Sofic and Almost of Finite Type Tree-Shifts

Nathalie Aubrun, Marie-Pierre Béal

Sofic and almost of finite type tree-shifts

Nathalie Aubrun and Marie-Pierre Béal

Université Paris-Est
Laboratoire d'informatique Gaspard-Monge, CNRS

Abstract. We introduce the notion of sofic tree-shifts which corresponds to symbolic dynamical systems of infinite trees accepted by finite tree automata. We show that, contrary to shifts of infinite sequences, there is no unique minimal deterministic irreducible tree automaton accepting an irreducible sofic tree-shift, but that there is a unique synchronized one, called the Shannon cover of the tree-shift. We define the notion of almost finite type tree-shift which is a meaningful intermediate dynamical class in between irreducible finite type tree-shifts and irreducible sofic tree-shifts. We characterize the Shannon cover of an almost finite type tree-shift and we design an algorithm to check whether a sofic tree-shift is almost of finite type.

1 Introduction

In a previous article [1], we introduced the notion of tree-shifts of finite type defined as sets of infinite trees avoiding a finite number of forbidden patterns. Infinite trees have a natural structure of one-sided symbolic systems equipped with several shift transformations. The ith shift transformation applied to a tree gives the subtree rooted at the child number i of the tree. Tree-shifts are highly interesting to study as they constitute an intermediate class between one-sided shifts of infinite sequences and multidimensional shifts.

The conjugacy of multidimensional shifts of finite type, also called textile systems or tiling systems (see for instance [11],[14],[8],[6]), is undecidable. However, the conjugacy of (one-sided) shift spaces of finite type of infinite sequences is decidable ([19], see also [12]). In [1], we extended William's result to trees, showing that the conjugacy of irreducible tree-shifts of finite type is decidable.

In this paper, we focus on sofic tree-shifts, which are shifts of infinite trees accepted by finite (bottom-up) tree automata, and thus whose set of patterns is a recognizable set of finite trees. The goal is to extend to trees some results of sofic shifts of infinite sequences, to define a hierarchy a sofic tree-shifts and characterize each level of this hierarchy.

We introduce the notion of irreducible sofic tree-shifts. We show, that, unlike for sofic shifts of sequences, an irreducible sofic tree-shift may be accepted by several minimal deterministic irreducible tree automata. This is due to the lack of a synchronizing block, even in minimal irreducible automata. We introduce the notion of synchronized tree automaton and the notion of Shannon cover of

an irreducible sofic tree-shift. We prove that the Shannon cover is the unique minimal synchronized tree automaton accepting an irreducible sofic tree-shift.

The existence of the Shannon cover allows us to introduce the class of almost finite type tree-shifts, which extends the known notion of almost of finite type shifts of sequences. The almost of finite type concept was introduced by Marcus in [13] for coding purposes. The class of almost of finite type shifts is a meaningful class of shifts between irreducible shifts of finite type and irreducible sofic shifts (see [4], [20], [9], [3], [2]). The class contains strictly the class of irreducible shifts of finite type and is strictly contained in the class of sofic shifts. The class is stable by conjugacy and it is also invariant by a flow equivalence [7]. We characterize the Shannon cover of an almost of finite type tree-shift and design an algorithm to check whether a sofic tree-shift is almost of finite type.

The paper is organized as follows. In Section 2.1 and Section 2.2, we give basic definitions about tree-shifts and conjugacies. In Section 3, we define the notion of automaton accepting a tree-shift. We refer to [5], [18], [15] for more general trees and automata on finite and infinite trees. The notion of Shannon cover is introduced in Section 3.4. The characterization of almost of finite type tree-shifts is done in Section 4. In the algorithmic issue, Section 5, we give a construction of the Shannon cover of a sofic tree-shift. We design a polynomial-time algorithm to check whether a sofic tree-shift given by its Shannon cover is almost of finite type. Some proofs are omitted in this version of the paper.

2 Definitions

2.1 Tree-shifts

We first recall some basic definitions of symbolic dynamics on infinite trees (see [1] for more details). We consider infinite trees whose nodes have a fixed number of children and are labeled in a finite alphabet. We restrict to binary trees, but all result extend to the case of trees with d children for all $d \geq 1$.

Let $\Sigma = \{0, 1\}$. An *infinite tree* t over a finite alphabet A is a complete function from Σ^* to A. Unless otherwise stated, a tree is an infinite tree. A node of a tree is a word of Σ^*. The empty word, that corresponds to the root of the tree, is denoted by ϵ. If x is a node, its children are xi with $i \in \Sigma$. Let t be a tree and let x be a node, we shall denote $t(x)$ by t_x. When Σ is fixed, we denote by $\mathcal{T}(A)$ the set of all infinite trees on A, hence the set A^{Σ^*}.

We define the shift transformations σ_i for $i \in \Sigma$ from $\mathcal{T}(A)$ to itself as follows. If t is a tree, $\sigma_i(t)$ is the tree rooted at the i-th child of t, i.e. $\sigma_i(t)_x = t_{ix}$ for all $x \in \Sigma^*$. The set $\mathcal{T}(A)$ equipped with the shift transformations σ_i is called the *full shift* of infinite trees over A. A sequence of words $(x_k)_{k\geq 0}$ of Σ^* is called a *path* if for all $k, x_{k+1} = x_k i_k$ with $i_k \in \Sigma$.

A *pattern* is a function $p : L \to A$, where L is a finite prefix-closed[1] subset of Σ^*. The set L is called the *support of the pattern*. A *block of height* n is a pattern with support $\Sigma^{\leq n}$, where n is some nonnegative integer, and $\Sigma^{\leq n}$

[1] each prefix of L belongs to L.

denotes the words of length at most n of letters of Σ. The *height* of a block u is denoted by height(u). A *leaf* of a pattern is a node with no child.

We say that a block u of support L is a *block of a tree* t if there is a word $x \in \Sigma^*$ such that $t_{xy} = u_y$ for all words $y \in \Sigma^*$. We say that u is a block of t rooted at the node x. If u is not a block of t, one says that t *avoids* p.

We define a *tree-shift space* (or *tree-shift*) X of $\mathcal{T}(A)$ as the set $\mathsf{X}_{\mathcal{F}}$ of all trees avoiding each element of a set of blocks $\mathcal{F}$. If the set of trees on A is equipped with the usual product topology, where the topology in A is the discrete one, a tree-shift space is closed and invariant for any shift transformation σ_i. A *tree-shift of finite type* (SFT) X of $\mathcal{T}(A)$ is a set $\mathsf{X}_{\mathcal{F}}$ of all trees avoiding each block of a *finite* set of blocks $\mathcal{F}$. The set $\mathcal{F}$ is called *a set of forbidden blocks* of X.

We denote by $\mathcal{L}(X)$ the set of patterns of all trees of the tree-shift X, by $\mathcal{B}(X)$ the set of all blocks of X and by $\mathcal{B}_n(X)$ the set of all blocks of height n of X. If u is a block of height n with $n \geq 1$, we denote by $\sigma_i(u)$ the block of height $n-1$ such that $\sigma_i(u)_x = b_{ix}$ for $x \in \Sigma^{\leq n-1}$. The block u is written $u = (u_\varepsilon, \sigma_0(u), \sigma_1(u))$.

Example 1. In Figure 1 is pictured an infinite tree of a tree-shift X on the alphabet $\{a, b\}$. The forbidden blocks are those containing an even number of a between two b on any path in the tree. This tree-shift is not of finite type.

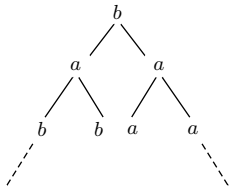


Fig. 1. An infinite tree of the tree-shift $\mathsf{X}_{\mathcal{F}}$, where $\mathcal{F}$ is the set of patterns containing an even number of a between two b on any path in the tree.

2.2 Block maps and conjugacies

Let A, A' be two finite alphabets, X be a tree-shift of $\mathcal{T}(A)$ and m be a positive integer. A map $\Phi : X \subseteq \mathcal{T}(A) \to \mathcal{T}(A')$ is called an *m-local map* (or an *m-block map*) if there exists a function $\phi : \mathcal{B}_m(X) \to A'$ such that, for all $x \in \Sigma^*$, $\Phi(t)_x = \phi(t_{x\Sigma^{\leq m-1}})$, where $t_{x\Sigma^{\leq m-1}}$ is the block q such that $q_y = t_{xy}$ for all $y \in \Sigma^{\leq m-1}$. The smallest integer $m-1$ such that Φ is an m-block map, is called the *memory* of the block map. A *block map* is a map which is an m-block map for some nonnegative integer m.

The image of X by a block map is also a tree-shift, and is called a *factor* of X. A one-to-one and onto block map from a tree-shift X onto a tree-shift Y has an inverse which is also a block map, as for shifts of sequences. It is called a

conjugacy from X onto Y. The tree-shifts X and Y are then *conjugate*. We call *sofic* a tree-shift which is a factor of a tree-shift of finite type.

Let X be a tree-shift and m a positive integer. We denote by $X^{(m)}$ the *higher block presentation of* X. It is a tree-shift on the alphabet $\mathcal{B}_m(X)$. For each tree t in $X^{(m)}$, there is tree t' in X such that, for each node x, t_x is the block of height m of t' rooted at x. The shifts X and $X^{(m)}$ are conjugate (see [1]).

3 Sofic tree-shifts

3.1 Tree automata

In this section we consider bottom-up automata of infinite trees. Such an automaton starts its computation from the infinite branches and moves upward. A *tree automaton* is here a structure $\mathcal{A} = (V, A, \Delta)$ where V is a finite set of states, A is a finite set of input symbols, and Δ is a set of transitions of the form $(q_0, q_1), a \to q$, with $q, q_i \in V$, $a \in A$. A transition $(q_0, q_1), a \to q$ is called a transition *labeled by* a, *going out of* the pair of states (q_0, q_1) and *coming in* the state q. A transition $(q_0, q_1), a \to q$ will be pictured by

Note that no initial nor final states are specified. This means that all states are both initial and final.

Such an automaton is *deterministic* if for each pair of states (q_0, q_1) and for each $a \in A$, there is at most one transition $(q_0, q_1), a \to q$. Then the set of transitions defines a partial function δ from $V^2 \times A$ to V.

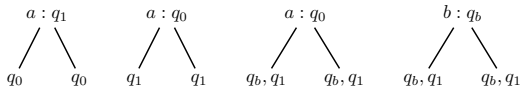
A (bottom-up) *computation* of $\mathcal{A}$ on the infinite tree t is an infinite tree C on V such that, for each node x, there is a transition $(C_{x0}, C_{x1}), t_x \to C_x \in \Delta$. A tree t is *accepted* by $\mathcal{A}$ if there exists a computation of $\mathcal{A}$ on t. The set of infinite trees accepted by $\mathcal{A}$ is a tree-shift. Given a tree automaton $\mathcal{A}$, it is always possible to transform it into a deterministic tree automaton which accepts the same set of trees (this process is called determinization, see for instance [5] for details). In the sequel, we assume that all states of an automaton are *accessible*, i.e. each state ends some computation of the automaton.

A (bottom-up) *finite computation* of $\mathcal{A}$ on the complete finite tree t is a complete finite tree C on V such that, for each node x which is not a leaf, there is a transition $(C_{x0}, C_{x1}), t_x \to C_x \in \Delta$.

A tree automaton is called *an edge tree automaton* is all transitions have distinct labels. An *edge tree-shift* is a tree-shift (of finite type) accepted by an edge tree automaton.

Example 2. We define a tree automaton $\mathcal{A}$ with three state q_b, q_0 and q_1 which accepts the tree-shift X of Example 1. The two states q_0 and q_1 only label nodes with an a, and they control the parity of the number of a encountered from any

last b below. The state q_b only labels nodes with a b. The transitions of the tree automaton $\mathcal{A}$ are



The proofs of the following proposition is similar to the one for shifts of infinite or bi-infinite sequences (see [12], [10]).

Proposition 1. *A tree-shift is sofic if and only if it is accepted by a tree automaton.*

3.2 Irreducible tree-shifts

In this section, we define a notion of irreducibility which is suitable for tree-shifts.

A *finite complete prefix code* of Σ^* is a prefix set [2] P of finite words in Σ^* such that each word of Σ^* longer than the words of P has a prefix in P.

A tree-shift X is *irreducible* if for each pair of blocks $u, v \in \mathcal{B}(X)$, there is a tree t in X and a finite complete prefix code $P \subset \Sigma^{\geq \text{height}(u)}$, such that u is a subtree of t rooted at ε, and v is a subtree of t rooted at x for all $x \in P$.

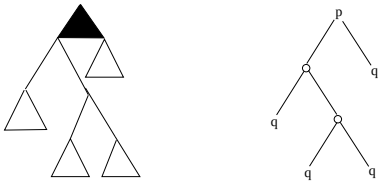


Fig. 2. (left) An irreducible tree-shift. Let t denotes the tree pictured. If u denotes the black block and v the white one, u is a subtree of t rooted at ε, and v is a subtree of t rooted at each $x \in P$, where P is the complete prefix code $\{00, 010, 011, 1\}$. (right) An hyperpath from q to p in a tree automaton.

A tree automaton is *irreducible* if for each pair of states p, q, there is a finite complete prefix code P of Σ^* and a finite computation C of the automaton on a pattern u such that $C_\varepsilon = p$ and $C_x = q$ for each $x \in P$. We say in this case that there is an *hyperpath* from q to p labeled by u. For two states p, q of a tree automaton, we say that p is *accessible* from q if there is a hyperpath from q to p.

Proposition 2. *An irreducible automaton accepts an irreducible sofic tree-shift. Conversely, for any irreducible sofic tree-shift, there is an irreducible automaton accepting it.*

Proposition 3. *Let S and T be two conjugate tree-shifts. Then S is irreducible if and only if T is irreducible.*

[2] i.e. no word is prefix of another one.

3.3 Synchronizing blocks

We define below the notion of synchronizing block[3] of a deterministic tree automaton.

Let $\mathcal{A} = (V, A, \Delta)$ be a deterministic tree automaton accepting a sofic tree-shift X, and u be a block (resp. pattern). We say that u is a *synchronizing block (resp. pattern) of* $\mathcal{A}$ if all computations of $\mathcal{A}$ on u terminate at the same state $q \in Q$. We say that u *focuses* to the state q. A deterministic tree automaton which has a synchronizing block is called *synchronized*.

3.4 Minimal deterministic tree automata

Let X be a tree-shift. A *context* c is a finite pattern with a marked leaf. If u is a pattern, $c(u)$ is the pattern c where the marked leaf is replaced by u. If $c(u) \in \mathcal{L}(X)$, we say that c is a *context of* u *in* X. Given a block u, we denote by $\mathrm{cont}_X(u)$ the set of all the contexts of u in X.

Given a tree automaton $\mathcal{A} = (V, A, \Delta)$ accepting a sofic tree-shift X, the *context* of a state $q \in V$ in $\mathcal{A}$ is the set of patterns u with a marked leaf x on which there exists a finite computation C of $\mathcal{A}$ with $C_x = q$ and x is a leaf of C. We denote it by $\mathrm{cont}_{\mathcal{A}}(q)$. Note that the context of a pattern u of X is the union of the contexts of the states p such that there is a computation of $\mathcal{A}$ on u ending in p. As a consequence, a sofic tree-shift has only a finite number of distinct contexts.

Let $\mathcal{A} = (V, A, \Delta)$ be a deterministic automaton accepting a sofic tree-shift X. We denote by $\delta(p, q, a)$ the unique state r such that $(p, q), a \to r \in \Delta$ when such a transition exists. We define a deterministic automaton $\mathcal{M}_{\mathcal{A}}$ accepting X called the *minimization of the automaton* $\mathcal{A}$ as follows. The states of $\mathcal{M}_{\mathcal{A}}$ are the classes of the coarsest partition of V such that if p, q belong to the same class, then for each letter a and each state r, $\delta(p, r, a)$ and $\delta(q, r, a)$ (resp. $\delta(r, p, a)$ and $\delta(r, q, a)$) belong to the same class, and $\delta(p, r, a)$ (resp. $\delta(r, p, a)$) is defined if and only if $\delta(q, r, a)$ (resp. $\delta(r, q, a)$) is defined. Let $[p]$ denotes the class of the state p. The transition $([p], [q]), a \to [\delta(p, q, a)]$ is a transition of $\mathcal{M}_{\mathcal{A}}$ if and only if $\delta(p, q, a)$ is defined. It can be shown that this definition is consistent, i.e. does not depend on the choice of the leader in each class. A deterministic automaton is *minimal* if it is equal to its minimization.

For any deterministic tree automaton, two states in a same class have the same context. If $\mathcal{A}$ is moreover irreducible and synchronized, it is minimal if and only if any two states have distinct contexts.

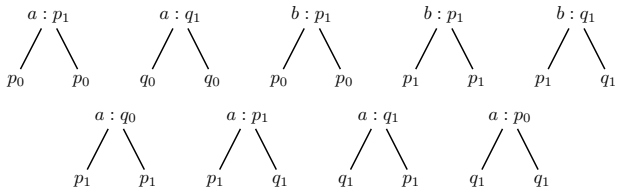
The minimization algorithm for deterministic tree automata accepting languages of finite trees, which is for instance described in [5, Section 1.5], can be applied to the tree automata accepting tree-shifts. Remember that all states in tree automata accepting tree-shifts are both initial and final.

In the framework of shifts of bi-infinite words, irreducible sofic shifts have a unique minimal deterministic automaton (i.e. all minimal deterministic automata accepting the shift are equal up to a renaming of the states), see [12].

[3] also called a homing pattern or a magic pattern.

The situation is quite different for trees since, as is shown below in Example 3, irreducible sofic tree-shifts may have several minimal deterministic tree automata. Indeed, contrary to the situation that we have for shifts of infinite or bi-infinite sequences, an irreducible minimal deterministic tree automaton may not have a synchronizing block.

Example 3. Let X be the full tree-shift on the alphabet $A = \{a, b\}$. It is accepted by a trivial one-state automaton. It is also accepted by the deterministic irreducible tree automaton $\mathcal{A} = (V, A, \delta)$ described by the following transitions and which is minimal.



One can overcome this difficulty with the notion of Shannon cover for irreducible tree-shifts.

Let X be a sofic tree-shift. The *context tree automaton* of X is the deterministic automaton $\mathcal{S} = (V, A, \Delta)$, where V is set of contexts of finite blocks of X. Since X is sofic, V is finite. The transitions of $\mathcal{S}$ are $(\mathrm{cont}_X(u), \mathrm{cont}_X(v)), a \to \mathrm{cont}_X(a, u, v)$, where $u, v \in \mathcal{B}(X)$.

Proposition 4. *The context tree automaton of a sofic tree-shift is synchronized.*

Proof. Let $\mathcal{S}$ be the context tree automaton of a sofic shift X. There is a finite computation C_1 in $\mathcal{S}$ on some block u_1 ending in $\mathrm{cont}_X(u_1)$. Let us assume that u_1 is not a synchronizing block of $\mathcal{S}$. Hence there is another computation C_2 of $\mathcal{S}$ on u_1 ending in $\mathrm{cont}_X(u_2)$ for some block $u_2 \neq u_1$. We get $\mathrm{cont}_X(u_2) \subsetneq \mathrm{cont}_X(u_1)$ since $\mathrm{cont}_X(u_2) \neq \mathrm{cont}_X(u_1)$. If u_2 is not a synchronizing block of $\mathcal{S}$, there is another computation C_2 of $\mathcal{S}$ on u_2 ending in $\mathrm{cont}_X(u_3)$. We get $\mathrm{cont}_X(u_3) \subsetneq \mathrm{cont}_X(u_2)$. By iterating this process, we either get a synchronizing block for $\mathcal{S}$ or an infinite strictly decreasing sequence of contexts. Hence, since the number of contexts is finite, there is a synchronizing block.

We define the *Shannon cover* of an irreducible sofic tree-shift X as the unique irreducible component $\mathcal{S}$ of its context tree automaton $\mathcal{C}$ obtained by keeping the states accessible from $\mathrm{cont}_{\mathcal{C}}(z)$, where z is a synchronizing block of $\mathcal{C}$. Since z is synchronizing and $\mathcal{C}$ is deterministic, each state in this component is the context of some synchronizing block.

Let us show that the states accessible from $\mathrm{cont}_{\mathcal{C}}(z)$ form an irreducible component and that it is the unique irreducible component of $\mathcal{S}$. It is enough to prove that there is a hyperpath from any state to $\mathrm{cont}_{\mathcal{C}}(z)$. Let $p = \mathrm{cont}_{\mathcal{C}}(u)$ be a state. Since X is irreducible, there is a pattern w of X and a finite complete prefix code P of $\Sigma^{\geq \mathrm{height}(z)}$, such that z is a subtree of w rooted at ε, and u is a subtree of w rooted at x for all $x \in P$. Let C be a computation of $\mathcal{C}$ on w.

We have $C_\varepsilon = \mathrm{cont}_{\mathcal{C}}(z)$, and for all $x \in P$, $C_x = p$. Hence there is an hyperpath from p to $\mathrm{cont}_{\mathcal{C}}(z)$. Finally, it is easy to check that the automaton $\mathcal{S}$ accepts X by using the irreducibility of X.

We now prove that the Shannon cover is the unique minimal deterministic irreducible and synchronized tree automaton accepting an irreducible sofic tree-shift.

Proposition 5. *Two minimal deterministic irreducible and synchronized tree automata accepting the same irreducible sofic tree-shift are equal up to a renaming of the states.*

4 Almost of finite type tree-shifts

The following notion of almost of finite type tree-shift extends to trees the notion of almost of finite type shift of bi-infinite sequences. The class contains strictly the class of irreducible tree-shifts of finite type and is strictly contained in the class of irreducible sofic tree-shifts.

Let X, Y be two tree-shifts. A block map $\Phi : X \to Y$ is *left closing* if there are non negative integers m, a (m for memory and a for anticipation) such that, whenever $\Phi(s) = s'$, $\Phi(t) = t'$ with $s'_x = t'_x$ for all $x \in \Sigma^i$ with $0 \leq i \leq (a+1+m)$, and $s_x = t_x$ for all $x \in \Sigma^i$ with $0 \leq i \leq a$, then $s_x = t_x$ for any $x \in \Sigma^{(a+1)}$. The map Φ is *left resolving* if m can be chosen equal to 0.

A tree automaton $\mathcal{A} = (V, A, \Delta)$ is *left closing* if any two computations of $\mathcal{A}$ on a same tree and ending in a same state are equal. Equivalently, there is a nonnegative integer m such any two finite computations C, C' of $\mathcal{A}$ on a same block $u \in \mathcal{B}_{m+1}(X)$ such that $C_\varepsilon = C'_\varepsilon$ satisfy $C_x = C'_x$ for all $x \in \{0, 1\}$. Hence a deterministic and left closing tree automaton corresponds, for trees, to the notion of automata of words which are both deterministic and co-deterministic with a finite delay.

A block map $\Phi : X \to Y$ is *right closing* if there are non negative integers m, a such that, whenever $\Phi(s) = s'$, $\Phi(t) = t'$ with $s'_x = t'_x$ for all $x \in \Sigma^i$ with $0 \leq i \leq (a + 1 + m)$, and $s_x = t_x$ for all $x \in \Sigma^i$ with $a + 1 \leq i \leq (a + 1 + m)$, then $s_x = t_x$ for all $x \in \Sigma^a$. The map Φ is *right resolving* if a can be chosen equal to 0.

Let X be a tree-shift. Let u be a block in $\mathcal{B}_m(X)$. A *cylinder* $\mathcal{C}[u]$ of X with basis u is the set of trees t in X such that $t_x = u_x$ for all $x \in \Sigma^*$ of length at most m.

A sofic tree-shift is *almost of finite type* (AFT) if it is the image of an irreducible tree-shift of finite type via a block map which is right resolving, left closing, and is one-to-one on a cylinder (equivalently, the map is one-to-one on a non trivial open set).

Proposition 6. *Let S and T be two conjugate irreducible sofic tree-shifts. Then S is AFT if and only if T is AFT.*

Proof. (sketch) It is easy to check that the composition of a right resolving map with a conjugacy is still right resolving. The composition of a left closing map with a conjugacy is still left closing. A conjugacy also keeps the property of being one-to-one on a cylinder.

We say that an automaton is an *almost of finite type automaton* (AFT automaton) if it is deterministic, irreducible, left closing and synchronized.

Proposition 7. *A tree-shift is AFT if and only if it is accepted by an AFT automaton.*

Proof. Let S be an AFT tree-shift on the alphabet A. Let $\Phi : X \to S$ be a block map from an irreducible tree-shift of finite type X onto S which is right resolving, left closing and one-to-one on a cylinder of S. Without loss of generality (by changing X into a higher block presentation of X), we can assume that X is an edge tree-shift and that Φ is a one-block map. Again by changing X into a higher block presentation of X, we can assume that Φ is left closing with parameters $a' = 1, m'$.

Let $\mathcal{A} = (V, E, \Delta)$ be an irreducible edge tree automaton accepting X. Let $\mathcal{B} = (V, A, \Delta')$ where $(p, q, \phi(e), r) \in \Delta'$ if and only if $(p, q, e, r) \in \Delta$.

Since Φ is a one-block right-resolving map (with some parameter m), $\mathcal{B}$ is a deterministic automaton.

We now prove that $\mathcal{B}$ has a synchronizing block. Since Φ is one-to-one on a cylinder $\mathcal{C}[z_0]$ of S, for any tree $t \in \mathcal{C}[z_0]$, any two computations of $\mathcal{B}$ on t are equal and thus end in a same state p. As a consequence, and by compacity arguments, there is a finite subtree z such that z_0 is a subtree of z at ε and such that each finite computation of $\mathcal{B}$ on z ends in the same state p_z, i.e. z is a synchronizing block of $\mathcal{B}$. Let us keep in $\mathcal{B}$ only the states accessible from p_z. Since S is irreducible, $\mathcal{B}$ remains irreducible and still accepts S.

Let us now show that $\mathcal{B}$ is left closing. If $\mathcal{B}$ were not left closing, there would be two distinct computations C, C' of $\mathcal{B}$ on a same tree t ending in a same state p. Let $(p, q, e, r) \in \Delta$ a transition going out of (p, q) for some states $p, r \in V$ (if there is none, there is a transition going out of (q, p) since $\mathcal{A}$ is irreducible). We get two distinct computations (r, C, D), (r, C', D) of $\mathcal{B}$ ending in r on a same tree $u = (\phi(e), t, t')$ for some tree t'. These two distinct computations of $\mathcal{B}$ are also two distinct computations of $\mathcal{A}$ on two trees s, s' of X with $s_\varepsilon = s'_\varepsilon = e$ and such that $\Phi(s) = \Phi(s') = u$. Since Φ is left closing with parameters $a' = 1, m'$, we get $s = s'$ and thus $C = C'$.

Conversely, if S is accepted by an AFT automaton $\mathcal{A} = (V, A, \Delta)$, let X be the edge tree-shift accepted by $\mathcal{T} = (V, \Delta, \Delta')$, whose transitions are $(p, q, (p, q, a, r)) \to r$ for $(p, q, a, r) \in \Delta$. Let Φ be the one-block map from X onto S defined by $\phi(p, q, a, r) = a$. This map is right resolving since $\mathcal{A}$ is deterministic. It is left closing since $\mathcal{A}$ is left closing. It is one-to-one on a cylinder since $\mathcal{A}$ is synchronized. As a consequence, S is AFT.

Corollary 1. *An irreducible sofic tree-shift is AFT if and only if its Shannon cover is AFT.*

Proof. Let X be an irreducible sofic tree-shift. By Proposition 5, any irreducible sofic tree-shift is accepted by an irreducible deterministic synchronized automaton $\mathcal{S}$ equal to the Shannon cover of X.

Let us assume that X is AFT. By Proposition 7, $\mathcal{S}$ is equal to the minimization of an AFT automaton $\mathcal{A}$. Let us show that $\mathcal{S}$ is left closing. If it is not left closing, then there is a tree $t \in X$ and two distinct computations of $\mathcal{S}$ on t ending in a same state. As a consequence, there are two distinct computations of $\mathcal{A}$ on t ending in two states p, p' which have the same context.

Let z be a synchronizing pattern of $\mathcal{A}$ focusing to the state q_z. Since $\mathcal{A}$ is irreducible, there is an hyperpath from p to q_z labeled by some pattern w such that z is a subtree of w rooted at ε. Since p and p' have the same context, there is also a hyperpath from p' to q labeled by w. Since z is synchronizing, $q = q_z$. This gives two distinct computations of $\mathcal{A}$ on a same tree ending in the same state q_z, which contradicts the fact that $\mathcal{A}$ is left closing. This proves that the Shannon cover of an AFT is left closing.

Conversely, if $\mathcal{S}$ is AFT, then X is AFT by Proposition 7.

Corollary 2. *Let S be an irreducible sofic shift accepted by a deterministic tree automaton. It is decidable whether S is AFT.*

Proof. One can compute the Shannon cover of the sofic tree-shift and check whether it is AFT as explained in Section 5.

5 The algorithmic issue

In this section, we design algorithms to check whether a deterministic tree automaton is synchronized, and whether it is left closing. We also describe an algorithm to compute the Shannon cover of an irreducible sofic tree-shift given by a deterministic tree automaton that accepts it.

5.1 Computation of the Shannon cover

Let $\mathcal{A} = (V, A, \delta)$ be a deterministic automaton. We define the deterministic tree automaton $\mathcal{D}(\mathcal{A})$ as the accessible part from the state V of the tree automaton $(\mathfrak{P}(V), A, \delta')$, where for, $P, Q \in \mathfrak{P}(V)$, $\delta'(P, Q, a) = \{\delta(p, q, a) \mid p \in P, q \in Q\}$ if this set is nonempty, and is not defined otherwise.

Proposition 8. *It can be checked in polynomial time whether a tree automaton is irreducible.*

Proposition 9. *It is decidable whether a deterministic tree automaton is synchronized.*

Proof. The automaton $\mathcal{A}$ is synchronized if and only if $\mathcal{D}(\mathcal{A})$ contains a singleton state. The time and space complexity of this algorithm is exponential in the number of states of $\mathcal{A}$.

Proposition 10. *Let $\mathcal{A} = (V, A, \delta)$ be a deterministic automaton accepting an irreducible sofic tree-shift X. The Shannon cover of X is computable from $\mathcal{A}$.*

Proof. Let $\mathcal{D}(\mathcal{A}) = (\mathfrak{P}(V), A, \delta')$ and R be a minimal state of $\mathcal{D}(\mathcal{A})$ for the inclusion. Let u the label of a hyperpath from V to R. Then u a synchronizing pattern of $\mathcal{D}(\mathcal{A})$. Indeed, any finite computation of $\mathcal{D}(\mathcal{A})$ ends in R by minimality of R. We keep in $\mathcal{D}(\mathcal{A})$ only the states accessible from R and get an irreducible and synchronized automaton accepting X. Its minimization gives the Shannon cover of X by Proposition 5.

We now describe algorithms to check whether an irreducible deterministic automaton is left closing.

5.2 The pair graph of a tree automaton

Given a deterministic tree automaton $\mathcal{A} = (V, A, \Delta)$, we define the *square automaton* of $\mathcal{A}$, denoted by $\mathcal{A} \times \mathcal{A} = (V \times V, A, \Delta')$, as the deterministic automaton whose transitions are $(p, p'), (q, q'), a \to (r, r')$ if and only if $(p, q), a \to r$ and $(p', q'), a \to r'$ are transitions of $\mathcal{A}$. A *diagonal state* of $\mathcal{A} \times \mathcal{A}$ is a state (p, p) for some $p \in V$.

Square automata of finite words (see for instance [16, p. 647]) are used to check properties of pairs of paths. We extend this notion, together with a notion a pair graph, to trees, to check properties of pairs of computations. Seidl [17] used branch automata to check the degree of unambiguity of finite tree automata.

Proposition 11. *A deterministic tree automaton is not left closing if and only if there is a computation in the square automaton ending in a diagonal state and containing a non diagonal one.*

Proof. By definition of $\mathcal{A} \times \mathcal{A}$, the existence of a computation in $\mathcal{A} \times \mathcal{A}$ ending in a state (p, p) and containing a state (r, s) with $r \neq s$ is equivalent to the existence of two distinct computations of $\mathcal{A}$ on a same tree.

In order to check the above property, we build *the pair graph* $G_{\mathcal{A}} = (V_G, E_G)$ of $\mathcal{A}$, where $V_G \subseteq (V^2 \times V^2) \cup V^2$ is the set of vertices, $E_G \subseteq V_G \times \{0, 1\} \times A \times V_G$ is the set of edges labeled by 0 or 1 and a letter from A. For more convenience, an edge labeled by 1 is noted by a plain arrow $\longrightarrow$ and is called a plain edge, and an edge labeled by 0 is noted by a dashed arrow $\dashrightarrow$ and is called a dashed edge. For each pair of transitions $(p, q), a \to r$ and $(p', q'), a \to r'$ of $\mathcal{A}$,

$$\begin{aligned}((r, r'), (s, s')) &\xrightarrow{0,a} ((p, p'), (q, q')),\\ ((s, s'), (r, r')) &\xrightarrow{1,a} ((p, p'), (q, q')),\\ (r, r') &\xrightarrow{0,a} ((p, p'), (q, q')),\\ (r, r') &\xrightarrow{1,a} ((p, p'), (q, q')),\end{aligned}$$

are edges of $G_{\mathcal{A}}$, for each pair (s, s').

A vertex of $G_{\mathcal{A}}$ is *useful* if it has at least one outgoing plain edge and at least one outgoing dashed edge. We keep the *essential part* of the pair graph obtained by discarding vertices which are not useful, and their incoming and outgoing edges. A vertex $((p,q),(r,s))$ of G_A is called *non diagnonal* if either $p \neq q$ or $r \neq s$.

Proposition 12. *A deterministic tree automaton is not left closing if and only its there is a path in its pair graph starting at a vertex (p,p) and ending in a non diagonal vertex.*

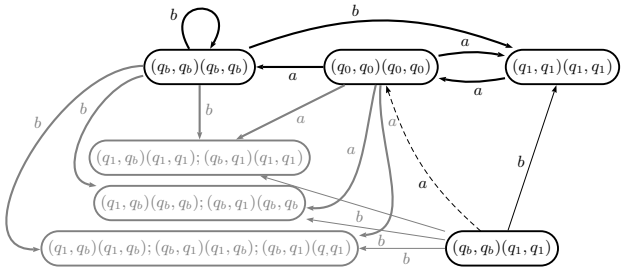


Fig. 3. The pair graph for the tree automaton of Example 1. A thick edge represents a plain edge and a dashed edge with the same label. The non useful edges and vertices are drawn in grey. Each vertex (p,q) is identified with the vertex $(p,q)(p,q)$. For the test, the pair $((p,q),(r,s))$ may not be represented if $((r,s),(p,q))$ does. The tree-shift X accepted by $\mathcal{A}$ satisfies the property of Proposition 12 since there is no path from a diagonal state to a non diagonal one. Then $\mathcal{A}$ is a left closing automaton and as a consequence the tree-shift X is AFT.

The number of vertices of $G_{\mathcal{A}}$ is at most $O(|V|^4)$ and its number of edges of $G_{\mathcal{A}}$ is at most $O(|V|^6)$. The property of Proposition 12 can be checked in a linear time in the size of $G_{\mathcal{A}}$. As a consequence, it can be checked in polynomial time whether the Shannon cover of an irreducible sofic tree-shift is AFT. Note that Seidl's check of the finite degree of ambiguity of tree automata in [17] has a similar complexity (the cube of the size of the transitions of the tree automaton). The pair graph for the tree automaton $\mathcal{A}$ of Example 1 is given in Figure 3.

Conclusion

In this article, we have shown that tree-shifts differ from one-sided shifts of infinite sequences at least concerning the following property: there may be more than one minimal deterministic irreducible tree automata accepting the same irreducible sofic tree-shift. The reason is that such automata do not always have a synchronizing block. For irreducible sofic tree-shifts, the Shannon cover remedy for this lack and allows us to define the class of almost finite type tree-shifts. In further work we will focus on topological and syntactic properties of AFT tree-shifts.

References

1. N. Aubrun and M.-P. Béal. Decidability of conjugacy of tree-shifts of finite type. In *ICALP '09: Proceedings of the 36th International Colloquium on Automata,*

Languages and Programming, pages 132–143, Berlin, Heidelberg, 2009. Springer-Verlag.
2. T. Bates, S. Eilers, and D. Pask. Reducibility of covers of AFT shifts. *Israel Journal of Mathematics*, 2010. to appear.
3. M.-P. Béal, F. Fiorenzi, and D. Perrin. A hierarchy of shift equivalent sofic shifts. *Theor. Comput. Sci.*, 345(2-3):190–205, 2005.
4. M. Boyle, B. Kitchens, and B. Marcus. A note on minimal covers for sofic systems. *Proc. Amer. Math. Soc.*, 95(3):403–411, 1985.
5. H. Comon, M. Dauchet, R. Gilleron, C. Löding, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. Tree automata techniques and applications. Available on: http://www.grappa.univ-lille3.fr/tata, 2007. release October, 12th 2007.
6. E. Coven, A. Johnson, N. Jonoska, and K. Madden. The symbolic dynamics of multidimensional tiling systems. *Ergodic Theory and Dynamical Systems*, 23(02):447–460, 2003.
7. M. Fujiwara and M. Osikawa. Sofic systems and flow equivalence. *Math. Rep. Kyushu Univ.*, 16(1):17–27, 1987.
8. A. S. A. Johnson and K. M. Madden. The decomposition theorem for two-dimensional shifts of finite type. *Proc. Amer. Math. Soc.*, 127(5):1533–1543, 1999.
9. N. Jonoska and B. Marcus. Minimal presentations for irreducible sofic shifts. *IEEE Trans. Inform. Theory*, 40(6):1818–1825, 1994.
10. B. P. Kitchens. *Symbolic dynamics*. Universitext. Springer-Verlag, Berlin, 1998. One-sided, two-sided and countable state Markov shifts.
11. D. Lind and K. Schmidt. Symbolic and algebraic dynamical systems. In *Handbook of dynamical systems, Vol. 1A*, pages 765–812. North-Holland, Amsterdam, 2002.
12. D. A. Lind and B. H. Marcus. *An Introduction to Symbolic Dynamics and Coding*. Cambridge University Press, 1995.
13. B. H. Marcus. Sofic systems and encoding data. *IEEE Transactions on Information Theory*, 31(3):366–377, 1985.
14. M. Nasu. *Textile Systems for Endomorphisms and Automorphisms of the Shift*. American Mathematical Society, 1995.
15. D. Perrin and J. Pin. *Infinite words*. Elsevier Boston, 2004.
16. J. Sakarovitch. *Elements of Automata Theory*. Cambridge University Press, 2009.
17. H. Seidl. On the finite degree of ambiguity of finite tree automata. In *Fundamentals of computation theory (Szeged, 1989)*, volume 380 of *Lecture Notes in Comput. Sci.*, pages 395–404. Springer, New York, 1989.
18. W. Thomas. Automata on infinite objects. In *Handbook of theoretical computer science, Vol. B*, pages 133–191. Elsevier, Amsterdam, 1990.
19. R. F. Williams. Classification of subshifts of finite type. In *Recent advances in topological dynamics (Proc. Conf. Topological Dynamics, Yale Univ., New Haven, Conn.*, pages 281–285. Lecture Notes in Math., Vol. 318. Springer, Berlin, 1973.
20. S. Williams. Covers of non-almost-finite type sofic systems. *Proc. Amer. Math. Soc.*, 104(1):245–252, 1988.