Abstract

We consider the languages of finite trees called tree-shift languages which are factors of subshifts of infinite trees. We give effective syntactic characterizations of two classes of regular tree-shift languages: the finite type tree languages and the tree languages which are almost of finite type. Each class corresponds to a class of subshifts of trees which is invariant by conjugacy. For this goal, we define a syntactic tree algebra which is stronger than the classical syntactic tree algebra based on contexts. This allows us to capture the notion of constant trees which is essential in the framework of tree-shift languages.

1 Introduction

Infinite $k$-ary trees have a natural structure of symbolic dynamical systems equipped with $k$ shift transformations [1]. The $i$th shift transformation applied to a tree gives the subtree rooted at the child number $i$ of the tree. A tree subshift is described by a set of finite block trees which are forbidden, i.e. which never appear as factor of some infinite tree of the subshift.

The set of trees factors of a subshift is a language of finite ranked trees, called a tree-shift language, which is closed and stable by any shift transformation. This set of factors characterizes the subshift and interesting properties of the subshift can be read in its associated tree-shift language. The simplest class of these languages is the class of tree languages of finite type which corresponds to subshifts defined by a finite set of forbidden block trees. In [1], we proved that the conjugacy of tree subshifts of finite type is decidable, thus extending Willams's conjugacy theorem for one-sided shifts of sequences [11]. A larger class of tree languages is the class of sofic tree languages (also called regular tree-shift languages), which corresponds to sofic subshifts of trees. These tree languages are accepted by tree automata where all states are both initial and final [2]. Among this class, the almost of finite type tree-shift languages have the property of being accepted by a (bottom-up) tree automaton which is both deterministic and co-deterministic with a finite delay. The corresponding class of subshifts constitutes a meaningful intermediate class in between irreducible shifts of finite type and general sofic shift [5]. In [2], we have shown that any irreducible tree subshift has a minimal presentation which is synchronized. We also described an algorithm for checking whether a sofic tree subshift is almost of finite type.

In this paper, we give the syntactic characterizations of tree-shift languages of finite type and of almost of finite type tree-shift languages. A syntactic characterization of almost of finite type word languages have been obtained in [3]. Logics for sofic and
finite type multi-dimensional subshifts has also been explored in [10]. In the framework of trees, we first consider the three sorted syntactic tree algebra of [14] and give characterizations of tree-shift languages and transitive ones in this algebra.

The concept of tree-shift languages of finite type is close to the notion of definite tree languages and forests studied in [8], [12] and [4], to the notion of frontier testable (also called reverse definite) tree languages in [14], and to the notion of generalized definite tree languages [9]. It is more distant from the notion of locally testable tree languages [13]. In [8], Heuter showed that it is decidable whether a regular language is definite. Nivat and Podelski obtained a syntactic characterization of this property in [12].

All these properties are however distinct from the finite type condition. In order to characterize the tree-shift languages of finite type, we introduce the important notion of constant trees known for sequences [7]. We show that the notion of constant tree is not well captured in the syntactic tree algebra of [14]. We thus define a stronger syntactic tree algebra which is computable and finite for regular languages. In this strong algebra, we give effective characterization of tree languages of finite type and tree languages which are almost of finite type. In the last section, we show that this algebra may be refined again while remaining finite and computable for regular languages.

2 Trees and contexts

2.1 Ranked trees

The trees in this paper are finite, labeled and ranked. We will consider only (complete) binary trees (each node has zero or two children) but all results extend to k-ary trees where k is a nonnegative integer. Formally, if Σ denotes the alphabet \{0, 1\} and A is a finite alphabet, a tree is a partial function from Σ* to A with a finite domain which is closed under prefixes. Trees alike will be denoted by the letters s, t, u, ... Labels of nodes will be denoted by a, b, c. We use letters A, B, C to denote the alphabets, i.e. finite sets of labels. We use x, y, z to denote nodes, i.e. words in Σ* included in the domain of some tree. A set \(L\) of trees over a given alphabet A is called a tree language over A.

The one-node tree made with label a is denoted by (a), or simply by a when then is no confusion with the label a. The empty tree is not allowed. If a is a label and s, t are trees, then a(s, t) denotes the tree rooted by a node labeled by a, with left child s and right child t. If we take a tree and replace one of the leaves by a special symbol □ called the hole, we obtain a context. We shall denote by \(T\) and \(C\) the set of trees and contexts respectively. Contexts will be denoted using letters p, q, r. The empty context, where the only node is the hole, is denoted by □. A tree s can be substituted in place of the hole of a context p, the resulting tree is denoted by ps, as illustrated below:

\[
\begin{align*}
\text{p} & \quad \circ \\
\text{s} & \quad \circ \\
\text{ps} & \quad \circ
\end{align*}
\]
We will write $pa$ instead of $p(a)$ for a context $p$ and a letter $a$.

There is a natural composition operation on contexts: the context $qp$ is formed by replacing the hole of $q$ with $p$. This context concatenation satisfies $(pq)s = p(qs)$ for all trees $s$. We also allow constructing contexts from a label $a$, a tree $s$, and context $q$. The resulting contexts are denoted by $a(s, q)$ and $a(q, s)$.

### 2.2 Sofic trees

A (bottom-up) tree automaton (see for instance [6]) works as follows. Fix a finite input alphabet $A$. The automaton has a finite set of states $Q$, a set of initial states $I$, a states of final states $F$, and a finite set of transitions of the form $(q_0, q_1), a \rightarrow q$, where $q_1, q_2, q \in Q$ and $a \in A$. A computation (or a run) of the automaton on a tree $t$ labeled on $A$ is a tree $s$ labeled on $Q$ which is consistent with the transition function in the following sense. Let $x$ be a node of $t$ with label $a$ and children $x_0, x_1$. If the node $x$ is labeled in $s$ and the nodes $x_0, x_1$ are labeled $q_0, q_1$, then there must be a transition $(q_0, q_1), a \rightarrow q$. A computation is accepting if the states labeling the leaves are initial and the state labeling the root is final. The tree $t$ is said to be accepted by the automaton. The set a trees accepted by the automaton is also called the language recognized by the automaton. A tree language is regular if it is recognized by some tree automaton. An automaton also accepts contexts (seen as uncomplete binary trees without their box).

A sofic tree language is a regular tree language accepted by a tree automaton where all states are both initial and final. Such a tree automaton is denoted $(Q, A, \Delta)$, where $\Delta$ is the set of transitions. Note that all its computations are accepting since we have assumed that all states are both initial and final. The full language of trees is the set of all trees of $A$. It is a sofic tree language corresponding to the full tree-shift.

An infinite (binary and complete) tree is a total map from $\Sigma^*$ to $A$. A factor (or subtree) of some (finite or infinite) tree $t$ with domain $D$ is a finite tree $s$ with domain $E$ such that there is a node $x$ of $t$ such that $x + E \subseteq D$ and $s$ and $t$ coincide on the domain of $s$. A factorial language of trees is a language of (finite) trees which is closed by factors. A language of trees $L$ is extendable if and only if, for any tree $s$ in $L$, there is a tree $t$ in $L$ such that, for any node $x$ of $s$, there is a node $y$ of $t$ where $x$ is a prefix of $y$ shorter than $y$. A tree-shift language is a factorial extendable tree language.

It is shown in [2] that a regular set of trees is sofic if and only if it is a regular tree-shift language. It exactly corresponds to the set of factors of sofic shifts of infinite trees.

### 2.3 Syntactic congruence

An equivalent definition of regular trees uses the Myhill-Nerode syntactic congruence (see [14]). Actually there will be three congruences, one for the trees, one for the contexts and one for the labels (seen as labels of internal nodes or as construction operators denoted $a(,)$). Let $L$ be a tree language. Two trees $s, s'$ are called equivalent under $L$, written $s \sim_L s'$, if

$$ps \in L \iff ps' \in L$$

holds for every context $p$. Two contexts $q, q'$ are called equivalent under $L$, written $q \sim_L q'$, if and only if, for any tree $t$, the two trees $qt$ and $q't$ are equivalent under $L$. 

3
When the language \( L \) in question is clear from the context, we omit the subscript \( \sim_L \) and simply write \( \sim \). Two labels \( a, a' \) are called equivalent under \( L \), written \( a \sim_L a' \), if
\[
pa(s, t) \in L \iff pa'(s, t) \in L
\]
holds for every context \( p \) and any trees \( s, t \).

Using standard techniques, one can show that a tree language is regular if and only if its three syntactic equivalences have finite index. Note that \( \sim_L \) on \( A \) has always a finite index since \( A \) is finite.

The syntactic equivalences form a congruence with respect to the operations
\[
a(s, t) \quad a(p, t) \quad a(s, q) \quad pq \quad ps
\]
on trees \( s, t \), contexts \( p, q \) and labels \( a \), as defined above. Note that this congruence is a three-sorted algebra (labels, trees, contexts) called a tree algebra (see [14]).

Two elements (labels, trees, or contexts) are equivalent if they are of the same sort and relate to \( L \) in the same way in every possible context. We respectively denote by \( A, T \) and \( C \) the sets of classes of labels, trees, contexts in the syntactic algebra of \( L \). If \( s \) is a tree, \( p \) a context, \( a \) a label, we will denote by \([s],[p]\) and \([a] \) their class in this syntactic algebra.

We define a partial order \( \leq \) on trees and contexts as follows. If \( s \) is a tree, we define the context of the tree as the set (denoted \( \text{cont}(s) \)) of contexts \( p \) such that \( ps \in L \). Let \( s, t \) be trees. We set \( s \leq t \) if and only if \( \text{cont}(s) \subseteq \text{cont}(t) \). Note that \([s] = [t]\) if and only if \( \text{cont}(s) = \text{cont}(t) \). We thus set \([s] \leq [t]\) if \( \text{cont}(s) \subseteq \text{cont}(t) \). If \([s]\) is a tree class we call context of \([s]\) the set of context classes \([p]\) such that \( ps \in L \).

### 2.4 Tree-shift languages

The goal of the paper is to give algebraic characterizations of symbolic dynamical trees. These characterizations are not purely syntactic but dot-syntactic (i.e. with the use of the zero classes for non full tree-shift languages). They may be used to obtain decidable characterizations of several classes of sofic languages.

We first present the characterization of basic properties of tree languages like being factorial, extendable, or transitive. Hence we give an algebraic characterization of tree-shift languages. The computation of all these properties can be done in the finite syntactic algebra for regular tree languages.

If \( L \) is a tree language, we denote by \( \overline{L} \) the classes images of trees of \( L \) in the tree algebra.

**Proposition 1.** A tree language is factorial if and only if its syntactic algebra satisfies the implication:
\[
va(g, h) \in \overline{L} \Rightarrow va \in \overline{L} \text{ and } a(g, h) \in \overline{L}
\]
for any context class \( v \), any tree classes \( g, h \), and any label class \( a \).

**Proposition 2.** A tree language is extendable if and only if its syntactic algebra satisfies the implication:
\[
va \in \overline{L} \Rightarrow \exists g, h \in T \ va(g, h) \in \overline{L}
\]
for any context class \( v \) and any label class \( a \).
When a tree-shift language (i.e. a factorial extendable tree language) $L$ is not the full language, we denote by 0 the equivalence class of the trees that do not belong to $L$. We also denote by 0 the equivalence class of the contexts $p$ such that, for any tree $s$, $ps \notin L$.

The notion of transitivity suitable for tree-shift languages was introduced in [2]. A finite complete prefix code of $\Sigma^*$ is a prefix-free set $X$ of finite words in $\Sigma^*$ such that any word in $\Sigma^*$ which is longer than the words of $X$ has a prefix in $X$.

**Figure 1:** A transitive tree-shift language. Let $u$ denotes the tree pictured. If $s$ denotes the black block and $t$ the white one, $s$ is a subtree of $u$ rooted at $\epsilon$, and $t$ is a subtree of $u$ rooted at any $x \in X$, where $X$ is the complete prefix code $\{00, 010, 011, 1\}$.

A tree-shift language $L$ is **transitive** if for each pair of trees $s, t$ there is a tree $u$ and a finite complete prefix code $X \subset \Sigma^*$ with words of length at least the height of $s$, such that $s$ is a subtree of $u$ rooted at $\epsilon$, and $t$ is a subtree of $u$ rooted at $x$ for any $x \in X$.

Let $P$ be a subset of a tree language. We denote by $A^\ast(P)$ the set of trees obtained by taking any tree in $T$ and replacing its leaves with some tree in $P$. For a tree class $h$ in $T$, we denote by $A^\ast(h)$ the set of classes of trees in $A^\ast(P)$ where $P$ is the set of trees whose class is $h$.

**Proposition 3.** A tree-shift language is transitive if its syntactic algebra satisfies the following property.

$$v \neq 0, h \neq 0 \Rightarrow \exists g \in A^\ast(h) \text{ such that } vg \neq 0$$

for any context class $v$ and any tree class $h$.

The computation of the above property for regular tree languages is based on the computation of the subset $A^\ast(h)$ of $T$. We set $A^0(h) = \{h\}$, $A(h) = A^0(h) \cup \{a(h, h) | a \in A\}$. For any positive integer $n$, we define $A^n(h) = A(A^{n-1}(h))$. Since the algebra is finite, there is an integer $n$ such that $A^n(h) = A^{n+1}(h)$. This set is equal to $A^\ast(h)$.

Note that Propositions 1, 2, and 3 hold for the full tree languages since the full language is factorial, extendable and transitive.

### 3 Strong syntactic tree algebra

We assume that $L$ is a tree-shift language. The image of trees not belonging to $L$ in the tree algebra is thus 0. We denote by $T^0$ the set of tree classes distinct from 0. We will moreover assume that $L$ is regular in order to work with a finite tree algebra.

Since the equivalence $\sim$ will be too weak to characterize synchronizing properties of tree-shift languages, we introduce the notion of strong syntactic algebra of tree-shift languages. We define a strong equivalence on trees, denoted $\approx$, as follows.

---

\[1\] i.e. no word is prefix of another one.
Let $s$ be a tree with $\ell$ leaves. Let $k = 2^\ell$ and $s_1, \ldots, s_k$ be $k$ trees. We denote by $s(s_1, \ldots, s_k)$ the tree obtained from $s$ by replacing the $j$th leaf labeled $a_j$ with $a_j(s_{2j-1}, s_j)$.

Two trees $s, t$ are called strong equivalent under $L$, written $s \approx t$ if and only if

$$\forall s_1, \ldots, s_k \; \exists t_1, \ldots, t_r \; \left( s(s_1, \ldots, s_k) = [t(t_1, \ldots, t_r)] \right), \tag{3}$$

$$\forall t_1, \ldots, t_r \; \exists s_1, \ldots, s_k \; \left( t(t_1, \ldots, t_r) = [s(s_1, \ldots, s_k)] \right). \tag{4}$$

We denote by $[s]$ the class of $s$ for $\approx$. We write $[s] \leq [t]$ when Condition 3 above holds. Note that this property is independent of the choice of the representant $s_i$ or $t_j$ in each class.

Two contexts $p, q$ are called strong equivalent under $L$, written $p \approx q$, if and only if, for any tree $t$, the two trees $qt$ and $q't$ are strong equivalent under $L$. Finally, the strong equivalence coincide with the equivalence $\approx$ for labels. If $e$ is a tree or a context, we denote by $[e]$ the class of $e$. The null class is still denoted by $0$. The strong context classes form a finite monoid.

One can easily check the following properties.

$$s \approx s' \implies ps \approx ps', \tag{5}$$

$$a \approx a', s \approx s', t \approx t' \implies a(s, t) \approx a(s', t'), \tag{6}$$

$$a \approx a', p \approx p', t \approx t' \implies a(p, t) \approx a(p', t'), \tag{7}$$

$$a \approx a', s \approx s', q \approx q' \implies a(s, q) \approx a(s', q'), \tag{8}$$

$$p \approx p', q \approx q' \implies pq \approx p'q', \tag{9}$$

$$p \approx p', s \approx s' \implies ps \approx p's'. \tag{10}$$

We respectively denote by $A$, $\mathbb{T}$ and $\mathbb{C}$ the sets of classes of labels, trees, contexts in this three-sorted strong syntactic algebra of $L$.

Since $s \approx s'$ implies $ps \approx ps'$, the concatenation of a context to a tree transforms a strong tree class into a strong tree class.

We define two special automata accepting tree-shift languages. The first one, the context tree automaton, is a deterministic tree automaton whose states are identified with context sets. All its states are both initial and final. The second one, called the determinized context tree automaton, has a unique initial state and is obtained by determinization of the context tree automaton.

The context tree automaton of a regular tree-shift language $L$ is the deterministic automaton $(T^0, A, \Delta)$, where $T^0$ is set of nonnull tree classes of $L$. The transitions of $\Delta$ are $([s], [t]), a \rightarrow [a(s, t)]$, where $s, t$ are trees and $[a(s, t)]$ is nonnull. Since $L$ is extendable, it recognizes the language $L$.

The determinized context tree automaton is the deterministic tree automaton $(\mathfrak{P}(T^0), A, \delta, I, F)$ whose set of states are the parts of $T^0$, with a unique initial state $i = T^0$, $F = \mathfrak{P}(T^0)$, and with labels $(P, Q), a \rightarrow \{a(g, h) \mid g \in P, h \in Q, a(g, h) \neq 0\}$. It accepts the language $L$. Since $T$ is finite, $\mathfrak{P}(T^0)$ is finite. We denote $\delta(i, s)$ by $I(s)$. Note that $I(s)$ is non empty when $s \in L$ since $L$ is extendable.

**Proposition 4.** Let $s, t$ be two trees. We have $[s] \leq [t]$ if and only if $I(s) \subseteq I(t)$ and thus $[s] = [t]$ if and only if $I(s) = I(t)$.

**Proof.** Let us assume that $I(s) \subseteq I(t)$. By definition, for any trees $s_1, \ldots, s_k$ such that $s(s_1, \ldots, s_k) \neq 0$, we have $[s(s_1, \ldots, s_k)] \in I(s) \subseteq I(t)$. Hence there are trees $t_1, \ldots, t_r$ such that $[t(t_1, \ldots, t_r)] = [s(s_1, \ldots, s_k)]$.
Conversely, if \( h \in I(s) \), then \( h = [s(s_1, \ldots, s_k)] \) for some trees \( s_1, \ldots, s_k \). If there are trees \( t_1, \ldots, t_r \) such that \([t(t_1, \ldots, t_r)] = [s(s_1, \ldots, s_k)]\), then \( h \in I(t) \) implying \( I(s) \subseteq I(t) \).

As a consequence, \([s] = [t]\) implies \([s] = [t]\).

**Corollary 5.** The strong syntactic algebra of a tree-shift language has finite index if and only if the language is regular.

**Proof.** The strong equivalence is finer that the equivalence \( \sim \). Thus it has an infinite number of classes when the language \( L \) is not regular. Conversely, if \( L \) is regular, the number of strong tree classes is bounded above by the number of states of the determinized context tree automaton.

We define below the subset \( T^\omega \) of \( T \) which represents the tree classes of block trees of any height (or of infinite complete trees). Let \( n \) be a positive integer. A **block tree** of height \( n \) is a tree whose domain is the set of all words of \( \Sigma^* \) of length at most \( n - 1 \). We denote by \( T_n \) the set of block trees of height \( n \) and by \( T_n^\omega \) the set of strong equivalence classes of trees in \( T_n \). There is an integer \( m \) such that \( T_m = T_{m+1} \). It follows that \( T_m = T_n \) for any integer \( n \geq m \). We denote this set by \( T^\omega \).

### 4 Tree-shift languages of finite type

#### 4.1 Constant trees

In this section, we introduce the notion of constant tree (or intrinsically synchronizing tree). It corresponds to the notion of constant of a semigroup which is used to capture the synchronization properties of word languages (see [7]), or to the notion of intrinsically synchronizing word of symbolic dynamical systems [11, exercice 3.3.4 p. 85].

A tree \( s \) is a **constant** for \( P \subseteq L \) if the following implication holds.

\[
\begin{align*}
ps(s_1, \ldots, s_k) &\in P, \\
qs(s'_1, \ldots, s'_k) &\in P,
\end{align*}
\Rightarrow
\begin{align*}
ps(s'_1, \ldots, s'_k) &\in P, \\
qs(s_1, \ldots, s_k) &\in P,
\end{align*}
\]

for any contexts \( p, q \) and any trees \( s_i, s'_i \). A tree is a **constant** if it is a constant for \( L \).

![Figure 2: A constant tree s.](image)

Thus, by definition, \( s \) is constant if and only if either \([s] = 0\) or \([s(s_1, \ldots, s_k)] = [s]\) for any trees \( s_i \) such that \([s(s_1, \ldots, s_k)] \neq 0\). Similarly, a context \( p \) is a **constant** if and only if \([pa(s_1, \ldots, s_k)] = [pb]\) for any letters \( a, b \) and any trees \( s_i \) such that \([pa(s_1, \ldots, s_k)] \neq 0\) and \([pb] \neq 0\).

Note that some tree classes may contain both constant and non constant trees. The following proposition gives a characterization of constant trees.
Proposition 6. Let $s$ be a tree. Then $s$ is constant if and only if either $s \notin L$ or $\text{card}(I(s)) = 1$.

Proof. Let $s$ be a tree of $L$ which is a constant. Since $L$ is extendable, $I(s)$ is non empty. Let us assume that $g, h \in I(s)$ and $g \neq h$. We have $g, h \neq 0$. Let $p$ be a context such that $[p]g \neq 0$ and $[p]h = 0$. Thus there are trees $s_1, \ldots, s_k, s'_1, \ldots, s'_k$ such that $g = [s(s_1, \ldots, s_k)]$ and $h = [s(s'_1, \ldots, s'_k)]$. We get $[ps(s_1, \ldots, s_k)] \neq 0$ and $[qs(s'_1, \ldots, s'_k)] \neq 0$.

As $s$ is constant, we obtain $[ps(s'_1, \ldots, s'_k)] \neq 0$ which is a contradiction.

Conversely, let us assume that $s$ is a tree in $L$ such that $\text{card}(I(s)) = 1$. Let $p, q$ be contexts and $s_1, s'_1$ be trees such that $ps(s_1, \ldots, s_k) \neq 0$ and $qs(s'_1, \ldots, s'_k) \neq 0$. Then $s(s_1, \ldots, s_k) \neq 0$, $s(s'_1, \ldots, s'_k) \neq 0$, and $I(s)$ contains the classes $[s(s_1, \ldots, s_k)]$ and $[s(s'_1, \ldots, s'_k)]$. By hypothesis $[s(s_1, \ldots, s_k)] = [s(s'_1, \ldots, s'_k)]$ and finally $ps(s_1, \ldots, s_k) \neq 0$, $qs(s_1, \ldots, s_k) \neq 0$.

We define the notion of constant classes in the weak and strong algebras as follows. A constant class of $T$ (resp. $C$) is a class for the equivalence $\sim$ containing only constant trees (resp. constant contexts). A constant class of $T$ (resp. $C$) is a class for the equivalence $\approx$ containing only constant trees (resp. constant contexts).

As a consequence of Proposition 6, if $s, t$ are trees and $[s] = [t]$, then $s$ is a constant if and only if $t$ is constant.

Corollary 7. Let $[s]$ be a strong tree class. Then $[s]$ is constant if and only $[s] = 0$ or $\text{card}(I(s)) = 1$.

A 0-minimal class (for the equivalence $\sim$) of trees is a nonnull class which is minimal for this partial order $\leq$. The following proposition proves the existence of nonnull constant tree classes.

Proposition 8. Any (weak) 0-minimal tree class is constant.

Proof. Let us assume that $h$ is a nonnull minimal tree class for the equivalence $\sim$. We show that any tree of this class is constant.

If $h$ is not constant, let $s$ in $h$ which is not constant. Let $h_1, h_2$ in $I(s)$ with $h_1 \neq h_2$. Since $h_1 \in I(s)$, $h_1 \leq h$ and $h_1 \neq 0$. Let us assume that $h_1 \neq [s]$. Then there is a context class $v$ such that $vh_1 = 0$ and $vh \neq 0$. Hence $h_1 < h$, a contradiction. □

Proposition 9 (proof omitted). Any strong tree class $[s]$ is constant if and only if $[s(s_1, \ldots, s_k)] = [s]$ whenever $s(s_1, \ldots, s_k) \neq 0$.

4.2 Characterization of shifts of finite type

A tree-shift language is of finite type if it is defined by a finite set of forbidden factors, i.e. there is a finite set of trees $F$ such that a tree belongs to the language if and only if it does not contain any tree in $F$ as factors. A tree-shift language is of finite type if there is an integer $m$ such that any block tree of height $m$ is a constant tree.

A tree-shift language of finite type is regular and a regular tree-shift language is of finite type if and only if it is recognized by a deterministic local automaton where all states are initial and final (see [2]). The locality of an automaton is defined as follows. A deterministic $m$-local tree automaton is a tree automaton $A$ such that any two computations of $A$ on a same block tree of height $m$ end in the same state. A tree
automaton is local (or definite) if it is \( m \)-local for some nonnegative integer \( m \) (and \( m \) stands for memory).

The notion of tree-shift languages of finite type is close to the notion of definite languages of trees and forests given in [8], [12], and [4], for which the membership depends only on the nodes of height at most \( n \). It is however different and a syntactic characterization of finite type tree-shift languages cannot be obtained in the syntactic tree algebra\(^2\) but in the strong syntactic tree algebra.

**Theorem 10.** A regular tree-shift language is of finite type if and only if the following property holds in the strong syntactic tree algebra,

\[
[s] \in T^\omega \Rightarrow [s] \text{ is constant.}
\]

**Proof.** Let \( L \) be a regular tree-shift language of finite type. There is an integer \( m \) such that every block tree of height \( m \) is constant. Let \([s] \in T^\omega \) which is nonnull. There is a block tree \( t \) in \( L \) of height greater than \( m \) which belongs to \([s]\). By Proposition 6, \( \text{card} I(t) = 1 \) and thus \([s]\) is a constant strong tree class.

Conversely, let us assume that \( L \) is a regular tree-shift language such that any strong tree class of \( T^\omega \) is constant. Let \( m \) be an integer such that \( T^\omega = T_m \). Let \( s \) be a block tree of height \( m \). Then \([s] \in T_m = T^\omega \). It follows that \([s]\) is constant and thus \( s \) is a constant tree.

**Example 1.** In Figure 3, Figure 2 is pictured a tree of a tree-shift language on the alphabet \( \{a, b\} \). The language is the set of trees containing an odd number of \( a \) between two \( b \) on any path in the tree. This tree-shift is not of finite type.

![Figure 3: An tree-shift language which is not of finite type and the computation of its class in its syntactic tree algebra. Factor trees containing an even number of \( a \) between two \( b \) on some path in the tree are forbidden.](image)

We have

\[
T = \{[a], [b], [b(a, a)], [b(b(a, a), b(a, a))], 0\},
\]

\[
T = \{[a], [b], [b(a, a)], [b(b(a, a), b(a, a))], 0\},
\]

\[
T^\omega = T.
\]

The strong tree class \([b]\) is not a constant class. Indeed, \([b(a, a)] \neq [b(b(a, a), b(a, a))]\).

As a consequence the language \( L \) is not of finite type.

\(^2\)In particular the condition \((v^g \neq 0, v^h \neq 0) \Rightarrow (v^g = v^h)\) is sufficient but not necessary for a tree-shift language to be of finite type.
5 Almost of finite type tree-shift languages

In this section, we define the notion of almost of finite type tree languages. These languages are regular transitive tree-shift languages recognized by a tree automaton which is both deterministic and co-deterministic with a finite delay. They correspond to a class of symbolic dynamical tree-shifts which is in between tree-shifts of finite type and sofic tree shifts. The class of almost finite type tree-shift languages strictly contains transitive tree-shift languages of finite type and is strictly smaller than the class of regular tree-shift languages.

A tree automaton is irreducible if for each pair of states $p,q$, there is a finite complete prefix code $X$ of $\Sigma^*$ and a finite computation $c$ of the automaton on a tree such that $c_x = p$ and $c_x = q$ for each $x \in X$. We say that a tree $s$ (or a context $p$) is a synchronizing tree (or context) of a tree automaton if all computations of the automaton on $s$ end at the same state. A synchronizing tree of the context tree automaton is a constant tree and conversely. A tree automaton $A = (V,A,\Delta)$ is co-deterministic (or left-closing) with delay $m$ if any two computations of $A$ on a same block tree of height $m + 1$ and ending in a same state, are equal at the nodes $x = 0$ and $x = 1$.

It is proved in [2] that there is a unique minimal deterministic irreducible and synchronized tree automaton accepting a transitive regular tree-shift language. It is equal to the unique irreducible component of the context tree automaton, called the Shannon cover\(^3\) of the tree language. Moreover a regular transitive tree-shift language is almost of finite type if and only its Shannon cover is co-deterministic with a finite delay.

**Theorem 11.** A regular transitive tree-shift language is almost of finite type if and only if the following property holds in its strong syntactic algebra.

$$\left\{ \begin{array}{c} [s] \leq [t], [s'] \leq [t] \\ \text{and } [ps] \neq 0, [ps'] \neq 0, \end{array} \right. \Rightarrow [s] = [s'], \quad (11)$$

where $[s]$, $[s']$ are constant strong tree classes, $[p]$ is a constant strong context class, and $[t]$ is a strong context class of $T^\omega$.

**Proof.** Let us assume that $L$ is an almost of finite type language. We denote by $S$ the unique irreducible component of the context automaton of $L$. There is an integer $m$ such that the automaton $S$ is co-deterministic with delay $m$.

Let $[s]$, $[s']$ be constant strong tree classes, $[t]$ be a strong tree class of $T^\omega$, $[p]$ be a strong constant context class, with $[s], [s'] \leq [t]$ and $[ps], [ps'] \neq 0$. These conditions imply $[pt] \neq 0$. Since $[s]$, $[s']$ are nonnull constant classes, $I(s) = \{h\}$ and $I(s') = \{h'\}$, where $h$ and $h'$ are states of the context automaton $S$. Since $[t] \in T^\omega$, there is a block tree $u$ of height greater than $m$ such that $[u] = [t]$. We have $h, h' \in I(u)$. Let $c$ (resp. $c'$) be a computation of $S$ on $u$ ending in $h$ (resp. $h'$). Let us assume that $p$ has a symbol $\Box$ at the node $x$. Since $[ps] \neq 0$, there is a computation $d$ (resp. $d'$) of $S$ on $pu$ such that the label of $d$ at the node $x$ is $h$ (resp. $h'$). Since the context $p$ is constant, these two computations $d$ and $d'$ end in a same state. The irreducible tree automaton $S$ being co-deterministic with delay $m$, we obtain that $h = h'$ and thus $[s] = [s']$.

Conversely, let us assume that the Property 11 is satisfied. Let $m$ be an integer such that $T^\omega = T_m$. Let $t = a(t_0,t_1)$ be a block tree of height $m + 1$ such that there

---

\(^3\)also called the right Fischer cover.
are two computations c and c′ of S on t ending in a same state. The strong tree classes [t₀] and [t₁] belong to Tₘ = Tw.

Let us assume that h₀ (resp. h₀′) is the label of c (resp. c′) at the node 0 while h₁ (resp. h₁′) is the label of c (resp. c′) at the node 1. Since h₀, h₀′, h₁, h₁′ are states of S, and since S is irreducible, there are constant trees s₀, s₀′, s₁, s₁′ such that [s₀] = h₀, [s₀′] = h₀′, [s₁] = h₁, [s₁′] = h₁′, for i = 0, 1. We have [s₀] ≤ [t₀], [s₀′] ≤ [t₀] (resp. [s₁] ≤ [t₁], [s₁′] ≤ [t₁]). Since S is irreducible there is a constant context p such that pt is nonnull. Let q = pa(□, t₁) and r = pa(t₀, □). As a consequence [qs₀] ≠ 0, [qs₀′] ≠ 0 (resp. [rs₁] ≠ 0, [rs₁′] ≠ 0). By hypothesis, we get [s₀] = [s₀′] and [s₁] = [s₁′]. Hence h₀ = h₀′ and h₁ = h₁′, showing that S is co-deterministic with delay m and L is almost of finite type.

Example 2. We consider again the language of trees containing an odd number of a between two b on any path in the tree. We have T = Tω = {[[a], [b]], [[b(a, a)], [b(b(a, b), b(a, a))], 0}. The nonnull constant strong tree classes are

\[ [s₁] = [[a]], \quad [s₂] = [[b(a, a)]], \quad [s₃] = [[b(b(a, a), b(a, a))]]. \]

The nonnull constant strong context classes are [[a(□, a)], [b(□, a)], and [b(b(□, a), b(a, a))]]. Since there is nothing to check when [t] is a constant strong tree class, we consider only the case where [t] = [b]. If [p] = [[a(□, a)]], then [ps₃] = 0. If [p] = [a(□, a)], then [ps₂] = 0. If [p] = [b(b(□, a), b(a, a))], then [ps₂] = 0. As a consequence, the language is almost of finite type.

6 Strong syntactic tree algebra of order two

In this final section, we show that an even stronger finite syntactic algebra, called the strong syntactic algebra of order two, can be defined for regular tree-shift languages. A new equivalence, denoted ≈₂, refines the strong equivalence as follows.

Let s be a tree with a set of leaves (tᵢ)ᵢ∈Iₗ. For any subset Jₗ of Jₛ, the tree obtained from s by replacing any leaf ℓₖ labeled aⱼ with j ∈ Jₛ by aⱼ(sⱼ, sⱼ′) is denoted s((sⱼ, sⱼ′)j∈Jₛ). Note that when [uⱼ] = [sⱼ] and [uⱼ′] = [sⱼ′], [s((sⱼ, sⱼ′)j∈Jₛ)] = [s((uⱼ, uⱼ′)j∈Jₛ)].

Let s, t be two trees. We write \( s \leq_L t \) if and only if

\[ \forall (sⱼ, sⱼ′)j∈Jₛ \quad \exists (tⱼ, tⱼ′)j∈Jₗ \quad [t((tⱼ, tⱼ′)j∈Jₗ)] = [((sⱼ, sⱼ′)j∈Jₛ)]. \]  \hspace{1cm} (12)

Two trees s, t, are called equivalent in the strong syntactic algebra of order two, written s ≈₂ t, if s ≤ₗ t and t ≤ₗ s. Thus s ≈₂ t implies s ≈ t but the converse is not true. The equivalence ≈₂ is thus finer than ≈. We denote by [s]₂ the class of s for ≈₂. We write \([s]₂ \leq_L [t]₂\) when s ≤ₗ t. This property is independent of the choice of the representant. We have s ≤ₗ t ⇒ [s] ≤ [t] ⇒ [s] ≤ [t] for any tree t.

Two contexts p, q are called equivalent in the strong syntactic algebra of order two, written p ≈₂ q, if and only if, for any tree t, the two trees qt and qt are equivalent for the strong equivalence of order two. Properties similar to the ones of Equations 5 to 10 are satisfied.

Corollary 12 (proof omitted). The strong syntactic algebra of order two of a tree-shift language has finite index if and only if the language is regular.
A tree class in the strong algebra of order two tries to approximate the analog of an $L$-class in the syntactic semigroup of a word language. A relation $R$ may be defined as follows. We say that $s \leq_R t$ if there is a context $p$ such that $s \approx_2 pt$ and $s \sim_R t$ if $s \leq_R t$ and $t \leq_R s$. Since $s \approx_2 s'$ implies $ps \approx_2 ps'$, the concatenation of a context to a tree transforms a strong tree class of order two into a strong tree class of order two.

References


