Algebraic Synchronization Trees and Processes* Draft dated April 26, 2012

Luca Aceto[1], Arnaud Carayol[2], Zoltán Ésik[3], and Anna Ingólfsdóttir[1]

[1] ICE-TCS, School of Computer Science, Reykjavik University, Iceland
[2] Institute Gaspard-Monge, University Paris-Est, France
[3] Institute of Informatics, University of Szeged, Hungary

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, 21, 27, 33, 36] 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

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

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 natural numbers that returns its second argument when the first is zero and

* Luca Aceto and Anna Ingólfsdóttir have been partially supported by the project 'Meta-theory of Algebraic Process Theories' (nr. 100014021) of the Icelandic Research Fund. The work on the paper was partly carried out while Luca Aceto and Anna Ingólfsdóttir held Abel Extraordinary Chairs at Universidad Complutense de Madrid, Spain, supported by the NILS Mobility Project. Arnaud Carayol has been supported by the project AMIS (ANR 2010 JCJC 0203 01 AMIS). Zoltán Ésik has been partially supported by the project TÁMOP-4.2.1/B-09/1/KONV-2010-0005 'Creating the Center of Excellence at the University of Szeged', supported by the European Union and co-financed by the European Regional Fund, and by the National Foundation of Hungary for Scientific Research, grant no. K 75249. Zoltán Ésik's work on this paper was also partly supported by grant T10003 from Reykjavik University's Development Fund and a chair from the LabEx Bézout.

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 [21], 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), 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 [21].

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 [31] and of Basic Process Algebra (BPA) [3] as a way of defining possibly infinite synchronization trees [30]. 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 2.

In the setting of process algebras such as CCS [31] 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, 23, 26, 27], and have been used in [9, 10, 24] to give semantics to recursion schemes over synchronization trees and words. (We refer the interested reader to [28] 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. Moreover, we compare the expressiveness of those recursion schemes to that of the low levels in Caucal's pushdown hierarchy. (As a benefit of the comparison with the Caucal hierarchy, we obtain structural properties and decidability of Monadic Second-Order Logic [38].) We show that each Δ-regular tree is Γ-algebraic (Theorem 1) 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 7 we prove that the synchronization tree associated with the bag over a binary alphabet (which is depicted on Figure 7) 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].

Since each Γ-algebraic synchronization tree is also Δ-algebraic, we obtain the following strict expressiveness hierarchy, which holds up to equivalences of synchronization trees such as bisimilarity [31, 34] and language equivalence:

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

In the setting of language equivalence, the notion of Γ-regularity corresponds to the regular languages, the one of Δ-regularity corresponds to the context-free languages and Δ-algebraicity corresponds to the macro languages [25], which coincide with the languages generated by Aho's indexed grammars [2].

In order to obtain a deeper understanding of Γ-algebraic recursion schemes, as a final main contribution of the paper, we characterize their expressive power by following the lead of Courcelle [19–21]. 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 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.

The technical developments in this paper make use of techniques and results from a variety of areas of theoretical computer science. We employ 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 2, characterizing the expressive power of Γ-algebraic recursion schemes in

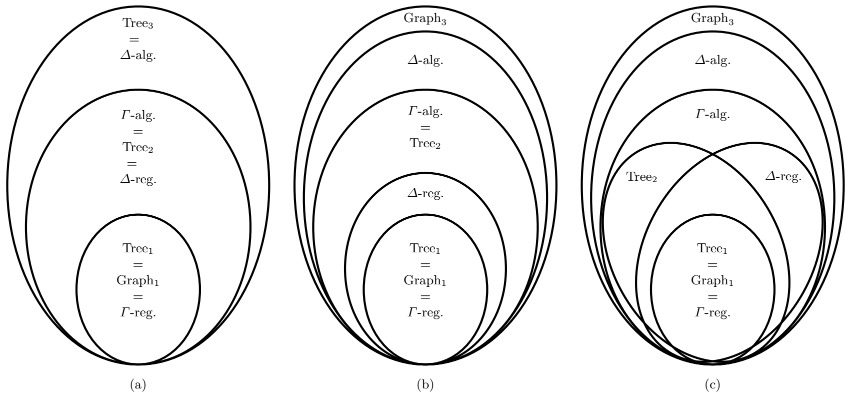

Fig. 1. The expressiveness hierarchies up to language equivalence (a), up to bisimilarity (b) and up to isomorphism (c)

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 tools 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 recall the notion of continuous categorical Σ-algebra. Synchronization trees are defined in Section 3, together with the signatures Γ and Δ that contain the operations on those trees that we use in this paper. We introduce regular and algebraic recursion schemes, as well as their initial solutions, in Section 5. Section 6 studies the expressive power of regular and algebraic recursion schemes over the signatures Γ and Δ. In Section 7, 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. Section 11 concludes the paper and lists topics for future research.

2 Continuous categorical algebras

In this section, we recall the notion of continuous categorical Σ-algebra. These structures were used in [9, 10, 24] 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 A equipped with a functor $\sigma^A : A^n \to A$ for each $\sigma \in \Sigma_n$, $n \geq 0$. A *morphism* between categorical Σ-algebras A and B is a functor $h : A \to B$

such that for each $\sigma \in \Sigma_n$, the diagram

$$\begin{tikzcd}
A^n \arrow[r, "\sigma^A"] \arrow[d, "h^n"'] & A \arrow[d, "h"] \\
B^n \arrow[r, "\sigma^B"'] & B
\end{tikzcd}$$

commutes *up to a natural isomorphism*. Here, the functor $h^n : A^n \to B^n$ maps each object and morphism $(x_1, \ldots, x_n)$ in A^n to $(h(x_1), \ldots, h(x_n))$ in B^n. A morphism h is *strict* if, for all $\sigma \in \Sigma$, the natural isomorphism π_σ is the identity.

Suppose that A is a categorical Σ-algebra. We call A *continuous* if A has a distinguished initial object (denoted $\perp^A$ or 0^A) and colimits of all ω-diagrams $(f_k : a_k \to a_{k+1})_{k \geq 0}$. Moreover, each functor σ^A 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 A with colimits $(x_k \xrightarrow{\phi_k} x)_k$ and $(y_k \xrightarrow{\psi_k} y)_k$, respectively, then

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

has colimit

$$(\sigma^A(x_k, y_k) \overset{\sigma^A(\phi_k,\psi_k)}{\longrightarrow} \sigma^A(x, y))_k.$$

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. Below we will often write just σ for σ^A, in particular when A is understood.

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

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

with $\langle f_1, \ldots, f_n \rangle$ standing for the target tupling of the continuous functors $f_1, \ldots, f_n : A^k \to A$. On natural transformations, $\sigma^{[A^k \to A]}$ is defined in a similar fashion. In $[A^k \to A]$, 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 $\perp$ or 0 of rank 0, interpreted in a continuous categorical Σ-algebra as its initial object.

2.1 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 [21, 23, 26, 27, 33, 36]. We say that a continuous categorical Σ-algebra A 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 A has initial object and colimits of ω-diagrams correspond to the requirements that there is a least element $\perp$ and each ω-chain $(a_n)_n$ has a supremum, denoted $\sup_n a_n$. Thus, A is an *ω-complete poset* as is each finite power A^n of A, equipped with the pointwise order. A continuous functor $A^n \to A$ is simply a function that preserves the supremum of ω-chains. Every such continuous function preserves the partial order. Continuous functions are ordered pointwise. As is well known, if A and B are ω-complete posets, then so is the poset $[A \to B]$ of all continuous functions $A \to B$.

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^ω_Σ denote the set of all Σ-term trees. We equip T^ω_Σ with a partial order defined by $t \leq t'$ iff t' can be constructed from t by replacing some leaves labelled $\perp$ by some term trees in T^ω_Σ. It is well-known (see e.g. [21, 26, 27]) that this relation turns T^ω_Σ into an ω-complete partial order with least element $\perp$. We may further turn T^ω_Σ 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 1. *For every continuous categorical Σ-algebra A there is, up to natural isomorphism, a unique homomorphism $T^\omega_\Sigma \to A$.*

An alternative representation of a term tree in T^ω_Σ 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 in $\overline{u} \in \overline{\Sigma}^*$ which encodes the unique path from the root to that vertex. The edge-labelled graph 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 $\overline{u}\sigma$.

In the sequel, we will be concerned with two signatures associated with an ordinary alphabet A, denoted Γ and Δ. Each letter $a \in A$ has rank 1 in Γ and rank 0 in Δ. In addition, Γ contains the symbols $+, 0, 1$, and Δ the symbols $+, \cdot, 0, 1$ where $+$ and $\cdot$ are of rank 2 and 0 and 1 have rank 0. An example of a continuous ordered Δ-algebra is the language semiring $P(A^*)$ of all subsets of A^*, 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 $L \subseteq A^*$.

3 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" $l : E \to A \cup \{\mathsf{ex}\}$.

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 $l'(\phi(u), \phi(v)) = l(u, v)$. Morphisms are therefore functional *simulations* [29, 34]. It is clear that the trees and tree morphisms form a category. 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.

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, for any tree t, $a(t)$ is the tree $a \cdot t$. 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 nonroot vertices of t and that agrees with h' on the nonroot 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

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

Remark 1. An account of synchronization trees based on complete partial orders and continuous functions was given in [39]. However, as pointed out in [7], the treatment in [39] is not abstract, since the order relation $\leq$ depends on the concrete representation of the trees: there exist isomorphic synchronization trees t_i, t_i', $i = 1, 2$ such that $t_1 \leq t_2$ but t_1' and t_2' are not related by $\leq$.

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.

Definition 1. *Two synchronization trees $t = (V, v_0, E, l)$ and $t' = (V', v_0', E', l')$ are* bisimilar *[31, 34] if there is some symmetric relation $R \subseteq (V \times V') \cup (V' \times V)$ 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 $(v_1', v_2') \in R$. 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.*

4 Morphisms

By Proposition 1, 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 to* 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
– 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.

In Proposition 2 below, we will prove that $\tau(t) = t'$, the synchronization tree denoted by t.

We give several examples. As a first example, consider the term

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

Then $G(t)$ 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, consider the term tree t defined by the scheme

$$\begin{aligned} G &= (a \cdot H) \cdot c + d \\ H &= H \cdot 1 \ . \end{aligned}$$

Now the root 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 $v_0, v_1, \ldots$, such that for each $i \geq 1$ there is an edge from v_{i+1} to v_i, which is labelled 1, and there is a c-labelled edge from v_1 to

v_0. Last, $*$ is a vertex, and there exist edges from u_2 and v_0 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, 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. *For every Δ-term tree t, the image t' of t with respect to the essentially unique continuous Δ-algebra morphism $T^\omega_\Delta \to \mathsf{ST}(A)$ 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. $\square$

Next we describe the essentially unique morphism of continuous categorical Γ-algebras $T_\Gamma^\omega \to \mathsf{ST}(A)$. Let $t \in T_\Gamma^\omega$. The set of vertices of the synchronization tree t' denoted by t is composed of the root of t, the vertices of t which are successors of vertices labelled in A, and all leaves labelled 1. Suppose that u and v are different vertices of t'. There is an edge from u to v in t' if v belongs to the subtree of t rooted at u, and all vertices along the unique path from u to v different from u and v are labelled $+$. The label of this edge in t' is the label of v in t.

Using the representation of a Γ-term tree t as an edge-labelled tree, the above procedure can alternatively be described as follows. Consider the set of those vertices of t given by words ending in 1 or a letter a_1, where $a \in A$, together with the root which is given by the empty word ϵ. Then there is an edge in t' from u to v labelled $a \in A$ exactly when $v = uw$ for some word in $\{+_1, +_2\}^* a_1$. Moreover, there is an edge from u to v in t' labelled ex exactly when v is of the form uw for a word $w \in \{+_1, +_2\}^* 1$.

Let τ' denote the function that maps t to t'.

Proposition 3. *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 simpler than that of Proposition 2.

Proposition 4. *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 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', and the empty tree is mapped to the empty language. 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 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. $\square$

Remark 2. The function τ' describes both the unique continuous categorical Γ-algebra and the unique continuous Δ-algebra morphism $\mathsf{ST}(A) \to P(A^*)$.

5 Algebraic objects and functors

When n is a nonnegative integer, we denote the set $\{1, \ldots, n\}$ by $[n]$.

Definition 2. *Let Σ be a signature. A Σ-*recursion scheme*, or* recursion scheme over Σ*, is a sequence E of equations*

$$\begin{aligned} F_1(v_1, \ldots, v_{k_1}) &= t_1 \\ &\vdots \\ F_n(v_1, \ldots, v_{k_n}) &= t_n \end{aligned} \tag{2}$$

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") of rank k_i, $i \in [n]$. A Σ-recursion scheme is regular *if $k_i = 0$, for each $i \in [n]$.*

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

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

Then $A^{r(\Phi)}$ is a continuous categorical Σ-algebra, as noted in Section 2.

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

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

More precisely, we define t^A as follows. Let f_i, g_i denote continuous functors $A^{k_i} \to A$, $i \in [n]$, 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 $t^A(f_1, \ldots, f_n)$ is the ith projection functor $A^m \to A$, and $t^A(\alpha_1, \ldots, \alpha_n)$ 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 $t^A(f_1, \ldots, f_n) = \sigma^A \circ \langle h_1, \ldots, h_k \rangle$ and $t^A(\alpha_1, \ldots, \alpha_n) = \sigma^A \circ \langle \beta_1, \ldots, \beta_k \rangle$, where $h_j = t_j^A(f_1, \ldots, f_n)$ and $\beta_j = t_j^A(\alpha_1, \ldots, \alpha_n)$ 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 $t^A = f_i \circ \langle h_1, \ldots, h_{k_i} \rangle$, 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 $t^A(\alpha_1, \ldots, \alpha_n)$ is the identity natural transformation $t^A(f_1, \ldots, f_n) \to t^A(f_1, \ldots, f_n)$.

In any continuous categorical Σ-algebra A, by target-tupling the functors t_i^A, we obtain a continuous functor

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

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

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

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

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

so that, in particular,

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

at least up to isomorphism, for each $i = 1, \ldots, n$.

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

Definition 3. *Suppose that A is a continuous categorical Σ-algebra. We call a functor $f : A^m \to A$* Σ-algebraic*, if there is a recursion scheme E such that f is isomorphic to $|E|_1^A$, 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.*

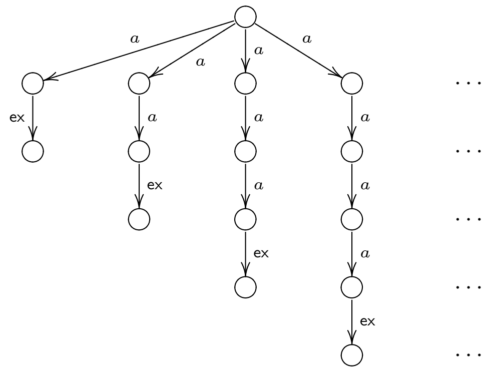

Fig. 2. The Δ-regular and Γ-algebraic tree $\sum_{i \geq 1} a^i$

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

Example 1. The following Δ-regular recursion scheme

$$X = (X \cdot a) + a \tag{3}$$

has the infinitely branching tree $\sum_{i \geq 1} a^i$ depicted on Figure 2 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-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 $a.1$.) So Γ-algebraic recursion schemes can be used to define infinitely branching trees that have an infinite number of subtrees, even up to bisimilarity [31, 34].

6 A comparison

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.

It is clear that every Γ-regular tree is Δ-regular and that the inclusion is proper, since every Γ-regular tree has, up to isomorphism, only a finite number of subtrees, see [8, 30], while there exist Δ-regular and Γ-algebraic trees that do not have this property. (See Example 1.) The strict inclusion also holds with respect to strong bisimulation equivalence or language equivalence. 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 [32], 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 [18]. 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 [32] for the details of the formal definitions, which are not needed to appreciate the results to follow, and further pointers to the literature.

Proposition 5. *If $f, f' : \mathsf{ST}(A)^k \to \mathsf{ST}(A)$ are Γ-algebraic, 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 $f \cdot f'$. □

Example 2. The tree b^ω is Γ-regular since it is the initial solution of the following Γ-regular recursion scheme

$$X = b.X.$$

According to the above proposition, the tree $(\sum_{i\geq 1} a^i)\cdot b^\omega$ is Γ-algebraic. Indeed, it is the initial solution of the recursion scheme

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

Theorem 1. *Every Δ-regular tree is Γ-algebraic.*

Proof. Consider a regular Δ-recursion scheme E,

$$\begin{aligned} F_1 &= t_1 \\ &\vdots \\ F_n &= t_n, \end{aligned}$$

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

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

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, since $t^{\mathsf{ST}(A)}(r_1, \ldots, r_n) = 1^{\mathsf{ST}(A)}$, 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 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

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

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

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

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

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

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

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

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. □

Example 3. Suppose that E is given by the single equation

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

Then E' is

$$\begin{aligned} G_0 &= G(1) \\ G(v) &= v + a.(G(b.v)) \end{aligned}$$

Both of them define the tree depicted on Figure 3.

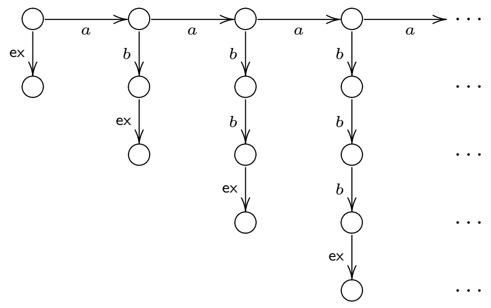

Fig. 3. The Δ-regular and Γ-algebraic tree from Example 3

The translation given in the proof of Theorem 1 resulted in an algebraic recursion scheme G over Γ of the form

$$\begin{aligned} G_0 &= G_1(1) \\ G_1(v_1) &= p_1 \\ &\vdots \\ G_n(v_1) &= p_n \end{aligned}$$

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

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

$$\begin{aligned} F_1 &= \widehat{p_1} \\ &\vdots \\ F_n &= \widehat{p_n} \end{aligned}$$

By the proof of Theorem 1, G corresponds to E. Thus, E and G define the same tree.

Proposition 6.

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 1. In order to prove the latter statement, consider the LTS depicted on Figure 4. 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 [32, 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.

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

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. □

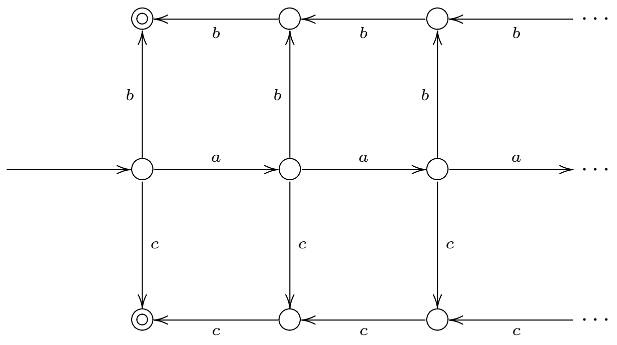

Fig. 4. An LTS whose unfolding is an algebraic synchronization tree

Further examples of algebraic synchronization trees Consider the LTS on Figure 5. 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 5, 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.

As noted above, this algebraic recursion scheme over Γ corresponds to the regular recursion scheme over Δ,

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

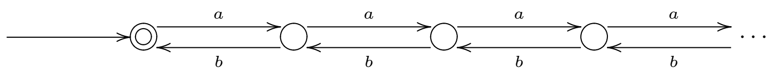

Fig. 5. An LTS accepting the Dyck language

As another example, consider the LTS on Figure 6. This LTS is not expressible in BPA modulo modulo bisimilarity—see [32, 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.

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

The idea underlying the above definition is as follows. At any given vertex in the LTS on Figure 6, 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.

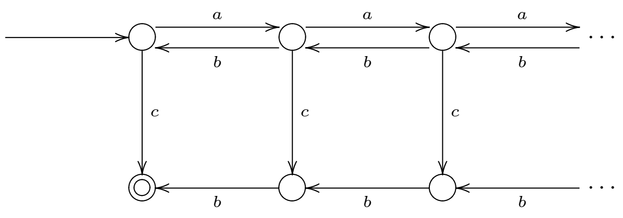

Fig. 6. Another LTS that cannot be expressed in BPA, but whose unfolding is an algebraic synchronization tree

7 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., [21]—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 4. *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*

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

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:

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

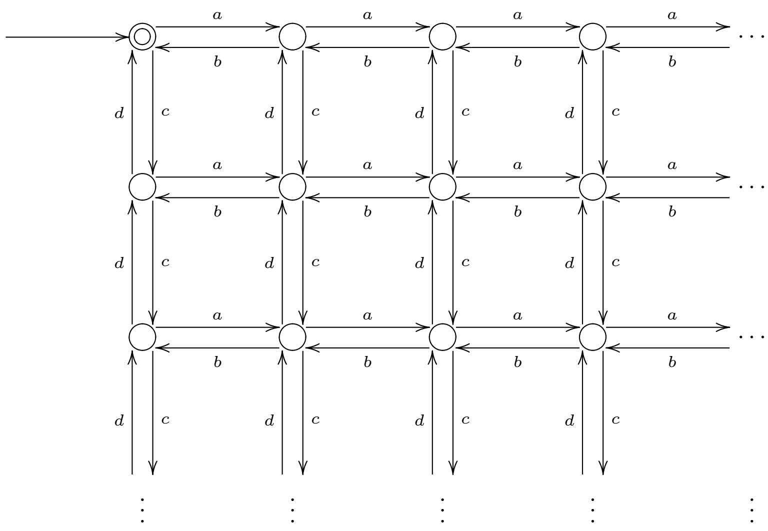

Fig. 7. An LTS whose unfolding is not an algebraic synchronization tree

By way of example, consider the LTS depicted in Figure 7. This LTS describes the behaviour of a bag over a two-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. The branch words corresponding to the nodes of any determinization of the tree t_{bag} have the form

$$3(a_1, i_1)3 \ldots 3(a_m, i_m)k_m,$$

where k_m is either 3 or 0, $i_1, \ldots, i_m \in [3]$ 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 $j \in [m]$, $a_j = \mathsf{ex}$ if and only if $j = m$ and $k_m = 0$. (Note that, when $a_m = \mathsf{ex}$, 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 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 a *term tree* $T_{t'}$ in the initial continuous categorical Σ-algebra T^ω_Σ. 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 $k_0(a_1, i_1) \dots (a_n, i_n)k_n$. Then $T_{t'}$ is defined on both words $i_1 1 \dots i_n 1$ and $i_1 1 \dots 1 i_n$, and

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

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 1. *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

$$+_i(t_1, \dots, t_i) = t_1 + \dots + t_i$$

for each $i \in [k]$ and $t_1, \dots, t_i \in \mathsf{ST}(A)$, and similarly for morphisms between trees in $\mathsf{ST}(A)$. Now up to natural isomorphism, there is a unique categorical Σ-algebra morphism $h_\Sigma : T^\omega_\Sigma \to \mathsf{ST}(A)$. 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. □

Lemma 2. *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 $T_0 \in T^\omega_\Gamma$ such that $h_\Gamma(T_0)$ is isomorphic to t, where h_Γ denotes the essentially unique continuous categorical Γ-algebra morphism $T^\omega_\Gamma \to \mathsf{ST}(A)$. We want to show that there is an alternating algebraic Σ-term tree $T_1 \in T^\omega_\Sigma$ such that $h_\Sigma(T_1)$ is isomorphic to t, where h_Σ is the essentially unique continuous categorical Σ-algebra morphism $T^\omega_\Sigma \to \mathsf{ST}(A)$. 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 $s_0(S_1, \dots, S_n)$, where s_0 is a finite term all of whose vertices are labelled $+$ or 0 or a variable in the set $\{v_1, \dots, v_n\}$ 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 $1, \dots, n$ connecting this vertex to the roots of $S_1, \dots, S_n$, 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 $A \cup \{1\}$:

$$\forall y.(x \leq y \Rightarrow (\lambda_+(y) \vee \lambda_0(y)) \wedge \forall y.(y < x \Rightarrow (\exists z. y \leq z \wedge \bigvee_{a \in A \cup \{1\}} \lambda_a(z))$$

Thus, the first transformation gives an algebraic Γ-term tree, since algebraic term trees (and in fact deterministic algebraic trees in the 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, 16]. 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

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

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,

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

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

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

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 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

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

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. $\square$

Proof of Theorem 2. Suppose that $t \in \mathsf{ST}(A)$ is bounded. If t is algebraic, then by Lemma 2 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 1.

A similar reasoning applies in the regular case. $\Box$

The language-theoretic characterization of the class of bounded algebraic synchronization trees offered in Theorem 2 can be used to prove that certain trees are *not* algebraic.

Proposition 7. *The synchronization tree t_{bag} associated with the bag over a binary alphabet depicted on Figure 7 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

$$3(a_1, i_1)3 \dots 3(a_m, i_m)k_m,$$

where k_m is either 3 or 0, $i_1, \dots, i_m \in [3]$ and $a_1 \dots a_m$ 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 $a_1 \dots a_m$ is equal to the number of occurrences of the letter b, and the number of occurrences of the letter c in $a_1 \dots a_m$ 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 2 yields that t_{bag} is not algebraic. $\Box$

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 6, the collection of synchronization trees that are definable in BPA is strictly included in the set of Γ-algebraic ones (Theorem 1). Therefore, Proposition 7 is stronger than the above-mentioned inexpressibility result by Bergstra and Klop, and offers an alternative proof for it. Up to tree isomorphism, we shall offer an even stronger statement in Section 10.1.

Example 4. Consider the following Δ-algebraic scheme:

$$F_0 = F(1)$$
$$F(v) = a \cdot F(b \cdot v) + v \cdot c \cdot v \cdot 0$$

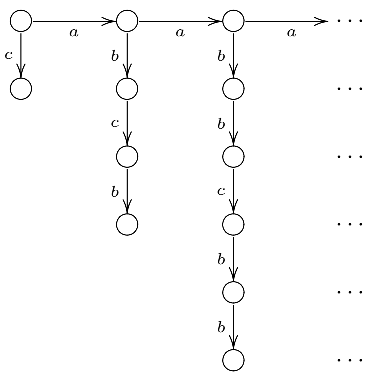

Fig. 8. A Δ-algebraic tree that is not Γ-algebraic

The synchronization tree defined by this scheme is depicted on Figure 8. Given any determinization of this tree, the language $\{a^n b^n c b^n : n \geq 0\}$ is a homomorphic image of the intersection of its branch language with a regular language. Thus there is no Γ-algebraic synchronization tree that is language equivalent to it.

We conclude:

Proposition 8. *There is a Δ-algebraic tree that is not language equivalent to any Γ-algebraic tree.*

Remark 3. A Δ-algebraic recursion scheme is essentially (more precisely, up to distributivity of $\cdot$ over finite sums) the same thing as a macro grammar [25]. Macro grammars generate the class of languages as Aho's indexed grammars [2].

8 The Caucal hierarchy and prefix recognizable graphs

In the following section, we will compare the expressiveness of recursion schemes to that of the low classes in the Caucal hierarchy [16]. For the sake of completeness, following [15], we recall that Tree_0 and Graph_0 denote the collections of finite, edge-labelled trees and graphs, respectively. Moreover, for each $n \geq 0$, Tree_{n+1} stands for the collection of 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) [22]. It is well known that Graph_1 is the set of all prefix-recognizable graphs [17].

8.1 Prefix recognizable graphs

In the remainder of this section, we will discuss in more detail the fundamental family of prefix-recognizable graphs and study some properties of the graphs in this family that will find application in Section 9.

Definitions on labelled graphs We begin by presenting labelled graphs and the operation of contraction on such graphs.

Definition 5 (edge-labelled graph). *An* edge-labelled graph*, whose edges are labelled by elements of a finite set A, is a pair (V, E) where V is a 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.*

An edge-labelled graph (V, E) id deterministic *iff for all $u, v_1, v_2 \in V$ and $a \in A$, if $(u, a, v_1) \in E$ and $(u, a, v_2) \in E$ then $v_1 = v_2$.*

In an edge-labelled graph $G = (E, V)$, 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. We write $u \xrightarrow{L} v$, for vertices u and v, and for some $L \subseteq A^*$, when there is some word $w \in L$ such that $u \xrightarrow{w} v$.

Figure 9 gives two examples of edge-labelled graphs. The names of the vertices are given between parentheses next to the dots representing the vertices.

Definition 6 (Root of a graph). *A vertex r is a* root *of an edge-labelled graph if there exists a path from r to each vertex of the graph.*

We now define the analog of ε-closure on NFAs for edge-labelled graphs.

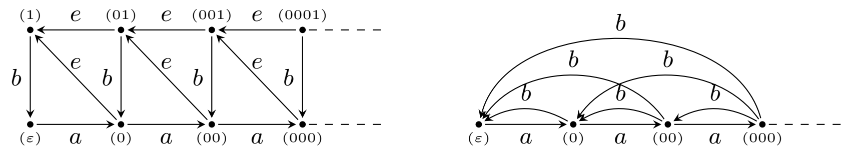

Fig. 9. An edge-labelled graph (on the left) admitting ε as a root and its $\{e\}$-contraction from ε (on the right)

Definition 7 (Contraction). *Let A and E be two disjoint sets of labels. The E-contraction of an edge-labelled graph $G = (V, E)$ with labels in $A \uplus E$ from a root r is the graph $H = (V', E')$ labelled in A whose set of nodes V' consists of the root r together with all the targets of A-labelled arcs in G and whose edges are such that:*

$$u \xrightarrow[H]{a} v \Leftrightarrow u \in V', v \in V' \text{ and } u \xrightarrow[G]{E^* a} v.$$

Example 5. Figure 9 presents, on the left, an edge-labelled graph and, on the right, its $\{e\}$-contraction.

Prefix-recognizable graphs: Definitions and examples Prefix-recognizable graphs are edge-labelled graphs introduced by Caucal [17]. The vertices of a prefix-recognizable graph are words over a finite alphabet and the edges labelled by a given symbol a are given by a so-called prefix recognizable relation.

For the rest of the definitions, we fix the alphabet C for the words used as vertices of a graph. Let us start with the definition of prefix-recognizable relations.

Definition 8 (Prefix-recognizable relations). *A* simple prefix-recognizable *relation over words in C^* is given by three non-empty regular languages U, V and $W \subseteq C^*$ and is equal to:*

$$\{(uv, uw) \mid u \in U, v \in V \textit{ and } w \in W\}.$$

This relation is denoted $U \cdot (V \times W)$.

A prefix-recognizable relation *is a finite union of simple prefix-recognizable relations.*

For a binary relation R over C^*, we write $\mathrm{Im}(R)$ *for the image of R and* $\mathrm{Dom}(R)$ *for its domain. Formally,*

$$\mathrm{Dom}(R) = \{u \mid (u, v) \in R \textit{ for some } v\} \quad \textit{and}$$
$$\mathrm{Im}(R) = \{v \mid (u, v) \in R \textit{ for some } u\}.$$

Example 6. The identity relation, which is equal to $C^* \cdot (\varepsilon \times \varepsilon)$, and the prefix relation, which is equal to $C^* \cdot (\varepsilon \times C^*)$, are examples of simple prefix-recognizable relations. Assuming that the alphabet C consists of two elements 0 and 1 with the natural order, the lexicographic order on C^* is an example of prefix-recognizable relation that can, for instance, be defined by $C^* \cdot (0C^* \times 1C^*) \cup C^* \cdot (\varepsilon \times C^*)$.

Definition 9. *A prefix-recognizable graph labelled by a finite set A is given by a tuple $(V, (R_a)_{a \in A})$ where V is a regular set of words in C^* and, for all $a \in A$, R_a is a prefix-recognizable relation such that* $\mathrm{Im}(R_a) \cup \mathrm{Dom}(R_a) \subseteq V$.

The corresponding graph has V as set of vertices and the following set of edges:

$$\{(u, a, v) \mid a \in A \textit{ and } (u, v) \in R_a\}.$$

Example 7. Assume that $C = \{0, 1\}$ and consider the prefix recognizable graph $(V, (R_a, R_b))$ labelled by $\{a, b\}$ with $V = 0^*$, $R_a = 0^* \cdot (\varepsilon \times 0)$ and $R_b = 0^* \cdot (0^+ \times \varepsilon)$. The resulting labelled graph is depicted on the right of Figure 9.

Definition 10 (Edge- and node-labelled graph). *An* edge- and node-labelled graph *whose edges and nodes are respectively labelled by finite sets A and B is a triple (V, E, ℓ) where (V, E) is an edge-labelled graph and ℓ is a mapping from V to B.*

All notions presented above naturally extend to the node-labelled setting. For instance, to obtain prefix-recognizable graphs that are also node labelled by a finite $B = \{b_1, \ldots, b_n\}$, we simply give a partition of V as $V_1 \uplus \cdots \uplus V_n$. Nodes in V_i are labelled by b_i.

Properties of prefix-recognizable relations and graphs In this section, we present various technical results concerning prefix-recognizable relations and graphs, which will find application in Section 9.

We start by presenting a normal form for prefix-recognizable relations in terms of what we call split prefix-recognizable relations. This normal form ensures that if an ordered pair (x, y) belongs to the union of relations

$$U_1 \cdot (V_1 \times W_1) \cup \cdots \cup U_n \cdot (V_n \times W_n)$$

then there exists a unique $i \in [n]$ and a unique decomposition $x = uv$ and $y = uw$ such that $u \in U_i$, $v \in V_i$ and $w \in W_i$.

Definition 11 (Split prefix-recognizable relation). *A prefix-recognizable relation is* split *if it is defined by simple prefix-recognizable relations $U_1 \cdot (V_1 \times W_1), \ldots, U_n \cdot (V_n \times W_n)$ such that:*

- *the relations $U_i \cdot (V_i \times W_i)$, $i \in [n]$ are pairwise disjoint;*
- *for all $i \in [n]$, $First(V_i) \cap First(W_i) \subseteq \{\varepsilon\}$, where $First(L) = \{c \mid cu \in L \text{ and } c \in C\} \cup \{\varepsilon \mid \text{if } \varepsilon \in L\}$, for each $L \subseteq C^*$.*

Lemma 3. *Every prefix-recognizable relation is equal to a split prefix-recognizable relation.*

Proof. Let R be a prefix-recognizable relation. From point (iii) of the proof of Theorem 3.17 in [17], we know that R can be expressed as a finite union $\bigcup_{i\in[n]} U_i \cdot (V_i \times W_i)$ of simple prefix-recognizable relations such that $\text{First}(V_i) \cap \text{First}(W_i) \subseteq \{\varepsilon\}$, for all $i \in [n]$.

To every binary n-tuple $\bar{b} = (b_1, \ldots, b_n) \in \{0,1\}^n$, we associate three regular sets $U_{\bar{b}}$, $V_{\bar{b}}$ and $W_{\bar{b}}$, which are respectively defined by:

$$\begin{aligned}
U_{\bar{b}} &= (\textstyle\bigcap_{i\in[n],b_i=1} U_i) \cap (\bigcap_{i\in[n],b_i=0} \overline{U_i}) \\
V_{\bar{b}} &= (\textstyle\bigcap_{i\in[n],b_i=1} V_i) \cap (\bigcap_{i\in[n],b_i=0} \overline{V_i}) \\
W_{\bar{b}} &= (\textstyle\bigcap_{i\in[n],b_i=1} W_i) \cap (\bigcap_{i\in[n],b_i=0} \overline{W_i})
\end{aligned}$$

where, for a language $L \subseteq C^*$, as usual $\overline{L}$ designates the complement $C^* \setminus L$ of L.

Claim. The relation R is the disjoint union, denoted by R', of all the simple prefix-recognizable relations $U_{\bar{d}} \cdot (V_{\bar{e}} \times W_{\bar{f}})$ with $\bar{d}, \bar{e}$ and $\bar{f} \in \{0,1\}^n$ such that $d_i = e_i = f_i = 1$, for some $i \in [n]$.

In order to show that R is included in R', let (x, y) be an ordered pair in R. There exists $i \in [n]$ such that x and y can be written as uv and uw respectively with $u \in U_i$, $v \in V_i$ and $w \in W_i$.

Consider the tuples $\bar{d}, \bar{e}$ and $\bar{f} \in \{0,1\}^n$ such that, for all $j \in [n]$, $d_j = 1$ if $u \in U_j$ and $d_j = 0$ otherwise, $e_j = 1$ if $v \in V_j$ and $e_j = 0$ otherwise, $f_j = 1$ if $w \in W_j$ and $f_j = 0$ otherwise. We have $(x, y) \in U_{\bar{d}} \cdot (V_{\bar{e}} \times W_{\bar{f}}) \subseteq R'$. The converse inclusion is immediate.

Let $\overline{d}, \overline{e}$ and $\overline{f} \in \{0,1\}^n$ such that $d_i = e_i = f_i = 1$ for some $i \in [n]$. We have $\mathrm{First}(V_{\bar{e}}) \subseteq \mathrm{First}(V_i)$ and $\mathrm{First}(W_{\bar{f}}) \subseteq \mathrm{First}(W_i)$ and hence $\mathrm{First}(V_{\bar{e}}) \cap \mathrm{First}(W_{\bar{f}}) \subseteq \mathrm{First}(V_i) \cap \mathrm{First}(W_i) \subseteq \{\,\varepsilon\,\}$.

Let $\overline{d}, \overline{d}', \overline{e}, \overline{e}', \overline{f}$ and $\overline{f}' \in \{0,1\}^n$ such that for some $i, i' \in [n]$, $d_i = e_i = f_i = 1$ and $d'_{i'} = e'_{i'} = f'_{i'} = 1$. Let $S = U_{\overline{d}} \cdot (V_{\overline{e}} \times W_{\overline{f}})$ and $S' = U_{\overline{d}'} \cdot (V_{\overline{e}'} \times W_{\overline{f}'})$.

As $\mathrm{First}(V_{\bar{e}}) \cap \mathrm{First}(W_{\bar{f}}) \subseteq \{\,\varepsilon\,\}$ and $\mathrm{First}(V_{\bar{e}'}) \cap \mathrm{First}(W_{\bar{f}'}) \subseteq \{\,\varepsilon\,\}$, for any ordered pair (x, y) and decomposition $x = uv$ and $y = uw$ we have that:

$$\begin{aligned}(x,y) \in S &\Leftrightarrow u = x \wedge y \text{ and } u \in U_{\overline{d}}, v \in V_{\overline{e}} \text{ and } w \in W_{\overline{f}}.\\ (x,y) \in S' &\Leftrightarrow u = x \wedge y \text{ and } u \in U_{\overline{d}'}, v \in V_{\overline{e}'} \text{ and } w \in W_{\overline{f}'}.\end{aligned}$$

where $x \wedge y$ denotes the longest common prefix of x and y. The existence of an ordered pair $(x, y) \in S \cap S'$ implies that $\overline{d} = \overline{d}'$, $\overline{e} = \overline{e}'$ and $\overline{f} = \overline{f}'$. Recall that by definition, $\overline{d} \neq \overline{d}'$ implies $U_{\overline{d}} \cap U_{\overline{d}'} = \emptyset$. □

Lemma 4. *For every prefix-recognizable graph G labelled in A admitting a root r, there exists a deterministic prefix-recognizable graph H labelled in $A \uplus E$ such that the contraction of H according to E from r is equal to G.*

Moreover for all vertices u and v of H, there exists at most one path from u to v labelled in E^*a for each $a \in A$ and for any vertex u, if u is the source of an E-labelled edge then all edges with u as source are labelled in E.

Proof. Let $G = (V, (R_a)_{a \in A})$ be a prefix-recognizable graph labelled in A. We can assume, without loss of generality, that the vertices in V are words over the alphabet $C = \{0, 1\}$. By Lemma 3, the relation R_a, $a \in A$, can be expressed as the disjoint union of simple relations $U_{a,1} \cdot (V_{a,1} \times W_{a,1}), \ldots, U_{a,n_a} \cdot (V_{a,n_a} \times W_{a,n_a})$ with $\mathrm{First}(V_{a,i}) \cap \mathrm{First}(W_{a,i}) \subseteq \{\,\varepsilon\,\}$, for all $i \in [n_a]$.

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 $H = (V', (R'_a)_{a \in A \cup E})$ satisfying the properties stated above.

The set of labels E is $E = \{e_0, e_1, e_2\} \cup \{e_{a,i} \mid a \in A \text{ and } i \in [n_a]\}$. 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 E^*a from a vertex $x \in V$ to a vertex $y \in V$ simulates the relation $R_{a,i}$ for some $a \in A$ and $i \in [n_a]$. For a fixed $a \in A$ and $i \in [n_a]$, the simulation is done using two set of vertices: vertices in $V_{a,i} = \{0,1\}^* Q_{a,i}(a, i)$ and vertices in $V'_{a,i} = \{0,1\}^* Q_{a,i}(a, i)$. 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

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 by $e_{a,i}$ 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 $uq(a,i) \xrightarrow[H]{e_1} u \star (a,i) \xrightarrow[H]{a} u$ (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:

$$\begin{array}{llll} u & \xrightarrow[H]{e_{a,i}} & u\, q_{a,i}(a,i) & \text{for } u \in V \quad (1) \\ u\, bq(a,i) & \xrightarrow[H]{e_0} & u\, \delta_{a,i}(q,b) & \text{for } b \in \{0,1\} \quad (2) \\ u\, q(a,i) & \xrightarrow[H]{e_1} & u\, q'_{a,i}(a,i) & \text{if } q \in F_{i,a} \text{ and } u \in U \quad (3) \\ u\, q'(a,i) & \xrightarrow[H]{e_b} & ub\, \delta'_{a,i}(q,b)(a,i) & \text{for } b \in \{0,1\} \quad (4) \\ u\, q'(a,i) & \xrightarrow[H]{e_2} & u \star (a,i) & \text{if } q' \in F'_{a,i}a \quad (5) \\ u \star (a,i) & \xrightarrow[H]{a} & u & \text{for } u \in V \quad (6) \end{array}$$

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

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

It remains to show the other direction.

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

Proof. Let $x, y \in \{0,1\}^*$ such that there exists a path from x to y in H labelled in E^*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

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

with $\delta(q_{a,i}, u) \in F_{a,i}$, $\tilde{w} = e_{w(1)} \cdots e_{w(|w|)}$, $\delta(q'_{a,i}, w) \in F'_{a,i}$ and $u \in U_{a,i}$. It follows that (x, y) belongs to $R_{a,i}$. As the relation was taken to be split (cf. Lemma 3), 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 E-contraction of H is equal to G. □

9 Synchronization trees in the Caucal hierarchy

In this section, we investigate the relationship between Γ-regular, Γ-algebraic, Δ-regular and Δ-algebraic schemes and the low levels of the Caucal hierarchy.

9.1 Δ-algebraic trees are in Graph_3

In this section, we use Proposition 2 to prove that all Δ-algebraic trees belong to Graph_3 and thus have decidable MSO theories.

Proposition 9. *All Δ-algebraic synchronization trees are in* Graph_3 *and have a decidable MSO-theory.*

Proof. Let E be an algebraic Δ-scheme. Let t be the Δ-term defined by E, and t' the synchronization tree defined by E. By the Mezei-Wright theorem and Proposition 2, we have that $t' = \tau(t)$. By [16], t is in Tree_2, thus by [15], $G(t) \in \mathsf{Graph}_2$. Recall that $\tau(t)$ is the tree obtained from $G(t)$ by first unfolding it from the root and then contracting the result with respect to $\{1\}$. The first transformation gives a tree in Tree_3, cf. [16], and the second a tree in Graph_3. Thus $t' \in \mathsf{Graph}_3$. The decidability of the MSO theory of t' now follows from [16, Proposition 2.1 and 2.3]. □

Corollary 1. *The unfolding of the LTS bag presented in Figure. 6 does not have a decidable MSO-theory and hence is not Δ-algebraic.*

9.2 Synchronization trees in Tree_2 are Γ-algebraic

Recall that tree is in Tree_2 if it is the unfolding of a prefix-recognizable graph.

In this subsection, we will represent a Γ-term tree as an edge labelled tree. The synchronization tree associated to a Γ-term tree was described in 4. For the reader's convenience, we repeat the definition here.

Suppose that t is a possibly infinite Γ-term tree. Then the set X of vertices of the associated synchronization tree in $\mathsf{ST}(A)$ is composed of the root ϵ of t and those vertices of t represented by words (over the alphabet containing the letters $+_1, +_2, a_1, 1$, where $a \in A$) ending in a_1 or 1. The edges of the synchronization tree associated to t are given by:

$$\begin{aligned} &\{u \xrightarrow{a} uv \mid u, uv \in X \text{ and } v \in \{+_1, +_2\}^* a_1\} \\ \cup\ &\{u \xrightarrow{\text{ex}} uv \mid u, uv \in X \text{ and } v \in \{+_1, +_2\}^* 1\} \end{aligned}$$

Recall that by the Mezei-Wright theorem, a synchronization tree is Γ-algebraic if and only if it is isomorphic to the synchronization tree associated to the Γ-term tree defined by an algebraic Γ-scheme.

It is convenient to add to the signature Γ sums of arbitrary nonzero rank. For this, we consider the signature[4] $\tilde{\Gamma} = \{+^n \mid n \geq 1\} \cup A \cup \{0, 1\}$ where the rank of each symbol $+^n$ is n. The synchronization tree associated to a $\tilde{\Gamma}$-term tree is defined similarly as the synchronization tree associated to a Γ-term tree (simply replace $\{+_1, +_2\}^*$ by $\{+_1, +_2, +^m_n \mid m \geq 1, n \leq m\}^*$.)

Lemma 5. *A synchronization tree in* $\mathsf{ST}(A)$ *is* Γ*-algebraic if and only if it is isomorphic to the synchronization tree associated to the* $\tilde{\Gamma}$*-term tree defined by an algebraic* $\tilde{\Gamma}$*-scheme.*

Proof. We can transform an algebraic $\tilde{\Gamma}$-scheme into a Γ-scheme defining the same synchronization tree by replacing every occurrence of a subterm on the right-hand side of an equation of the form $+^n(t_1, \ldots, t_n)$ 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 $t_1 + 0$. □

Proposition 10. *Synchronization trees in* Tree_2 *are* Γ*-algebraic.*

Proof. Let t be a synchronization tree in Tree_2. Let G be a prefix recognizable graph with a distinguished vertex r such that t is isomorphic to the unfolding of G from r. We can assume, without loss of generality, that r is a root of G (otherwise we consider the graph obtained by restricting to vertices reachable from r which is again prefix-recognizable. The closure of prefix-recognizable graphs under restriction to reachable vertices from a given vertex follows from the closure of this class of graph under MSO-interpretation [12, Theorem 3]).

Consider the prefix recognizable graph H labelled in $(A \cup \{\mathsf{ex}\}) \uplus E$ obtained in Lemma 4. We have that the E-contraction of H from r is equal to G. Due to the additional properties of H stated in Lemma 4, the unfolding of the E-contraction of H from r is isomorphic to the E-contraction of the tree obtained by unfolding H from r. In other terms, the E-contraction from the root of $t' = \mathrm{Unf}(H, r)$ is isomorphic to t.

Let $<$ be an arbitrary total order on E. We are going to transform t' into a Γ-term tree by applying a transduction to H. This transduction $\mathcal{T}$ does the following:

- relabels a-labelled edges by a_1, for all $a \in A$,
- relabels ex-labelled edges by 1,
- whenever a vertex v is the target of an edge labelled in A which is not the source of an edge, then 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 $e_1 < \ldots < e_k$, respectively, relabels these edges by $+^k_1, \ldots, +^k_k$.

[4] Although the signature is infinite, we will always only use a finite subset of it.

Let t_1 be $\tilde{\Gamma}$-term obtained by unfolding $\mathcal{T}(H)$ from r. It is easy to check that the synchronization tree associated to t_1 is isomorphic to the E-contraction of $\mathrm{Unf}(H, r)$. We know from [16, Theorem 3.5] that t_1 is the term tree defined by some algebraic $\tilde{\Gamma}$-scheme. This establishes by the Mezei-Wright theorem that t_1 is $\tilde{\Gamma}$-algebraic and hence Γ-algebraic (by Lemma 5). $\square$

9.3 Γ-algebraic trees as contractions of trees in Tree_2

Proposition 11. *The contraction of Γ-algebraic synchronization trees from their root are Γ-algebraic synchronization trees.*

Proof. Let t be a synchronization tree defined by an algebraic Γ-scheme E. Let B be a subset of A and let t' be the synchronization tree obtained by contracting t from its root 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 $\overline{F(v_1 \dots v_n)} = \overline{t}$ 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 $b \in B$.

The Γ-term defined by E' is the contraction of the Γ-term defined by S from its root with respect to B. It then follows that the synchronization tree defined by E' is the contraction of the synchronization tree defined by E from its root with respect to B. $\square$

Proposition 12. *The contractions of the synchronization trees in* Tree_2 *are the Γ-algebraic synchronization trees.*

Proof. By Proposition 10 and 11, 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 Γ-scheme. From [16, 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 $+_2$. $\square$

Proposition 13. *Every Γ-algebraic tree is bisimilar to a tree in* Tree_2.

Proof. This property is based on the following simple remark. Let G be a graph labelled in $(A \cup \{1\}) \uplus E$ with a root r, the E-contraction of $\mathrm{Unf}(G, r)$ is bisimilar to the unfolding from r of the E-contraction of G from r.

Let t be a Γ-algebraic synchronization tree. By Proposition 12, t is the contraction of a tree $t' \in \mathsf{Tree}_2$. By definition, t' is the unfolding of a prefix recognizable graph G from a root r. By the above, t is bisimilar to the unfolding of the contraction of G. $\square$

Proposition 14. *The synchronization trees of bounded-degree in $Tree_2$ are the Γ-algebraic synchronization trees of bounded-degree.*

9.4 A $\varDelta$-regular tree not in Tree_2

A *prefix recognizable relation* over a finite alphabet C is a finite union of relations of the form $U \cdot (V \times W)$, for some nonempty regular languages $U, V, W \subseteq C^*$. A *prefix recognizable graph* over an alphabet B is an edge-labelled directed 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 relations $\xrightarrow{b}$ are prefix recognizable relations over C. (Of course, the alphabet C may be fixed to be a 2-element alphabet.) A rooted prefix recognizable graph has a distinguished vertex, called the root. The vertices reachable from the root form a prefix recognizable graph cf. [], so that if we needed, we may assume that all vertices are reachable from the root. The class of graphs Tree_2 consists of all unfoldings of rooted prefix recognizable graphs (such that every vertex is reachable from the root).

Lemma 6. *Let G be a prefix recognizable graph whose vertices have finite outdegree. 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 outdegree of u_i is at most $C(i+1)$.*

Proof. 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 outdegree, 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 outdegree of a vertex u of G is at most $(|u|+1)W$, we conclude that the outdegree of u_i is at most $C(i+1)$ for all $i \geq 0$ with $C = (d'+1)W$. □

Proposition 15. *There is a $\varDelta$-regular synchronization tree which is not in* Tree_2.

Proof. Consider the following $\varDelta$-regular recursion scheme:

$$G = 1 + a \cdot G \cdot (1+1).$$

The synchronization tree defined by it has a single infinite branch $u_0, u_1, \ldots$ (with edges labelled a). The outdegree of each vertex u_i is $2^i + 1$ since it is the source of 2^i edges labelled ex. By the previous lemma, this synchronization tree is not in Tree_2. □

9.5 A $\varDelta$-algebraic synchronization tree not in Tree_3.

Proposition 16. *There exists a $\varDelta$-algebraic synchronization tree which does not belong to* Tree_3.

Proof. Consider the Δ-algebraic scheme

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

The tree t defined by this scheme has a unique infinite branch whose edges are labelled a. The vertex at depth n along this branch has 2^{2^n} outgoing edges labelled b. Assume toward a contradiction that t belongs to Tree_3. Then there exists a graph in Graph_2 from which t can be obtained by unfolding, contradicting [13, Theorem 4.5.3]. $\square$

9.6 A Γ-algebraic tree not bisimilar to any Δ-regular tree

In this section, our aim is to prove that there is a Γ-algebraic synchronization tree which is not bisimilar to any Δ-regular synchronization tree.

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, $j \in \{0, \ldots, n\}$ are terms over the signature Δ not containing variables. (The empty sum stands for 0.) It is rather standard to prove the following fact:

Lemma 7. *A synchronization tree in $\mathsf{ST}(A)$ is (Γ-)regular iff it can be defined by a right-linear 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 any evaluation of G in $\mathsf{ST}(A)$, the terms $t \cdot G$ and $t'(G)$ evaluate to the same synchronization tree. 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 $t_0'(G)$. $\square$

Proposition 17. *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:

$$\begin{aligned} S &= F(1 + b.1) \\ F(v_1) &= v_1 + a.(F(1 + b.v_1)) \ . \end{aligned}$$

The tree T 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 Lemma 7, this would imply that T is 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 $c \in \{a, b\} \cup \{0, 1\}$.

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 8. *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 8, 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 8 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. $\square$

10 Synchronization trees and logic

10.1 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 7 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 9, is not Δ-algebraic.

Consider a 2-counter machine whose program P is given as a sequence of instructions $I_1, \dots, I_n$ 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 nonnegative 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 $X_1, \dots, X_n$ be set 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 7 is a model of φ_P.

Remark 4. Thomas showed in [37, 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 [35].

10.2 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, 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. To this end, we define $v \sim v'$, for $v, v' \in S(u)$, if the subtrees rooted at v and v' are bisimilar. 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 18. *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-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_0 with edges labelled in $\{a, b, c, d\}$ such that every vertex of t_0 is also the origin of two additional parallel disjoint branches. Since t_0 is deterministic, we may identify each vertex of t_0 with a word $u \in \{a, b, c, d\}^*$. The two additional branches at vertex u of t_0 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_0 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 [40]. 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 uc (resp. ud). Finally, the test for equality between the two counters is performed by testing that vertex u has only one out-going edge labelled e. The details of the construction are similar to those in Section 10.1. □

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.

11 Conclusions

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

17. D. Caucal. On infinite transition graphs having a decidable monadic theory. *Theoretical Computer Science* 290(2003), 79–115.
18. S. Christensen. Decidability and decomposition in process algebras. PhD thesis ECS-LFCS-93-278, Department of Computer Science, University of Edinburgh, 1983.
19. B. Courcelle. A representation of trees by languages I. *Theoretical Computer Science* 6 (1978), 255–279.
20. B. Courcelle. A representation of trees by languages II. *Theoretical Computer Science* 7 (1978), 25–55.
21. B. Courcelle. Fundamental properties of infinite trees, *Theoretical Computer Science* 25 (1983), 69–95.
22. B. Courcelle. Monadic second-order definable graph transductions: A survey. *Theoretical Computer Science*, 126 (1994), 53–75.
23. B. Courcelle and M. Nivat. Algebraic families of interpretations. In *17th Annual Symposium on Foundations of Computer Science*, IEEE Computer Society, 1976, 137–146.
24. 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.
25. 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.
26. 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.
27. I. Guessarian, *Algebraic Semantics.* LNCS 99, Springer, 1981.
28. S. Milius and L. Moss. The category-theoretic solution of recursive program schemes. *Theoretical Computer Science* 366 (2006), 3–59.
29. 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.
30. R. Milner. *A Calculus of Communicating Systems.* LNCS 92, Springer, 1980.
31. R. Milner. *Communication and Concurrency.* Prentice Hall, 1989.
32. F. Moller. Infinite results. Proceedings of CONCUR '96, Concurrency Theory, 7th International Conference, LNCS 1119, pp. 195–216, Springer, 1986
33. M. Nivat, On the interpretation of recursive polyadic program schemes. *Symposia Mathematica* XV (1975), 255 -281.
34. D.M.R. Park. Concurrency and automata on infinite sequences. In Theoretical Computer Science, 5th GI-Conference, LNCS 104, pp. 167–183, Springer, 1981.
35. M.O. Rabin. Decidability of second-order theories and automata on infinite trees. *Transactions of the American Mathematical Society* 141 (1969), 1–35.
36. 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.
37. W. Thomas. A short introduction to infinite automata. In Developments in Language Theory, LNCS 2295, pp. 130–144, Springer, 2001.
38. 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.
39. G. Winskel. Synchronization trees. *Theoretical Computer Science* 34 (1984), no. 1–2, 33–82.

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