The commutativity problem for effective varieties of formal series, and applications

Lorenzo Clemente
Department of Informatics
University of Warsaw
Warszawa, Poland

***Abstract*—A *formal series* in noncommuting variables Σ over the rationals is a mapping $\Sigma^* \to \mathbb{Q}$ from the free monoid generated by Σ to the rationals. We say that a series is *commutative* if the value in the output does not depend on the order of the symbols in the input. The *commutativity problem* for a class of series takes as input a (finite presentation of) a series from the class and amounts to establishing whether it is commutative. This is a very natural, albeit nontrivial problem, which has not been considered before from an algorithmic perspective.**

We show that commutativity is decidable for all classes of series that constitute a so-called *effective prevariety*, a notion generalising Reutenauer's varieties of formal series. For example, the class of *rational series*, introduced by Schützenberger in the 1960's, is well-known to be an effective (pre)variety, and thus commutativity is decidable for it.

In order to showcase the applicability of our result, we consider classes of formal series generalising the rational ones. We consider *polynomial automata*, *shuffle automata*, and *infiltration automata*, and we show that each of these models recognises an effective prevariety of formal series. Consequently, their commutativity problem is decidable, which is a novel result. We find it remarkable that commutativity can be decided in a uniform way for such disparate computation models.

Finally, we present applications of commutativity outside the theory of formal series. We show that we can decide *solvability* in sequences and in power series for restricted classes of algebraic difference and differential equations, for which such problems are undecidable in full generality. Thanks to this, we can prove that the syntaxes of multivariate *polynomial recursive sequences* and of *constructible differentially algebraic power series* are effective, which are new results which were left open in previous work.

***Index Terms*—formal series, commutativity, weighted automata, number sequences, differential equations**

Introduction

A *univariate formal series* over the rationals is a function $\mathbb{N} \to \mathbb{Q}$ (sometimes written $\mathbb{Q}[[x]]$). They arise in a variety of disciplines, e.g., Taylor series in mathematical analysis [65], generating functions in combinatorics [68], [69], and Z-transforms in engineering [57]. The denomination "formal" highlights the algebraic aspect, rather than the analytic one; for brevity, we will refer to formal series simply as series.

The theory of univariate series can be developed in several directions. One direction is more popular in mathematics, and considers *multivariate series in commuting indeterminates* $X = \{x_1, \ldots, x_d\}$, which are functions of the form $\mathbb{N}^d \to \mathbb{Q}$ for some $d \in \mathbb{N}_{\geq 1}$ (sometimes written $\mathbb{Q}[[X]]$) [59]. Another direction is more popular in theoretical computer science, and considers *multivariate series in noncommuting indeterminates* $\Sigma = \{a_1, \ldots, a_d\}$, which are functions of the form $\Sigma^* \to \mathbb{Q}$ (sometimes called *weighted languages* and written $\mathbb{Q}\langle\langle\Sigma\rangle\rangle$) [9]. In fact, noncommuting indeterminates are more general, since we can recover commuting indeterminates as a special case: Namely, as those series $f \in \mathbb{Q}\langle\langle\Sigma\rangle\rangle$ which are *commutative*, i.e., $f(u) = f(v)$ for all words $u, v \in \Sigma^*$ which are permutations of each other. This is the unifying view that we shall adopt.

In the context of formal languages and automata theory, series in noncommuting indeterminates have been introduced by Schützenberger, who considered the class of *rational series* [66] (cf. [9] for an extensive introduction). This class has since appeared in various areas of computer science and mathematics [25], such as speech recognition [54], digital image compression [25, Part IV], learning [2], quantum computing [10], as well as in Bell and Smertnig's resolution [5] of a conjecture of Reutenauer [62].

Series have also been applied to control theory, as put forward by the pioneering work of Fliess [32]. Namely, he associated to a control system its *characteristic series*, in such a way that the behaviour of the system as an input-output transformer is uniquely determined by it [32, Lemme II.1]. He then recovered the rational series as the characteristic series of the so-called *bilinear systems* [31]. This is an extensively studied class [26], whose theory can be derived from well-known results for rational series (e.g., decidability of equivalence checking and existence of minimal realisations).

It is in relation with control theory that commutativity naturally enters the picture. Fliess isolated an important subclass of bilinear systems, namely those where the output at time t depends only on t and the inputs at time t (but not at times $< t$), and proved that their characteristic series coincide with the *commutative* rational series [31, Proposition 3.6].

Commutativity, besides arguably being a fundamental algebraic notion per se, also appears in a number of other works. For instance, it is used as a simplifying assumption in Karhumäki's study on $\mathbb{N}$-*rational series* [40] (later corrected in [49]), it is featured in Litow's short note on the undecidability of the multivariate *Skolem problem* [48], in the study of *DT0L systems* [41], in quantum finite automata [37], and in the formal verification of MapReduce frameworks [17]. Under

Supported by the ERC grant INFSYS, agreement no. 950398.

the assumption of commutativity, an elementary upper bound for the computation of *algebraic invariants* of weighted finite automata has been recently obtained [56] (the complexity of the general problem being open [35], [36]).

All these works leveraging commutativity offer no indication as whether one can decide it algorithmically. Moreover, most decision problems for rational series are undecidable e.g., *containment* and *nonemptiness* [55, Theorem 21] (cf. also [58, Theorem 6.17]), therefore finding problems which are decidable for the whole class is challenging.

Contributions

We study the *commutativity problem* for classes of series from a computational perspective. For rational series, thanks to strong structural properties [29, Proposition 3.1] commutativity reduces to whether the linear maps in a *minimal linear representation* commute pairwise. Thanks to Schützenberger's seminal result, such minimal representations can be computed efficiently [66, III.B.1.], and thus we can check commutativity in polynomial time by a direct numerical computation. This is in line with the fact that decidable problems on rational series often rely on minimisation in an essential way [4], [43], [38]. For more general classes where minimisation is not available, commutativity is an open problem.

First contribution: We show that commutativity is decidable for *effective prevarieties* of series, a generalisation of Reutenauer's *varieties* [63, Sec. III.1]. *Prevarieties* are classes of series closed under linear combinations, right, and left derivatives (cf. § I). *Effectiveness* demands that series in the class can be manipulated by algorithms working on finite representations. This achieves a delicate balance: It is sufficiently strong to ensure decidability of commutativity, but sufficiently weak to be applicable to several classes of series not known to have minimal representations.

Theorem 1. *Let $\mathcal{S}$ be an effective prevariety of series. The commutativity problem for $\mathcal{S}$ is decidable.*

This gives us a uniform framework in which many classes of series can be shown to have decidable commutativity. For example, rational series are effective (pre)varieties [63, Theorem III.1.1], giving an alternative decidability proof.

Second contribution: We apply Theorem 1 to decide commutativity for three, mutually incomparable classes of series, generalising the rational ones. Namely, we consider series recognised by *polynomial automata* [6] (§ III), *shuffle automata* [18] (§ IV), and *infiltration automata* (§ V-A). Polynomial automata (also known as *cost register automata* over the rationals [1]) and shuffle automata are known to have a decidable equality problem [6], [18], however they have very different computational properties, e.g., Ackermannian for the former while elementary for the latter. They were not known to be prevarieties. The notion of infiltration automata is novel, as well as decidability of their equality problem. Thanks to the notion of effective prevariety, we can decide commutativity for these classes in a common framework, which we find remarkable. As far as we know, no other natural problem beyond equality was known to be decidable for these classes.

Theorem 2. *Polynomial automata, shuffle automata, and infiltration automata recognise effective prevarieties. As a consequence, the commutativity problem is decidable for them.*

We reduce commutativity to checking finitely many equalities (Lemma 2). In turn, we show that equality reduces to commutativity, yielding a complete picture.

Example 1. As an application of Theorem 2, consider a polynomial automaton over alphabet $\Sigma = \{a_1, a_2\}$ with one register r. Initially, the register starts at some rational value $r := c \in \mathbb{Q}$. Upon reading a_1, it is updated with the rule $r := r^2$ and upon reading a_2 with $r := 1 - r^2$. After reading the input, the automaton outputs the value of r. For instance, reading $a_1 a_2$, the automaton outputs $1 - c^4$, while over $a_2 a_1$ it outputs $(1 - c^2)^2$. For $c = -1$, these two outputs coincide, and in fact our algorithms determines that the semantics of the automaton is commutative for this initial value. For other values, e.g., $c = 2$, the semantics is not commutative: $a_1 a_2 \mapsto 1 - 2^4 = -15$, but $a_2 a_1 \mapsto (1 - 2^2)^2 = 9$.

Third contribution: We apply Theorem 2 to two problems on series in commuting variables. We find it intriguing that we can solve problems about series in commuting variables using an algorithmic result in the theory of noncommuting variables. This corroborates the hypothesis that the latter are a more fundamental object than the former.

a) Multivariate polyrec sequences: In the first problem, we propose a notion of *multivariate polynomial recursive sequences* (*polyrec*), which are a subset of $\mathbb{N}^d \to \mathbb{Q}$ generalising the univariate polyrec sequences from [15]. Multivariate polyrec sequences are defined via systems of *polynomial difference equations* of a certain form (cf. § III-F for a precise definition). However, such systems may not have any solution at all, and it is not clear whether this can be decided. Solvability of polynomial difference equations is in general undecidable (Remark 6), which poses an obstacle to the development of a computational theory of multivariate polyrec sequences. We overcome this difficulty by showing that commutativity of polynomial automata can be used to decide whether polyrec systems are solvable (Theorem 5). Thus, the syntax of multivariate polyrec sequences is effective.

Example 2. Consider the bivariate polyrec recursions $f(n_1 + 1, n_2) := f(n_1, n_2)^2$ and $f(n_1, n_2 + 1) := 1 - f(n_1, n_2)^2$, for all $n_1, n_2 \in \mathbb{N}$. As shown in Figure 1, $f(2, 2)$ can be computed in six possible ways, all of which must give the same value. By reducing to commutativity for the polynomial automaton of Example 1, our algorithm determines the existence of a sequence solution $f : \mathbb{N}^2 \to \mathbb{Q}$ starting at $f(0, 0) = -1$,

b) Multivariate CDA *series:* In the second problem, we consider the notion of *constructible differentially algebraic* (CDA) series in commuting variables [7]. They were introduced in the multivariate context by Bergeron and Sattler [8], as solutions of systems of *polynomial differential equations* of a restricted form (cf. § IV-C for details). Such systems may

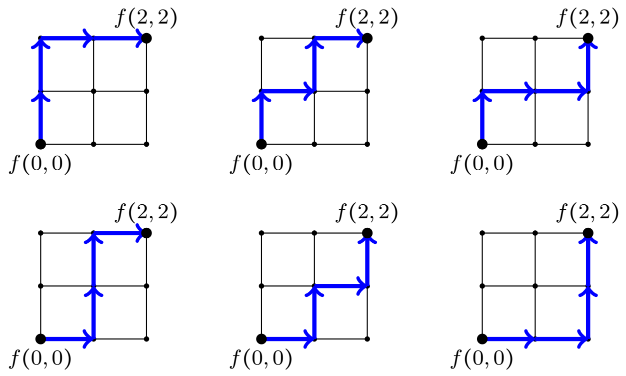

Fig. 1. The six constraints on the computation of $f(2,2)$.

not have solutions and deciding whether this is the case is open [18, Remark 27]. Solvability of polynomial differential equations is undecidable by a result of Denef and Lipshitz [24, Theorem 4.11] (Remark 8), and thus it is not clear whether CDA series have a decidable syntax. We overcome this issue by showing that commutativity of shuffle automata can be used to decide whether CDA systems are solvable (Theorem 8), thus answering the open question positively.

Organisation: In the next section § I we present some necessary preliminaries, followed in § II by a proof of Theorem 1. Polynomial automata will be studied in § III, followed by shuffle automata in § IV. In § V we discuss infiltration automata and other extensions. More details about infiltration automata, and full proofs, can be found in the appendix.

I. Preliminaries

We say that a structure (such as a class of series, a class of languages, an algebra, etc...) is *effective* if its elements can be finitely represented (with a decidable syntax), equality between elements can be decided based on their representations, and all the structure operations can be carried over algorithmically. For instance the class of regular languages is effective. Most results in the paper hold for any field, however for computability considerations we restrict our presentation to $\mathbb{Q}$, which is an effective field. We use f, g, h to denote series in $\mathbb{Q}\langle\langle\Sigma\rangle\rangle$, and we write a series f as $\sum_{w\in\Sigma^*} f_w \cdot w$, where the value of f at w is $f_w \in \mathbb{Q}$. We also write $[w]f$ for f_w. The *support* of a series f is the set of words w such that $f_w \neq 0$. A *polynomial* is a series with finite support. Thus, $3aab-\frac{5}{2}bc$ is a polynomial and $1+a+a^2+\cdots$ is a series. The set of series is equipped with a variety of operations. It carries the structure of a vector space, with *zero* $\mathbb{0}$, *scalar multiplication* $c \cdot f$ with $c \in \mathbb{Q}$, and *addition* $f+g$ defined element-wise by $[w]\mathbb{0} := 0$, $[w](c \cdot f) := c \cdot [w]f$, and $[w](f+g) := [w]f + [w]g$, for every $w \in \Sigma^*$. This vector space is equipped with two important families of linear maps, called *left* and *right derivatives* $\delta_a^L, \delta_a^R : \mathbb{Q}\langle\langle\Sigma\rangle\rangle \to \mathbb{Q}\langle\langle\Sigma\rangle\rangle$, for every $a \in \Sigma$. They are the series analogues of language quotients: For every series $f \in \mathbb{Q}\langle\langle\Sigma\rangle\rangle$, they are defined by $[w]\delta_a^L f := [a \cdot w]f$, resp., $[w]\delta_a^R f := [w \cdot a]f$, for every $w \in \Sigma^*$. The following elementary observation will be very useful.

Lemma 1 (Left and right derivatives commute)**.** *For every two letters $a, b \in \Sigma$, we have $\delta_a^L \circ \delta_b^R = \delta_b^R \circ \delta_a^L$.*

(By $\delta_a^L \circ \delta_b^R$ we mean the *composition* of the two derivatives, which is the function mapping a series f to the series $\delta_a^L \delta_b^R f$. We avoid writing parenthesis for function application when possible.) We will later equip series with several multiplication operations, turning them into algebras.

Let $\mathcal{S}$ be a class of series, and for an alphabet Σ let $\mathcal{S}_\Sigma$ the subset of $\mathcal{S}$ consisting of series over input alphabet Σ. A class of series $\mathcal{S}$ is a *prevariety* if the following two conditions hold, for every alphabet Σ:

(V.1) $\mathcal{S}_\Sigma$ is a vector space over $\mathbb{Q}$.
(V.2) For every series $f \in \mathcal{S}_\Sigma$ and letter $a \in \Sigma$, the series $\delta_a^L f$ and $\delta_a^R f$ are in $\mathcal{S}_\Sigma$.

(A *variety*, as introduced by Reutenauer, is a prevariety which satisfies an additional condition (cf. [63, Sec. III.1]); we will not need this notion in the rest of the paper.) A class of series $\mathcal{S}$ is an *effective prevariety* if it is a prevariety, and moreover

1) every series in $\mathcal{S}$ admits a finite presentation,
2) the closure properties **(V.1)** and **(V.2)** can be carried over algorithmically by manipulating finite presentations, and
3) the equality problem is decidable for series in $\mathcal{S}$.

By *finite presentation* for series we mean a concrete syntax in the same sense, e.g., finite automata are finite presentations for regular languages. Thanks to the equivalence $f = g \iff f - g = \mathbb{0}$ and **(V.1)**, the notion of effective prevariety does not change if we replace the third condition above with

3') the zeroness problem is decidable for series in $\mathcal{S}$.

For instance, the class of rational series is an effective prevariety [63, Sec. III.2.b]: A rational series is finitely represented by a finite weighted automaton, closure under the vector space operations and both derivatives can be carried over by algorithms manipulating automata, and the equality problem is decidable [66].

II. Commutativity problem

We introduce the main problem that we study. The *commutative image* of a word $w \in \Sigma^*$ (sometimes called *Parikh image*) over a totally ordered alphabet $\Sigma = \{a_1, \ldots, a_d\}$ is the vector $\#w := (\#_{a_1}w, \ldots, \#_{a_d}w) \in \mathbb{N}^d$, where $\#_{a_j}w$ is the number of occurrences of a_j in w. For two words u, v we write $u \sim v$ if they have the same commutative image, i.e., one word can be obtained from the other by permuting the positions of letters. For instance, $a_1a_1a_2 \sim a_1a_2a_1 \sim a_2a_1a_1$, since the commutative image of all three words is $(2,1) \in \mathbb{N}^2$, but $a_1 \not\sim a_2$. A series $f \in \mathbb{Q}\langle\langle\Sigma\rangle\rangle$ is *commutative* (called *échangeable* in [32]) if for every two words $u, v \in \Sigma^*$ with the same commutative image $u \sim v$, we have $f_u = f_v$. Let $\mathbb{Q}\langle\langle\Sigma\rangle\rangle^{\mathsf{comm}}$ be the set of commutative series. It is easy to check that all operations defined in this paper preserve commutativity.

On the face of it, commutativity demands to check infinitely many equalities. Our simple, but crucial observation is that it can be characterised by finitely many equations. Clearly, if f is commutative, then for every input symbols $a, b \in \Sigma$ we have

$$\delta_a^L \delta_b^L f = \delta_b^L \delta_a^L f, \quad \text{and} \qquad \text{(swap)}$$
$$\delta_a^L f = \delta_a^R f. \qquad \text{(rotate)}$$

Indeed, the first equation corresponds to exchanging the first two input letters $abw \sim baw$, and the second equation corresponds to a rotation $aw \sim wa$. Since swaps and rotations suffice to generate all commutatively equivalent words, commutativity admits the following characterisation.

Lemma 2 (Characterisation of commutativity)**.** *A series $f \in \mathbb{Q}\langle\langle\Sigma\rangle\rangle$ is commutative if, and only if, it satisfies equations* (swap) *and* (rotate)*, for every $a, b \in \Sigma$.*

Thanks to Lemma 2, commutativity is decidable for effective prevarieties of series. This simple but fruitful observation will allow us to decide commutativity for several classes of series for which the commutativity problem appears to be nontrivial.

Theorem 1. *Let $\mathcal{S}$ be an effective prevariety of series. The commutativity problem for $\mathcal{S}$ is decidable.*

Proof. Thanks to Lemma 2, commutativity reduces to finitely many equality tests (swap) and (rotate). Since $\mathcal{S}$ is a prevariety, if $f \in \mathcal{S}$ then $\delta_a^L f, \delta_a^R f \in \mathcal{S}$ as well. By effectiveness, we can construct finite representations for these series and decide the equality tests. □

Remark 1. Theorem 1 allows us to reduce commutativity to finitely many equality tests. The observation that commutativity can be characterised by finitely many equations has been exploited before [17, Prop. 2], albeit in a different context.

Thanks to the following observation, commutativity generalises equality in all our cases.

Lemma 3. *For every $f \in \mathbb{Q}\langle\langle\Sigma\rangle\rangle$ and fresh input symbols $a, b \notin \Sigma$, the series $ab \sqcup\!\sqcup f$ is commutative if, and only if, $f = \mathbb{0}$.*

(The expression "$ab \sqcup\!\sqcup f$" denotes the *shuffle product* of ab and f, which will be formally introduced in § IV-A2.) In § III and § IV we leverage Theorem 1 for two expressive classes of series, together with respective applications. Extensions will be mentioned in the last section § V.

III. Hadamard automata and series

In this section, we define a weighted model of computation called *Hadamard automata*, strongly related to polynomial automata. We show that the series they recognise constitute an effective prevariety (Theorem 3), and thus the commutativity problem is decidable for this model (Theorem 4). We conclude the section by an application to a class of multivariate number sequences (Theorem 5).

A. Preliminaries

1) Difference algebra: Let $\mathbb{A} = (A; +, *)$ be an *algebra* (over the field of rational numbers), that is a vector space equipped with a bilinear product [13, Ch. 3, §1]. An algebra *endomorphism* is a linear function $F : A \to A$ s.t.

$$F(\alpha * \beta) = F\alpha * F\beta, \quad \text{for all } \alpha, \beta \in A. \qquad (1)$$

A *difference algebra* [22], [46] is an algebra $(A; +, *, (F_j)_{1 \le j \le d})$ together with finitely many endomorphisms $F_1, \dots, F_d : A \to A$. We do not require the F_i's to pairwise commute. Difference algebras will provide the semantic background for the rest of the section.

As an example, consider the algebra of multivariate polynomials $(\mathbb{Q}[X]; +, \cdot)$, where $X = (X_1, \dots, X_k)$ is a tuple of commuting variables. When the variable names do not matter, we sometimes write $\mathbb{Q}[k]$. An endomorphism F acts on a polynomial $\alpha \in \mathbb{Q}[X]$ just by evaluation $F\alpha = \alpha(F(X_1), \dots, F(X_k))$, and thus it is uniquely defined once we have fixed $F(X_1), \dots, F(X_k) \in \mathbb{Q}[X]$. If we fix distinguished endomorphisms $F_1, \dots, F_d$, then the algebra of polynomials acquires the structure of a difference algebra $(\mathbb{Q}[X]; +, \cdot, (F_j)_{1 \le j \le d})$. Other examples of difference algebras will be presented in the next sections § III-A2 and § III-F1.

2) Hadamard difference algebra: We define a product operation on series which will be central for the understanding of Hadamard automata. The *Hadamard product* [30] of two series $f, g \in \mathbb{Q}\langle\langle\Sigma\rangle\rangle$, denoted by $f \odot g$, is defined element-wise: For every $w \in \Sigma^*$, $[w](f \odot g) := [w]f \cdot [w]g$. For instance, if we have two polynomials $2aab - c$ and $3aab + b$ (in noncommuting variables), their Hadamard product is $6aab$. This is an associative and commutative operation, it distributes over sum, and the identity element is the series $\mathbb{1} = \sum_{w \in \Sigma} 1 \cdot w$ mapping every word to 1. The Hadamard product for series is reminiscent of the notion of *intersection* in language theory and of *fully synchronous composition* in concurrency theory.

In order to make it easier to compare the Hadamard product to the shuffle and infiltration products studied later, we find it convenient to provide the following alternative, coinductive definition (cf. [3, Definition 8.1]): For any two series $f, g \in \mathbb{Q}\langle\langle\Sigma\rangle\rangle$, their Hadamard product $f \odot g$ is the unique series s.t.

$$[\varepsilon](f \odot g) = f_\varepsilon \cdot g_\varepsilon, \qquad (\odot\text{-}\varepsilon)$$
$$\delta_a^L(f \odot g) = \delta_a^L f \odot \delta_a^L g, \ \forall a \in \Sigma. \qquad (\odot\text{-}\delta_a^L)$$

Equation ($\odot$-δ_a^L) states that left derivatives are endomorphisms w.r.t. the Hadamard product. We thus obtain the *Hadamard difference algebra* of series $\mathbb{Q}\langle\langle\Sigma\rangle\rangle_\odot := \left(\mathbb{Q}\langle\langle\Sigma\rangle\rangle; +, \odot, (\delta_a^L)_{a \in \Sigma}\right)$, which will provide the semantic background for the forthcoming Hadamard automata. By dropping the difference structure, we obtain the *Hadamard algebra* $(\mathbb{Q}\langle\langle\Sigma\rangle\rangle; +, \odot)$. At this point, it is worth noting that right derivatives are also Hadamard algebra endomorphisms. This will be useful in the proof of Lemma 8.

Lemma 4. *Right derivatives δ_a^R (with $a \in \Sigma$) are Hadamard algebra endomorphisms. I.e., they are linear and commute with Hadamard product,*

$$\delta_a^R(f \odot g) = \delta_a^R f \odot \delta_a^R g, \quad \forall a \in \Sigma. \qquad (2)$$

B. Hadamard automata

We can now define the central computation model of the section. A *Hadamard automaton* is a tuple $A = (\Sigma, X, F, \Delta)$ where Σ is a finite *input alphabet*, $X = \{X_1, \dots X_k\}$ is a finite set of *nonterminal symbols*, $F : X \to \mathbb{Q}$ is the *output function*, and $\Delta : \Sigma \to X \to \mathbb{Q}[X]$ is the *transition function*. A *configuration* of a Hadamard automaton is a polynomial $\alpha \in \mathbb{Q}[X]$. For every input symbol $a \in \Sigma$, the transition function $\Delta_a : X \to \mathbb{Q}[X]$ is extended to a unique endomorphism $\Delta_a : \mathbb{Q}[X] \to \mathbb{Q}[X]$ of the polynomial algebra of configurations. In other words, Δ_a acts by evaluation: $\Delta_a(\alpha) = \alpha(\Delta_a(X_1), \dots, \Delta_a(X_k))$. Finally, transitions from single letters are extended to all finite input words homomorphically: For every configuration $\alpha \in \mathbb{Q}[X]$, input word $w \in \Sigma^*$, and letter $a \in \Sigma$, we have $\Delta_\varepsilon \alpha := \alpha$ and $\Delta_{a \cdot w} \alpha := \Delta_w \Delta_a \alpha$. We have all ingredients to define the *semantics* of a configuration $\alpha \in \mathbb{Q}[X]$, which is the series $[\![\alpha]\!] \in \mathbb{Q}\langle\langle\Sigma\rangle\rangle$ s.t.:

$$[\![\alpha]\!]_w := F \Delta_w \alpha, \quad \text{for all } w \in \Sigma^*. \tag{3}$$

Here, F is extended from nonterminals to configurations also endomorphically, by $F\beta = \beta(FX_1, \dots, FX_k)$. The semantics of a Hadamard automaton A is the series recognised by nonterminal X_1.

Remark 2. When the transition function is *linear* (i.e., $\Delta_a X_i$ is a polynomial of degree ≤ 1 for all $a \in \Delta$ and $1 \le i \le k$), we recover Schützenberger's *weighted finite automata* [66]. Thus Hadamard automata generalise weighted automata.

Example 3. Revisiting Example 1 in the syntax of Hadamard automata, we have a single nonterminal $X = \{A\}$, output function F_c defined by $F_c A := c$ ($c \in \mathbb{Q}$) and transitions by

$$\Delta_{a_1} A := A^2 \quad \text{and} \quad \Delta_{a_2} A := 1 - A^2.$$

For instance, when $c = 0$ starting from the initial configuration A and reading input $u = a_1 a_2$ we obtain configuration $\Delta_{a_2} \Delta_{a_1} A = \Delta_{a_2} A^2 = (1 - A^2)^2$ and thus $[\![A]\!]_u = 1$. Reading $v = a_2 a_1$ yields a different configuration $\Delta_{a_1} \Delta_{a_2} A = 1 - A^4$, however $[\![A]\!]_v = 1$. In fact, $[\![A]\!]_w$ is the count modulo 2 of the number of a_2's in w, and thus $[\![A]\!]$ is a commutative series.

It can be checked that also the output functions F_1, F_{-1} defined by $F_1 A := 1$, resp., $F_{-1} A := -1$ give rise to commutative series. However, no other choice does. For instance, F_2 defined by $F_2 A := 2$ does not yield a commutative series, since $[\![A]\!]_{a_1 a_2} = (1 - 2^2)^2 = 9$, but $[\![A]\!]_{a_2 a_1} = 1 - 2^4 = -15$.

A Hadamard automaton endows the algebra of configurations with the structure of a difference algebra $(\mathbb{Q}[X]; +, \cdot, (\Delta_a)_{a \in \Sigma})$. This allows us to connect the difference algebra of configurations with the difference algebra of Hadamard series they recognise.

Lemma 5 (Properties of the semantics)**.** *The semantics of a Hadamard automaton is a* homomorphism *from the difference algebra of polynomials* $(\mathbb{Q}[X]; +, \cdot, (\Delta_a)_{a \in \Sigma})$ *to the difference Hadamard algebra of series* $\left(\mathbb{Q}\langle\langle\Sigma\rangle\rangle; +, \odot, (\delta_a^L)_{a \in \Sigma}\right)$. *In other words,* $[\![0]\!] = \mathbb{0}$, $[\![1]\!] = \mathbb{1}$, *and, for all* $\alpha, \beta \in \mathbb{Q}[X]$,

$$[\![c \cdot \alpha]\!] = c \cdot [\![\alpha]\!] \qquad \forall c \in \mathbb{Q}, \tag{4}$$

$$[\![\alpha + \beta]\!] = [\![\alpha]\!] + [\![\beta]\!], \tag{5}$$

$$[\![\alpha \cdot \beta]\!] = [\![\alpha]\!] \odot [\![\beta]\!], \tag{6}$$

$$[\![\Delta_a \alpha]\!] = \delta_a^L [\![\alpha]\!] \qquad \forall a \in \Sigma. \tag{7}$$

Remark 3 (Hadamard vs. polynomial automata). Hadamard automata are very similar to *polynomial automata* [6], a common generalisation of weighted automata [66] and vector addition systems [53]. The two models are equivalent, up to reversal of the input: A series f is recognised by a Hadamard automaton iff its reversal f^R is recognised by a polynomial automaton. (The *reversal* of f is the series f^R mapping $a_1 \cdots a_n$ to $f(a_n \cdots a_1)$.) Since commutativity is invariant under reversal, our results also apply to polynomial automata.

Over more general commutative semirings, polynomial automata have been introduced as *alternating weighted automata* [44], further studied in [34] (albeit without algorithmic results). In the special case of an input alphabet with just one letter $\Sigma = \{a\}$, Hadamard automata have been considered in [12] in a coalgebraic setting, where equality is shown to be decidable [12, Theorem 4.1] (without complexity analysis).

C. Hadamard-finite series

Hadamard automata provide an operational way to syntactically describe a class of series. In this section, we explore a semantic way, which will give us a very short proof of the effective prevariety property (cf. Theorem 3).

For series $f_1, \dots, f_k \in \mathbb{Q}\langle\langle\Sigma\rangle\rangle$, let $\mathbb{Q}[f_1, \dots, f_k]_\odot$ be the smallest Hadamard algebra containing $f_1, \dots, f_k$. Hadamard algebras of this form are called *finitely generated*, with *generators* $f_1, \dots, f_k$. It is a *difference Hadamard algebra* if it is additionally closed under the endomorphisms δ_a^L, $a \in \Sigma$.

Definition 1 (Hadamard-finite series)**.** A series is *Hadamard finite* if it belongs to a finitely generated difference Hadamard algebra.

The following lemma will be our working definition of Hadamard-finite series.

Lemma 6. *A series f is Hadamard finite iff there are generators $f_1, \dots, f_k$ s.t.:*

1) $f \in \mathbb{Q}[f_1, \dots, f_k]_\odot$.
2) $\delta_a^L f_i \in \mathbb{Q}[f_1, \dots, f_k]_\odot$ *for every* $a \in \Sigma$ *and* $1 \le i \le k$. *I.e., there are polynomials* $p_i^{(a)} \in \mathbb{Q}[k]$ *s.t.*

$$\delta_a^L f_i = p_i^{(a)}(f_1, \dots, f_k), \quad \textit{for all } a \in \Sigma, 1 \le i \le k. \tag{8}$$

Moreover, we can assume without loss of generality that f equals one of the generators $f = f_1$.

We call (8) *systems of Hadamard-finite equations*. The following lemma states that Hadamard automata recognise precisely the Hadamard-finite series. Its proof relies on Lemmas 5 and 6.

Lemma 7. *A series is recognised by a Hadamard automaton if, and only if, it is Hadamard finite.*

Remark 4. Hadamard-finite equations have been considered under the name of *polynomial recurrent relations* by Sénizergues [67, Definition 5], where they are claimed, without proof, to coincide with those recognised by deterministic pushdown automata of level 3 [67, Corollary 3], and to have a decidable equality problem [67, Theorem 5]. In the context of arbitrary semirings, they also appear under the name of *Hadamard-polynomial equations* [44, Sec. 6].

Example 4. We illustrate Lemma 7 on the Hadamard automaton from Example 3. The corresponding Hadamard-finite equations are $\delta^L_{a_1} [\![A]\!] = [\![A]\!]^2$ and $\delta^L_{a_2} [\![A]\!] = \mathbb{1} - [\![A]\!]^2$. (The square operation is to be understood as Hadamard square, i.e., $[\![A]\!]^2 = [\![A]\!] \odot [\![A]\!]$; recall that $\mathbb{1} = \sum_{w \in \Sigma^*} 1 \cdot w$ is the Hadamard identity.) It follows that $\mathbb{Q}[[\![A]\!]]_\odot$ is a difference Hadamard algebra, and thus $[\![A]\!]$ is Hadamard finite.

D. Closure properties of Hadamard-finite series

We recall some basic closure properties of Hadamard-finite series. All of them follow directly from the definitions, except closure under right derivative, which is a novel result.

Lemma 8. *The class of Hadamard-finite series is an effective difference algebra. Moreover, if f is a Hadamard finite, then $\delta^R_a f$ is also effectively Hadamard finite.*

When we say that $\delta^R_a f$ is "effectively Hadamard finite" we mean that there is an algorithm that, given in input a finite presentation for f and an input symbol $a \in \Sigma$, produces in output a finite presentation for $\delta^R_a f$.

Proof. We have to show that Hadamard-finite series contain $\mathbb{0}, \mathbb{1}$ (which is obvious), and are effectively closed under scalar product $c \cdot f$ (with $c \in \mathbb{Q}$), sum $f + g$, Hadamard product $f \odot g$, left and right derivatives $\delta^L_a f, \delta^R_a f$ (with $a \in \Sigma$). Simple manipulations of the generators suffice in each case. We show closure under right derivative, since it is the only property that is not immediate from the definitions. Let $f_1, \dots, f_k$ be generators for f and fix $a \in \Sigma$. We consider new generators $\delta^R_a f_i$, for every $1 \le i \le k$, yielding the finitely generated Hadamard algebra $A := \mathbb{Q}[\delta^R_a f_1, \dots, \delta^R_a f_k]_\odot$. By Lemma 6, the proof is concluded by the following two claims.

Claim 1. $\delta^R_a f \in A$.

Proof of the claim. Since f is in the Hadamard algebra generated by the f_i's and right derivative is an endomorphism by Lemma 4, $\delta^R_a f$ is in the Hadamard algebra generated by the $\delta^R_a f_i$'s, i.e., $\delta^R_a f \in A$. □

Claim 2. *For every $b \in \Sigma$ and $1 \le i \le k$, $\delta^L_b \delta^R_a f_i \in A$.*

Proof of the claim. Since left and right derivatives commute by Lemma 1, we have $\delta^L_b \delta^R_a f_i = \delta^R_a \delta^L_b f_i$. But $\delta^L_b f_i$ is in the Hadamard algebra generated by the f_i's and since right derivative is an endomorphism by Lemma 4, $\delta^R_a \delta^L_b f_i \in A$. □

□

Theorem 3. *The class of Hadamard-finite series is an effective prevariety.*

Proof. The two closure conditions **(V.1)** and **(V.2)** follow from Lemma 8. By Lemma 7 and Remark 3, the equality problem for Hadamard-finite series is (effectively) equivalent to the equality problem for polynomial automata, and the latter is decidable by [6, Corollary 1]. □

We conclude this section by showing that Hadamard-finite series over disjoint alphabets are closed under shuffle product. This is a surprising observation, since it is mixing together products of a very different nature. It will be used in Theorem 4 to provide a hardness result for the commutativity problem.

Lemma 9. *Let Σ, Γ be two finite and disjoint alphabets $\Sigma \cap \Gamma = \varnothing$. If $f \in \mathbb{Q}\langle\langle\Sigma\rangle\rangle$ and $g \in \mathbb{Q}\langle\langle\Gamma\rangle\rangle$ are Hadamard finite, then $f \sqcup\!\sqcup\, g$ is effectively Hadamard finite.*

E. Commutativity problem for Hadamard-finite series

We now have all ingredients to prove our main result for Hadamard-finite series.

Theorem 4. *The commutativity problem for Hadamard-finite series (equivalently, Hadamard and polynomial automata) is Ackermann-complete.*

Proof. Thanks to Theorem 1 and Theorem 3, the commutativity problem for Hadamard-finite series is decidable. In fact, we have reduced the commutativity problem over alphabet Σ to $|\Sigma|^2 + |\Sigma|$ equivalence queries (cf. Lemma 2).

On the other hand, the equivalence problem for Hadamard-finite series efficiently reduces to the commutativity problem (i.e., in polynomial time). Indeed, let $f \in \mathbb{Q}\langle\langle\Sigma\rangle\rangle$ be a Hadamard-finite series and consider two fresh input symbols $a, b \notin \Sigma$. Thanks to Lemma 3, $g := ab \sqcup\!\sqcup\, f$ is commutative if, and only if, $f = \mathbb{0}$. Moreover, since ab is Hadamard-finite (even rational), g is Hadamard finite by Lemma 9. Furthermore, a finite representation for g can be constructed in polynomial time from a finite representation of f. Thus, we have reduced checking whether $f = \mathbb{0}$ to checking whether g is commutative.

Summarising, commutativity and equivalence are polynomial-time equivalent. The complexity of equivalence for polynomial automata is Ackermann-complete [6, Theorem 1 + Corollary 1], thus the same is true for Hadamard-finite series (cf. Remark 3). □

F. Application: Multivariate polynomial recursive sequences

We introduce a class of multivariate recursive sequences generalising the univariate *polynomial recursive sequences* (in short, *polyrec*) from [15]. We will make a crucial use of Theorem 4 to argue that this class can be presented *effectively* by polynomial recursive equations. In other words, we will show that polynomial recursive equations are a *decidable syntax* for polyrec sequences: There is an algorithm which, given a system of polynomial recursive equations in input, decides whether they actually define a (polyrec) sequence at all. The lack of an effective presentation has been an obstacle to the study of a multivariate generalisation of polyrec sequences. In § III-F1 we begin with some preliminaries about multivariate sequences, in § III-F2 we define multivariate polyrec sequences and study some of their basic properties, and finally in § III-F3 we use Theorem 4 to argue that their syntax is effective.

1) Difference sequence algebra: Fix a dimension $d \in \mathbb{N}_{\geq 1}$. For every $1 \leq j \leq d$, by e_j we denote the j-th unit vector in $\mathbb{N}^d$. A *multivariate sequence* is a function $f : \mathbb{N}^d \to \mathbb{Q}$. We equip the set of sequences with the structure of a difference algebra $\left(\mathbb{N}^d \to \mathbb{Q}; +, \cdot, (\sigma_j)_{1 \leq j \leq d}\right)$, where $\mathbb{0}$, $\mathbb{1}$, scalar product $c \cdot f$ ($c \in \mathbb{Q}$), sum $f + g$, and multiplication $f \cdot g$ are defined element-wise, and the *j-th left shift* endomorphism σ_j maps a sequence $f : \mathbb{N}^d \to \mathbb{Q}$ to the sequence $\sigma_j f$ s.t.

$$(\sigma_j f)(n) := f(n + e_j), \quad \text{for all } n \in \mathbb{N}^d.$$

Unlike the difference algebra of Hadamard series from § III-A2, the endomorphisms σ_j's pairwise commute $\sigma_j \circ \sigma_h = \sigma_h \circ \sigma_j$.

2) Polyrec sequences: A d-variate sequence $f : \mathbb{N}^d \to \mathbb{Q}$ is *polyrec* if there exist $k \in \mathbb{N}_{\geq 1}$, auxiliary sequences $f_1, \dots, f_k : \mathbb{N}^d \to \mathbb{Q}$ with $f = f_1$, and polynomials $p_i^{(j)} \in \mathbb{Q}[k]$ for all $1 \leq i \leq k$ and $1 \leq j \leq d$, s.t.

$$\sigma_j f_i = p_i^{(j)}(f_1, \dots, f_k), \quad \forall 1 \leq i \leq k, 1 \leq j \leq d. \tag{9}$$

In other words, for each f_i and coordinate $1 \leq j \leq d$ there is a distinct recursive equation specifying how future values can be computed from the current one.

The univariate case $d = 1$ corresponds to the polyrec sequences from [15], a rich class containing the linear recursive sequences (such as the Fibonacci numbers), as well as fast growing sequences such as 2^{2^n}. Moreover, the zeroness and equivalence problems for univariate polyrec sequences are decidable (with Ackermann complexity [6]), and whether there is an elementary algorithm has been open for some time (cf. [20]). In order to develop intuition, we show below some examples of polyrec sequences.

Example 5. We begin with a simple example, univariate and linear. The *Fibonacci sequence* $F : \mathbb{N} \to \mathbb{Q}$ is well-known to satisfy the linear recursion $F(n+2) = F(n+1) + F(n)$. We can turn this into the polyrec format by introducing an auxiliary sequence $G : \mathbb{N} \to \mathbb{Q}$ s.t. $\sigma_1 F = F + G$ and $\sigma_1 G = F$. In this way, one shows that all linear recursive sequences are polyrec.

Example 6. We now consider a bivariate example $f : \mathbb{N}^2 \to \mathbb{Q}$. For every $n_1, n_2 \in \mathbb{N}$, let $f(n_1, n_2) := 2^{2^{n_1}} \cdot 2^{3^{n_2}}$. Introducing auxiliary sequences $g(n_1, n_2) := 2^{2^{n_1}}$ and $h(n_1, n_2) := 2^{3^{n_2}}$ we have polyrec equations

$$\begin{aligned} \sigma_1 f &= f \cdot g, & \sigma_1 g &= g^2, & \sigma_1 h &= h, \\ \sigma_2 f &= f \cdot h^2, & \sigma_2 g &= g, & \sigma_2 h &= h^3. \end{aligned}$$

Example 7. The bivariate factorial $f(n_1, n_2) := n_1! \cdot n_2!$ satisfies the recursion

$$\begin{aligned} f(n_1 + 1, n_2) &= (\underline{n_1} + 1) \cdot f(n_1, n_2), \\ f(n_1, n_2 + 1) &= (\underline{n_2} + 1) \cdot f(n_1, n_2). \end{aligned}$$

This is not in the polyrec format (9), because of the two underlined occurrences of the index variables n_1, n_2. Linear recursions with coefficients polynomial in $n_1, \dots, n_d$ are called *P-recursive* [70]. We can introduce auxiliary sequences $g(n_1, n_2) := n_1$ and $h(n_1, n_2) := n_2$ and transform P-recursions into polyrec equations

$$\begin{aligned} \sigma_1 f &= (g + \mathbb{1}) \cdot f, & \sigma_1 g &= \mathbb{1} + g, & \sigma_1 h &= h, \\ \sigma_2 f &= (h + \mathbb{1}) \cdot f, & \sigma_2 g &= g, & \sigma_2 h &= \mathbb{1} + h. \end{aligned}$$

A similar argument shows that $n_1! + n_2!$ is polyrec. In fact, polyrec include all P-recursive definitions with a constant leading term (cf. [47, Proposition 3.6]), also called *monic P-recursive* in [14, page 5]. For instance, $n!$ is monic P-recursive since $(n+1)! = (n+1) \cdot n!$, but the *Catalan numbers C_n* are not since $\underline{(n+2)} \cdot C_{n+1} = (4n+2) \cdot C_n$ (the underlined term is non-constant).

Remark 5. All the examples so far are sums of products of univariate polyrec sequences. This needs not be the case in general. For instance, $(n_1 + n_2)!$ and $2^{2^{n_1 + n_2}}$ are polyrec, but not of that form. This shows that the characterisation of multivariate *linear* recursive sequences in [40, Proposition 2.1] as sums of products of univariate linear recursive sequences does not generalise to the polyrec case.

More examples will be shown in § III-F3. Multivariate polyrec sequences are a well-behaved class of sequences. They are closed under the algebra operations (scalar product, sum, and multiplication), shifts $\sigma_j f$, *sections* [47, Definition 3.1], and *diagonals* [47, Definition 2.6]. More details about closure properties can be found in § C-E.

3) Consistency of polyrec equations: So far we have defined a class of multivariate sequences, and we have argued that this class is robust by presenting examples and closure properties. However, we have sidestepped a crucial issue, namely whether the equations (9) together with an initial condition do actually define a sequence. By the shape of the equations, a given initial condition gives rise to *at most one* solution. However, whether a solution exists at all is nontrivial.

Problem 1 (Polyrec consistency problem)**.** *The* consistency problem *for polyrec equations takes as input a set of polyrec equations* (9) *together with an initial condition $c \in \mathbb{Q}^k$, and it amounts to decide whether there is a sequence solution $f \in (\mathbb{N}^d \to \mathbb{Q})^k$ extending the initial condition $f(0) = c$.*

We emphasise that Problem 1 is about polyrec equations, which may or may not represent a polyrec sequence, and the question is finding out which one is the case. We show that Problem 1 can algorithmically be decided, thus showing that the syntax of polyrec sequences (9) is *effective*. In the <u>univariate</u> case $d = 1$, *every* initial condition extends to a solution, and thus the consistency problem is trivial. However, in the multivariate case $d \geq 2$, whether a system of equations (9) actually defines a (tuple of) sequence(s) involves an infinite number of constraints to be satisfied (cf. Figure 1).

Example 8 (All initial conditions extend to a solution). Let $d = 2$ and consider polyrec equations

$$\sigma_1 f = f^3 \quad \text{and} \quad \sigma_2 f = f^5.$$

The update functions $p(x) := x^3$ and $q(x) := x^5$ commute $p(q) = q(p) = x^{15}$ identically. This is a very strong condition, implying that *every* initial condition $f(0,0) := c \in \mathbb{Q}$ extends to a sequence solution, namely $f(n_1, n_2) = c^{3^{n_1} \cdot 5^{n_2}}$.

Example 9 (Some initial conditions extend to a solution). Revisiting Example 2, let $d = 2$ and consider polyrec equations

$$\sigma_1 f = f^2 \quad \text{and} \quad \sigma_2 f = \mathbb{1} - f^2.$$

We claim that the set of initial values extending to a solution is $\{-1, 0, 1\}$. Indeed, if we let $p(x) := x^2$ and $q(x) := 1 - x^2$, then the equation $p(q(x)) = q(p(x))$ has exactly three solutions $x \in \{-1, 0, 1\}$. This set is stable under $p(x)$ and $q(x)$:

$$\begin{tikzcd}
0 \arrow[r, "p"] & 0 & & 1 \arrow[r, "p"] & 1 & & 0 \arrow[r, "p"] & 0 \\
-1 \arrow[r, "p"'] \arrow[u, "q"] & 1 \arrow[u, "q"'] & & 0 \arrow[r, "p"'] \arrow[u, "q"] & 0 \arrow[u, "q"'] & & 1 \arrow[r, "p"'] \arrow[u, "q"] & 1 \arrow[u, "q"']
\end{tikzcd}$$

Consequently, any $f(0,0) \in \{-1, 0, 1\}$ extends to a solution.

Example 10 (No initial condition extends to a solution). Let $d = 2$ and consider the polyrec equations

$$\sigma_1 f = f^2 \quad \text{and} \quad \sigma_2 f = f + \mathbb{1}. \tag{10}$$

Let the initial condition be $f(0,0) := c \in \mathbb{Q}$. From the first equation we have $f(1,0) = c^2$, and from the second $f(0,1) = c + 1$. But then we can obtain $f(1,1)$ in two ways, $c^2 + 1$ and $(c+1)^2 = c^2 + 2c + 1$, implying $c = 0$ and thus $f(1,1) = 1$. But now we compute $f(2,2)$ in two ways obtaining $1^2 + 1 = 2$ and $(1+1)^2 = 4$. This leads to a contradiction, and thus there is no initial condition extending to a solution of (10).

Problem 1 is an instance of the more general problem solvability in sequences for systems of polynomial difference equations, not necessarily of the polyrec kind (9). We now recall that the latter problem is undecidable in full generality.

Remark 6. The more general problem of solvability in sequences for systems of polynomial difference equations is undecidable [60, Proposition 3.9]. Nonlinear bivariate systems of two unknowns $k = d = 2$ suffice. The undecidable systems [60, Eq. (5) on pg. 21] in our notation can be written as follows:

$$\begin{aligned}
(f-1)(f-2)\cdots(f-N) &= 0, \\
(g-1)(g-2)\cdots(g-N) &= 0, \\
\prod_{i=1}^{\ell} \left((a_\ell - f)^2 + (b_\ell - \sigma_2 f)^2 + (c_\ell - g)^2 + (d_\ell - \sigma_1 g)^2 \right) &= 0,
\end{aligned}$$

for suitable constants $N, a_1, \ldots, a_\ell, b_1, \ldots, b_\ell, c_1, \ldots, c_\ell \in \mathbb{N}$, where for simplicity, we identify $n \in \mathbb{N}$ with $n \cdot \mathbb{1}$. (This is used to encode plane tiling problems, where the first two constraints force $f, g : \mathbb{N}^d \to \{1, \ldots, N\}$ to range over a finite set, and the last equation encodes horizontal and vertical tiling constraints.) None of the equations above is in the polyrec format (9).

Consequently, in order to solve Problem 1 we need to exploit the special shape of polyrec. To this end, we develop a connection between sequence solutions of polyrec equations (9) and Hadamard-finite series.

We make the very simple observation that number sequences and commutative series contain the same information. For instance, the sequence $2^{n_1} 3^{n_2}$ is "the same" as the series that maps $w \in \{a_1, a_2\}^*$ to $2^{\#_{a_1} w} 3^{\#_{a_2} w}$. This is made precise by saying that the difference sequence algebra $\left(\mathbb{N}^d \to \mathbb{Q}; +, \cdot, (\sigma_j)_{1 \le j \le d} \right)$ and the difference Hadamard algebra of commutative series $\left(\mathbb{Q}\langle\!\langle \Sigma \rangle\!\rangle^{\mathsf{comm}}; +, \cdot, (\delta^L_{a_j})_{1 \le j \le d} \right)$ (with $\Sigma = \{a_1, \ldots, a_d\}$) are isomorphic. The isomorphism maps a commutative series $f \in \mathbb{Q}\langle\!\langle \Sigma \rangle\!\rangle^{\mathsf{comm}}$ to the number sequence $\widetilde{f} : \mathbb{N}^d \to \mathbb{Q}$ s.t. $\widetilde{f}(n_1, \ldots, n_d) := f(a_1^{n_1} \cdots a_d^{n_d})$ for every $n_1, \ldots, n_d \in \mathbb{N}$. This map preserves the algebra operations and the endomorphisms, which has pleasant consequences.

Given a system of polyrec equations (9), we construct a system of Hadamard-finite equations

$$\delta^L_{a_j} g_i = p_i^{(j)}(g_1, \ldots, g_k), \qquad \forall 1 \le i \le k, 1 \le j \le d, \tag{11}$$

which we call the *companion system* to (9).

Example 11. The polyrec equations from Example 9 $\left\{ \sigma_1 f = f^2, \sigma_2 f = \mathbb{1} - f^2 \right\}$ yield the companion system from Example 4 $\left\{ \delta^L_{a_1} g = g^2, \delta^L_{a_2} g = \mathbb{1} - g^2 \right\}$.

Every initial condition extends to a *unique* series solution $g = (g_1, \ldots, g_k) \in \mathbb{Q}\langle\!\langle \Sigma \rangle\!\rangle^k$ of (11), which can be proved by induction on the length of words. By definition, this solution is a tuple of Hadamard-finite series. When all components $g_1, \ldots, g_k$ are commutative, we can extract corresponding number sequences $f_1 := \widetilde{g_1}, \ldots, f_k := \widetilde{g_k} : \mathbb{N}^d \to \mathbb{Q}$, which are precisely the unique sequence solution to the original polyrec equations (9):

$$\sigma_j f_i = \widetilde{\delta^L_{a_j} g_i} = \left(p_i^{(j)} \widetilde{(g_1, \ldots, g_k)} \right) = p_i^{(j)}(f_1, \ldots, f_k).$$

On the other hand, when any component g_i is noncommutative, (9) does not have any number sequence solution. To see this, it suffices to notice that any sequence solution of (9) gives rise to a commutative Hadamard-finite series solution of the companion system (11) (via the inverse of the isomorphism above), but as we have observed (11) has a unique solution.

The argument above gives us an algorithm for the consistency problem: Construct the companion system (11) and check that all components of the unique Hadamard-finite series solution are commutative, which is decidable by Theorem 4.

Theorem 5. *The polyrec consistency problem (Problem 1) is decidable with Ackermann complexity.*

IV. Shuffle automata and series

In this section we recall *weighted basic parallel processes* [18], a nonlinear weighted model of computation generalising weighted finite automata inspired from the theory of concurrency. For reasons of uniformity with the rest of the paper, we will refer to it as *shuffle automata*. We will briefly recall the necessary basic results on shuffle automata from [18], to which we refer for more details. The novel technical result in this section is that the series recognised by shuffle automata are effectively closed under right derivatives (Lemma 12), and thus constitute an effective prevariety (Theorem 6). The closure property is nontrivial, since the definition of shuffle automata is asymmetric w.r.t. right and left derivatives. Together with the 2-EXPSPACE algorithm for the equivalence problem [18, Theorem 1], we obtain that commutativity is decidable, with the same complexity (Theorem 7). We then present an application to multivariate *constructible differentially algebraic* power series (CDA) introduced by Bergeron and Sattler [8]. We show that their syntax is effective (Theorem 8), which was an open question [18, Remark 27].

A. Preliminaries

1) Differential algebra: A *derivation* of an algebra $(A; +, *)$ is a linear function $\delta : A \to A$ satisfying the following product rule (*Leibniz rule*)

$$\delta(\alpha * \beta) = \delta\alpha * \beta + \alpha * \delta\beta, \quad \text{for all } \alpha, \beta \in A. \tag{12}$$

A *differential algebra* $(A; +, *, (\delta_j)_{1 \le j \le d})$ is an algebra together with finitely many derivations $\delta_1, \ldots, \delta_d : A \to A$ [64], [39], [42]. We do not require the δ_i's to pairwise commute. As an example, consider the algebra of polynomials $(\mathbb{Q}[X]; +, \cdot)$. A derivation $\Delta : \mathbb{Q}[X] \to \mathbb{Q}[X]$ acts on a polynomial by linearity and (12). For instance, $\Delta(X_1 \cdot X_2) = \Delta X_1 \cdot X_2 + X_1 \cdot \Delta X_2$ and $\Delta(X_1^5) = 5 \cdot X_1^4 \cdot \Delta X_1$. Thus, a derivation Δ of the polynomial algebra is uniquely determined once we fix $\Delta X_1, \ldots, \Delta X_k$. For instance, the familiar partial derivative $\frac{\partial}{\partial X_i}$ (for short, ∂_{X_i}) is the unique derivation Δ s.t. $\Delta X_i = 1$ and $\Delta X_j = 0$ for $j \neq i$. If we fix distinguished derivations $\Delta_1, \ldots, \Delta_d$, then we obtain the polynomial differential algebra $(\mathbb{Q}[X]; +, \cdot, (\Delta_j)_{1 \le j \le d})$. Polynomial differential algebras will provide the state space of configurations for shuffle automata. Other examples of differential algebras will be presented in § IV-A2 and § IV-C.

2) Shuffle differential algebra: We recall a product operation on series which will be central to the semantics of shuffle automata. The *shuffle product* of two series $f \sqcup\!\sqcup g$ is defined coinductively by (cf. [3, Definition 8.1])

$$[\varepsilon](f \sqcup\!\sqcup g) = f_\varepsilon \cdot g_\varepsilon, \tag{$\sqcup\!\sqcup$-ε}$$

$$\delta_a^L(f \sqcup\!\sqcup g) = \delta_a^L f \sqcup\!\sqcup g + f \sqcup\!\sqcup \delta_a^L g, \quad \forall a \in \Sigma. \tag{$\sqcup\!\sqcup$-δ_a^L}$$

This is an associative and commutative operation, with identity the series $1 \cdot \varepsilon$. It originates in the work of Eilenberg and MacLane in homological algebra [27], and was introduced in automata theory by Fliess under the name of *Hurwitz product* [30]. It is the series analogue of the shuffle product in language theory, and it finds applications in concurrency theory, where it models the *interleaving semantics* of process composition. Intuitively, it models all possible ways in which two sequences can be interleaved. For instance, when we run processes ab and a in parallel we obtain

$$\begin{aligned} ab \sqcup\!\sqcup a &= a(\delta_a^L(ab \sqcup\!\sqcup a)) + b(\delta_b^L(ab \sqcup\!\sqcup a)) = \\ &= a(\delta_a^L(ab) \sqcup\!\sqcup a + ab \sqcup\!\sqcup \delta_a^L a) + b(0) = a(b \sqcup\!\sqcup a + ab \sqcup\!\sqcup \varepsilon) = \\ &= a(ba + ab + ab) = 2aab + aba. \end{aligned}$$

Thus, there are two executions over input aab and one over aba. By ($\sqcup\!\sqcup$-δ_a^L), left derivatives δ_a^L ($a \in \Sigma$) are derivations for the shuffle product, and we have thus obtained the *differential shuffle algebra* $\mathbb{Q}\langle\langle\Sigma\rangle\rangle_{\sqcup\!\sqcup} := \left(\mathbb{Q}\langle\langle\Sigma\rangle\rangle; +, \sqcup\!\sqcup, (\delta_a^L)_{a \in \Sigma}\right)$. This is where the semantics of shuffle automata lives.

Before proceeding, it is worth noting that right derivatives are also derivations of the shuffle algebra. This will be useful in the proof of Lemma 12.

Lemma 10. *Right derivatives δ_a^R (with $a \in \Sigma$) are shuffle algebra derivations. I.e., they are linear and satisfy Leibniz rule (12) for the shuffle product:*

$$\delta_a^R(f \sqcup\!\sqcup g) = \delta_a^R f \sqcup\!\sqcup g + f \sqcup\!\sqcup \delta_a^R g, \quad \forall a \in \Sigma. \tag{13}$$

B. Shuffle automata and shuffle-finite series

Syntactically, a shuffle automaton $A = (\Sigma, X, F, \Delta)$ is identical to a Hadamard automaton (cf. § III-B), however the semantics is different: The transition $\Delta_a : X \to \mathbb{Q}[X]$ (for $a \in \Sigma$) is extended to a unique *derivation* $\Delta_a : \mathbb{Q}[X] \to \mathbb{Q}[X]$ of the differential polynomial algebra of configurations (cf. (12)). The fact that such an extension exists and is unique is a basic fact from differential algebra, cf. [39, page 10, point 4]. The extension to all finite words and the definition of the semantics of a configuration are unchanged (cf. (3)). The series recognised by the shuffle automaton A is $[\![X_1]\!]$.

Example 12. Consider a binary alphabet $\Sigma = \{a_1, a_2\}$ and three nonterminals $X = \{X_1, X_2, X_3\}$. Let the output function be $FX_1 := 1$ and $FX_2 := FX_3 := 0$, and consider transitions

$$\begin{aligned} &\Delta_{a_1} X_1 := X_1 \cdot (1 + X_3), && \Delta_{a_1} X_2 := 1, && \Delta_{a_1} X_3 := 0, \\ &\Delta_{a_2} X_1 := X_1 \cdot X_2, && \Delta_{a_2} X_2 := 0, && \Delta_{a_2} X_3 := 1. \end{aligned}$$

The run starting from X_1 and reading input $w = a_1 a_2$ is

$$\begin{aligned} X_1 &\xrightarrow{a_1} X_1 \cdot (1 + X_3) \xrightarrow{a_2} \Delta_{a_2}(X_1 \cdot (1 + X_3)) = \\ &= \Delta_{a_2} X_1 \cdot (1 + X_3) + X_1 \cdot \Delta_{a_2}(1 + X_3) = \\ &= X_1 \cdot X_2 \cdot (1 + X_3) + X_1. \end{aligned}$$

and thus $[\![X_1]\!]_{a_1 a_2} = F(X_1 \cdot X_2 \cdot (1 + X_3) + X_1) = 1$. We will see in Example 13 that the series recognised by X_1 maps w to $\binom{n}{k} \cdot k!$ where $n := \#_{a_1} w$ and $k := \#_{a_2} w$.

Shuffle automata have been introduced in [18] as a simultaneous generalisation of weighted finite automata [66] and a model of concurrency called *basic parallel processes* [28]. In the rest of the section, we recall their basic properties. A shuffle automaton endows the algebra of configurations with the structure of a differential algebra $(\mathbb{Q}[X]; +, \cdot, (\Delta_a)_{a \in \Sigma})$. The following connection with the differential algebra of series bears similarity to Lemma 5.

Lemma 11 (Properties of the semantics [18, Lemma 8 + Lemma 9])**.** *The semantics of a shuffle automaton is a homomorphism from the differential algebra of polynomials* $(\mathbb{Q}[X]; +, \cdot, (\Delta_a)_{a\in\Sigma})$ *to the differential shuffle algebra of series* $\left(\mathbb{Q}\langle\langle\Sigma\rangle\rangle; +, \sqcup\!\sqcup, (\delta_a^L)_{a\in\Sigma}\right)$*. In other words,* $[\![0]\!] = \mathbb{0}$*,* $[\![1]\!] = 1\cdot\varepsilon$*, and, for all configurations* $\alpha, \beta \in \mathbb{Q}[X]$*,*

$$[\![c\cdot\alpha]\!] = c\cdot[\![\alpha]\!] \qquad \forall c\in\mathbb{Q}, \tag{14}$$

$$[\![\alpha+\beta]\!] = [\![\alpha]\!] + [\![\beta]\!], \tag{15}$$

$$[\![\alpha\cdot\beta]\!] = [\![\alpha]\!] \sqcup\!\sqcup [\![\beta]\!], \tag{16}$$

$$[\![\Delta_a\alpha]\!] = \delta_a^L\,[\![\alpha]\!] \qquad \forall a\in\Sigma. \tag{17}$$

Shuffle automata are data structures representing series. The same class of series admits a semantic presentation, which we find more convenient to work with and which we now recall. For series $f_1,\dots,f_k \in \mathbb{Q}\langle\langle\Sigma\rangle\rangle$, we denote by $\mathbb{Q}[f_1,\dots,f_k]_{\sqcup\!\sqcup}$ the smallest shuffle algebra containing $f_1,\dots,f_k$, which we call *finitely generated* with *generators* $f_1,\dots,f_k$. When it is closed under the derivations δ_a^L (for all $a\in\Sigma$), it is called a *differential shuffle algebra*. A series $f\in\mathbb{Q}\langle\langle\Sigma\rangle\rangle$ is *shuffle finite* [18, Section 2.4] if it belongs to a finitely generated differential shuffle algebra (note the analogy with Hadamard-finite series, cf. Definition 1). By [19, Lemma 39], this is equivalent to the existence of generators $f_1,\dots,f_k\in\mathbb{Q}\langle\langle\Sigma\rangle\rangle$ s.t.:

1) $f\in\mathbb{Q}[f_1,\dots,f_k]_{\sqcup\!\sqcup}$, and
2) $\delta_a^L f_i \in \mathbb{Q}[f_1,\dots,f_k]_{\sqcup\!\sqcup}$, for all $a\in\Sigma$ and $1\le i\le k$.

Thanks to this, one can prove that shuffle-finite series coincide with the series recognised by shuffle automata [18, Theorem 12], and constitute an effective differential algebra [18, Lemma 10]. We complete the picture by observing that they are effectively closed under right derivatives.

Lemma 12. *If f is shuffle finite, then $\delta_a^R f$ is also effectively shuffle finite.*

The definition of shuffle-finite series is asymmetric since it is biased towards left derivatives, therefore closure under right derivatives is nontrivial. Nonetheless, the technology of shuffle-finite series gives a rather short proof of this fact, showing their effectiveness. Thanks to the lemma and [18, Theorem 12], series recognised by shuffle automata are closed under right derivatives. We do not know how to establish this without shuffle-finite series, since it is not clear how transitions for the new automaton should be defined.

Proof. The proof is similar to the one of Lemma 8. Consider $f\in A := \mathbb{Q}[f_1,\dots,f_k]_{\sqcup\!\sqcup}$, where the latter algebra is differential. Consider the shuffle algebra B generated by the f_i's and their right derivatives, $B := \mathbb{Q}[f_1,\dots,f_k,\delta_a^R f_1,\dots,\delta_a^R f_k]_{\sqcup\!\sqcup}$.

Claim. *For every series $g\in A$, we have $\delta_a^R g\in B$.*

Proof. We proceed by induction on polynomial expressions. In the base case, we have $g = f_i \in A$ and there is nothing to prove. In the case $g = g_1 + g_2$ with $g_1, g_2\in A$, we have $\delta_a^R g = \delta_a^R g_1 + \delta_a^R g_2$ and the claim follows from the inductive assumption applied to $\delta_a^R g_1, \delta_a^R g_2$. Finally, consider $g = g_1 \sqcup\!\sqcup g_2$ with $g_1, g_2\in A$. By the product rule (13), we have $\delta_a^R g = \delta_a^R g_1 \sqcup\!\sqcup g_2 + g_1 \sqcup\!\sqcup \delta_a^R g_2$. By the inductive assumption, $\delta_a^R g_1, \delta_a^R g_2 \in B$, and thus the same holds for $\delta_a^R g$, □

Since by assumption $f\in A$, by the claim $\delta_a^R f\in B$. Left derivatives of the old generators f_i are in $A\subseteq B$ since A is differential by assumption. It remains to show the same for the new generators $\delta_a^R f_i$. So consider $\delta_b^L\delta_a^R f_i$ for an arbitrary $b\in\Sigma$ and we have to show that it is in B. By Lemma 1, we have $\delta_a^L\delta_b^R f_i = \delta_b^R\delta_a^L f_i$. But by assumption $\delta_a^L f_i\in A$, therefore by the claim $\delta_b^R\delta_a^L f_i\in B$, as required. □

We thus obtain the following synthesis of the above-mentioned results from [18] and Lemma 12.

Theorem 6. *The class of shuffle-finite series is an effective prevariety.*

From Theorem 1, Theorem 6, and the fact that the equality problem for shuffle-finite series is in 2-EXPSPACE [18, Theorem 1], we obtain the main result of the section.

Theorem 7. *The commutativity problem for shuffle-finite series (equivalently, series recognised by shuffle automata) is decidable in 2-EXPSPACE.*

As directly inherited from the equality problem, the complexity is doubly exponential in the number of generators k.

C. Application: Multivariate constructible differentially algebraic power series

In this section, we recall the definition of a rich class of multivariate series in commuting indeterminates called *constructible differentially algebraic* (CDA). Based on Theorem 7, we show that they have an effective syntax, addressing a problem left open in [18, Remark 27]. We begin in § IV-C1 with some preliminaries, then in § IV-C2 we recall the definition of CDA, and finally we show effectiveness in § IV-C3.

1) Differential power series algebra: In the rest of the section, fix a dimension $d\in\mathbb{N}_{\ge 1}$ and consider d pairwise commuting *independent variables* $x = (x_1,\dots,x_d)$. For a multi-index $n = (n_1,\dots,n_d)\in\mathbb{N}^d$, we write x^n for $x_1^{n_1}\cdots x_d^{n_d}$, and $n!$ for $n_1!\cdots n_d!$. An *exponential multivariate power series* (in commuting variables) is a function $f:\mathbb{N}^d\to\mathbb{Q}$, customarily written as $f = \sum_{n\in\mathbb{N}^d} f_n\cdot\frac{x^n}{n!}$ where $f_n\in\mathbb{Q}$ is the coefficient of term x^n. We write the set of power series as $\mathbb{Q}[[x]]$. For example $1 + x_1 + x_1^2 + \cdots = \frac{1}{1-x_1}$ is a power series. Polynomials $\mathbb{Q}[x]$ consist precisely of those power series with *finite support*, i.e., with $f_n = 0$ almost everywhere. We endow the set of power series with the structure of a vector space with zero 0, scalar product $c\cdot f$, and addition $f+g$ all defined element-wise. Therefore, the vector spaces of power series and number sequences are isomorphic. However, multiplication is different. While multiplication of number sequences is element-wise, power series multiplication $f\cdot g$ extends to infinite supports the familiar multiplication of polynomials (sometimes called *Cauchy product*). Finally, the *j-th partial derivative* ∂_{x_j} (for $1\le j\le d$) maps the power series f to the power series $\partial_{x_j} f = \sum_{n\in\mathbb{N}^d} f_{n+e_j}\frac{x^n}{n!}$. Notice that ∂_{x_j} is a derivation with respect to the product of power series, since it satisfies the familiar Leibniz rule from calculus (cf. (12)):

$$\partial_{x_j}(f \cdot g) = (\partial_{x_j} f) \cdot g + f \cdot (\partial_{x_j} g).$$

For example, $\partial_{x_1} \frac{1}{1-x_1} = \frac{1}{(1-x_1)^2}$. We have thus defined the *differential power series algebra* $\left(\mathbb{Q}[[x]]; +, \cdot, (\partial_{x_j})_{1 \le j \le d}\right)$, where partial derivatives are pairwise commuting $\partial_{x_j} \circ \partial_{x_h} = \partial_{x_h} \circ \partial_{x_j}$. This will constitute the semantic background for the developments in the rest of the section.

2) CDA power series: A power series $f \in \mathbb{Q}[[x]]$ is CDA [7], [8] if there exist $k \in \mathbb{N}_{\ge 1}$ and auxiliary power series $f_1, \dots, f_k \in \mathbb{Q}[[x]]$ with $f = f_1$ satisfying a system of polynomial partial differential equations

$$\partial_{x_j} f_i = p_i^{(j)}(f_1, \dots, f_k), \quad \forall 1 \le i \le k, 1 \le j \le d, \tag{18}$$

where $p_i^{(j)} \in \mathbb{Q}[y]$ (with $y = (y_1, \dots, y_k)$) for all $1 \le i \le k$ and $1 \le j \le d$. We call systems of the form above *systems of autonomous CDA equations*.

Remark 7 (Autonomous vs. non-autonomous). We could allow the r.h.s. of equations to contain the independent variables x_j's, i.e., $p_i^{(j)} \in \mathbb{Q}[x, y]$, obtaining *non-autonomous* equations. This does not increase expressiveness since we can introduce fresh power series $g_j := x_j$ together with equations $\partial_{x_j} g_j = 1$, $\partial_{x_h} g_j = 0$ for $h \ne j$, and initial conditions $g_j(0) = 0$. For this reason, we focus on autonomous equations.

CDA power series include all algebraic power series (i.e., solutions of polynomial equations [45, Chapter III.16]), the exponential power series e^x, the trigonometric power series $\sin x, \cos x$, etc..., and are closed under natural operations, such as scalar product, addition, multiplication, differentiation, and composition (cf. [18, Lemma 21]). They find applications in combinatorics, since the generating functions of a large class of finite structures are CDA [18, Theorem 31]. Finally, the CDA equivalence problem is decidable in 2-EXPTIME, thus of elementary complexity [18, Theorem 3]; this should be contrasted with polyrec sequences, for which the best upper bound is Ackermannian § III-F2.

Example 13 (Binomial coefficients). Consider the mixed generating power series of the binomial coefficients $f_1 := \sum_{n,k \in \mathbb{N}} \binom{n}{k} \cdot \frac{x_1^n}{n!} \cdot x_2^k = e^{x_1 \cdot (1+x_2)}$. By introducing auxiliary power series $f_2 := x_1$ and $f_3 := x_2$, we obtain the CDA equations:

$$\begin{aligned} \partial_{x_1} f_1 &= f_1 \cdot (1 + f_3), & \partial_{x_1} f_2 &= 1, & \partial_{x_1} f_3 &= 0, \\ \partial_{x_2} f_1 &= f_1 \cdot f_2, & \partial_{x_2} f_2 &= 0, & \partial_{x_2} f_3 &= 1. \end{aligned}$$

3) Solvability in power series of CDA equations: The definition of CDA power series relies on the *promise* that (18) does indeed have a power series solution. This is a nontrivial problem, left open in [18, Remark 27].

Problem 2 (CDA solvability problem)**.** *The* solvability problem *for CDA equations takes as input a set of CDA equations* (18) *together with an initial condition $c \in \mathbb{Q}^k$, and it amounts to decide whether there is a power series solution $f \in \mathbb{Q}[[x]]^k$ extending the initial condition $f(0) = c$.*

We remark that the initial condition is part of the input to the solvability problem. In the univariate case $d = 1$, by the classic Picard-Lindelöf theorem [21, Theorem 3.1], *every* initial condition extends to a solution, and thus the solvability problem is trivial. The multivariate case $d \ge 2$ is nontrivial, as the following examples demonstrate.

Example 14 (Unsolvable systems). A CDA system may not have any power series solution when $d \ge 2$, regardless of the initial condition. For instance, the following equations are unsatisfiable in power series:

$$\begin{aligned} \partial_{x_1} f &= 0, \\ \partial_{x_2} f &= g, \end{aligned} \quad \text{and} \quad \begin{aligned} \partial_{x_1} g &= 1, \\ \partial_{x_2} g &= 1. \end{aligned}$$

The equation $\partial_{x_1} g = 1$ implies $g = x_1 + c(x_2)$ for some $c \in \mathbb{Q}[[x_2]]$, and $\partial_{x_2} g = 1$ implies $g = x_2 + d(x_1)$ for some $d \in \mathbb{Q}[[x_1]]$. Taken together, we have $g = a + x_1 + x_2$ for some $a \in \mathbb{Q}$. The equation $\partial_{x_1} f = 0$ implies $f \in \mathbb{Q}[[x_2]]$, and thus $\partial_{x_2} f = g \in \mathbb{Q}[[x_2]]$, which is a contradiction.

Example 15 (Sensitivity to the inital condition). Existence of power series solutions may depend on the initial value when $d \ge 2$. Consider equations

$$\begin{aligned} \partial_{x_1} f &= f + g, \\ \partial_{x_2} f &= 0, \end{aligned} \quad \text{and} \quad \begin{aligned} \partial_{x_1} g &= 0, \\ \partial_{x_2} g &= g. \end{aligned}$$

Thus $f \in \mathbb{Q}[[x_1]]$, and $g = a \cdot e^{x_2 + b} \in \mathbb{Q}[[x_2]]$, for some $a, b \in \mathbb{Q}$. Since $g = \partial_{x_1} f - f \in \mathbb{Q}[[x_2]]$, there are no solutions with $g(0) = a \cdot e^b \ne 0$. When $g(0) = 0$ (iff $a = 0$), we have the family of solutions $g = 0$ and $f = c \cdot e^{x_1 + d}$ for every $c, d \in \mathbb{Q}$.

Solvability in power series of polynomial partial differential equations is undecidable, as we now recall.

Remark 8. A seminal result of Denef and Lipshitz shows that solvability in power series for polynomial partial differential equations is *undecidable* [24, Theorem 4.11]. In fact, this holds already for a *single*, *linear* partial differential equation in *one unknown*, with polynomial coefficients in $\mathbb{Q}[x]$. In order to gain insight into the shape of the undecidable equations, we present their reduction. Let $P(y_1, \dots y_d) \in \mathbb{Z}[y_1, \dots, y_d]$ be a multivariate polynomial with integer coefficients. Consider the non-autonomous differential equation

$$P(x_1 \cdot \partial_{x_1}, \dots, x_d \cdot \partial_{x_d}) f = \frac{1}{1 - x_1} \cdots \frac{1}{1 - x_d}, \tag{19}$$

with initial condition $f(0) = P(0)^{-1}$ (assuming $P(0) \ne 0$). For example, if $P(y_1, y_2) = 1 - 2 \cdot y_1 + y_2^2$ then we would get the equation $(1 - 2 \cdot x_1 \cdot \partial_{x_1} + (x_2 \cdot \partial_{x_2})^2) f = \frac{1}{1-x_1} \frac{1}{1-x_2}$ where $(x_2 \cdot \partial_{x_2})^2$ is the derivation $x_2 \cdot \partial_{x_2}$ applied twice. Equation (19) has a solution in power series iff $P = 0$ has no nonnegative integer solutions: Indeed, the r.h.s. is just $\sum_{n \in \mathbb{N}^d} x^n$ and if $f \in \mathbb{C}[[x]]$ is the power series $f = \sum_{n \in \mathbb{N}^d} f_n \cdot x^n$, then the l.h.s. equals $\sum_{n \in \mathbb{N}^d} f_n \cdot P(n) \cdot x^n$. As a consequence of Hilbert 10th problem (cf. [51]), whether $P = 0$ has a nonnegative integer solution is undecidable (already for $d = 9$), and thus solvability in power series for (19) is undecidable as well.

Note that (19) is not in the CDA format. The first reason is that it is non-autonomous since it contains occurrences of the independent variables x_j's. This is inessential and it can be addressed by Remark 7. The rational power series $\frac{1}{1-x_j}$ on the r.h.s. of (19) can be eliminated by multiplying both sides of the equation by $(1-x_1)\cdots(1-x_d)$. The second reason is more fundamental: Both $\partial_{x_j} f$ and $\partial_{x_h} f$ (with $j \neq h$) appear in the same equation, which in CDA is not allowed.

As a consequence of Remark 8, in order to decide Problem 2 we need to exploit the CDA format. To this end we develop a connection between power series solutions of CDA equations (18) and shuffle automata, in a way analogous to § III-F3.

Consider an alphabet of noncommuting indeterminates $\Sigma = \{a_1, \dots, a_d\}$. The differential algebra of power series $(\mathbb{Q}[[x]]; +, \cdot, (\partial_{x_j})_{1 \le j \le d})$ and the differential shuffle algebra of *commutative* series $(\mathbb{Q}\langle\langle\Sigma\rangle\rangle^{\mathsf{comm}}; +, ⧢, (\delta^L_{a_j})_{1\le j\le d})$. The isomorphism maps a commutative series $f \in \mathbb{Q}\langle\langle\Sigma\rangle\rangle^{\mathsf{comm}}$ to the power series $\widetilde{f} = \sum_{n\in\mathbb{N}^d} \widetilde{f}_n \cdot \frac{x^n}{n!} \in \mathbb{Q}[[x]]$ with coefficients $\widetilde{f}_{n_1,\dots,n_d} := f(a_1^{n_1}\cdots a_d^{n_d})$ for every $n_1,\dots,n_d \in \mathbb{N}$. In other words, this is a bijective mapping preserving the algebraic and differential structure, i.e., $\widetilde{\mathbb{0}} = 0$, $\widetilde{1\cdot\varepsilon} = 1$, and

$$\begin{aligned} \widetilde{c\cdot f} &= c\cdot\widetilde{f}, & \widetilde{f ⧢ g} &= \widetilde{f}\cdot\widetilde{g}, \\ \widetilde{f+g} &= \widetilde{f}+\widetilde{g}, & \widetilde{\delta^L_{a_j} f} &= \partial_{x_j}\widetilde{f}, \end{aligned} \tag{20}$$

for all $f, g \in \mathbb{Q}\langle\langle\Sigma\rangle\rangle^{\mathsf{comm}}$, $c \in \mathbb{Q}$, and $1 \le j \le d$. For a system of CDA equations (18), together with an initial condition $c = (c_1, \dots, c_k) \in \mathbb{Q}^k$, we construct its *companion shuffle automaton* (Σ, X, F, Δ), which has nonterminals $X = (X_1, \dots, X_k)$, output function $FX_i := c_i$ for all $1 \le i \le k$, and transitions

$$\Delta_{a_j} X_i := p_i^{(j)}(X_1, \dots, X_k), \quad \forall 1 \le i \le k, 1 \le j \le d. \tag{21}$$

Let $f_i := [\![X_i]\!]$ be the series recognised by nonterminal X_i of the automaton, for all $1 \le i \le k$, and write $f = (f_1, \dots, f_k)$.

Lemma 13. *The initial condition c extends to a power series solution of the CDA equations if, and only if, f is a tuple of commutative series.*

Proof. On the one hand, if f is a tuple of commutative series with initial condition $[\varepsilon]f = c$ solving (21), then, by following the isomorphism, $\widetilde{f} \in \mathbb{Q}[[x]]^k$ is a tuple of power series solutions of (18) with initial condition $f(0) = c$. Indeed, by the properties of the semantics of shuffle automata (Lemma 11), component f_i satisfies differential equations

$$\delta^L_{a_j} f_i = p_i^{(j)}(f_1, \dots, f_k), \quad \forall 1 \le i \le k, 1 \le j \le d.$$

By applying the isomorphism to both sides of the equation above, and by (20), we get

$$\partial_{x_j}\widetilde{f_i} = \widetilde{\delta^L_{a_j} f_i} = \left(\widetilde{p_i^{(j)}(f_1, \dots, f_k)}\right) = p_i^{(j)}(\widetilde{f}_1, \dots, \widetilde{f}_k),$$

which is just (18). On the other hand, if $g \in \mathbb{Q}[[x]]^k$ is a tuple of power series solutions of (18) with initial condition $g(0) = c$, then by following the inverse of the isomorphism we obtain a tuple of commutative series $h \in (\mathbb{Q}\langle\langle\Sigma\rangle\rangle^{\mathsf{comm}})^k$ s.t. $\widetilde{h} = g$ and $[\varepsilon]h = c$, solving equations (21). Since equations (21) admit exactly one series solution with c as initial condition, $h = f$ is the semantics of the shuffle automaton. Consequently, f is a tuple of commutative series. □

Example 16 (Uniformably solvable systems). Consider the CDA equations from Example 13, and fix the initial condition $c := (1, 0, 0)$. The companion shuffle automaton is the one from Example 12. It can be verified that $\Delta_{a_1} \circ \Delta_{a_2} = \Delta_{a_2} \circ \Delta_{a_1}$. It suffices to consider the generators X_1, X_2, X_3:

$$\begin{aligned} &\Delta_{a_1}\Delta_{a_2} X_1 = \Delta_{a_1}(X_1 \cdot X_2) = X_1 \cdot (1 + X_3) \cdot X_2 + X_1, \\ &\Delta_{a_2}\Delta_{a_1} X_1 = \Delta_{a_2}(X_1 \cdot (1 + X_3)) = X_1 \cdot X_2 \cdot (1 + X_3) + X_1, \\ &\Delta_{a_1}\Delta_{a_2} X_2 = \Delta_{a_2}\Delta_{a_1} X_2 = 0, \\ &\Delta_{a_1}\Delta_{a_2} X_3 = \Delta_{a_2}\Delta_{a_1} X_3 = 0. \end{aligned}$$

Thus, *every* initial condition extends to a solution, i.e., Example 13 is *uniformably solvable*.

We now have all ingredients to solve Problem 2. For CDA equations (18) and initial condition $c \in \mathbb{Q}^k$ we build their companion shuffle automaton (cf. (21)), and check that all series $[\![X_1]\!], \dots, [\![X_k]\!]$ recognised by its nonterminals are commutative. The latter is decidable in 2-EXPSPACE by Theorem 7. Correctness follows from Lemma 13.

Theorem 8. *The CDA solvability problem (Problem 2) is decidable in 2-EXPSPACE.*

V. Extensions

A. Infiltration product, automata, and series

Besides Hadamard and shuffle products, in the literature one finds another product, called *infiltration*. It is denoted by $f \uparrow g$ and is defined coinductively as follows [3, Def. 8.1]:

$$\begin{aligned} &[\varepsilon](f \uparrow g) := f_\varepsilon \cdot g_\varepsilon, && (\uparrow\text{-}\varepsilon) \\ &\delta^L_a(f \uparrow g) := \delta^L_a f \uparrow g + f \uparrow \delta^L_a g + \delta^L_a f \uparrow \delta^L_a g, \ \forall a \in \Sigma. && (\uparrow\text{-}\delta^L_a) \end{aligned}$$

This is an associative and commutative operation, with the same identity $1\cdot\varepsilon$ as the shuffle product. It has been introduced by Chen, Fox, and Lyndon [16] in the context of the *free differential calculus* of Fox [33]. More recently, it has been considered in a coalgebraic setting [3, Sec. 8]. From a concurrency perspective, it models the *synchronising interleaving* where processes are allowed (but not forced) to jointly perform the input actions (thanks to the term $\delta^L_a f \uparrow \delta^L_a g$ in ($\uparrow$-δ^L_a)). E.g., in the interleaving semantics we have $ab ⧢ a = 2aab + aba$, but in the synchronising interleaving $ab \uparrow a = 2aab + aba + ab$.

We observe that the infiltration product falls under the scope of our techniques. In a manner analogous to the Hadamard and shuffle products, one can define *infiltration automata*, and corresponding *infiltration-finite series*, which are novel models not previously investigated. One can then prove that they constitute an effective prevariety, and thus the commutativity problem is decidable for this class. More details in § E.

B. Mixed product, automata, and series

One can go one step further, and consider a more general model whether the Hadamard, shuffle, and infiltration semantics are combined. Consider a finite input alphabet Σ

partitioned into three subalphabets $\Sigma = \Sigma_\odot \cup \Sigma_\sqcup \cup \Sigma_\uparrow$. Based on this partitioning, consider the binary operation on series "$\|$" (whose dependency on the partitioning is hidden for readability) s.t. $[\varepsilon](f \parallel g) = [f]\varepsilon \cdot [g]\varepsilon$, and for every $a \in \Sigma$,

$$\delta_a^L(f \parallel g) = \begin{cases} \delta_a^L f \parallel \delta_a^L g & \text{if } a \in \Sigma_\odot, \\ \delta_a^L f \parallel g + f \parallel \delta_a^L g & \text{if } a \in \Sigma_\sqcup, \\ \delta_a^L f \parallel g + f \parallel \delta_a^L g + \delta_a^L f \parallel \delta_a^L g & \text{if } a \in \Sigma_\uparrow. \end{cases}$$

This models actions in $\Sigma_\odot$ with a synchronising semantics, actions in $\Sigma_\sqcup$ with an interleaving semantics, and in $\Sigma_\uparrow$ with a synchronising interleaving semantics. This yields an associative and commutative operation "$\|$", specialising to the Hadamard $\Sigma = \Sigma_\odot$, shuffle $\Sigma = \Sigma_\sqcup$, or infiltration product $\Sigma = \Sigma_\uparrow$. Techniques developed in this paper can be applied to this more general product, obtaining a notion of $\|$*-finite series* (which therefore generalise the shuffle-finite, Hadamard-finite, and infiltration-finite series), and corresponding $\|$*-automata*, recognising the same class. Finally, $\|$-finite series are an effective prevariety, and thus equality and commutativity are decidable for them.

C. Effective algebraic varieties of commutative series

We can generalise the commutativity problem and show that the output functions leading to a commutative semantics form an *effective algebraic variety* (in the sense of algebraic geometry), as we now explain. Fix a Hadamard, shuffle, or infiltration automaton (Σ, X, Δ), where we have omitted the output function F. We consider a slightly more general class of outputs functions $F \in \mathbb{C}^X$, mapping nonterminals to complex numbers. Fix an initial configuration $\alpha \in \mathbb{Q}[X]$ and let $\mathcal{F} \subseteq \mathbb{C}^X$ be the set of output functions giving rise to a commutative semantics, i.e.,

$$\mathcal{F} := \left\{ F \in \mathbb{C}^X \;\middle|\; [\![\alpha]\!] \text{ with output function } F \text{ is commutative} \right\}.$$

$\mathcal{F}$ is the set of common zeroes of the polynomials

$$P := \{\Delta_u \alpha - \Delta_v \alpha \in \mathbb{Q}[X] \mid u, v \in \Sigma^*, u \sim v\}, \tag{22}$$

where recall that $u \sim v$ means that u, v are permutation equivalent. In other words, $\mathcal{F}$ is an *algebraic variety* [23, §2, Ch. 1] and commutativity amounts to decide whether a given output function F is in $\mathcal{F}$ (membership problem). Thanks to decidability of commutativity, there is a computable bound $N \in \mathbb{N}$ s.t. $\mathcal{F}$ is the set of common zeroes of the *finite* and *computable* set of polynomials $P_N \subseteq P$ obtained by considering words of length $\leq N$. For Hadamard automata, N is Ackermannian, and for shuffle automata it is doubly exponential. As a consequence, $\mathcal{F}$ is an *effective* algebraic variety, since we can decide $F \in \mathcal{F}$ by checking whether all polynomials in P_N vanish on F. We can use P_N to solve the following two variants of the commutativity problem:

1) Decide whether *there exists* an output function $F \in \mathbb{C}^X$ giving rise to a commutative semantics.
2) Decide whether *every* output function $F \in \mathbb{C}^X$ gives rise to a commutative semantics.

The second problem is equivalent to $P_N = \{0\}$. Thanks to *Hilbert's Nullstellensatz* [23, Theorem 1, §1, Ch. 4], the first problem is equivalent to whether the *ideal* generated by P_N does not contain 1. The latter can be established with standard techniques from computational algebraic geometry, such as *Gröbner bases* [23, Ch. 2]. More details in § F.

Acknowledgements

We warmly thank Arka Ghosh, Aliaume Lopez, and Filip Mazowiecki for valuable discussions during the preparation of this work.

References

[1] Rajeev Alur, Loris D'Antoni, Jyotirmoy Deshmukh, Mukund Raghothaman, and Yifei Yuan. Regular functions and cost register automata. In *2013 28th Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 13–22. IEEE, jun 2013. doi:10.1109/lics.2013.65.

[2] Borja Balle and Mehryar Mohri. Learning weighted automata. In *Algebraic Informatics*, pages 1–21. Springer International Publishing, 2015. doi:10.1007/978-3-319-23021-4_1.

[3] Henning Basold, Helle Hvid Hansen, Jean-Éric Pin, and Jan Rutten. Newton series, coinductively: a comparative study of composition. *Mathematical Structures in Computer Science*, 29(1):38–66, jun 2017. doi:10.1017/s0960129517000159.

[4] J. P. Bell and D. Smertnig. Computing the linear hull: Deciding deterministic? and unambiguous? for weighted automata over fields. In *2023 38th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–13, Los Alamitos, CA, USA, jun 2023. IEEE Computer Society. doi:10.1109/LICS56636.2023.10175691.

[5] Jason Bell and Daniel Smertnig. Noncommutative rational Pólya series. *Selecta Mathematica*, 27(3):34, 2021. doi:10.1007/s00029-021-00629-2.

[6] Michael Benedikt, Timothy Duff, Aditya Sharad, and James Worrell. Polynomial automata: Zeroness and applications. In *Proc. of LICS'17*, pages 1–12, June 2017. doi:10.1109/LICS.2017.8005101.

[7] François Bergeron and Christophe Reutenauer. Combinatorial resolution of systems of differential equations iii: A special class of differentially algebraic series. *European Journal of Combinatorics*, 11(6):501–512, 1990.

[8] François Bergeron and Ulrike Sattler. Constructible differentially finite algebraic series in several variables. *Theoretical Computer Science*, 144(1):59–65, 1995.

[9] J. Berstel and C. Reutenauer. *Noncommutative rational series with applications*. CUP, 2010.

[10] Vincent D. Blondel, Emmanuel Jeandel, Pascal Koiran, and Natacha Portier. Decidable and undecidable problems about quantum automata. *SIAM Journal on Computing*, 34(6):1464–1473, jan 2005. doi:10.1137/s0097539703425861.

[11] Michele Boreale, Luisa Collodi, and Daniele Gorla. Products, polynomials and differential equations in the stream calculus. *ACM Trans. Comput. Logic*, 25(1), jan 2024. doi:10.1145/3632747.

[12] Michele Boreale and Daniele Gorla. Algebra and Coalgebra of Stream Products. In Serge Haddad and Daniele Varacca, editors, *32nd International Conference on Concurrency Theory (CONCUR 2021)*, volume 203 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 19:1–19:17, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.CONCUR.2021.19.

[13] Nicolas Bourbaki. *Algebra I: Chapters 1-3*. Addison-Wesley Pub. Co., Hermann, 1974.

[14] Alex Buna-Marginean, Vincent Cheval, Mahsa Shirmohammadi, and James Worrell. On learning polynomial recursive programs. *Proceedings of the ACM on Programming Languages*, 8(POPL):1001–1027, jan 2024. doi:10.1145/3632876.

[15] Michaël Cadilhac, Filip Mazowiecki, Charles Paperman, Michał Pilipczuk, and Géraud Sénizergues. On polynomial recursive sequences. *Theory of Computing Systems*, 2021. doi:10.1007/s00224-021-10046-9.

[16] Kuo-Tsai Chen, Ralph H. Fox, and Roger C. Lyndon. Free differential calculus, iv. the quotient groups of the lower central series. *The Annals of Mathematics*, 68(1):81, jul 1958. doi:10.2307/1970044.

[17] Yu-Fang Chen, Lei Song, and Zhilin Wu. *The Commutativity Problem of the MapReduce Framework: A Transducer-Based Approach*, pages 91–111. Springer International Publishing, 2016. doi:10.1007/978-3-319-41540-6_6.

[18] Lorenzo Clemente. Weighted Basic Parallel Processes and Combinatorial Enumeration. In Rupak Majumdar and Alexandra Silva, editors, *35th International Conference on Concurrency Theory (CONCUR 2024)*, volume 311 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 18:1–18:22, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.CONCUR.2024.18.

[19] Lorenzo Clemente. Weighted basic parallel processes and combinatorial enumeration. *arXiv e-prints*, page arXiv:2407.03638, jul 2024. arXiv:2407.03638, doi:10.48550/arXiv.2407.03638.

[20] Lorenzo Clemente, Maria Donten-Bury, Filip Mazowiecki, and Michał Pilipczuk. On Rational Recursive Sequences. In Petra Berenbrink, Patricia Bouyer, Anuj Dawar, and Mamadou Moustapha Kanté, editors, *40th International Symposium on Theoretical Aspects of Computer Science (STACS 2023)*, volume 254 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 24:1–24:21, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.STACS.2023.24.

[21] Earl A. Coddington and Norman Levinson. *Theory of ordinary differential equations*. R. E. Krieger, 1984.

[22] Richard M. Cohn. *Difference Algebra*. Interscience Publishers, 1965.

[23] David A. Cox, John Little, and Donal O'Shea. *Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra*. Undergraduate Texts in Mathematics. Springer International Publishing, 4 edition, 2015.

[24] J. Denef and L. Lipshitz. Power series solutions of algebraic differential equations. *Mathematische Annalen*, 267(2):213–238, 1984.

[25] Manfred Droste, Werner Kuich, and Heiko Vogler, editors. *Handbook of Weighted Automata*. Monographs in Theoretical Computer Science. Springer, 2009.

[26] Paolo D'Alessandro, Alberto Isidori, and Antonio Ruberti. Realization and structure theory of bilinear dynamical systems. *SIAM Journal on Control*, 12(3):517–535, aug 1974. doi:10.1137/0312040.

[27] Samuel Eilenberg and Saunders Mac Lane. On the Groups $H(\Pi, n)$, I. *The Annals of Mathematics*, 58(1):55, jul 1953. doi:10.2307/1969820.

[28] Javier Esparza. Petri nets, commutative context-free grammars, and basic parallel processes. *Fundamenta Informaticae*, 31(1):13–25, 1997. doi:10.3233/fi-1997-3112.

[29] Michel Fliess. Séries formelles rationnelles et reconnaissables. *Séminaire Delange-Pisot-Poitou. Théorie des nombres*, 13(1):1–14, 1971-1972. URL: http://eudml.org/doc/110783.

[30] Michel Fliess. Sur divers produits de séries formelles. *Bulletin de la Société Mathématique de France*, 102:181–191, 1974. doi:10.24033/bsmf.1777.

[31] Michel Fliess. Un outil algebrique: Les series formelles non commutatives. In Giovanni Marchesini and Sanjoy Kumar Mitter, editors, *Mathematical Systems Theory*, pages 122–148, Berlin, Heidelberg, 1976. Springer Berlin Heidelberg. doi:10.1007/978-3-642-48895-5_9.

[32] Michel Fliess. Fonctionnelles causales non linéaires et indéterminées non commutatives. *Bulletin de la Société mathématique de France*, 79:3–40, 1981. doi:10.24033/bsmf.1931.

[33] Ralph H. Fox. Free differential calculus. i: Derivation in the free group ring. *The Annals of Mathematics*, 57(3):547, may 1953. doi:10.2307/1969736.

[34] Gustav Grabolle. A nivat theorem for weighted alternating automata over commutative semirings. *Electronic Proceedings in Theoretical Computer Science*, 346:241–257, sep 2021. doi:10.4204/eptcs.346.16.

[35] Ehud Hrushovski, Joël Ouaknine, Amaury Pouly, and James Worrell. Polynomial invariants for affine programs. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '18, pages 530–539, New York, NY, USA, 2018. Association for Computing Machinery. doi:10.1145/3209108.3209142.

[36] Ehud Hrushovski, Joël Ouaknine, Amaury Pouly, and James Worrell. On strongest algebraic program invariants. *J. ACM*, aug 2023. Just Accepted. doi:10.1145/3614319.

[37] Feidan Huang and Fasheng Cao. Weak commutativity of quantum automata. In *2021 3rd International Conference on Advances in Computer Technology, Information Science and Communication (CTISC)*, pages 62–66. IEEE, apr 2021. doi:10.1109/ctisc52352.2021.00019.

[38] Ismaël Jecker, Filip Mazowiecki, and David Purser. Determinisation and unambiguisation of polynomially-ambiguous rational weighted automata. In *Proceedings of the 39th Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '24, New York, NY, USA, 2024. Association for Computing Machinery. doi:10.1145/3661814.3662073.

[39] Irving Kaplansky. *An Introduction to Differential Algebra*. Actualites Scientifiques Et Industrielles, 1251. Hermann, 1st edition, 1957.

[40] Juhani Karhumäki. Remarks on commutative n-rational series. *Theoretical Computer Science*, 5(2):211–217, oct 1977. doi:10.1016/0304-3975(77)90008-1.

[41] Juhani Karhumäki. On commutative dt0l systems. *Theoretical Computer Science*, 9(2):2070–220, aug 1979. doi:10.1016/0304-3975(79)90025-2.

[42] E. R. Kolchin. *Differential Algebra and Algebraic Groups*. Pure and Applied Mathematics 54. Academic Press, Elsevier, 1973.

[43] Peter Kostolányi. Bideterministic weighted automata. *Information and Computation*, 295:105093, dec 2023. doi:10.1016/j.ic.2023.105093.

[44] Peter Kostolányi and Filip Mišún. Alternating weighted automata over commutative semirings. *Theoretical Computer Science*, 740:1–27, sep 2018. doi:10.1016/j.tcs.2018.05.003.

[45] Werner Kuich and Arto Salomaa. *Semirings, Automata, Languages*. EATCS Monographs on Theoretical Computer Science 5. Springer, 1986. doi:10.1007/978-3-642-69959-7.

[46] Alexander Levin. *Difference Algebra*. Algebras and applications 8. Springer, 2008.

[47] Leonard Lipshitz. D-finite power series. *Journal of Algebra*, 122(2):353–373, 1989. doi:10.1016/0021-8693(89)90222-6.

[48] B. Litow. A note on commutative multivariate rational series. *Information Processing Letters*, 87(6):283–285, sep 2003. doi:10.1016/s0020-0190(03)00347-8.

[49] Aliaume Lopez. Commutative $\mathbb{N}$-Rational Series of Polynomial Growth. In Olaf Beyersdorff, Michał Pilipczuk, Elaine Pimentel, and Nguyen Kim Thang, editors, *42nd International Symposium on Theoretical Aspects of Computer Science (STACS 2025)*, volume 327 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 67:1–67:16, Dagstuhl, Germany, 2025. Schloss Dagstuhl - Leibniz-Zentrum für Informatik. URL: https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.STACS.2025.67, doi:10.4230/lipics.stacs.2025.67.

[50] M. Lothaire. *Combinatorics on Words*. Cambridge Mathematical Library. Cambridge University Press, 2 edition, 1997.

[51] Ju. V. Matijasevič. *Enumerable sets are Diophantine*, pages 269–273. CO-PUBLISHED WITH SINGAPORE UNIVERSITY PRESS, aug 2003. doi:10.1142/9789812564894_0013.

[52] Ernst Mayr. Membership in polynomial ideals over $\mathbb{Q}$ is exponential space complete. In B. Monien and R. Cori, editors, *In Proc. of STACS'89*, pages 400–406, Berlin, Heidelberg, 1989. Springer Berlin Heidelberg. doi:10.1007/BFb0029002.

[53] Ernst W. Mayr. An algorithm for the general petri net reachability problem. In *Proc. of STOC'81*, STOC '81, pages 238–246, New York, NY, USA, 1981. ACM. doi:10.1145/800076.802477.

[54] Mehryar Mohri, Fernando Pereira, and Michael Riley. Weighted finite-state transducers in speech recognition. *Computer Speech Language*, 16(1):69–88, jan 2002. doi:10.1006/csla.2001.0184.

[55] Masakazu Nasu and Namio Honda. Mappings induced by pgsm-mappings and some recursively unsolvable problems of finite probabilistic automata. *Information and Control*, 15(3):250—-273, sep 1969. doi:10.1016/s0019-9958(69)90449-5.

[56] Klara Nosan, Amaury Pouly, Sylvain Schmitz, Mahsa Shirmohammadi, and James Worrell. On the computation of the zariski closure of finitely generated groups of matrices. In *Proceedings of the 2022 International Symposium on Symbolic and Algebraic Computation*, ISSAC '22, pages 129–138, New York, NY, USA, 2022. Association for Computing Machinery. doi:10.1145/3476446.3536172.

[57] Alan V. Oppenheim and Ronald W. Schafer. *Digital Signal Processing*. Prentice-Hall, 1 edition, 1975.

[58] Azaria Paz. *Introduction to Probabilistic Automata*. Computer Science and Applied Mathematics. Elsevier Inc, Academic Press Inc, 1971.

[59] Robin Pemantle, Mark C. Wilson, and Stephen Melczer. *Analytic Combinatorics in Several Variables - Expanded Second Edition*. Cambridge University Press, 2 edition, 2024.

[60] Gleb Pogudin, Thomas Scanlon, and Michael Wibmer. Solving difference equations in sequences: Universality and undecidability. *Forum of Mathematics, Sigma*, 8:e33, 2020. doi:10.1017/fms.2020.14.

[61] Rimhak Ree. Lie elements and an algebra associated with shuffles. *The Annals of Mathematics*, 68(2):210, sep 1958. doi:10.2307/1970243.

[62] Christophe Reutenauer. On polya series in noncommuting variables. In Lothar Budach, editor, *Fundamentals of Computation Theory, FCT 1979, Proceedings of the Conference on Algebraic, Arthmetic, and Categorial Methods in Computation Theory, Berlin/Wendisch-Rietz, Germany, September 17-21, 1979*, pages 391–396. Akademie-Verlag, Berlin, 1979.

[63] Christophe Reutenauer. Séries formelles et algèbres syntactiques. *Journal of Algebra*, 66(2):448–483, oct 1980. doi:10.1016/0021-8693(80)90097-6.

[64] Joseph Fels Ritt. *Differential Algebra*. Dover Publications Inc., 1950.

[65] Walter Rudin. *Principles of mathematical analysis*. International series in pure and applied mathematics. McGraw-Hill, 3d ed edition, 1976.

[66] Marcel Paul Schützenberger. On the definition of a family of automata. *Information and Control*, 4(2–3):245–270, sep 1961. doi:10.1016/s0019-9958(61)80020-x.

[67] Géraud Sénizergues. Sequences of level 1, 2, 3,..., k,... In *Computer Science – Theory and Applications*, pages 24–32. Springer Berlin Heidelberg, 2007. doi:10.1007/978-3-540-74510-5_6.

[68] Richard P. Stanley. *Enumerative Combinatorics*. The Wadsworth & Brooks/Cole Mathematics Series 1. Springer, 1986.

[69] Herbert S. Wilf. *generatingfunctionology*. A K Peters / CRC Press, 3 edition, 2005.

[70] Doron Zeilberger. Sister celine's technique and its generalizations. *Journal of Mathematical Analysis and Applications*, 85(1):114–145, 1982. doi:10.1016/0022-247X(82)90029-4.

Appendix A
Preliminaries

In this section, we provide more details about § I.

Lemma 1 (Left and right derivatives commute)**.** *For every two letters $a, b \in \Sigma$, we have $\delta_a^L \circ \delta_b^R = \delta_b^R \circ \delta_a^L$.*

Proof. This follows from associativity of concatenation of finite words: Indeed, for any series $f \in \mathbb{Q}\langle\langle\Sigma\rangle\rangle$ and word $w \in \Sigma^*$ we have,

$$[w]\delta_u^L\delta_v^R f = [uw]\delta_v^R f = [uwv]f, \quad \text{and}$$
$$[w]\delta_v^R\delta_u^L f = [wv]\delta_u^R f = [uwv]f.\square$$

Appendix B
Commutativity

In this section, we provide more details about § II.

Lemma 3. *For every $f \in \mathbb{Q}\langle\langle\Sigma\rangle\rangle$ and fresh input symbols $a, b \notin \Sigma$, the series $ab \sqcup\!\sqcup f$ is commutative if, and only if, $f = \mathbb{0}$.*

In the proof of the lemma we will need a simple fact about the shuffle product. Recall that a *zero divisor* in a commutative ring is an element $x \in R$ s.t. there exists a nonzero $y \in R$ with $x \cdot y = 0$. An *integral domain* is a commutative ring without nontrivial zero divisors. The ring of series with the shuffle product is an integral domain [61, Theorem 3.2], and thus it does not have nontrivial zero divisors.

Proof. Clearly if $f = \mathbb{0}$ then $ab \sqcup\!\sqcup f = \mathbb{0}$ is commutative. For the other direction, assume that f is not the zero series. Since the shuffle product does not have nontrivial zero divisors and ab is not zero, also $g := ab \sqcup\!\sqcup f$ is not zero and thus its support is nonempty. Let w be any word in the support of g, thus $g_w \neq 0$. Such a word is of the form $w = xaybz$ for some words $x, y, z \in \Sigma^*$ s.t. $f_{xyz} \neq 0$. Now consider the word $w' := xbyaz$ obtained from w by swapping a and b. By the definition of shuffle product, every word in the support of g features an a followed by a b, but x, y, z do not contain neither a or b. Thus w' is not in the support of g, i.e., $g_{w'} = 0$. Since $w \sim w'$ are commutatively equivalent, g is not commutative. □

Appendix C
Hadamard automata and series

In this section, we provide more details about § III.

A. Preliminaries

Lemma 4. *Right derivatives δ_a^R (with $a \in \Sigma$) are Hadamard algebra endomorphisms. I.e., they are linear and commute with Hadamard product,*

$$\delta_a^R(f \odot g) = \delta_a^R f \odot \delta_a^R g, \quad \forall a \in \Sigma. \tag{2}$$

Proof. Linearity follows directly from the definition of δ_a^R. Equation (2) could be proved coinductively. In fact, a direct argument suffices since the Hadamard product acts elementwise:

$$[w]\delta_a^R(f \odot g) = [wa](f \odot g) = [wa]f \cdot [wa]g =$$
$$= [w]\delta_a^R f \cdot [w]\delta_a^R g = [w](\delta_a^R f \odot \delta_a^R g).\square$$

B. Hadamard automata

Lemma 5 (Properties of the semantics)**.** *The semantics of a Hadamard automaton is a* homomorphism *from the difference algebra of polynomials $(\mathbb{Q}[X]; +, \cdot, (\Delta_a)_{a\in\Sigma})$ to the difference Hadamard algebra of series $\left(\mathbb{Q}\langle\langle\Sigma\rangle\rangle; +, \odot, (\delta_a^L)_{a\in\Sigma}\right)$. In other words, $[\![0]\!] = \mathbb{0}$, $[\![1]\!] = \mathbb{1}$, and, for all $\alpha, \beta \in \mathbb{Q}[X]$,*

$$[\![c \cdot \alpha]\!] = c \cdot [\![\alpha]\!] \qquad \forall c \in \mathbb{Q}, \tag{4}$$
$$[\![\alpha + \beta]\!] = [\![\alpha]\!] + [\![\beta]\!], \tag{5}$$
$$[\![\alpha \cdot \beta]\!] = [\![\alpha]\!] \odot [\![\beta]\!], \tag{6}$$
$$[\![\Delta_a\alpha]\!] = \delta_a^L [\![\alpha]\!] \qquad \forall a \in \Sigma. \tag{7}$$

Proof. The property $[\![0]\!] = \mathbb{0}$ is trivial, and $[\![1]\!] = \mathbb{1}$ holds by definition: $[w][\![1]\!] = F\Delta_w 1 = F1 = 1$. The property (7) holds by definition of the semantics. The other proofs follow natural coinductive arguments. Properties (4) and (5) follow from linearity of δ_a^L and Δ_a. For property (6), we argue coinductively. Clearly the constant terms agree:

$$[\varepsilon][\![\alpha \cdot \beta]\!] = F(\alpha \cdot \beta) = F\alpha \cdot F\beta = [\varepsilon][\![\alpha]\!] \cdot [\varepsilon][\![\beta]\!] =$$
$$= [\varepsilon]([\![\alpha]\!] \odot [\![\beta]\!]).$$

For the coinductive step, we have (for readability, we write α^a for $\Delta_a\alpha$ when $\alpha \in \mathbb{Q}[X]$)

$$\delta_a^L [\![\alpha \cdot \beta]\!] = [\![\Delta_a(\alpha \cdot \beta)]\!] = [\![\alpha^a \cdot \beta^a]\!] = \delta_a^L [\![\alpha]\!] \odot \delta_a^L [\![\beta]\!] =$$
$$= \delta_a^L([\![\alpha]\!] \odot [\![\beta]\!]).\square$$

C. Hadamard-finite series

The following lemma constitutes our working definition of Hadamard-finite series. We provide a full proof.

Lemma 6. *A series f is Hadamard finite iff there are generators $f_1, \dots, f_k$ s.t.:*

1) *$f \in \mathbb{Q}[f_1, \dots, f_k]_\odot$.*
2) *$\delta_a^L f_i \in \mathbb{Q}[f_1, \dots, f_k]_\odot$ for every $a \in \Sigma$ and $1 \le i \le k$. I.e., there are polynomials $p_i^{(a)} \in \mathbb{Q}[k]$ s.t.*

$$\delta_a^L f_i = p_i^{(a)}(f_1, \dots, f_k), \quad \text{for all } a \in \Sigma, 1 \le i \le k. \tag{8}$$

Moreover, we can assume without loss of generality that f equals one of the generators $f = f_1$.

Proof. If f is Hadamard finite, then the two conditions hold by definition. W.l.o.g. we can take $f = f_1$ to be one of the generators since we can just add f to the generators, without changing the validity of the two conditions.

For the other direction, assume that there are generators $f_1, \dots, f_k$ with $f = f_1$ satisfying the two conditions. We need to show that the finitely generated algebra $A := \mathbb{Q}[f_1, \dots, f_k]_\odot$ is closed under left derivatives. To this end, consider a series $g \in A$. There is a polynomial $p \in \mathbb{Q}[k]$ s.t. $g = p(f_1, \dots, f_k)$. Take now the left derivative of both sides. Since they are homomorphisms of A, we have $\delta_a^L g = \delta_a^L(p(f_1, \dots, f_k)) = p(\delta_a^L f_1, \dots, \delta_a^L f_k)$. By the second condition, $\delta_a^L f_1, \dots, \delta_a^L f_k \in A$, and thus $\delta_a^L g \in A$, as required. □

Lemma 7. *A series is recognised by a Hadamard automaton if, and only if, it is Hadamard finite.*

Proof. For the "only if" direction, let $f = [\![X_1]\!]$ be recognised by a Hadamard automaton (Σ, X, F, Δ) with nonterminals $X = (X_1, \dots, X_k)$. We show that f is Hadamard finite by applying Lemma 6. Consider series $f_i := [\![X_i]\!]$ for all $1 \le i \le k$ generating the Hadamard algebra $A := \mathbb{Q}[f_1, \dots, f_k]_\odot$. Clearly $f \in A$. Now consider generator f_i and input symbol $a \in \Sigma$, and we need to show $\delta_a^L f_i \in A$. By Lemma 5, the semantics is a homomorphism of difference algebras, and thus

$$\begin{aligned}\delta_a^L f_i = \delta_a^L [\![X_i]\!] = [\![\Delta_a X_i]\!] = (\Delta_a X_i)([\![X_1]\!], \dots [\![X_k]\!]) = \\ = (\Delta_a X_i)(f_1, \dots, f_k) \in A,\end{aligned}$$

as required.

For the "if" direction, let $f \in \mathbb{Q}\langle\langle\Sigma\rangle\rangle$ be Hadamard finite. By Lemma 6, there are series $f_1, \dots, f_k \in \mathbb{Q}\langle\langle\Sigma\rangle\rangle$ with $f = f_1$ generating a Hadamard algebra $A = \mathbb{Q}[f_1, \dots, f_k]_\odot$ s.t., for every generator f_i and input symbol $a \in \Sigma$, there is a polynomial $p_i^a \in \mathbb{Q}[X_1, \dots, X_k]$ s.t. $\delta_a^L f_i = p_i^a(f_1, \dots, f_k)$. We have all ingredients to build a Hadamard automaton recognising f_1. Consider the automaton (Σ, X, F, Δ) with nonterminals $X := (X_1, \dots, X_k)$, output mapping $F(X_i) := [\varepsilon] f_i$ and transitions

$$\Delta_a(X_i) := p_i^a(X_1, \dots, X_k), \quad \text{for all } 1 \le i \le k \text{ and } a \in \Sigma.$$

The proof is concluded by showing $[\![X_i]\!] = f_i$ for every $1 \le i \le k$. We argue coinductively. First, the two series have the same constant term by construction,

$$[\varepsilon] [\![X_i]\!] = F \Delta_\varepsilon X_i = F X_i = [\varepsilon] f_i.$$

Second, the tails agree

$$\begin{aligned}
\delta_a^L [\![X_i]\!] &= && \text{(by Lemma 5)} \\
&= [\![\Delta_a X_i]\!] = && \text{(def. of } \Delta_a) \\
&= [\![p_i^a(X_1, \dots, X_k)]\!] = && \text{(by Lemma 5)} \\
&= p_i^a([\![X_1]\!], \dots, [\![X_k]\!]) = && \text{(by coind.)} \\
&= p_i^a(f_1, \dots, f_k) = && \text{(def. of } p_i^a) \\
&= \delta_a^L f_i. \square
\end{aligned}$$

Lemma 8. *The class of Hadamard-finite series is an effective difference algebra. Moreover, if f is a Hadamard finite, then $\delta_a^R f$ is also effectively Hadamard finite.*

Proof. Closure under right derivative was shown already in the main text. Closure under scalar product is trivial. Regarding closure under addition and Hadamard product, consider two Hadamard-finite series $f, g \in \mathbb{Q}\langle\langle\Sigma\rangle\rangle$ s.t. $f \in A := \mathbb{Q}[f_1, \dots, f_k]_\odot$, $g \in B := \mathbb{Q}[g_1, \dots, g_\ell]_\odot$, and A, B are closed under left derivatives. Consider the finitely generated Hadamard algebra obtained by concatenating the generators,

$$C := \mathbb{Q}[f_1, \dots, f_k, g_1, \dots, g_\ell]_\odot.$$

Clearly $f + g, f \odot g \in C$. It remains to show that C is closed under left derivatives. But this is clear, since both A and B are.

Finally, consider closure under left derivative. We have $\delta_a^L f \in A$ since $f \in A$, A is closed under left derivatives, and δ_a^L is a homomorphism of difference algebras. □

We now prove that Hadamard-finite series over disjoint alphabets are closed under shuffle product.

Lemma 9. *Let Σ, Γ be two finite and disjoint alphabets $\Sigma \cap \Gamma = \varnothing$. If $f \in \mathbb{Q}\langle\langle\Sigma\rangle\rangle$ and $g \in \mathbb{Q}\langle\langle\Gamma\rangle\rangle$ are Hadamard finite, then $f \sqcup\!\sqcup g$ is effectively Hadamard finite.*

Proof. Recall that elements in a finitely generated Hadamard algebra over alphabet Σ are either the constant $\mathbb{0}$, the constant $\mathbb{1} := \sum_{w \in \Sigma^*} 1 \cdot w$ (the algebra identity), a generator, or are built inductively from scalar product, sum, and Hadamard product.

Let now Σ, Γ be disjoint alphabets. For a series $f \in \mathbb{Q}\langle\langle\Sigma\rangle\rangle$, let "$f \sqcup\!\sqcup \_$" be the operation $\mathbb{Q}\langle\langle\Gamma\rangle\rangle \to \mathbb{Q}\langle\langle\Sigma \cup \Gamma\rangle\rangle$ of shuffling with f applied to series in $\mathbb{Q}\langle\langle\Gamma\rangle\rangle$. In the following claims, we show that "$f \sqcup\!\sqcup \_$" distributes over the primitives of finitely generated Hadamard algebras $\subseteq \mathbb{Q}\langle\langle\Gamma\rangle\rangle$.

Distributivity over scalar product is true in general, and does not require the assumption that Σ, Γ are disjoint.

Claim. *For every series $f \in \mathbb{Q}\langle\langle\Sigma\rangle\rangle$ and $g \in \mathbb{Q}\langle\langle\Gamma\rangle\rangle$, we have*

$$f \sqcup\!\sqcup (c \cdot g) = c \cdot (f \sqcup\!\sqcup g). \tag{23}$$

Distributivity over addition is true in general, and does not require the assumption that Σ, Γ are disjoint.

Claim. *For every series $f \in \mathbb{Q}\langle\langle\Sigma\rangle\rangle$ and $g, h \in \mathbb{Q}\langle\langle\Gamma\rangle\rangle$, we have*

$$f \sqcup\!\sqcup (g + h) = f \sqcup\!\sqcup g + f \sqcup\!\sqcup h. \tag{24}$$

Finally, we have distributivity over the Hadamard product, where we rely in an essential way on the assumption that the alphabets Σ, Γ are disjoint. In the rest of the proof, for brevity we write f^a for $\delta_a^L f$.

Claim. *For every series $f \in \mathbb{Q}\langle\langle\Sigma\rangle\rangle$ and $g, h \in \mathbb{Q}\langle\langle\Gamma\rangle\rangle$ s.t. $f(\varepsilon) \in \{0,1\}$ we have*

$$f \sqcup\!\sqcup (g \odot h) = (f \sqcup\!\sqcup g) \odot (f \sqcup\!\sqcup h). \tag{25}$$

Proof of the claim. We argue coinductively. The constant terms of the two sides of (25) are $f(\varepsilon)g(\varepsilon)h(\varepsilon)$, resp., $f(\varepsilon)^2 g(\varepsilon)h(\varepsilon)$, and these two quantities are equal when $f(\varepsilon) \in \{0,1\}$. Now we take the left derivative of both sides w.r.t. $a \in \Sigma \cup \Gamma$, and apply the coinductive hypothesis:

$$\begin{aligned}
&\delta_a^L(f \sqcup\!\sqcup (g \odot h)) = \\
&= f^a \sqcup\!\sqcup (g \odot h) + f \sqcup\!\sqcup (g^a \odot h^a) = \\
&= (f^a \sqcup\!\sqcup g) \odot (f^a \sqcup\!\sqcup h) + (f \sqcup\!\sqcup g^a) \odot (f \sqcup\!\sqcup h^a), \\
&\delta_a^L((f \sqcup\!\sqcup g) \odot (f \sqcup\!\sqcup h)) = \\
&= (f^a \sqcup\!\sqcup g + f \sqcup\!\sqcup g^a) \odot (f^a \sqcup\!\sqcup h + f \sqcup\!\sqcup h^a).
\end{aligned}$$

In general the two terms above are not equal. However, since the alphabets are disjoint, they are equal in both of the two cases of interest below:

1) $a \in \Sigma$: in this case $g^a = h^a = 0$;
2) $a \in \Gamma$: in this case $f^a = 0$. □

The previous three claims come together in the following last claim.

Claim. *For every series $f \in \mathbb{Q}\langle\langle\Sigma\rangle\rangle$ s.t. $f(\varepsilon) \in \{0,1\}$, polynomial $p \in \mathbb{Q}[y_1, \dots, y_k]$ without constant term $p(0) = 0$, and series $g_1, \dots, g_k \in \mathbb{Q}\langle\langle\Gamma\rangle\rangle$, we have*

$$f \sqcup\!\sqcup p(g_1, \dots, g_k) = p(f \sqcup\!\sqcup g_1, \dots, f \sqcup\!\sqcup g_k). \tag{26}$$

Proof. Write $p(g_1, \dots, g_k)$ as

$$p(g_1, \dots, g_k) = \sum_{n_1, \dots, n_k \in \mathbb{N}} c_{n_1, \dots, n_k} \cdot g_1^{n_1} \odot \cdots \odot g_k^{n_k},$$

where $c_{n_1, \dots, n_k} \in \mathbb{Q}$ with $c_{0, \dots, 0} = 0$. By $g_i^{n_i}$ we mean the repeated Hadamard product $g_i \odot \cdots \odot g_i$ of g_i with itself n_i times. By the previous claims, we have

$$\begin{aligned}
&f \sqcup\!\sqcup p(g_1, \dots, g_k) = \\
&= f \sqcup\!\sqcup \sum_{n_1, \dots, n_k \in \mathbb{N}} c_{n_1, \dots, n_k} \cdot g_1^{n_1} \odot \cdots \odot g_k^{n_k} = \\
&= \sum_{n_1, \dots, n_k \in \mathbb{N}} f \sqcup\!\sqcup (c_{n_1, \dots, n_k} \cdot g_1^{n_1} \odot \cdots \odot g_k^{n_k}) = \\
&= \sum_{n_1, \dots, n_k \in \mathbb{N}} c_{n_1, \dots, n_k} \cdot f \sqcup\!\sqcup (g_1^{n_1} \odot \cdots \odot g_k^{n_k}) = \\
&= \sum_{n_1, \dots, n_k \in \mathbb{N}} c_{n_1, \dots, n_k} \cdot (f \sqcup\!\sqcup g_1)^{n_1} \odot \cdots \odot (f \sqcup\!\sqcup g_k)^{n_k} = \\
&= p(f \sqcup\!\sqcup g_1, \dots, f \sqcup\!\sqcup g_k). \square
\end{aligned}$$

Now suppose $f = f_1$ and $g = g_1$ are Hadamard finite, thus belonging to finitely generated difference Hadamard algebras

$$A := \mathbb{Q}[f_1, \dots, f_m]_\odot, \quad \text{resp.,} \quad B := \mathbb{Q}[g_1, \dots, g_n]_\odot.$$

By rescaling if necessary, we can assume without loss of generality that all constant terms of the generators are either zero or one $f_i(\varepsilon), g_j(\varepsilon) \in \{0,1\}$. Now consider the finitely generated Hadamard algebra

$$C := \mathbb{Q}[f_i \sqcup\!\sqcup g_j \mid 1 \le i \le m, 1 \le j \le n]_\odot.$$

Clearly $f_1 \sqcup\!\sqcup g_1$ belongs to C. We need to show that C is closed under left derivatives. Recall that $f_i^a = p_i^a(f_1, \dots, f_m)$ and $g_j^a = q_j^a(g_1, \dots, g_n)$ for polynomials p_i^a, q_j^a without constant term $p_i^a(0) = q_j^a(0) = 0$. Take now the left derivative of an arbitrary generator of C, and we have

$$\begin{aligned}
&\delta_a^L(f_i \sqcup\!\sqcup g_j) = f_i^a \sqcup\!\sqcup g_j + f_i \sqcup\!\sqcup g_j^a = \\
&= p_i^a(f_1, \dots, f_m) \sqcup\!\sqcup g_j + f_i \sqcup\!\sqcup q_j^a(g_1, \dots, g_n) = \\
&= p_i^a(f_1 \sqcup\!\sqcup g_j, \dots, f_m \sqcup\!\sqcup g_j) + q_j^a(f_i \sqcup\!\sqcup g_1, \dots, f_i \sqcup\!\sqcup g_n),
\end{aligned}$$

where in the last equality we have applied the last claim (26). It follows that $\delta_a^L(f_i \sqcup\!\sqcup g_j) \in C$, as required. □

D. *Polynomial vs. Hadamard automata*

In this section, we give more details about the relationship between polynomial and Hadamard automata, expanding on Remark 3. A *polynomial automaton* [6] is a tuple

$$A = (k, \Sigma, Q, q_I, \Delta, F) \tag{27}$$

where $k \in \mathbb{N}$ is the *dimension*, Σ is a finite *input alphabet*, $Q = \mathbb{Q}^k$ is the set of *states*, $q_I \in Q$ is the *initial state*, $\Delta : \Sigma \to Q \to Q$ is the polynomial *transition function*, and $F : Q \to \mathbb{Q}$ is the polynomial *output function*. For every input symbol $a \in \Sigma$, the polynomial map $\Delta^a : Q^k \to Q^k$ is presented as a tuple of polynomials $\Delta^a = (\Delta_1^a, \dots, \Delta_k^a) \in \mathbb{Q}[k]^k$, inducing a polynomial action on states

$$q \cdot a := \Delta^a(q) = (\Delta_1^a(q), \dots, \Delta_k^a(q)) \in Q, \quad \text{for all } q \in Q.$$

Similarly, the polynomial output function F is presented as a polynomial $F \in \mathbb{Q}[k]$ inducing the corresponding polynomial map $F(q) = F(q_1, \dots, q_k) \in \mathbb{Q}$ for every $q = (q_1, \dots, q_k) \in Q$. The action of Σ is extended to words $w \in \Sigma^*$ homomorphically: $q \cdot \varepsilon := q$ and $q \cdot (a \cdot w) := (q \cdot a) \cdot w$. The *semantics* of a state $q \in Q$ is the series defined as follows:

$$\begin{aligned}
&[\![q]\!] \in \mathbb{Q}\langle\langle\Sigma\rangle\rangle \\
&[\![q]\!](w) := F(q \cdot w), \quad \text{for every } w \in \Sigma^*.
\end{aligned}$$

The semantics of the automaton A is the series recognised by the initial state $[\![A]\!] := [\![q_I]\!]$.

Lemma 14. *A series $f \in \mathbb{Q}\langle\langle\Sigma\rangle\rangle$ is recognisable by a polynomial automaton if, and only if, its reversal f^R is Hadamard finite.*

Recall that f^R is the series that maps a word $w \in \Sigma^*$ to $f(w^R)$, where $w^R = a_n \cdots a_1$ is the reversal of $w = a_1 \cdots a_n$.

Proof. For the "only if" direction, let A be the polynomial automaton recognising f as in (27). For every $1 \leq i \leq k$, consider the series $g_i \in \mathbb{Q}\langle\langle\Sigma\rangle\rangle$ defined as

$$g_i(w) := \pi_i(q_I \cdot w^R), \quad \text{for all } w \in \Sigma^*.$$

In other words, $g_i(w)$ is the value of component i after reading w^R from the initial state. Consider now the Hadamard algebra

$$A := \mathbb{Q}[g_1, \dots, g_k]_\odot \subseteq \mathbb{Q}\langle\langle\Sigma\rangle\rangle.$$

Thanks to Lemma 6, this part of the proof is concluded by the following two claims.

Claim. *f^R is in A.*

Proof. By definition of f, we have $f = [\![q_I]\!]$. Consequently for every $w \in \Sigma^*$ we have

$$f^R(w) = F(q_I \cdot w^R) = F(\pi_1(q_I \cdot w^R), \dots, \pi_k(q_I \cdot w^R)) = \\ = F(g_1(w), \dots, g_k(w)),$$

and thus $f^R = F(g_1, \dots, g_k)$ is in A, as required. □

Claim. *For every $a \in \Sigma$ and $1 \leq i \leq k$, we have $\delta_a^L g_i \in A$.*

Proof. By the definition of g_i, for every $w \in \Sigma^*$ we have

$$(\delta_a^L g_i)(w) = g_i(a \cdot w) = \pi_i(q_I \cdot w^R \cdot a) = \Delta_i^a(q_I \cdot w^R) = \\ = \Delta_i^a(g_1(w), \dots, g_k(w)),$$

and thus $\delta_a^L g_i = \Delta_i^a(g_1, \dots, g_k)$ is in A, as required. □

For the "if" direction, let $g \in \mathbb{Q}\langle\langle\Sigma\rangle\rangle$ belong to a finitely generated Hadamard algebra $A := \mathbb{Q}[g_1, \dots, g_k]_\odot$ closed under left derivatives δ_a^L (with $a \in \Sigma$). Since $g \in A$, we can write $g = F(g_1, \dots, g_k)$ for some polynomial $F \in \mathbb{Q}[k]$. For every $1 \leq i \leq k$ and $a \in \Sigma$, the series $\delta_a^L g_i$ is in A and thus it can be written as $\Delta_i^a(g_1, \dots, g_k)$ for some polynomial $\Delta_i^a \in \mathbb{Q}[k]$. Finally, let $q_I := (g_1(\varepsilon), \dots, g_k(\varepsilon)) \in \mathbb{Q}^k$ be the tuple of constant terms of the generators. This provides us with the data required to construct a polynomial automaton of dimension k, as in (27). The proof is concluded with the following two claims.

Claim. *For every $1 \leq i \leq k$ and $w \in \Sigma^*$, we have*

$$g_i(w) = \pi_i(q_I \cdot w^R).$$

Proof. We proceed by induction on w. In the base case $w = \varepsilon$, by definition we have $\pi_i(q_I \cdot \varepsilon^R) = \pi_i(q_I) = g_i(\varepsilon)$. In the inductive case, we have

$$g_i(a \cdot w) = [w]\delta_a^L g_i = [w]\Delta_i^a(g_1, \dots, g_k) = \\ = \Delta_i^a(g_1(w), \dots, g_k(w)) = \\ = \Delta_i^a(\pi_1(q_I \cdot w^R), \dots, \pi_k(q_I \cdot w^R)) = \\ = \Delta_i^a(q_I \cdot w^R) = \pi_i(q_I \cdot w^R \cdot a) = \\ = \pi_i(q_I \cdot (a \cdot w)^R). \square$$

Claim. *We have $g = [\![q_I]\!]^R$.*

Proof. Consider an arbitrary $w \in \Sigma^*$, and we need to show $g(w) = F(q_I \cdot w^R)$. Thanks to the previous claim, we have

$$g(w) = F(g_1(w), \dots, g_k(w)) = \\ = F(\pi_1(q_I \cdot w^R), \dots, \pi_k(q_I \cdot w^R)) = \\ = F(q_I \cdot w^R). \square$$

□

E. Number sequences

In the main text we have mentioned closure properties of polyrec sequences, namely closure under scalar product, sum, Hadamard product, shifts, sections, and diagonals. The last two operations have not been defined yet. A *section* of a d-variate sequence $f : \mathbb{N}^d \to \mathbb{Q}$ is an e-variate sequence with $e < d$ which is obtained from f by fixing one or more coordinates to a constant value [47, Definition 3.1]. For instance, if $f(n_1, n_2)$ is a bivariate sequence, then $g(n_1) := f(n_1, 7)$ is a section thereof, obtained by fixing the second coordinate to be $n_2 := 7$. Since sections are defined element-wise, they commute with the operations of scalar product, sum, Hadamard product, and shifts not involving the coordinates being fixed. It follows that sections of polyrec sequences are polyrec.

Example 17. For instance, if $\sigma_1 f = \mathbb{1} - f^2$, then its section g satisfies the same equation $\sigma_1 g = \mathbb{1} - g^2$, preserving the polyrec format (information about $\sigma_2 f$ is discarded when fixing the second coordinate).

The other operation that we have not yet defined is that of taking diagonals. A *diagonal* of a d-variate sequence $f : \mathbb{N}^d \to \mathbb{Q}$ is an e-variate sequence with $e < d$ which is obtained from f by requiring a subset of the coordinates to be equal and then by projecting them to a single coordinate [47, Definition 2.6]. For instance, the sequence $h : \mathbb{N}^2 \to \mathbb{Q}$ s.t.

$$h(n_1, n_3) := f(n_1, n_1, n_3), \quad \text{for all } n_1, n_3 \in \mathbb{N},$$

is the diagonal of $f : \mathbb{N}^3 \to \mathbb{Q}$ obtained by identifying the first two coordinates. Again, this is an element-wise operation, and thus it commutes with the operations of scalar product, sum, Hadamard product, and shifts not involving the coordinates being identified. Consequently, diagonals of polyrec sequences are polyrec.

Example 18. For instance, if $\sigma_1 f = \mathbb{1} - f^2$, $\sigma_2 f = f^3$, and $\sigma_3 f = \mathbb{1} + 2 \cdot f$ then the diagonal h satisfies $\sigma_1 h = (\mathbb{1} - h^2)^3$ and $\sigma_2 h = \mathbb{1} + 2 \cdot h$, preserving the polyrec format. (By composing in the other order, we get $\sigma_1 h = 1 - (h^3)^2$, but one equation suffices: If f exists, then it will satisfy both equations.)

Appendix D
Shuffle automata and series

In this section, we provide more details about § IV.

Lemma 10. *Right derivatives δ_a^R (with $a \in \Sigma$) are shuffle algebra derivations. I.e., they are linear and satisfy Leibniz rule (12) for the shuffle product:*

$$\delta_a^R(f \sqcup\!\sqcup g) = \delta_a^R f \sqcup\!\sqcup g + f \sqcup\!\sqcup \delta_a^R g, \quad \forall a \in \Sigma. \tag{13}$$

Proof. We have already observed linearity in § I. Regarding (13), we prove it by coinduction. First of all, the constant term of both sides is the same, since

$$[\varepsilon]\delta_a^R(f \sqcup\!\sqcup g) = [a](f \sqcup\!\sqcup g) = f_a \cdot g_\varepsilon + f_\varepsilon \cdot g_a, \text{ and}$$

$$\begin{aligned}[\varepsilon](\delta_a^R f \sqcup\!\sqcup g + f \sqcup\!\sqcup \delta_a^R g) &= [\varepsilon](\delta_a^R f \sqcup\!\sqcup g) + [\varepsilon](f \sqcup\!\sqcup \delta_a^R g) = \\ &= [\varepsilon]\delta_a^R f \cdot [\varepsilon]g + [\varepsilon]f \cdot [\varepsilon]\delta_a^R g = \\ &= f_a \cdot g_\varepsilon + f_\varepsilon \cdot g_a.\end{aligned}$$

The proof is concluded by showing that, for every $b \in \Sigma$, the b-left derivative δ_b^L of both sides is the same:

$$\delta_b^L\delta_a^R(f \sqcup\!\sqcup g) = \delta_b^L(\delta_a^R f \sqcup\!\sqcup g + f \sqcup\!\sqcup \delta_a^R g).$$

We indeed have

$$\begin{aligned}&\delta_b^L\delta_a^R(f \sqcup\!\sqcup g) = \\ &= \delta_a^R\delta_b^L(f \sqcup\!\sqcup g) && \text{(by Lemma 1)}\\ &= \delta_a^R(\delta_b^L f \sqcup\!\sqcup g + f \sqcup\!\sqcup \delta_b^L g) && \text{(by } (\sqcup\!\sqcup\text{-}\delta_a^L))\\ &= \delta_a^R(\delta_b^L f \sqcup\!\sqcup g) + \delta_a^R(f \sqcup\!\sqcup \delta_b^L g) && \text{(by lin.)}\\ &= \delta_a^R\delta_b^L f \sqcup\!\sqcup g + \delta_b^L f \sqcup\!\sqcup \delta_a^R g + \\ &\quad + \delta_a^R f \sqcup\!\sqcup \delta_b^L g + f \sqcup\!\sqcup \delta_a^R\delta_b^L g && \text{(by coind.)}\\ &= \delta_b^L\delta_a^R f \sqcup\!\sqcup g + \delta_b^L f \sqcup\!\sqcup \delta_a^R g + \\ &\quad + \delta_a^R f \sqcup\!\sqcup \delta_b^L g + f \sqcup\!\sqcup \delta_b^L\delta_a^R g && \text{(by Lemma 1)}\\ &= \delta_b^L(\delta_a^R f \sqcup\!\sqcup g) + \delta_b^L(f \sqcup\!\sqcup \delta_a^R g) && \text{(by } (\sqcup\!\sqcup\text{-}\delta_a^L))\\ &= \delta_b^L(\delta_a^R f \sqcup\!\sqcup g + f \sqcup\!\sqcup \delta_a^R g) && \text{(by lin.).}\square\end{aligned}$$

Since we prove that shuffle series are a prevariety, one may wonder whether they are even a variety, in the sense of Reutenauer. Recall that varieties are prevarieties satisfying the following additional closure condition:

(V.3) For every series $f \in \mathcal{S}_\Sigma$ and algebra homomorphism[1] $\varphi : \mathbb{Q}\langle\Gamma\rangle \to \mathbb{Q}\langle\Sigma\rangle$, the series $f \circ \varphi$ is in $\mathcal{S}_\Gamma$.

In the next remark we rule out this possibility, thanks to a simple growth argument.

Remark 9 (Shuffle-finite series are not a variety). The class of shuffle-finite series is not a variety of series. This can be shown by a simple growth argument. Consider a unary input alphabet $\Sigma = \{a\}$. Univariate shuffle-finite series $f = \sum_{n\in\mathbb{N}} f_n \cdot a^n$ are in bijective correspondence with univariate CDA power series $\widetilde{f}(x) = \sum_{n\in\mathbb{N}} f_n \cdot \frac{x^n}{n!}$ [18, Lemma 25]. For the latter class it is known that $f_n \in O(\alpha^n \cdot n!)$ for some constant $\alpha > 0$ [7, Theorem 3.(i)]. Consider the shuffle-finite series s.t. $f(\varepsilon) := 1$ and $\delta_a^L f := f \sqcup\!\sqcup f$. It can be checked that $f(a^n) = n!$, and thus $\widetilde{f}(x) = \sum_{n\in\mathbb{N}} f_n \cdot \frac{x^n}{n!}$ with $f_n = n!$. Take the homomorphism $\varphi : \Sigma^* \to \Sigma^*$ defined by $\varphi(a) := aa$. We have that the composition series $g := f \circ \varphi$ satisfies $g(a^n) = f(a^{2n}) = (2n)!$. The corresponding power series is $\widetilde{g}(x) = \sum_{n\in\mathbb{N}}(2n)! \cdot \frac{x^n}{n!}$, which grows too fast in order to be CDA. Consequently g is not shuffle finite and thus shuffle-finite series are not a variety.

[1]The corresponding requirement in [63, Sec. III.1] demands closure w.r.t. algebra homomorphisms of the form $\varphi \in \mathbb{Q}\langle\langle\Gamma\rangle\rangle \to \mathbb{Q}\langle\langle\Sigma\rangle\rangle$. However $f \circ \varphi$ may not be defined when φ produces series with infinite supports. For instance take $f = \varphi(a) = 1 + a + a^2 + \cdots$. Then $(f \circ \varphi)(a) = \langle f, \varphi(a)\rangle = 1 + 1 + \cdots$ is not defined. For this reason φ needs to be restricted to be a homomorphism of series with finite supports.

Appendix E
Infiltration automata and series

In this section, we provide more details about § V-A. We introduce a model of weighted computation, called *infiltration automata*, and the associated class of *infiltration-finite series*. All results in this section are new.

A. Infiltration product

Recall that the *infiltration product* of two series $f, g \in \mathbb{Q}\langle\langle\Sigma\rangle\rangle$, denoted by $f \uparrow g$, is defined coinductively as follows:

$$[\varepsilon](f \uparrow g) = f_\varepsilon \cdot g_\varepsilon, \tag{$\uparrow$-ε}$$

$$\delta_a^L(f \uparrow g) = \delta_a^L f \uparrow g + f \uparrow \delta_a^L g + \delta_a^L f \uparrow \delta_a^L g, \ \forall a \in \Sigma. \tag{$\uparrow$-δ_a^L}$$

To see why this uniquely defines a series $f \uparrow g$, we can reason by induction on the length of words. The first rule gives us the value for the constant term $[\varepsilon](f \uparrow g)$. For nonempty words, we have

$$\begin{aligned}&[a \cdot w](f \uparrow g) = \\ &= [w](\delta_a^L(f \uparrow g)) = \\ &= [w](\delta_a^L f \uparrow g + f \uparrow \delta_a^L g + \delta_a^L f \uparrow \delta_a^L g) = \\ &= [w](\delta_a^L f \uparrow g) + [w](f \uparrow \delta_a^L g) + [w](\delta_a^L f \uparrow \delta_a^L g),\end{aligned}$$

where the three coefficients for word w are known by the inductive assumption. For example,

$$\begin{aligned}[ab](f \uparrow g) &= [b](f^a \uparrow g + f \uparrow g^a + f^a \uparrow g^a) = \\ &= [\varepsilon](f^{ab} \uparrow g + f^a \uparrow g^b + f^{ab} \uparrow g^b + \\ &\quad f^b \uparrow g^a + f \uparrow g^{ab} + f^b \uparrow g^{ab} + \\ &\quad f^{ab} \uparrow g^a + f^a \uparrow g^{ab} + f^{ab} \uparrow g^{ab}) \\ &= f_{ab} \cdot g + f_a \cdot g_b + f_{ab} \cdot g_b + \\ &\quad f_b \cdot g_a + f \cdot g_{ab} + f_b \cdot g_{ab} + \\ &\quad f_{ab} \cdot g_a + f_a \cdot g_{ab} + f_{ab} \cdot g_{ab}.\end{aligned}$$

For brevity, we use the convention that f^a denotes $\delta_a^L f$, and similarly for longer words. Infiltration product is an associative and commutative operation on series, with identity the series $1 \cdot \varepsilon$ (the same identity as for the shuffle product). We call the resulting structure $\mathbb{Q}\langle\langle\Sigma\rangle\rangle_\uparrow := (\mathbb{Q}\langle\langle\Sigma\rangle\rangle; +, \uparrow)$ the *infiltration algebra* of series.

Remark 10. We have chosen a coinductive approach to the definition of infiltration product since it brings to the foreground the similarity with Hadamard and shuffle products. Alternatively, one can define the infiltration product on finite words by induction on their length, and then lift it to series by linearity and continuity [50, page 126].

Remark 11 (Differential algebra of series). We note that the infiltration algebra has a natural differential structure. Indeed, for every $a \in \Sigma$, consider the mapping σ_a on series s.t.

$$\sigma_a f := f + \delta_a^L f, \text{ for every } f \in \mathbb{Q}\langle\langle\Sigma\rangle\rangle. \tag{28}$$

In short, we have $\sigma_a := 1 + \delta^L_a$. It is easy, albeit lengthy, to check that σ_a is an infiltration algebra endomorphism:

$$\begin{aligned}
\sigma_a(c \cdot f) &= c \cdot f + \delta^L_a(c \cdot f) = \\
&= c \cdot f + c \cdot \delta^L_a f = \\
&= c \cdot (f + \delta^L_a f) = \\
&= c \cdot \sigma_a f, \\
\sigma_a(f + g) &= f + g + \delta^L_a(f + g) = \\
&= f + \delta^L_a f + g + \delta^L_a g = \\
&= \sigma_a f + \sigma_a g. \\
\sigma_a(f \uparrow g) &= f \uparrow g + \delta^L_a(f \uparrow g) = \\
&= f \uparrow g + f^a \uparrow g + \sigma_a f \uparrow g^a = \\
&= \sigma_a f \uparrow g + \sigma_a f \uparrow g^a = \\
&= \sigma_a f \uparrow (g + g^a) = \\
&= \sigma_a f \uparrow \sigma_a g.
\end{aligned}$$

Using the definition of σ_a, the rule ($\uparrow$-δ^L_a) can be rewritten as follows:

$$\delta^L_a(f \uparrow g) := \delta^L_a f \uparrow g + \sigma_a f \uparrow \delta^L_a g \;\; \forall a \in \Sigma. \qquad (\uparrow\text{-}\delta^L_a\text{'})$$

Using differential algebra terminology, this makes δ^L_a a σ_a-*derivation* of the infiltration algebra, and thus $\left(\mathbb{Q}\langle\langle\Sigma\rangle\rangle; +, \uparrow, \sigma, \delta^L\right)$ is a σ-differential algebra. This should be compared to the fact that δ^L_a is a derivation (i.e., for the identity endomorphism) of the shuffle algebra. This helps explaining why the infiltration product behaves more similarly to the shuffle product than to the Hadamard one.

B. *Infiltration automata*

a) Syntax: An *infiltration automaton* is a tuple $A = (\Sigma, X, F, \Delta)$ where Σ is a finite *input alphabet*, $X = \{X_1, \dots X_k\}$ is a finite set of *nonterminal symbols*, $F : X \to \mathbb{Q}$ is the *output function*, and $\Delta : \Sigma \to X \to \mathbb{Q}[X]$ is the *transition function*. Thus, at the syntactic level infiltration automata contain the same information as Hadamard and shuffle automata. However, their semantics makes them different.

b) Semantics: A *configuration* is a polynomial $\alpha \in \mathbb{Q}[N]$. For every input symbol $a \in \Sigma$, the transition function $\Delta_a : X \to \mathbb{Q}[X]$ is extended to the unique linear function $\Delta_a : \mathbb{Q}[X] \to \mathbb{Q}[X]$ satisfying the following product rule, for all $\alpha, \beta \in \mathbb{Q}[X]$:

$$f(\alpha \cdot \beta) = f(\alpha) \cdot \beta + \alpha \cdot f(\beta) + f(\alpha) \cdot f(\beta). \quad \text{(infiltration)}$$

In turn, this allows us to extend transitions from single letters to all finite input words homomorphically: For every configuration $\alpha \in \mathbb{Q}[X]$, input word $w \in \Sigma^*$ and letter $a \in \Sigma$, we have $\Delta_\varepsilon \alpha := \alpha$ and $\Delta_{a \cdot w} \alpha := \Delta_w \Delta_a \alpha$. We have all ingredients to define the *semantics* of a configuration $\alpha \in \mathbb{Q}[X]$, which is the series $[\![\alpha]\!] \in \mathbb{Q}\langle\langle\Sigma\rangle\rangle$ defined as follows:

$$[\![\alpha]\!]_w := F \Delta_w \alpha, \text{ for all } w \in \Sigma^*.$$

Here, F is extended from nonterminals to configurations homomorphically $F(\alpha \cdot \beta) = F\alpha \cdot F\beta$ (i.e., polynomial evaluation). The semantics of an infiltration automaton A is the series recognised by its distinguished nonterminal $[\![X_1]\!]$.

Remark 12 (Differential algebra of polynomials). For every $a \in \Sigma$, consider the mapping $S_a : \mathbb{Q}[X] \to \mathbb{Q}[X]$ s.t.

$$S_a(\alpha) := \alpha + \Delta_a(\alpha), \text{ for all } \alpha \in \mathbb{Q}[X]. \qquad (29)$$

In short, $S_a := 1 + \Delta_a$. It can be checked that this is an endomorphism of the polynomial ring and that

$$\Delta_a(\alpha \cdot \beta) = \Delta_a \alpha \cdot \beta + S_a(\alpha) \cdot \Delta_a \beta. \qquad (30)$$

Thus Δ_a is an S_a-derivation of the polynomial ring for every $a \in \Sigma$, and thus $(\mathbb{Q}[X]; +, \cdot, (S_a)_{a \in \Sigma}, (\Delta_a)_{a \in \Sigma})$ is an S-differential algebra.

Lemma 15 (Properties of the semantics)**.** *The semantic function $[\![\_]\!]$ is a differential homomorphism from the S-differential algebra of polynomials to the σ-differential infiltration algebra of series. In other words,*

$$\begin{aligned}
[\![c \cdot \alpha]\!] &= c \cdot [\![\alpha]\!], & (31)\\
[\![\alpha + \beta]\!] &= [\![\alpha]\!] + [\![\beta]\!], & (32)\\
[\![\alpha \cdot \beta]\!] &= [\![\alpha]\!] \uparrow [\![\beta]\!], & (33)\\
[\![S_a \alpha]\!] &= \sigma_a [\![\alpha]\!], & (34)\\
[\![\Delta_a \alpha]\!] &= \delta^L_a [\![\alpha]\!]. & (35)
\end{aligned}$$

Proof. Property (35) holds by definition of the semantics. The other proofs follow natural coinductive arguments. Properties (31) and (32) follow from linearity of δ^L_a and Δ_a. From these, (34) follows easily:

$$[\![S_a \alpha]\!] = [\![\alpha + \Delta_a \alpha]\!] = [\![\alpha]\!] + [\![\Delta_a \alpha]\!] = [\![\alpha]\!] + \delta^L_a [\![\alpha]\!] = \sigma_a [\![\alpha]\!].$$

For the last property (33), we proceed coinductively. Clearly, the constant term of $[\![\alpha \cdot \beta]\!]$ and $[\![\alpha]\!] \uparrow [\![\beta]\!]$ is the same, namely $F\alpha \cdot F\beta$. For the coinductive step, we show that the left derivatives of $[\![\alpha \cdot \beta]\!]$ and $[\![\alpha]\!] \uparrow [\![\beta]\!]$ coincide. For readability, we write α^a for $\Delta_a \alpha$ when $\alpha \in \mathbb{Q}[X]$ and f^a for $\delta^L_a f$ when $f \in \mathbb{Q}\langle\langle\Sigma\rangle\rangle$.

$$\begin{aligned}
\delta^L_a [\![\alpha \cdot \beta]\!] &= [\![\Delta_a(\alpha \cdot \beta)]\!] = \\
&= [\![\alpha^a \cdot \beta + S_a \alpha \cdot \beta^a]\!] = \\
&= [\![\alpha^a \cdot \beta]\!] + [\![S_a \alpha \cdot \beta^a]\!] = \\
&= [\![\alpha^a]\!] \uparrow [\![\beta]\!] + [\![S_a \alpha]\!] \uparrow [\![\beta^a]\!] = \\
&= (\delta^L_a [\![\alpha]\!]) \uparrow [\![\beta]\!] + (\sigma_a [\![\alpha]\!]) \uparrow \delta^L_a [\![\beta]\!] = \\
&= \delta^L_a ([\![\alpha]\!] \uparrow [\![\beta]\!]). \square
\end{aligned}$$

C. *Infiltration-finite series*

In order to gain insights about the class of series recognised by infiltration automata, it is convenient to introduce the same class from a different, semantic point of view. For series $f_1, \dots, f_k \in \mathbb{Q}\langle\langle\Sigma\rangle\rangle$, we denote by $\mathbb{Q}[f_1, \dots, f_k]_\uparrow$ the infiltration subalgebra generated by the f_i's. An infiltration algebra is *differential* if it is closed under δ^L_a, for all $a \in \Sigma$.

Definition 2. A series is *infiltration finite* if it belongs to a finitely generated differential infiltration algebra of series.

The following lemma unpacks the definition above and provides our working definition for infiltration finite series. Its proof is similar to Lemma 6.

Lemma 16. *A series $f \in \mathbb{Q}\langle\langle\Sigma\rangle\rangle$ is infiltration finite iff there are generators $f_1, \ldots, f_k \in \mathbb{Q}\langle\langle\Sigma\rangle\rangle$ s.t.:*

1) $f \in \mathbb{Q}[f_1, \ldots, f_k]_\uparrow$ and
2) $\delta_a^L f_i \in \mathbb{Q}[f_1, \ldots, f_k]_\uparrow$ for every $a \in \Sigma$ and $1 \le i \le k$.

Moreover, we can assume w.l.o.g. that $f = f_1$.

The following characterisation connects infiltration automata and infiltration-finite series. Thanks to it, we can regard infiltration automata as finite syntactic presentations of infiltration-finite series.

Lemma 17. *A series is recognised by an infiltration automaton if, and only if, it is infiltration finite.*

Proof. The proof is analogous to that of Lemma 7, relying on Lemmas 15 and 16. □

D. Closure properties

In this section, we show that the series recognised by infiltration automata (equivalently, the infiltration-finite series by Lemma 17) have several pleasant closure properties. There are two ways of presenting and proving such closure properties. One way proceeds by syntactically manipulating infiltration automata. We have opted for another, more elegant approach relying on infiltration finiteness.

Lemma 18. *The class of infiltration-finite series is an effective differential infiltration algebra. In particular, infiltration-finite series are effectively closed under scalar product $c \cdot f$, sum $f + g$, infiltration product $f \uparrow g$, and left derivative $\delta_a^L f$.*

Proof. We show closure under infiltration product, the other properties being similar. Let f, g be infiltration finite with generators $f_1, \ldots, f_k$, resp., $g_1, \ldots, g_\ell$ (by applying Lemma 16). Consider the finitely generated infiltration algebra

$$A := \mathbb{Q}[f_1, \ldots, f_k, g_1, \ldots, g_\ell]_\uparrow.$$

Clearly $f \uparrow g \in A$. Moreover A is closed under δ_a^L, for all $a \in \Sigma$, since $\delta_a^L f_i$ is in $\mathbb{Q}[f_1, \ldots, f_k]_\uparrow$ for all $1 \le i \le k$ and $\delta_a^L g_j$ is in $\mathbb{Q}[g_1, \ldots, g_\ell]_\uparrow$ for all $1 \le j \le \ell$. □

Lemma 19. *The right derivative operator δ_a^R (with $a \in \Sigma$) is linear and satisfies the following property:*

$$\delta_a^R(f \uparrow g) = \delta_a^R f \uparrow g + f \uparrow \delta_a^R g + \delta_a^R f \uparrow \delta_a^R g, \quad \forall a \in \Sigma. \tag{36}$$

In other words, δ_a^R is a σ_a^R-derivation for the endomorphism $\sigma_a^R := 1 + \delta_a^R$, since we can rewrite (36) as

$$\delta_a^R(f \uparrow g) = \delta_a^R f \uparrow g + \sigma_a^R f \uparrow \delta_a^R g, \quad \forall a \in \Sigma. \tag{37}$$

Proof. We proceed by coinduction. The constant terms of the two sides coincide:

$$\begin{aligned}
&[\varepsilon](\delta_a^R(f \uparrow g)) = \\
&= [a](f \uparrow g) = \\
&= [\varepsilon](\delta_a^L(f \uparrow g)) = \\
&= [\varepsilon](f^a \uparrow g + f \uparrow g^a + f^a \uparrow g^a) = \\
&= [\varepsilon](f^a \uparrow g) + [\varepsilon](f \uparrow g^a) + [\varepsilon](f^a \uparrow g^a) = \\
&= f_a \cdot g_\varepsilon + f_\varepsilon \cdot g_a + f_a \cdot g_a = \\
&= [\varepsilon](\delta_a^R f \uparrow g) + [\varepsilon](f \uparrow \delta_a^R g) + [\varepsilon](\delta_a^R f \uparrow \delta_a^R g) = \\
&= [\varepsilon](\delta_a^R f \uparrow g + f \uparrow \delta_a^R g + \delta_a^R f \uparrow \delta_a^R g).
\end{aligned}$$

Now consider an input symbol $b \in \Sigma$, and we have

$$\begin{aligned}
&\delta_b^L \delta_a^R(f \uparrow g) = && \text{(Lemma 1)}\\
&= \delta_a^R \delta_b^L(f \uparrow g) = && \text{(def. } \uparrow)\\
&= \delta_a^R(f^b \uparrow g + f \uparrow g^b + f^b \uparrow g^b) = && \text{(lin. } \delta_a^R)\\
&= \delta_a^R(f^b \uparrow g) + \delta_a^R(f \uparrow g^b) + \delta_a^R(f^b \uparrow g^b) = && \text{(coind.)}\\
&= \delta_a^R f^b \uparrow g + f^b \uparrow \delta_a^R g + \delta_a^R f^b \uparrow \delta_a^R g + \\
&\quad + \delta_a^R f \uparrow g^b + f \uparrow \delta_a^R g^b + \delta_a^R f \uparrow \delta_a^R g^b + \\
&\quad + \delta_a^R f^b \uparrow g^b + f^b \uparrow \delta_a^R g^b + \delta_a^R f^b \uparrow \delta_a^R g^b = && \text{(Lemma 1)}\\
&= (\delta_a^R f)^b \uparrow g + f^b \uparrow \delta_a^R g + (\delta_a^R f)^b \uparrow \delta_a^R g + \\
&\quad + \delta_a^R f \uparrow g^b + f \uparrow (\delta_a^R g)^b + \delta_a^R f \uparrow (\delta_a^R g)^b + \\
&\quad + (\delta_a^R f)^b \uparrow g^b + f^b \uparrow (\delta_a^R g)^b + (\delta_a^R f)^b \uparrow (\delta_a^R g)^b = && \text{(def. } \uparrow)\\
&= \delta_b^L(\delta_a^R f \uparrow g) + \delta_b^L(f \uparrow \delta_a^R g) + \delta_b^L(\delta_a^R f \uparrow \delta_a^R g) = && \text{(lin. } \delta_b^L)\\
&= \delta_b^L(\delta_a^R f \uparrow g + f \uparrow \delta_a^R g + \delta_a^R f \uparrow \delta_a^R g). \square
\end{aligned}$$

Lemma 20. *The class of infiltration-finite series is effectively closed under right derivative $\delta_a^R f$.*

Proof. Let f be an infiltration-finite series and fix an input symbol $a \in \Sigma$. By Lemma 16, there are generators $f_1, \ldots, f_k$ with $f = f_1$ generating the δ^L-closed infiltration algebra

$$A := \mathbb{Q}[f_1, \ldots, f_k]_\uparrow.$$

We adjoin new generators $\delta_a^R f_1, \ldots, \delta_a^R f_k$, obtaining the larger infiltration algebra

$$B := \mathbb{Q}[f_1, \ldots, f_k, \delta_a^R f_1, \ldots, \delta_a^R f_k]_\uparrow.$$

By construction, $\delta_a^R f = \delta_a^R f_1$ is in B. It remains to show that B is δ^L-closed. Since A is δ^L-closed, it suffices to show this for the new generators. So consider $\delta_b^L \delta_a^R f_i$ for an arbitrary some $b \in \Sigma$ and $1 \le i \le k$. By Lemma 1, left and right derivatives commute, so we have $\delta_b^L \delta_a^R f_i = \delta_a^R \delta_b^L f_i$. Since A is δ^L-closed, $\delta_b^L f_i$ is in A, and thus we can write $\delta_b^L f_i = p_i^b(f_1, \ldots, f_k)$ for some polynomial p_i^a (where product is interpreted as infiltration product). Thanks to linearity of the right derivative operator and Lemma 19, a proof by structural induction on polynomials shows that $\delta_a^R(p_i^a(f_1, \ldots, f_k))$ can be written as a polynomial expression $q_i^{a,b}(f_1, \ldots, f_k, \delta_a^R f_1, \ldots, \delta_a^R f_k)$. We thus have, as required,

$$\delta_b^L \delta_a^R f_i = \delta_a^R \delta_b^L f_i = q_i^{a,b}(f_1, \ldots, f_k, \delta_a^R f_1, \ldots, \delta_a^R f_k) \in B. \square$$

Theorem 9. *The class of infiltration-finite series is an effective prevariety.*

Proof. Closure property **(V.1)** and closure under left derivatives have been established in Lemma 18. Closure property **(V.2)** has been established in Lemma 20. We will establish decidability of equality in Theorem 10. □

We conclude this section by showing that infiltration-finite series over disjoint alphabets are closed under shuffle product. This will be used in Lemma 24 to show that the commutativity problem generalises the equality problem.

Lemma 21. *Let Σ, Γ be two finite and disjoint alphabets $\Sigma \cap \Gamma = \varnothing$. If $f \in \mathbb{Q}\langle\langle\Sigma\rangle\rangle$ and $g \in \mathbb{Q}\langle\langle\Gamma\rangle\rangle$ are infiltration finite, then $f \sqcup\!\sqcup g$ is infiltration finite.*

The proof is different from the case of Hadamard-finite series (Lemma 9), and relies on the fact that over disjoint alphabets shuffle and infiltration products coincide.

Proof. We will use the fact that infiltration and shuffle product coincide on series with disjoint alphabets.

Claim. *For every series $f \in \mathbb{Q}\langle\langle\Sigma\rangle\rangle$ and $g \in \mathbb{Q}\langle\langle\Gamma\rangle\rangle$ over disjoint alphabets $\Sigma \cap \Gamma = \varnothing$,*

$$f \sqcup\!\sqcup g = f \uparrow g. \tag{38}$$

Proof of the claim. We proceed by coinduction. By definition we have $[\varepsilon](f \sqcup\!\sqcup g) = f_\varepsilon \cdot g_\varepsilon = [\varepsilon](f \uparrow g)$. Recall the coinductive definition of the two products:

$$\begin{aligned}\delta_a^L(f \sqcup\!\sqcup g) &= \delta_a^L f \sqcup\!\sqcup g + f \sqcup\!\sqcup \delta_a^L g, \text{ and}\\ \delta_a^L(f \uparrow g) &= \delta_a^L f \uparrow g + f \uparrow \delta_a^L g + \delta_a^L f \uparrow \delta_a^L g.\end{aligned}$$

There are two cases to consider.

1) In the first case, assume $a \in \Sigma$ and $a \notin \Gamma$. We have $\delta_a^L g = \mathbb{0}$, so the two equations reduce to

$$\delta_a^L(f \sqcup\!\sqcup g) = \delta_a^L f \sqcup\!\sqcup g \quad \text{and} \quad \delta_a^L(f \uparrow g) = \delta_a^L f \uparrow g,$$

and we can conclude since the two l.h.s. are equal by the coinductive hypothesis.

2) The second case, $a \in \Gamma$ and $a \notin \Sigma$, is dealt with similarly. □

Thanks to the claim and the fact that infiltration-finite series are closed under infiltration product (Lemma 18), we obtain the statement of the lemma. □

E. Equality problem

The equality problem for infiltration-finite series is decidable. This can be seen as a generalisation from univariate sequences (a.k.a. *streams*) $\mathbb{N} \to \mathbb{Q}$ to multivariate series $\mathbb{Q}\langle\langle\Sigma\rangle\rangle$ of the decidability result [11, Theorem 4.2] instantiated to the stream infiltration product.

Theorem 10. *The equality problem is decidable for infiltration-finite series.*

To this end, we adapt a classic technique based on chains of polynomial ideals and Hilbert's finite basis theorem; cf. analogous algorithms for polynomial automata [6], shuffle-finite series [18], and univariate infiltration-finite series [12], [11].

Fix an infiltration-finite series $f \in \mathbb{Q}\langle\langle\Sigma\rangle\rangle$ recognised by an infiltration automaton $A = (\Sigma, X, F, \Delta)$. Since equality reduces to zeroness, we show how to decide $[\![X_1]\!] = \mathbb{0}$. For a length $n \in \mathbb{N}$, consider the polynomial ideal I_n generated by all configurations reachable from X_1 by reading words of length $\le n$:

$$I_n \coloneqq \langle \Delta_w X_1 \mid w \in \Sigma^{\le n} \rangle.$$

(Recall that a *polynomial ideal* is a set of polynomials $I \subseteq \mathbb{Q}[X]$ s.t. $I + I \subseteq I$ and $I \cdot \mathbb{Q}[X] \subseteq I$. For a set of polynomials P, we write $\langle P \rangle$ for the smallest polynomial ideal containing P. We refer to [23] for more details on algebraic geometry.) The following lemma describes the relevant properties of these ideals.

Lemma 22. *1) $I_n \subseteq I_{n+1}$.*
2) $\Delta_a I_n \subseteq I_{n+1}$.
3) $I_{n+1} = I_n + \langle \Delta_a I_n \mid a \in \Sigma \rangle$.
4) $I_n = I_{n+1}$ implies $I_n = I_{n+1} = I_{n+2} = \cdots$.

Proof. The first point holds by definition.

For the second point, let $\alpha \in \Delta_a I_n$. Then α is of the form

$$\begin{aligned}\alpha &= \Delta_a \sum_{i=1}^m \beta_i \cdot \Delta_{w_i} X_1 =\\ &= \sum_{i=1}^m \Delta_a(\beta_i \cdot \Delta_{w_i} X_1) =\\ &= \sum_{i=1}^m (\Delta_a \beta_i \cdot \Delta_{w_i} X_1 + S_a \beta_i \cdot \Delta_a \Delta_{w_i} X_1) =\\ &= \sum_{i=1}^m \left(\Delta_a \beta_i \cdot \underbrace{\Delta_{w_i} X_1}_{I_n} + S_a \beta_i \cdot \underbrace{\Delta_{a \cdot w_i} X_1}_{I_{n+1}} \right),\end{aligned}$$

which is clearly in I_{n+1}.

We now consider the third point. The "$\supseteq$" inclusion holds by the first two points and the fact that I_{n+1} is an ideal. For the other inclusion, let $\alpha \in I_{n+1}$. We can write α separating words of length exactly $n+1$ from the rest, and we have

$$\begin{aligned}\alpha &= \underbrace{\beta}_{I_n} + \sum_{w \in \Sigma^{n+1}} \beta_w \cdot \Delta_w X_1 =\\ &= \beta + \sum_{a \in \Sigma} \sum_{w \in \Sigma^n} \beta_{w \cdot a} \cdot \Delta_{w \cdot a} X_1 =\\ &= \beta + \sum_{a \in \Sigma} \sum_{w \in \Sigma^n} \beta_{w \cdot a} \cdot \Delta_a \underbrace{\Delta_w X_1}_{I_n},\end{aligned}$$

where the latter quantity is clearly in the ideal generated by $\Delta_a I_n$.

The last point follows from the previous one since we have shown that I_{n+1} is a function of I_n. □

Now consider the chain of ideals

$$I_0 \subseteq I_1 \subseteq \cdots \subseteq \mathbb{Q}[X]. \tag{39}$$

By *Hilbert finite basis theorem* [23, Theorem 4, §5, Ch. 2], there is $N \in \mathbb{N}$ s.t. the chain stabilises $I_N = I_{N+1} = \cdots$. Thanks to the last point of Lemma 22, such an N is computable: Indeed, N can be taken to be the smallest $n \in \mathbb{N}$ s.t. $I_n = I_{n+1}$, and ideal equality can be decided by checking that the generators of I_{n+1} are all in I_n. The latter test is an instance of the *ideal membership problem*, which is decidable (in exponential space) [52].

The following property is easy to check.

Lemma 23. *Let $N \in \mathbb{N}$ be the stabilisation index of the ideal chain (39). $[\![X_1]\!] = \mathbb{0}$ if, and only if, $[w]\,[\![X_1]\!] = 0$ for all $w \in \Sigma^{\leq N}$.*

Proof. One direction is trivial. For the other direction we have, for every word $w \in \Sigma^*$,

$$\delta^L_w [\![X_1]\!] = [\![\Delta_w X_1]\!] = \left[\!\!\left[\sum_{u \in \Sigma^{\leq N}} \beta_u \cdot \Delta_u X_1 \right]\!\!\right] = \\ = \sum_{u \in \Sigma^{\leq N}} [\![\beta_u]\!] \uparrow [\![\Delta_u X_1]\!] \,,$$

where we have used the fact that $\Delta_w X_1$ is in I_N. By taking constant terms on both sides, we have

$$[w]\,[\![X_1]\!] = [\varepsilon]\delta^L_w [\![X_1]\!] = \sum_{u \in \Sigma^{\leq N}} [\varepsilon]\,[\![\beta_u]\!] \cdot [\varepsilon]\,[\![\Delta_u X_1]\!] = \\ = \sum_{u \in \Sigma^{\leq N}} [\varepsilon]\,[\![\beta_u]\!] \cdot [u]\,[\![X_1]\!] \,.$$

But u has length $\leq N$, and thus $[u]\,[\![X_1]\!] = 0$, implying $[w]\,[\![X_1]\!] = 0$, as required. □

Proof of Theorem 10. By using the effective closure properties of infiltration-finite series, we reduce equality $f = g$ to zeroness $f - g = \mathbb{0}$. So let f be an infiltration-finite series, recognised by an infiltration automaton A as above. Compute the stabilisation index $N \in \mathbb{N}$ of the ideal chain (39). Thanks to Lemma 23, we can decide $f = \mathbb{0}$ by enumerating all words w of length $\leq N$ and checking $[w]f = 0$. The latter test is performed by applying the semantics of the infiltration automaton. □

F. Commutativity problem

Having established that infiltration-finite series are an effective prevariety by Theorem 9, it follows from Theorem 1 that commutativity is decidable for them.

Theorem 11. *The commutativity problem is decidable for infiltration-finite series.*

In fact, commutativity is polynomial time inter-reducible with the equivalence problem.

Lemma 24. *The commutativity problem for infiltration-finite series is at least as hard as the equivalence problem, under polynomial time reductions.*

Proof. Let $f \in \mathbb{Q}\langle\!\langle\Sigma\rangle\!\rangle$ be an infiltration-finite series and consider two fresh input symbols $a, b \notin \Sigma$. Thanks to Lemma 3, $g := ab \sqcup\!\sqcup f$ is commutative if, and only if, $f = \mathbb{0}$. Moreover, since ab is infiltration finite (even rational), by Lemma 21 the series g is infiltration finite. Additionally, a finite representation for g can be constructed in polynomial time from a finite representation of f. We have reduced checking whether $f = \mathbb{0}$ to checking whether g is commutative. □

Appendix F
Effective varieties and commutativity

In this section we provide more details about § V-C. Given a set of polynomials $P \subseteq \mathbb{Q}[X]$, their *zero set* $V(P) \subseteq \mathbb{C}^k$ is the set of their common zeroes,

$$V(P) := \left\{ x \in \mathbb{C}^k \;\middle|\; \forall p \in P : p(x) = 0 \right\}.$$

Notice that $V(P)$ does not change if we close P under addition, and under product with $\mathbb{Q}[X]$. This gives rise to the notion of a *polynomial ideal* [23, §4, Ch. 1], which is a set of polynomials $I \subseteq \mathbb{Q}[X]$ s.t. $I + I \subseteq I$ and $I \cdot \mathbb{Q}[X] \subseteq I$. For a set of polynomials P, let $\langle P \rangle$ be the smallest polynomial ideal containing P. Since $V(P) = V(\langle P \rangle)$, we can consider polynomial ideals from now on.

Recall that $\mathcal{F} = V(P)$ is the zero set of the polynomials P from (22). Thus, $\mathcal{F} = V(\langle P \rangle)$, which, by definition, means that $\mathcal{F}$ is an *algebraic variety* [23, §2, Ch. 1]. By *Hilbert's finite basis theorem* [23, Theorem 4, §5, Ch. 2], $\langle P \rangle$ is finitely generated, however in general there is no computable bound on the number of generators. In our case however, we can compute finitely many generators for $\langle P \rangle$. Thanks to the proof leading to decidability of commutativity, there is a computable bound $N \in \mathbb{N}$ s.t. the ideal $\langle P \rangle$ is already generated by $P_N \subseteq P$. In other words, we have $\langle P \rangle = \langle P_N \rangle$, where

$$P_N := \left\{ \Delta_u \alpha - \Delta_v \alpha \in \mathbb{Q}[X] \;\middle|\; u, v \in \Sigma^{\leq N}, u \sim v \right\}. \tag{40}$$

Notice that P_N is finite and computable. The number N can be extracted from the proof of decidability of commutativity. For Hadamard automata, it is Ackermannian, and for shuffle automata it is doubly exponential.

The existential variant of the commutativity problem amounts to decide whether there exists an output function $F \in \mathbb{C}^k$ giving rise to a commutative semantics. In other words, this asks whether $V(\langle P_N \rangle) \neq \varnothing$. By *Hilbert's weak Nullstellensatz* [23, Theorem 1, §1, Ch. 4], this is the same as checking whether $1 \notin \langle P_N \rangle$, which can be decided with Gröbner bases [23, Ch. 2].

The universal variant of commutativity amounts to decide whether all output functions $F \in \mathbb{C}^k$ give rise to a commutative semantics. This is the same as whether $V(\langle P_N \rangle) = \mathbb{C}^k$, and thus it means that P_N contains no nontrivial constraint, i.e., $P_N = \langle P_N \rangle = \{0\}$.