Theoretical Computer Science 998 (2024) 114509

Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

Factorisation in the semiring of finite dynamical systems ☆

Émile Naquin [a], Maximilien Gadouleau [b,*]

[a] École Normale Supérieure de Lyon, France
[b] Department of Computer Science, Durham University, UK

ARTICLE INFO

Communicated by D.-Z. Du

Keywords:
Finite dynamical systems
Factorisation
Cancellative elements
Trees
Graph direct product

ABSTRACT

Finite dynamical systems (FDSs) are commonly used to model systems with a finite number of states that evolve deterministically and at discrete time steps. Considered up to isomorphism, those correspond to functional graphs. As such, FDSs have a sum and product operation, which correspond to the direct sum and direct product of their respective graphs; the collection of FDSs endowed with these operations then forms a semiring. The algebraic structure of the product of FDSs is particularly interesting. For instance, an FDS can be factorised if and only if it is composed of two sub-systems running in parallel. In this work, we further the understanding of the factorisation, division, and root finding problems for FDSs. Firstly, an FDS A is cancellative if one can divide by it unambiguously, i.e. $AX = AY$ implies $X = Y$. We prove that an FDS A is cancellative if and only if it has a fixed point. Secondly, we prove that if an FDS A has a k-th root (i.e. B such that $B^k = A$), then it is unique. Thirdly, unlike integers, the monoid of FDS product does not have unique factorisation into irreducibles. We instead exhibit a large class of monoids of FDSs with unique factorisation. To obtain our main results, we introduce the unrolling of an FDS, which can be viewed as a space-time expansion of the system. This allows us to work with (possibly infinite) trees, where the product is easier to handle than its counterpart for FDSs.

1. Introduction

Finite dynamical systems are commonly used to model systems with a finite number of states that evolve deterministically and at discrete time steps. Multiple models have been proposed for various settings, such as Boolean networks [13,14], reaction systems [9], or sandpile models [2], with applications to biology [18,19,4], chemistry [9], or information theory [11,10].

The dynamics of an FDS are easily described via its state space graph, which consists of a collection of cycles containing the periodic states, to which are attached tree-like structures containing the transient states. As such, two families of FDSs are of particular interest: permutations only have disjoint cycles in their graphs, while the so-called dendrons, where all states eventually converge towards the same fixed point, only have a tree in their graphs. Therefore, any FDS can be viewed as a collection of trees attached to a given permutation.

This article will use a terminology that is inspired from graph theory and discrete mathematics. In particular, we will simply talk about the "graph" of an FDS to mean its "state space graph". In the terminology of dynamical systems, the cycles are the attractors of the systems, while their states are the periodic points. The states of the trees attached to the partition are the transient states

☆ This article belongs to Section C: Theory of natural computing, Edited by Lila Kari.

* Corresponding author.
E-mail addresses: emile.touileb@ens-lyon.fr (É. Naquin), m.r.gadouleau@durham.ac.uk (M. Gadouleau).

https://doi.org/10.1016/j.tcs.2024.114509
Received 20 October 2022; Received in revised form 8 September 2023; Accepted 13 March 2024

Available online 21 March 2024

(except their roots). While dynamical systems are usually thought of as maps, we will switch between this functional view and a graph-theoretical view through the functional graph of the map.

Given two FDSs A and B, we can either *add* them (that is, create a system that behave like A when it starts in a state of A, and like B when it starts in a state of B) or *multiply* them (that is, create a system that corresponds to A and B evolving in parallel). Thus, the set $\mathbb{D}$ of FDSs, endowed with the sum and product above, forms a semiring.

Since the introduction of the semiring of finite dynamical systems (FDSs) in [6] as an abstract way of studying FDSs, some research has been devoted to understand more thoroughly the multiplicative structure of this semiring [8,1,5,12]. We can highlight three important problems related to the multiplicative structure of $\mathbb{D}$.

1. Perhaps the most obvious problem is factorisation: given an FDS C, can we find two non-trivial FDSs A and B (with fewer states than C) such that $C = A \times B$? This corresponds to whether the system modelled by C is actually composed of two independent parts working in parallel.
   In [8,1], it is shown that the answer is usually negative: the proportion of reducible FDSs of size n vanishes when $n \to \infty$. Moreover, unlike for integers, the semiring $\mathbb{D}$ does not have unique factorisation into irreducible elements. Worse yet, this is true when we restrict ourselves to permutations or to dendrons. This adds another layer of difficulty for problems related to factorisation in the semiring of FDSs.
2. Another important problem is division: given C and A such that $C = AB$ for some B, can we find B? Or in other words, if C is indeed composed of two parts, and we know one part, what is the other? This problem is particularly interesting, as the FDS B may not be unique: there exist many examples of FDSs A, B, D such that $AB = AD$ (an example is provided at the beginning of Section 3).
3. The third problem is k-th root: given an FDS A and an integer k, is there B such that $B^k = A$, and how many such roots exist? Until now, very little is known about this problem; for instance there was no result asserting that the solution B should be unique.

In this paper, we establish important connections between FDSs and infinite, periodic trees. In particular, we introduce the unrolling of an FDS, which can be viewed as a space-time expansion of the system. The unrolling preserves all the information about the transient dynamics of an FDS, and preserves the product operation. However, the product on trees (and in particular, on unrollings of FDSs) is much better behaved than its counterpart for FDSs and hence allows us to prove our main results.

This paper makes four main contributions towards the understanding of the three problems listed above.

1. An FDS is connected if its graph is connected; in other words, it has only one periodic cycle. We first prove a fundamental property of connected FDSs. For any FDS A, if X and Y are connected and $AX = AY$, then $X = Y$. Intuitively, this means that division is unambiguous when we know the quotient is connected.
2. Intuitively, a cancellative FDS is one those that can be unambiguously divided by. Formally, A is cancellative if $AB = AC$ implies $B = C$. Our first and major result is the characterisation of cancellative FDSs: they are exactly those with a fixed point. This result is close that Theorem 8 of [17], but not equivalent, since Lovász's paper studies cancellativity in the semiring of digraphs (and thus, there could be an FDS that is cancellative on $\mathbb{D}$ but not as a general digraph).
3. Our proof methods involve working with (possibly infinite) trees, and going back and forth between FDSs and trees. As a by-product, we obtain an algorithm for division of dendrons. That is, given two dendrons A and B, the algorithm determines the dendron C such that $A = BC$ or returns a failure if no such dendron exist. It is easily shown that this algorithm runs in time polynomial in the size of the input.
4. Our result on cancellative FDSs has an important consequence: many polynomials in $\mathbb{D}[X]$ are injective. We then prove that the polynomial $P(X) = X^k$ is injective, i.e. if an FDS has a k-th root then it is unique.
5. Throughout the paper, we investigate the structure of division and factorisation in dendrons. We further this investigation by exhibiting large monoids of dendrons with unique factorisation.

While writing up this paper, we have discovered the related paper [7]. This work and [7] have two main similarities: both have independently introduced the unrolling construction, and both have proved the cancellative nature of the product on infinite trees (Lemma 22 in this work, Theorem 3.3 in [7]). However, we would like to stress the significant differences between these two papers. First, the respective proofs of the result mentioned above are completely different: ours is based on a lexicographic order on trees, while theirs is based on counting tree homomorphisms. Second, and more importantly, both papers consider completely different problems about FDSs. As such, the last four main contributions of this work (items 2 to 5 in the list above) are novel and do not appear in the literature so far. Third, [7] proposes the following conjecture (Conjecture 3.1): let A and B be two connected FDSs, then for all FDSs X and Y, if $AX = B$ and $AY = B$, then $X = Y$. Our first main contribution (item 1 in the list above) was added once we were aware of [7]; it is actually a more general result than their conjecture.

The rest of the paper is organised as follows. Section 2 introduces all the necessary definitions to work on the semiring of FDSs. Then, Section 3 shows that the cancellative elements of the semiring of finite dynamical systems are exactly those with a fixed point. From this, we give in Section 4 a polynomial-time algorithm for division in dendrons. Section 5 then proves the unicity of k-th roots. Then Section 6 constructs a class of monoids with unique factorisation on each of them. Finally, some avenues for further work are proposed in Section 7.

2. General definitions

A finite dynamical system (FDS) is a function from a finite set into itself. We denote by $\mathbb{D}$ the set of all FDSs. Given an FDS A, we denote by S_A the finite set on which it acts, also called the state space in standard dynamical system terminology.

We note that our definition of an FDS does not technically match the classical definition of a dynamical system. However, it is easily shown that a discrete-time dynamical system over a finite space is fully characterised by a self-map over a finite set. Moreover, our definition is equivalent to the definition of a topological dynamical system over a finite set, equipped with the discrete topology. As such, our definition of an FDS truly encapsulates the dynamical properties of any dynamical system defined over a finite set.

Given two FDSs A and B, we can assume that $S_A \cap S_B = \varnothing$ (if that is not the case, we can simply rename the elements of one of those sets). Then, we define their *sum* as follows:

$$\begin{array}{rccl} A+B: & S_A \sqcup S_B & \to & S_A \sqcup S_B \\ & x & \mapsto & \begin{cases} A(x) & \text{if } x \in S_A \\ B(x) & \text{otherwise.} \end{cases} \end{array}$$

Given two FDSs A and B, we define their *product* as follows:

$$\begin{array}{rccl} AB: & S_A \times S_B & \to & S_A \times S_B \\ & (a,b) & \mapsto & (A(a), B(b)). \end{array}$$

Defining the size of an FDS A as $|A| = |S_A|$, we see that: $|A+B| = |A| + |B|$ and $|AB| = |A||B|$.

When multiplying two FDSs A and B (for example, in Fig. 1), we get AB along with a labelling of the states of AB by pairs of states of A and of B. That is, we get an isomorphism $S_{AB} \simeq S_A \times S_B$ that respects the product structure. However, we shall consider FDSs up to isomorphism, hence we do not get this labelling along with the FDS in general. The problem of factorising an FDS C, for example, just means labelling the states S_C with $S_A \times S_B$, where $A, B \in \mathbb{D}$, such that this labelling respects the product $AB = C$. A formalization of this idea of labelling the states of a product with the Cartesian product of the state sets of its factors is provided below:

Definition 1. Given a sequence $(A_i)_{i\in I} \in \mathbb{D}^I$ for some finite set I and a product $B = \prod_{i\in I} A_i$ we say that the function $\phi : S_B \mapsto \prod_{i\in I} S_{A_i}$ is a *product isomorphism for the product* $B = \prod_{i\in I} A_i$ if:

1. it is a bijection, and
2. for any sequence of states $(s_i)_{i\in I} \in \prod_{i\in I} S_{A_i}$, we have: $B(\phi^{-1}((s_i)_{i\in I})) = \phi^{-1}((A_i(s_i))_{i\in I})$.

We remark that computing the product gives such a product isomorphism.

Proposition 2 *([6]). The set of FDSs, with the above sum and product, forms a semiring [16], with additive identity the empty function and multiplicative identity the function* $1 : \{1\} \to \{1\}, 1 \mapsto 1$.

For FDSs, we can adopt a graph-theoretical point of view, by associating to an FDS $A \in \mathbb{D}$ an oriented graph $\mathcal{G}_A = (V, E)$ where $V = S_A$ and $E = \{(x, y) \in S_A^2 : y = A(x)\}$. Then, for $A, B \in \mathbb{D}$, $\mathcal{G}_{A+B}$ is the disjoint union of $\mathcal{G}_A$ and $\mathcal{G}_B$, and $\mathcal{G}_{AB}$ is the direct product $\mathcal{G}_A \times G_B$ (see the corresponding section in [15]). In the following, we will often identify an FDS and its graph, and thus, implicitly quotient $\mathbb{D}$ by graph isomorphism, that is, we consider that $A = B$ if and only if $\mathcal{G}_A$ and $\mathcal{G}_B$ are isomorphic, as A and B have the same dynamics in this case.

Definition 3. Given $A, B \in \mathbb{D}$, we say that B is a sub-FDS of A if $\mathcal{G}_B$ is a subgraph of $\mathcal{G}_A$.

Since FDSs take their values in a finite set, the structure of their graphs is simple: they consist of some cycles, on the states of which, trees (with arrows going upwards, towards the root) are connected, as in the example of Fig. 1. This leads us to several definitions that are useful to study FDSs.

Definition 4. Let $A \in \mathbb{D}$. A state $s \in S_A$ is said to be a *cycle state* if it is on a cycle of $\mathcal{G}_A$, or, equivalently, if there exists $n > 0$ such that $A^n(s) = s$. We denote by S_A^C the set of cycle states of A. Otherwise, s is said to be a *tree state*.

We define a function $\mathrm{depth}_A : S_A \to \mathbb{N}$ that gives the *depth* of any state of A, and is defined recursively as follows:

$$\begin{aligned} \forall s \in S_A^C,\ & \mathrm{depth}_A(s) = 0 \\ \forall s \in S_A \setminus S_A^C,\ & \mathrm{depth}_A(s) = \mathrm{depth}_A(A(s)) + 1. \end{aligned}$$

Furthermore, for any $k \in \mathbb{N}$, we define the *truncature of A at depth k*, denoted by $[A]_k$, as the sub-FDS of A which contains all the states of A at depth at most k.

A very useful and simple result is the following:

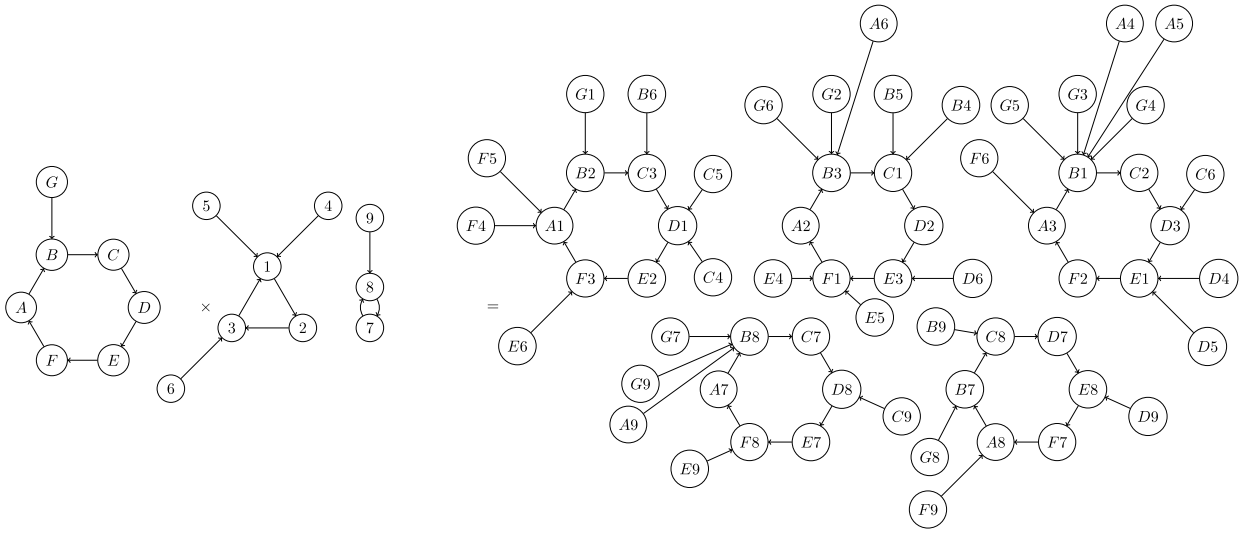


Fig. 1. Product of two FDSs. For instance, the transitions $B \to C$ in the first factor and $1 \to 2$ in the second factor combine to form the transition $(B,1) \to (C,2)$ in the product, written as $B1 \to C2$.

Lemma 5. *For any $A, B \in \mathbb{D}$ and $k \in \mathbb{N}$, $[AB]_k = [A]_k[B]_k$.*

Of particular interest are the FDSs we call *dendrons*, that is, connected FDSs (i.e. with a connected graph) with a fixed point. Those FDS can be seen as rooted trees with arrows pointing towards the root, with a loop on the root. We denote $\mathbb{D}_D$ the set of dendrons (remark that it is not a semiring, since the sum of two dendrons is not a dendron).

Let's now focus on those two types of parts of FDSs: trees (which, when summed, form forests) and cycles (which, when summed, form permutations).

2.1. Forests

We introduce forests as a way to have a product between FDSs that has an inductive definition that works level by level. In FDSs, the state set of a product is the Cartesian product of the state sets of the factors. This makes the identification of states of an unlabelled FDS difficult. For forests, the pairs of states which end up in the product are those of even depth. Finally, Lemma 30 is the reason forests are useful: their product is compatible with that of FDSs.

Definition 6. By *tree*, we shall mean an in-tree [3, p.21], i.e. an oriented connected acyclic graph with a special vertex called its *root*, such that every edge is oriented towards the root. The trees we consider may be infinite, but the degree of each vertex shall always be finite.

We denote the root of a tree $\mathbf{T}$ as $\mathrm{root}(\mathbf{T})$.

A *forest* is a disjoint union of trees. The set of forests is denoted by $\mathbb{F}$, and that of trees is denoted $\mathbb{F}_T$. In the following, we denote forests in bold face to distinguish them from FDSs.

If $\mathbf{T} \in \mathbb{F}_T$, and if there is a single infinite path starting from the root of $\mathbf{T}$, we can extract the sequence $tseq(\mathbf{T})$ of trees anchored on that path. If this sequence is periodic, we say that $\mathbf{T}$ is *periodic*, and that $\mathbf{T}$ is of *tree period* the period of the sequence. We denote the set of periodic trees as $\mathbb{F}_P$.

We consider trees as dendrons which have had their fixed point transformed into a sink, and extend the notations from dendrons whenever they make sense. In particular, we denote by $S_{\mathbf{A}}$ the set of vertices of the forest $\mathbf{A}$. Moreover, the parent of a vertex $x \in S_{\mathbf{A}}$ is denoted by $\mathbf{A}(x)$. For an FDS $A \in \mathbb{D}$ and a state $s \in S_A$, we say that $\mathbf{T}$ is the tree *anchored on* s if the tree of the tree state predecessors of s in the graph is $\mathbf{T}$; we naturally extend this notation to any forest $\mathbf{A}$. By convention, the depth of an infinite dendron is ∞, while the depth of an empty dendron is -1.

Given a tree $\mathbf{T}$, we define $\mathcal{D}(\mathbf{T})$ to be the multiset containing the subtrees anchored on the children of the root of $\mathbf{T}$.

Now, we define a sum and a product operation on forests in order to endow the set of forests with a semiring structure.

The *sum* of two forests $\mathbf{A}, \mathbf{B}$ (for which we can assume $S_{\mathbf{A}} \cap S_{\mathbf{B}} = \varnothing$) is the forest $\mathbf{C}$ defined as the disjoint union of the graphs $\mathbf{A}$ and $\mathbf{B}$.

Let $\mathbf{A}, \mathbf{B} \in \mathbb{F}_T$. Then the *product* of $\mathbf{A}$ and $\mathbf{B}$ is $\mathbf{AB} = (V, A)$ with

$$V = S_{\mathbf{AB}} = \{(a,b) \in S_{\mathbf{A}} \times S_{\mathbf{B}} : \mathrm{depth}_{\mathbf{A}}(a) = \mathrm{depth}_{\mathbf{B}}(b)\},$$
$$A = \{((a,b), (\mathbf{A}(a), \mathbf{B}(b))) : (a,b) \in S_{\mathbf{AB}}\}.$$

This product is almost the same as that on FDSs but here only states of same depth get multiplied together.

It will often prove useful to use multisets with the following product. Given two multisets $\mathcal{A}$ and $\mathcal{B}$, their *product* $\mathcal{AB}$ is $\{\{ab : a \in \mathcal{A}, b \in \mathcal{B}\}\}$.

The following lemma explains why trees are interesting: the product is done level by level. Moreover, the root does not behave differently than the other states (as it does on dendrons), which means that this product is much easier to work with.

Lemma 7. *If $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{F}_T$ are finite, then:*

$$\mathcal{D}(\boldsymbol{AB}) = \mathcal{D}(\boldsymbol{A})\mathcal{D}(\boldsymbol{B}) = \{\{\boldsymbol{T} \times \boldsymbol{T'} : \boldsymbol{T} \in \mathcal{D}(\boldsymbol{A}), \boldsymbol{T'} \in \mathcal{D}(\boldsymbol{B})\}\}.$$

Proof. The proof is by induction on the depths of **A** and **B**. The case for trees with depth ≤ 1 is trivial. The depth 1 vertices of **AB** form the set $\{(a,b) \in S_{\mathbf{A}} \times S_{\mathbf{B}} : \mathrm{depth}_{\mathbf{A}}(a) = \mathrm{depth}_{\mathbf{B}}(b) = 1\}$. We simply show that the tree $\mathbf{T}_{(a,b)}$ anchored on (a,b) in **AB** is the product of the tree $\mathbf{T}_a$ anchored on a in **A** with the tree $\mathbf{T}_b$ anchored on b in **B**. By induction, we know that $\mathcal{D}(\mathbf{T}_{(a,b)}) = \mathcal{D}(\mathbf{T}_a)\mathcal{D}(\mathbf{T}_b)$, so $\mathbf{T}_{(a,b)} = \mathbf{T}_a\mathbf{T}_b$. This concludes the proof. □

It is easy to verify that the set of forests becomes a semiring with these operations:

Lemma 8. *The set $\mathbb{F}$ of forests becomes a semiring when endowed with the sum and product defined above. Its additive identity is $\boldsymbol{0}$, the empty tree with $(V = \emptyset, A = \emptyset)$, while its multiplicative identity is the rooted infinite directed path $\boldsymbol{P}_\infty$ with $(V = \mathbb{N}, A = \{(n+1, n) | n \in \mathbb{N}\})$.*

A straightforward inductive proof gives the following lemma:

Lemma 9. *If $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{F}$, then for $a \in S_{\boldsymbol{A}}, b \in S_{\boldsymbol{B}}$ such that $(a,b) \in S_{\boldsymbol{AB}}$, we have* $\mathrm{depth}_{\boldsymbol{AB}}((a,b)) = \mathrm{depth}_{\boldsymbol{A}}(a) = \mathrm{depth}_{\boldsymbol{B}}(b)$.

Lemma 10. *If $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{F}$, then* $\mathrm{depth}(\boldsymbol{AB}) = \min(\mathrm{depth}(\boldsymbol{A}), \mathrm{depth}(\boldsymbol{B}))$.

Proof. We have:

$$S_{\mathbf{AB}} = \{(a,b) \in S_{\mathbf{A}} \times S_{\mathbf{B}} : \mathrm{depth}_{\mathbf{A}}(a) = \mathrm{depth}_{\mathbf{B}}(b)\}.$$

From Lemma 9, a state $(a,b) \in S_{\mathbf{A}} \times S_{\mathbf{B}}$ has depth at most $\min(\mathrm{depth}_{\mathbf{A}}(a), \mathrm{depth}_{\mathbf{B}})$. Moreover, if $k = \min(\mathrm{depth}(\mathbf{A}), \mathrm{depth}(\mathbf{B}))$ and we let $a \in S_{\mathbf{A}}, b \in S_{\mathbf{B}}$ two states that have both depth k in their respective trees, then (a,b) has depth k too. □

2.2. Permutations

For every $k \geq 1$, we denote by C_k the *cycle of length* k defined as the FDS whose graph is the directed cycle of length k. We say that $A \in \mathbb{D}$ is a *permutation* if the function A is bijective. In that case, all the states of A are cycle states. We denote the semiring of permutations by $\mathbb{D}_P$ (it has multiplicative identity C_1 and additive identity the empty function). In particular, for any $A \in \mathbb{D}$, $[A]_0 \in \mathbb{D}_P$. In the dynamical system formalism, permutations are called *invertible*, although we will not use this terminology here.

We introduce two shortened notations: $a \vee b = \mathrm{lcm}(a,b)$ and $a \wedge b = \gcd(a,b)$. In [6], the following very useful and simple result is proven:

Lemma 11. $C_a \times C_b = (a \wedge b)C_{a \vee b}$.

We now extend it to arbitrary products of cycles. For any multiset J of positive integers, we let $\bigwedge J = \bigwedge_{j \in J} j$ and $\bigvee J = \bigvee_{j \in J} j$; if J is empty, then those terms are equal to 1.

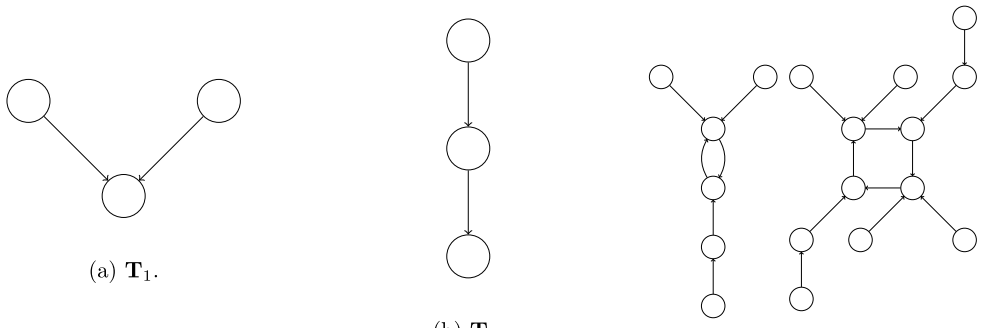
Lemma 12. *Let J be a multiset of positive integers. Then $\prod_{j \in J} C_j = \delta_J C_{\bigvee J}$, where δ_J is recursively defined as $\delta_\emptyset = 1$ and for any $a \in \mathbb{N}$*

$$\delta_{J \cup \{a\}} = (a \wedge \bigvee J)\delta_J.$$

Proof. The proof is by induction on the cardinality of J. The result is clear when J is empty. Assume it is true for J, and let $a \in \mathbb{N}$. Then

$$\prod_{j \in J \cup \{a\}} C_j = C_a \prod_{j \in J} C_j = \delta_J C_a C_{\bigvee J} = \delta_J (a \wedge \bigvee J) C_{\bigvee J \cup \{a\}}. \quad \square$$

Given a permutation $A \in \mathbb{D}_P$, such that the length of each of its cycles is a multiple of some $k \in \mathbb{N}$, and k trees $\mathbf{T}_0, \dots, \mathbf{T}_{k-1} \in \mathbb{F}_T$, we denote by $A(\mathbf{T}_0, \dots, \mathbf{T}_{k-1})$ the FDS obtained by taking each cycle of A, traversing it by following the arrows, and anchoring on the i-th state encountered the dendron $T_{i \mod k}$. This is pictured in Fig. 2.



(a) $\mathbf{T}_1$. (b) $\mathbf{T}_2$. (c) $A(\mathbf{T}_1, \mathbf{T}_2)$.

Fig. 2. The FDS $A(\mathbf{T}_1, \mathbf{T}_2)$ for $A = C_2 + C_4$, and two trees $\mathbf{T}_1, \mathbf{T}_2$.

We use the following notation: for $A \in \mathbb{D}$, and for all $i \in \mathbb{N}$, we denote by λ_i^A the number of cycles of length i in A.

Finally, given an FDS $A \in \mathbb{D}$, and a set $L \subseteq \mathbb{N}$, we define the *L-support* of A, denoted by $\mathrm{supp}_L(A)$, as the FDS made of the connected components of A with cycle size in L.

The following two results will prove useful to understand the product of a permutation with a dendron.

Lemma 13. *For any $\ell, k \geq 1$ and trees $\boldsymbol{T}_1, \dots, \boldsymbol{T}_k \in \mathbb{F}_T$, $C_k(\boldsymbol{T}_1, \dots, \boldsymbol{T}_k) \times C_\ell = (C_k C_\ell)(\boldsymbol{T}_1, \dots, \boldsymbol{T}_k)$.*

Proof. Take a product isomorphism for the product $C_k \times C_\ell = (k \wedge \ell) C_{k \vee \ell}$, and write $S_{(k\wedge\ell)C_{k\vee\ell}} \simeq S_{C_k} \times S_{C_\ell}$ accordingly. Then, take $(i, j) \in S_{(k\wedge\ell)C_{k\vee\ell}}$. Let's show that the tree that is anchored on this state in $S_{C_k(\mathbf{T}_1,\dots,\mathbf{T}_k)\times C_\ell}$ is the tree that is anchored on i in $C_k(\mathbf{T}_1, \dots, \mathbf{T}_k)$, say $\mathbf{T}_i$. Indeed, since C_ℓ has no tree states, the tree states over (i, j) have a first component with is a tree state, and a second one which is a cycle state. But since each state of C_ℓ has exactly one predecessor, and the tree anchored on (i, j) is indeed $\mathbf{T}_i$. This proves the result. □

Corollary 14. *For any $A \in \mathbb{D}_P$ and trees $\boldsymbol{T}_1, \dots, \boldsymbol{T}_k \in \mathbb{F}_T$, $C_k(\boldsymbol{T}_1, \dots, \boldsymbol{T}_k) \times A = (C_k A)(\boldsymbol{T}_1, \dots, \boldsymbol{T}_k)$.*

Proof. Let's write $A = \sum_{i\in\mathbb{N}} \lambda_i^A C_i$. Then,

$$C_k(\mathbf{T}_1, \dots, \mathbf{T}_k) \times A = \sum_{i\in\mathbb{N}} \lambda_i^A (C_k C_i)(\mathbf{T}_1, \dots, \mathbf{T}_k) = (C_k A)(\mathbf{T}_1, \dots, \mathbf{T}_k)$$

from the previous lemma. □

3. Cancellative finite dynamical systems

It is known that the division operation can sometimes not yield a unique result; a well-known example is: $C_2^2 = 2C_2$. We can also show that if we have $AB = AC$ for $A, B, C \in \mathbb{D}$, even assuming $[B]_0 = [C]_0$ does not guarantee that $B = C$, since, given two different trees $\mathbf{T}_1$ and $\mathbf{T}_2$, we have the identity $C_2(2\mathbf{T}_1 + C_2(\mathbf{T}_2)) = C_2(C_2(\mathbf{T}_1) + 2\mathbf{T}_2)$. We therefore consider the elements for which division is unambiguous, defined as follows.

Definition 15 *(Cancellative element).* An FDS $A \in \mathbb{D}$ is said *cancellative* if for all $B, C \in \mathbb{D}$, $AB = AC \implies B = C$.

In this section, we prove that an FDS is cancellative if and only if it has a fixed point (Theorem 34). We approach this theorem in steps. First, we introduce an order on trees, based on a code for trees. Then, we move on to show that we can transform FDSs into forests in a way that works well with both the product on forests and on FDSs. Finally, we prove the theorem.

3.1. Order on trees

It will prove very useful to have a total order on finite trees that is compatible with the product. That is, if $\mathbf{T}_1, \mathbf{T}_2, \mathbf{T}_3, \mathbf{T}_4 \in \mathbb{F}_T$, and $\mathbf{T}_1 < \mathbf{T}_2$ and $\mathbf{T}_3 \leq \mathbf{T}_4$, we want to have $\mathbf{T}_1\mathbf{T}_3 < \mathbf{T}_2\mathbf{T}_4$. This will be guaranteed by Corollary 20.

To do so, we define a code C_f from finite trees to $\mathbb{N}^*$ (the set of finite sequences of nonnegative integers), and we say that $\mathbf{T}_1 < \mathbf{T}_2 \iff C_f(\mathbf{T}_1) <_{\mathrm{lex}} C_f(\mathbf{T}_2)$ (where $<_{\mathrm{lex}}$ is the lexicographical order).

The code is computed as follows, using two mutually recursive functions. We consider for a moment that trees are ordered: the children of a node are stored in an ordered list, say, from left to right. That is, for a tree $\mathbf{T}$, $D(\mathbf{T})$ is now a tuple rather than a multiset. Then, we define a procedure *collect* which takes a finite tree, sorts it (using the function *sort* defined below), and then traverses

```
if |T| > 1 then
  T ← sort(T);
end
t = [];
for i ∈ [[0, depth(T)]] do
  for v in T's depth i, from left to right do
    d ← number of children of v;
    t ← t :: d;
  end
end
return t;
```

(a) $collect(\mathbf{T})$ (:: is the concatenation operator)

```
(T_1, ..., T_n) := D(T);
for i ∈ [[1, n]] do
  c_i ← collect(T_i);
end
(U_1, ..., U_n) ← sort (T_1, ..., T_n) by
 increasing (c_1, ..., c_n);
Let T' such that D(T') = (U_1, ..., U_n);
return T'.
```

(b) $sort(\mathbf{T})$

Fig. 3. The two mutually recursive functions for computing C_f.

level by level, following the order of the predecessors, starting from depth 0, and outputs a tuple of the number of predecessors of each node encountered.

We also define a procedure *sort* that takes a finite tree $\mathbf{T}$, begins by calling *collect* on each of the subtrees anchored on direct predecessors of the root, and then order those predecessors from left to right by increasing return value of *collect*. Finally, $C_f(\mathbf{T}) = collect(\mathbf{T})$.

A pseudocode implementation of *sort* and *collect* is found in Fig. 3.

Example 16. The tree in Fig. 4a has the code $(2,0,2,0,0)$; its states are traversed in the following order: A, B, C, D, E in the topmost call to *collect*.

Lemma 17. *The code C_f is prefix-free. That is, if $\mathbf{T}, \mathbf{T'} \in \mathbb{F}_T$ are such that $C_f(\mathbf{T})$ is a prefix of $C_f(\mathbf{T'})$, we have $C_f(\mathbf{T}) = C_f(\mathbf{T'})$.*

Proof. Given a code c, and an index i, write $\delta(c,i) = \sum_{j=1}^{i} c_j - i$. This is the number of vertices that have been announced as children of vertices in $c_1, \dots, c_i$ but which are not themselves in $c_1, \dots, c_i$. Thus, if we are reading a code c, and we have read the i first elements, we know that we must read at least $\delta(c,i)$ other elements. Moreover, remark that if $\delta(c,i) = 0$, then we are at the end of the code, since we have already read the children of every vertex.

Now, suppose that $c := C_f(\mathbf{T})$ is a prefix of $c' := C_f(\mathbf{T'})$. Let $i = |c|$: we have $\delta(c,i) = 0$ since c is completely read once we have read the i first elements. Moreover, we must have $\delta(c',i) = \delta(c,i)$ since $c'_1 \dots c'_i = c_1 \dots c_i$. So, $\delta(c',i) = 0$ too, and thus, $c = c'$. □

We now say that, for two trees $\mathbf{T}, \mathbf{T'}$, we have $\mathbf{T} \leq_f \mathbf{T'}$ if $C_f(\mathbf{T}) \leq_{\text{lex}} C_f(\mathbf{T'})$. We claim that this defines a total order on trees. Reflexivity and transitivity are trivial, and its antisymmetry is guaranteed by the following lemma:

Lemma 18. *For any two finite trees $\mathbf{T}, \mathbf{T'}$, $C_f(\mathbf{T}) = C_f(\mathbf{T'}) \implies \mathbf{T} = \mathbf{T'}$.*

Proof. We just show that we can reconstruct $\mathbf{T}$ from $C_f(\mathbf{T}) = collect(\mathbf{T})$. We can ignore the call to $sort(\mathbf{T})$ in $collect(\mathbf{T})$: we can consider that the tree $\mathbf{T}$ we will recover is already sorted. To shorten notations, let's write $c := C_f(\mathbf{T})$.

First, we can partition c into levels. Indeed, remark that if we know that the indices corresponding to states at depth d form the set $[\![k, \ell]\!]$, then we know that the number of states at depth $d+1$ is $\sum_{j=k}^{\ell} c_j$, and so the states at depth $d+1$ correspond to indices $[\![\ell+1, \ell + \sum_{j=k}^{\ell} c_j]\!]$. So, we can now iterate on the levels of c: let's write for convenience $k_d, \ell_d \in \mathbb{N}$ the first and last indices of states at depth d.

The first level, corresponding to depth 0, is easy to reconstruct: simply create the root. For our induction, we also create the predecessors of the root, of which we know the number, so the induction begins at depth 1.

Now, suppose we have uniquely reconstructed $\mathbf{T}$ up to depth d, and that we want to reconstruct level $d+1$. We traverse our reconstructed depth 1 from left to right, and simultaneously traverse $c_{k_d}, \dots, c_{\ell_d}$. The j-th state we encounter has degree c_{k_d+j}, so we create c_{k_d+j} children for that state. Thus, the level at depth $d+1$ is reconstructed uniquely too.

Thus, we reconstruct $\mathbf{T}$, and this concludes the proof. □

The results which make this code useful are the following lemma and its corollary.

Lemma 19. *For all finite trees $\mathbf{T}_1, \mathbf{T}_2, \mathbf{T}_3 \in \mathbb{F}_T$, we have $[\mathbf{T}_1]_{\text{depth}(\mathbf{T}_3)} <_f [\mathbf{T}_2]_{\text{depth}(\mathbf{T}_3)} \implies \mathbf{T}_1\mathbf{T}_3 <_f \mathbf{T}_2\mathbf{T}_3$.*

Proof. Let's prove this by induction on $\mathbf{T}_1$ and $\mathbf{T}_2$'s depth. It's trivial at depth 0. Take $\mathbf{T}_1, \mathbf{T}_2$ of depth $\leq k+1$, with k such that the result stands for trees of depth $\leq k$.

Because of Lemma 17, since $\mathbf{T}_1 <_f \mathbf{T}_2$, $C_f(\mathbf{T}_1)$ cannot be a prefix of $C_f(\mathbf{T}_2)$.

Thus, there exists an index i such that $C_f(\mathbf{T}_1)_i < C_f(\mathbf{T}_2)_i$ and for all $j < i$, we have $C_f(\mathbf{T}_1)_j = C_f(\mathbf{T}_2)_j$. Let $x \in S_{\mathbf{T}_1}$ be the vertex at index i in $C_f(\mathbf{T}_1)$, and let $y \in S_{\mathbf{T}_2}$ be the vertex at index i in $C_f(\mathbf{T}_2)$. In the following, for a tree $\mathbf{T}$ and a vertex $u \in S_{\mathbf{T}}$, we denote by $C_{f\,\mathbf{T}}(u)$ the code of the subtree with root u in $\mathbf{T}$. Since the codes share the same prefix of length $i-1$, $\mathrm{depth}_{\mathbf{T}_1}(x) = \mathrm{depth}_{\mathbf{T}_2}(y)$ (as seen in the proof of Lemma 18, this shared prefix of length $i-1$ holds all the information necessary to reconstruct everything above x and y). Let's denote by d this depth. Because $[\mathbf{T}_1]_{\mathrm{depth}(\mathbf{T}_3)} <_f [\mathbf{T}_2]_{\mathrm{depth}(\mathbf{T}_3)}$, we have $d < \mathrm{depth}(\mathbf{T}_3)$.

It is clear that we have $[\mathbf{T}_1]_{d-1} = [\mathbf{T}_2]_{d-1}$, so in particular, $[\mathbf{T}_1\mathbf{T}_3]_{d-1} = [\mathbf{T}_2\mathbf{T}_3]_{d-1}$. Let z be the root of the tree with minimal code in $\mathbf{T}_3$ at depth d. Now, we show that the first difference between the codes of $\mathbf{T}_1\mathbf{T}_3$ and $\mathbf{T}_2\mathbf{T}_3$ is at the index j corresponding to the vertex (x,z) in $C_f(\mathbf{T}_1\mathbf{T}_3)$, and to the vertex (y,z) in $C_f(\mathbf{T}_2\mathbf{T}_3)$. There might be multiple possibilities for z; we can assume that we take the one which gives the minimum j.

Indeed, assume that a vertex of the form (x,t) for some vertex t of $\mathbf{T}_3$ appears in $C_f(\mathbf{T}_1\mathbf{T}_3)$ at depth d before index j. Since it appears before vertex (x,z), by induction hypothesis, it means that the code of the subtree anchored on t must be smaller than that of the subtree anchored on z. By minimality of z, this means that $C_{f\,\mathbf{T}_3}(z) = C_{f\,\mathbf{T}_3}(t)$. Since we have chosen z to be the first occurrence of this code at this depth, we must have $t = z$. So, (x,z) is the first vertex in $C_f(\mathbf{T}_1\mathbf{T}_3)$ in which x appears. A similar reasoning shows that no vertex involving y appears before index j in $C_f(\mathbf{T}_2\mathbf{T}_3)$. Since every element before x is shared between $C_f(\mathbf{T}_1)$ and $C_f(\mathbf{T}_2)$, this means that the first difference between $C_f(\mathbf{T}_1\mathbf{T}_3)$ and $C_f(\mathbf{T}_2\mathbf{T}_3)$ is at or after index j.

At index j, the number of predecessors $C_f(\mathbf{T}_1\mathbf{T}_3)_j$ is $\mathrm{npreds}_{\mathbf{T}_1}(x)\,\mathrm{npreds}_{\mathbf{T}_3}(z)$ while $C_f(\mathbf{T}_1\mathbf{T}_3)_j$ is $\mathrm{npreds}_{\mathbf{T}_2}(y)\,\mathrm{npreds}_{\mathbf{T}_3}(z)$. Since $\mathrm{npreds}_{\mathbf{T}_1}(x) < \mathrm{npreds}_{\mathbf{T}_2}(y)$, this shows that $\mathbf{T}_1\mathbf{T}_3 <_f \mathbf{T}_2\mathbf{T}_3$. □

Corollary 20. *For all finite trees $\boldsymbol{T}_1, \boldsymbol{T}_2, \boldsymbol{T}_3, \boldsymbol{T}_4 \in \mathbb{F}_T$, if $[\boldsymbol{T}_1]_{\mathrm{depth}(\boldsymbol{T}_3)} <_f [\boldsymbol{T}_2]_{\mathrm{depth}(\boldsymbol{T}_3)}$ and $[\boldsymbol{T}_3]_{\mathrm{depth}(\boldsymbol{T}_2)} \leq_f [\boldsymbol{T}_4]_{\mathrm{depth}(\boldsymbol{T}_2)}$, we have $\boldsymbol{T}_1\boldsymbol{T}_3 <_f \boldsymbol{T}_2\boldsymbol{T}_4$.*

Proof. By Lemma 19, we have $\mathbf{T}_1\mathbf{T}_3 <_f \mathbf{T}_2\mathbf{T}_3$. If $\mathbf{T}_3 = \mathbf{T}_4$, we can conclude now. Otherwise, $\mathbf{T}_3 <_f \mathbf{T}_4$, and we have, by Lemma 19, $\mathbf{T}_2\mathbf{T}_3 <_f \mathbf{T}_2\mathbf{T}_4$. Combining the two inequalities, we get: $\mathbf{T}_1\mathbf{T}_3 <_f \mathbf{T}_2\mathbf{T}_4$. □

We are now ready for the recovery algorithm on finite trees.

Lemma 21. *If $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C} \in \mathbb{F}_T$, $\boldsymbol{A}$ is finite, and $\boldsymbol{AB} = \boldsymbol{AC}$, then $[\boldsymbol{B}]_{\mathrm{depth}(\boldsymbol{A})} = [\boldsymbol{C}]_{\mathrm{depth}(\boldsymbol{A})}$.*

Proof. Since $<_f$ is a complete order, if $[\mathbf{B}]_{\mathrm{depth}(\mathbf{A})} \neq [\mathbf{C}]_{\mathrm{depth}(\mathbf{A})}$, we can assume without loss of generality that we are in the case $[\mathbf{B}]_{\mathrm{depth}(\mathbf{A})} < [\mathbf{C}]_{\mathrm{depth}(\mathbf{A})}$. In that case, by Lemma 19, we have $\mathbf{AB} <_f \mathbf{AC}$. This concludes the proof. □

Lemma 22. *If $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C} \in \mathbb{F}_T$ and $\boldsymbol{A}$ is infinite, and $\boldsymbol{AB} = \boldsymbol{AC}$, then $\boldsymbol{B} = \boldsymbol{C}$.*

Proof. For every $d \in \mathbb{N}$, we have $[\mathbf{A}]_d[\mathbf{B}]_d = [\mathbf{A}]_d[\mathbf{C}]_d$, and thus, from Lemma 21, $[\mathbf{B}]_d = [\mathbf{C}]_d$. This implies that $\mathbf{B} = \mathbf{C}$. □

Corollary 23. *If $A, B, C \in \mathbb{D}_D$, and $AB = AC$, then $B = C$.*

Proof. If $AB = AC$, then $\widetilde{A}\widetilde{B} = \widetilde{A}\widetilde{C}$. Using Lemma 22, this means that $\widetilde{B} = \widetilde{C}$. Thus, $B = C$. □

We can now extend the order on possibly infinite trees; this will be of use for our results on the unicity of k-th roots. For a tree $\mathbf{T}$, define its code as $C(\mathbf{T}) := (C_f([\mathbf{T}]_i))_{i\in\mathbb{N}}$, and say that $\mathbf{T} \leq \mathbf{U}$ if and only if $C(\mathbf{T}) \leq_{\mathrm{lex}} C(\mathbf{U})$.

Lemma 24. *For all trees $\boldsymbol{T}_1, \boldsymbol{T}_2, \boldsymbol{T}_3 \in \mathbb{F}_T$, if $\boldsymbol{T}_1 < \boldsymbol{T}_2$, we have $\boldsymbol{T}_1\boldsymbol{T}_3 < \boldsymbol{T}_2\boldsymbol{T}_3$.*

Proof. If $\mathbf{T}_1 < \mathbf{T}_2$, then $\mathbf{T}_1 \neq \mathbf{T}_2$. In particular, there is a minimal depth d such that $[\mathbf{T}_1]_d \neq [\mathbf{T}_2]_d$. Since for every $i < d$, we have $[\mathbf{T}_1]_i = [\mathbf{T}_2]_i$, we have $C(\mathbf{T}_1\mathbf{T}_3)_1, \ldots, C(\mathbf{T}_1\mathbf{T}_3)_{d-1} = C(\mathbf{T}_2\mathbf{T}_3)_1, \ldots, C(\mathbf{T}_2\mathbf{T}_3)_{d-1}$.

What is left to prove is that $C(\mathbf{T}_1\mathbf{T}_2)_d < C(\mathbf{T}_1\mathbf{T}_3)_d$, that is $C_f([\mathbf{T}_1\mathbf{T}_3]_d) < C_f([\mathbf{T}_1\mathbf{T}_3]_d)$. This follows from the fact that $C_f([\mathbf{T}_1]_d) < C_f([\mathbf{T}_2]_d)$ and Lemma 19. □

Corollary 25. *For all trees $\boldsymbol{T}_1, \boldsymbol{T}_2, \boldsymbol{T}_3, \boldsymbol{T}_4 \in \mathbb{F}_T$, if $\boldsymbol{T}_1 < \boldsymbol{T}_2$ and $\boldsymbol{T}_3 \leq \boldsymbol{T}_4$, we have $\boldsymbol{T}_1\boldsymbol{T}_3 < \boldsymbol{T}_2\boldsymbol{T}_4$.*

Proof. By Lemma 24, we have $\mathbf{T}_1\mathbf{T}_3 < \mathbf{T}_2\mathbf{T}_3$. If $\mathbf{T}_3 = \mathbf{T}_4$, we can conclude now. Otherwise, $\mathbf{T}_3 < \mathbf{T}_4$, and we have, by Lemma 24, $\mathbf{T}_1\mathbf{T}_3 < \mathbf{T}_2\mathbf{T}_3$ and $\mathbf{T}_2\mathbf{T}_3 < \mathbf{T}_2\mathbf{T}_4$. Combining the two, we get: $\mathbf{T}_1\mathbf{T}_3 < \mathbf{T}_2\mathbf{T}_4$. □

3.2. Transforming an FDS into a forest

In this subsection, we introduce a way of converting a general FDS into a forest, since the product on forests works level by level. We do as follows:

Definition 26. Let $A = C_n(\mathbf{T}_1, \dots, \mathbf{T}_n)$ be a connected FDS. For any $a \in S_A$, we write $A^{-k}(a) := \{s \in S_A : A^k(s) = a\}$. Then, for each $a \in [A]_0$, we set

$$S_a := \{(s,k) : s \in A^{-k}(a), k \in \mathbb{N}\}$$

and

$$E_a := \{((s,k),(A(s),k-1)) : (s,k) \in S_a\}.$$

Lemma 27. *The directed graph $\mathbf{T}_a(A)$ with vertex set S_a and edge set E_a defined above is a tree. Moreover S_a and S_b are disjoint for all $a \neq b$.*

Proof. Take $a \in [A]_0$. We will show that $\mathbf{T}_a(A)$ is a tree of root $(a,0)$. First, $\mathbf{T}_a(A)$ is acyclic because k necessarily decreases following any arc, which also shows that $\mathbf{T}_a(A)$ is correctly oriented. Furthermore, if $b \in S_a$, then there exists k such that $A^k(b) = a$, and thus we have the following path from a to b:

$$(b,k) \to (A(b),k-1) \to \cdots \to (A^{k-1}(b),1) \to (a,0),$$

which has all of its edges in E_a. So, $\mathbf{T}_a(A)$ is a well-defined tree.

Now, we show that if $a, b \in S_{[A]_0}$ and $a \neq b$, then $S_a \cap S_b = \emptyset$. Suppose that $(s,k) \in S_a \cap S_b$. Then, $A^k(s) = a = b$, which is the desired contradiction. □

We thus define the *unrolling* of A as $\widetilde{A} := \sum_{a \in S_{[A]_0}} \mathbf{T}_a(A)$, with $S_{\widetilde{A}} = \bigcup_{a \in S_{[A]_0}} S_a$.

We can then extend this to general FDSs, by writing: $\widetilde{A+B} = \widetilde{A} + \widetilde{B}$. Note that the unrolling is not injective. Indeed, for instance, $\widetilde{C_3} = \widetilde{3C_1}$. This is not true even for FDSs with the same periodic part: if $\mathbf{T}$ and $\mathbf{U}$ are two distinct trees and $X = 2C_1(\mathbf{T}) + C_2(\mathbf{U},\mathbf{U})$ and $Y = 2C_1(\mathbf{U}) + C_2(\mathbf{T},\mathbf{T})$, then $\widetilde{X} = \widetilde{Y}$. However, in the connected case, we have injectivity.

Lemma 28. *Let $X, Y \in \mathbb{D}$. If X and Y are connected and $[X]_0 = [Y]_0$, then $\widetilde{X} = \widetilde{Y} \implies X = Y$.*

Proof. Let $X = C_x(\mathbf{T}_1, \dots, \mathbf{T}_x)$. Then $\widetilde{X}$ has x infinite trees $\mathbf{X}_1, \dots, \mathbf{X}_x$, each a periodic shift of the previous one:

$$tseq(\mathbf{X}_1) = (\mathbf{T}_1, \mathbf{T}_2, \dots, \mathbf{T}_x), \dots, tseq(\mathbf{X}_x) = (\mathbf{T}_x, \mathbf{T}_1, \dots, \mathbf{T}_{x-1}).$$

We have $Y = C_x(\mathbf{U}_1, \dots, \mathbf{U}_x)$, and similarly $\widetilde{Y}$ consists of the trees $\mathbf{Y}_1, \dots, \mathbf{Y}_x$ where

$$tseq(\mathbf{Y}_1) = (\mathbf{U}_1, \mathbf{U}_2, \dots, \mathbf{U}_x), \dots, tseq(\mathbf{U}_x) = (\mathbf{U}_x, \mathbf{U}_1, \dots, \mathbf{U}_{x-1}).$$

Then $\mathbf{U}_1 \in \{\mathbf{T}_1, \dots, \mathbf{T}_x\}$, without loss say $\mathbf{U}_1 = \mathbf{T}_1$, then $\mathbf{U}_y = \mathbf{T}_y$ for all $1 \leq y \leq x$ and $X = Y$. □

Example 29. See Figs. 4 and 5 for examples of unrollings.

The following lemma explains why the unrolling operation makes sense: it is compatible with the product. The proof is rather technical, but the intuition for this result is simple. A cycle behaves very much like an infinite path in terms of predecessors, and the unrolling converts the cycle into an infinite path that behaves similarly. Moreover, the reason we create multiple infinite trees for each cycle is to avoid problems with cases where the product of two connected FDSs gives a non-connected FDS.

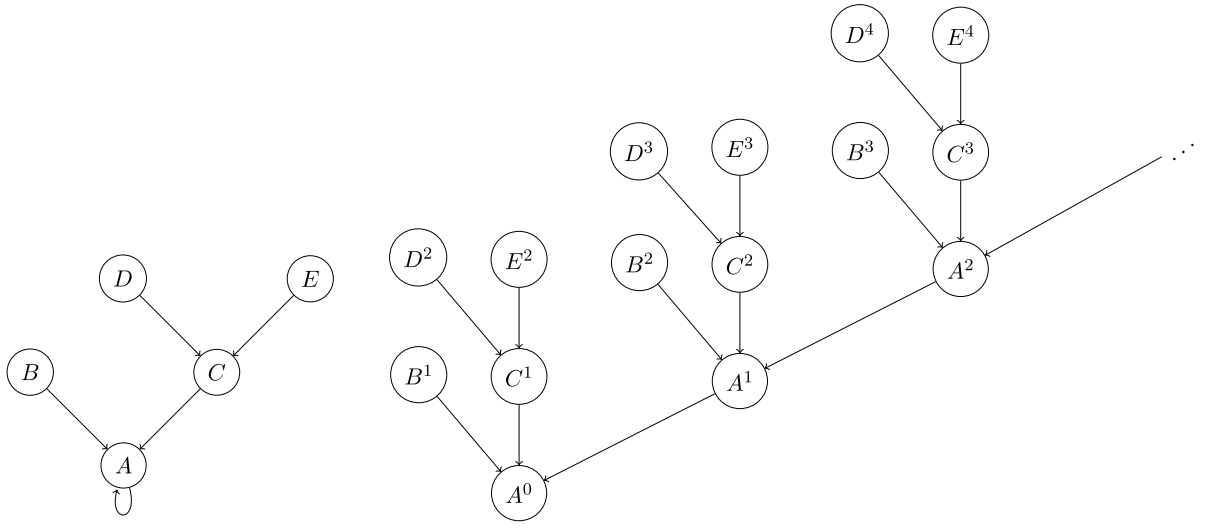
Lemma 30. *For any $A, B \in \mathbb{D}$, we have: $\widetilde{A}\widetilde{B} = \widetilde{AB}$.*

Proof. We show this result for connected A and B as the other cases follow by distributivity. Thus, we write $A = C_m(\mathbf{T}_0, \dots, \mathbf{T}_{m-1})$ and $B = C_n(\mathbf{U}_0, \dots, \mathbf{U}_{n-1})$. Now, we can write:

$$S_{\widetilde{A}} = \bigcup_{a \in [A]_0} \{(s,k) : s \in A^{-k}(a), k \in \mathbb{N}\},$$
$$S_{\widetilde{B}} = \bigcup_{b \in [B]_0} \{(s,k) : s \in B^{-k}(b), k \in \mathbb{N}\}.$$

Now, the product $\widetilde{A}\widetilde{B}$ has the following state set:

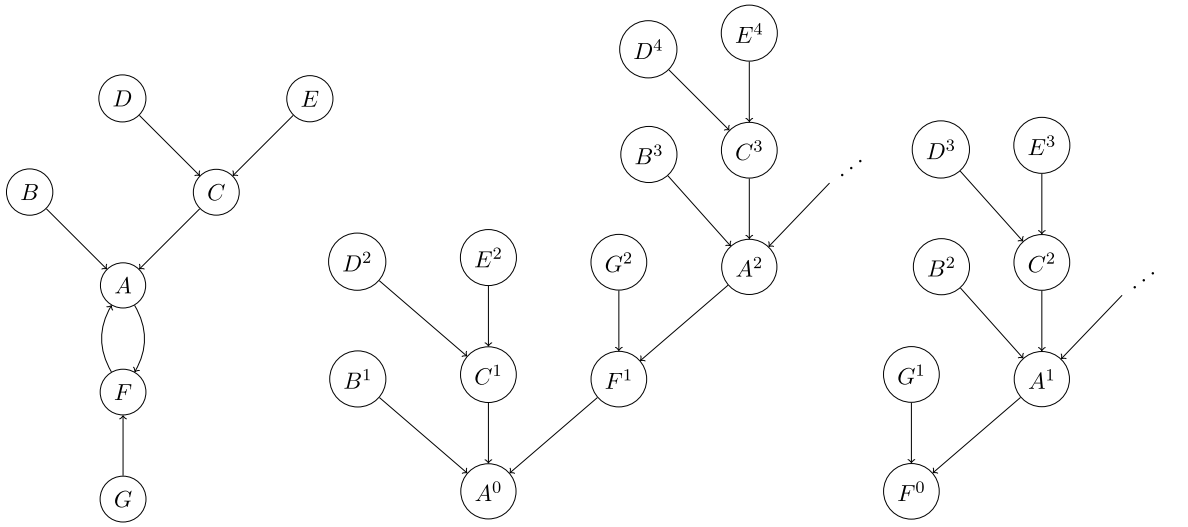
$$\begin{aligned} S_{\widetilde{A}\widetilde{B}} &= \{(a,b) \in S_{\widetilde{A}} \times S_{\widetilde{B}} : \mathrm{depth}_{\widetilde{A}}(a) = \mathrm{depth}_{\widetilde{B}}(b)\} \\ &\simeq \{((s_a,k_a),(s_b,k_b)) \in S_{\widetilde{A}} \times S_{\widetilde{B}} : k_a = k_b\} \\ &\simeq \bigcup_{(a,b) \in [A]_0 \times [B]_0} \{(s_a,s_b,k) \in S_A \times S_B \times \mathbb{N} : s_a \in A^{-k}(a), s_b \in B^{-k}(b); k \in \mathbb{N}\}. \end{aligned}$$



(a) A dendron T.

(b) The tree $\widetilde{T}$, rooted in A^0.

Fig. 4. The $\widetilde{\cdot}$ operation on a dendron.



(a) A connected FDS S.

(b) The forest $\widetilde{S}$, with roots A^0 and F^0.

Fig. 5. The $\widetilde{\cdot}$ operation on a connected FDS.

Now, let's show that this is isomorphic to $S_{\widetilde{AB}}$ (remember that $S_{AB} = S_A \times S_B$):

$$\begin{aligned} S_{\widetilde{AB}} &= \bigcup_{c\in[AB]_0} \{(s,k) \in S_{AB} \times \mathbb{N} : s \in AB^{-k}(c), k \in \mathbb{N}\} \\ &= \bigcup_{(a,b)\in[A]_0\times[B]_0} \{((s_a,s_b),k) \in S_{AB} \times \mathbb{N} : (s_a,s_b) \in AB^{-k}((a,b)), k \in \mathbb{N}\} \\ &= \bigcup_{(a,b)\in[A]_0\times[B]_0} \{((s_a,s_b),k) \in S_{AB} \times \mathbb{N} : s_a \in A^{-k}(a), s_b \in B^{-k}(b), k \in \mathbb{N}\}. \end{aligned}$$

The last step comes from the following identity: for $c = (a,b) \in S_{AB} = S_A \times S_B$, we have $AB^{-k}(c) = A^{-k}(a) \times B^{-k}(b)$. Thus, we have shown that $S_{\widetilde{AB}} \simeq S_{\widetilde{AB}}$.

Now, what is left to do is show that the edges are also isomorphic. Thus, we must show that for any $(s_a,s_b,k),(s'_a,s'_b,k') \in S_{\widetilde{AB}}$, we have $(s_a,s_b,k) \to (s'_a,s'_b,k')$ in $\widetilde{AB}$ if and only if we have $(s_a,s_b,k) \to (s'_a,s'_b,k')$ in $\widetilde{AB}$.

In the end of the proof, we denote by $x \to Cy$ the existence of an edge from x to y in the forest or FDS C (if C is an FDS, $x \to Cy$ means $y = C(x)$). Now, we can reason by equivalence:

$$\begin{aligned}
&(s_a, s_b, k) \xrightarrow{\widetilde{AB}} (s'_a, s'_b, k') \\
\iff\ & k' = k-1 \wedge (s_a, k) \xrightarrow{\widetilde{A}} (s'_a, k') \wedge (s_b, k) \xrightarrow{\widetilde{B}} (s'_b, k') \\
\iff\ & k' = k-1 \wedge s_a \xrightarrow{A} s'_a \wedge s_b \xrightarrow{B} s'_b \\
\iff\ & k' = k-1 \wedge (s_a, s_b) \xrightarrow{AB} (s'_a, s'_b) \\
\iff\ & (s_a, s_b, k) \xrightarrow{\widetilde{AB}} (s'_a, s'_b, k').
\end{aligned}$$

This concludes the proof. □

We can now show that division is unambiguous when restricted to connected FDSs.

Theorem 31. *For any FDS $A \in \mathbb{D}$, if $X, Y \in \mathbb{D}$ are connected, then*

$$AX = AY \implies X = Y.$$

Proof. Suppose $AX = AY$. Let $[X]_0 = C_x$ and $[Y]_0 = C_y$, then $|[AX]_0| = x|[A]_0|$ and $|[AY]_0| = y|[A]_0|$ show that $x = y$, that is $[X]_0 = [Y]_0$. Thus,

$$AX = AY \implies \widetilde{AX} = \widetilde{AY} \xRightarrow{\text{Lemma 30}} \widetilde{A}\widetilde{X} = \widetilde{A}\widetilde{Y} \xRightarrow{\text{Lemma 22}} \widetilde{X} = \widetilde{Y} \xRightarrow{\text{Lemma 28}} X = Y. \quad \square$$

We remark that Theorem 31 implies [7, Conjecture 3.1]. Indeed, if A and B are connected and $AX = AY = B$, then X and Y are connected, thus $X = Y$.

3.3. Cancellative FDSs are those with a fixed point

Using the results of the previous part, we have the following lemma:

Lemma 32. *If $A \in \mathbb{D}$, and A has a fixed point, then A is cancellative.*

Proof. Take $B, D \in \mathbb{D}$ such that $AB = D$. Let's show that we can recover B by induction on the size of D. The base case is trivial: if $|D| = 0$, then $D = 0$ and since A has a fixed point, $A \neq 0$, so $B = 0$.

Let ℓ be the size of the smallest cycle of D. Since A has a cycle of length 1, it means that the smallest cycle of B is of length ℓ too. Let $L \subseteq \mathbb{N}$ be the set of divisors of ℓ. We let $A' = \mathrm{supp}_L(A)$, and similarly $B' = \mathrm{supp}_L(B)$ and $D' = \mathrm{supp}_L(D)$. Then we have $A'B' = D'$. Indeed, cycles of length ℓ in D come from a product of a cycle of length a in A and length b in B, such that $a \vee b = \ell$. In particular, this implies that $a|\ell$, and since $b \geq \ell$ because ℓ is the smallest cycle length in B, this implies $b = \ell$.

So, we have $A'B' = D'$, which implies $\widetilde{A'}\widetilde{B'} = \widetilde{D'}$. Take the smallest tree in $\widetilde{A'}$, denote it by $\mathbf{T}_A$, and take the smallest tree in $\widetilde{D'}$, denote it by $\mathbf{T}_D$. Then, there is a tree $\mathbf{T}_B$ in $\widetilde{B'}$ such that $\mathbf{T}_A\mathbf{T}_B = \mathbf{T}_D$, by Corollary 20 and minimality of $\mathbf{T}_A$ and $\mathbf{T}_D$.

This means that by Lemma 22, we find $\mathbf{T}_B$ by dividing $\mathbf{T}_D$ by $\mathbf{T}_A$. Moreover, since $\mathbf{T}_B$ is in $\widetilde{B'}$, we know that it comes from a cycle of length ℓ in B. So, we set $E = C_\ell(tseq(\mathbf{T}_B)_1, \dots, tseq(\mathbf{T}_B)_\ell)$ the "reconstruction" of this cycle. The useful property of E is that it is part of B. Thus, the equation becomes $A(B - E) = D - AE$ (those two subtractions are well-defined since E is a connected component of B, and AE is a connected component of D), which involves a product strictly smaller than D. □

Now, we show that if an FDS has no fixed point, then it is not cancellative.

Lemma 33. *Let $\mathcal{A}$ be a finite set of integers greater than 1. Then there exist $X \neq X' \in \mathbb{D}_P$ such that $C_a X = C_a X'$ for all $a \in \mathcal{A}$.*

Proof. Recall the sequence δ_J from Lemma 12. For all $I \subseteq \mathcal{A}$, let $\alpha_I = \delta_{\mathcal{A}} \prod_{a \in \mathcal{A}} a$ and $\alpha'_I = \alpha_I + (-1)^{|I|}\delta_I \prod_{a \in A \setminus I} a$.

Since $\alpha_I, \alpha'_I \geq 0$, we can then define the FDSs $X = \sum_{I \subseteq \mathcal{A}} \alpha_I C_{\bigvee I}$ and $X' = \sum_{I \subseteq \mathcal{A}} \alpha'_I C_{\bigvee I}$. We remark that the number of fixed points in X and X' are $\alpha_\varnothing$ and $\alpha'_\varnothing$, respectively. Since $\alpha'_\varnothing = \alpha_\varnothing + \prod_{a \in \mathcal{A}} a \neq \alpha_\varnothing$, X and X' are distinct FDSs.

Let $b \in \mathcal{A}$. For all $I \subseteq \mathcal{A} \setminus \{b\}$, let $J = I \cup \{b\}$. Then we have

$$\begin{aligned}
C_b(\alpha'_I C_{\bigvee I} + \alpha'_J C_{\bigvee J}) &= (\alpha'_I(b \wedge \bigvee I) + \alpha'_J b) C_{\bigvee J} \\
&= ((\alpha_I + (-1)^{|I|}\delta_I \prod_{a \in \mathcal{A} \setminus I} a)(b \wedge \bigvee I) + (\alpha_J - (-1)^{|I|}\delta_J \prod_{a \in A \setminus J} a) b) C_{\bigvee J}
\end{aligned}$$

$\mathcal{M}_{\mathbf{C}} \leftarrow \mathcal{D}(\mathbf{C})$;
$\mathcal{M}' \leftarrow \varnothing$;
while $\mathcal{M}_C \neq \varnothing$ **do**
  $d \leftarrow \mathrm{depth}(\mathbf{C}) - 1$;
  $\mathcal{T}_{\mathbf{C}} \leftarrow \{\{\mathbf{X} \in \mathcal{M}_{\mathbf{C}} : \mathrm{depth}(\mathbf{X}) \geq d\}\}$;
  $\mathcal{T}_{\mathbf{A}} \leftarrow \{\{\mathbf{Y} \in \mathcal{D}(\mathbf{A}) : \mathrm{depth}(\mathbf{Y}) \geq d\}\}$;
  $\mathbf{t}_{\mathbf{C}} \leftarrow \arg\min_{\mathbf{X} \in \mathcal{T}_{\mathbf{C}}} \mathcal{C}_f(\mathbf{X})$;
  $\mathbf{t}_{\mathbf{A}} \leftarrow \arg\min_{\mathbf{Y} \in \mathcal{T}_{\mathbf{A}}} \mathcal{C}_f(\mathbf{Y})$;
  $\mathbf{t}_{\mathbf{B}} \leftarrow divide(\mathbf{t}_{\mathbf{C}}, \mathbf{t}_{\mathbf{A}})$;
  if $\boldsymbol{t}_B = \perp$ *or* $\boldsymbol{t}_B \mathcal{D}(\boldsymbol{A}) \not\subseteq \mathcal{M}_C$ **then**
    return $\perp$;
  end
  $\mathcal{M}_{\mathbf{C}} \leftarrow \mathcal{M}_{\mathbf{C}} \setminus \mathbf{t}_{\mathbf{B}} \mathcal{D}(\mathbf{A})$;
  $\mathcal{M}' \leftarrow \mathcal{M}' \cup \{\mathbf{t}_{\mathbf{B}}\}$;
end
Let $\mathbf{B}$ such that $\mathcal{D}(\mathbf{B}) = \mathcal{M}'$;
return $\mathbf{B}$;

Fig. 6. $divide(\mathbf{C}, \mathbf{A})$ to divide $\mathbf{C}$ by $\mathbf{A}$, for finite $\mathbf{C}$ and $\mathbf{A}$.

$$
\begin{aligned}
&= (\alpha_I (b \wedge \bigvee I) + \alpha_J b) C_{\bigvee J} \\
&\quad + \left[\left((-1)^{|I|} \delta_I \prod_{a \in A \setminus I} a \right) (b \wedge \bigvee I) - \left((-1)^{|I|} \delta_I (b \wedge \bigvee I) \prod_{a \in A \setminus I} a \right) \right] C_{\bigvee J} \\
&= C_b (\alpha_I C_{\bigvee I} + \alpha_J C_{\bigvee J}).
\end{aligned}
$$

Therefore,

$$C_b X' = \sum_{I \subseteq \mathcal{A} \setminus \{b\}} C_b (\alpha'_I C_{\bigvee I} + \alpha'_J C_{\bigvee J}) = \sum_{I \subseteq \mathcal{A} \setminus \{b\}} C_b (\alpha_I C_{\bigvee I} + \alpha_J C_{\bigvee J}) = C_b X. \quad \square$$

This lemma above combined with Lemma 32 gives:

Theorem 34. *An FDS is cancellative if and only if it has a fixed point.*

Proof. The case where the FDS has a fixed point is handled by Lemma 32. Suppose A has no fixed point and let $\mathcal{A}$ be the set of all cycle lengths of A. Following Lemma 33, there exist $X, X' \in \mathbb{D}_P$ such that $C_a X = C_a X'$ for all $a \in \mathcal{A}$. Let $B = C_a(\mathbf{T}_1, \ldots, \mathbf{T}_2)$ be a connected component of A, where $a \in \mathcal{A}$. According to Corollary 14, we have $BX = BX'$. Summing over all connected components of A, we finally obtain $AX = AX'$. $\square$

From now on, we define $\mathbb{D}^*$ to be the set of cancellable FDSs. Its algebraic structure is that of a cancellative subsemiring of $\mathbb{D}$, but $\mathbb{D}^*$ does not have an additive identity.

4. Polynomial-time algorithm for tree and dendron division

The algorithm Fig. 6 provides an algorithmic proof of Lemma 21, as formalised below:

Lemma 35. *The $divide$ algorithm is correct: for all $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C} \in \mathbb{F}_T$, $\boldsymbol{AB} = \boldsymbol{C} \implies [\boldsymbol{B}]_{\mathrm{depth}(\boldsymbol{A})} = divide(\boldsymbol{C}, \boldsymbol{A})$, and $[\boldsymbol{C}]_{\mathrm{depth}(\boldsymbol{A})} \not| \boldsymbol{A} \implies divide(\boldsymbol{C}, \boldsymbol{A}) = \perp$.*

Proof. In the case in which $\mathbf{AB} = \mathbf{C}$, we show that we can recover uniquely $[\mathbf{B}]_{\mathrm{depth}(\mathbf{A})}$ from $\mathbf{A}$ and $\mathbf{AB}$ by induction on $\mathrm{depth}(\mathbf{A})$. The base case is for $\mathrm{depth}(\mathbf{A}) = -1$, in which $\mathbf{A} = \mathbf{0}$ is the empty tree. Then, the result is trivial since $[\mathbf{B}]_{-1} = \mathbf{0}$ for any $\mathbf{B} \in \mathbb{F}_T$.

Now, for the general case, we do an induction on the size of the product $\mathbf{C} = \mathbf{AB}$. The base case for $\mathbf{C} = \mathbf{0}$ is trivial. Let's write $\{\{\mathbf{T}_1, \ldots, \mathbf{T}_n\}\} = D(\mathbf{A})$ with $\mathbf{T}_1 \leq_f \cdots \leq_f \mathbf{T}_n$, $\{\{\mathbf{U}_1, \ldots, \mathbf{U}_k\}\} = D(\mathbf{B})$ with $\mathbf{U}_1 \leq_f \cdots \leq_f \mathbf{U}_k$, and finally, write $\{\{\mathbf{V}_1, \ldots, \mathbf{V}_{nk}\}\} = D(\mathbf{C})$ with $\mathbf{V}_1 \leq_f \cdots \leq_f \mathbf{V}_{nk}$. We remark that to recover $[\mathbf{B}]_{\mathrm{depth}(\mathbf{A})}$, all we need is to recover $[\mathbf{U}_j]_{\mathrm{depth}(\mathbf{A})-1}$ for all $1 \leq j \leq k$.

Let d be $\mathrm{depth}(\mathbf{C}) - 1$ as in the algorithm. Then let $\mathbf{t}_{\mathbf{A}}$ (respectively $\mathbf{t}_{\mathbf{B}}$, $\mathbf{t}_{\mathbf{C}}$) be the minimum tree in $D(\mathbf{A})$ (respectively $D(\mathbf{B})$, $D(\mathbf{C})$) of depth $\geq d$. We can then write $\mathbf{t}_{\mathbf{A}} \mathbf{t}_{\mathbf{B}} = \mathbf{t}_{\mathbf{C}}$ without loss of generality. Since $\mathbf{t}_{\mathbf{A}}$ has depth $< \mathrm{depth}(\mathbf{A})$, the outer induction hypothesis shows that $divide(\mathbf{t}_{\mathbf{C}}, \mathbf{t}_{\mathbf{A}}) = [\mathbf{t}_{\mathbf{B}}]_d$.

There are two cases. If $\mathrm{depth}(\mathbf{B}) \leq \mathrm{depth}(\mathbf{A})$, then $d = \mathrm{depth}(\mathbf{B})$ by Lemma 10 and so $[\mathbf{t}_{\mathbf{B}}]_d = \mathbf{t}_{\mathbf{B}}$. Otherwise, if $\mathrm{depth}(\mathbf{B}) > \mathrm{depth}(\mathbf{A})$, then $\mathrm{depth}(\mathbf{C}) = \mathrm{depth}(\mathbf{A})$ by Lemma 10 and so $\mathbf{t}_{\mathbf{B}} = [\mathbf{t}_{\mathbf{B}}]_{\mathrm{depth}(\mathbf{A})-1}$, which is a depth 1 subtree of $[\mathbf{B}]_{\mathrm{depth}(\mathbf{A})}$. So, in both cases, $\mathbf{t}_{\mathbf{B}}$ is a depth 1 subtree of $[\mathbf{B}]_{\mathrm{depth}(\mathbf{A})}$.

Now that we have $\mathbf{t_B}$, the algorithm computes $\mathbf{t_B}\mathcal{D}(\mathbf{A}) = \{\{\mathbf{t_B}\mathbf{T}_1, \dots, \mathbf{t_B}\mathbf{T}_n\}\}$, which are n subtrees of $\mathbf{C}$, and removes them from $\mathbf{C}$. Finally, the next loop iteration corresponds to applying the internal induction hypothesis to the identity $\mathbf{AB'} = \mathbf{D'}$ where

$$\mathcal{D}(\mathbf{B'}) = \mathcal{D}([\mathbf{B}]_{\mathrm{depth}(\mathbf{A})}) \setminus \{\mathbf{t_B}\}$$

and

$$\mathcal{D}(\mathbf{D'}) = \mathcal{D}(\mathbf{D}) \setminus \mathbf{t_B}\mathcal{D}(\mathbf{A}).$$

To conclude, if we are in the case where $[\mathbf{C}]_{\mathrm{depth}(\mathbf{A})} \not| \mathbf{A}$, we need to show that if $divide([\mathbf{C}]_{\mathrm{depth}(\mathbf{A})}, \mathbf{A})$ does not return $\perp$ but some tree $\mathbf{B}$, then $\mathbf{AB} = [\mathbf{C}]_{\mathrm{depth}(\mathbf{A})}$ which is a contradiction. To do so, remark that by construction during the while loop, $\mathcal{D}(\mathbf{A})\mathcal{D}(\mathbf{B}) = \mathcal{D}([\mathbf{C}]_{\mathrm{depth}(\mathbf{A})})$, which means that $\mathbf{AB} = [\mathbf{C}]_{\mathrm{depth}(\mathbf{A})}$. □

This algorithm only works on trees. But Lemma 38 allows one to use it on dendrons, using the truncature of their unrollings. First, we need the following definition, adapting the definition of product isomorphism for forests:

Definition 36. Given a product $\mathbf{B} = \prod_{i\in I} \mathbf{A}_i$ for some finite set I, a family $(\mathbf{A}_i)_{i\in I} \in \mathbb{F}^I$, and denoting $S_{\prod_{i\in I}\mathbf{A}_i} = \bigcup_{k\in\mathbb{N}}\{(a_i)_{i\in I} \in \prod_{i\in I} S_{\mathbf{A}_i} : \mathrm{depth}_{\mathbf{A}_i}(a_i) = k\}$, we say that the function $\psi : S_{\mathbf{B}} \mapsto S_{\prod_{i\in I}\mathbf{A}_i}$ is a *forest product isomorphism for the product* $\mathbf{B} = \prod_{i\in I}\mathbf{A}_i$ if:

1. it is a bijection,
2. for any $b \in S_{\mathbf{B}}$, $\psi(b)$ is a root if and only if b is a root, and
3. for any families of non-root states $(s_i)_{i\in I}, (s'_i)_{i\in I} \in S_{\prod_{i\in I}\mathbf{A}_i}$, we have: $\psi^{-1}((s_i)_{i\in I}) \to \psi^{-1}((s'_i)_{i\in I})$ is an edge of $\mathbf{B}$ if and only if for each $i \in I$, $s_i \to s'_i$ is an edge of $\mathbf{A}_i$.

As for the first definition of a product isomorphism, if there is a tree product isomorphism between $\mathbf{B}$ and $\prod_{i\in I}\mathbf{A}_i$, this means that $\mathbf{B} = \prod_{i\in I}\mathbf{A}_i$. A simple inductive proof shows that:

Lemma 37. *Given a tree product isomorphism ψ for a product $\boldsymbol{B} = \prod_{i\in I}\boldsymbol{A}_i$ is such that for any $(a_i)_{i\in I} \in S_{\prod_{i\in I}\boldsymbol{A}_i}$ and $b \in S_{\boldsymbol{B}}$, such that $\psi(b) = (a_i)_{i\in I}$, we have $\mathrm{depth}_{\boldsymbol{B}}(b) = \mathrm{depth}_{\prod_{i\in I}\boldsymbol{A}_i}((a_i)_{i\in I})$.*

Lemma 38. *Let $A, B, C \in \mathbb{D}_D$, and let $k \geq \mathrm{depth}(A)$. Then $A = BC$ if and only if $[\widetilde{A}]_k = [\widetilde{B}]_k[\widetilde{C}]_k$.*

Proof. Remember that we already know that $A = BC \iff \widetilde{A} = \widetilde{B}\widetilde{C}$. Now, one direction is trivial: if $A = BC$, then $\widetilde{A} = \widetilde{B}\widetilde{C}$ so $[\widetilde{A}]_k = [\widetilde{B}]_k[\widetilde{C}]_k$ for every k. Now, we assume that $[\widetilde{A}]_k = [\widetilde{B}]_k[\widetilde{C}]_k$ for some $k \geq \mathrm{depth}(A)$ and we show that $\widetilde{A} = \widetilde{B}\widetilde{C}$.

Now, we want to create a tree product isomorphism $\phi : S_{\widetilde{A}} \to S_{\widetilde{B}\widetilde{C}}$ for the product $\widetilde{A} = \widetilde{B}\widetilde{C}$. To do so, we start from the tree product isomorphism $\psi : S_{[\widetilde{A}]_k} \to S_{[\widetilde{B}]_k[\widetilde{C}]_k}$ for the product $[\widetilde{A}]_k = [\widetilde{B}]_k[\widetilde{C}]_k$.

We can extend ψ to ϕ easily. For all $(a,d) \in S_A \times \mathbb{N}$ where $d \geq \mathrm{depth}_A(a)$, set $\phi(a,d) = ((b,d),(c,d))$ where $\psi(a, \mathrm{depth}_A(a)) = ((b, \mathrm{depth}_A(a)), (c, \mathrm{depth}_A(a)))$. This is a well-defined function since $\psi(a, \mathrm{depth}_A(a))$ will always exist as $k \geq \mathrm{depth}(A)$.

Let's prove that this is a valid tree product isomorphism. First, ϕ is bijective. Indeed, suppose that $\psi(a,d) = \psi(a',d')$. Denote by $\psi(a,d) = ((b,d),(c,d))$ and $\psi(a',d') = ((b',d'),(c',d'))$. We directly have $(b,c,d) = (b',c',d')$. This means that $\mathrm{depth}_A(a) = \mathrm{depth}_A(a')$, by definition of ϕ, because b and c are at the same depth as a (this follows from Lemma 37, since ψ is a tree product isomorphism). This means that $\psi(a, \mathrm{depth}_A(a)) = \psi(a', \mathrm{depth}_A(a))$, which implies $a = a'$ by bijectivity of ψ.

Now, for any $(a,d) \in S_A \times \mathbb{N}$ such that $d \geq \mathrm{depth}_A(a)$, $\phi(a,d) = ((b,d),(c,d))$ is a root if and only if $d = 0$ and b and c are roots. Because of the definition of ψ, b and c are roots if and only if a is a root in A, since ψ is a tree product isomorphism.

For the last property we need to check, we write $x \xrightarrow{\mathbf{C}} y$ to mean that there is an edge from $x \in S_{\mathbf{C}}$ to $y \in S_{\mathbf{C}}$ in $\mathbf{C}$.

Finally, we show that for all $((b,d),(c,d)),((b',d'),(c',d')) \in S_{\widetilde{B}\widetilde{C}}$, we have: $\phi^{-1}(((b,d),(c,d))) \xrightarrow{\widetilde{A}} \phi^{-1}(((b',d'),(c',d')))$ if and only if $(b,d) \xrightarrow{\widetilde{B}} (b',d')$ and $(c,d) \xrightarrow{\widetilde{C}} (c',d')$. Indeed, following the definition of ϕ from ψ, we can write $\phi^{-1}(((b,d),(c,d))) = (a,d) \in S_{\widetilde{A}}$ and $\phi^{-1}(((b',d'),(c',d'))) = (a',d') \in S_{\widetilde{A}}$.

Since ψ is a tree product isomorphism, there is an edge $(a,d) \xrightarrow{\widetilde{A}} (a',d')$ if and only if there is an edge $((b, \mathrm{depth}_A(a)), (c, \mathrm{depth}_A(a))) \xrightarrow{[\widetilde{B}]_k[\widetilde{C}]_k} ((b', \mathrm{depth}_A(a)), (c', \mathrm{depth}_A(a)))$, that is, if and only if there is an edge $((b,d),(c,d)) \xrightarrow{\widetilde{B}\widetilde{C}} ((b',d),(c',d))$, which is equivalent to the existence of $(b,d) \xrightarrow{\widetilde{B}} (b',d')$ and $(c,d) \xrightarrow{\widetilde{C}} (c',d')$.

This proves that $\widetilde{A} = \widetilde{B}\widetilde{C}$, which in turn proves that $A = BC$, and concludes. □

Theorem 39. *Given $A, B \in \mathbb{D}_D$, we can find $C \in \mathbb{D}_D$ such that $A = BC$ or prove that it does not exist in polynomial time in the sizes of A and B.*

Proof. Given $A, B \in \mathbb{D}_D$, let $k = \text{depth}(A)$. Then, call $divide([\widetilde{A}]_k, [\widetilde{B}]_k)$. If this function returns $\perp$, then there is no $X \in \mathbb{F}_T$ such that $[\widetilde{A}]_k = [\widetilde{B}]_k X$, which shows that there is no $C \in \mathbb{D}_D$ such that $A = BC$ by Lemma 38.

Otherwise, if this function returns some $X \in \mathbb{F}_T$, then we have $[\widetilde{A}]_k = [\widetilde{B}]_k X$ with $\text{depth}(X) = k$. Now, remark that if there is some $C \in \mathbb{D}_D$ such that $A = BC$, we have $\text{depth}(C) \leq k$ and thus $[\widetilde{A}]_k = [\widetilde{B}]_k [\widetilde{C}]_k$, so by Lemma 21, we have $X = [\widetilde{C}]_k$. Therefore, if the function returns an $X \in \mathbb{F}_T$, either X is of the form $[\widetilde{C}]_k$ for some $C \in \mathbb{D}_D$, and then we recover C such that $A = BC$ from the reverse direction of Lemma 38, or X is not of that form, and by Lemma 21, there is no $C \in \mathbb{D}_D$ such that $A = BC$.

The $divide$ algorithm is indeed in polynomial time since a call to $divide(\mathbf{T}, \mathbf{U})$ ends up making at most one call to $divide(\mathbf{V}, \mathbf{W})$ for $\mathbf{V}$ some subtree of $\mathbf{T}$ and $\mathbf{W}$ some subtree of $\mathbf{U}$. Since every operation in a call to $divide$ is in polynomial time, this concludes. □

5. Unicity of k-th roots

Using Theorem 34, we can prove a simple result above polynomials, which in particular states that a polynomial with a coefficient of degree 1 which is cancellative is injective.

Proposition 40. *Let $P = \sum_{i=0} a_i X^i \in \mathbb{D}[X]$ and $A, B \in \mathbb{D}$ such that $P(A) = P(B)$. Then, we have $A = B$ if $a_1 \in \mathbb{D}^*$ or if for some $i > 1$, $a_i \in \mathbb{D}^*$ and $A \in \mathbb{D}^*$.*

Proof. Write $P(X) = \sum_{i=0}^{d} a_i X^i$. We can assume that $a_0 = 0$, and we still have $P(A) = P(B)$. Let $D = \sum_{i=1}^{d} a_i \sum_{j=0}^{i-1} A^{i-1-j} B^j$. Then:

$$\begin{aligned}
AD &= \sum_{i=1}^{d} a_i \sum_{j=0}^{i-1} A^{i-j} B^j \\
&= \sum_{i=1}^{d} a_i \left(A^i + \sum_{j=1}^{i-1} A^{i-j} B^j \right) \\
&= P(A) + \sum_{i=1}^{d} a_i \sum_{j=1}^{i-1} A^{i-j} B^j \\
&= P(B) + \sum_{i=1}^{d} a_i \sum_{j=1}^{i-1} A^{i-j} B^j \\
&= \sum_{i=1}^{d} a_i \left(B^i + \sum_{j=1}^{i-1} A^{i-j} B^j \right) \\
&= \sum_{i=1}^{d} a_i \sum_{j=1}^{i} A^{i-j} B^j \\
&= \sum_{i=1}^{d} a_i \sum_{j=0}^{i-1} A^{i-1-j} B^{j+1} \\
&= BD.
\end{aligned}$$

In the case where a_1 has a fixed point, remark that the term for $i = 1$ in $D = \sum_{i=1}^{d} a_i \sum_{j=0}^{i-1} A^{i-1-j} B^j$ is simply a_1, and so, D has a fixed point. Otherwise, in the case where there is $i > 1$ such that a_i with a fixed point, and A has a fixed point, the term in the sum for that i is: $a_i \sum_{j=0}^{i-1} A^{i-1-j} B^j$, in which we find the term $a_i A^{i-1}$, which has a fixed point, so $D \in \mathbb{D}^*$.

Since $D \in \mathbb{D}^*$, $AD = BD$ implies $A = B$. □

A general characterisation of injective polynomials would be very interesting. It seems unlikely that the condition $a_1 \in \mathbb{D}^*$ is necessary since that would mean that if $a_1 \notin \mathbb{D}^*$ then, even if every other coefficient is in $\mathbb{D}^*$, one could find $A \neq B$ such that $P(A) = P(B)$.

In the rest of this section, we show that for any $k \geq 1$, the polynomial $P(X) = X^k$ is injective.

Theorem 41. *For all $k \geq 1$ and $A, B \in \mathbb{D}$, if $A^k = B^k$, then $A = B$.*

Our first step is to prove the injectivity of the mapping $\mathbf{X} \mapsto \mathbf{X}^k$ on $\mathbb{F}$. Given a forest $\mathbf{F} \in \mathbb{F}$, let $\mathcal{R}(\mathbf{F}) \in \mathbb{F}_T$ be the tree obtained by joining all the trees of $\mathbf{F}$ to a new common root. More formally, if $\mathcal{F}(\mathbf{F})$ is the multiset of trees of $\mathbf{F}$, then $\mathcal{D}(\mathcal{R}(\mathbf{F})) = \mathcal{F}$.

Lemma 42. *For any forest $\mathbf{F} \in \mathbb{F}$ and any $k \geq 1$, we have $\mathcal{R}^k(\mathbf{F}) = \mathcal{R}(\mathbf{F})$.*

Proof. By Lemma 7, we have $\mathcal{D}(\mathcal{R}^k(\mathbf{F})) = \mathcal{F}^k(\mathbf{F})$. Now, it is clear that $\mathcal{F}^k(\mathbf{F}) = \mathcal{F}(\mathbf{F}^k)$. This concludes the proof. □

Lemma 43. *The mapping $\boldsymbol{X} \mapsto \boldsymbol{X}^k$ is injective on $\mathbb{F}$.*

Proof. We first prove that the mapping $\mathbf{X} \mapsto \mathbf{X}^k$ is injective on $\mathbb{F}_T$. Let $\mathbf{T}_1, \mathbf{T}_2 \in \mathbb{F}_T$ with $\mathbf{T}_1 < \mathbf{T}_2$. Then by induction on k, Corollary 25 shows that $\mathbf{T}_1^k < \mathbf{T}_2^k$.

We now prove injectivity on $\mathbb{F}$. Let $\mathbf{A}, \mathbf{B} \in \mathbb{F}$, such that $\mathbf{A}^k = \mathbf{B}^k$. By Lemma 42, we have $\mathcal{R}^k(\mathbf{A}) = \mathcal{R}^k(\mathbf{B})$. By injectivity on $\mathbb{F}_T$, we obtain $\mathcal{R}(\mathbf{A}) = \mathcal{R}(\mathbf{B})$, which implies $\mathbf{A} = \mathbf{B}$. □

Our second step is to prove the result for bijective FDSs.

Lemma 44. *Let $A, B \in \mathbb{D}$. If $A^k = B^k$, then $[A]_0 = [B]_0$.*

Proof. Remark that $A^k = B^k$ implies $[A]_0^k = [B]_0^k$. All that's left to show is that if $A, B \in \mathbb{D}_P$ and $A^k = B^k$, then $A = B$.

Take $D \in \mathbb{D}_P$, and write $D = \sum_i \lambda_i^A C_i$. Assume there exists $B = \sum_i \lambda_i^B C_i$ such that $B^k = D$. For all $i \in \mathbb{N}$, let $F_i = \{ L = (l_j)_{j \in [\![1,k]\!]} : \bigvee_j l_j = i \}$ be the possible ways a product of k cycles $C_{l_1} \times \cdots \times C_{l_k}$ is equal to some scalar multiple of C_i.

For any sequence $L = (l_j)$, we abuse notation and identify L with the multiset of its entries; we can then use the notation δ_L. By Lemma 12, we obtain for all $i \in \mathbb{N}$

$$\sum_{L \in F_i} \delta_L \prod_{j=1}^{k} \lambda_{l_j}^B = \lambda_i^A.$$

This is a set of triangular positive polynomial equations (as the equation for i only involves $\lambda_1^B, \dots, \lambda_i^B$), thus it has at most one solution. Therefore, if B exists, it is unique. □

Our third and final step proves the theorem.

Lemma 45. *Let $P \in \mathbb{N}[X]$ be a polynomial with coefficients in $\mathbb{N}$, and let $A \in \mathbb{D}$. Then, for any $\ell \in \mathbb{N}$, we have $\operatorname{supp}_{\leq \ell}(P(A)) = P(\operatorname{supp}_{\leq \ell}(A))$.*

Proof. Write $P = \sum_{i=1}^{d} a_i X^i$. If $A = \sum_{j=1}^{n} A_j$ where each A_j is connected, then the products that appear in $P(A)$ are the $a_i \prod_{k=1}^{i} A_{\beta_k}$ for each $i \in [\![1, n]\!]$ and $\beta = (\beta_k)_{k \in [\![1,i]\!]} \in [\![1, n]\!]^i$. Remark that for such a product $a_i \prod_{k=1}^{i} A_{\beta_k}$ to have a cycle length $\leq \ell$, every A_{β_k} must have cycle length $\leq \ell$. This concludes the proof. □

Lemma 46. *The mapping $A \mapsto A^k$ is injective on $\mathbb{D}$.*

Proof. Given A^k, we find $[A]_0$ and thus we know the lengths of the cycles of A by Lemma 44; denote them by $\ell_1 < \cdots < \ell_n$. We show by induction on $i \in [\![0, n]\!]$ that we can recover $\operatorname{supp}_{\leq \ell_i}(A)$ from A^k (with an implicit $\ell_0 = 0$, such that $\operatorname{supp}_{\leq \ell_0}(A) = 0$, to make for a trivial base case and avoid repetition).

Take some $i \in [\![1, n-1]\!]$ such that the induction hypothesis stands for i. We show that it also stands for $i+1$. By Lemma 45, $\operatorname{supp}_{\leq \ell_{i+1}}(A^k) = (\operatorname{supp}_{\leq \ell_{i+1}}(A))^k$. By the lemma's hypothesis, we recover $\widetilde{\operatorname{supp}_{\leq \ell_{i+1}}}(A)$ from $(\operatorname{supp}_{\leq \ell_{i+1}}(A))^k$. Now, since we have $\widetilde{\operatorname{supp}_{\leq \ell_i}}(A)$ from the induction hypothesis, we recover

$$\widetilde{\operatorname{supp}_{\ell_{i+1}}}(A) = \widetilde{\operatorname{supp}_{\leq \ell_{i+1}}}(A) \setminus \widetilde{\operatorname{supp}_{\leq \ell_i}}(A).$$

It is straightforward to reconstruct $\operatorname{supp}_{\ell_{i+1}}(A)$ from $\widetilde{\operatorname{supp}_{\ell_{i+1}}}(A)$ since we know there every tree in $\widetilde{\operatorname{supp}_{\ell_{i+1}}}(A)$ comes from a connected component of cycle length ℓ_{i+1}. And thus, we recover $\operatorname{supp}_{\leq \ell_{i+1}}(A) = \operatorname{supp}_{\ell_{i+1}}(A) + \operatorname{supp}_{\leq \ell_i(A)}$, which concludes the induction. □

6. A family of monoids with unique factorisation

The $C_2^2 = 2C_2$ identity shows that factorisation into irreducible FDSs is not unique on $\mathbb{D}$. Moreover, it is shown in [5] that factorisation is also not necessarily unique on $\mathbb{D}_D$, for example with the identity presented in Fig. 7. We can however exhibit an example of an interesting class of trees in which every element has a unique factorisation in irreducible FDSs. Although our example might not be useful in practice, it is interesting as a generalisation of the simpler result that shows that factorisation is unique on the multiplicative monoid generated by products of paths (which is called LD_1 with the notations below).

Definition 47. A *rhizome* is a path from a leaf to the fixed point in a dendron. The length of a rhizome is its number of transitions, that is its number of non-fixed point states.

According to our terminology, the depth of a dendron is the length of its longest rhizome.

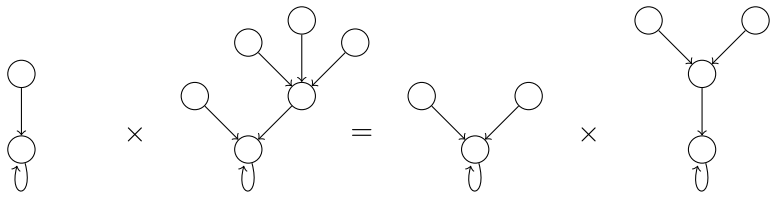


Fig. 7. A dendron that admits two different factorisations in irreducible factors.

Definition 48. An FDS $A \in \mathbb{D}$ is a *linear dendron* if it is a dendron such that only its fixed point may have more than one predecessor. A linear dendron has K rhizomes if its fixed point has K non-fixed point predecessors.

A *star* S_n is a linear dendron of depth 1 and n states, while a *path* P_n is a linear dendron with only one rhizome and $n+1$ states.

We are now in position to show that most linear dendrons are irreducible. We remark that the semigroup of stars is isomorphic to that of the positive integers: $S_{ab} = S_a \times S_b$. Therefore, composite stars have a unique factorisation in $\mathbb{D}$.

Proposition 49. *The only reducible linear dendrons are the stars with a composite number of states.*

Proof. The case of stars is straightforward. Let T be a linear dendron of depth $k > 1$. Then any rhizome of maximum length of T contains a state with exactly one predecessor: the state at depth 1 of the rhizome.

Suppose T is reducible towards a contradiction, say $T = A \times B$. The depth of either A or B is at least k, say P_k is a subdendron of A. Moreover, P_1 is a subdendron of B. Thus, $P_k \times P_1$ is a subdendron of $A \times B = T$. It's easy to see that $P_k \times P_1$ contains a path of depth k states with more than one predecessor each (except the leaf at the end). This is a rhizome of maximal length in T in which no state has exactly one predecessor. This concludes the proof. □

Definition 50. For all $K \in \mathbb{N}$, we define LD_K the multiplicative monoid generated by linear dendrons with K rhizomes.

Based on Proposition 49, if $P \in LD_K$ has a unique factorisation in LD_K, then it has a unique factorisation in $\mathbb{D}$. Thus, we focus on factorisation in LD_K.

Let $P \in LD_K$ be factorised as $P = F_1 \times \cdots \times F_N$ where F_j is a linear dendron for each $1 \le j \le N$. Each state $s \in S_P$ can be expressed as $s = (s_1, \dots, s_N)$ where $s_j \in S_{F_j}$ for all j. Some of those s_j's could be fixed points; let $I(s) = \{j : F_j(s_j) = s_j\}$. Then the number of predecessors of s is either 0 if any s_j is a leaf, or equal to $(K+1)^{|I(s)|}$ otherwise. This suggests the following notation.

Definition 51. Let $P \in LD_K$ and $i \in \mathbb{N}$. A state s of P is *i-fixed* if it has $(K+1)^i$ predecessors.

Lemma 52. *Any i-fixed state has a unique i-fixed predecessor; all other predecessors are either leaves or j-fixed for some $j < i$.*

Proof. Let $s = (s_1, \dots, s_N)$ be i-fixed and without loss let $I(s) = [\![1, i]\!]$. Remark that $s_1, \dots, s_i$ are fixed points, while $s_{i+1}, \dots, s_N$ have a unique predecessor each, say $t_{i+1}, \dots, t_N$ respectively. Then any predecessor of s is of the form $u = (u_1, \dots, u_i, t_{i+1}, \dots, t_N)$ where u_l is a predecessor of s_l for all $1 \le l \le i$. Therefore u is at most i-fixed, with equality if and only if $u = (s_1, \dots, s_i, t_{i+1}, \dots, t_N)$. □

Now that we have all the necessary definitions, we can introduce the following lemma, which enables a partial recovery of some factors from a product of linear dendrons. This is the core lemma, and it is from it that we can finally recover every factor.

Lemma 53 *(Linear extraction lemma). Let $P = F_1 \times \cdots \times F_N \in LD_K$ and let s be a depth 1, codepth ℓ, i-fixed state of P. Consider the tree anchored on s in P and remove the unique i-fixed predecessor of s and all its antecedents. Then the obtained dendron is $E_s = [\prod_{j \in I(s)} F_j]_\ell$.*

Proof. Without loss, let $I(s) = [\![1, i]\!]$. Let $s = (s_1, \dots, s_n)$ and for all $i+1 \le j \le N$ and $d \in \mathbb{N}$ let t_j^d be the unique state of F_j satisfying $F_j^d(t_j^d) = s_j$. All the states in the dendron E_s are either s or of the form $u = (u_1, \dots, u_i, t_{i+1}^d, \dots, t_N^d)$, where d is the depth of u in E_s and $(u_1, \dots, u_i) \neq (s_1, \dots, s_i)$. By removing the coordinates $i+1, \dots, N$ from each state, we see that E_s is a sub-FDS of $F_1 \times \cdots \times F_i$.

All that is left is to show that we do indeed get the truncature at depth ℓ. Remark that the codepth ℓ of s is the length of the smallest path among the rhizomes anchored at $s_{i+1}, \dots, s_N$ in their respective factors, minus 1. Thus, the sub-FDS of $F_1 \times \cdots \times F_i$ we obtain is indeed truncated at depth ℓ. □

Let $P = F_1 \times \cdots \times F_N$ where all the factors have depth $k+1$. Let $\mathfrak{A} = \{[F_i]_k : i \in [\![1, N]\!]\}$ be the collection of truncated factors and for each $B \in \mathfrak{A}$, denote its multiplicity by $n_B = |\{i \in [\![1, N]\!] : [F_i]_k = B\}|$. We denote by D_i the set of i-fixed depth 1 states of codepth k of P.

Lemma 54. *For all $B \in \mathfrak{A}$, there exists $s \in D_i$ with $E_s = B^i$ if and only if $i \leq n_B$.*

Proof. Without loss, let $B \in \mathfrak{A}$ such that $B = F_1 = \cdots = F_{n_B}$. Let $i \leq n_B$ and consider a state $s = (s_1, \ldots, s_N)$ of P where $s_1, \ldots, s_i$ are fixed points of B, while for every $i+1 \leq j \leq N$, s_i is a depth 1 state on a path of depth $k+1$. Then $s \in D_i$, and the extraction lemma extracts B^n from s. Conversely, if $E_s = B^j$, then B^j divides $[P]_k$ and hence $j \leq n_B$. □

We now show that factorisation is unique on products of linear dendrons which share the same depth.

Lemma 55. *A product of elements of LD_K which have the same depth k is uniquely factorisable.*

Proof. We do this by induction on the depth k. For $k = 0$, this lemma is obvious (the factorisation is C_1). Take some k such that the lemma stands for depth k. We show that the lemma is also true for depth $k+1$. The proof is in four steps. First, we identify the number of factors, then we recover the set of their depth k truncatures, then we recover the multiset of these truncatures and finally, we recover the full, untruncated factors. Take P a product of elements of LT_K.

Number of factors. We recover N the number of factors of P by remarking that its fixed point is N-fixed: thus by counting its number of predecessors, we can recover N from P and write $P = F_1 \times \cdots \times F_N$.

Set of truncatures. According to Lemma 54, by applying the extraction lemma to all the elements of D_1, we recover all the factors $B \in \mathfrak{A}$.

Multiset of truncatures. By Lemma 54, for all $B \in \mathfrak{A}$, $n_B = \max\{i : \exists s \in D_i E_s = B^i\}$. As such, applying the extraction lemma on D_i for $1 \leq i \leq N$ then yields n_B for all $B \in \mathfrak{A}$.

Untruncated factors. As of now, we have all the factors and their multiplicity, but they are truncated at depth k. To fully reconstruct the linear dendron F_i of depth $k+1$ from $[F_i]_k$, all we need is the number f_i of paths of depth $k+1$ in F_i. We now show how to determine this number.

Fix $B \in \mathfrak{A}$. Let's denote by $f_1, \ldots, f_{n_B}$ the number of paths of depth $k+1$ of and let $G_1, \ldots, G_{n_B}$ be the elements of $\phi(B)$. For any $n \in [\![0, n_B]\!]$, let's count in P the number of states of D_{N-n} from which the extraction lemma extracts $[P]_k / B^n$. Each of these states corresponds to an n-uple of depth 1 states of $G_1, \ldots, G_{n_B}$ (each in a distinct factor) on which a path of depth $k+1$ is anchored. As such, there are $p_n := \sum_{\substack{I \subseteq [\![1, n_B]\!] \\ |I| = n}} \prod_{i \in I} f_i$ of them (given the set of factors of $G_1, \ldots, G_{n_B}$ of index in I, the number of depth 1 states of codepth $k+1$ is $\prod_{i \in I} f_i$). Finding that number for all $n \in [\![0, n_B]\!]$ makes it possible to express the $f_1, \ldots, f_{n_B}$ as the roots of a polynomial of degree n_B and thus, allows one to find them. Here is how we proceed. Write $R(X) = \sum_{m=0}^{n_B} (-1)^m p_m X^{n_B - m}$. By Vieta's relations, we know that the n_B roots of R are $f_1, \ldots, f_{n_B}$. □

Now, we show that we can always get to this case:

Theorem 56. *Factorisation is unique on LD_K.*

Proof. Let $P = F_1 \times \cdots \times F_N \in LD_K$ have depth $k+1$. Let $I = \{1 \leq i \leq N : \mathrm{depth}(F_i) \leq k\}$ be the set of indices of factors with no paths of depth $k+1$. Now, let S be the set of depth 1 states belonging to a rhizome in P of depth $k+1$. For all $s \in S$, since s has depth 1 and codepth k, s_i is a fixed point for all $i \in I$ and hence s is at least $|I|$-fixed. Conversely, if $s \in S$ such that s_j has codepth k for all $j \notin I$, then s is $|I|$-fixed. Using the extraction lemma on such a state s, we recover $[\prod_{i \in I} F_i]_k = \prod_{i \in I} F_i$.

Let's divide S by $\prod_{i \in I} F_i$. The result is unique by Corollary 23. So, we get $\prod_{j \notin I} F_j$ the product of the factors of depth $k+1$, and $\prod_{i \in I} F_i$ the product of the factors of depth at most k. An induction on the second subproduct means that we can extract all the subproducts of shared depth, and apply the previous lemma on each of them. □

7. Conclusion

In this article, we have obtained results which may lead to a deeper understanding of the structure of the semiring of FDSs $\mathbb{D}$. In particular, we have characterised the cancellative elements of $\mathbb{D}$, shown how to perform division of dendrons in polynomial time, proved that k-th roots are unique, and we have exhibited a family of monoids with unique factorisation. While this sheds some light on the structure of $\mathbb{D}$, there are still many questions.

An interesting direction is the complexity of division on general FDSs, or on cycles. Contrary to the situation on trees, this algorithmic problem may not be in P. On the other hand, it is clearly in NP. The question of knowing whether it is NP-complete is still open, as a reduction (if it exists) does not seem obvious at all.

Another important direction to better understand the structure of $\mathbb{D}$ is the study of primality, defined as follows: $A \in \mathbb{D}$ is prime if and only if for every $B, C \in \mathbb{D}$, $A | BC$ implies $A | B$ or $A | C$. Most of the work on this has been done in [5], in which Couturier proves that for an FDS to be prime, it must be a dendron. Still, as of now, no example of a prime FDS is known, and no finite-time algorithm to check primality is known.

One could also be interested in more practical applications of FDS factorisation. Imagine for example a "grey box" (some deterministic mechanism that does not display its internal workings, but displays its state such that two different states can always be recognized) that is observed by a probe that records the evolution of its state, until this state falls into a cycle, at which point the

probe launches the process again, and so on. Thus, the probe reconstructs the FDS S governing the evolution of the grey box's state. We are interested in a way to know, with the current partial recovery of S, how many more states we need to add at the minimum in order to get a factorisable system. This is useful because suppose that the probabilistic model of exploration shows that there is a 90% chance that the probe has recovered at least 90% of the states of S. Then, if we know that, say, in order to get a factorisable recovered system, we need to add at least 30% more states than the ones we already have recovered, we know that with probably at least 90%, the grey box is not factorisable, that is, it does not contain two independent mechanisms running in parallel.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

References

[1] Florian Bridoux, Antonio E. Porreca, Composing Behaviours in the Semiring of Dynamical Systems, January 2020 Talk given at the International Workshop on Boolean Networks (IWBN 2020), Universidad de Concepción, Chile.
[2] P. Bak, C. Tang, K. Wiesenfeld, Self-organized criticality: an explanation of 1/f noise, Phys. Rev. Lett. 59 (4) (1987) 381–384.
[3] Jørgen Bang-Jensen, Gregory Gutin, Digraphs: Theory, Algorithms and Applications, Springer, 2009.
[4] Gilles Bernot, Jean-Paul Comet, Adrien Richard, Madalena Chaves, Jean-Luc Gouzé, Frédéric Dayan, Modeling in computational biology and biomedicine, in: Modeling and Analysis of Gene Regulatory Networks, Springer-Verlag, 2013, pp. 47–80.
[5] Johan Couturier, Méthodes algébriques et algorithmiques pour la décomposition de systèmes dynamiques, Final university year thesis, 2021, in French, unpublished, Aix-Marseille Université.
[6] Alberto Dennunzio, Valentina Dorigatti, Enrico Formenti, Luca Manzoni, Antonio E. Porreca, Polynomial equations over finite, discrete-time dynamical systems, in: Giancarlo Mauri, Samira El Yacoubi, Alberto Dennunzio, Katsuhiro Nishinari, Luca Manzoni (Eds.), Cellular Automata, Springer International Publishing, Cham, 2018, pp. 298–306.
[7] François Doré, Enrico Formenti, Antonio E. Porreca, Sara Riva, Algorithmic reconstruction of discrete dynamics, arXiv:2208.08310, September 2022.
[8] Valentina Dorigatti, Algorithms and complexity of the algebraic analysis of finite discrete dynamical systems, Master's thesis, Università degli Studi di Milano Bicocca, 2017.
[9] Andrzej Ehrenfeucht, Grzegorz Rozenberg, Reaction systems, Fundam. Inform. 75 (1–4) (2007) 263–280.
[10] Maximilien Gadouleau, Adrien Richard, Søren Riis, Fixed points of Boolean networks, guessing graphs, and coding theory, SIAM J. Discrete Math. 29 (4) (2015) 2312–2335.
[11] Maximilien Gadouleau, Søren Riis, Graph-theoretical constructions for graph entropy and network coding based communications, IEEE Trans. Inf. Theory 57 (10) (October 2011) 6703–6717.
[12] Caroline Gaze-Maillot, Antonio E. Porreca, Profiles of dynamical systems and their algebra, arXiv:2008.00843.
[13] Carlos Gershenson, Introduction to random Boolean networks, arXiv e-prints, nlin/0408006, August 2004.
[14] Eric Goles, Servet Martínez, Neural and Automata Networks: Dynamical Behavior and Applications, Kluwer Academic Publishers, Norwell, MA, USA, 1990.
[15] Richard Hammack, Wilfried Imrich, Sandi Klavzar, Handbook of Product Graphs, 2nd edition, CRC Press, Inc., USA, 2011.
[16] U. Hebisch, H.J. Weinert, Semirings: Algebraic Theory and Applications in Computer Science, World Scientific, 1998.
[17] L. Lovász, On the cancellation law among finite relational structures, Period. Math. Hung. 1 (2) (1971) 145–156.
[18] R. Thomas, Boolean formalization of genetic control circuits, J. Theor. Biol. 42 (3) (1973) 563–585.
[19] René Thomas, Richard D'Ari, Biological Feedback, CRC Press, 1990.