum enjoys the usual commutativity

[1] In particular, overloading of the symbol x is motivated by the fact that our semantics of polynomials maps the variable x to the stream $(0, 1, 0, 0, ...)$.

and associativity properties, and has the stream $0 = (0, 0, ...)$ as an identity. Various forms of stream products can also be considered – this is indeed a central theme of our paper. In particular, the *convolution product* $\sigma \times \tau$ and the *shuffle product* $\sigma \otimes \tau$ are defined as follows: $(\sigma \times \tau)(i) := \sum_{0 \leq j \leq i} \sigma(j) \cdot \tau(i-j)$ and $(\sigma \otimes \tau)(i) := \sum_{0 \leq j \leq i} \binom{i}{j} \sigma(j) \cdot \tau(i-j)$, where operations on the right-hand side are carried out in $\mathbb{K}$ and $i \geq 0$. The above operations enjoy alternative, easier to handle formulations based on stream differential equations – see next subsection; there, a crucial notion will be the *derivative* of a stream σ, that is the stream σ' obtained from σ by removing its first element.

Both products are commutative, associative, have $1 = (1, 0, 0, ...)$ as an identity, and distribute over $+$; multiplication of $\sigma = (r_0, r_1, ...)$ by a scalar $r \in \mathbb{K}$, denoted $r\sigma = (r\, r_0, r\, r_1, ...)$, is also defined and makes $(\Sigma, +, 0)$ a vector space over $\mathbb{K}$. Therefore, $(\Sigma, +, \pi, 0, 1)$ forms a $\mathbb{K}$-algebra for each of the considered product operations π. Let us record the following useful properties for future use: $x \times \sigma = (0, r_0, r_1, ...)$ and $r\, \pi\, \sigma = (r\, r_0, r\, r_1, ...)$, where $r \in \mathbb{K}$ and $\pi \in \{\times, \otimes\}$. In view of the second equation above, $r\, \pi\, \sigma$ coincides with $r\sigma$. The first equation above leads to the so called fundamental theorem of the stream calculus, whereby for each $\sigma \in \Sigma$

$$\sigma = \sigma(0) + x \times \sigma' \,. \tag{1}$$

Less commonly found forms of products, like Hadamard and Infiltration products, will be introduced in the next subsection; equations similar to (1) exist also for such products [4, 14].

2.3 Coalgebras and bisimulation

We quickly review some basic definitions and results about coalgebras and bisimulation; see e.g. [20] for a comprehensive treatment. A *(stream) coalgebra with outputs in* $\mathbb{K}$ is an automaton $C = (S, \delta, o)$, where S is a nonempty set of *states*, $\delta : S \to S$ is the *transition* function, and $o : S \to \mathbb{K}$ is the *output* function. A *bisimulation* on C is a binary relation $R \subseteq S \times S$ such that, whenever $(s, t) \in R$, then $o(s) = o(t)$ and $(\delta(s), \delta(t)) \in R$. As usual, there always exists a largest bisimulation on C, denoted $\sim$; it is the union of all bisimulations and it is an equivalence relation on S. Given two coalgebras C_1 and C_2, a *coalgebra morphism* between them is a function $\mu : S_1 \to S_2$ from the states of C_1 to the states of C_2 that preserves transitions and outputs, that is (with obvious notation): $\mu(\delta_1(s)) = \delta_2(\mu(s))$ and $o_1(s) = o_2(\mu(s))$, for each $s \in S_1$. Coalgebra morphisms preserve bisimilarity, in the sense that $s \sim_1 t$ in C_1 if and only if $\mu(s) \sim_2 \mu(t)$ in C_2. A coalgebra C_0 is *final* in the class of coalgebras with outputs in $\mathbb{K}$ if, from every coalgebra C in this class, there exists a unique morphism μ from C_0 to C. In this case, $\sim_0$ in C_0 coincides with equality, and the following *coinduction principle* holds: for every C and $s \sim t$ in C, it holds that $\mu(s) = \mu(t)$ in C_0.

The set of streams Σ can be naturally given a stream coalgebra structure $(\Sigma, (\cdot)', o(\cdot))$, as follows. The *output* of a stream $\sigma = (r_0, r_1, \ldots)$ is $o(\sigma) := r_0$ and its *derivative* is $\sigma' := (r_1, r_2, ...)$, that is σ' is obtained from σ by removing its first element, that constitutes the output of σ. In fact, this makes Σ final in the class of all coalgebras with outputs in $\mathbb{K}$ [20]. This also implies that one can prove equality of two streams by exhibiting an appropriate bisimulation relation relating them (coinduction).

It is sometimes convenient to consider an enhanced form of bisimulation on Σ that relies on the notion of *linear closure*.[2] Given a relation $R \subseteq \Sigma \times \Sigma$, its linear closure $\widehat{R}$ is the set of pairs of the form $(\sum_{i=1}^{n} r_i \sigma_i \;,\; \sum_{i=1}^{n} r_i \tau_i)$, where $n \in \mathbb{N}$, $(\sigma_i, \tau_i) \in R$ and $r_i \in \mathbb{K}$, for every

[2] More general notions that we could have used here are *contextual closure* (see [4, Thm. 2.4] and works on distributive laws for bialgebras [6]. However, the simpler notion of linear closure suffices for our purposes here.

$i \in \{1, \ldots, n\}$. We say that R is a *bisimulation up to linearity* if, for every $(\sigma, \tau) \in R$, it holds that $o(\sigma) = o(\tau)$ and $(\sigma', \tau') \in \widehat{R}$. If R is a bisimulation up to linearity, then $\widehat{R}$ is a bisimulation [20]; since by definition $R \subseteq \widehat{R}$, this implies that $R \subseteq \sim$, the bisimilarity on streams, which coincides with equality.

A *stream differential equation* (SDE) in the unknown σ is a pair of equations of the form $\sigma(0) = r$ and $\sigma' = \phi$, for $r \in \mathbb{K}$ and a stream expression ϕ (that can depend on σ or its components, or even on σ' itself). Under certain conditions on ϕ [14, 20], it can be proven that there is a unique stream σ satisfying the above SDE. In this paper, we shall focus on the case where ϕ is represented by a polynomial expression – this will be formalized in the next section. For the time being, we observe that the product operations defined in the preceding subsection enjoy a formulation in terms of SDEs. In particular (see [4, 14, 20]), for given σ and τ, their convolution and shuffle products are the unique streams satisfying the following SDEs (recall that, as a stream, x denotes $(0, 1, 0, 0, ...)$):

$$(\sigma \times \tau)(0) = \sigma(0) \cdot \tau(0) \qquad (\sigma \times \tau)' = \sigma' \times \tau + \sigma \times \tau' - x \times \sigma' \times \tau' \tag{2}$$

$$(\sigma \otimes \tau)(0) = \sigma(0) \cdot \tau(0) \qquad (\sigma \otimes \tau)' = \sigma' \otimes \tau + \sigma \otimes \tau' \,. \tag{3}$$

From the last equation, note the analogy between shuffle and interleaving of languages. Moreover, the derivative of convolution product is usually defined as $(\sigma \times \tau)' = \sigma' \times \tau + \sigma(0) \times \tau'$; however, we prefer the formulation in (2) because it is symmetric. Two additional examples of stream products are introduced below; see [4] for the underlying motivations. The *Hadamard product* $\odot$ and the *infiltration product* $\uparrow$ can be defined by the following two SDEs.

$$(\sigma \odot \tau)(0) = \sigma(0)\tau(0) \qquad (\sigma \odot \tau)' = \sigma' \odot \tau' \tag{4}$$

$$(\sigma \uparrow \tau)(0) = \sigma(0)\tau(0) \qquad (\sigma \uparrow \tau)' = (\sigma' \uparrow \tau) + (\sigma \uparrow \tau') + (\sigma' \uparrow \tau') \,. \tag{5}$$

Hadamard product $\odot$ is reminiscent of synchronization in concurrency theory and has $\underline{1} := (1, 1, 1, ...)$ as an identity; it is just the componentwise product of two streams, i.e. $(\sigma \odot \tau)(i) = \sigma(i)\tau(i)$, for every $i \geq 0$. Infiltration product $\uparrow$ is again reminiscent of a notion in concurrency theory, namely the fully synchronized interleaving; it has $1 = (1, 0, 0, ...)$ as an identity.

3 (Co)algebraic semantics of polynomials and differential equations

The main result of this section is that, once fixed an initial value problem $(\mathcal{D}, \rho)$, for every product π (with identity 1_π) defined on streams and satisfying certain syntactic conditions, one can build a coalgebra over polynomials such that the corresponding final *morphism* into Σ is also a $\mathbb{K}$-algebra *homomorphism* from $(\mathcal{P}, +, \times, 0, 1)$ to $(\Sigma, +, \pi, 0, 1_\pi)$. In essence, the polynomial syntax and operational semantics reflects exactly the algebraic and coalgebraic properties of the considered π on streams.

To make polynomials a coalgebra, we need to define the output $o : \mathcal{P} \to \mathbb{K}$ and transition $\delta : \mathcal{P} \to \mathcal{P}$ functions. The definition of $o(\cdot)$ is straightforward and only depends on the given initial conditions ρ: we let $o := o_\rho$ be the homomorphic extension of ρ, seen as a function defined over $\{x\} \cup X$, to $\mathcal{P}$. Equivalently, seeing ρ as a point in $\mathbb{K}^{n+1}$, we let $o_\rho(p) := p(\rho)$, that this the polynomial p evaluated at the point ρ. It can be easily checked that $o_\rho(1) = 1$.

The definition of δ, on the other hand, depends on π and is not straightforward. We will confine to products π satisfying SDEs of the form: $(\sigma \,\pi\, \tau)' = F(\sigma, \tau, ...)$, for a given *polynomial* function F. Then we will require that δ on polynomials mimics this equation. For instance, in the case of shuffle product, we expect that $\delta(pq) = p\delta(p) + q\delta(p)$. Therefore,

our first step is to precisely define the class of products on streams that satisfy a polynomial SDE. To this purpose, in what follows we shall consider polynomials $G(y_1) \in \mathbb{K}[y_1]$ and $F(x, y_1, ..., y_4) \in \mathbb{K}[x, y_1, y_2, y_3, y_4]$. These can be identified with polynomial functions on streams: we shall write $G(\sigma_1)$, $F(x, \sigma_1, ..., \sigma_4)$ for the evaluation of G, F in $(\Sigma, +, \pi, 0, 1_\pi)$ with specific streams $x = (0, 1, 0, ...)$ and $\sigma_1, ..., \sigma_4$.

▶ **Definition 3.1** ((F, G)-product on streams). *Let $(\Sigma, +, \pi, 0, 1_\pi)$ be a $\mathbb{K}$-algebra, $F \in \mathbb{K}[x, y_1, y_2, y_3, y_4]$ and $G \in \mathbb{K}[y_1]$. We say that π is a* (F, G)-product *if, for each $\sigma, \tau \in \Sigma$, the following equations are satisfied:*

1. $(\sigma\ \pi\ \tau)(0) = \sigma(0)\tau(0)$;
2. $(\sigma\ \pi\ \tau)' = F(x, \sigma, \sigma', \tau, \tau')$;
3. $1_\pi(0) = 1$ *and* $1'_\pi = G(1_\pi)$.

▶ Remark 3.2. Notice that $1_\pi(0) = 1$ in Definition 3.1(3) is a necessary condition, that follows from Definition 3.1(1). Indeed, let $1_\pi(0) = r \in \mathbb{K}$. Since 1_π is the identity of π, for every σ we must have $\sigma\, \pi\, 1_\pi = \sigma$, hence $(\sigma\, \pi\, 1_\pi)(0) = \sigma(0)$. On the other hand, by Definition 3.1(1), $(\sigma\, \pi\, 1_\pi)(0) = \sigma(0)\, 1_\pi(0) = \sigma(0)\, r$. As σ is arbitrary, we can take $\sigma(0) \neq 0$ and multiply $\sigma(0)\, r = \sigma(0)$ by $\sigma(0)^{-1}$; this gives $r = 1$. However, we prefer to keep $1_\pi(0) = 1$ explicit in the definition, for the sake of clarity. Finally, let us note that the general theory of SDEs [14] ensures that conditions (1), (2), (3) in Definition 3.1 univocally define a binary operation π on streams, but in general not that π enjoys the ring axioms for product, a fact that we must assume from the outset.

▶ **Example 3.3.** For the products introduced in Section 2, the pairs of polynomials (F, G) are as defined as follows.

- $F_\times = y_2y_3 + y_1y_4 - xy_2y_4$. Note that $F_\times = y_2y_3 + (y_1 - xy_2)y_4$, where $y_1 - xy_2$ corresponds to $\sigma - x \times \sigma' = \sigma(0)$; this gives the asymmetric definition of convolution.
- $F_\otimes = y_2y_3 + y_1y_4$.
- $F_\odot = y_2y_4$.
- $F_\uparrow = y_2y_3 + y_1y_4 + y_2y_4$.

The identity stream for convolution, shuffle and infiltration is defined by $1_\pi(0) = 1$ and $1'_\pi = 0$, i.e., in these cases the polynomial G is 0. For the Hadamard product, the identity is given by $1_\pi(0) = 1$ and $1'_\pi = 1_\pi$, i.e., the polynomial G in this case is y_1.

Given a (F, G)-product π on streams, δ_π is defined in a straightforward manner on monomials, then extended to polynomials by linearity. Below, we assume a total order on variables $x_0 < x_1 < \cdots < x_n$ and, for $m \neq 1$, let $\min(m)$ denote the smallest variable occurring in m w.r.t. such a total order[3].

▶ **Definition 3.4** (transition function δ_π). *Let π be a (F, G)-product on streams. We define $\delta_\pi : \mathcal{P} \to \mathcal{P}$ by induction on the size of $p \in \mathcal{P}$ as follows.*

$$\delta_\pi(1) = G(1) \tag{6}$$

$$\delta_\pi(x_i) = \mathcal{D}(x_i) \tag{7}$$

$$\delta_\pi(x_i\, m) = F(x, x_i, \delta_\pi(x_i), m, \delta_\pi(m)) \quad (m \neq 1,\ \ x_i = \min(x_i m)) \tag{8}$$

$$\delta_\pi\left(\sum_{i \in I} r_i\, m_i\right) = \sum_{i \in I} r_i\, \delta_\pi(m_i)\,. \tag{9}$$

[3] In Definition 3.4, we are in effect totally ordering monomials by graded lexicographic order (grlex, see [12, Ch.1]), and then proceeding by induction on this order.

Returning to the products defined in Section 2, we have:

$$\delta_\pi(1) = \begin{cases} 0 & \text{for } \pi \in \{\times, \otimes, \uparrow\} \text{ (convolution, shuffle, infiltration)} \\ 1 & \text{for } \pi = \odot \text{ (Hadamard)} \end{cases}$$

$$\delta_\pi(x_i\, m) = \begin{cases} \mathcal{D}(x_i) \cdot m + x_i \cdot \delta_\pi(m) - x \cdot \mathcal{D}(x_i) \cdot \delta_\pi(m) & \text{for } \pi = \times \text{ (convolution)} \\ \mathcal{D}(x_i) \cdot m + x_i \cdot \delta_\pi(m) & \text{for } \pi = \otimes \text{ (shuffle)} \\ \mathcal{D}(x_i) \cdot \delta_\pi(m) & \text{for } \pi = \odot \text{ (Hadamard)} \\ \mathcal{D}(x_i) \cdot m + \mathcal{D}(x_i) \cdot \delta_\pi(m) + x_i \cdot \delta_\pi(m) & \text{for } \pi = \uparrow \text{ (infiltration)}\,. \end{cases}$$

We must now impose certain additional sanity conditions on F, to ensure that the final coalgebra morphism induced by δ_π, as just defined, is also an algebra homomorphism. In the rest of the paper, we will make use of the following abbreviation $F_\pi[p;q] := F(x, p, \delta_\pi(p), q, \delta_\pi(q))$. The necessity of the following conditions is self-evident, if one thinks of $F_\pi[p;q]$ as $\delta_\pi(p \cdot q)$ (see Lemma 3.6 below).

▶ **Definition 3.5** (well-behaved F). *Let π be a (F, G)-product on streams. We say that π is* well-behaved *if the following equalities hold, for every $p, q \in \mathcal{P}$, $m_1, m_2, m_i \in \mathcal{M}$, $x_i \in \{x\} \cup X$ and $r_i \in \mathbb{K}$:*

$$F_\pi[1;q] = \delta_\pi(q) \tag{10}$$

$$F_\pi[x_i m_1; m_2] = F_\pi[m_1; x_i m_2] \tag{11}$$

$$F_\pi\left[\sum_{i\in I} r_i\, m_i \;;\; q\right] = \sum_{i\in I} r_i\, F_\pi[m_i; q] \tag{12}$$

$$F_\pi[p;q] = F_\pi[q;p]\,. \tag{13}$$

All products defined in Section 2 are well-behaved. The following key technical result connects morphism to homomorphism properties induced by π and is crucial in the proof of Theorem 3.7, that is the main result of this section.

▶ **Lemma 3.6.** *Let π be a well-behaved (F, G)-product. Then, for every $p, q \in \mathcal{P}$, it holds that $\delta_\pi(p \cdot q) = F_\pi[p;q]$.*

▶ **Theorem 3.7.** *Let π be a well-behaved (F, G)-product. Then the (unique) coalgebra morphism μ_π from $(\mathcal{P}, \delta_\pi, o_\rho)$ to $(\Sigma, (\cdot)', o)$ is a $\mathbb{K}$-algebra homomorphism from $(\mathcal{P}, +, \cdot, 0, 1)$ to $(\Sigma, +, \pi, 0, 1_\pi)$.*

Intuitively, the proof consists in showing that μ_π preserves all the operations in $\mathcal{P}$, by exhibiting in each case an appropriate bisimulation relation in $\Sigma \times \Sigma$ and then applying coinduction. The most crucial case is product, where one shows that the relation consisting of all pairs $(\mu_\pi(p_1 \cdot \ldots \cdot p_k)\,,\ \mu_\pi(p_1)\,\pi \ldots \pi\,\mu_\pi(p_k))$ $(k > 0)$ is a bisimulation up to linearity. Lemma 3.6 is used to prove that μ_π preserves transitions: e.g., by letting $p = p_2 \cdot \ldots \cdot p_k$, it allows one to conclude that the pair of derivatives $\mu_\pi(p_1 \cdot p)' = \mu_\pi(F_\pi[p_1; p])$ and (slightly abusing the $F_\pi[\cdot;\cdot]$ notation) $(\mu_\pi(p_1)\,\pi\,\mu_\pi(p))' = F_\pi[\mu_\pi(p_1); \mu_\pi(p)]$ are still in relation, up to linearity.

To conclude the section, we also present a sort of converse of the previous theorem. That is, μ_π is the only $\mathbb{K}$-algebra homomorphism that respects the initial value problem, i.e. that satisfies $\mu_\pi(x_i)' = \mu_\pi(\mathcal{D}(x_i))$ and $\mu_\pi(x_i)(0) = \rho(x_i)$. This is an immediate corollary of the following result and of the uniqueness of the final coalgebra morphism.

▶ **Proposition 3.8.** *Let π be a well-behaved (F, G)-product and ν be a $\mathbb{K}$-algebra homomorphism from $(\mathcal{P}, +, \cdot, 0, 1)$ to $(\Sigma, +, \pi, 0, 1_\pi)$ that respects the initial value problem $(\mathcal{D}, \rho)$. Then, ν is a coalgebra morphism from $(\mathcal{P}, \delta_\pi, o_\rho)$ to $(\Sigma, (\cdot)', o)$.*

4 Deciding stream equality

One benefit of a polynomial syntax is the possibility of applying techniques from algebraic geometry to reason about stream equality. We will devise an algorithm for checking whether two given polynomials are semantically equivalent, that is, are mapped to the same stream under μ_π. Note that, by linearity of $\mu_\pi(\cdot)$, we have that $\mu_\pi(p) = \mu_\pi(q)$ if and only if $\mu_\pi(p) - \mu_\pi(q) = \mu_\pi(p-q) = 0$. Therefore, checking semantic equivalence of two polynomials reduces to the problem of checking if a polynomial is equivalent (bisimilar) to 0. Before introducing the actual algorithm for checking this, we quickly recall a few notions from algebraic geometry; see [12, Ch.1–4] for a comprehensive treatment.

A set of polynomials $I \subseteq \mathcal{P}$ is an *ideal* if $0 \in I$ and, for all $p_1, p_2 \in I$ and $q \in \mathcal{P}$, it holds that $p_1 + p_2 \in I$ and $q \cdot p_1 \in I$. Given a set of polynomials S, the *ideal generated by* S is

$$\langle\, S \,\rangle := \left\{ \sum_{j=1}^{k} q_j \cdot p_j \;:\; k \geq 0 \;\wedge\; \forall j \leq k.(q_j \in \mathcal{P} \wedge p_j \in S) \right\}.$$

By the previous definition, we have that $\langle\, \emptyset \,\rangle := \{0\}$. Trivially, $I = \langle\, S \,\rangle$ is the smallest ideal containing S, and S is called a set of *generators* for I. It is well-known that every ideal I admits a finite set S of generators (Hilbert's basis theorem). By virtue of this result, any infinite ascending chain of ideals, $I_0 \subseteq I_1 \subseteq I_2 \subseteq \cdots \subseteq \mathcal{P}$, stabilizes in a finite number of steps: that is, there is $k \geq 0$ s.t. $I_{k+j} = I_k$ for each $j \geq 0$ (Ascending Chain Condition, ACC). A key result due to Buchberger is that, given a finite $S \subseteq \mathcal{P}$, it is possible to decide whether $p \in I = \langle\, S \,\rangle$, for any polynomial p. As a consequence, also ideal inclusion $I_1 \subseteq I_2$ is decidable, given finite sets of generators for I_1, I_2. These facts are consequences of the existence of a set of generators B for I, called *Gröbner basis*, with a special property: $p \in I$ if and only if $p \bmod B = 0$, where "mod B" denotes the remainder of the multivariate polynomial division of p by B. There exist algorithms to build Gröbner bases which, despite their exponential worst-case complexity, turn out to be effective in many practical cases [12, Ch.4].

In what follows, we fix a well-behaved (F, G)-product π, and let δ_π and μ_π denote the associated transition function and coalgebra morphism. Moreover, we denote by $p^{(j)}$ the j-th derivative of p, i.e. $p^{(0)} := p$ and $p^{(j+1)} := \delta_\pi(p^{(j)})$. The actual decision procedure is presented below as Algorithm 1. Intuitively, to prove that $\mu_\pi(p) = 0$, one might check if $o_\rho(p^{(j)}) = 0$ for every j, which is of course non effective. But due to ACC, at some point $p^{(j)} \in \langle\, \{p^{(0)}, \ldots, p^{(j-1)}\} \,\rangle$, which implies the condition $o_\rho(p^{(j)}) = 0$ holds for *all* j's. The correctness of this algorithm can be proven easily, under an additional mild condition on F: we require that $F \in \langle\, \{y_3, y_4\} \,\rangle$ seen as an ideal in $\mathbb{K}[x, y_1, ..., y_4]$. Explicitly, $F = h_1 y_3 + h_2 y_4$ for some $h_1, h_2 \in \mathbb{K}[x, y_1, ..., y_4]$. The polynomials F for the products in Section 2 all satisfy this condition: for example, $F_\times = y_2 y_3 + (y_1 - x y_2) y_4$.

Algorithm 1 Checking equivalence to zero.

Input: $p \in \mathcal{P}$, a well-behaved (F, G)-product π
Output: YES ($\mu_\pi(p) = 0$) or NO ($\mu_\pi(p) \neq 0$)
1: **for all** $k \geq 0$ **do**
2: **if** $o_\rho(p^{(k)}) \neq 0$ **then return** NO
3: **if** $p^{(k)} \in \langle\, \{p^{(0)}, \ldots, p^{(k-1)}\} \,\rangle$ **then return** YES
4: **end for**

▶ **Theorem 4.1.** *Let π be a well-behaved (F, G)-product, with $F \in \langle\, \{y_3, y_4\} \,\rangle$. Algorithm 1 terminates, and returns YES if and only if $\mu_\pi(p) = 0$.*

Proof. Non termination for some input polynomial p would imply that, for all $k \geq 0$, $p^{(k+1)} \notin I_k := \langle \{p^{(0)}, \ldots, p^{(k)}\} \rangle$. This in turn would imply an ever ascending chain of ideals $I_0 \subsetneq I_1 \subsetneq \cdots$, contradicting ACC.

If the algorithm returns NO, then for some k we must have (recall that $\sigma^{(k)}$ stands for the k-th stream derivative of σ): $o_\rho(p^{(k)}) = o(\mu_\pi(p)^{(k)}) = (\mu_\pi(p)^{(k)})(0) \neq 0$, thus $\mu_\pi(p) \neq 0$.

Assume now the algorithm returns YES. Then there exists $k \geq 0$ such that $o_\rho(p^{(j)}) = 0$, for every $0 \leq j \leq k$, and $p^{(k)} \in \langle \{p^{(0)}, \ldots, p^{(k-1)}\} \rangle$. Excluding the trivial case $p = 0$, we can assume $k \geq 1$. If we prove that $p^{(k+j)} \in \langle \{p^{(0)}, \ldots, p^{(k-1)}\} \rangle$ for every $j \geq 0$, the thesis follows: indeed, by $p^{(k+j)} = \sum_{i=0}^{k-1} q_i \cdot p^{(i)}$, for some $q_i \in \mathcal{P}$, and by $o_\rho(p^{(i)}) = 0$ for every $0 \leq i \leq k-1$, it also follows $(\mu_\pi(p))(j) = (\mu_\pi(p))^{(j)}(0) = o_\rho(p^{(k+j)}) = 0$. Now the proof that $p^{(k+j)} \in \langle \{p^{(0)}, \ldots, p^{(k-1)}\} \rangle$ is by induction on j. The base case ($j = 0$) holds by assumption. For the induction step, let us consider $p^{(k+j+1)}$. By definition, $p^{(k+j+1)} = \delta_\pi(p^{(k+j)})$; by induction $p^{(k+j)} = \sum_{i=0}^{k-1} q_i \cdot p^{(i)}$, for some $q_i \in \mathcal{P}$. By (9) and Lemma 3.6, $p^{(k+j+1)} = \sum_{i=0}^{k-1} \delta_\pi(q_i \cdot p^{(i)}) = \sum_{i=0}^{k-1} F_\pi[q_i; p^{(i)}]$. By hypothesis $F \in \langle \{y_3, y_4\} \rangle$, hence $F_\pi[q_i; p^{(i)}] \in \langle \{p^{(i)}, p^{(i+1)}\} \rangle$, for every i, therefore $F_\pi[q_i; p^{(i)}] \in \langle \{p^{(0)}, \ldots, p^{(k-1)}\} \rangle$, as by hypothesis $p^{(k)} \in \langle \{p^{(0)}, \ldots, p^{(k-1)}\} \rangle$. This suffices to conclude. ◀

We first illustrate the algorithm with a simple, linear example.

▶ **Example 4.2** (Fibonacci numbers)**.** Consider the initial value problem $(\mathcal{D}, \rho)$ given by the following equations.

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = x_1 + x_2 \end{cases} \qquad \begin{cases} \rho(x_1) = 0 \\ \rho(x_2) = 1\,. \end{cases} \tag{14}$$

Let us consider here the convolution product $\times$. It is easily checked that x_1 defines the Fibonacci numbers: $\mu_\times(x_1) = (0, 1, 1, 2, 3, 5, 8, 13, \ldots)$. We want to prove the following equation:

$$\mu_\times(x_1 \cdot (1 - x - x^2)) = \mu_\times(x)\,. \tag{15}$$

Equivalently, using Algorithm 1, we check that $\mu_\times(x_1 \cdot (1 - x - x^2) - x) = 0$. Let $p(x, x_1) := x_1 \cdot (1 - x - x^2) - x$. Then, an execution of Algorithm 1 consists of the following steps.

- **($k = 0$):** $\rho(p) = p(0, 1) = 0$ and $p^{(0)} = p(x, x_1) \notin \langle \emptyset \rangle = \{0\}$.
- **($k = 1$):** $p^{(1)} = x_2 \cdot (1 - x - x^2) - x_1 \cdot (1 + x) - x \cdot x_2 \cdot (1 + x) - 1 = x_2 - x_1 - x_1 x - 1$. Hence, $\rho(p^{(1)}) = 1 - 1 = 0$ and $p^{(1)} \notin \langle p \rangle$.
- **($k = 2$):** $p^{(2)} = x_1 + x_2 - x_2 - (x_2 x + x_1 - x x_2) = 0$. Hence, $\rho(p^{(2)}) = 0$ and trivially $p^{(2)} \in \langle p, p^{(1)} \rangle$.

We conclude that $\mu_\times(p) = 0$.

We now discuss a nonlinear example based on shuffle product.

▶ **Example 4.3** (double factorial of odd numbers)**.** Consider the initial value problem $(\mathcal{D}, \rho)$ given by the following equation.

$$\dot{y} = y^3 \qquad \rho(y) = 1\,. \tag{16}$$

Let us consider here the shuffle product $\otimes$. It is easily checked that $\mu_\otimes(y) = (1, 1, 3, 15, 105, 945, 10395, 135135, \ldots)$, the sequence of double factorials of odd numbers (sequence A001147 in [1]). We want to check the following equation

$$\mu_\otimes(y^2(x - 1/2) + 1/2) = 0\,. \tag{17}$$

using Algorithm 1. Let $q(x, y) := y^2(x - 1/2) + 1/2$. An execution of Algorithm 1 consists of the following steps.

- **($k = 0$):** $\rho(q) = q(0,1) = 0$ and $q^{(0)} = q(x,y) \notin \langle\, \emptyset \,\rangle = \{0\}$.
- **($k = 1$):** $q^{(1)} = 2y^4x - y^4 + y^2 = 2y^2q$, hence $q^{(1)} \in \langle\, q \,\rangle$.

We conclude that $\mu_\otimes(q) = 0$.

▶ Remark 4.4. Note that we can define the generating function associated to Fibonacci numbers, that is the function $g(z)$ whose Taylor series expansion is $\sum_{j\geq 0} f_j z^j$ (where f_j are the Fibonacci numbers); such a generating function is

$$g(z) = \frac{z}{1 - z - z^2}\,. \tag{18}$$

Now, from [4] it is known that the convolution product admits an inverse of a given stream σ whenever $\sigma(0) \neq 0$. Thus, from (15) we obtain $\mu_\times(x_1) = \mu_\times(x) \times (\mu_\times(1 - x - x^2))^{-1} = \frac{\mu_\times(x)}{1-\mu_\times(x)-\mu_\times(x)^2}$, where we use the usual notation $\frac{\sigma}{\tau}$ to denote $\sigma \times \tau^{-1}$. This equation for $\mu_\times(x_1)$ is structurally identical to (18): this is of course no coincidence, as algebraic identities on streams correspond exactly to algebraic identities on generating functions. This will be made precise in the next section – see in particular Proposition 5.2.

Similarly, the equivalence $\mu_\otimes(p) = 0$ obtained for the double factorial equations, when solved algebraically for x_1 yields the exponential generating function for A001147, that is $g(z) = \sqrt{1/(1-2z)}$: see Example 5.8 in Subsection 5.3.

We finally point out that Algorithm 1 can be easily modified to actually *find* all polynomials p, up to a prescribed degree, s.t. $\mu_\pi(p) = 0$, along the lines of a similar procedure in [10]. Indeed, we actually found the polynomials in both examples above using this modified algorithm[4].

5 Shuffle, convolution and generating functions

We study the relation of the shuffle and convolution products, and of the corresponding morphisms, with algebraic sequences arising in enumerative combinatorics [13, 22], and with solutions of ordinary differential equations; Hadamard product plays also a role in connecting the other two products. Our aim here is not to prove any new identity, but rather to relate our framework with certain well established notions and results in these fields. In particular, we will argue that our results can be useful for combinatorial reasoning on sequences and ODEs: this means chiefly finding generating functions of sequences, ODE solutions, and/or establishing nontrivial relations among them.

5.1 Generating functions

For a stream $\sigma = (r_0, r_1, ..., r_j, ...)$, we let the *ordinary generating function* [13, 22] of σ in the variable z be the power series $\mathcal{G}[\sigma](z) := \sum_{j\geq 0} r_j z^j$. We shall normally understand $\mathcal{G}[\sigma](z)$ as a formal power series, which is just another convenient, functional notation for the stream σ. When $\mathbb{K} = \mathbb{R}$ or $\mathbb{K} = \mathbb{C}$, it is sometimes convenient to consider z as a real or complex variable[5]: in this case, $\mathcal{G}[\sigma](z)$ defines a (real or complex) analytic function around 0, provided that its radius of convergence is positive. In fact, we shall see that, when $\sigma = \mu_\times(p)$, then $\mathcal{G}[\sigma](z)$ is analytic. We denote by $\mathcal{G}^{-1}[g(z)]$ the inverse transformation,

[4] Python code, with instructions and examples, available at `https://local.disia.unifi.it/boreale/papers/streams.py`.

[5] For example, the study of the generating function in a complex analytic sense, in particular of its poles, provides detailed information on the asymptotic growth of the elements of σ; see [13].

mapping a power series $g(z) = \sum_{j\geq 0} r_j z^j$ back to $\sigma = (r_0, r_1, ...)$. More precisely, for any (formal or analytic) power series $g(z)$ around the origin, $\mathcal{G}^{-1}[g(z)]$ can be obtained by taking the Taylor coefficients of $g(z)$:

$$\mathcal{G}^{-1}[g(z)] = \left(\frac{g^{(0)}(0)}{0!}, \frac{g^{(1)}(0)}{1!}, \frac{g^{(2)}(0)}{2!}, ...\right) \tag{19}$$

where $g^{(j)}(z)$ denotes the j-th derivative of $g(z)$, in either formal or analytic sense. With the same convention on z, we let the *exponential* generating function of σ to be the Taylor series $\mathcal{E}[\sigma](z) := \sum_{j\geq 0} \frac{r_j}{j!} z^j$. Again, $\mathcal{E}^{-1}[g(z)]$ denotes the inverse transformation, mapping a (formal or analytic) power series $g(z)$ to the stream of its derivatives evaluated at 0:

$$\mathcal{E}^{-1}[g(z)] = (g^{(0)}(0), g^{(1)}(0), g^{(2)}(0), ...)\,. \tag{20}$$

Letting $\mathsf{fact} := (0!, 1!, 2!, ...)$ and $\exp(z) := \sum_{j\geq 0} \frac{z^j}{j!}$, the relation between $\mathcal{G}$ and $\mathcal{E}$ can be written as follows, where the Hadamard product on power series is defined as $(\sum_j a_j z^j) \odot (\sum_j b_j z^j) := \sum_j (a_j b_j) z^j$ as expected:

$$\mathcal{E}[\sigma](z) = \exp(z) \odot \mathcal{G}[\sigma](z) \tag{21}$$

$$\mathcal{E}^{-1}[g(z)] = \mathsf{fact} \odot \mathcal{G}^{-1}[g(z)]\,. \tag{22}$$

Again, for $\sigma = \mu_\otimes(p)$, we will see that $\mathcal{E}[\sigma](z)$ is analytic. The maps $\mathcal{G}[\cdot]$ and $\mathcal{E}[\cdot]$ act as $\mathbb{K}$-algebra homomorphisms between streams and functions. In particular, products of streams is transformed into product of functions[6], that is [13, 22]:

$$\mathcal{G}[\sigma \times \tau](z) = \mathcal{G}[\sigma](z) \cdot \mathcal{G}[\tau](z) \qquad\qquad \mathcal{E}[\sigma \otimes \tau](z) = \mathcal{E}[\sigma](z) \cdot \mathcal{E}[\tau](z)\,.$$

These relations allow one to transform algebraic equations on streams into algebraic equations on generating functions. One reason to perform this transformation is that, if a closed expression for the generating function can be found via analytic manipulations, the actual stream can be recovered by applying the inverse transforms (19) and (20) – that is essentially via Taylor expansion.

5.2 Algebraic streams

In what follows, we let p range over $\mathcal{P} = \mathbb{K}[x, x_1, ..., x_n]$ and $q = q(x, y)$ over $\mathbb{K}[x, y]$, while $g(z)$ still denotes a formal power series or analytic function at the origin.

▶ **Definition 5.1** (algebraic streams, [13]). *A function $g(z)$ is* algebraic *if there is a nonzero polynomial $q(x, y)$ such that $q(z, g(z))$ is identically 0. In this case, $g(z)$ is called a* branch *of $q(x, y)$. A stream σ is algebraic if $\mathcal{G}[\sigma](z)$ is algebraic.*

If the degree of $q(x, y)$ in y is k, then $q(x, y)$ has at most k branches. For example, $q(x, y) = y^2 + x - 1$ has two distinct branches, that is algebraic functions: $g(z) = \pm\sqrt{1 - z}$. When the coefficients of q are drawn from a subfield of $\mathbb{C}$, then it can be shown that the corresponding branches are also complex analytic (hence real analytic when restricted to $\mathbb{R}$); see [2]. Our starting point in the study of the connections between coalgebra morphisms and algebraic streams is the following simple result, whose easy proof relies on the fact that both μ_π and $\mathcal{G}$ are $\mathbb{K}$-algebra homomorphisms.

[6] When interpreted in a purely formal sense, hence in terms of streams: the equation for $\mathcal{G}$ just defines an alternative notation for convolution product; the equation for $\mathcal{E}$ reduces to (25).

▶ **Proposition 5.2.** *Let $p \in \mathcal{P}$ and $\sigma = \mu_\times(p)$. Suppose there is a polynomial $q(x,y) \neq 0$ such that $\mu_\times(q(x,p)) = 0$. Then $\mathcal{G}[\sigma](z)$ is a branch of q. The corresponding statement for $\mu_\otimes(\cdot)$ and $\mathcal{E}[\cdot]$ is also true.*

Pragmatically, the above result implies that, if one proves a nontrivial polynomial equation $q(x,\sigma) = 0$ for $\sigma = \mu_\pi(p)$ ($\pi \in \{\times, \otimes\}$), e.g. by using the algorithm in the previous section, then one can recover σ by Taylor expansion of one of the branches of q; see Example 5.4 below.

In the case of the convolution product $\times$, the result also implies that, under the given hypotheses, σ is algebraic. In fact, something more general can be said. Let the considered system of differential equations and initial conditions be $\mathcal{D} = \{\dot{x}_1 = p_1, ..., \dot{x}_n = p_n\}$ and $\rho = (r_1, ..., r_n) \in \mathbb{K}^n$, respectively; let $\sigma_i := \mu_\times(x_i)$ for $i = 1, ..., n$. As a consequence of (1), it is easy to check that the streams σ_i, hence the corresponding generating functions $\mathcal{G}[\sigma_1](z), ..., \mathcal{G}[\sigma_n](z)$, satisfy the following system of polynomial equations in the variables $x_1, ..., x_n$:

$$x_1 = r_1 + xp_1 \qquad \cdots \qquad x_n = r_n + xp_n\,. \tag{23}$$

In the terminology of Kuich and Salomaa [17, Ch.14], (23) is a *weakly strict* polynomial system (in the single letter alphabet $\{x\}$). They prove that there is a unique tuple of formal power series that solves this system, which therefore coincides with $(\sigma_1, ..., \sigma_n)$. Moreover, by invoking elimination theory, Kuich and Salomaa prove that, for each $i = 1, ..., n$, (23) implies a nontrivial polynomial equation $q(x, x_i) = 0$ for the variable x_i: see [17, Ch.16, Cor.16.11], which covers the case $\mathbb{K} = \mathbb{Q}$. We sum up the above discussion in the following.

▶ **Corollary 5.3** (algebraicity of $\mu_\times$)**.** *Suppose that $\mathbb{K} = \mathbb{Q}$. Then, for each $p \in \mathcal{P}$, $\mu_\times(p)$ is an algebraic stream in the sense of Definition 5.1.*

When $\mathbb{K} = \mathbb{Q}$, the above result implies that $\mathcal{G}[\mu_\times(p)](z)$ is analytic. At present we do not know if the converse of this corollary is true, i.e. if all algebraic functions are expressible via polynomial SDE.

▶ **Example 5.4** (Catalan numbers)**.** Let $\mathbb{K} = \mathbb{R}$. Consider the differential equation in one dependent variable (here $y = x_1$)

$$\dot{y} = y^2 \tag{24}$$

with the initial condition $y(0) = 1$. Let us analyse this equation from the point of view of convolution product. By (1), we have $\mu_\times(y) = \mu_\times(y)(0) + x \times (\mu_\times(y))' = 1 + x \times \mu_\times(\delta_\times(y)) = 1 + x \times \mu_\times(y^2) = 1 + x \times \mu_\times(y)^2$. Let $\sigma = \mu_\times(y)$, this leads to the polynomial equation $q(x,\sigma) = 0$, where $q(x,y) = y - xy - y^2 - 1$. Solving for y as a function of x (and renaming x to z), we obtain two branches, $y(z) = (1 \pm \sqrt{1-4z})/2z$. By Proposition 5.2, σ must be the series of Taylor coefficients of one or the other of these two functions. One checks that the stream obtained using the minus sign solves the equation:

$$\sigma \;=\; \mathcal{G}^{-1}\left[\frac{1-\sqrt{1-4z}}{2z}\right] = (1, 1, 2, 5, 14, 42, 132, ...)\,.$$

These are the Catalan numbers, sequence A000108 in [1].

5.3 Solutions of ODEs

The shuffle product $\otimes$ provides a connection between streams and differential equations. A recurrent motif here is that streams and their generating functions can be used to reason on solutions of ODEs – and the other way around. In what follows, solutions might be considered in both formal and analytic sense.

When applied to $\otimes$, Proposition 5.2 may help one to recover closed forms for algebraic solutions of a ODE system, in case they exist. This is entailed by Corollary 5.6 below. In the rest of the section, we let $\mathbf{x}(z) = (x_1(z), ..., x_n(z))$ denote a solution around 0 of $\mathcal{D} = \{\dot{x}_1 = p_1, ..., \dot{x}_n = p_n\}$, considered as a system of ODEs, with the given initial conditions $\mathbf{x}(0) := \rho \in \mathbb{K}^n$. In particular, note that, when $\mathbb{K} = \mathbb{R}$, a solution always exists, is unique and analytic (Picard-Lindelöf theorem). For $p(x, x_1, ..., x_n) \in \mathcal{P}$, we let $p(z, \mathbf{x}(z))$ denote the composition of p as a function with $(z, \mathbf{x}(z))$; in turn, $p(z, \mathbf{x}(z))$ is a formal power series or analytic function around the origin. The following proposition provides a link between solutions of ODEs and shuffle product and the induced morphism, via exponential generating functions. The essential point here is that $\delta_\otimes$ coincides with Lie derivative.

▶ **Proposition 5.5.** $p(z, \mathbf{x}(z)) = \mathcal{E}[\mu_\otimes(p)](z)$.

When $\mathbb{K} = \mathbb{R}$, the above result implies that $\mathcal{E}[\mu_\otimes(p)](z)$ is always real analytic.

▶ **Corollary 5.6** (algebraic solutions of ODEs)**.** *Suppose that, for some nonzero $q = q(x, y)$, we have $\mu_\otimes(q(x, p)) = 0$. Then $p(z, \mathbf{x}(z))$ is a branch of $q(x, y)$.*

Proof. By Proposition 5.2, we deduce that $\mathcal{E}[\mu_\otimes(p)](z)$ is a branch of $q(x, y)$. But, by Proposition 5.5, $p(z, \mathbf{x}(z)) = \mathcal{E}[\mu_\otimes(p)](z)$. ◀

A discussion on the relation of $\mu_\otimes$ with algebraic and other classes of streams is deferred to the end of the section. We illustrate now the above results with a simple example.

▶ **Example 5.7** (factorial numbers and the solution of $\dot{y} = y^2$)**.** Consider again the equation $\dot{y} = y^2$ with $y(0) = 1$ of Example 5.4. This time we analyse this equation from the point of view of shuffle product. Let $\sigma = \mu_\otimes(y)$. Consider the polynomial $q = q(x, y) := yx - y + 1$. One checks that $q \sim 0$ in the coalgebra over $\mathcal{P}$ induced by $\delta_\otimes$: to see this, one applies the algorithm in Section 4, noting that $o(q) = q(0, 1) = 0$ and that $\delta_\otimes(q) = yq \in \langle q \rangle$. This implies $\mu_\otimes(q(x, y)) = 0$, hence, according to Proposition 5.2, $\mathcal{E}[\sigma](z)$ is a branch of $q(x, y)$. Now $q(x, y)$ defines a unique branch, $y(z) = \frac{1}{1-z}$. Then using also (22):

$$\sigma \;=\; \mathcal{E}^{-1}\left[\frac{1}{1-z}\right] = \mathsf{fact} \odot \mathcal{G}^{-1}\left[\frac{1}{1-z}\right] = \mathsf{fact} \odot (1, 1, 1, ...) = (0!, 1!, 2!, ...)\,.$$

Finally, by Corollary 5.6, the solution of (24) as an ODE with the initial condition $y(0) = 1$ is the unique branch of q, that is $y(z) = \frac{1}{1-z}$.

▶ **Example 5.8** (double factorials, again)**.** Consider again the equation $\dot{y} = y^3$ with $y(0) = 1$ of Example 4.3, and the equivalence $\mu_\otimes(q) = 0$, for $q(x, y) := y^2(x - 1/2) + 1/2$, we proved there. Let $\sigma = \mu_\otimes(y)$. According to Proposition 5.2, the exponential generating function $\mathcal{E}[\sigma](z)$ is a branch of $q(x, y)$. Now $q(x, y)$ has two branches, which are obtained by solving for y the corresponding quadratic equation. Of these, $y(z) = \sqrt{1/(1 - 2z)}$ solves the ODE and, by Proposition 5.5, is the exponential generating function of σ.

Let us also point out an interesting interplay between $\times$ and $\otimes$, that may ease compositional reasoning on streams. Depending on the equations at hand, the convolution of two streams might be more easily understood and described than their shuffle product; or a stream can be better understood in terms of the solution of an ODE. The following equality, that can be readily checked, allows one to transform convolution into stream product, and back. We let $\mathsf{fact}^{-1} := (1/0!, 1/1!, ..., 1/j!, ...)$.

$$\mathsf{fact}^{-1} \odot (\sigma \otimes \tau) = (\mathsf{fact}^{-1} \odot \sigma) \times (\mathsf{fact}^{-1} \odot \tau)\,. \tag{25}$$

We illustrate this idea with a simple example.

▶ **Example 5.9** (harmonic numbers)**.** Consider the system of two equations $\dot{y} = y^2$, $\dot{w} = y$ with initial conditions $y(0) = 1$ and $w(0) = 0$. We want to analyze this system in terms of $\otimes$. In Example 5.7, we have seen that $y(z) = \frac{1}{1-z}$ and that $\sigma := \mu_\otimes(y) = \mathsf{fact}$. We can obtain $\mu_\otimes(w)$ via Proposition 5.5 and (22), after solving the second ODE: $w(z) = \int_0^z y(u)du = \ln(\frac{1}{1-z})$, hence $\tau := \mu_\otimes(w) = \mathcal{E}^{-1}[w(z)] = \mathsf{fact} \odot (0, 1, 1/2, ..., 1/j, ...)$. To understand what $\mu_\otimes(yw) = \mu_\otimes(y) \otimes \mu_\otimes(w)$ represents, it is convenient to switch to the convolution product, by applying (25). We have

$$\begin{aligned}\mathsf{fact}^{-1} \odot \mu_\otimes(yw) &= \mathsf{fact}^{-1} \odot (\sigma \otimes \tau) = (\mathsf{fact}^{-1} \odot \sigma) \times (\mathsf{fact}^{-1} \odot \tau) \\ &= (1, 1, 1, ...) \times (0, 1, 1/2, ..., 1/j, ...) = (0, 1, 3/2, ..., \sum_{i=1}^{j} \frac{1}{i}, ...)\end{aligned}$$

which is the sequence $\alpha = (h_0, h_1, ...)$ of the harmonic numbers. Therefore $\mu_\otimes(yw) = \mathsf{fact} \odot \alpha$, and $y(z)w(z) = \frac{1}{1-z}\ln(\frac{1}{1-z}) = \mathcal{E}[\mathsf{fact} \odot \alpha](z) = \sum_{j\geq 0} h_j z^j = \mathcal{G}[\alpha](z)$ is the ordinary generating function of the harmonic numbers.

Another example of interplay between the two products arises in connection with the solutions of linear ODEs and Laplace transform; this is elaborated in the full version of the present article [11].

▶ Remark 5.10. One would like to prove for $\mu_\otimes$ a result analogous to Corollary 5.3. In this respect, let us first note that $\mu_\otimes(p)$ need not be algebraic: as we have seen in Example 5.4, $\mu_\otimes(y) = \mathsf{fact} = (0!, 1!, 2!, ...)$, which is not an algebraic stream – cf. [22], or simply note that $\mathcal{G}[\mathsf{fact}](z)$ is not analytic. The next natural candidate class to consider for inclusion is that of streams with a *holonomic* (a.k.a. *D-finite*) ordinary generating function [22]: that is, a function $y(z)$ satisfying a linear differential equation with polynomial coefficients in z. This class includes strictly algebraic streams, but $\mu_\otimes(p)$ need not be holonomic either. To see this, consider the single ODE $\dot{f} = 1 + f^2$ with $f(0) = 0$, which defines the trigonometric tangent function: $f(z) = \tan(z)$. It is known that $\sigma = \mathcal{G}^{-1}[\tan(z)]$ is not holonomic, see e.g. [18, Ch.1]. It is also known that fact is holonomic, and that the class of holonomic functions is closed under the Hadamard product [22]. Now, from Proposition 5.5 and (22), we have that: $\mu_\otimes(f) = \mathcal{E}^{-1}[\tan(z)] = \mathsf{fact} \odot \sigma$. This equality implies that $\mu_\otimes(f)$ is not holonomic, because otherwise σ would be as well. At present, we also ignore if algebraic and/or holonomic streams are included in streams obtainable via $\mu_\otimes$.

6 Conclusion

We have studied connections between polynomials, differential equations and streams, in terms of algebra and coalgebra. Our main result shows that, given any stream product that satisfies certain reasonable assumptions, there is a way to define a transition function on polynomials such that the induced unique coalgebra morphism into streams is a $\mathbb{K}$-algebra homomorphism – and vice-versa. We have applied this result to the design of a decision algorithm for polynomial stream equivalence, and to reasoning on generating functions and ordinary differential equations.

As for future work, it would be interesting to see whether we can define new notions of products that respect the format we devised in this paper. Somewhat orthogonal to this, the relation of our framework with bialgebras [16] deserves further investigation. Finally, in the field of nonlinear dynamical systems [15], convolution of discrete sequences arises as a means to describe the composition of distinct signals or subsystems (e.g., a plant and a controller); we would like to understand if our approach can be useful to reason on such systems as well.

References

1 The on-line encyclopedia of integer sequences. URL: https://oeis.org.

2 Lars V. Ahlfors. *Complex Analysis: An Introduction to the Theory of Analytic Functions of One Complex Variable (3rd edition)*. Mc Graw Hill, 1979.

3 Henning Basold, Marcello M. Bonsangue, Helle Hvid Hansen, and Jan Rutten. (co)algebraic characterizations of signal flow graphs. In *Horizons of the Mind. A Tribute to Prakash Panangaden - Essays Dedicated to Prakash Panangaden on the Occasion of His 60th Birthday*, volume 8464 of *LNCS*, pages 124–145. Springer, 2014. doi:10.1007/978-3-319-06880-0_6.

4 Henning Basold, Helle Hvid Hansen, Jean-Éric Pin, and Jan Rutten. Newton series, coinductively: a comparative study of composition. *Math. Struct. Comput. Sci.*, 29(1):38–66, 2019. doi:10.1017/S0960129517000159.

5 Filippo Bonchi, Marcello M. Bonsangue, Michele Boreale, Jan J. M. M. Rutten, and Alexandra Silva. A coalgebraic perspective on linear weighted automata. *Inf. Comput.*, 211:77–105, 2012. doi:10.1016/j.ic.2011.12.002.

6 Filippo Bonchi, Daniela Petrisan, Damien Pous, and Jurriaan Rot. Coinduction up-to in a fibrational setting. In *Proc. of CSL-LICS*, pages 20:1–20:9. ACM, 2014. doi:10.1145/2603088.2603149.

7 Michele Boreale. Algebra, coalgebra, and minimization in polynomial differential equations. *Log. Methods Comput. Sci.*, 15(1), 2019. doi:10.23638/LMCS-15(1:14)2019.

8 Michele Boreale. On the Coalgebra of Partial Differential Equations. In *Proc. of MFCS*, volume 138 of *LIPIcs*, pages 24:1–24:13. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2019. doi:10.4230/LIPIcs.MFCS.2019.24.

9 Michele Boreale. Automatic pre- and postconditions for partial differential equations. In Marco Gribaudo, David N. Jansen, and Anne Remke, editors, *Proc. of QEST*, volume 12289 of *LNCS*, pages 193–210. Springer, 2020. doi:10.1007/978-3-030-59854-9_15.

10 Michele Boreale. Complete algorithms for algebraic strongest postconditions and weakest preconditions in polynomial odes. *Sci. Comput. Program.*, 193:102441, 2020. doi:10.1016/j.scico.2020.102441.

11 Michele Boreale and Daniele Gorla. Algebra and coalgebra of stream products. Full version of this work, available at *CoRR*, arXiv:2107.04455, 2021.

12 D. Cox, J. Little, and D. O'Shea. *Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra*. Undergraduate Texts in Mathematics. Springer, 2007.

13 Philippe Flajolet and Robert Sedgewick. Analytic combinatorics: functional equations, rational and algebraic functions. Research Report RR-4103, INRIA, 2001. URL: https://hal.inria.fr/inria-00072528.

14 Helle Hvid Hansen, Clemens Kupke, and Jan Rutten. Stream differential equations: Specification formats and solution methods. *Log. Methods Comput. Sci.*, 13(1), 2017. doi:10.23638/LMCS-13(1:3)2017.

15 Hassan K. Khalil. *Nonlinear Systems (3rd ed.)*. Prentice Hall, 2002.

16 Bartek Klin. Bialgebras for structural operational semantics: An introduction. *Theoretical Computer Science*, 412(38):5043–5069, 2011. doi:10.1016/j.tcs.2011.03.023.

17 W. Kuich and A. Salomaa. *Semirings, Automata, Languages*. Monographs in Theoretical Computer Science: An EATCS Series. Springer, 1986.

18 Christian Mallinger. *Algorithmic manipulations and transformations of univariate holonomic functions and sequences*. Diplomarbeit, Johannes Kepler Universität Linz, 1996.

19 Dusko Pavlovic and M. Escardó. Calculus in coinductive form. In *Proc. of LICS*, pages 408–417. IEEE, 1998.

20 Jan J. M. M. Rutten. Behavioural differential equations: a coinductive calculus of streams, automata, and power series. *Theor. Comput. Sci.*, 308(1-3):1–53, 2003. doi:10.1016/S0304-3975(02)00895-2.

21 Jan J. M. M. Rutten. A coinductive calculus of streams. *Math. Struct. Comput. Sci.*, 15(1):93–147, 2005. doi:10.1017/S0960129504004517.

22 Richard P. Stanley. *Enumerative Combinatorics, 2nd edition*. CUP, 2012.

23 Joost Winter. *Coalgebraic Characterizations of Automata-Theoretic Classes*. PhD thesis, Radboud Universiteit Nijmegen, 2014.