ALGEBRAIC SYNCHRONIZATION TREES AND PROCESSES

LUCA ACETO, ARNAUD CARAYOL, ZOLTÁN ÉSIK, AND ANNA INGÓLFSDÓTTIR

ICE-TCS, School of Computer Science, Reykjavik University, Iceland

Université Paris-Est, LIGM (UMR 8049), CNRS, ENPC, ESIEE, UPEM, France

Institute of Informatics, University of Szeged, Hungary

ICE-TCS, School of Computer Science, Reykjavik University, Iceland

ABSTRACT. We study algebraic synchronization trees, i.e., initial solutions of algebraic recursion schemes over the continuous categorical algebra of synchronization trees. In particular, we investigate the relative expressive power of algebraic recursion schemes over two signatures, which are based on those for Basic CCS and Basic Process Algebra, as a means for defining synchronization trees up to isomorphism as well as modulo bisimilarity and language equivalence. The expressiveness of algebraic recursion schemes is also compared to that of the low levels in Caucal's pushdown hierarchy.

1. INTRODUCTION

The study of recursive program schemes is one of the classic topics in programming language semantics. (See, e.g., [5, 24, 31, 38, 41] for some of the early references.) One of the main goals of this line of research is to define the semantics of systems of recursive equations such as

$$F(n) = \text{ifzero}(n, 1, \text{mult}(2, F(\text{pred}(n)))). \tag{1.1}$$

In the above recursion scheme, the symbols ifzero, add, pred, 1 and 2 denote given function symbols; these are used to define the derived unary function $F(n)$, which we will refer to as a functor variable. Interpreting ifzero as the function over the set of triples of $\mathbb{N}^3$ that returns its second argument when the first is zero and the third otherwise, mult as multiplication and pred as the predecessor function, intuitively one would expect the above

recursion scheme to describe the function over the natural numbers that, given an input n, returns 2^n. This expectation has been formalized in several ways in the literature on recursive program schemes. A classic answer can be summarized by the 'motto' of the initial-algebra-semantics approach: 'The semantics of a recursive program scheme is the infinite term tree (or ranked tree) that is the least fixed point of the system of equations associated with the program scheme.'

In the light of the role that infinite term trees play in defining the semantics of recursive program schemes, it is not surprising that the study of infinite term trees has received a lot of attention in the research literature. Here we limit ourselves to mentioning Courcelle's classic survey paper [24], which presents results on topological and order-theoretic properties of infinite trees, notions of substitutions for trees as well as regular and algebraic term trees. *Algebraic term trees* are those that arise as solutions of recursive program schemes that, like (1.1), are 'first order'. On the other hand, *regular term trees* are the solutions of systems of equations like

$$
\begin{aligned}
X &= f(X, Y) \\
Y &= a,
\end{aligned}
$$

which define parameterless functions X and Y. Regular term trees arise naturally as the unfoldings of flowcharts, whereas algebraic term trees stem from the unfoldings of recursion schemes that correspond to functional programs [24].

In this paper, we study recursion schemes from a process-algebraic perspective and investigate the expressive power of algebraic recursion schemes over the signatures of Basic CCS [36] and of Basic Process Algebra (BPA) [3] as a way of defining possibly infinite synchronization trees [35]. Both these signatures allow one to describe every finite synchronization tree and include a binary choice operator $+$. The difference between them is that the signature for Basic CCS, which is denoted by Γ in this paper, contains a unary action prefixing operation $a.\_$ for each action a, whereas the signature for BPA, which we denote by Δ, has one constant a for each action that may label the edge of a synchronization tree and offers a full-blown sequential composition, or sequential product, operator. Intuitively, the sequential product $t \cdot t'$ of two synchronization trees is obtained by appending a copy of t' to the leaves of t that describe successful termination of a computation. In order to distinguish successful and unsuccessful termination, both the signatures Γ and Δ contain constants 0 and 1, which denote unsuccessful and successful termination, respectively.

As an example of a regular recursion scheme over the signature Δ, consider

$$X = (X \cdot a) + a.$$

On the other hand, the following recursion scheme is Γ-algebraic, but not Γ-regular:

$$
\begin{aligned}
F_1 &= F_2(a.1) \\
F_2(v) &= v + F_2(a.v).
\end{aligned}
$$

It turns out that both these recursion schemes define the infinitely branching synchronization tree depicted on Figure 3.

In the setting of process algebras such as CCS [36] and ACP [3], synchronization trees are a classic model of process behaviour. They arise as unfoldings of labelled transition systems that describe the operational semantics of process terms and have been used to give denotational semantics to process description languages—see, for instance, [1]. Regular synchronization trees over the signature Γ are unfoldings of processes that can be described

in the regular fragment of CCS, which is obtained by adding to the signature Γ a facility for the recursive definition of processes. On the other hand, regular synchronization trees over the signature Δ are unfoldings of processes that can be described in Basic Process Algebra (BPA) [3] augmented with constants for the deadlocked and the empty process as well as recursive definitions.

As is well known, the collection of regular synchronization trees over the signature Δ strictly includes that of regular synchronization trees over the signature Γ even up to language equivalence. Therefore, the notion of regularity depends on the signature. But what is the expressive power of algebraic recursion schemes over the signatures Γ and Δ? The aim of this paper is to begin the analysis of the expressive power of those recursion schemes as a means for defining synchronization trees, and their bisimulation or language equivalence classes.

In order to characterize the expressive power of algebraic recursion schemes defining synchronization trees, we interpret such schemes in continuous categorical Γ- and Δ-algebras of synchronization trees. Continuous categorical Σ-algebras are a categorical generalization of the classic notion of continuous Σ-algebra that underlies the work on algebraic semantics [11, 26, 30, 31], and have been used in [9, 10, 28] to give semantics to recursion schemes over synchronization trees and words. (We refer the interested reader to [32] for a recent discussion of category-theoretic approaches to the solution of recursion schemes.) In this setting, the Γ-regular (respectively, Γ-algebraic) synchronization trees are those that are initial solutions of regular (respectively, algebraic) recursion schemes over the signature Γ. Δ-regular and Δ-algebraic synchronization trees are defined in similar fashion.

Our first contribution in the paper is therefore to provide a categorical semantics for first-order recursion schemes that define processes, whose behaviour is represented by synchronization trees. The use of continuous categorical Σ-algebras allows us to deal with arbitrary first-order recursion schemes; there is no need to restrict oneself to, say, 'guarded' recursion schemes, as one is forced to do when using a metric semantics (see, for instance, [14] for a tutorial introduction to metric semantics), and this categorical approach to giving semantics to first-order recursion schemes can be applied even when the order-theoretic framework either fails because of the lack of a 'natural' order or leads to undesirable identities.

As a second contribution, we provide a comparison of the expressive power of regular and algebraic recursion schemes over the signatures Γ and Δ, as a formalism for defining processes described by their associated synchronization trees. We show that each Δ-regular tree is Γ-algebraic (Theorem 3.2) by providing an algorithm for transforming a Δ-regular recursion scheme into an equivalent Γ-algebraic one that involves only unary functor variables. In addition, we prove that every synchronization tree that is defined by a Γ-algebraic recursion scheme of a certain form that involves only unary functor variables can be transformed into an equivalent Δ-regular recursion scheme.

We provide examples of Γ-algebraic synchronization trees that are not Δ-regular and of Δ-algebraic trees that are not Γ-algebraic, not even up to bisimulation equivalence. In particular, in Proposition 5.5 we prove that the synchronization tree associated with the bag over a binary alphabet (which is depicted on Figure 18 on page 39) is not Γ-algebraic, even up to language equivalence, and that it is not Δ-algebraic up to bisimilarity. These results are a strengthening of a classic theorem from the literature on process algebra proved by Bergstra and Klop in [6].
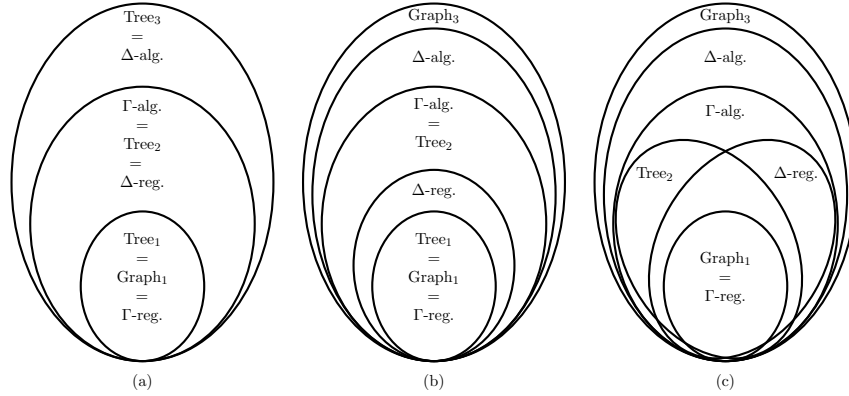
Figure 1: The expressiveness hierarchies up to language equivalence (a), up to bisimilarity (b) and up to isomorphism (c)

Since each Γ-algebraic synchronization tree is also Δ-algebraic, we obtain the following strict expressiveness hierarchy, which holds up bisimilarity [36, 39]:

$$\Gamma\text{-regular} \subset \Delta\text{-regular} \subset \Gamma\text{-algebraic} \subset \Delta\text{-algebraic}.$$

In the setting of language equivalence, the notion of Γ-regularity corresponds to the regular languages, the one of Δ-regularity or Γ-algebraicity corresponds to the context-free languages, and Δ-algebraicity corresponds to the macro languages [29], which coincide with the languages generated by Aho's indexed grammars [2].

In order to obtain a deeper understanding of Γ-algebraic recursion schemes, we characterize their expressive power by following the lead of Courcelle [22, 23, 24]. In those references, Courcelle proved that a term tree is algebraic if, and only if, its branch language is a deterministic context-free language. In our setting, we associate with each synchronization tree with bounded branching a family of branch languages and we show that a synchronization tree with bounded branching is Γ-algebraic if, and only if, the family of branch languages associated with it contains a deterministic context-free language (Theorem 5.2). In conjunction with standard tools from formal language theory, this result can be used to show that certain synchronization trees are not Γ-algebraic.

As a final main contribution of the paper, we compare the expressiveness of those recursion schemes to that of the low levels in Caucal's hierarchy. This hierarchy is already known to include the term trees defined by safe[1] higher-order recursion schemes when interpreted over the free continuous algebra [18]. Unsurprisingly, we show that the classes of Γ-algebraic and Δ-algebraic synchronization trees belong to the third level of the Caucal hierarchy. We provide a more detailed comparison, which is summarized in Figure 1.

As a benefit of the comparison with the Caucal hierarchy, we obtain structural properties and decidability of the monadic second-order theories of Δ-algebraic synchronization trees [44]. By contraposition, this implies that a synchronization tree with an undecidable monadic second-order theory cannot be Δ-algebraic. This allows us to show that Γ-algebraic (and hence Δ-algebraic trees) are not closed under minimization with respect to bisimulation equivalence (Proposition 6.4).

---

[1]Safety is a syntactic restriction which is trivial for first order schemes. In particular, it does not play any role in our setting.

The technical developments in this paper make use of techniques and results from a variety of areas of theoretical computer science. We employ elementary tools from category theory and initial-algebra semantics to define the meaning of recursion schemes over algebras of synchronization trees. Tools from concurrency and formal-language theory are used to obtain separation results between the different classes of synchronization trees we consider in this paper. The proof of Theorem 5.2, characterizing the expressive power of Γ-algebraic recursion schemes in the style of Courcelle, uses a Mezei-Wright theorem for categorical algebras [10] as well as tools from monadic second-order logic [13, 15]. Overall, we find it pleasing that methods and results developed by different communities within theoretical computer science play a role in the study of natural structures like the synchronization trees that arise from the solution of algebraic recursion schemes.

The paper is organized as follows. In Section 2, we introduce (first-order) recursion schemes and the synchronization trees they define using continuous categorical algebras. In Section 3, we compare the classes of Γ-regular, Δ-regular, Γ-algebraic and Δ-algebraic synchronization trees up to isomorphism, bisimulation and language equivalence. In Section 4, we compare our classes of synchronization trees to the first levels of the Caucal hierarchy. In Section 5, following Courcelle, we characterize the expressive power of Γ-algebraic recursion schemes by studying the branch languages of synchronization trees whose vertices have bounded outdegree. In Section 6, we show how the previous results can be used to prove that some synchronization trees cannot be defined using algebraic recursion schemes. Section 7 concludes the paper and lists topics for future research.

2. CATEGORICAL SEMANTICS OF FIRST-ORDER RECURSION SCHEMES

In this section, we provide a categorical semantics for first-order recursion schemes using continuous categorical algebras, and introduce several notions and results that will be used throughout the paper. Section 2.1 introduces the necessary categorical background. The continuous categorical algebras associated with synchronization trees are introduced in Section 2.2. Sections 2.3–2.5 present edge-labelled graphs, monadic second order logic over such structures and the kinds of graph transformations used in the paper. The definition of the synchronization tree defined by a recursion scheme is given in Section 2.6. Sections 2.7 and 2.8 describe the essentially unique morphisms from the initial continuous ordered Γ-algebra (resp. Δ-algebra) of Γ-term trees (resp. Δ-term trees) to the continuous Γ-algebra (resp. Δ-algebra) of synchronization trees. These morphisms play an important role when comparing the synchronization trees defined by recursion schemes to the low levels of the Caucal hierarchy in Section 4. We conclude by giving some basic properties of synchronization trees defined by first-order recursion schemes in Section 2.9.

Throughout the paper, when n is a non-negative integer, we denote the set $\{1, \ldots, n\}$ by $[n]$.

2.1. **Continuous categorical algebras.** In this section, we recall the notion of continuous categorical Σ-algebra. These structures were used in [9, 10, 28] to give semantics to recursion schemes over synchronization trees and words.

Let $\Sigma = \bigcup_{n \geq 0} \Sigma_n$ be a ranked set (or 'signature'). A *categorical Σ-algebra* is a small category C equipped with a functor $\sigma^C : C^n \to C$ for each $\sigma \in \Sigma_n$, $n \geq 0$. A *morphism*

between categorical Σ-algebras C and D is a functor $h : C \to D$ such that for each $\sigma \in \Sigma_n$, the diagram

$$
\begin{array}{ccc}
C^n & \xrightarrow{\ \sigma^C\ } & C \\
\downarrow{\scriptstyle h^n} & & \downarrow{\scriptstyle h} \\
D^n & \xrightarrow{\ \sigma^D\ } & D
\end{array}
$$

commutes *up to a natural isomorphism.* Here, the functor $h^n : C^n \to D^n$ maps each object and morphism $(x_1, \ldots, x_n)$ in C^n to $(h(x_1), \ldots, h(x_n))$ in D^n.

Suppose that C is a categorical Σ-algebra. We call C *continuous* if C has a distinguished initial object (denoted $\bot^C$ or 0^C) and colimits of all ω-diagrams $(f_k : a_k \to a_{k+1})_{k \geq 0}$. Moreover, each functor σ^C is continuous, i.e., preserves colimits of ω-diagrams. Thus, if $\sigma \in \Sigma_2$, say, and if

$$x_0 \xrightarrow{f_0} x_1 \xrightarrow{f_1} x_2 \xrightarrow{f_2} \ldots$$
$$y_0 \xrightarrow{g_0} y_1 \xrightarrow{g_1} y_2 \xrightarrow{g_2} \ldots$$

are ω-diagrams in C with colimits $(x_k \xrightarrow{\phi_k} x)_k$ and $(y_k \xrightarrow{\psi_k} y)_k$, respectively, then

$$\sigma^C(x_0, y_0) \xrightarrow{\sigma^C(f_0, g_0)} \sigma^C(x_1, y_1) \xrightarrow{\sigma^C(f_1, g_1)} \sigma^C(x_2, y_2) \xrightarrow{\sigma^C(f_2, g_2)} \ldots$$

has colimit

$$(\sigma^C(x_k, y_k) \xrightarrow{\sigma^C(\phi_k, \psi_k)} \sigma^C(x, y))_k.$$

A morphism of continuous categorical Σ-algebras is a categorical Σ-algebra morphism which preserves the distinguished initial object and colimits of all ω-diagrams. In what follows, we will often write just σ for σ^C, in particular when C is understood.

For later use, we note that if C and D are continuous categorical Σ-algebras then so is $C \times D$. Moreover, for each $k \geq 0$, the category $[C^k \to C]$ of all continuous functors $C^k \to C$ is also a continuous categorical Σ-algebra, where for each $\sigma \in \Sigma_n$,

$$\sigma^{[C^k \to C]}(f_1, \ldots, f_n) = \sigma^C \circ \langle f_1, \ldots, f_n \rangle$$

with $\langle f_1, \ldots, f_n \rangle$ standing for the target tupling of the continuous functors $f_1, \ldots, f_n : C^k \to C$. On natural transformations, $\sigma^{[C^k \to C]}$ is defined in a similar fashion. In $[C^k \to C]$, colimits of ω-diagrams are formed pointwise.

In the rest of the paper, we will assume, without loss of generality, that the signature Σ contains a special symbol denoted $\bot$ or 0 of rank 0, interpreted in a continuous categorical Σ-algebra as its initial object.

2.2. **Synchronization trees.** A *synchronization tree* $t = (V, v_0, E, l)$ over an alphabet A of 'action symbols' consists of

- a finite or countably infinite set V of 'vertices' and an element $v_0 \in V$, the 'root';
- a set $E \subseteq V \times V$ of 'edges';
- a 'labelling function' $l : E \to A \cup \{\mathsf{ex}\}$ where $\mathsf{ex} \notin A$ is a label used to denote successful termination.

These data obey the following restrictions.

- (V, v_0, E) is a rooted tree: for each $u \in V$, there is a unique path $v_0 \rightsquigarrow u$.
- If $e = (u, v) \in E$ and $l(e) = \mathsf{ex}$, then v is a leaf, and u is called an *exit vertex*.

A synchronization tree is *deterministic* if no node in the tree has two equally-labelled outgoing edges.

A *morphism* $\phi : t \to t'$ of synchronization trees is a function $V \to V'$ which preserves the root, the edges and the labels, so that if (u, v) is an edge of t, then $(\phi(u), \phi(v))$ is an edge of t', and $l'(\phi(u), \phi(v)) = l(u, v)$. In particular such a morphism maps the root of t to the root of t'. Morphisms are therefore functional *simulations* [34, 39]. It is clear that the trees over A and tree morphisms form a category, which we denote by $\mathsf{ST}(A)$. The tree that has a single vertex and no edges is initial. It is known that the category of trees has colimits of all ω-diagrams, see [8]. (It also has binary coproducts.) In order to make the category of trees small, we may require that the vertices of a tree form a subset of some fixed infinite set.

Remark 2.1. Suppose that $(\phi_n : t_n \to t_{n+1})_{n \geq 0}$ is an ω-diagram in $\mathsf{ST}(A)$, where $t_n = (V_n, v_n, E_n, l_n)$ for each $n \geq 0$. Then the colimit $\mathsf{Colim}(\phi_n : t_n \to t_{n+1})_{n \geq 0}$ can be constructed in the expected way. First, we define for each $n \leq m$ the tree morphism $\phi_{n,m}$ as the composition of the morphisms $\phi_n, \dots, \phi_{m-1}$ or the identify morphism if $n = m$. Then we take the disjoint union $\{(v, n) \mid v \in V_n \text{ and } n \geq 0\}$ of the sets V_n, $n \geq 0$, and define a vertex (v, n) equivalent to a vertex (v', m) if and only if there is some $k \geq \max\{n, m\}$ with $\phi_{n,k}(v) = \phi_{m,k}(v')$. The set of vertices of the colimit tree will be the equivalence classes of $\{(v, n) \mid v \in V_n \text{ and } n \geq 0\}$. The root will be the equivalence class containing the roots (v_n, n) of the trees t_n, $n \geq 0$. When C and C' are equivalence classes, the pair (C, C') is an edge in the colimit tree exactly when there exist $(v, n) \in C$ and $(v', n) \in C'$ for some n such that E_n contains the edge (v, v'). The label of this edge is $l((C, C')) = l_n((v, v'))$. Note that, in this case, for each $k \geq n$ we have that $(\phi_{n,k}(v), \phi_{n,k}(v')) \in E_k$, and that the label of this edge in t_k is equal to $l_n(v, v')$. This defines the object part t of the colimit. The canonical morphism $t_n \to t$ maps a vertex $v \in V_n$ to its equivalence class.

In the particular case when the morphisms ϕ_n are injective, we may usually assume that $V_n \subseteq V_{n+1}$ for each n and that the morphism ϕ_n is the inclusion of V_n into V_{n+1}. In this case the colimit is simply the 'union' of the trees t_n.

Consider the sequence $t_n = (\{0, \dots, n\}, 0, E_n, l_n)$, $n \geq 0$ of synchronization trees, where $E_n = \{(i, i+1) \mid 0 \leq i < n\}$ and l_n maps each edge in E_n to the action symbol a. Let $\psi_n : t_n \to t_{n+1}$ be the inclusion map from $\{0, \dots, n\}$ into $\{0, \dots, n+1\}$. Then $(\phi_n = t_n \to t_{n+1})_{n \geq 0}$ is an ω-diagram and its colimit is $(f_n : t_n \to t_\omega)_n$, where

$$t_\omega = (\{i \mid i \geq 0\}, 0, \{(i, i+1) \mid i \geq 0\}, l)$$

and l maps each edge to a. Pictorially, we have:

$$t_0 \qquad t_1 \qquad t_2 \qquad t_3 \qquad\qquad t_\omega$$
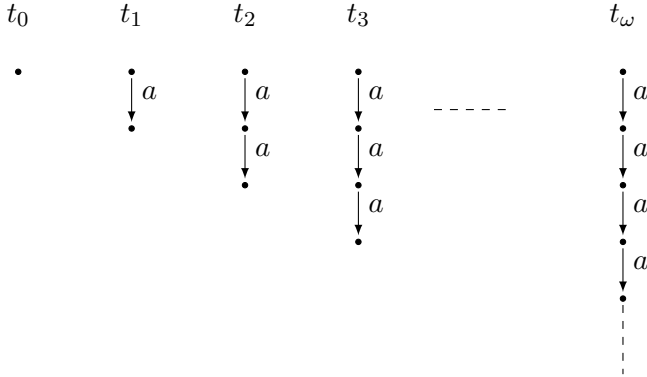
The category $\mathsf{ST}(A)$ of synchronization trees over A is equipped with two binary operations: $+$ (sum) and $\cdot$ (sequential product or sequential composition), and either with a unary operation or a constant associated with each letter $a \in A$.

The *sum* $t + t'$ of two trees is obtained by taking the disjoint union of the vertices of t and t' and identifying the roots. The edges and labelling are inherited. The *sequential product* $t \cdot t'$ of two trees is obtained by replacing each edge of t labelled ex by a copy of t'. With each letter $a \in A$, we can either associate a constant, or a unary *prefixing operation*. As a constant, a denotes the tree with vertices v_0, v_1, v_2 and two edges: the edge (v_0, v_1), labelled a, and the edge (v_1, v_2), labelled ex. As an operation $a(t)$ is the tree $a \cdot t$, for each tree t. Let 0 denote the tree with no edges and 1 the tree with a single edge labelled ex. On morphisms, all operations are defined in the expected way. For example, if $h : t \to t'$ and $h' : s \to s'$, then $h + h'$ is the morphism that agrees with h on the non-root vertices of t and that agrees with h' on the non-root vertices of s. The root of $t + s$ is mapped to the root of $t' + s'$.

In the sequel we will consider two signatures for synchronization trees, Γ and Δ. The signature Γ contains $+$, $0, 1$ and each letter $a \in A$ as a *unary* symbol. In contrast, Δ contains $+, \cdot$, $0, 1$ and each letter $a \in A$ as a *nullary* symbol. It is known that, for both signatures, $\mathsf{ST}(A)$ is a continuous categorical algebra. See [8] for details.

In what follows, for each $a \in A$ and term/tree t, we write $a.t$ for $a(t)$. We shall also abbreviate $a.1$ to a, and write $a^n.t$ for

$$\underbrace{a.\dots.a.}_{n \text{ times}} t.$$

Remark 2.2. Note that $\cdot$ is associative and has 1 as unit, at least up to isomorphism, and that for all trees t_1, t_2 and s, $(t_1 + t_2) \cdot s = (t_1 \cdot s) + (t_2 \cdot s)$ and $0 \cdot s = 0$ up to isomorphism.

Remark 2.3. A continuous categorical Δ-algebra C naturally induces an associated Γ-algebra D. For each letter $a \in A$, the functor a^D associated to unary symbol a is defined for each object and morphism x by $a^D(x) = a^C \cdot^C x$. The functors for the other symbols are inherited (i.e., for all $f \in \Sigma \setminus A$, $f^D = f^C$). It is easy to see that D is indeed a continuous categorical Γ-algebra. In particular the Δ-algebra of synchronization trees induces in this sense the Γ-algebra of synchronization trees.

Definition 2.4. Two synchronization trees $t = (V, v_0, E, l)$ and $t' = (V', v'_0, E', l')$ are *bisimilar* [36, 39] if there is some symmetric relation $R \subseteq (V \times V') \cup (V' \times V)$, called a *bisimulation*, that relates their roots, and such that if $(v_1, v_2) \in R$ and there is some edge (v_1, v'_1), then there is an equally-labelled edge (v_2, v'_2) with $(v'_1, v'_2) \in R$.
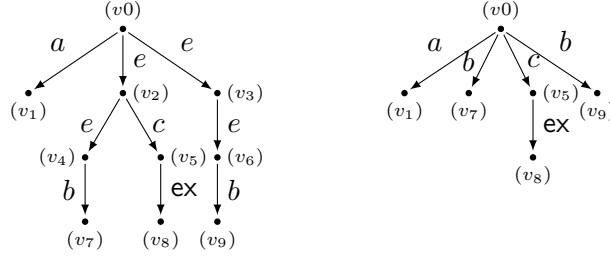
Figure 2: A synchronization tree (on the left) over $\{a, b, c, e\}$ and its $\{e\}$-contraction (on the right)

The *path language* of a synchronization tree is composed of the words in A^* that label a path from the root to the source of an exit edge. Two trees are *language equivalent* if they have the same path language.

Lemma 2.5. *If t and t' are deterministic, bisimilar synchronization trees, then they are isomorphic.*

Remark 2.6. The above lemma fails up to language equivalence. For example, the trees $\bullet$ and $\overset{\bullet}{\underset{\bullet}{|}} a$ are language equivalent but not isomorphic.

We conclude this section by defining an operation on synchronization trees over $A \uplus B$ that contracts all edges labelled by some symbol in B. The definition of this operation is illustrated in Figure 2. This notion will, in particular, be used to characterize Γ-algebraic trees in the Caucal hierarchy.

Definition 2.7. Let $t = (V, v_0, E, l)$ be synchronization tree over an alphabet $A \uplus B$. The B-contraction of t is the synchronization tree $t' = (V', v_0, E', l')$ defined by:
- $V' = \{v' \in V \mid \exists v \in V \ (v, v') \in E \text{ with } l(v, v') \in A \cup \{\mathsf{ex}\}\} \cup \{v_0\}$,
- a pair $(v, v') \in V' \times V'$ is an edge in E' if there exists a path in t from v to v' labelled by a word in $B^*(A \cup \{\mathsf{ex}\})$,
- for each $(v, v') \in E'$, $l(v, v')$ is the unique letter $a \in A \cup \{\mathsf{ex}\}$ such that the path from v to v' in t is labelled in B^*a.

2.3. **Edge-labelled graphs.** An *edge-labelled graph* (or simply graph), whose edges are labelled by letters in a finite alphabet A, is a pair (V, E) where V is a finite or countable set of vertices and $E \subseteq V \times A \times V$ is a set of labelled edges. The edge (u, a, v) has source u, target v and label a. The existence of an edge (u, a, v) in E is denoted by $u \xrightarrow[G]{a} v$ or simply $u \xrightarrow{a} v$ when G is clear from the context. This notation is extended to words in A^* in the usual way. For a language L over A, we write $u \xrightarrow{L} v$ if there is some word $w \in L$ such that $u \xrightarrow{w} v$.

A graph (V, E) is *deterministic* if for all $u, v_1, v_2 \in V$ and $a \in A$, if $u \xrightarrow{a} v_1$ and $u \xrightarrow{a} v_2$ then $v_1 = v_2$.

A *root* of a graph (V, E) labelled by A is a vertex v_0 such that for all $v \in V$, there exists a path from v_0 to v. If this path is unique for each vertex v, the graph (V, E) is a *tree* with v_0 as its root.

A graph (V, E) labelled in $A \cup \{\mathsf{ex}\}$ represents a synchronization tree if (V, E) is a tree with root v_0 and if the target of an ex-labelled edge is always a leaf of the tree. We will identify such a graph with the synchronization tree (V, v_0, E', l) where $E' = \{(u, v) \mid \exists a \in A \cup \{\mathsf{ex}\}.u \xrightarrow{a} v\}$ and for all $(u, v) \in E'$, $l((u, v))$ is the unique symbol in $A \cup \{\mathsf{ex}\}$ such that $(u, l((u, v)), v)$ belongs to E.

The definition of B-contraction for labelled graphs generalizes the case of synchronization trees introduced in Section 2.2. The B-contraction of a graph G labelled in $A \uplus B$ from one of its root vertices r is the graph labelled by A whose vertices are the targets (in G) of edges labelled in A together with the vertex r. There is an edge from a vertex u to a vertex v if there is a path from u to v in G labelled by a word in B^*A.

2.4. Monadic second-order logic on edge labelled graphs. *Monadic second-order logic (MSO)* on graphs is the extension of first-order logic with the ability to quantify over sets of vertices. We use monadic-second order logic over edge-labelled graphs with the standard syntax and semantics (see e.g. [27] for a detailed presentation).

Monadic second-order formulas are built using *first-order variables*, which are interpreted as vertices of the graph and are denoted by lower case letters $x, y \ldots$ and *monadic second-order variables*, which are interpreted as sets of vertices of the graph and are denoted by upper case letters $X, Y \ldots$ Atomic formulas are of the form $x \in X$, $x = y$ and $x \xrightarrow{a} y$ where x and y are first-order variables, X is a second-order variable and a is an edge label. Complex formulas are built as usual from atomic ones using Boolean connectives and quantifiers. Free and bound occurrences of variables in a formula are defined in the standard fashion. We write $\varphi(X_1, \ldots, X_n, y_1, \ldots, y_m)$ to denote that the formula φ has free variables in $\{X_1, \ldots, X_n, y_1, \ldots, y_m\}$. A *closed formula* has no free variables.

For a formula $\varphi(X_1, \ldots, X_n, y_1, \ldots, y_m)$, a graph G, vertices $u_1, \ldots, u_m$ of G and sets of vertices $U_1, \ldots, U_n$ of G, we write $G \models \varphi[U_1, \ldots, U_n, u_1, \ldots, u_m]$ to denote that G *satisfies* φ when the free variable X_i, $i \in [n]$, is interpreted as the set of vertices U_i and the free variable x_j, $j \in [m]$, is interpreted as the vertex u_j. For a closed formula φ, we simply write $G \models \varphi$. The *MSO-theory* of G is the set of closed formulas satisfied by G.

For example, the MSO formula

$$\varphi(x, y) = \forall X. \left[\left(\forall x \forall y (x \in X \land x \xrightarrow{a} y) \Rightarrow y \in X \right) \land x \in X \right] \Rightarrow y \in X$$

expresses that there exists a path from x to y whose edges are labelled a.

2.5. Graph transformations. In the remainder of the paper, we will use several transformations of edge-labelled graphs, namely the unfolding of a graph, MSO-interpretations and MSO-transductions.

The *unfolding* $\mathrm{Unf}(G, r)$ of a graph G from one of its vertices r is the tree whose vertices are the paths in G starting from r and with an edge labelled a between two such paths π and π' if π' extends π by exactly one edge labelled a.

An *MSO-interpretation* is a graph transformation specified using MSO-formulas. An MSO-interpretation is given by a tuple of MSO-formulas of the form $(\delta(x), (\varphi_b(x, y))_{b \in B})$, where B is a set of labels. This interpretation when applied to a graph G will produce a graph, denoted $\mathcal{I}(G)$, whose edges have labels in B. The set of vertices of $\mathcal{I}(G)$ is the subset of the vertices of G satisfying the formula $\delta(x)$. For each edge label $b \in B$, there is

an edge from u to v labelled by b in $\mathcal{I}(G)$ if and only if G satisfies $\varphi_b[u, v]$. More formally, the graph (V', E') obtained by applying $\mathcal{I}$ to a graph (V, E) is such that:

$$
\begin{aligned}
V' &= \{u \in V \mid G \models \delta[u]\} \\
E' &= \{(u, b, v) \in V' \times B \times V' \mid G \models \varphi_b[u, v]\}.
\end{aligned}
$$

MSO-interpretations cannot increase the number of vertices of a graph. To overcome this weakness the notion of a transduction was introduced by Courcelle in [25]. The idea is to perform an operation that will increase the number of vertices before applying the MSO-interpretation. Let $K = \{k_1, \ldots, k_m\}$ be a finite set of edge labels. A *K-copying operation* applied to a graph G adds, to every vertex of G, m outgoing arcs respectively labelled by $k_1, \ldots k_{m-1}$ and k_m all going to fresh vertices. An *MSO-transduction* is a K-copying operation followed by an MSO-interpretation.

2.6. **Algebraic objects and functors.** We start by introducing the types of recursion schemes studied in this paper.

Definition 2.8. Let Σ be a signature. A Σ-*recursion scheme*, or *recursion scheme over* Σ, is a sequence E of equations

$$
\begin{aligned}
F_1(v_1, \ldots, v_{k_1}) &= t_1 \\
&\vdots \quad\quad\quad\quad\quad\quad\quad\quad (2.1) \\
F_n(v_1, \ldots, v_{k_n}) &= t_n
\end{aligned}
$$

where each t_i is a term over the signature $\Sigma_\Phi = \Sigma \cup \Phi$ in the variables $v_1, \ldots, v_{k_i}$, and Φ contains the symbols F_i (sometimes called 'functor variables' or 'function variables") of rank k_i, $i \in [n]$. A Σ-recursion scheme is *regular* if $k_i = 0$, for each $i \in [n]$.

Suppose that C is a continuous categorical Σ-algebra. Define

$$
C^{r(\Phi)} = [C^{k_1} \to C] \times \cdots \times [C^{k_n} \to C].
$$

Then $C^{r(\Phi)}$ is a continuous categorical Σ-algebra, as noted in Section 2.1.

When each F_i, $i \in [n]$, is interpreted as a continuous functor $f_i : C^{k_i} \to C$, each term over the extended signature $\Sigma_\Phi = \Sigma \cup \Phi$ in the variables $v_1, \ldots, v_m$ induces a continuous functor $C^m \to C$ that we denote by $t^C(f_1, \ldots, f_n)$. In fact, t^C is a continuous functor

$$
t^C \;:\; C^{r(\Phi)} \to [C^m \to C].
$$

More precisely, we define t^C as follows. Let f_i, g_i denote continuous functors $C^{k_i} \to C$, $i \in [n]$, and let α_i be a natural transformation $f_i \to g_i$ for each $i \in [n]$. When t is the variable v_i, say, then $t^C(f_1, \ldots, f_n)$ is the ith projection functor $C^m \to C$, and $t^C(\alpha_1, \ldots, \alpha_n)$ is the identity natural transformation corresponding to this projection functor. Suppose now that t is of the form $\sigma(t_1, \ldots, t_k)$, where $\sigma \in \Sigma_k$ and $t_1, \ldots, t_k$ are terms. Then $t^C(f_1, \ldots, f_n) = \sigma^C \circ \langle h_1, \ldots, h_k \rangle$ and $t^C(\alpha_1, \ldots, \alpha_n) = \sigma^C \circ \langle \beta_1, \ldots, \beta_k \rangle$, where $h_j = t_j^C(f_1, \ldots, f_n)$ and $\beta_j = t_j^C(\alpha_1, \ldots, \alpha_n)$ for all $j \in [k]$. (Here, we use the same notation for a functor and the corresponding identity natural transformation.) Finally, when t is of the form $F_i(t_1, \ldots, t_{k_i})$, then $t^C = f_i \circ \langle h_1, \ldots, h_{k_i} \rangle$, and the corresponding natural transformation is $\alpha_i \circ \langle \beta_1, \ldots, \beta_{k_i} \rangle$, where the h_j and β_j, $j \in [k_i]$, are defined similarly as above.

Note that if each $\alpha_i : f_i \to f_i$ is an identity natural transformation (so that $f_i = g_i$, for all $i \in [n]$), then $t^C(\alpha_1, \ldots, \alpha_n)$ is the identity natural transformation $t^C(f_1, \ldots, f_n) \to t^C(f_1, \ldots, f_n)$.

In any continuous categorical Σ-algebra C, by target-tupling the functors t_i^C associated with the right-hand sides of the equations in a recursion scheme E as in (2.1), we obtain a continuous functor

$$E^C : C^{r(\Phi)} \;\; \to \;\; C^{r(\Phi)}.$$

Indeed, $t_i^C : C^{r(\Phi)} \to [C^{k_i} \to C]$, for $i \in [n]$, so that

$$E^C = \langle t_1^C, \ldots, t_n^C \rangle : C^{r(\Phi)} \;\; \to \;\; C^{r(\Phi)}.$$

Thus, E^C has an initial fixed point in $C^{r(\Phi)}$, unique up to natural isomorphism, that we denote by

$$|E^C| \;\; = \;\; (|E|_1^C, \ldots, |E|_n^C),$$

so that, in particular,

$$|E|_i^C \;\; = \;\; t_i^C(|E|_1^C, \ldots, |E|_n^C),$$

at least up to isomorphism, for each $i \in [n]$.

It is well known that the initial fixed point $|E|^C$ can be 'computed' in the following way. Let $g_0 = (g_{0,1}, \ldots, g_{0,n})$, where, for each $j \in [n]$, $g_{0,j} = 0^{[C^{k_j} \to C]}$ is the constant functor $C^{k_j} \to C$ determined by the object 0^C. Then, for each $i \geq 0$, define $g_{i+1} = (g_{i+1,1}, \ldots, g_{i+1,n})$, where $g_{i+1,j} = t_j^C(g_i)$ for all $j \in [n]$. Next, let $\phi_0 = (\phi_{0,1}, \ldots, \phi_{0,n})$, where $\phi_{i,j}$, $j \in [n]$, is the unique natural transformation $0^{[C^{k_j} \to C]} \to g_{1,j}$. Moreover, for each $i \geq 0$, define $\phi_{i+1} = (\phi_{i+1,1}, \ldots, \phi_{i+1,n})$ as the natural transformation $\phi_{i+1,j} = t_j^C(\phi_i)$. Then $|E|^C$ is the colimit of the ω-diagram $(\phi_i : g_i \to g_{i+1})_{i \geq 0}$ in the continuous functor category $C^{r(\Phi)}$.

Definition 2.9. Suppose that C is a continuous categorical Σ-algebra. We call a functor $f : C^m \to C$ Σ-*algebraic*, if there is a recursion scheme E such that f is isomorphic to $|E|_1^C$, the first component of the above-mentioned initial solution of E. When $m = 0$, we identify a Σ-algebraic functor with a Σ-*algebraic object*. Last, a Σ-*regular object* is an object isomorphic to the first component of the initial solution of a Σ-regular recursion scheme.

In particular, we get the notions of Γ-algebraic and Γ-regular trees, and Δ-algebraic and Δ-regular trees.

Remark 2.10. Suppose that C is the continuous categorical Σ-algebra $\mathsf{ST}(A)$, where Σ is either Γ or Δ, and consider a recursion scheme E of the form (2.1). If the continuous functors $g_1 : \mathsf{ST}(A)^{k_1} \to \mathsf{ST}(A)$, ..., $g_n : \mathsf{ST}(A)^{k_n} \to \mathsf{ST}(A)$ preserve injective synchronization tree morphisms (in each variable), then so does each $t_j(g) : \mathsf{ST}(A)^{r(\Phi)} \to \mathsf{ST}(A)^{[k_j \to k]}$, where $g = (g_1, \ldots, g_n)$. It follows by induction that the functors $g_{i,j}$ preserve injective synchronization tree morphisms. moreover, the components of the natural transformations $\phi_{i,j}$ are injective synchronization tree morphisms. Thus, when $k_1 = 0$, so that the recursion scheme defines a tree, each $\phi_{i,1}$ is an embedding of the tree $g_{i,1}$ into $g_{i+1,1}$. If we represent the trees $g_{i,1}$ so that these embeddings are inclusions, then the colimit $|E|_1^{\mathsf{ST}(A)}$ of the ω-diagram $(\phi_{i,1} : g_{i,1} \to g_{i+1,1})_{i \geq 0}$ becomes the union of the trees $g_{i,1}$, $i \geq 0$.

Remark 2.11. It is sometimes convenient to add to the signature Γ sums of arbitrary nonzero rank. For this, we consider the signature[2] $\tilde{\Gamma} = \{+^n \mid n \geq 1\} \cup A \cup \{0, 1\}$ where

---

[2]Although the signature is infinite, we will always only use a finite subset of it.
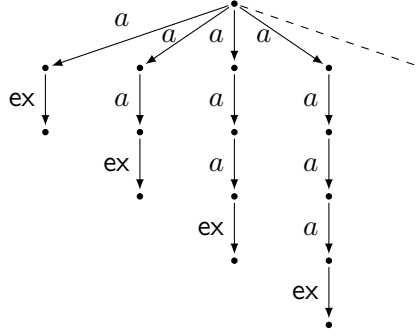
Figure 3: An infinitely branching synchronization tree

the rank of each symbol $+^n$ is n. The synchronization tree associated with a $\tilde{\Gamma}$-term tree is defined similarly as the synchronization tree associated to a Γ-term tree. The $\tilde{\Gamma}$-algebraic synchronization trees are Γ-algebraic.

Indeed we can transform an algebraic $\tilde{\Gamma}$-scheme into a Γ-scheme defining the same synchronization tree by replacing every occurrence of a subterm of the form $+^n(t_1, \ldots, t_n)$ on the right-hand side of an equation by $((t_1 + t_2) + t_3) \ldots + t_n$ for $n \geq 3$, every subterm of the form $t_1 +^2 t_2$ by $t_1 + t_2$, and finally every subterm of the form $+^1(t_1)$ by $t_1 + 0$.

Example 2.12. The following Δ-regular recursion scheme

$$X = (X \cdot a) + a \tag{2.2}$$

has the infinitely branching tree $\sum_{i \geq 1} a^i$ depicted on Figure 3 as its initial solution. That tree is therefore Δ-regular. Note that the tree $\sum_{i \geq 1} a^i$ is also Γ-algebraic because it is the initial solution of the following Γ-algebraic recursion scheme

$$
\begin{aligned}
F_1 &= F_2(a) \\
F_2(v) &= v + F_2(a.v).
\end{aligned}
$$

(Recall that we use a as an abbreviation of $a.1$.) The path language associated with the tree $\sum_{i \geq 1} a^i$ is $\{a^i \mid i \geq 1\}$. Note that the subtrees of that tree whose roots are children of the root of $\sum_{i \geq 1} a^i$ are pairwise inequivalent modulo language equivalence. So Γ-algebraic recursion schemes can be used to define infinitely branching trees that have an infinite number of subtrees, even up to bisimilarity [36, 39] or language equivalence.

Further examples of algebraic synchronization trees. Consider the labelled transition system (LTS) on Figure 4. The synchronization tree associated with that LTS is Γ-algebraic because it is defined by the recursion scheme below.

$$
\begin{aligned}
F_1 &= F_2(1) \\
F_2(v) &= v + a.F_2(b.F_2(v))
\end{aligned}
$$

The idea underlying the above definition is as follows. At any given vertex in the LTS on Figure 4, the argument v of F_2 denotes the subtree obtained by unfolding the LTS from that vertex *and not* taking the a-labelled edge as a first step. $F_2(v)$ denotes the tree obtained by unfolding the LTS at that vertex.
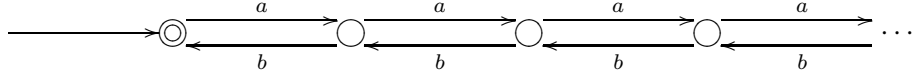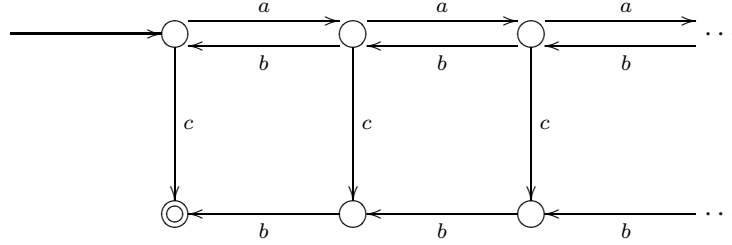
Figure 4: An LTS accepting the Dyck language

Figure 5: An LTS that cannot be expressed in BPA, but whose unfolding is an algebraic synchronization tree

This algebraic recursion scheme over Γ corresponds to the regular recursion scheme over Δ,

$$G \;=\; 1 + a \cdot G \cdot b \cdot G.$$

As another example, consider the LTS on Figure 5. This LTS is not expressible in BPA modulo modulo bisimilarity—see [37, page 206, Example (c)]. On the other hand, the synchronization tree associated with that LTS is Γ-algebraic because it is the unique solution of the Γ-algebraic recursion scheme below.

$$\begin{aligned} G &= F(1,1) \\ F(v_1, v_2) &= v_1 + c.v_2 + a.F(b.F(v_1, v_2), b.v_2) \end{aligned}$$

The idea underlying the above definition is as follows. At any given vertex in the LTS on Figure 5, the argument v_1 of F denotes the subtree obtained by unfolding the LTS from the vertex *and not* taking the a-labelled or c-labelled edges as a first step. The argument v_2 of F instead encodes the number of b-labelled edges that one must perform in a sequence in order to terminate. $F(v_1, v_2)$ denotes the tree obtained by unfolding the LTS from that vertex.

In our arguments, we will often make use of the following Mezei-Wright theorem [10]:

Theorem 2.13. *Suppose that $h : C \to D$ is a morphism of continuous categorical Σ-algebras. If c is an algebraic (or regular) object of C, then $h(c)$ is an algebraic (or regular) object of D. Moreover, for every algebraic object d of D there is an algebraic object c of C such that d is isomorphic to $h(c)$, and similarly for regular objects.*

2.7. **Continuous ordered algebras.** An important subclass of continuous categorical algebras is formed by the continuous *ordered* algebras, which constitute the classical framework for algebraic semantics [24, 26, 30, 31, 38, 41]. We say that a continuous categorical Σ-algebra C is ordered if its underlying category is a poset category, so that there is at most one morphism between any two objects. The objects of a continuous ordered algebra are usually called its elements and, as usual, the categorical structure is replaced by a partial

order $\leq$. The assumptions that C has initial object and colimits of ω-diagrams correspond to the requirements that there is a least element $\bot$ and each ω-chain $(a_n)_n$ has a supremum, denoted $\sup_n a_n$. Thus, C is an ω-*complete poset* as is each finite power C^n of C, equipped with the pointwise order. A continuous functor $C^n \to C$ is simply a function that preserves the suprema of ω-chains. Every such continuous function preserves the partial order. Continuous functions are ordered pointwise. As is well known, if C and D are ω-complete posets, then so is the poset $[C \to D]$ of all continuous functions $C \to D$.

Suppose that A is an alphabet and recall the definition of the signatures Γ and Δ. An example of a continuous ordered Δ-algebra is the language semiring $P(A^*)$ of all subsets of A^*, ordered by set inclusion, where each letter $a \in A$ denotes the set $\{a\}$ and 0 and 1 are interpreted as the empty set and the set $\{\epsilon\}$ containing only the empty word ϵ, and where $L_1 + L_2 = L_1 \cup L_2$ and $L_1 \cdot L_2 = L_1 L_2$ is the concatenation of L_1 and L_2, for all $L_1, L_2 \subseteq A^*$. Alternatively, we may view $P(A^*)$ as a continuous ordered Γ-algebra with $a(L) = \{a\} \cdot L$ for all $a \in A$ and $L \subseteq A^*$.

The initial continuous Σ-algebra may be described as the algebra of Σ-*term trees*. A term tree may be represented as (the isomorphism class of) a possibly infinite directed ordered tree whose vertices are labelled with the letters in Σ such that a vertex labelled in Σ_n has exactly n successors. In particular, vertices labelled in Σ_0 are leaves. Finite term trees over Σ may be identified with variable-free Σ-terms. As usual, we identify each symbol in Σ_0 with a term.

Let T_Σ^ω denote the set of all Σ-term trees. We equip T_Σ^ω with a partial order defined by $t \leq t'$ iff t' can be constructed from t by replacing some leaves labelled $\bot$ by some term trees in T_Σ^ω. It is well-known (see e.g. [24, 30, 31]) that this relation turns T_Σ^ω into an ω-complete partial order with least element $\bot$. We may further turn T_Σ^ω into a continuous ordered Σ-algebra. For each $\sigma \in \Sigma_n$ and term trees $t_1, \ldots, t_n$, the tree $\sigma(t_1, \ldots, t_n)$ is defined as usual as the tree whose root is labelled σ which has n subtrees isomorphic in order to $t_1, \ldots, t_n$. The following fact is known, see e.g. [10].

Proposition 2.14. *For every continuous categorical Σ-algebra C there is, up to natural isomorphism, a unique continuous categorical Σ-algebra morphism $T_\Sigma^\omega \to C$.*

Sometimes it is convenient to represent a term tree in T_Σ^ω as a partial function $t : \mathbb{N}^* \to \Sigma$ whose domain is a prefix closed nonempty subset of $\mathbb{N}^*$, where $\mathbb{N}$ denotes the set of positive integers, such that whenever $t(u) \in \Sigma_n$, then for each $i \in \mathbb{N}$, $t(ui)$ is defined if and only if $i \in [n]$. The operations are defined in the usual way. For trees $t, t' \in T_\Sigma^\omega$, we have $t \leq t'$ if and only if for all $u \in \mathbb{N}^*$, either $t(u) = t'(u)$ or $t(u) = \bot$. See [31, 38, 41] for details.

Yet another representation of a term tree in T_Σ^ω is by edge-labelled trees. To this end, let $\overline{\Sigma}$ denote the ordinary alphabet whose letters are the symbols σ_i where $\sigma \in \Sigma_n$, $n > 0$ and $i \in [n]$. When t is a Σ-term tree, each vertex u may be 'addressed" by a word $\overline{u} \in \overline{\Sigma}^*$ which encodes the unique path from the root to that vertex. The edge-labelled tree corresponding to t has as its vertex sets the addresses of its vertices, together with a vertex $\overline{u}\sigma$ whenever $\overline{u}$ is the address of a leaf labelled σ. The edges are given as follows. Suppose that u and v are vertices of t with associated addresses $\overline{u}$ and $\overline{v}$, respectively. If v is the ith successor of u and u is labelled $\sigma \in \Sigma_n$ (so that $n > 0$ and $i \in [n]$), then there is an edge from $\overline{u}$ to $\overline{v}$ labelled σ_i. Moreover, if u is a leaf vertex of t, labelled $\sigma \in \Sigma_0$, then there is an edge labelled σ from $\overline{u}$ to $\overline{u}\sigma$.

Consider, for example, the following term tree over Γ.
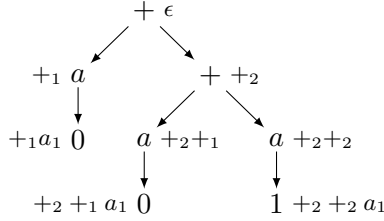
The resulting edge-labelled tree is
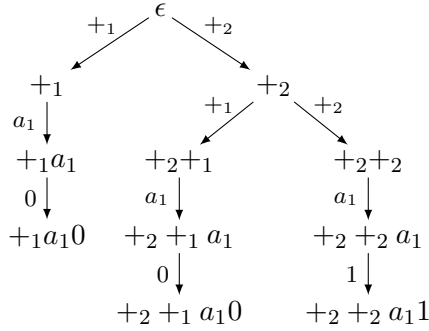
Remark 2.15. Suppose that C is a continuous categorical Σ-algebra and h is the essentially unique continuous categorical Σ-algebra morphism $T_\Sigma^\omega \to C$. Then, for each finite tree $t \in T_\Sigma^\omega$, $h(t)$ is uniquely determined up to isomorphism, since h preserves the operations. Moreover, when t, t' are finite with[3] $t \le t'$, $h(t \le t')$ is determined by the conditions that $h(\bot)$ is initial and h preserves the operations. When t is an infinite tree, t is the supremum of an ω-chain $(t_n)_n$ of finite trees—say t_n is the approximation of t obtained by removing all vertices of t at distance greater than or equal to n from the root (and relabeling vertices at distance n by $\bot$). Then $h(t)$ is the colimit of the ω-diagram $(\phi_n : h(t_n) \to h(t_{n+1}))_n$, where $\phi_n = h(t_n \le t_{n+1})$. When $\Sigma = \Gamma$ or $\Sigma = \Delta$, and $C = \mathsf{ST}(A)$, the morphisms ϕ_n are injective, and thus the colimit $h(t)$ is the 'union' of the $h(t_n)$.

2.8. **Morphisms.** By Proposition 2.14, there is an (essentially) unique morphism of continuous categorical Γ-algebras $T_\Gamma^\omega \to \mathsf{ST}(A)$, as well as an essentially unique continuous categorical Δ-algebra morphism $T_\Delta^\omega \to \mathsf{ST}(A)$. In the first part of this section, we provide a combinatorial description of (the object part) of these morphisms. In the second part of the section, we will consider morphisms from $\mathsf{ST}(A)$ to $P(A^*)$, seen as a Γ-algebra or a Δ-algebra.

We start by describing the (object part of the) essentially unique continuous categorical Δ-algebra morphism $T_\Delta^\omega \to \mathsf{ST}(A)$. We will call the image t' of a term tree t under this morphism the *synchronization tree denoted by* t, or the *synchronization tree associated with* t.

Suppose that t is a Δ-term tree and u and v are leaves of t, labelled in $A \cup \{1\}$. Let p (resp. q) denote the sequence of vertices along the unique path from the root to u (resp. v), including u (resp. v). We say that $v \in S_t(u)$ if p and q are of the form $p = rww_1p'$ and $q = rww_2q'$ with w labelled by $\cdot$ and successors w_1, w_2, ordered as indicated, such that

- every vertex appearing in p' which is different from u is either labelled $+$, or if it is labelled $\cdot$ then its second successor belongs to p', and

----
[3]Here, we denote by $t \le t'$ also the unique morphism $t \to t'$ in T_Σ^ω seen as a category.
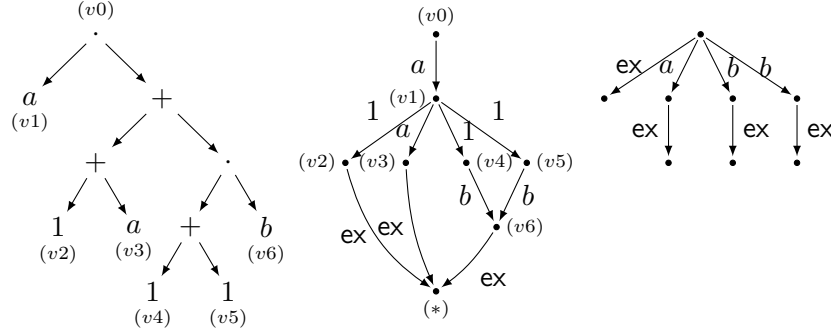
Figure 6: The term tree $a \cdot ((1 + a) + ((1 + 1) \cdot b))$ (on the left), the associated graph $G(t)$ (in the middle) and the synchronization tree $\tau(t)$ (on the right).

- every vertex appearing in q' which is different from v is either labelled $+$, or if it is labelled $\cdot$ then its first successor belongs to q'.

We say that $u \in M(t)$ if each vertex appearing in p which is different from u is either labelled $+$, or if it is labelled $\cdot$, then its first successor belongs to p. Finally, we say that $u \in E(t)$ if each vertex appearing in p which is different from u is either labelled $+$, or if it is labelled $\cdot$, then its second successor appears in p. Note that if $u \in M(t)$ then there is no w such that $u \in S_t(w)$, and similarly, if $u \in E(t)$ then $S_t(u) = \emptyset$.

Now form the edge-labelled directed graph $G(t)$ whose vertices are a new *root vertex* v_0, the leaves of t labelled in $A \cup \{1\}$, and the *exit vertex* denoted $*$. The edges of $G(t)$ are the following:

- For each $u \in M(t)$, there is an edge from the root v_0 to u, labelled by the label of u in t.
- For all leaf vertices u and v of t labelled in $A \cup \{1\}$ such that $v \in S(u)$, there is an edge in $G(t)$ from u to v whose label is the same as the label of v in t.
- Whenever u belongs to $E(t)$, there is an edge in $G(t)$ labelled ex from u to $*$.

Note that $G(t)$ does not contain any cycle. We denote by $\tau(t)$ the synchronization tree obtained by unfolding $G(t)$ from v_0 and then contracting edges labelled by 1.

In Proposition 2.16 below, we will prove that $\tau(t) = t'$, the synchronization tree denoted by t. However, before doing so, we find it instructive to give several examples.

As a first example, consider the term

$$t = a \cdot ((1 + a) + ((1 + 1) \cdot b)) \ .$$

As seen in Figure 6, the graph $G(t)$ contains 8 vertices, the root v_0, the exit vertex $*$, and vertices $v_1, \ldots, v_6$ corresponding to the 6 leaves of t. There is an a-labelled edge from v_0 to v_1, edges from v_1 to v_2, v_4 and v_5 labelled 1, an edge from v_1 to v_3 labelled a, and edges from v_4 and v_5 to v_6 labelled b. Finally, there is an edge labelled ex from each of v_2, v_3 and v_6 to $*$. It is clear that $\tau(t)$ is the synchronization tree denoted by t.

In the second example illustrated in Figure 7, consider the term tree t defined by the scheme

$$\begin{aligned} G &= (a \cdot H) \cdot c + d \\ H &= H \cdot 1 \ . \end{aligned}$$
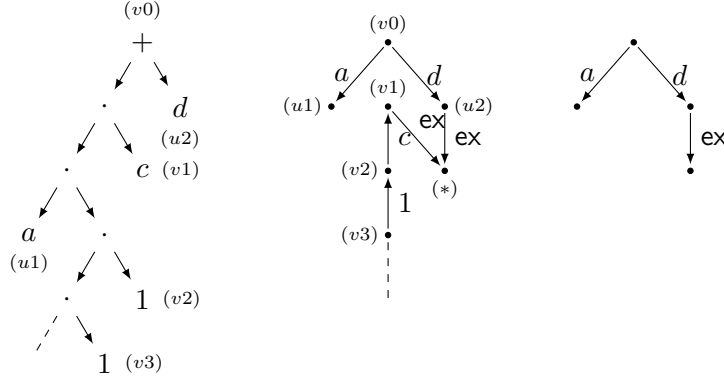
Figure 7: A term tree (on the left) and the associated graph $G(t)$ (on the right).
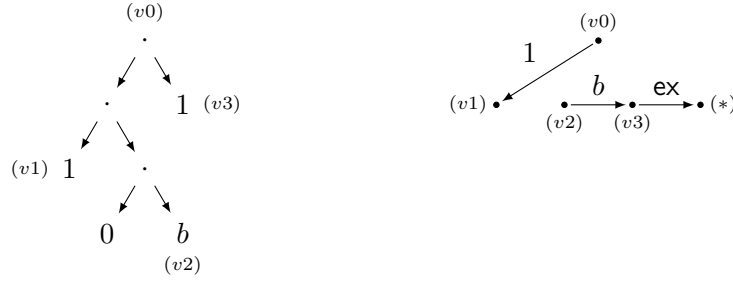
Figure 8: The term tree of $t = (1 \cdot (0 \cdot b)) \cdot 1$ (on the left) and the associated graph $G(t)$ (on the right).

Now the root v_0 of $G(t)$ has two outgoing edges leading to different vertices u_1 and u_2, say. These edges are labelled a and d, respectively. In addition, there is a sequence of vertices, call them $v_1, v_2, \ldots$, such that for each $i \geq 2$ there is an edge from v_{i+1} to v_i, which is labelled 1, and there is a c-labelled edge from v_2 to v_1. Last, $*$ is a vertex, and there exist edges from u_2 and v_1 to $*$ labelled ex. $G(t)$ is infinite, but the tree $\tau(t)$ constructed from it is finite since $\tau(t) = (a \cdot 0) + d$ is the synchronization tree denoted by t.

In the last example illustrated in Figure 8, consider the term $t = (1 \cdot (0 \cdot b)) \cdot 1$. The vertices of $G(t)$ are the root v_0, vertices v_1, v_2, v_3 corresponding respectively to the leaves of t with nonzero label, and the exit vertex $*$. There are 3 edges, an edge labelled 1 from v_0 to v_1, an edge labelled 1 from v_2 to v_3, and an edge labelled ex from v_3 to $*$. Now $\tau(t)$ contains only the root and no edges, so that $\tau(t)$ is the synchronization tree 0 denoted by t.

We still need to prove that the definition of $\tau(t)$ is correct.

Proposition 2.16. *For every Δ-term tree t, the image t' of t with respect to the essentially unique continuous Δ-algebra morphism $T_\Delta^\omega \to \mathsf{ST}(A)$ is $\tau(t)$.*

Proof. Suppose first that t is finite. We will prove the claim by induction on the structure of t. If $t = 0$, then $G(t)$ has two vertices, the root and the exit vertex, and no edges. If t is a symbol in $A \cup \{1\}$, then $G(t)$ has three vertices, the root v_0, a vertex v_1 and the vertex $*$. There is an edge from v_0 to v_1 and an edge from v_1 to $*$. The first edge is labelled a if $t = a \in A$, and 1 if $t = 1$. The second edge is labelled ex. In either case, $\tau(t) = t'$.

In the induction step, first suppose that $t = t_1 + t_2$ for some terms t_1, t_2 denoting the synchronization trees t_1' and t_2', respectively. Then $G(t)$ is isomorphic to the disjoint union of $G(t_1)$ and $G(t_2)$ with roots and exit vertices merged. Thus, $\tau(t) = \tau(t_1) + \tau(t_2) = t_1' + t_2' = t'$.

Suppose next that $t = t_1 \cdot t_2$ with t_1 denoting t_1' and t_2 denoting t_2'. Then $G(t)$ can be constructed from $G(t_1)$ and $G(t_2)$ as follows. For all edges from the root v_2 of $G(t_2)$ to a vertex v of $G(t_2)$ (which necessarily corresponds to a leaf vertex of t_2 in $M(t_2)$), and for all $u \in E(t_1)$, add a new edge from u to v labelled by the symbol which is the label of the edge from v_2 to v in $G(t_2)$ (i.e., the label of v in t_2). Then remove the edge from u to the exit vertex of $G(t_1)$. Finally remove all edges originating in the root of $G(t_2)$. The vertices of $G(t)$ are the non-exit vertices of $G(t_1)$ and the non-root vertices of $G(t_2)$. It should be clear that $\tau(t)$ is the sequential product of $\tau(t_1) \cdot \tau(t_2)$ and thus $\tau(t) = t_1' \cdot t_2' = t'$ by the induction hypothesis.

Suppose now that t is infinite. For each $n \geq 0$, let t_n denote the approximation of t obtained by relabelling each vertex of t of depth n by 0 and removing all vertices of depth greater than n. For each leaf vertex u of t there is some n_0 such that u is a vertex of t_n with the same label for all $n \geq n_0$. Moreover, for any two leaf vertices u, v of t with a nonzero label, $v \in S_t(u)$ iff there is some n_0 such that $v \in S_{t_n}(u)$ for all $n \geq n_0$. Similarly, for each leaf u of t with a nonzero label, we have $u \in M(t)$ ($u \in E(t)$) iff there is some n_0 such that $u \in M(t_n)$ ($u \in E(t_n)$, resp.) for all $n \geq n_0$. This implies that $G(t)$ is the union (colimit) of the $G(t_n)$ and then it follows that $\tau(t)$ is also the union (colimit) of the $\tau(t_n)$. We conclude that $t' = \mathsf{Colim}\ t_n' = \mathsf{Colim}\ \tau(t_n) = \tau(t)$, where for each n, t_n' is the synchronization tree denoted by t_n. $\qquad\square$

Next we describe the essentially unique morphism of continuous categorical Γ-algebras $T_\Gamma^\omega \to \mathsf{ST}(A)$. Fix a Γ-term tree $t \in T_\Gamma^\omega$. We define a synchronization tree, denoted $H(t)$, which is essentially a representation of t as a synchronization tree in which leaves labelled by 1 (in t) are added a dangling outgoing edge labelled by ex.

The vertices of the synchronization tree $H(t)$ are the vertices of t together with a fresh vertex u' for each leaf u labelled by 1. If u is labelled by $+$ in t then there is in $H(t)$ an edge labelled by $+_1$ from u to its first successor and an edge labelled by $+_2$ from u to its second successor. If u is labelled by $a \in A$ in t, then there is in $H(t)$ an edge labelled by a from u to its unique successor. Finally, for each leaf of u labelled by 1, there is an edge in $H(t)$ from u to u' labelled by ex. The synchronization tree defined by t is the $\{+_1, +_2\}$-contraction of $H(t)$. This construction is illustrated for the Γ-term $a.(c.0 + d.1) + 1$ in Figure 9.

Let τ' denote the function that maps t to the $\{+_1, +_2\}$-contraction of $H(t)$.

Proposition 2.17. *For every $t \in T_\Gamma^\omega$, $\tau'(t)$ is the synchronization tree in $\mathsf{ST}(A)$ denoted by t.*

We omit the proof which is essentially a simplification of that of Proposition 2.16.

Proposition 2.18. *The function which maps a synchronization tree $t \in \mathsf{ST}(A)$ to its path language in $P(A^*)$ is the (object part) of a categorical Γ-algebra, as well as Δ-algebra, morphism $\mathsf{ST}(A) \to P(A^*)$.*

Proof. It is clear that the empty tree is mapped to the empty language and the operations are preserved. When there is a morphism $t \to t'$ for synchronization trees $t, t' \in \mathsf{ST}(A)$, then the path language of t is included in the path language of t'. Finally, suppose that $(\phi_n : t_n \to t_{n+1})_n$ is an ω-diagram in $\mathsf{ST}(A)$ with colimit $(\psi_n : t_n \to t)_n$. Then for every branch of t ending in an edge labelled ex there is some n_0 such that the branch is the image
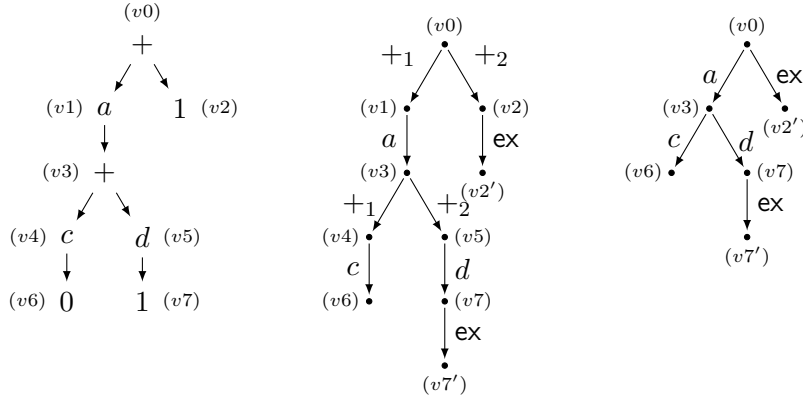
Figure 9: The term tree of $t = a.(c.0 + d.1) + 1$ (on the left) and the synchronization tree $H(t)$ (in the middle) and its $\{+_1, +_2\}$-contraction (on the right).

of a corresponding branch of t_{n_0} with respect to the morphism $\psi_{n_0} : t_{n_0} \to t$, and then the same holds for each $n \geq n_0$. Using this fact, it follows easily that the path language of t is the union of the path languages of the t_n, proving that colimits of ω-diagrams are preserved. $\square$

2.9. **Basic Properties.** In this section, we give some basic properties of synchronization trees defined by first-order recursion schemes.

First, we remark that Γ-regular and Γ-algebraic functors are closed under the sequential product. This closure property is immediate for Δ-regular and Δ-algebraic functors as the sequential product is an operation of the continuous Δ-algebra $\mathsf{ST}(A)$ (which is not the case for the continuous Γ-algebra $\mathsf{ST}(A)$).

Proposition 2.19. *If $f, f' : \mathsf{ST}(A)^k \to \mathsf{ST}(A)$ are Γ-algebraic (resp. Γ-regular), then so is $f \cdot f'$.*

Proof. Suppose that f and f' are the first components of the initial solutions over $\mathsf{ST}(A)$ of the Γ-recursion schemes E and E'. Without loss of generality, we may assume that E and E' have disjoint sets of functor variables. Let $F_1(x_1, \ldots, x_k)$ and $F_1'(x_1, \ldots, x_k)$ denote the left-hand sides of the first equations of E and E'. Then replace each occurrence of the symbol 1 on the right-hand-side term of each equation of E with $F_1'(x_1, \ldots, x_k)$, and consider the recursion scheme consisting of these equations together with the equations of E'. The component of the initial solution of this scheme which corresponds to F_1 is $f \cdot f'$. $\square$

Example 2.20. The tree b^ω determined by single infinite branch with edges labelled b is Γ-regular since it is the initial solution of the following Γ-regular recursion scheme

$$X = b.X.$$

We have already seen in Example 2.12 on page 13 that the tree $\sum_{i \geq 1} a^i$ is Γ-algebraic. According to the above proposition, the tree $(\sum_{i \geq 1} a^i) \cdot b^\omega$ is also Γ-algebraic. Indeed, it is

the initial solution of the recursion scheme

$$\begin{aligned} F_1 &= F_2(a.X) \\ F_2(v) &= v + F_2(a.v) \\ X &= b.X. \end{aligned}$$

All our classes of synchronization trees are also closed under the contraction operation for each non-empty subset of A.

Proposition 2.21. *The classes of Γ-regular, Γ-algebraic, Δ-regular and Δ-algebraic synchronization trees in $\mathsf{ST}(A)$ are closed under contraction for any non-empty subset of A.*

Proof. Let B be a non-empty subset of A. Let t be a synchronization tree defined by an algebraic Γ-scheme E and let t' be the synchronization tree obtained by contracting t with respect to B.

A Γ-scheme E' defining t' is easily obtained from E by replacing each equation of the form $F(v_1 \dots v_n) = t$ by the equation $F(v_1 \dots v_n) = \overline{t}$ where $\overline{t}$ is inductively defined by $\overline{0} = 0$, $\overline{1} = 1$, $\overline{t_1 + t_2} = \overline{t_1} + \overline{t_2}$, $\overline{G(t_1, \dots, t_k)} = G(\overline{t_1}, \dots, \overline{t_k})$, $\overline{a(t)} = a(\overline{t})$ if $a \in A \setminus B$ and $\overline{b(t)} = \overline{t}$ for $b \in B$. Note that E' is Γ-regular, if so is E.

The Γ-term defined by E' is the contraction of the Γ-term defined by E from its root with respect to B. It then follows, from the definition of the mapping τ' in 2.8 and Proposition 2.17, that the synchronization tree defined by E' is the contraction of the synchronization tree defined by E with respect to B.

Let t be a synchronization tree defined by an algebraic Δ-scheme E and let t' be the synchronization tree obtained by contracting t with respect to B. Consider the Δ-scheme E' obtained by replacing in E all occurrences of a constant in B by the constant 1. (Note that E' is Δ-regular, if so is E.) The Δ-term defined by E' is the Δ-term defined by E in which each occurrence of a constant in B is replaced by the constant 1. It follows from the definition of the mapping τ in 2.8 and Proposition 2.16, that the synchronization tree defined by E' is the contraction of the synchronization tree defined by E with respect to B. $\square$

Finally, the Γ-regular synchronizations trees can be characterized by a syntactic subfamily of the Δ-regular recursion schemes. This characterization is similar in spirit to the characterization of regular languages by right-linear context-free grammars. A Δ-regular scheme is said to be *right-linear* if the right-hand side of each equation is of the form

$$t_0 + \sum_{i \in [n]} t_i \cdot G_i$$

up to commutativity and associativity of sum, where the G_i, $i \in [n]$, are (constant) functor variables and the t_j, $j \in \{0, \dots, n\}$ are terms over the signature Δ not containing variables. (The empty sum stands for 0.) It is rather standard to prove the following fact:

Proposition 2.22. *A synchronization tree in $\mathsf{ST}(A)$ is Γ-regular iff it can be defined by a right-linear Δ-regular scheme.*

Proof. It is clear how to transform a Γ-regular recursion scheme into a right-linear Δ-scheme by turning each prefixing operation into a sequential product.

We show how to transform a Δ-term $t \cdot G$ with t containing no functor variables into a corresponding Γ-term $t'(G)$, possibly containing G. We proceed by induction on the

structure of t. When $t = 0$, let $t'(G) = 0$, and when $t = 1$ let $t'(G) = G$. When $t = a$ where $a \in A$, define $t'(G) = a.G$. Suppose now that $t = t_1 + t_2$. Then let $t' = t'_1(G) + t'_2(G)$. Finally, consider the case when $t = t_1 \cdot t_2$. In this case define $t'(G)$ as the term obtained by substituting $t'_2(G)$ for each occurrence of G in $t'_1(G)$.

A routine calculation shows that for each evaluation of G in $\mathsf{ST}(A)$, the terms $t \cdot G$ and $t'(G)$ yield the same synchronization tree. For an example of such calculation, we refer the reader to the proof of Theorem 3.2 on page 23. Using this fact, we may transform a right-linear Δ-scheme into a regular Γ-scheme by changing the right-hand side $t_0 + \sum_{i \in [n]} t_i \cdot G_i$ of each equation to $t'_0(1) + \sum_{i \in [n]} t'_i(G_i)$, where $t'_0(1)$ is the term obtained by substituting 1 for G in $t'_0(G)$. $\qquad\square$

3. Comparison between the Γ-algebra and the Δ-algebra

In this section, we interpret recursion schemes over the continuous categorical algebra $\mathsf{ST}(A)$, viewed either as a Γ-algebra or a Δ-algebra. We compare the resulting classes of synchronization trees with respect to language equivalence, bisimulation equivalence and isomorphism equivalence.

First for language equivalence, we show in Section 3.1 that the following hierarchy holds.

$$\underbrace{\Gamma\text{-regular}}_{\text{regular languages}} \quad \subsetneqq \quad \underbrace{\Delta\text{-regular} \;=\; \Gamma\text{-algebraic}}_{\text{context-free languages}} \quad \subsetneqq \quad \underbrace{\Delta\text{-algebraic}}_{\text{indexed languages}} \tag{3.1}$$

Up to bisimulation or isomorphism, we show in Section 3.2 that the following hierarchy holds.

$$\Gamma\text{-regular} \subsetneqq \Delta\text{-regular} \subsetneqq \Gamma\text{-algebraic} \subsetneqq \Delta\text{-algebraic} \tag{3.2}$$

We conclude the section by a comparison with the classes of synchronization trees defined by BPA and BPP.

3.1. Comparison up to language equivalence.
As already mentioned in the introduction, the path languages of the different classes can be characterized as follows.

Proposition 3.1. *The following properties holds.*
  (1) *The path languages of the Γ-regular trees are the regular languages.*
  (2) *The path languages of the Δ-regular trees and of the Γ-algebraic trees are the context-free languages.*
  (3) *The path languages of the Δ-algebraic languages are the indexed languages.*

Proof. By the Mezei-Wright theorem, the path languages of the Γ-regular trees and the Δ-regular trees in $\mathsf{ST}(A)$ are just the Γ-regular and Δ-regular elements (objects) of $P(A^*)$, seen as a continuous Γ-algebra or Δ-algebra. By classic results (e.g. [33, Theorem 1.21, page 116]), these in turn are the regular and context-free languages Similarly, the indexed languages (or OI-macro languages) are exactly the Δ-algebraic elements of $P(A^*)$, cf. [29], i.e., the path languages of the Δ-algebraic trees by the Mezei-Wright theorem. By Theorem 3.2, every Δ-regular tree is Γ-algebraic. Thus, to complete the proof, it remains to show that the path language of a Γ-algebraic tree is context-free.

Suppose that L is the path language of a Γ-algebraic tree in $\mathsf{ST}(A)$, and let E denote a Γ-algebraic scheme defining it. Then consider the term tree $t \in T_\Gamma^\omega$ defined by E. By the

Mezei-Wright theorem, L is the image of t with respect to the unique continuous Γ-algebra morphism $h : T_\Gamma^\omega \to P(A^*)$.

Suppose that v is a leaf of t labelled 1, and consider the branch of t from the root to v. The label of this branch is a word w_v over the alphabet $A \cup \{+_1, +_2\}$. Let us consider the image $\pi(w_v)$ of w_v under the erasing morphism $\pi : (A \cup \{+_1, +_2\})^* \to A^*$ that removes the letters $+_1$ and $+_2$. Then $L = h(t) \subseteq A^*$ is the set of all such words $\pi(w_v)$ obtained by considering all leaves v of t labelled 1.

It follows from Courcelle's characterization of the algebraic term trees by deterministic context-free languages [24, Theorem 5.5.1,page 157] (see also [22, 23]) that the set of all words w_v, where v is a leaf of t labelled 1, is a deterministic context-free language. Since the image of a (deterministic) context-free language with respect to a homomorphism is context-free, we conclude that L is a context-free language. $\qquad\square$

3.2. **Comparison up to bisimulation and isomorphism.** The aim of this section is to establish the strict inclusions stated in Equation (3.2).

As noted in Remark 2.3 on page 8, a Γ-term can be transformed into an equivalent Δ-term by replacing for each letter $a \in A$, all occurrences of a subterm $a(t)$ by $a \cdot t$. In particular, every Γ-regular tree is Δ-regular and every Γ-algebraic tree is Δ-algebraic. This establishes the first and third inclusions of (3.2). These inclusions are strict up to bisimulation and up to isomorphism as they are already strict with respect to language equivalence (cf. (3.1)).

It only remains to establish the second strict inclusion of Equation (3.2). First we establish the inclusion up to isomorphism in Theorem 3.2. In Corollary 3.4, we characterize a syntactical subfamily of the Γ-algebraic schemes which captures exactly the Δ-regular schemes.

Theorem 3.2. *Every Δ-regular tree is Γ-algebraic.*

Proof. Consider a regular Δ-recursion scheme E,

$$F_1 \ = \ t_1$$
$$\vdots$$
$$F_n \ = \ t_n,$$

which defines the Δ-regular tree $s \in \mathsf{ST}(A)$.

Let $\Phi = \{F_1, \ldots, F_n\}$ and $\Psi = \{G_0, G_1, \ldots, G_n\}$, where each F_i is of rank 0, G_0 is of rank 0, and each G_i with $i \geq 1$ is of rank 1.

Given a variable-free $\Delta \cup \Phi$-term t, we define its translation t' to be a $\Gamma \cup (\Psi - \{G_0\})$-term in the variable v_1.

- If $t = F_i$, for some $i \in [n]$, then $t' = G_i(v_1)$.
- If $t = 0$ then $t' = 0$.
- If $t = 1$ then $t' = v_1$.
- If $t = a$ then $t' = a.v_1$.
- If $t = t_1 + t_2$ then $t' = t_1' + t_2'$.
- If $t = t_1 \cdot t_2$ then $t' = t_1'(t_2')$, the term obtained by substituting t_2' for each occurrence of v_1 in t_1'.

Note that

$$t^{\mathsf{ST}(A)} : \mathsf{ST}(A)^n \to \mathsf{ST}(A)$$

and

$$t'^{\mathsf{ST}(A)} : [\mathsf{ST}(A) \to \mathsf{ST}(A)]^n \to [\mathsf{ST}(A) \to \mathsf{ST}(A)]$$

The two functors are related.

For a tree $r \in \mathsf{ST}(A)$, let $\overline{r}$ denote the functor 'left composition with r', $r \cdot (-)$ in $[\mathsf{ST}(A) \to \mathsf{ST}(A)]$.

Claim 1. For each variable-free $\Delta \cup \Phi$-term t and its translation t', and for each sequence of trees $r_1, \ldots, r_n$ in $\mathsf{ST}(A)$, it holds that

$$\overline{t^{\mathsf{ST}(A)}(r_1, \ldots, r_n)} \quad = \quad t'^{\mathsf{ST}(A)}(\overline{r}_1, \ldots, \overline{r}_n).$$

Indeed, when $t = F_i$, for some $i \in [n]$, then t' is $G_i(v_1)$ and both sides are equal to the functor $\overline{r}_i$. If $t = 0$, then both sides are equal to the constant functor $\mathsf{ST}(A) \to \mathsf{ST}(A)$ determined by the tree $0^{\mathsf{ST}(A)}$, and when $t = 1$, both sides are equal to the identity functor $\mathsf{ST}(A) \to \mathsf{ST}(A)$. Indeed, seen $t^{\mathsf{ST}(A)}(r_1, \ldots, r_n) = 1^{\mathsf{ST}(A)}$, left composition with $t^{\mathsf{ST}(A)}(r_1, \ldots, r_n)$ is the identity functor as is $t'^{\mathsf{ST}(A)}(\overline{r}_1, \ldots, \overline{r}_n) = v_1^{\mathsf{ST}(A)}(\overline{r}_1, \ldots, \overline{r}_n)$. Suppose now that $t = a$ for some a. Then both sides are equal to the functor $\overline{a}$, left composition with a. Next let $t = t_1 + t_2$, and suppose that the claim holds for t_1 and t_2. Then

$$
\begin{aligned}
\overline{t^{\mathsf{ST}(A)}(r_1, \ldots, r_n)} \quad &= \quad \overline{t_1^{\mathsf{ST}(A)}(r_1, \ldots, r_n)} + \overline{t_2^{\mathsf{ST}(A)}(r_1, \ldots, r_n)} \\
&= \quad t_1'^{\mathsf{ST}(A)}(\overline{r}_1, \ldots, \overline{r}_n) + t_2'^{\mathsf{ST}(A)}(\overline{r}_1, \ldots, \overline{r}_n) \\
&= \quad t'^{\mathsf{ST}(A)}(\overline{r}_1, \ldots, \overline{r}_n).
\end{aligned}
$$

Last, suppose that $t = t_1 \cdot t_2$, and that the claim holds for both terms t_1 and t_2. Then

$$\overline{t^{\mathsf{ST}(A)}(r_1, \ldots, r_n)} = \overline{t_1^{\mathsf{ST}(A)}(r_1, \ldots, r_n)} \circ \overline{t_2^{\mathsf{ST}(A)}(r_1, \ldots, r_n)}$$

is the composition of the functors $\overline{t_1^{\mathsf{ST}(A)}(r_1, \ldots, r_n)}$ and $\overline{t_2^{\mathsf{ST}(A)}(r_1, \ldots, r_n)}$ (where the second functor is applied first), as is the functor

$$
\begin{aligned}
t'^{\mathsf{ST}(A)}(\overline{r}_1, \ldots, \overline{r}_n) \quad &= \quad t_1'^{\mathsf{ST}(A)}(\overline{r}_1, \ldots, \overline{r}_n) \circ t_2'^{\mathsf{ST}(A)}(\overline{r}_1, \ldots, \overline{r}_n) \\
&= \quad \overline{t_1^{\mathsf{ST}(A)}(r_1, \ldots, r_n)} \circ \overline{t_2^{\mathsf{ST}(A)}(r_1, \ldots, r_n)}.
\end{aligned}
$$

Claim 2. For each variable-free $\Delta \cup \Phi$-term t and its translation t', and for each sequence of trees $r_1, \ldots, r_n$ in $\mathsf{ST}(A)$, it holds that

$$t^{\mathsf{ST}(A)}(r_1, \ldots, r_n) \quad = \quad (t'^{\mathsf{ST}(A)}(\overline{r}_1, \ldots, \overline{r}_n))(1^{\mathsf{ST}(A)})$$

Indeed, by Claim 1, we have

$$t^{\mathsf{ST}(A)}(r_1, \ldots, r_n) = (\overline{t^{\mathsf{ST}(A)}(r_1, \ldots, r_n)})(1^{\mathsf{ST}(A)}) = (t'^{\mathsf{ST}(A)}(\overline{r}_1, \ldots, \overline{r}_n))(1^{\mathsf{ST}(A)}).$$

Let E' denote the Γ-algebraic scheme

$$
\begin{aligned}
G_0 \quad &= \quad G_1(1) \\
G_1(v_1) \quad &= \quad t_1' \\
&\vdots \\
G_n(v_1) \quad &= \quad t_n'
\end{aligned}
$$

We claim that E' is equivalent to E, i.e., E' also defines s. To this end, let G denote the recursion scheme consisting of the last n equations of E'. In order to compare the schemes E and G, define

$$
\begin{aligned}
s_0 &= (s_{0,1}, \ldots, s_{0,n}) = (0^{\mathsf{ST}(A)}, \ldots, 0^{\mathsf{ST}(A)}) \\
s_{i+1} &= (s_{i+1,1}, \ldots, s_{i+1,n}) = (t_1^{\mathsf{ST}(A)}(s_i), \ldots, t_n^{\mathsf{ST}(A)}(s_i)) \\
g_0 &= (g_{0,1}, \ldots, g_{0,n}) = (0^{[\mathsf{ST}(A) \to \mathsf{ST}(A)]}, \ldots, 0^{[\mathsf{ST}(A) \to \mathsf{ST}(A)]}) \\
g_{i+1} &= (g_{i+1,1}, \ldots, g_{i+1,n}) = (t_1'^{\mathsf{ST}(A)}(g_i), \ldots, t_n'^{\mathsf{ST}(A)}(g_i))
\end{aligned}
$$

For each i, define $\overline{s}_i = (\overline{s_{i,1}}, \ldots, \overline{s_{i,n}})$. We prove by induction on i that $g_i = \overline{s_i}$.

This is clear when $i = 0$, since for each $j \in [n]$, $g_{0,j} = 0^{[\mathsf{ST}(A) \to \mathsf{ST}(A)]} = \overline{0^{\mathsf{ST}(A)}} = \overline{s_{0,j}}$. To prove the induction step, suppose that we have established our claim for some $i \geq 0$. Then for all $j \in [n]$,

$$
\begin{aligned}
g_{i+1,j} &= t_j'^{\mathsf{ST}(A)}(g_i) \\
&= t_j'^{\mathsf{ST}(A)}(\overline{s_i}) \\
&= \overline{t_j^{\mathsf{ST}(A)}(s_i)}, \quad \text{by Claim 1,} \\
&= \overline{s_{i+1,j}}
\end{aligned}
$$

For each $j \in [n]$, let $\phi_{0,j}$ denote the unique morphism $0^{\mathsf{ST}(A)} \to s_{1,j}$. Then define

$$
\begin{aligned}
\phi_0 &= (\phi_{0,1}, \ldots, \phi_{0,n}) \\
\phi_{i+1} &= (\phi_{i+1,1}, \ldots, \phi_{i+1,n}) = (t_1^{\mathsf{ST}(A)}(\phi_i), \ldots, t_n^{\mathsf{ST}(A)}(\phi_i))
\end{aligned}
$$

Next, let $\psi_{0,j}$ denote the unique natural transformation $0^{[\mathsf{ST}(A) \to \mathsf{ST}(A)]} \to g_{1,j}$, for each $j \in [n]$. Define

$$
\begin{aligned}
\psi_0 &= (\psi_{0,1}, \ldots, \psi_{0,n}) \\
\psi_{i+1} &= (\psi_{i+1,1}, \ldots, \psi_{i+1,n}) = (t_1'^{[\mathsf{ST}(A) \to \mathsf{ST}(A)]}(\psi_i), \ldots, t_n'^{[\mathsf{ST}(A) \to \mathsf{ST}(A)]}(\psi_i))
\end{aligned}
$$

Thus, each $\psi_{i,j}$ is a natural transformation from $g_{i,j} = \overline{s_{i,j}}$ to $g_{i+1,j} = \overline{s_{i+1,j}}$, and each $\phi_{i,j}$ is a morphism $s_{i,j} \to s_{i+1,j}$. Define $\overline{\phi_{i,j}}$ to be the natural transformation $\overline{s_{i,j}} \to \overline{s_{i+1,j}}$ such that for any tree f, the corresponding component of $\overline{\phi_{i,j}}$ is $\phi_{i,j} \cdot f$. Let $\overline{\phi_i} = (\overline{\phi_{i,1}}, \ldots, \overline{\phi_{i,n}})$.

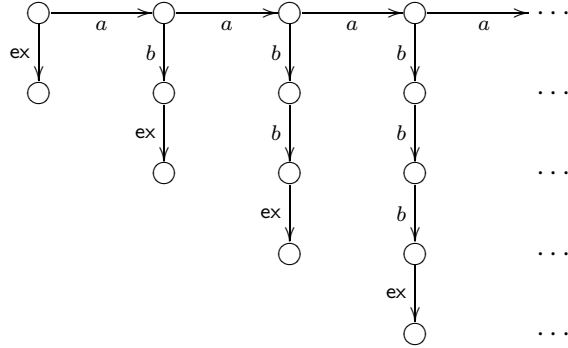*Claim 3.* For each i, it holds that $\psi_i = \overline{\phi_i}$.

The proof is similar to the above argument. Using this claim, it follows that $\phi_{i,j} = \psi_{i,j}(1^{\mathsf{ST}(A)})$ for each i, j, since

$$
\psi_{i,j}(1^{\mathsf{ST}(A)}) = \overline{\phi_{i,j}}(1^{\mathsf{ST}(A)}) = \phi_{i,j}.
$$

It is now easy to complete the proof. By the Bekić identity,

$$
\begin{aligned}
|E'^{\mathsf{ST}(A)}| &= |G^{\mathsf{ST}(A)}|(1^{\mathsf{ST}(A)}) \\
&= \mathsf{Colim}((\psi_{i,1}(1^{\mathsf{ST}(A)}) : g_{i,1}(1) \to g_{i+1,1}(1^{\mathsf{ST}(A)}))_{i \geq 0}) \\
&= \mathsf{Colim}((\phi_{i,1} : s_{i,1} \to s_{i+1,1})_{i \geq 0}) \\
&= |E|^{\mathsf{ST}(A)},
\end{aligned}
$$

up to isomorphism. $\qquad\square$

Figure 10: The Δ-regular and Γ-algebraic tree from Example 3.3

Example 3.3. Suppose that E is given by the single equation

$$F \quad = \quad 1 + a \cdot F \cdot b$$

Then E' is

$$G_0 \quad = \quad G(1)$$
$$G(v) \quad = \quad v + a.(G(b.v))$$

Both of them define the tree depicted on Figure 10.

The translation given in the proof of Theorem 3.2 allows us to characterize Δ-regular trees by a syntactic restriction on Γ-algebraic schemes.

Corollary 3.4. *A tree is Δ-regular if and only if it is defined by an algebraic recursion scheme G over Γ of the form*

$$G_0 \quad = \quad G_1(1)$$
$$G_1(v_1) \quad = \quad p_1$$
$$\vdots$$
$$G_n(v_1) \quad = \quad p_n$$

where G_0 has rank 0, each G_i with $i \geq 1$ has rank 1, and where none of the terms p_i has an occurrence of the constant 1.

Proof. In the light of the proof of Theorem 3.2, it is enough to show that any such scheme can be obtained from some regular recursion scheme E over Δ.

Let $\Psi = \{G_1, \ldots, G_n\}$ and $\Phi = \{F_1, \ldots, F_n\}$, where each F_i has rank 0. We give a transformation of a $\Gamma \cup \Psi$-term p in the variable v_1, which contains no occurrence of the constant 1, into a variable-free $\Delta \cup \Phi$-term $\widehat{p}$ such that $(\widehat{p})' = p$, where $(\widehat{p})'$ is defined in the proof of Theorem 3.2.

(1) If $p = 0$ then $\widehat{p} = 0$.
(2) If $p = v_1$ then $\widehat{p} = 1$.
(3) If $p = p_1 + p_2$ then $\widehat{p} = \widehat{p_1} + \widehat{p_2}$.
(4) If $p = a.p_1$ then $p' = a \cdot \widehat{p_1}$.
(5) If $p = G_i(p_1)$ then $\widehat{p} = F_i \cdot \widehat{p_1}$.

Claim. Let p be a $\Gamma \cup \Psi$-term in the variable v_1 containing no occurrence of the constant 1. Then $(\widehat{p})' = p$.

We prove this claim by induction on the structure of p. When $p = 0$, $\widehat{p} = 0$ and $(\widehat{p})' = 0$, and when $p = v_1$, $\widehat{p} = 1$ and $(\widehat{p})' = v_1$. Suppose that p is of the form $p_1 + p_2$ and that the claim holds for p_1 and p_2. In this case $\widehat{p} = \widehat{p_1} + \widehat{p_2}$ and $(\widehat{p})' = (\widehat{p_1})' + (\widehat{p_2})' = p_1 + p_2 = p$, by the induction hypothesis. Next let $p = a.p_1$ where $a \in A$ and suppose that the claim holds for p_1. Then $\widehat{p} = a \cdot \widehat{p_1}$ and $(\widehat{p})' = (a.v_1)((\widehat{p_1})') = a.(\widehat{p_1})' = a.p_1 = p$. Last, suppose that $p = G_i(p_1)$ for some $i \in [n]$ and p_1 satisfying the claim. Then we have $\widehat{p} = F_i \cdot \widehat{p_1}$ and $(\widehat{p})' = G_i((\widehat{p_1})') = G_i(p_1) = p$. This completes the proof of the claim.

Now consider the regular Δ-scheme E

$$F_1 \;\; = \;\; \widehat{p_1}$$
$$\vdots$$
$$F_n \;\; = \;\; \widehat{p_n}$$

By the proof of Theorem 3.2, G corresponds to E. Thus, E and G define the same tree. $\square$

Remark 3.5. The restriction on the use of the constant 1 in Corollary 3.4 is necessary. Indeed the proof of Proposition 3.6 to follow gives an example of Γ-algebraic scheme of rank 1 generating a tree that is not Δ-regular even up to bisimulation equivalence.

To establish Equation (3.2), it remains to show the strictness of the second inclusion up to bisimulation equivalence.

Proposition 3.6. *There exists a Γ-algebraic synchronization tree that is not bisimilar to any Δ-regular tree.*

Proof. Let $A = \{a, b\}$, and consider the synchronization tree $T \in \mathsf{ST}(A)$, defined by the Γ-algebraic recursion scheme:

$$S \;\; = \;\; F(1 + b.1)$$
$$F(v_1) \;\; = \;\; v_1 + a.(F(1 + b.v_1)) \; .$$

The tree T, depicted in Figure 11, has a single infinite branch whose edges are labelled a, and the out-degree of each vertex on this branch is 3, since each such vertex is the source of an edge labelled a, an edge labelled b, and an edge labelled ex. Since T is deterministic, its vertices may be identified with the words in the prefix closed language

$$\{a^n b^m, a^n b^m \mathsf{ex} \mid n \geq 0 \text{ and } m \leq n + 1\}.$$

A key feature is that every vertex is the source of an edge labelled ex.

We are going to show that every Δ-regular scheme E defining a synchronization tree bisimilar to T is equivalent to a right-linear one, modulo bisimilarity. By Proposition 2.22, this would imply that T is Γ-regular, which yields a contradiction. Indeed, the tree T has a countably infinite set of subtrees that are pairwise non-bisimilar.

Without loss of generality, we may assume that the equations of E are of one of three forms:

(1) $G = G_1 + G_2$,
(2) $G = G_1 \cdot G_2$, and
(3) $G = c$ for $c \in \{a, b\} \cup \{0, 1\}$.
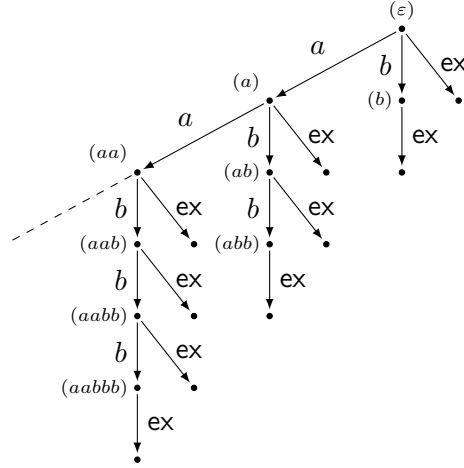
Figure 11: The tree T.

In the following, we are interested in a particular family $\mathcal{F}$ of 'subtrees" of T, containing, for all $k \geq 0$, the subtree T_k rooted at a^k, and the trees obtained from T_k by removing the exit edge originating in the root together with its target, or the edge labelled b originating in the root together with all vertices and edges accessible from the end vertex of that edge, or both. We denote these synchronization trees by $T_k(1), T_k(b)$ and $T_k(1, b)$, respectively.

Lemma 3.7. *Suppose that a tree* $s \in \mathcal{F}$ *is bisimilar to a tree* $s_1 \cdot s_2$, *where neither* s_1 *nor* s_2 *is bisimilar to 1. Then for some* k, $s = T_k(1, b)$, s_1 *is bisimilar to* $a = a.1$, *and* s_2 *is bisimilar to* T_{k+1}.

Proof. Suppose that s is bisimilar to $s_1 \cdot s_2$ and neither s_1 nor s_2 is bisimilar to the tree 1. Note that s_2 is not 0. Clearly, each vertex of s_1, except possibly the root, must be the source of an exit edge. Suppose that s contains and edge labelled a from x_1 to x_2 such both x_1 and x_2 are sources of an exit edge. Then in the tree $s_1 \cdot s_2$, they have successor vertices y_1 and y_2 such that the subtrees rooted at y_1 and y_2 are isomorphic (and thus bisimilar) and contain at least one edge. But T does not have such vertices connected by an edge labelled a and therefore neither does s. For this reason, s_1 cannot have two consecutive edges labelled a either. This in turn yields that s_2 has at least one edge labelled a and therefore s_1 cannot have an edge labelled b. We conclude that s_1 is bisimilar to the tree $a = a.1$ and then s_2 is bisimilar to T_{k+1} for some k. □

Now, by Lemma 3.7, we may transform E into a right-linear scheme defining T up to bisimilarity. First mark all those variables G such that the corresponding component in the initial solution of E over $\mathsf{ST}(A)$ has an infinite branch. The first variable is clearly marked. Suppose that G is marked. If the equation for G is $G = G_1 + G_2$, then G_1 or G_2 is marked. If one of them is not marked, then it can be replaced by a variable-free term. If the equation for G is $G = G_1 \cdot G_2$ and the component in the initial solution of E over $\mathsf{ST}(A)$ corresponding to one of the G_i is bisimilar to 1, then we may simply remove it. Otherwise we apply Lemma 3.7 and replace G_1 by a and mark G_2 if it is not yet marked. Eventually, we keep only the marked functor variables and obtain a right-linear scheme defining T up to bisimilarity. □

3.3. **Comparison with BPA and BPP.** The Δ-regular trees that can be defined using regular Δ-recursion schemes that do not contain occurrences of the constants 0 and 1 correspond to unfoldings of the labelled transition systems denoted by terms in Basic Process Algebra (BPA) with recursion, see, for instance, [3, 4, 6]. Indeed, the signature of BPA contains one constant symbol a for each action as well as the binary $+$ and $\cdot$ operation symbols, denoting nondeterministic choice and sequential composition, respectively. In the remainder of this paper, we write BPA for 'BPA with recursion'.

Alternatively, following [37], one may view BPA as the class of labelled transition systems associated with context-free grammars in Greibach normal form in which only leftmost derivations are permitted.

The class of Basic Parallel Processes (BPP) is a parallel counterpart of BPA introduced by Christensen [20]. BPP consists of the labelled transition systems associated with context-free grammars in Greibach normal form in which arbitrary derivations are allowed. We refer the interested readers to [37] for the details of the formal definitions, which are not needed to appreciate the results to follow, and further pointers to the literature.

Proposition 3.8.
  (1) *Every synchronization tree that is the unfolding of a BPA process is Γ-algebraic.*
  (2) *There is a Γ-algebraic synchronization tree that is neither definable in BPA modulo bisimilarity nor in BPP modulo language equivalence.*

Proof. The former claim follows easily from Theorem 3.2. In order to prove the latter statement, consider the LTS depicted on Figure 12. This LTS is not expressible in BPA modulo modulo bisimilarity and is not expressible in BPP modulo language equivalence (if the states q are r are the only final states in the LTS)—see [37, page 206, Example (f)]. On the other hand, the synchronization tree associated with that LTS is Γ-algebraic because it is the unique solution of the recursion scheme below.

$$\begin{aligned} F_1 &= b + c + a.F_2(b^2, c^2) \\ F_2(v_1, v_2) &= v_1 + v_2 + a.F_2(b.v_1, c.v_2) \end{aligned}$$

$\square$

So non-regular Γ-algebraic recursion schemes are more expressive than BPA modulo bisimilarity and can express synchronization trees that cannot be defined in BPP up to language equivalence, and therefore up to bisimilarity.

4. Comparison with the Caucal hierarchy

In this section, we compare the expressiveness of recursion schemes to that of the low classes in the Caucal hierarchy [18]. Section 4.1 gives a general overview of the Caucal hierarchy. Section 4.2 presents in more details the properties of the graphs and trees sitting in the first levels of the hierarchy. Section 4.3 shows that the classes of synchronization trees we introduced belong to the Caucal hierarchy. Finally Section 4.4 characterises the Γ-algebraic synchronization trees as contractions of the synchronization trees in the class Tree_2.
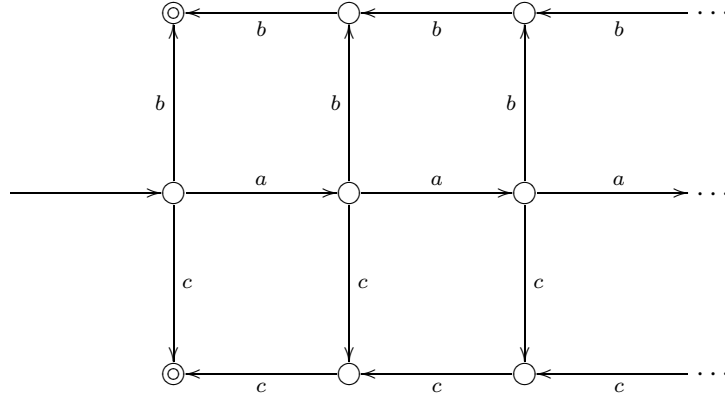
Figure 12: An LTS whose unfolding is an algebraic synchronization tree

4.1. **The Caucal hierarchy.** The Caucal hierarchy (also known as the pushdown hierarchy) is a hierarchy of classes of edge-labelled graphs. Following [15], the Caucal hierarchy is

$$\mathsf{Tree}_0 \subseteq \mathsf{Graph}_0 \subseteq \mathsf{Tree}_1 \subseteq \mathsf{Graph}_1 \subseteq \cdots$$

where Tree_0 and Graph_0 denote the classes of finite, edge-labelled trees and graphs, respectively. Moreover, for each $n \geq 0$, Tree_{n+1} stands for the class of trees isomorphic to unfoldings of graphs in Graph_n, and the graphs in Graph_{n+1} are those that can be obtained from the trees in Tree_{n+1} by applying a monadic interpretation (or transduction) [25]. As shown by Caucal in [19], every graph in the Caucal hierarchy has a decidable theory for monadic second order logic.

Remark 4.1. For a graph $G \in \mathsf{Graph}_n$ and a vertex v of G, the graph H obtained by restricting G to the set of vertices reachable from v is also a graph in Graph_n [15]. In particular, we can always assume that a tree in Tree_{n+1} is obtained by unfolding a graph in Graph_n from one of its root.

Example 4.2. Figure 13 shows a sequence of transformations constructing the synchronization tree $\sum_{i \geq 1} a^i$ starting from a finite graph. The vertices from which the graphs are unfolded are signalled by an incoming arrow. The MSO-interpretation $\mathcal{I}_1$ adds an a-labelled edge from a vertex u to a vertex v if u has no outgoing edges and there exist two vertices s and t with $s \xrightarrow{e} t$, $s \xrightarrow{e} v$, $t \xrightarrow{e} u$. The MSO-interpretation $\mathcal{I}_2$ is the $\{e\}$-contraction operation.

4.2. **First levels of the hierarchy.** The class Tree_1 contains the regular trees of finite outdegree (i.e., the trees of finite degree having finitely many non-isomorphic subtrees). It is well known that Graph_1 is the set of all prefix-recognizable graphs [19].

A *prefix recognizable relation* over a finite alphabet C is a finite union of relations of the form[4] $U \cdot (V \times W)$, for some nonempty regular languages $U, V, W \subseteq C^*$. A *prefix recognizable graph* over an alphabet B is an edge-labelled graph that is isomorphic to a graph of the form $(V, (\xrightarrow{b})_{b \in B})$, where for some alphabet C, V is a regular subset of C^* and the edge

---

[4] $U \cdot (V \times W)$ denotes the relation $\{(uv, uw) \mid u \in U, v \in V \text{ and } w \in W\}$.
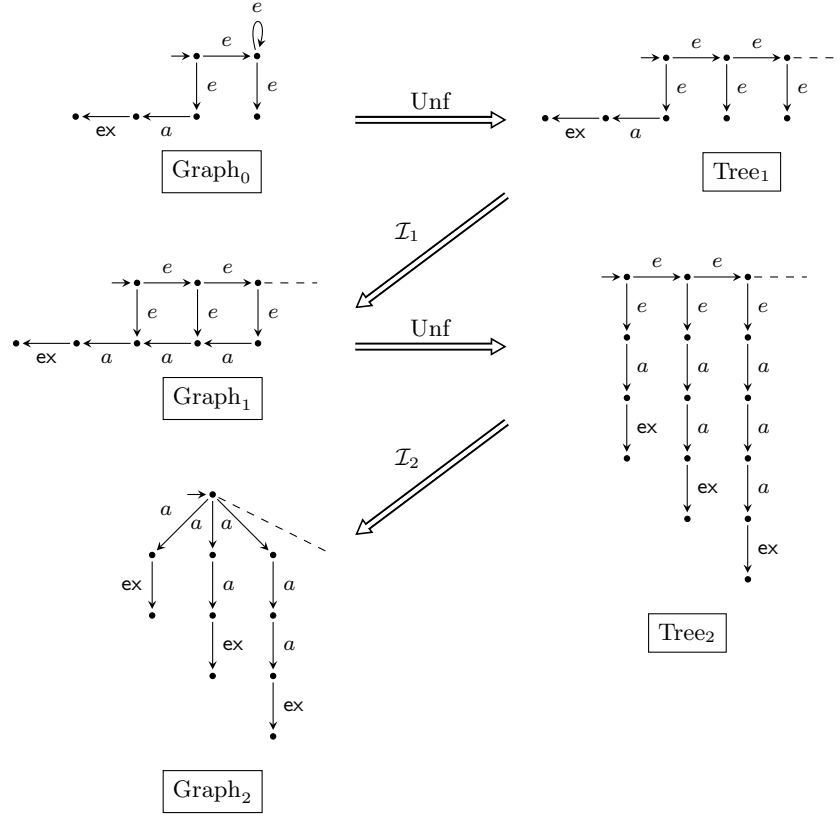
Figure 13: A construction of the synchronization tree $\sum_{i\geq 1} a^i$ in the Caucal hierarchy.

relations $\xrightarrow{b}$ are prefix recognizable relations over C. (Of course, the alphabet C may be fixed to be a 2-element alphabet.)

Proposition 4.3. *For a labelled graph G, the following statements are equivalent:*

- *G belongs to Graph_1,*
- *G is isomorphic to a prefix-recognizable graph* [18],
- *G can be MSO-interpreted in the full binary tree Δ_2* [12].

Proposition 4.4. *Let G be a graph in Graph_1 labelled in A, and let r be a root of G. The graph G is isomorphic to the B-contraction of a deterministic graph $H \in \mathsf{Graph}_1$ labelled in $A \uplus B$ from one of its root r'.*

Moreover H can be chosen such that for any $a \in A$ and any two vertices u and v belonging to the B-contraction from r', there exists at most one path from u to v labelled by a word in B^*a.

Proof. Let $G = (V, (R_a)_{a \in A})$ be a prefix-recognizable graph labelled in A. We may assume, without loss of generality, that the vertices in V are words over the alphabet $C = \{0, 1\}$. Moreover we may assume that the relations R_a, $a \in A$, can be expressed as the disjoint union of relations $U_{a,1} \cdot (V_{a,1} \times W_{a,1}), \ldots, U_{a,n_a} \cdot (V_{a,n_a} \times W_{a,n_a})$. In addition, we may require that $\mathrm{First}(V_{a,i}) \cap \mathrm{First}(W_{a,i}) \subseteq \{\varepsilon\}$, for all $i \in [n_a]$ where $\mathrm{First}(L) = \{a \in C \mid au \in L$ for some $u \in C^*\} \cup \{\varepsilon \mid \varepsilon \in L\}$ [16, Proposition 2.1]. These assumptions guarantee that if

(x, y) belongs to R_a, then there exists a unique $i \in [n_a]$ and a unique decomposition $x = uv$ and $y = uw$ such that $u \in U_{a,i}, v \in V_{a,i}$ and $w \in W_{a,i}$.

Before proceeding with the construction, we need to introduce some notations. Let $a \in A$ and let $i \in [n_a]$. We take $\mathcal{A}_{a,i} = (Q_{a,i}, q_{i,a}, F_{a,i}, \delta_{a,i})$ to be a complete DFA accepting the reverse of $V_{a,i}$ and $\mathcal{B}_{a,i} = (Q'_{a,i}, q'_{i,a}, F'_{a,i}, \delta'_{a,i})$ to be a complete DFA accepting $W_{a,i}$. We assume that the sets of states of these automata are pairwise disjoint and we take $Q = \bigcup_{a \in A, i \in [n_a]} Q_{a,i}$ and $Q' = \bigcup_{a \in A, i \in [n_a]} Q'_{a,i}$.

We are now going to define a prefix-recognizable graph $H = (V', (R'_a)_{a \in A \cup B})$ satisfying the properties stated above.

The set of labels B is $B = \{e_0, e_1, e_2\} \cup \{e_{a,i} \mid a \in A \text{ and } i \in [n_a]\}$. The vertices of H are words over the alphabet $\{0, 1\} \cup Q \cup Q' \cup \{(a, i) \mid a \in A \text{ and } i \in [n_a]\} \cup \{\star\}$.

Intuitively, the graph H is constructed in such a way that a path labelled by a word in B^*a from a vertex $x \in V$ to a vertex $y \in V$ simulates the relation $R_{a,i}$ for $a \in A$ and $i \in [n_a]$. For a fixed $a \in A$ and $i \in [n_a]$, the simulation is done using two sets of vertices $V_{a,i} = \{0, 1\}^* Q_{a,i}(a, i)$ and $V'_{a,i} = \{0, 1\}^* Q'_{a,i}(a, i)$. Starting from a vertex $x \in V$, the vertices in $V_{a,i}$ are used to remove a suffix v of x belonging to $V_{a,i}$ and the vertices in $V'_{a,i}$ are used to add a suffix w in $W_{a,i}$. When moving from the vertices in $V_{a,i}$ to the vertices in $V'_{a,i}$, we check that the remaining prefix u belongs to $U_{a,i}$. This guarantees that x and y can be respectively written as uv and uw with $u \in U_{a,i}, v \in V_{a,i}$ and $w \in W_{a,i}$.

Formally, from a vertex $x \in V$, there is an edge labelled $e_{a,i}$ to the vertex $xq_{a,i}(a, i)$ (cf. (1) below). For all words $u, v \in \{0, 1\}^*$ and for all $q \in Q_{a,i}$, $uvq(a, i) \xrightarrow[H]{e_0^{|v|}} u\delta_{a,i}(q, u)(a, i)$ (cf. (2) below). If $u \in \{0, 1\}^*$ belongs to $U_{a,i}$ and $q \in F_{a,i}$, $uq(a, i) \xrightarrow[H]{e_1} uq'_{a,i}(a, i)$ (cf. (3) below). For all words $u, v \in \{0, 1\}^*$ and for all $q \in Q'_{a,i}$, $uq(a, i) \xrightarrow[H]{e_0^*} uv\delta'_{a,i}(q, v)(a, i)$ (cf. (4) below). Finally, for $u \in \{0, 1\}^*$ and $q \in F'_{a,i}$, we have $uq(a, i) \xrightarrow[H]{e_2} u \star (a, i) \xrightarrow[H]{a} u$ (cf. (5) and (6) below).

The set of vertices V' is taken to be $V \cup \bigcup_{a \in A} \mathrm{Dom}(R'_a) \cup \mathrm{Im}(R'_a)$. The edges of H are defined, for all $a \in A$, $i \in [n_a]$, $q \in Q_{a,i}, q' \in Q'_{a,i}$, $u \in \{0, 1\}^*$ and $b \in \{0, 1\}^*$, by:

$$u \qquad \xrightarrow[H]{e_{a,i}} \qquad u\, q_{a,i}(a, i) \qquad\qquad \text{for } u \in V \qquad\qquad (1)$$

$$u\, bq(a, i) \qquad \xrightarrow[H]{e_0} \qquad u\, \delta_{a,i}(q, b) \qquad\qquad \text{for } b \in \{0, 1\} \qquad\qquad (2)$$

$$u\, q(a, i) \qquad \xrightarrow[H]{e_1} \qquad u\, q'_{a,i}(a, i) \qquad\qquad \text{if } q \in F_{i,a} \text{ and } u \in U \quad (3)$$

$$u\, q'(a, i) \qquad \xrightarrow[H]{e_0} \qquad ub\, \delta'_{a,i}(q, b)(a, i) \qquad \text{for } b \in \{0, 1\} \qquad\qquad (4)$$

$$u\, q'(a, i) \qquad \xrightarrow[H]{e_2} \qquad u \star (a, i) \qquad\qquad \text{if } q' \in F'_{a,i} \qquad\qquad (5)$$

$$u \star (a, i) \qquad \xrightarrow[H]{a} \qquad u \qquad\qquad\qquad \text{for } u \in V \qquad\qquad (6)$$

The graph H is a deterministic prefix-recognizable graph. We have already seen that:

Claim 4.5. *For all $x, y \in \{0, 1\}^*$, if $x \xrightarrow[G]{a} y$ then there exists a path in H labelled in B^*a from x to y.*
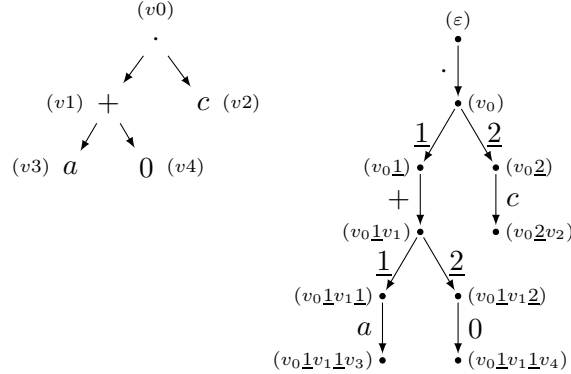
Figure 14: The representation of the Δ-term $(a + 0) \cdot c$ as a labelled tree following [18]

It remains to show the other direction.

Claim 4.6. *For all $x, y \in \{0,1\}^*$, if there exists a path from x to y in H labelled in $B^* a$ then this path is unique and $x \xrightarrow[G]{a} y$.*

Proof. Let $x, y \in \{0,1\}^*$ be such that there exists a path from x to y in H labelled in $B^* a$. By the construction of H, there exist $i \in [n_a]$, and u, v and $w \in \{0,1\}^*$ with $x = uv$ and $y = uv$ such that the path is of the form

$$x \xrightarrow[H]{e_{a,i}} xq_{a,i} \xrightarrow[H]{e_0^{|u|}} u\delta(q_{a,i}, u)(a, i) \xrightarrow{e_1}$$
$$uq'_{a,i}(a, i) \xrightarrow[H]{e_0^{|w|}} uw\delta'_{a,i}(q'_{a,i}, w) \xrightarrow[u]{e_2} w \star (a, i) \xrightarrow[H]{a} uw$$

with $\delta(q_{a,i}, u) \in F_{a,i}$, $\delta(q'_{a,i}, w) \in F'_{a,i}$ and $u \in U_{a,i}$. It follows that (x, y) belongs to $R_{a,i}$. As remarked before, the index i is unique and so is the decomposition into u, v and w. Hence the uniqueness of the path follows. □

From the above two claims, it follows that the B-contraction of H is equal to G. □

As first remarked in [42], the graphs in Graph_1 can be obtained by a form of contraction[5] of the configurations graph of pushdown automata [15].

Let us now consider the class Tree_2. The main property of the class Tree_2 is that it contains the Σ-term trees defined by any Σ-algebraic recursion scheme for any signature Σ.

As Tree_2 only contains labelled trees, we need to fix a representation of Σ-term trees as labelled trees. We follow the one from [18] which is slightly different from the one presented in Section 2.8 (see also Remark 4.8). A Σ-term tree t is represented by a tree t' labelled by $\Sigma \cup \{\underline{1}, \ldots, \underline{m}\}$ where m is the maximum rank of a symbol of Σ.

The vertices of t' are prefixes of sequences of the form $u_0 \underline{d_0} \cdots u_{n-1} \underline{d_{n-1}} u_n$ where u_0 is the root of t, u_n is a leaf of t and for all $i \in [n]$, u_i is the d_{i-1}-th successor of u_{i-1}. Let u be a vertex of t labelled by $a \in \Sigma$, if w and wu are vertices of t' then there is an edge from w to wu labelled by a. Let $i \in [m]$, if w and $w\underline{i}$ are vertices of t' then there is an edge from w to $w\underline{i}$ labelled by $\underline{i}$. This representation is illustrated in Figure 14 for the Δ-term $(a+0) \cdot c$.

---

[5]In [42], the notion of B-contraction keeps the vertices that are the source of an edge in A and adds an edge labelled by $a \in A$ between two such vertices whenever there is a path labelled in aB^*.
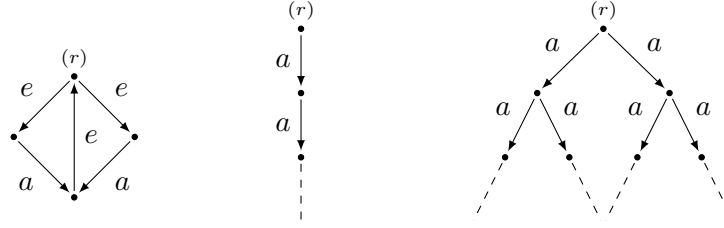
Figure 15: A graph G (on the left), the unfolding from r of its $\{e\}$-contraction from r (in the middle) and the $\{e\}$-contraction of its unfolding from r (on the right).

Theorem 4.7 ([17, 18])**.** *Let Σ be a signature. A Σ-term tree is defined by some Σ-algebraic recursion scheme if and only if the tree representing it belongs to Tree_2.*

Remark 4.8. It immediately follows that the synchronization tree $H(t)$ associated to a Γ-term tree t defined in Section 2.8 belongs to Tree_2 (resp. Tree_1) whenever t is defined by a Γ-algebraic (resp. Γ-regular) recursion scheme. Indeed the transformation going from the presentation of [18] to ours is an MSO-interpretation that commutes with the unfolding operation.

We conclude with some properties of the trees of Tree_2 and their contractions.

Proposition 4.9. *The following properties hold:*
  (1) The contraction of a tree in Tree_2 is bisimilar to some tree in Tree_2.
  (2) Every tree in Tree_2 can be obtained as the contraction of a deterministic tree in Tree_2.

Proof. As illustrated in Figure 15, the contraction operation does not in general commute with the unfolding operation. However, it is easy to show that the contraction of the unfolding from a root r and the unfolding of the contraction from the same root are bisimilar.

For the first property, let t be a tree in Tree_2 which is obtained by unfolding a graph G in Graph_1 from one of its roots r. Let t' be the B-contraction of t from its root. The B-contraction G' of G from r also belongs to Graph_1 (as the B-contraction is a particular case of MSO-interpretation). From the previous remark, we have that t' is bisimilar to $\mathrm{Unf}(G', r)$ which belongs to Tree_2 as G belongs to Graph_1.

For the second property, let t be a tree in Tree_2 which is obtained by unfolding a graph G in Graph_1 from one of its root r. Furthermore assume that both t and G are labelled by A. By Proposition 4.4, G can be obtained by B-contraction of a deterministic graph $H \in \mathsf{Graph}_1$ from one of its roots r.

For the B-contraction to commute with the unfolding, it is enough that the graph H satisfies, for any label $a \in A$ and any two vertices u and v of H belonging to the B-contraction, that there exists at most one path labelled in B^*a from u to v. This is the case for the graph H obtained from Proposition 4.4. Hence t is isomorphic to the B-contraction of $\mathrm{Unf}(H, r')$, which is a deterministic tree in Tree_2. $\qquad\square$

4.3. **Synchronization trees in the Caucal hierarchy.** The Γ-regular synchronization trees, being the regular trees with potentially infinite degree, belong to Graph_1. As Tree_1 only contains the regular trees of finite degree, it does not contain the class of Γ-regular synchronization trees. In this section, we show that the classes of Δ-regular and Γ-algebraic synchronization trees are included in Graph_2 but not in Tree_2. Similarly, we establish that the class of Δ-algebraic trees is included in Graph_3 but not in Tree_3.

Proposition 4.10. *The Γ-algebraic (and hence the Δ-regular) synchronization trees are contractions of trees in Tree_2 and hence belong to Graph_2.*

Proof. Let E be a Γ-algebraic recursion scheme defining a Γ-term tree t and a synchronization tree t'. From Proposition 2.17, t' is obtained by contraction of the synchronization tree $H(t)$ with respect to $\{+_1, +_2\}$. From [18, Theorem 3.5] (cf. Remark 4.8), we know that $H(t)$ belongs to Tree_2, and, as the contraction operation is a particular case of MSO-interpretation, t' belongs to Graph_2. $\square$

Proposition 4.11. *There is a Δ-regular synchronization tree which is not in Tree_2.*

Proof. Consider the following Δ-regular recursion scheme:
$$G \;=\; 1 + a \cdot G \cdot (1+1).$$
The synchronization tree t defined by it has a single infinite branch $u_0, u_1, \ldots$ (with edges labelled a). The out-degree of each vertex u_i is $2^i + 1$ since it is the source of 2^i edges labelled ex.

Assume, towards a contradiction, that t belongs to Tree_2. Hence t is isomorphic to the unfolding of some graph H in Graph_1 from a vertex v_0. By Remark 4.1, we can assume that all vertices are reachable from v_0 and hence that they all have finite out-degree. This implies that H contains a path $v_0, v_1 \ldots$ along which the out-degree of the vertices grows exponentially (i.e. the out-degree of v_i is $2^i + 1$). The following lemma shows that this is not possible in Graph_1.

Lemma 4.12. *Let G be a graph in Graph_1 whose vertices have finite out-degree. Let $(u_i)_{i\geq 0}$ be an infinite path in G, possibly with repetitions. Then there exists a constant $C \geq 0$ such that for all $i \geq 0$, the out-degree of u_i is at most $C(i+1)$.*

Proof. By Proposition 4.3, we can assume without loss of generality that G is a prefix-recognizable graph. Let $U_1(V_1 \times W_1), \ldots, U_n(V_n \times W_n)$ be the relations involved in the definition of G. As the vertices of G have finite out-degree, without loss of generality we may assume that all the W_j are finite sets. Let d denote the length of the longest word appearing in the W_j, and let W denote the number of words appearing in the W_j. Then clearly $|u_{i+1}| \leq |u_i| + d$ for all $i \geq 0$, where for a word u we denote its length by $|u|$. Thus, introducing the notation $d' = \max\{d, |u_0|\}$, we have $|u_i| \leq d'(i+1)$ for all $i \geq 0$. As the out-degree of a vertex u of G is at most $(|u| + 1)W$, we conclude that the out-degree of u_i is at most $C(i+1)$ for all $i \geq 0$ with $C = (d'+1)W$. $\square$

$\square$

Proposition 4.13. *All the Δ-algebraic synchronization trees are in Graph_3 and therefore have a decidable MSO-theory.*

Proof. Let E be an algebraic scheme over Δ. Let t be the Δ-term defined by E, and t' be the synchronization tree defined by E. By the Mezei-Wright theorem and Proposition 2.16,

we have that $t' = \tau(t)$. More precisely, the tree t' is obtained by unfolding the graph $G(t)$ (as defined in Section 2.8) from the vertex v_0 and then applying a contraction with respect to $\{1\}$. The graph $G(t)$ can be interpreted in the tree representing t, which belongs to Tree_2 (by Theorem 4.7), and hence $G(t)$ belongs to Graph_2. The unfolding of $G(t)$ from v_0 belongs to Tree_3 and its contraction t' to Graph_3 (as contractions are particular cases of MSO-interpretations). $\qquad\square$

Proposition 4.14. *There exists a Δ-algebraic synchronization tree which does not belong to Tree_3.*

Proof. In this proof, we consider $\{a, b\}$-labelled trees of a particular shape illustrated in Figure 16. These trees have a unique infinite branch whose edges are labelled a. In addition the vertex at depth n along this branch has $d(n)$ outgoing edges labelled b, for some mapping $d : \mathbb{N} \to \mathbb{N}$. Up to isomorphism, the tree is entirely characterized by the mapping d and is denoted t_d.

In [13, Theorem 4.5.3], Braud gives a necessary condition for a tree of this form[6] to belong to Graph_2. Namely, if a tree t_d, for $d : \mathbb{N} \to \mathbb{N}$ belongs to Graph_2 then there exists a constant $c > 0$ such that $d(n) < 2^{c(n+1)}$ for all $n \geq 0$.

Our proof goes as follows. We first show that for the mapping $d_0 : n \mapsto 2^{2^n}$, the tree t_{d_0}, depicted in Figure 16, is Δ-algebraic. Towards a contradiction, we assume that t_{d_0} belongs to Tree_3. We then show there would exist a mapping d_1 satisfying $d_1(n) \geq 2^{2^{n-1}}$, $n \geq 1$ such that t_{d_1} belongs to Graph_2. This statement contradicts Braud's condition as there cannot exist a constant c such that $2^{2^{n-1}} < 2^{c(n+1)}$, for all $n \geq 1$.

Consider the Δ-algebraic scheme

$$\begin{aligned} S &= G(1+1) \\ G(v) &= a \cdot G(v \cdot v) + v \cdot (b \cdot 0) \end{aligned}$$

The tree defined by this scheme is isomorphic to the tree t_{d_0} with $d_0(n) = 2^{2^n}$ for all $n \geq 0$.
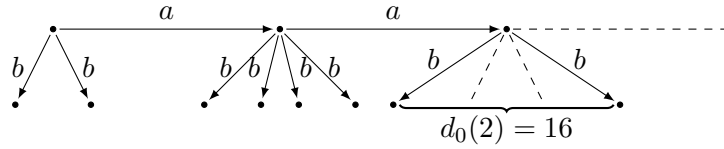

Figure 16: The tree t_{d_0} where $d_0(n) = 2^{2^n}$ for all $n \geq 0$.

Assume now, towards a contradiction, that t_{d_0} belongs to Tree_3. Consider the graph G in Graph_2 from which t can be obtained by unfolding from one of its roots r.

For all $n \geq 0$, we denote by a_n the unique vertex in G such that $r \xrightarrow{a^n} a_n$ and by V_n the set of vertices $\{v \mid r \xrightarrow{a^n b} v\}$. As $\mathrm{Unf}(G, r) \approx t_{d_0}$, we must have $a_n \neq a_m$ for $n \neq m$ and $|V_n| = 2^{2^n}$ for all $n \geq 0$. The set of vertices of G is therefore equal to $\{a_n \mid n \geq 0\} \cup \bigcup_{n \geq 0} V_n$ and its set of edges is:

$$\begin{aligned} &\{(a_n, a, a_{n+1}) \mid n \geq 0\} \\ \cup\ &\{(a_n, b, v) \mid n \geq 0 \text{ and } v \in V_n\} \end{aligned}$$

---

[6]Actually the condition holds for a larger class of graphs called #-graph-combs which encompasses all trees t_d.

Note that the sets V_n are not necessarily pairwise disjoint and that in particular, G is not necessarily a tree, as illustrated in Figure 17.
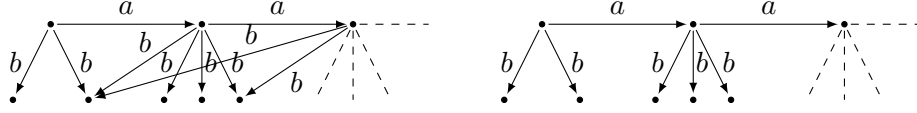

Figure 17: An example of a possible graph G whose unfolding is isomorphic to t_{d_0} (on the left) and of the tree $\mathcal{I}(G)$ (on the right).

Consider the MSO-interpretation $\mathcal{I}$ that erases the b-labelled edges (u, v) whenever there exists a vertex s such that $s \xrightarrow{a^+} u$ and $s \xrightarrow{b} v$ and keeps all other edges unchanged. By applying $\mathcal{I}$ to G (as illustrated in Figure 17, we obtain a tree $\mathcal{I}(G) \in \mathsf{Graph}_2$ with the same set of vertices as G. This tree has a unique infinite branch $(a_n)_{n \geq 0}$ and, for all $n \geq 0$, the vertex a_n has outgoing edges labelled b to the vertices in

$$V'_n = V_n \setminus (\cup_{m < n} V_m)$$

The tree $\mathcal{I}(G)$ is hence isomorphic to t_{d_1} where $d_1(n) = |V'_n|$ for all $n \geq 0$. To obtain a contradiction with [13, Theorem 4.5.3], we remark that for all $n \geq 1$, we have:

$$
\begin{aligned}
d_1(n) = |V'_n| &\geq 2^{2^n} - \sum_{m=0}^{n-1} 2^{2^m} \\
&\geq 2^{2^n} - n \cdot 2^{2^{n-1}} \\
&\geq 2^{2^{n-1}}(2^{2^{n-1}} - n) \\
&\geq 2^{2^{n-1}}.
\end{aligned}
$$

$\square$

4.4. **Contractions of synchronization trees in Tree_2.** In this section, we prove Theorem 4.16 which states that Γ-algebraic synchronization trees are the contractions of trees in Tree_2. We start by showing that the synchronization trees in Tree_2 are Γ-algebraic.

Proposition 4.15. *Synchronization trees in Tree_2 are Γ-algebraic.*

Proof. Let t be a synchronization tree in Tree_2. By the first property of Proposition 4.9, t is the B-contraction of a deterministic tree $t' \in \mathsf{Tree}_2$ labelled by $A \cup \{\mathsf{ex}\}$. As Γ-algebraic synchronization trees are closed under contraction (see Proposition 2.21), it is enough to show that t' is Γ-algebraic.

We are going to show that a tree t'' representing a $\tilde{\Gamma}$-term tree defining t' belongs to Tree_2. Recall that $\tilde{\Gamma}$ is the signature $\{+^n \mid n \geq 1\} \cup A \cup \{0, 1\}$ allowing for sums of arbitrary arity that was introduced in Remark 2.11 on page 12. Thanks to Theorem 4.7, this implies that t'' is defined by a $\tilde{\Gamma}$-algebraic recursion scheme. In turn this implies that t' is $\tilde{\Gamma}$-algebraic and hence Γ-algebraic (see Remark 2.11).

Let $<$ be an arbitrary total order on A. The tree t'' is obtained by applying a transduction $\mathcal{T}$ to t'. This transduction $\mathcal{T}$ does the following:

- Whenever a vertex v is the target of an edge labelled in A and is not the source of an edge, then the transduction adds a new edge labelled 0 from v to a new vertex introduced by the transduction.

- For every vertex u with $k \geq 1$ outgoing edges labelled $a_1 < \ldots < a_k \in A$, respectively, going to vertices $u_1, \ldots, u_k$, the transduction adds new vertices $v, v_1, \ldots, v_k$, $v'_1, \ldots, v'_k$ and edges $u \xrightarrow{+^k} v$, $v \xrightarrow{i} v_i$ and $v_i \xrightarrow{a_i} v'_i$. If u has an outgoing edge labelled by ex to a vertex v then the transduction adds a vertex v' and two edges $u \xrightarrow{k+1} v'$ and $v' \xrightarrow{1} v$.
- All the edges of the original structure are removed.

The tree t'' obtained by applying $\mathcal{T}$ to t' represents a term tree that defines t'. As t' belongs to Tree_2, it is (up to isomorphism) obtained by unfolding a graph $G \in \mathsf{Graph}_1$ from one of its roots r. Furthermore it can be checked that the transduction $\mathcal{T}$ commutes with the unfolding operation. Hence the tree t'' is isomorphic to $\mathrm{Unf}(\mathcal{T}(G), r)$ and therefore belongs to Tree_2 (as $\mathcal{T}(G)$ belongs to Graph_1). □

Theorem 4.16. *The contractions of the synchronization trees in Tree_2 are the Γ-algebraic synchronization trees.*

Proof. By Propositions 4.15 and 2.21, we have that each contraction of a synchronization tree in Tree_2 is Γ-algebraic. For the converse, a Γ-algebraic synchronization tree is defined as the contraction with respect to $\{+_1, +_2\}$ of the Γ-term tree defined by an algebraic scheme over Γ. From [18, Theorem 3.5], such a Γ-term tree belongs to Tree_2. Moreover, it may be seen as a synchronization tree over the alphabet which contains, in addition to the letters in A, the symbols $+_1$ and $+_2$. □

Thanks to the second property of Proposition 4.9, we have the following corollary.

Corollary 4.17. *Every Γ-algebraic tree is bisimilar to a tree in Tree_2.*

5. Branch languages of bounded synchronization trees

Call a synchronization tree *bounded* if there is a constant k such that the outdegree of each vertex is at most k. Our aim in this section will be to offer a language-theoretic characterization of the expressive power of Γ-algebraic recursion schemes defining synchronization trees. We shall do so by following Courcelle—see, e.g., [24]—and studying the branch languages of synchronization trees whose vertices have bounded outdegree. More precisely, we assign a family of branch languages to each bounded synchronization tree over an alphabet A and show that a bounded tree is Γ-algebraic if, and only if, the corresponding language family contains a deterministic context-free language (DCFL). Throughout this section, we will call Γ-algebraic trees just algebraic trees, and similarly for regular trees.

Definition 5.1. Suppose that $t = (V, v_0, E, l)$ is a bounded synchronization tree over the alphabet A. Denote by k the maximum of the outdegrees of the vertices of t. Let B denote the alphabet $A \times [k]$. A *determinization of t* is a tree $t' = (V, v_0, E, l')$ over the alphabet B which differs from t only in the labelling as follows. Suppose that $v \in V$ with outgoing edges $(v, v_1), \ldots, (v, v_\ell)$ labelled $a_1, \ldots, a_\ell \in \mathcal{A} \cup \{\mathsf{ex}\}$ in t. Then there is some permutation π of the set $[\ell]$ such that the label of each (v, v_i) in t' is $(a_i, \pi(i))$.

Consider a determinization t' of t. Let $v \in V$ and let $v_0, v_1, \ldots, v_m = v$ denote the vertices on the unique path from the root to v. The *branch word* corresponding to v in t' is the alternating word
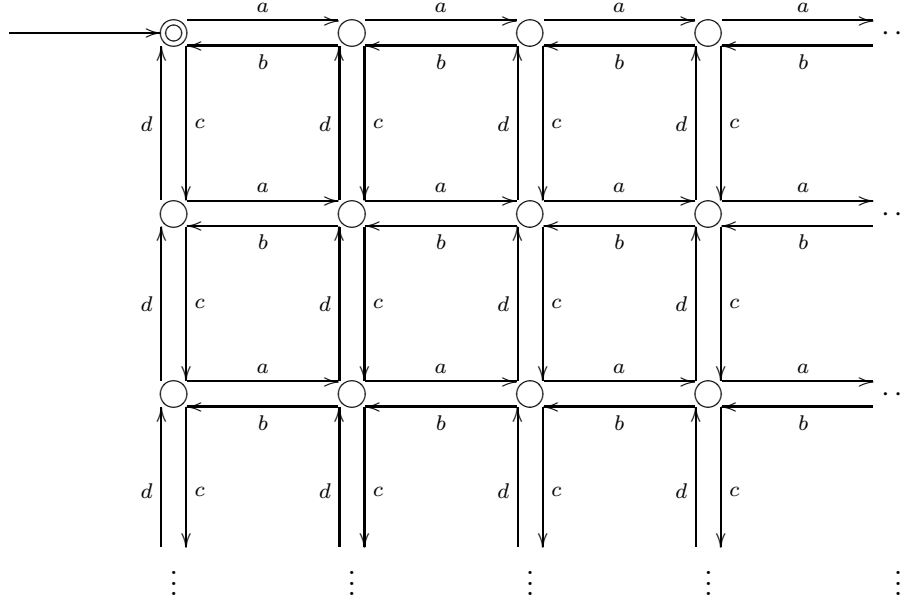
$$k_0(a_1, i_1)k_1 \ldots k_{m-1}(a_m, i_m)k_m$$

Figure 18: An LTS whose unfolding is not an algebraic synchronization tree. The initial node as outdegree 3 (recall that there is an implicit outgoing edge labelled by ex), the other nodes on the top-most row and on the left-most column also have outdegree 3 and all other "internal" nodes have outdegree 4.

where $k_0, \ldots, k_m$ denote the outdegrees of the vertices $v_0, \ldots, v_m$, and for each $j \in [m]$, (a_j, i_j) is the label of the edge (v_{j-1}, v_j) in t'. The *branch language* $L(t')$ corresponding to a determinization t' of t consists of all branch words of t'.

Finally, the family of branch languages corresponding to t is:

$$\mathcal{L}(t) = \{L(t') \mid t' \text{ is a determinization of } t\}.$$

By way of example, consider the LTS depicted in Figure 18. This LTS describes the behaviour of a bag over a four-letter alphabet, when we consider b to stand for the output of an item that was input via a, and d to signal the output of an item that was input via c. The synchronization tree t_{bag} that is obtained by unfolding this LTS from its start state is bounded. In fact, the outdegree of each non-leaf node is three or four. The branch words corresponding to the nodes of any determinization of the tree t_{bag} have the form

$$k_0(a_1, i_1)k_1 \ldots k_{m-1}(a_m, i_m)k_m,$$

where $k_0 = 3$, $k_m \in \{3, 4, 0\}$, $k_i \in \{3, 4\}$ for each $i \in [m-1]$, each $i_j \in [k_{j-1}]$ for $j \in [m]$ and $a_1 \ldots a_m$ is a word with the property that, in any of its prefixes, the number of occurrences of the letter a is greater than or equal to the number of occurrences of the letter b, and the number of occurrences of the letter c is greater than or equal to the number of occurrences of the letter d. Moreover, for each $j \in [m]$, $a_j = \text{ex}$ if and only if $j = m$ and $k_m = 0$. (Note that, when $a_m = \text{ex}$, the number of a's in $a_1 \ldots a_{m-1}$ equals the number of b's, and similarly for c and d.)

Theorem 5.2. *A bounded synchronization tree t is algebraic (respectively, regular) iff $\mathcal{L}(t)$ contains a DCFL (respectively, regular language).*

In the proof, we make use of the following construction. Suppose that $t = (V, v_0, E, l)$ is a bounded synchronization tree over A. Let k be defined as above, and let Σ be the signature containing the symbol $+_i$ of rank i for each $i \in [k]$, the constant symbols 0 and 1, and the letters of A as symbols of rank 1. We will sometimes write $+_0$ for 0. Then each determinization t' of t naturally corresponds to an 'alternating' *term tree* $T_{t'}$ in the initial continuous categorical Σ-algebra T_Σ^ω. As a partial function $\mathbb{N}^* \to \Sigma$, the term tree $T_{t'}$ is defined as follows. Consider a vertex $v \in V$ with corresponding branch word $k_0(a_1, i_1) \ldots (a_n, i_n) k_n$. Then $T_{t'}$ is defined on both words $i_1 1 \ldots i_n 1$ and $i_1 1 \ldots 1 i_n$, and

$$T_{t'}(i_1 1 \ldots i_n 1) = +_{k_n}$$
$$T_{t'}(i_1 1 \ldots 1 i_n) = \begin{cases} a_n & \text{if } a_n \in A \\ 1 & \text{if } a_n = \mathsf{ex}. \end{cases}$$

In addition, $T_{t'}$ is defined on the empty word ϵ and $T_{t'}(\epsilon) = +_{k_0}$, where k_0 is the outdegree of the root.

Lemma 5.3. *Suppose that $t \in \mathsf{ST}(A)$ is bounded and has a determinization t' such that the Σ-term tree $T_{t'}$ is algebraic (regular, resp.). Then t is algebraic (regular, resp.).*

Proof. We can turn $\mathsf{ST}(A)$ into a continuous categorical Σ-algebra by defining

$$+_i(t_1, \ldots, t_i) = t_1 + \cdots + t_i$$

for each $i \in [k]$ and $t_1, \ldots, t_i \in \mathsf{ST}(A)$, and similarly for morphisms between trees in $\mathsf{ST}(A)$. Now up to natural isomorphism, there is a unique categorical Σ-algebra morphism $h_\Sigma : T_\Sigma^\omega \to \mathsf{ST}(A)$. For any $t \in \mathsf{ST}(A)$ and for any determinization t' of t, h maps $T_{t'}$ to a tree isomorphic to t. Since, by the Mezei-Wright theorem [10], h_Σ preserves algebraic and regular objects, if $T_{t'}$ is algebraic or regular, then so is t. $\qquad\square$

Lemma 5.4. *Suppose that $t \in \mathsf{ST}(A)$ is bounded and algebraic (regular, resp.). Then t has a determinization t' such that the Σ-term tree $T_{t'}$ is algebraic (regular, resp.).*

Proof. Suppose that t is algebraic and bounded by k. Then, by the Mezei-Wright theorem [10], there is an algebraic term tree $T_0 \in T_\Gamma^\omega$ such that $h_\Gamma(T_0)$ is isomorphic to t, where h_Γ denotes the essentially unique continuous categorical Γ-algebra morphism $T_\Gamma^\omega \to \mathsf{ST}(A)$. We want to show that there is an alternating algebraic Σ-term tree $T_1 \in T_\Sigma^\omega$ such that $h_\Sigma(T_1)$ is isomorphic to t, where h_Σ is the essentially unique continuous categorical Σ-algebra morphism $T_\Sigma^\omega \to \mathsf{ST}(A)$. Since such a term tree T_1 is $T_{t'}$ for some determinization t' of t, this completes the proof.

We construct T_1 from T_0 in two steps. First, we replace all maximal subterms all of whose vertices are labelled $+$ or 0 by a single vertex labelled 0 to obtain a Γ-term tree T_0'. Second, we consider vertices of T_0' labelled $+$ whose parent, if any, is labelled in A. Since t is bounded by k, the subterm rooted at such a vertex can be written as $s_0(S_1, \ldots, S_n)$, where s_0 is a finite term all of whose vertices are labelled $+$ or 0 or a variable in the set $\{v_1, \ldots, v_n\}$ for some $n \leq k$, whose frontier word is $v_1 \cdots v_n$, and each S_i is a Γ-term tree whose root is labelled in $A \cup \{1\}$. We replace each such vertex by a vertex labelled $+_n$ having n outgoing edges labelled $1, \ldots, n$ connecting this vertex to the roots of $S_1, \ldots, S_n$, respectively.

The first transformation is a monadic colouring [13, 15] (a special case of monadic interpretation, which is also known as monadic marking [15]), since there is a monadic second-order formula $\phi(x)$ characterizing those vertices x of T_0 such that all vertices of the

subterm rooted at x are labelled $+$ or 0, but any other subterm containing x has a vertex labelled in $A \cup \{1\}$:

$$\forall y.(x \leq y \Rightarrow (l_+(y) \vee l_0(y)) \wedge \forall y.(y < x \Rightarrow (\exists z.y \leq z \wedge \bigvee_{a \in A \cup \{1\}} l_a(z))$$

(Here, $x \leq y$ denotes that there is a path from x to y and $x < y$ stands for $x \leq y \wedge x \neq y$. Moreover, $l_a(z)$ denotes that z is labelled a.) Thus, the first transformation gives a Γ-algebraic term tree, since algebraic term trees (and in fact deterministic algebraic trees in the Caucal's pushdown hierarchy) are closed under monadic colourings [15, Proposition 1].

In order to prove that the second transformation also gives an algebraic term tree, we argue on the level of graphs. Suppose that T_0' is the algebraic term tree obtained after the first step. Since T_0' is algebraic, it is the unfolding of a (deterministic) prefix recognizable graph G from its root r, see [15, 18]. Without loss of generality we may assume that every vertex of G is accessible from r. Our aim is to define a monadic transduction which, when applied to G, produces a graph G' whose unfolding from vertex r is T_1. Since, by Proposition 1 in conjunction with Lemma 2 in [15], prefix recognizable graphs are closed under monadic transductions, it follows that T_1 is algebraic.

We start by considering G together with a disjoint copy of G, whose vertices are ordered pairs $(v, 1)$, where v is vertex of G. The label of a vertex $(v, 1)$ is the label of v in G. The edges are the edges of G and an edge $v \to (v, 1)$ labelled $\#$ for each vertex v of G. Edges in G retain their label.

Then we drop all vertices of the form $(v, 1)$, where the label of v is different from $+$, using the formula

$$\exists y.E_\#(y, x) \wedge \neg l_+(x)$$

where the meaning of $E_\#(y, x)$ is that there is an edge labelled $\#$ from y to x, which is satisfied by exactly those vertices $(v, 1)$ labelled $+$. Moreover, we define new edges. First of all, we keep all edges $v \to v'$ of G such that v is labelled in A, or v is labelled $+$ but v' is not. Each such edge retains its label. Second, whenever v and v' are both labelled $+$ in G we introduce an edge $v \to (v', 1)$ and an edge $(v, 1) \to (v', 1)$ whose label is the same as that of the edge $v \to v'$ in G. For this purpose, we use the formulas in the free variables x, y,

$$l_+(x) \wedge l_+(y) \wedge \exists y'.(E_\#(y', y) \wedge (E_i(x, y') \vee \exists x'.(E_\#(x', x) \wedge E_i(x', y'))))$$

where $i = 1, 2$. Last, for each edge $v \to v'$ of G such that v is labelled $+$ but v' is not, we introduce an edge $(v, 1) \to v'$ whose label is that of the edge $v \to v'$. This is done by utilizing the formula

$$l_+(x) \wedge \neg l_+(y) \wedge \exists x'.(E_\#(x', x) \wedge E_i(x', y))$$

where again $i = 1, 2$.

Note that the unfolding of the graph constructed above from the vertex r is T_0'. Next, consider any vertex v labelled $+$ together with all the paths originating in v leading to a vertex labelled in $A \cup \{1\}$. Since t' is bounded by k, each such path is simple and the number of such paths is at most k. Let $v_1, \ldots, v_n$ denote the (not necessarily different) end vertices of these paths, ordered 'lexicographically'. We relabel v by $+_n$ and introduce a new edge $v \to v_i$ labelled i for each i. This is accomplished by using the following formulas. Let $\mathsf{Path}(x, X, y)$ denote a formula that says that the set of vertices X forms a path from x to y, the label of each vertex in X other than y is different from $+$, and the label of y is in

$A \cup \{1\}$. Then for each n, the formula

$$\exists X_1 \ldots X_n, \exists x_1, \ldots, x_n. \bigwedge_{i<j} \neg (X_i = X_j) \wedge \bigwedge_i \mathsf{Path}(x, X_i, x_i)$$

expresses that there are at least n different paths from x to some vertex y labelled in $A \cup \{1\}$, all of whose vertices different from y are labelled $+$. With the help of these formulas we can also express that there are exactly n such paths from x. Finally, when $\mathsf{Path}(x, X, y)$ and $\mathsf{Path}(x, X', y')$ with $X \neq X'$, we can express the fact that X is lexicographically less than X' by the formula

$$\exists Y, Z, Z'. \exists z_0, z, z'(X = Y \cup Z \wedge X' = Y \cup Z' \wedge \mathsf{Path}(x, Y, z_0) \wedge$$
$$(z = y \vee \mathsf{Path}(z, Z, y)) \wedge (z' = y' \vee \mathsf{Path}(z', Z', y')) \wedge E_0(z_0, z) \wedge E_1(z_0, z')$$

In addition to these new edges, we keep all edges originating from a vertex labelled in A (that are necessarily labelled 1). All vertices of the form $(v, 1)$ become inaccessible from r. The unfolding of the new graph from vertex r is almost an alternating term. In order to make it alternating, we have to add a new root labelled $+_1$ if r is labelled in A together with an edge to r, and replace each edge $v \to v'$ where both v and v' are labelled in A by new edges $v \to u$ and $u \to v$, where u is a new vertex labelled $+_1$. These edges are labelled 1. The new graph is still obtained by monadic transduction, and its unfolding is the alternating term T_1.

The same argument works in the regular case using the fact that regular terms are unfoldings of finite (deterministic) graphs, and that finite graphs are closed under monadic transduction. $\qquad \square$

Proof of Theorem 5.2. Suppose that $t \in \mathsf{ST}(A)$ is bounded. If t is algebraic, then by Lemma 5.4 there is some determinization t' of t such that $T_{t'}$ is algebraic. By Courcelle's theorem, the branch language of $T_{t'}$ is a DCFL. But the branch language of $T_{t'}$ is essentially $L(t')$.

Suppose now that t has a determinization t' such that $L(t')$ is a DCFL. Then the branch language of $T_{t'}$ is a DCFL, and thus by Courcelle's theorem, $T_{t'}$ is algebraic. The proof is completed by using Lemma 5.3.

A similar reasoning applies in the regular case.

The language-theoretic characterization of the class of bounded algebraic synchronization trees offered in Theorem 5.2 can be used to prove that certain trees are *not* algebraic.

Proposition 5.5. *The synchronization tree t_{bag} associated with the bag over a binary alphabet depicted on Figure 18 is not algebraic, even up to language equivalence.*

Proof. Recall that the branch words corresponding to the nodes of any determinization of the tree t_{bag} have the form

$$k_0(a_1, i_1)k_1 \ldots k_{m-1}(a_m, i_m)k_m,$$

where $k_0 = 3$, $k_m \in \{3, 4, 0\}$, $k_i \in \{3, 4\}$ for each $i \in [m-1]$, each $i_j \in [k_{j-1}]$ for $j \in [m]$ and $a_1 \ldots a_m$ is a word with the property that, in any of its prefixes, the number of occurrences of the letter a is greater than or equal to the number of occurrences of the letter b, and the number of occurrences of the letter c is greater than or equal to the number of occurrences of the letter d. Moreover, $a_j = \mathsf{ex}$ if and only if $j = m$ and $k_m = 0$. The words accepted by that LTS are those that in addition satisfy that the total number of occurrences of the letter a in $a_1 \ldots a_m$ is equal to the number of occurrences of the letter b, and the number

of occurrences of the letter c in $a_1 \ldots a_m$ is equal to the number of occurrences of the letter d and which end in 0.

If the language associated with any determinization of t_{bag} were context-free, then so would the language obtained by applying to each word in it the morphism that erases the letters k_j and renames each (a_j, i_j) to a_j. However, that language is not context free. Therefore, Theorem 5.2 yields that t_{bag} is not algebraic. □

The above proposition is a strengthening of a classic result from the literature on process algebra proved by Bergstra and Klop in [6]. Indeed, in Theorem 4.1 in [6], Bergstra and Klop showed that the bag over a domain of values that contains at least two elements is not expressible in BPA. Moreover, by Proposition 3.8, the collection of synchronization trees that are definable in BPA is strictly included in the set of Γ-algebraic ones (Theorem 3.2). Therefore, Proposition 5.5 is stronger than the above-mentioned inexpressibility result by Bergstra and Klop, and offers an alternative proof for it. Up to bisimilarity, we shall offer an even stronger statement in Section 6.1 (see Proposition 6.3).

As a final result, we can use Theorem 5.2 to characterize the synchronization trees of bounded degree in Tree_2.

Corollary 5.6. *The synchronization trees of bounded-degree in Tree$_2$ are the Γ-algebraic synchronization trees of bounded-degree.*

Proof. The direct inclusion is immediate. For the converse inclusion, we know by Theorem 5.2 that a Γ-algebraic synchronization tree of bounded-degree t has a determinization t' whose branch language is a deterministic context-free language. In particular, this implies that t' belongs to Tree_2. Indeed from a deterministic pushdown automaton accepting the branch language of t', we can construct a deterministic graph $G \in \text{Graph}_1$ whose unfolding from some root r is isomorphic to t'. Let $A = \{x_1, \ldots, x_k\}$ be the alphabet labelling G. Consider the transduction $\mathcal{T}$ defined as follows:

- for every vertex u, the transduction introduces new vertices $u_1, \ldots, u_k$;
- for every edge from u to v labelled by $x_\ell = (a, i)$, the transduction adds edges from each of the u_j, $j \in [k]$, to v_ℓ labelled by a.

Unfolding the graph $\mathcal{T}(G)$ from any of the vertices added by the transduction $\mathcal{T}$ for the root r gives a tree isomorphic to t. □

Note that in the previous proof, $\mathcal{T}$ cannot simply rename the edges labelled (a, i) to a. Consider for instance the case where G consists of two vertices r and s and two edges $r \xrightarrow{(a,1)} s$ and $r \xrightarrow{(a,2)} s$. Applying such a transduction would lead to a graph with only one edge.

6. Synchronization trees and logic

In this section, we investigate the consequences of the decidability of monadic second-order logic for our synchronization trees.

6.1. **A synchronization tree with an undecidable monadic theory.** In this subsection we point out that the synchronization tree associated with the bag process depicted on Figure 18 has an undecidable monadic theory (even without the root being the source of an exit edge). Hence that tree is not in the Caucal hierarchy and therefore, by Proposition 4.13, is not Δ-algebraic not even up to bisimilarity (Proposition 6.3). A similar result was obtained in [21, Section 6.6.2] for a slightly richer structure. For completeness sake, we give below a detailed proof of this result.

Proposition 6.1. *The synchronization tree t_{bag} associated with the bag process has an undecidable MSO-theory.*

Proof. Consider a 2-counter machine whose program P is given as a sequence of instructions $I_1, \ldots, I_n$ where each I_j has one of the following forms:

$$z := z + 1, k \quad z := z - 1, k, \quad z = 0?, k_1, k_2$$

where z is one of the counters x, y and k, k_1, k_2 are integers between 1 and n. At any moment of time, the value of the counters x and y is described by an ordered pair (m, n) of non-negative integers. The meaning of the above instructions is standard, where k denotes the index of the instruction to be performed after execution of the given instruction, if the instruction is an increment or decrement, and k_1 and k_2 denote the indices of the instructions to be performed depending on the outcome of the test, if the instruction is of the last form. The machine with program P halts if a decrement instruction $z := z - 1, k$ is executed, but the current value of the counter z is 0. A well-known undecidable question for 2-counter machines asks whether a 2-counter machine started with $(0, 0)$ ever halts.

We encode the halting program for a counter machine with program P in monadic second order logic as follows. Let $X_1, \ldots, X_n$ be a set of variables associated with the instructions of P. Then, for each instruction I_i, we consider the formula $\varphi_i(u)$ in the free first-order variable u in the language with four binary predicates associated with the edge relations $\xrightarrow{e}$, $e \in \{a, b, c, d\}$.

- If I_i is of the form $z := z + 1, k$, then $\varphi_i(u)$ expresses that $X_k(v)$ holds for all v such that $u \xrightarrow{e} v$, where e is a if $z = x$ and c if $z = y$.
- If I_i is of the form $z := z - 1, k$, then $\varphi_i(u)$ expresses that there exists a v with $u \xrightarrow{e} v$, and $X_k(v)$ holds for all such v, where e is b if $z = x$ and d if $z = y$.
- If I_i is of the form $z = 0?, k_1, k_2$, then $\varphi_i(u)$ expresses that $X_i(v_1)$ and $X_{k_1}(v_2)$ hold for all v_1, v_2 such that $u \xrightarrow{a} v_1$ and $v_1 \xrightarrow{b} v_2$, provided that there is no v with $u \xrightarrow{e} v$, and that $X_i(v_1)$ and $X_{k_2}(v_2)$ hold for all such v_1, v_2 otherwise, where again e is b if $z = x$ and d if $z = y$.

Now we assign to the machine with program P the formula

$$\varphi_P = \exists X_1 \ldots \exists X_n [\psi_1 \wedge \psi_2 \wedge \forall u (X_1(u) \to \varphi_1(u) \wedge \ldots \wedge X_n(u) \to \varphi_n(u))]$$

where ψ_1 asserts that the $X_1, \ldots, X_n$ are pairwise disjoint and jointly form an infinite path starting with the root, and ψ_2 says that the root belongs to X_1. Then the machine does not halt iff the synchronization tree defined by the process on Figure 18 is a model of φ_P. $\square$

Remark 6.2. Thomas showed in [43, Theorem 10] that the monadic second-order theory of the infinite two-dimensional grid is undecidable. However, we cannot use that result to prove that the synchronization tree t_{bag} has an undecidable monadic second-order theory. Indeed, the unfolding of the infinite two-dimensional grid is the full binary tree, which has a decidable monadic second-order theory by Rabin's celebrated Tree Theorem [40].

Proposition 6.3. *The synchronization tree t_{bag} is not Δ-algebraic up to bisimilarity.*

Proof. This statement follows from the more general remark that any synchronization tree t that is bisimilar to a deterministic synchronization tree t_0 having an undecidable MSO-theory also has an undecidable MSO-theory. Given a formula φ with no free variables, consider the formula

$$\varphi^* = \exists X \ \varphi_{\det}(X) \wedge \varphi'$$

where φ' is the formula φ in which all quantifications are relativized to X and $\varphi_{\det}(X)$ states that if a node has a successor by an a-labelled edge then it has one and only one successor by an a-labelled edge which belongs to X. If the formula $\varphi_{\det}(X)$ is satisfied on t for some set of vertices U then t restricted to the vertices in U is a deterministic tree isomorphic to t_0 (cf. Lemma 2.5 on page 9). Clearly φ^* holds on t if and only if φ holds on t_0. This implies that the MSO-theory of t is undecidable. $\square$

The argument used in this proof can also be used to show that t_{bag} does not belong to the Caucal hierarchy up to bisimilarity.

6.2. **Minimization.** It is well known that, for each bisimulation equivalence class $\mathcal{C}$ of synchronization trees in $\mathsf{ST}(A)$, there is a tree $t_{\mathcal{C}} \in \mathcal{C}$ such that for all $t \in \mathcal{C}$ there is a *surjective* morphism $\varphi : t \to t_{\mathcal{C}}$, and, moreover, the relation

$$R = \varphi \cup \varphi^{-1} = \{(u,v),(v,u) : \varphi(u) = v\}$$

is a bisimulation between t and $t_{\mathcal{C}}$. Furthermore, $t_{\mathcal{C}}$ is unique up to isomorphism. When $t \in \mathcal{C}$, we call $t_{\mathcal{C}}$ the *minimization* of t.

The minimization of a tree $t \in \mathsf{ST}(A)$ can be constructed in the following way. We define an increasing sequence $V_0, V_1, \ldots$ of sets of vertices of t, where V_0 is a singleton set containing only the root of t. The construction will guarantee that, for each i, the set V_i contains only vertices of depth at most i, and whenever a vertex v belongs to V_i and there is a path from a vertex u to v, then u is also in V_i. Given V_i and $u \in V_i$ of depth i, consider the set $S(u)$ of successors of u. We may divide $S(u)$ into equivalence classes according to the bisimulation equivalence classes of the corresponding subtrees and the label of the edge coming from u. To this end, we define $v \sim v'$, for $v, v' \in S(u)$, if the subtrees rooted at v and v' are bisimilar and if $u \xrightarrow{a} v$ and $u \xrightarrow{a} v'$ for some $a \in A$. Then we select a representative of each $\sim$-equivalence class. The set V_{i+1} consists of all vertices in V_i together with those vertices in $S(u)$ of depth $i + 1$ selected above, where u ranges over the set of all vertices in V_i of depth i. Finally, let $V = \bigcup_{i \geq 0} V_i$. The 'subtree" of t spanned by the vertices in V is the minimization of t.

It is known, see e.g. [8], that the minimization of a Γ-regular synchronization tree is Γ-regular. In contrast with this result, we have:

Proposition 6.4. *There exists a Γ-algebraic synchronization tree whose minimization does not have a decidable MSO-theory, and hence does not belong to the Caucal hierarchy and is neither a Γ-algebraic nor a Δ-algebraic synchronization tree.*

Proof. Let $A = \{a, b, c, d, e, f\}$. Consider the following Γ-algebraic scheme:

$$\begin{aligned} S \ &= \ F(0,0) \\ F(x,y) \ &= \ a.F(f.x, y) + b.F(x, f.y) + c.F(0, y) + d.F(x, 0) + e.x + e.y \ . \end{aligned}$$

Let t be the synchronization tree defined the above scheme. For a word $u \in \{a, b, c, d\}^*$, we designate by $\|u\|_a$ (resp. $\|u\|_b$) the number of a's (resp. b's) in the longest suffix of u that does *not* contain any occurrence of the letter c (resp. d). Intuitively, the tree t consists of a full quaternary deterministic tree t with edges labelled in $\{a, b, c, d\}$ such that every vertex of t is also the origin of two additional parallel disjoint branches. Since t is deterministic, we may identify each vertex of t with a word $u \in \{a, b, c, d\}^*$. The two additional branches at vertex u of t are such that their edge labels form the words $ef^{\|u\|_a}$ and $ef^{\|u\|_b}$, respectively.

The minimization t' of t is obtained by removing one of the two branches labelled $ef^{\|u\|_a}$ for all vertices u of t such that $\|u\|_a = \|u\|_b$.

The fact that t' has an undecidable MSO-theory is based on a reduction from the halting problem for 2-counter machines with increment, reset and equality test [46]. The idea is, as usual, to define, for every such machine M, a closed MSO-formula φ_M such that $t' \models \varphi_M$ if and only if M does not halt.

When constructing the formula φ_M, we associate to a vertex u in $\{a, b, c, d\}^*$ representing a possible history of the computation of M from its initial state, the integer $\|u\|_a$ as the current value of the first counter, and the integer $\|u\|_b$ as the current value of the second counter of the machine. Assuming that the current values of the counters are given by vertex u, we show how to simulate the various operations of the machine. The increment of the first (resp. second) counter is obtained by moving to vertex ua (resp. ub). The reset of the first (resp. second) counter is obtained by moving to vertex uc (resp. ud). Finally, the test for equality between the two counters is performed by testing that vertex u has only one outgoing edge labelled e. The details of the construction are similar to those in Section 6.1. ◻

The above result yields that the collection of synchronization trees in the Caucal hierarchy is *not* closed under minimization. Indeed, there is a Γ-algebraic tree whose minimization is not in the Caucal hierarchy. This leaves open the corresponding question for Δ-regular synchronization trees.

7. Open questions

There are several questions that we leave open in this paper and that lead to interesting directions for future research.

- Is there a strict expressiveness hierarchy for Γ- and Δ-algebraic recursion schemes that is induced by the maximum rank of the functor variables used in defining recursion schemes?
- Is the minimal synchronization tree that is bisimilar to a Δ-regular synchronization tree also Δ-regular? If not, is it in the Caucal hierarchy?
- The Γ-algebraic scheme we use in the proof of Proposition 6.4 uses a binary functor variable. Is the minimal synchronization tree that is bisimilar to a tree defined by a Γ-algebraic scheme involving only unary functor variables Γ-algebraic?

References

[1] S. Abramsky. A domain equation for bisimulation. *Inform. and Comput.*, 92 (1991), no. 2, 161–218.
[2] A.V. Aho. Indexed grammars — an extension of context-free grammars. *Journal of the ACM* 15 (1968), 647–671.

[3] J.C.M. Baeten, T. Basten, and M.A. Reniers. *Process Algebra: Equational Theories of Communicating Processes*, volume 50 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, November 2009.

[4] J.C.M. Baeten and W.P. Weijland. *Process Algebra*. Cambridge Tracts in Theoretical Computer Science, 18. Cambridge University Press, Cambridge, 1990.

[5] J.W. de Bakker. *Recursive Procedures*. Mathematical Centre Tracts, No. 24. Mathematisch Centrum, Amsterdam, iv+108 pp., 1971.

[6] J.A. Bergstra and J.W. Klop. The algebra of recursively defined processes and the algebra of regular processes. Proceedings of ICALP 1984, LNCS 172, pp. 82–94, Springer, 1984.

[7] S.L. Bloom, Z. Ésik and D. Taubner. Iteration theories of synchronization trees. *Inform. and Comput.*, 102 (1993), no. 1, 1–55.

[8] S.L. Bloom and Z. Ésik. *Iteration Theories.* Springer, 1993.

[9] S.L. Bloom and Z. Ésik. The equational theory of regular words. *Information and Computation*, 197 (2005), 55–89.

[10] S.L. Bloom and Z. Ésik. A Mezei-Wright theorem for categorical algebras. *Theoretical Computer Science*, 411 (2010), 341–359.

[11] S.L. Bloom, J. W. Thatcher, E. G. Wagner and J. B. Wright. Recursion and iteration in continuous theories: The "*M*-construction". *J. Comput. System Sci.*, 27 (1983), 148–164.

[12] A. Blumensath. Prefix recognizable graphs and monadic second order logic. Technical Report AIB-06-2001, RWTH Aachen, 2001.

[13] L. Braud. The structure of linear orders in the pushdown hierarchy. PhD thesis, Institut Pascal Monge, 2010.

[14] F. van Breugel. An introduction to metric semantics: Operational and denotational models for programming and specification languages. *Theoretical Computer Science*, 258(2001), 1–98.

[15] A. Carayol and S. Wöhrle. The Caucal hierarchy of infinite graphs in terms of logic and higher-order pushdown automata. Proceedings of FSTTCS 03, LNCS 2914, pp. 112–123, Springer, 2003.

[16] Arnaud Carayol Regular sets of higher-order pushdown stacks. Proceedings of MFCS 2005, LNCS 3618, pp. 168–179, Springer, 2005.

[17] A. Carayol. Automates infinis, logique et langages. PhD thesis, Université Rennes I, 2006.

[18] D. Caucal. On infinite terms having a decidable monadic theory. Proceedings of MFCS 02, LNCS 2420, 165–176, Springer, 2002.

[19] D. Caucal. On infinite transition graphs having a decidable monadic theory. *Theoretical Computer Science* 290(2003), 79–115.

[20] S. Christensen. Decidability and decomposition in process algebras. PhD thesis ECS-LFCS-93-278, Department of Computer Science, University of Edinburgh, 1983.

[21] T. Colcombet. Propriétés et représentation de structures infinies. PhD thesis, Université Rennes I, March 2004.

[22] B. Courcelle. A representation of trees by languages I. *Theoretical Computer Science* 6 (1978), 255–279.

[23] B. Courcelle. A representation of trees by languages II. *Theoretical Computer Science* 7 (1978), 25–55.

[24] B. Courcelle. Fundamental properties of infinite trees, *Theoretical Computer Science* 25 (1983), 69–95.

[25] B. Courcelle. Monadic second-order definable graph transductions: A survey. *Theoretical Computer Science*, 126 (1994), 53–75.

[26] B. Courcelle and M. Nivat. Algebraic families of interpretations. In *17th Annual Symposium on Foundations of Computer Science*, IEEE Computer Society, 1976, 137–146.

[27] H.D. Ebbinghaus and J. Flum. *Finite Model Theory.* Springer-Verlag, 1995.

[28] Z. Ésik. Continuous additive algebras and injective simulations of synchronization trees. Fixed Points in Computer Science, 2000 (Paris). *Journal of Logic and Computation*, 12 (2002), 271–300.

[29] M.J. Fischer, Grammars with macro-like productions, In 9th Annual Symp. Switching and Automata Theory, Schenedtady, NY, USA, 1968, IEEE Press, 1968, 131–142.

[30] J.A. Goguen, J.W. Thatcher, E.G. Wagner and J.B. Wright. Initial algebra semantics and continuous algebras. *Journal of the ACM* 24 (1977), 68–95.

[31] I. Guessarian, *Algebraic Semantics*. LNCS 99, Springer, 1981.

[32] S. Milius and L. Moss. The category-theoretic solution of recursive program schemes. *Theoretical Computer Science* 366 (2006), 3–59.

[33] D. Mandrioli and C. Ghezzi. *Theoretical foundations of Computer Science.* John Wiley and Sons, 1988.

[34] R. Milner. An algebraic definition of simulation between programs. In Proceedings 2nd Joint Conference on Artificial Intelligence, pages 481–489. BCS, 1971. Also available as Report No. CS-205, Computer Science Department, Stanford University.

[35] R. Milner. *A Calculus of Communicating Systems.* LNCS 92, Springer, 1980.

[36] R. Milner. *Communication and Concurrency.* Prentice Hall, 1989.

[37] F. Moller. Infinite results. Proceedings of CONCUR '96, Concurrency Theory, 7th International Conference, LNCS 1119, pp. 195–216, Springer, 1986

[38] M. Nivat, On the interpretation of recursive polyadic program schemes. *Symposia Mathematica* XV (1975), 255 -281.

[39] D.M.R. Park. Concurrency and automata on infinite sequences. In Theoretical Computer Science, 5th GI-Conference, LNCS 104, pp. 167–183, Springer, 1981.

[40] M.O. Rabin. Decidability of second-order theories and automata on infinite trees. *Transactions of the American Mathematical Society* 141 (1969), 1–35.

[41] D.S. Scott. The lattice of flow diagrams. In Symposium on Semantics of Algorithmic Languages 1971, Lecture Notes in Mathematics, vol. 188, Springer, 1971, pp. 311–366.

[42] C. Stirling. Decidability of bisimulation equivalence for pushdown processes. Technical Report EDI-INF-RR-0005, School of Informatics, University of Edinburgh, 2000.

[43] W. Thomas. A short introduction to infinite automata. In Developments in Language Theory, LNCS 2295, pp. 130–144, Springer, 2001.

[44] W. Thomas. Constructing infinite graphs with a decidable MSO-theory. In Proceedings of the 28th International Symposium on Mathematical Foundations of Computer Science, LNCS 2747, pp. 113–124, Springer, 2003.

[45] G. Winskel. Synchronization trees. *Theoretical Computer Science* 34 (1984), no. 1–2, 33–82.

[46] A. Wojna. Counter machines. *Information Processing Letters* 71 (1991), 193–197.