Algebraic synchronization trees and processes

Luca Aceto, Arnaud Carayol, Zoltan Esik, Anna Ingolfsdottir

Algebraic Synchronization Trees and Processes*

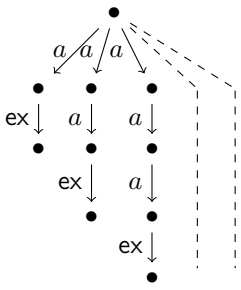
Luca Aceto[1], Arnaud Carayol[2], Zoltán Ésik[3], and Anna Ingólfsdóttir[1]

[1] ICE-TCS, School of Computer Science, Reykjavik University, Iceland
[2] LIGM, Université Paris-Est, CNRS, 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 the Caucal hierarchy.

1 Introduction

The study of recursive program schemes is one of the classic topics in programming language semantics. (See, e.g., [4,16,20,27] for some of the early references.) 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 [24] and of Basic Process Algebra (BPA) [3] as a way of defining possibly infinite synchronization trees [23], which are essentially edge-labelled trees with a distinguished exit label ex. As depicted here, this exit label can only occur on edges whose target is a leaf. 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 pa-



* A full version of this paper may be found at `http://www.ru.is/faculty/luca/PAPERS/algsynch.pdf`. 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. 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.

per, 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. An example of a regular recursion scheme over the signature Δ is

$$X = (X \cdot a) + a, \tag{1}$$

and an example of an algebraic recursion scheme over the signature Γ is

$$F_1 = F_2(a.1), \quad F_2(v) = v + F_2(a.v). \tag{2}$$

The synchronization tree defined by these two schemes is depicted on page 1. In the setting of process algebras such as CCS [24] and ACP [3], synchronization trees are a classic model of process behaviour. They arise as unfoldings of labelled transition systems (LTSs) 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. For example, the tree that is defined by (1) is Δ-regular.

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 expressiveness 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 [19,20], and have been used in [8,9,17] to give semantics to recursion schemes over synchronization trees and words. (We refer the interested reader to [21] 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, [10] 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 up to isomorphism, bisimilarity [24,26] and language equivalence. Moreover, we compare the expressiveness of those recursion schemes to that of the low levels in the Caucal hierarchy. (As a benefit of the comparison with the Caucal hierarchy, we obtain structural properties and decidability of Monadic Second-Order Logic [29].) 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 [18], which coincide with the languages generated by Aho's indexed grammars [2]. We present a pictorial summary of our expressiveness results on Figure 1. Moreover, we prove that the synchronization tree that is the unfolding of the bag (also known as multiset) over a binary alphabet depicted on Figure 2 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 [5].

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 [15,16]. 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 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,

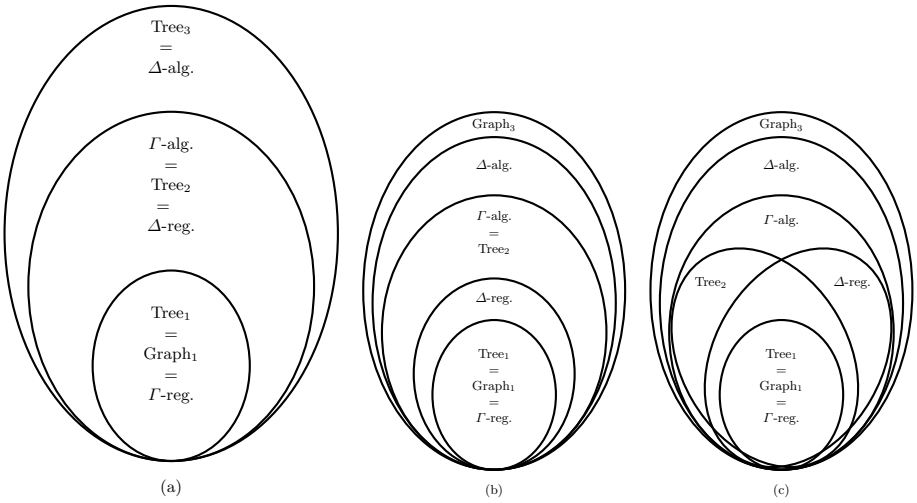


Fig. 1. The expressiveness hierarchies up to language equivalence (a), up to bisimilarity (b) and up to isomorphism (c)

as well as their initial solutions, in Section 4. Section 5 studies the expressive power of regular and algebraic recursion schemes over the signatures Γ and Δ. In Section 6, 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.

2 Continuous Categorical Algebras

In this section, we recall the notion of continuous categorical Σ-algebra. Continuous categorical Σ-algebras were used in [8,9,17] 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

$$\begin{array}{ccc} A^n & \xrightarrow{\sigma^A} & A \\ {\scriptstyle h^n}\downarrow & & \downarrow{\scriptstyle h} \\ B^n & \xrightarrow[\sigma^B]{} & B \end{array}$$

commutes *up to a natural isomorphism* π_σ. 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 $x_0 \stackrel{f_0}{\to} x_1 \stackrel{f_1}{\to} x_2 \stackrel{f_2}{\to} \ldots$ and $y_0 \stackrel{g_0}{\to} y_1 \stackrel{g_1}{\to} y_2 \stackrel{g_2}{\to} \ldots$ are ω-diagrams in A with colimits $(x_k \stackrel{\phi_k}{\to} x)_k$ and $(y_k \stackrel{\psi_k}{\to} y)_k$, respectively, then

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

has colimit $(\sigma^A(x_k, y_k) \stackrel{\sigma^A(\phi_k,\psi_k)}{\to} \sigma^A(x, y))_k$.

A morphism of continuous categorical Σ-algebras is a categorical Σ-algebra morphism that 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.

3 Synchronization Trees

A *synchronization tree* $t = (V, v_0, E, l)$ over an alphabet $\mathcal{A}$ 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" and a 'labelling function' $l : E \to \mathcal{A} \cup \{\mathsf{ex}\}$. 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 *morphism* $\phi : t \to t'$ of synchronization trees is a function $V \to V'$ that 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* [22,26]. 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 [7]. (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 $\mathsf{ST}(\mathcal{A})$ of synchronization trees over $\mathcal{A}$ 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 $a \in \mathcal{A}$.

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 $a \in \mathcal{A}$, we can either associate a constant, or a unary *prefixing operation*. As a constant, a denotes the tree with vertices v_0, v_1, v_2 and two edges: the edge (v_0, v_1), labelled a, and the edge (v_1, v_2), labelled ex. As an operation, $a(t)$ is the tree $a \cdot t$, for any tree t. Let 0 denote the tree with no edges and 1 the tree with a single edge labelled ex. On morphisms, all operations are defined in the expected way. For example, if $h : t \to t'$ and $h' : s \to s'$, then $h + h'$ is the morphism that agrees with h on the 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 $a \in \mathcal{A}$ as a *unary* symbol. In contrast, Δ contains $+, \cdot$, $0, 1$ and each letter $a \in \mathcal{A}$ as a *nullary* symbol. It is known that, for both signatures, $\mathsf{ST}(\mathcal{A})$ is a continuous categorical algebra. See [7] for details.

Two synchronization trees $t = (V, v_0, E, l)$ and $t' = (V', v_0', E', l')$ are *bisimilar* or *bisimulation equivalent* [24,26] 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 $\mathcal{A}^*$ 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 Algebraic Objects and Functors

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

Definition 1. *Let Σ be a signature. A* Σ-recursion scheme*, or* recursion scheme over Σ*, is a sequence E of equations*

$$F_1(v_1, \ldots, v_{k_1}) = t_1, \ldots, F_n(v_1, \ldots, v_{k_n}) = t_n,$$

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, and consider a Σ-recursion scheme of the form given above. 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 $t^A(f_1, \ldots, f_n) = f_i \circ \langle h_1, \ldots, h_{k_i} \rangle$, 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, we have that $t_i^A : A^{r(\Phi)} \to [A^{k_i} \to A]$, 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 $i \in [n]$.

Definition 2. *Suppose that A is a continuous categorical Σ-algebra. We say that $f : A^m \to A$ is* 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.*

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

Example 1. The Δ-regular recursion scheme (1) and the Γ-algebraic one (2) have the infinitely branching tree $\sum_{i \geq 1} a^i$ depicted on page 1 as their initial solutions. That tree is therefore both Δ-regular and Γ-algebraic. So Δ-regular and Γ-algebraic recursion schemes can be used to define infinitely branching trees that have an infinite number of subtrees, even up to language equivalence.

5 Comparing the Expressiveness of Classes of Recursion Schemes

In this section, we interpret recursion schemes over the continuous categorical algebra $\mathsf{ST}(\mathcal{A})$, viewed either as a Γ-algebra or as a Δ-algebra, and study the expressive power of classes of recursion schemes over the signatures Γ and Δ. 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 [7,23], 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. It is well-known that the languages of synchronization trees defined by Γ-regular schemes are the regular languages. On the other hand, modulo language equivalence, Δ-regular schemes are nothing but context-free grammars and have the same expressive power as Γ-algebraic schemes (see Theorem 1(1) and (4) below).

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, [3,5]. 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. (Below, we write BPA for 'BPA with recursion'.) Alternatively, following [25], 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 [14]. We refer the interested readers to [25] for the details of the formal definitions, which are not needed to appreciate the results to follow, and further pointers to the literature.

In the results to follow, we will compare the expressiveness of recursion schemes to that of the low levels in the Caucal hierarchy [12]. For the sake of completeness, following [11], 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). It is well known that Graph_1 is the class of all prefix-recognizable graphs [13].

The following theorem collects our main results on the expressiveness of recursion schemes over the signatures Δ and Γ. A pictorial summary of all our expressiveness results may be found on Figure 1. All the inclusions on that figure are strict, with the possible exception of the inclusion of the collection of the Δ-algebraic trees in Graph_3 up to bisimilarity and up to isomorphism. To the best of our knowledge, it is open whether those inclusions are strict. The fact that the path language of every synchronization tree in Tree_3 (respectively, Tree_2) is an indexed language (respectively, context-free language) is known from [11, Theorem 4].

Theorem 1.

1. *Every Δ-regular tree is Γ-algebraic.*
2. *There is a Γ-algebraic synchronization tree that is not bisimilar to any Δ-regular tree. Moreover, there is a Γ-algebraic synchronization tree that is neither definable in BPA modulo bisimilarity nor in BPP modulo language equivalence.*
3. *Each synchronization tree in Tree_2 is Γ-algebraic, but there is a Δ-regular (and hence Γ-algebraic) synchronization tree that is not in Tree_2.*
4. *Every Γ-algebraic synchronization tree is bisimilar to a tree in Tree_2. Therefore, modulo bisimilarity, the Γ-algebraic synchronization trees coincide with those in Tree_2. Moreover, each Γ-algebraic synchronization tree is language equivalent to a Δ-regular one.*
5. *Each Δ-algebraic synchronization tree is in Graph_3 and hence has a decidable monadic second-order theory. Moreover, there is a Δ-algebraic synchronization tree that does not belong to Tree_3.*
6. *The synchronization tree t_{bag} associated with the bag over a binary alphabet depicted on Figure 2 has an undecidable monadic second-order theory (even without the root being the source of an exit edge). Hence, it is not in the Caucal hierarchy and is therefore not Δ-algebraic, even up to bisimilarity. Moreover, t_{bag} is not Γ-algebraic up to language equivalence.*
7. *There exists a Γ-algebraic synchronization tree whose minimization with respect to bisimilarity does not have a decidable monadic second-order theory and hence is not in the Caucal hierarchy.*

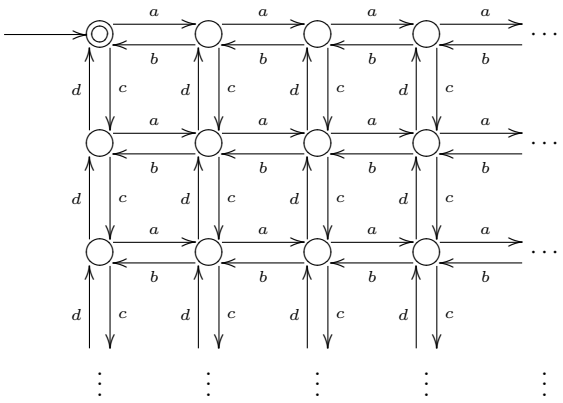


Fig. 2. An LTS whose unfolding is not a Δ-algebraic synchronization tree

Statement 6 in the above theorem is a strengthening of a classic result from the literature on process algebra proved by Bergstra and Klop in [5]. Indeed, in Theorem 4.1 in [5], Bergstra and Klop showed that the bag over a domain

of values that contains at least two elements is not expressible in BPA, and the collection of synchronization trees that are expressible in BPA is strictly included in the Δ-algebraic synchronization trees.

Thomas showed in [28, 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.

Finally, we remark that Theorem 1(7) yields that the collection of synchronization trees in the Caucal hierarchy is *not* closed under quotients with respect to bisimilarity. Indeed, there is a Γ-algebraic tree whose quotient with respect to bisimilarity is not in the Caucal hierarchy. Nevertheless the result is sort of folklore.

6 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 is 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., [16]—and studying the branch languages of bounded synchronization trees. More precisely, we assign a family of branch languages to each bounded synchronization tree over an alphabet $\mathcal{A}$ 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 3. *Suppose that $t = (V, v_0, E, l)$ is a bounded synchronization tree over the alphabet $\mathcal{A}$. Denote by k the maximum of the outdegrees of the vertices of t. Let $\mathcal{B}$ denote the alphabet $\mathcal{A} \times [k]$. A* determinization of t *is a tree $t' = (V, v_0, E, l')$ over the alphabet $\mathcal{B}$ 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 the branch words of t'. Finally, the family of branch languages corresponding to t is:*

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

By way of example, consider the LTS depicted in Figure 2. 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.

1. *A bounded synchronization tree t is algebraic (respectively, regular) if, and only if, $\mathcal{L}(t)$ contains a DCFL (respectively, regular language).*
2. *The bounded synchronization trees in Tree_2 are the bounded Γ-algebraic synchronization trees.*

Using statement 2 in the above theorem, we can show that Figure 1(b) also applies for bounded synchronization trees.

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. For example, consider the following Δ-algebraic scheme:

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

Given any determinization of the synchronization tree t defined by this scheme, the non-context-free 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 t is not Γ-algebraic.

References

1. S. Abramsky. A Domain Equation for Bisimulation. Inf. Comput. 92(2), 161–218 (1991)
2. A.V. Aho. Indexed Grammars — an Extension of Context-Free Grammars. J. ACM 15, 647–671 (1968)
3. J.C.M. Baeten, T. Basten, and M.A. Reniers. Process Algebra: Equational Theories of Communicating Processes. Cambridge University Press (2009)
4. J.W. de Bakker. Recursive Procedures. Mathematical Centre Tracts, No. 24. Mathematisch Centrum, Amsterdam (1971)
5. J.A. Bergstra and J.W. Klop. The Algebra of Recursively Defined Processes and the Algebra of Regular Processes. In: Paredaens, J. (ed.) ICALP 1984, LNCS, vol. 172, pp. 82–94. Springer (1984)

6. S.L. Bloom, Z. Ésik and D. Taubner. Iteration Theories of Synchronization Trees. Inf. Comput. 102(1), 1–55 (1993)
7. S.L. Bloom and Z. Ésik. Iteration Theories. Springer (1993)
8. S.L. Bloom and Z. Ésik. The Equational Theory of Regular Words. Inf. Comput. 197, 55–89 (2005)
9. S.L. Bloom and Z. Ésik. A Mezei-Wright Theorem for Categorical Algebras. Theor. Comput. Sci. 411, 341–359 (2010)
10. F. van Breugel. An Introduction to Metric Semantics: Operational and Denotational Models for Programming and Specification Languages. Theor. Comput. Sci. 258, 1–98 (2001)
11. A. Carayol and S. Wöhrle. The Caucal Hierarchy of Infinite Graphs in Terms of Logic and Higher-order Pushdown Automata. In: Pandya P., and Radhakrishnan, J. (eds.) FSTTCS 03, LNCS, vol. 2914, pp. 112–123. Springer (2003)
12. D. Caucal. On Infinite Terms Having a Decidable Monadic Theory. In: Diks, K., Rytter, W. (eds.) MFCS 02, LNCS, vol. 2420, pp. 165–176. Springer (2002)
13. D. Caucal. On Infinite Transition Graphs Having a Decidable Monadic Theory. Theor. Comput. Sci. 290, 79–115 (2003)
14. S. Christensen. Decidability and decomposition in process algebras. PhD thesis ECS-LFCS-93-278, Department of Computer Science, Univ. of Edinburgh (1983)
15. B. Courcelle. A Representation of Trees by Languages I and II. Theor. Comput. Sci. 6, 255–279, and Theor. Comput. Sci. 7, 25–55 (1978)
16. B. Courcelle. Fundamental Properties of Infinite Trees, Theor. Comput. Sci. 25, 69–95 (1983)
17. Z. Ésik. Continuous Additive Algebras and Injective Simulations of Synchronization Trees. J. Log. Comput. 12, 271–300 (2002)
18. M.J. Fischer. Grammars with Macro-like Productions. In: 9th Annual Symp. Switching and Automata Theory, pp. 131–142. IEEE Press (1968)
19. J.A. Goguen, J.W. Thatcher, E.G. Wagner and J.B. Wright. Initial algebra semantics and continuous algebras. J. ACM 24, 68–95 (1977)
20. I. Guessarian, Algebraic Semantics. LNCS, vol. 99, Springer (1981)
21. S. Milius and L. Moss. The Category-Theoretic Solution of Recursive Program Schemes. Theor. Comput. Sci. 366, 3–59 (2006)
22. R. Milner. An Algebraic Definition of Simulation Between Programs. In: Proceedings 2nd Joint Conference on Artificial Intelligence, pp. 481–489. BCS (1971)
23. R. Milner. A Calculus of Communicating Systems. LNCS, vol. 92, Springer (1980)
24. R. Milner. Communication and Concurrency. Prentice Hall (1989)
25. F. Moller. Infinite Results. In: Montanari, U., Sassone, V. (eds.) CONCUR '96, LNCS, vol. 1119, pp. 195–216. Springer (1986)
26. D.M.R. Park. Concurrency and Automata on Infinite Sequences. In: Deussen, P. (ed.) Theoretical Computer Science, 5th GI-Conference, LNCS, vol. 104, pp. 167–183. Springer (1981)
27. D.S. Scott. The Lattice of Flow Diagrams. In: Symposium on Semantics of Algorithmic Languages 1971, Lecture Notes in Mathematics, vol. 188, pp. 311–366, Springer (1971)
28. W. Thomas. A Short Introduction to Infinite Automata. In: Kuich, W., and Rozenberg, G., Salomaa, A. (eds.) DLT 2001, LNCS, vol. 2295, pp. 130–144. Springer(2001)
29. W. Thomas. Constructing Infinite Graphs with a Decidable MSO-theory. In: Rovan, B., Vojtás, P. (eds.) MFCS 2003, LNCS, vol. 2747, pp. 113–124. Springer (2003)