HAL
open science

Sofic Tree-Shifts

Nathalie Aubrun, Marie-Pierre Béal

To cite this version:

Nathalie Aubrun, Marie-Pierre Béal. Sofic Tree-Shifts. Theory of Computing Systems, 2013, 53 (4), pp.621-644. 10.1007/s00224-013-9456-1 . hal-00627797v2

HAL Id: hal-00627797

https://hal.science/hal-00627797v2

Submitted on 18 Feb 2013

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

myjournal manuscript No.
(will be inserted by the editor)

Sofic tree-shifts

Nathalie Aubrun and Marie-Pierre Béal

Abstract We introduce the notion of sofic tree-shifts which corresponds to symbolic dynamical systems of infinite ranked trees accepted by finite tree automata. We show that, contrary to shifts of infinite sequences, there is no unique reduced deterministic irreducible tree automaton accepting an irreducible sofic tree-shift, but that there is a unique synchronized one, called the Fischer automaton of the tree-shift. We define the notion of almost of finite type tree-shift which are sofic tree-shifts accepted by a tree automaton which is both deterministic and co-deterministic with a finite delay. It is a meaningful intermediate dynamical class in between irreducible finite type tree-shifts and irreducible sofic tree-shifts. We characterize the Fischer automaton of an almost of finite type tree-shift and we design an algorithm to check whether a sofic tree-shift is almost of finite type. We prove that the Fischer automaton is a topological conjugacy invariant of the underlying irreducible sofic tree-shift.

1 Introduction

In [1] we introduced the notion of tree-shifts of finite type defined as sets of infinite ranked trees avoiding a finite number of forbidden patterns. Infinite trees have a natural structure of one-sided symbolic dynamical systems equipped with several shift transformations. The ith shift transformation applied to a

This work is supported by the French National Agency (ANR) through "Programme d'Investissements d'Avenir" (Project ACRONYME n°ANR-10-LABX-58) and through the ANR SubTile

Nathalie Aubrun
LIP, UMR 5668, ENS de Lyon, CNRS
E-mail: nathalie.aubrun@ens-lyon.fr

Marie-Pierre Béal
Université Paris-Est
Laboratoire d'informatique Gaspard-Monge, UMR 8049 CNRS
E-mail: beal@univ-mlv.fr

tree gives the subtree rooted at the child number i of the tree. Sets of finite patterns of tree-shifts of finite type are strictly locally testable tree languages [29] (also called k-testable tree languages, or k-grams in the case of sequences). For these languages, the effect of events that occured beyond a certain depth window are ignored when processing a tree. Probabilistic k-testable models are used for pattern classification and stochastic learning [29].

Tree-shifts are also highly interesting to study as they constitute an intermediate class in 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 [19],[22],[15],[11]), is undecidable. The decidability of the conjugacy of finite type shifts of bi-infinite sequences is unknown. However, the conjugacy of finite shifts of one-sided infinite sequences was shown to be decidable by Williams ([30], see also [20]). In [3], we extended Williams'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. Sofic tree-shifts have been studied in [9] and [12] using top-down tree automata accepting the tree-shifts. In [12], topological properties of cellular automata (or block maps) on trees are investigated.

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 reduced deterministic irreducible tree automata. This is due to the lack of a synchronizing block, even in reduced deterministic irreducible automata. We introduce the notion of synchronized tree automaton and the notion of Fischer automaton of an irreducible sofic tree-shift. We prove that the Fischer automaton is the unique reduced synchronized deterministic tree automaton accepting an irreducible sofic tree-shift. We also define the Krieger automaton of a sofic tree-shift and compare it to the Fischer automaton in the case the tree-shift is irreducible.

The existence of the Fischer automaton allows us to introduce the class of almost of finite type tree-shifts, which extends the known notion of almost of finite type shifts of sequences. Almost of finite type tree-shifts are sofic shifts accepted by a tree automaton which is both deterministic and co-deterministic with a finite delay. The almost of finite type concept was introduced by Marcus in [21] 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 [8], [31], [16], [6], [4]). The class is stable by conjugacy and it is also invariant by flow equivalence [13]. Almost of finite type shifts enjoy various properties which are no shared by all sofic shifts. It is the one big, natural class of nice sofic shifts.

We characterize the Fischer automaton of an almost of finite type tree-shift and design an algorithm to check whether a sofic tree-shift is almost of

finite type. We prove that the Fischer automaton is a topological conjugacy invariant of the underlying irreducible sofic tree-shift. This extends a similar result from Krieger in [18] (see also [8]) in the framework of sequences.

The paper is organized as follows. In Section 2.1 and Section 2.2, we give basic definitions about tree-shifts and conjugacies between tree-shifts. In Section 3, we define the notion of a tree automaton accepting a tree-shift. We refer to [10], [27], and [24] for more general trees and automata on finite and infinite trees. We define irreducible tree-shifts and irreducible tree automata in Section 3.3, synchronized tree automata in Section 3.4. The notions of Fischer and Krieger automata are introduced in Section 3.5. Almost of finite type tree-shifts are defined and characterized in Section 4. In the algorithmic issue, Section 6, we give a construction of the Fischer automaton of an irreducible sofic tree-shift. We design a polynomial-time algorithm to check whether an irreducible sofic tree-shift given by its Fischer automaton is almost of finite type. A short version of this paper was presented in [2].

2 Definitions

2.1 Tree-shifts

We first recall some basic definitions of symbolic dynamics on infinite trees. We consider infinite trees whose nodes have a fixed number of children and are labelled in a finite alphabet. We restrict us to binary trees, but all results extend to the case of trees with d children for a fixed positive integer d, thus including the case of one-sided sequences.

Let $\Sigma = \{0, 1\}$ and Σ^* be the set of words over Σ. An *infinite tree* t over a finite alphabet A is a total function from Σ^* to A. A node of an infinite 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. 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 $u : L \to A$, where L is a finite prefix-closed subset[1] of Σ^*. The set L is called the *support of the pattern*. A *finite tree* is a pattern with a finite domain such that for each node x, $x0, x1$ are either both not in the domain or both in the domain. This implies that the domain is prefix-closed. A node of a finite tree is a word of L. A node without children is called a *leaf*. A *subtree* of a (finite or infinite) tree t rooted at a node x of t is the tree s defined by $s_y = t_{xy}$ for all $y \in \Sigma^*$ such that xy is a node of t.

If n is nonnegative integer, we denote by $\Sigma^{\leq n}$ the set of words of length at most n on Σ. A *block of height* n, where n is a positive integer, is a pattern whose support is $\Sigma^{\leq n-1}$. The *height* of a block u is denoted by height(u).

We say that a pattern (resp. block) u of support L is a *pattern of a tree* t (resp. a *block of a tree* t) if there is a word $x \in \Sigma^*$ such that $t_{xy} = u_y$ for all

[1] each prefix of L belongs to L.

node y of u. We say that u is a pattern (resp. block) of t rooted at the node x. If u is not a pattern (resp. block) of t, one says that t *avoids* u.

Unless otherwise stated, a tree is an infinite tree. When Σ are A are fixed, we denote by $\mathcal{T}$ 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 $\mathcal{T}$ 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 $\mathcal{T}$ equipped with the shift transformations σ_i is called the *full tree-shift* of infinite trees over A.

If t, t' are two trees, we define the distance $d(t, t') = 2^{-n}$, where n is the length of the shortest word x in Σ^* such that $t_x \neq t'_x$ if such a word exists, and $d(t, t) = 0$ otherwise. The definition satisfies the conditions of a distance is similar as the one the for infinite sequences (see for instance [20, Example 6.1.10]). This distance induces a topology on $\mathcal{T}$ where two trees are close when large blocks of coordinates rooted at their root agree.

We define a *tree-shift* X of $\mathcal{T}$ as the set $\mathsf{X}_{\mathcal{F}}$ of all trees avoiding any element of a set of blocks $\mathcal{F}$. The set $\mathcal{F}$ is called *a set of forbidden blocks* of X. If the set of trees on A is equipped with topology induced by the distance d, a tree-shift X is a closed subset of $\mathcal{T}$ such that $\sigma_i(X) \subseteq X$ for all shift transformation σ_i. A *tree-shift of finite type* (SFT) X of $\mathcal{T}$ is a set $\mathsf{X}_{\mathcal{F}}$ of all trees avoiding any block of a *finite* set of blocks $\mathcal{F}$.

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. The set $\mathcal{L}(X)$ is called *the set of allowed patterns* of X. If u is a block of height n with $n \geq 2$ and $i \in \Sigma$, we denote by $\sigma_i(u)$ the block of height $n-1$ such that $\sigma_i(u)_x = b_{ix}$ for $x \in \Sigma^{\leq n-2}$. The block u is written $u = (u_\varepsilon, \sigma_0(u), \sigma_1(u))$.

A set of patterns L is *factorial* if $u \in L$ and v is a subtree of u rooted at a node x of u (*i.e.* v is a sub-pattern of u) implies $v \in L$. The set L is *extensible* if for any pattern u in L with support $S(u)$, there is a pattern $v \in L$ with support $S(v)$, such that $S(u) \subset S(v)$, v coincides with u on $S(u)$, and for any x in $S(u)$, $x0$ and $x1$ belong to $S(v)$.

The language of allowed patterns of a tree-shift is factorial and extensible. Conversely, such a language of finite trees is the set of allowed patterns of a tree-shift as is shown in the following result which is similar as the one known for shifts of bi-infinite words (see for instance [7]).

Proposition 1 *Let L be a factorial and extensible set of finite trees. The set $\mathcal{X}(L)$ of infinite trees whose patterns belong to L is a tree-shift and $\mathcal{L}(\mathcal{X}(L)) = L$. Conversely, if X is a tree-shift, then $X = \mathcal{X}(\mathcal{L}(X))$.*

Proof Let $X = \mathcal{X}(L)$. It is clear that X is a tree-shift and $\mathcal{L}(X) \subseteq L$. We show that $L \subseteq \mathcal{L}(X)$. Let u be a finite tree of L. Since L is extensible and factorial, we can construct an infinite tree t such that u is a subtree of t rooted at the empty node, and such that all block of t belongs to L. Then $u \in \mathcal{L}(X)$.

For the second assertion, let $L = \mathcal{L}(X)$. It is clear that $X \subseteq \mathcal{X}(\mathcal{L}(X))$. Conversely, let $t \in \mathcal{X}(\mathcal{L}(X))$. Since all patterns of t belong to L, the block u_n of height n of t rooted at ε belongs to $L = \mathcal{L}(X)$. Thus there is an infinite

tree $t^{(n)}$ in X whose block of height n of t rooted at ε is u_n. The sequence $(t^{(n)})_{n\geq 1}$ converges to the tree t. Since X is closed, t belongs to X.

Example 1 In Figure 1 is pictured an infinite tree of a tree-shift $\mathsf{X}_{\mathcal{F}_{\mathrm{odd}}}$ on the alphabet $\{a, b\}$, whose trees contain an even number of a between two b on any path in the tree. Moreover, any two paths starting at a same node and ending at nodes labelled by b have the same number of a modulus 2. Hence the tree $a(a(b, b), b)$ is not allowed. This tree-shift is not of finite type. The forbidden blocks are those containing an odd number of a between two b on some path in the tree or containing two paths starting at a same node and ending at nodes labelled by b and where the number of a modulus 2 are distinct.

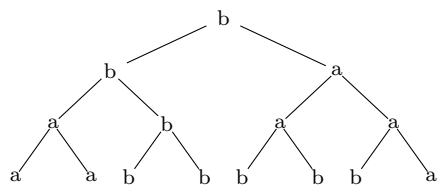


Fig. 1 A part of an infinite tree of the tree-shift $\mathsf{X}_{\mathcal{F}_{\mathrm{odd}}}$, where $\mathcal{F}_{\mathrm{odd}}$ is the set of patterns containing an odd number of a between two b on some path in the tree.

2.2 Block maps and conjugacies

Let A, A' be two finite alphabets, $\mathcal{T}$ (resp. $\mathcal{T}'$) the set of trees on A (resp. A'). Let X be a tree-shift of $\mathcal{T}$ and m be a positive integer. A map $\Phi : X \to \mathcal{T}'$ 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 $x \in \Sigma^*$, $\Phi(t)_x = \phi(u)$, where u is the block of t of height m rooted at x. 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 positive integer m.

It is known from the Curtis-Lyndon-Hedlund theorem (see [14]) that block maps are exactly the maps $\Phi : X \to Y$ which are continuous and commute with all tree-shifts transformations, *i.e.* such that $\sigma_i(\Phi(t)) = \Phi(\sigma_i(t))$ for all $t \in X$ and all $i \in \Sigma$. Actually, the proof in [14] applies to shifts of sequences but it extends directly to shifts of trees.

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 tree-shifts X and $X^{(m)}$ are conjugate (see [3]).

3 Sofic tree-shifts

3.1 Tree automata

In this section we consider bottom-up tree automata accepting finite or infinite trees. Such a tree automaton starts its computation from the leaves, or 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 $\Delta \subset V \times V \times A \times V$ is a set of transitions of the form $(p, q) \xrightarrow{a} r$, with $p, q, r \in V$, $a \in A$. A transition $(p, q) \xrightarrow{a} r$ is called a transition *labelled by* a, *going out of* the pair of states (p, q) and *coming in* the state q. A transition $(p, q) \xrightarrow{a} r$ will be pictured by

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

Such a tree automaton is *deterministic* if for each pair of states (p, q) and for each $a \in A$, there is at most one transition $(p, q) \xrightarrow{a} q$. 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 labelled in V such that, for each node x of t, there is a transition $(c_{x0}, c_{x1}) \rightarrow t_x c_x \in \Delta$. A tree t is *accepted* by $\mathcal{A}$ if there exists a computation of $\mathcal{A}$ on t.

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 a determinization, see for instance [10] for details). In the sequel, we assume that all states of a tree automaton are *accessible*, *i.e.* each state is the root of some computation of the tree automaton.

In this paper, a finite tree is complete binary (*i.e.* all node has zero or two children). A (bottom-up) *finite computation* of $\mathcal{A}$ on a finite tree t is a finite complete binary tree c on V such that, for each node x of t, there is a transition $(c_{x0}, c_{x1}) \xrightarrow{t_x} c_x \in \Delta$.

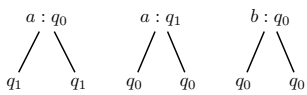
The set of infinite trees accepted by $\mathcal{A}$ is a tree-shift. Indeed, since all states of $\mathcal{A}$ are accessible, the language L of patterns accepted by $\mathcal{A}$ is the factorial and extensible. Then X is the tree-shift $\mathcal{X}(L)$.

A tree automaton is called *a transition tree automaton* if all transitions have distinct labels. A *transition tree-shift* is a tree-shift (of finite type) which is accepted by a transition tree automaton.

Let $\mathcal{A} = (V, A, \Delta)$ be a tree automaton. The set of trees t labelled on the alphabet Δ such that $t_x = (c_{x0}, c_{x1}) \xrightarrow{t'_x} c_x$ for some computation c of $\mathcal{A}$ on

a tree t' is called the *transition tree-shift* $X_{\mathcal{A}}$ *of* $\mathcal{A}$. Let $\mathcal{S}_{\mathcal{A}}$ be the tree-shift accepted by $\mathcal{A}$. There is a one-block map $\Phi_{\mathcal{A}} : X_{\mathcal{A}} \to S_{\mathcal{A}}$ assigning to each transition $e = (c_{x0}, c_{x1}) \xrightarrow{a} c_x$ its label a. The transition tree-shift $X_{\mathcal{A}}$ is a tree-shift of finite type. It is accepted by the tree automaton $\mathcal{B} = (V, \Delta, \Delta')$ whose transitions are $(p, q) \xrightarrow{(p,q,a,r)} r$ for $(p, q) \xrightarrow{a} r \in \Delta$. It is defined by the finite set of forbidden blocks (e, f, g) of height 2, where $e = (p, q, a, r)$, $f = (p_1, q_1, a_1, r_1)$, $g = (p_2, q_2, a_2, r_2)$, and either $r_1 \neq p$ or $r_2 \neq q$.

Example 2 We define a tree automaton $\mathcal{A}$ with two states q_0, q_1 which accepts the tree-shift $X = \mathsf{X}_{\mathcal{F}_{\mathrm{odd}}}$ of Example 1. The two states q_0 and q_1 control the parity of the number of a encountered from any last b below. The transitions of the tree automaton $\mathcal{A}$ are



The proof of the following proposition is similar to the one for shifts of infinite or bi-infinite sequences (see [20], [17]).

Proposition 2 *A tree-shift is sofic if and only if it is accepted by a tree automaton.*

Proof Let S be a sofic tree-shift. By definition, it is the image of a tree-shift of finite type X by an m-block map Φ. Without loss of generality, by changing X into its conjugate $X^{(m)}$, we can assume that Φ is a one-block map (*i.e.* $m = 1$). Then the function ϕ associated to Φ extends to patterns. Since X is of finite type, there is a positive integer n such that $X = \mathsf{X}_{\mathcal{F}}$ where $\mathcal{F}$ is a subset of the set of blocks of height n. The tree-shift X is accepted by the tree automaton $\mathcal{A} = (V, V, \Delta)$ where V is the set of blocks of height $n-1$ and $(p, q) \xrightarrow{r} r \in \Delta$ if and only if $(r_\varepsilon, p, q) \notin \mathcal{F}$. The tree-shift S is thus accepted by the tree automaton $\mathcal{B} = (V, A, \Delta')$ where $(p, q) \xrightarrow{\phi(r)} r \in \Delta'$ if and only if $(p, q) \xrightarrow{r} r \in \Delta$.

Conversely, if S is a set of infinite trees accepted by a tree automaton $\mathcal{B}$, then there is a one block map $\Phi_{\mathcal{B}}$ from the transition tree-shift $X_{\mathcal{B}}$ to S. Since the transition tree-shift is of finite type, S is a sofic tree-shift.

3.2 Symbolic conjugacy

We define the notion of symbolic conjugacy for tree automata which extends the notion of symbolic conjugacy of automata on sequences (see [5]). Let $\mathcal{A}$ and $\mathcal{B}$ be two tree automata accepting the sofic tree-shifts $S_{\mathcal{A}}$ and $S_{\mathcal{B}}$ and whose transition tree-shifts are $X_{\mathcal{A}}$ and $X_{\mathcal{B}}$. Let $\Phi_{\mathcal{A}} : X_{\mathcal{A}} \to S_{\mathcal{A}}$ (resp. $\Phi_{\mathcal{B}} : X_{\mathcal{B}} \to S_{\mathcal{B}}$) be the one-block map assigning to each transition its label. We say

that $\mathcal{A}$ and $\mathcal{B}$ are *symbolic conjugate* if there exists conjugacies $\Phi : X_{\mathcal{A}} \to X_{\mathcal{B}}$ and $\Psi : S_{\mathcal{A}} \to S_{\mathcal{B}}$ such that the following diagram commutes.

$$\begin{array}{ccc} X_{\mathcal{A}} & \xrightarrow{\Phi} & X_{\mathcal{B}} \\ {\scriptstyle \Phi_{\mathcal{A}}}\downarrow & & \downarrow{\scriptstyle \Phi_{\mathcal{B}}} \\ S_{\mathcal{A}} & \xrightarrow{\Psi} & \mathcal{S}_{\mathcal{B}} \end{array}$$

This notion will be used in Section 5 to provide a conjugacy invariant of tree-shifts.

3.3 Irreducible tree-shifts

In this section, we define a notion of irreducibility which is suitable for tree-shifts. As it will appear below, this definition is very strong. We nevertheless think that it is the good one since it allows one to entend the theory of irreducible sofic shifts of sequences naturally to irreducible sofic tree-shifts. Moreover the class of irreducible sofic tree-shifts is very large.

A *finite complete prefix code* of Σ^* is a prefix set [2] P of finite words in Σ^* such that any word of Σ^* which is 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 with $u, v \in \mathcal{B}_n(X)$, there is a tree t in X and a finite complete prefix code $P \subset \Sigma^{\geq n}$, such that u is a subtree of t rooted at ε, and v is a subtree of t rooted at x for all $x \in P$.

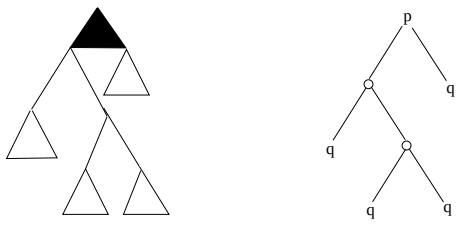


Fig. 2 (left) An irreducible tree-shift. Let t denote the tree pictured. If u denotes the black block and v the white one, u is a subtree of t rooted at ε, and v is a subtree of t rooted at each $x \in P$, where P is the complete prefix code $\{00, 01, 10, 110, 111\}$. (right) An hyperpath from q to 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 $P \subset \Sigma^*$ and a finite computation c of the tree 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 labelled 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.

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

Proposition 3 *An irreducible tree automaton accepts an irreducible sofic tree-shift. Conversely, for any irreducible sofic tree-shift, there is an irreducible tree automaton accepting it.*

Proof The forward implication is easy. The converse will be proved later by using the notion of Fischer automaton.

Proposition 4 *Let S and T be two conjugate tree-shifts. Then S is irreducible if and only if T is irreducible.*

Proof The proof is straightforward.

3.4 Synchronizing blocks

We define below the notion of synchronizing pattern [3] or block 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 pattern (resp. block). We say that u is a *synchronizing pattern (resp. block) of* $\mathcal{A}$ if the set of computations of $\mathcal{A}$ on u is nonempty and all computations of $\mathcal{A}$ on u end in the same state $q \in Q$. We say that the pattern u *focuses* to the state q. A tree t such that all computations of $\mathcal{A}$ on t are rooted with the same state $q \in Q$ is a *synchronizing tree* of $\mathcal{A}$. A deterministic tree automaton which has a synchronizing pattern is called *synchronized*.

Note that if u is a synchronizing pattern, then any pattern v such that u is a subtree of v rooted at ε is also synchronizing.

3.5 Minimal deterministic tree automata

Let X be a tree-shift. A *context* ℓ is a pattern where one leaf is replaced by a hole. If u is a pattern, ℓu is the pattern obtained by substituting u in place of the hole of ℓ. If $\ell u \in \mathcal{L}(X)$, we say that ℓ 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 contexts ℓ, with a hole at the node x, on which there exists a finite computation c of $\mathcal{A}$ with $c_x = q$. We denote this set 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 rooted by p. As a consequence, a sofic tree-shift has only a finite number of of sets $\mathrm{cont}_X(u)$.

Let $\mathcal{A} = (V, A, \Delta)$ be a tree automaton. The tree automaton $\mathcal{A}$ is said to be *reduced* if $p \neq q$ implies $\mathrm{cont}_{\mathcal{A}}(p) \neq \mathrm{cont}_{\mathcal{A}}(q)$ for all states p, q in V.

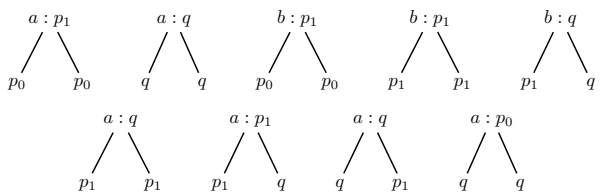
[3] also called a homing pattern or a magic pattern.

Let $\mathcal{A} = (V, A, \Delta)$ and $\mathcal{B} = (V', A, \Delta')$ be two deterministic automata. A *reduction* from $\mathcal{A}$ onto $\mathcal{B}$ is a map h from V onto V' such that, for any letter $a \in A$, one has $(p, q) \xrightarrow{a} r \in \Delta$ if and only if $(h(p), h(q)) \xrightarrow{a} r \in \Delta'$.

In the framework of shifts of bi-infinite words, irreducible sofic shifts have a unique reduced deterministic tree automaton (*i.e.* all reduced deterministic automata accepting the shift are equal up to a renaming of the states), see [20]. The situation is quite different for trees since, as shown below in Example 3, irreducible sofic tree-shifts may have several reduced deterministic tree automata. Indeed, contrary to the situation that we have for shifts of infinite or bi-infinite sequences, an irreducible reduced deterministic tree automaton may not have a synchronizing block.

Proposition 5 *There are reduced irreducible deterministic tree automata which are not synchronized.*

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



This tree automaton is irreducible. Indeed there is an hyperpath

$$p_0 \rightarrowtail p_1 \rightarrowtail q \rightarrowtail p_0$$

where $p \rightarrowtail q$ denotes a hyperpath from p to q. It is reduced since any two states have distinct contexts. Indeed, the state q has not the same context as p_0, p_1 since the pattern whose root is b with a left hole is a context of p_0, p_1 only. Furthermore the following pattern whose hole is represented with a black dot

is a context of p_0 only. Hence all states have distinct contexts.

Nevertheless, this tree automaton is not synchronized. Indeed, let us denote by $\mathfrak{P}(V)$ the set of subsets of V. Let $\mathcal{D}(\mathcal{A})$ be the accessible part from V of the deterministic tree automaton $(\mathfrak{P}(V), A, \Delta')$, where, if $P, Q \in \mathfrak{P}(V)$, $(P, Q) \xrightarrow{a} R$ is in Δ' if the set R of states r such that $(p, q) \xrightarrow{a} r \in \Delta$ for some $p \in P$ and $q \in Q$, is nonempty. The tree automaton $\mathcal{D}(\mathcal{A})$ contains two

states $V = \{p_0, p_1, q\}$ and $\{p_1, q\}$ and hence no singleton state. For any pair of states (P, Q) of $\mathcal{D}(\mathcal{A})$, there is a transition labelled by a going out of (P, Q) and there is a transition labelled by b going out of (P, Q). This proves that X is the full tree shift.

One can overcome this difficulty with the notion of Fischer automaton for irreducible tree-shifts.

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

Note that the definition of the transitions is consistent. For any block u of X, there is a finite computation of C on u rooted by a state of $\mathrm{cont}_X(u)$. Indeed, let c be the finite tree which has the same support as u and is defined by $c_x = \mathrm{cont}_X(\mathrm{sub}(u, x))$, where $\mathrm{sub}(u, x)$ is the subblock of u rooted at x. The tree c is a finite computation of C on u since

$$(\mathrm{cont}_X(\mathrm{sub}(u, x0)), \mathrm{cont}_X(\mathrm{sub}(u, x1))) \xrightarrow{u_x} \mathrm{cont}_X(\mathrm{sub}(u, x))$$

is a transition of $\mathcal{C}$. Furthermore, note that if S set of states ending computations of $\mathcal{C}$ on u, then $\mathrm{cont}_X(u) = \cup_{\mathrm{cont}_X(v) \in S} \mathrm{cont}_X(v)$.

The context tree automaton is the minimal (uncomplete) deterministic tree automaton of the finite tree language of patterns of X. It can be computed by applying the Myhill-Nerode minimization algorithm to any deterministic tree automaton accepting the language (see [10, Section 1.5]).

As is shown below, this tree automaton also accepts all infinite trees of X although it may contain states which are not reachable by any infinite computation.

Lemma 1 *Let X be a tree-shift and $\mathcal{A}$ be a tree automaton accepting all blocks of X. Then $\mathcal{A}$ accepts X.*

Proof Let t be a tree of X. We denote by u_n the subblock of t of height n rooted at ε. Since u_n is accepted by $\mathcal{A}$, there is a finite computation c_n of $\mathcal{A}$ of X on u_n. Since the number of states of $\mathcal{A}$ is finite, by taking an infinite subsequence of $(c_n)_{n \geq 1}$, we may assume that all computations share the same root. For any $k \geq 1$, by taking again a subsequence, we may assume that the root of all computations is a subblock v_k of height k. We define an infinite computation c such that the subblock of c of height k at the root is v_k. Then c is a computation of $\mathcal{A}$ on t. Hence t is accepted by $\mathcal{A}$.

Proposition 6 *The context tree automaton of a tree-shift X accepts X.*

Proof Since any block in $\mathcal{B}(X)$ is accepted by the context tree automaton of X, any infinite tree is also accepted by Lemma 1. Conversely, if there is a computation of the context tree automaton of X on a tree t, any block of t belongs to $\mathcal{B}(X)$. Thus t is in X.

Proposition 7 *The context tree automaton of a sofic tree-shift is synchronized.*

Proof Let $\mathcal{C}$ be the context tree automaton of a sofic tree-shift X. There is a finite computation c_1 in $\mathcal{C}$ on some block u_1 whose root belongs to $\mathrm{cont}_X(u_1)$. Let us assume that u_1 is not a synchronizing block of $\mathcal{C}$. Hence there is another computation c_2 of $\mathcal{C}$ on u_1 whose root belongs to $\mathrm{cont}_X(u_2)$ for some block u_2 such that $\mathrm{cont}_X(u_2) \neq \mathrm{cont}_X(u_1)$. 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 $\mathcal{C}$, there is another computation c_2 of $\mathcal{C}$ on u_2 whose root belongs to $\mathrm{cont}_X(u_3)$. We get $\mathrm{cont}_X(u_3) \subsetneq \mathrm{cont}_X(u_2)$. By iterating this process, we either get a synchronizing block for $\mathcal{C}$ or an infinite strictly decreasing sequence of contexts. Hence, since the number of contexts is finite, there is a synchronizing block.

We say that a tree automaton $\mathcal{A}' = (V, A', \Delta')$ is a *subautomaton* of a tree automaton $\mathcal{A} = (V, A, \Delta)$ if $V' \subseteq V$ and $\Delta' = \Delta \cap (V' \times V' \times A \times V')$. A subautomaton $\mathcal{A}'$ is *minimal* if any transition of $\mathcal{A}$ going out of states of $\mathcal{A}'$ ends in $\mathcal{A}'$. A *minimal irreducible component* of a tree automaton is a minimal subautomaton which is irreducible.

Proposition 8 *Let X be an irreducible sofic tree-shift and $\mathcal{C}$ its context automaton. Let z be a synchronizing block of $\mathcal{C}$ and $\mathcal{F}$ be the tree automaton obtained from $\mathcal{F}$ by keeping only the states accessible from* $\mathrm{cont}_{\mathcal{C}}(z)$*. The tree automaton $\mathcal{F}$ is the unique minimal irreducible component of $\mathcal{C}$.*

Proof We assume that the synchronizing block z is of height n. Let $q = \mathrm{cont}_{\mathcal{C}}(z)$. Let us show that $\mathcal{F}$ is a minimal irreducible subautomaton. We first prove that there is a hyperpath from any state of $\mathcal{C}$ to q. Let $p = \mathrm{cont}_{\mathcal{C}}(u)$ be a state of $\mathcal{C}$. Since X is irreducible, there is a pattern w of X and a finite complete prefix code $P \subset \Sigma^{\geq n}$, 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 $c_\varepsilon = q$, and for all $x \in P$, $c_x = p$. Hence there is an hyperpath from p to q. This implies that $\mathcal{F}$ is a minimal irreducible subautomaton of $\mathcal{C}$.

Let $\mathcal{G}$ be another minimal irreducible subautomaton of $\mathcal{C}$. Since there is an hyperpath from any state of $\mathcal{G}$ to any state of $\mathcal{F}$ and $\mathcal{F}, \mathcal{G}$ are both minimal, then $\mathcal{G} = \mathcal{F}$.

We define the *Fischer automaton*[4] of an irreducible sofic tree-shift X as the unique minimal irreducible component of its context tree automaton.

Note that each state in this tree automaton is the context of some synchronizing block of the context tree automaton. Furthermore, any computation of $\mathcal{C}$ on a synchronizing block of $\mathcal{C}$ ends in the Fischer automaton.

The Fischer automaton is irreducible and synchronized. It is also reduced since each of its state is some $\mathrm{cont}_X(u)$ for some pattern of X.

[4] also called the Shannon automaton of the tree-shift. It corresponds to the right Fischer automaton of a sofic shift of sequences.

Proposition 9 *The Fischer automaton of an irreducible tree-shift X accepts X.*

Proof Since the Fischer automaton $\mathcal{F}$ of X is a subautomaton of the context tree automaton, any tree accepted by $\mathcal{F}$ is in X. Conversely, let t be a tree of X. We show that any block u of t of height n is accepted by $\mathcal{F}$. Let z be a synchronizing block of $\mathcal{C}$. Since X is irreducible, there is a pattern w of X and a finite complete prefix code $P \subset \Sigma^{\geq n}$, such that u is a subtree of w rooted at ε, and z is a subtree of w rooted at x for all $x \in P$. It follows that there is computation of $\mathcal{C}$ on w. Since any computation of $\mathcal{C}$ on a synchronizing block ends in $\mathcal{F}$, and since $\mathcal{F}$ is a minimal subautomaton of $\mathcal{C}$, we obtain a computation of $\mathcal{F}$ on u. Thus u is accepted by $\mathcal{F}$. By Lemma 1, all trees of X are accepted by $\mathcal{F}$.

We now prove that the Fischer automaton is the unique reduced deterministic irreducible and synchronized tree automaton accepting an irreducible sofic tree-shift.

Proposition 10 *Two reduced deterministic irreducible and synchronized tree automata accepting the same irreducible sofic tree-shift are equal up to a renaming of the states.*

Proof Let $\mathcal{F} = (V, A, \Delta)$ and $\mathcal{F}' = (V', A, \Delta')$ be two reduced deterministic irreducible synchronized automata accepting a same sofic tree-shift X. Let $u \in \mathcal{B}_m(X)$ (resp. $v \in \mathcal{B}_m(X)$) be a synchronizing block of $\mathcal{F}$ (resp. $\mathcal{F}'$). We assume that u focuses to the state p in $\mathcal{F}$. Since X is irreducible, there is a pattern z of X and a finite complete prefix code $P \subset \Sigma^{\geq m}$, such that u is a subtree of z rooted at ε, and that v is a subtree of z rooted at x for all $x \in P$. Since $\mathcal{F}$ and $\mathcal{F}'$ are deterministic, the pattern z is a synchronizing block for both $\mathcal{F}$ and $\mathcal{F}'$. Each computation of $\mathcal{F}$ on z focuses to the state p while each computation of $\mathcal{F}'$ on z focuses to some state p' in V'.

We define a bijection $\varphi : V \to V'$ as follows. Let q be a state of $\mathcal{F}$. Since $\mathcal{F}$ is irreducible, there is an hyperpath from p to q labelled by w. For all positions x of the support of w such that xi with $i \in \{0, 1\}$ is not in this support, we add the pattern z to w at the position xi. We denote by w' the new pattern obtained. Since $\mathcal{F}$ is deterministic and z synchronizing for $\mathcal{F}$, the root of any finite computation of $\mathcal{F}$ on w' belongs to q. Hence the context of q in $\mathcal{F}$ is the context of w' in X. Moreover, the root of any finite computation of $\mathcal{F}'$ on w' is a same state q' and the context of q' in $\mathcal{F}'$ is the context of w' in X.

Since $\mathcal{F}$ (resp. $\mathcal{F}'$) is reduced, two states having the same context in $\mathcal{F}$ (resp. $\mathcal{F}'$) are equal. Hence if $q = \mathrm{cont}_X(w')$, $r = \mathrm{cont}_X(v)$, and $(q, r) \xrightarrow{a} s$ is a transition of Δ, we have $s = \mathrm{cont}_X(a, w, v)$. We define $\varphi(q) = q'$. One can check that φ defines an isomorphism between $\mathcal{F}$ and $\mathcal{F}'$ in the sense that $(q, r) \xrightarrow{a} s$ is a transition of Δ if and only if $(\varphi(q), \varphi(r)) \xrightarrow{a} \varphi(s)$ is a transition of Δ'.

Hence two reduced deterministic irreducible and synchronized tree automata accepting a same tree-shift are equal to the Fischer automaton of this tree-shift.

If $t \in \mathcal{T}$ and ℓ a context. Let ℓt denotes the pattern obtained where the hole is replaced by the infinite tree t. If ℓt is an (infinite) pattern of an infinite tree of X, we say that ℓ is a *context of* t *in* X. Given a tree t in $\mathcal{T}$, we denote by $\mathrm{cont}_X(t)$ the set of all the contexts of t in X. The *Krieger automaton* of a tree-shift X is the tree automaton whose states are the nonempty sets of the form $\mathrm{cont}_X(t)$, and whose transitions are $(\mathrm{cont}_X(t), \mathrm{cont}_X(t')) \xrightarrow{a} \mathrm{cont}_X(a, t, t'))$, where $t, t' \in \mathcal{T}$ and $\mathrm{cont}_X(a, t, t')$ is nonempty. Note the consitency of the definition of the transitions. Hence the states of the Krieger automaton are contexts of infinite trees while the states of the context automaton are contexts of finite trees.

Proposition 11 *The Krieger automaton of a sofic tree-shift X accepts X.*

Proof It is clear that a tree of X is accepted by the Krieger automaton of X. Conversely, if there is a computation the Krieger automaton of X on a tree t, any block of t belongs to $\mathcal{B}(X)$. Thus t is in X.

Proposition 12 *The Krieger automaton of a sofic tree-shift X is, up to an isomorphism, a subautomaton of the context tree automaton of X. It is reduced and synchronized. If X is irreducible, the Krieger automaton of X has a unique minimal irreducible subautomaton which is the Fischer automaton of X.*

Proof Let $\mathcal{K} = (V, A, \Delta)$ be the Krieger automaton of a sofic tree-shift X and $\mathcal{C} = (V', A, \Delta')$ its context tree automaton. Let t be an infinite tree and u_n its subblock of height n rooted at ε. We have $\mathrm{cont}_X(u_{i+1}) \subseteq \mathrm{cont}_X(u_i)$ for any $i \geq 0$. Since the number of sets $\mathrm{cont}_X(v)$, where v is a pattern of X is finite, there exists a nonnegative integer n such that $\mathrm{cont}_X(u_{n+i}) = \mathrm{cont}_X(u_n)$ for all nonnegative integers i. Let us define $s(t) = \mathrm{cont}_X(u_n)$.

We show that the map $\mathrm{cont}_X(t) \mapsto s(t)$ is well defined and one to one. We have $s(t) = \mathrm{cont}_X(t)$. Indeed, clearly $\mathrm{cont}_X(t) \subseteq s(t)$. Conversely, let $\ell \in s(t)$ with a marked leaf x. There is a nonnegative integer n such that $\ell u_i \in \mathcal{L}(X)$ for all $i \geq n$. Let s_i be an infinite tree such that ℓu_i is a subtree of s_i rooted at ε, for all $i \geq n$. By compacity of X, there is a subsequence of $(s_i)_{i \geq n}$ which converges to an infinite tree s of X. Then the infinite pattern ℓt is a pattern of s, and thus $\ell \in \mathrm{cont}_X(t)$. As a consequence, if t, t' are two infinite trees, then $\mathrm{cont}_X(t) = \mathrm{cont}_X(t')$ if and only if $s(t) = s(t')$. Let now $\delta = ((s(t), s(t') \xrightarrow{a} \mathrm{cont}_X(a, s(t), s(t'))$ be a transition of $\mathcal{C}$ starting from states of $\mathcal{K}$. We have $\mathrm{cont}_X(a, s(t), s(t')) = s(a, t, t')$ and thus δ is a transition of $\mathcal{K}$. Thus $\mathcal{K}$ is a minimal subautomaton of $\mathcal{C}$.

We next show that $\mathcal{K}$ is reduced. If p is a state equal to $\mathrm{cont}_X(t)$ for some tree $t \in \mathcal{T}$, then $\mathrm{cont}_{\mathcal{K}}(p) = \mathrm{cont}_X(t)$. As a consequence, $\mathcal{K}$ is reduced.

We now show that $\mathcal{K}$ is synchronized. By Proposition 7, the context tree automaton is synchronized. Hence, by definition, there is a block u of X which is synchronizing for $\mathcal{C}$. Let t be an infinite tree such that u is a subtree of t rooted at ε. Since u is a synchronizing block of $\mathcal{C}$, we have $\mathrm{cont}_X(u) = \mathrm{cont}_X(t)$. Thus $\mathcal{K}$ is synchronized.

Let us now assume that X is irreducible. The Krieger automaton contains a state $\mathrm{cont}_X(t) = \mathrm{cont}_X(u)$, where u is a synchronizing block of $\mathcal{C}$, belonging to

the Fischer automaton $\mathcal{F}$ of X. Since $\mathcal{K}$ and $\mathcal{F}$ are are minimal subautomaton of $\mathcal{C}$ and $\mathcal{F}$ is irreducible, $\mathcal{F}$ is a minimal subautomaton of $\mathcal{K}$. Since $\mathcal{F}$ is the unique minimal irreducible component of $\mathcal{C}$, it is also the unique minimal irreducible component of $\mathcal{K}$.

We have considered two automata associated with a sofic tree-shift, the context-tree automaton and the Krieger automaton. If the tree-shift is irreducible, we also defined the Fischer automaton which is a subautomaton of the Krierger. Note that the Krieger automaton is not simply the subautomaton of the context automaton in which all states that cannot appear in an infinite run are deleted. Indeed, this situation may appear already for shifts of infinite sequences which are tree-shifts with arity 1 (see [5]). The context tree automaton provides a natural way for computing the Fischer automaton of an irreducible sofic tree-shift.

3.6 Tree-shifts of finite type

Tree-shifts of finite type are accepted by tree automata having very strong synchronization properties.

Let m be a nonnegative integer. A *deterministic m-local tree automaton* (or an *m-definite tree automaton*) is a deterministic tree automaton $\mathcal{A} = (V, A, \delta)$ such that whenever there are two finite computations c, c' of $\mathcal{A}$ on a same block of height m, then $c_\varepsilon = c'_\varepsilon$ (the computations share the same root). A tree automaton is *local* (or *definite*) if it is m-local for some nonnegative integer m (and m stands for memory). For instance a transition tree-shift is accepted by a 1-local automaton.

The following proposition was shown in [3].

Proposition 13 *A deterministic local tree automaton accepts a tree-shift of finite type. Conversely, a tree-shift of finite type is accepted by a deterministic tree local automaton.*

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. Almost of finite tree-shifts are irreducible and accepted by tree automata which are deterministic and codeterministic with a finite delay. The class is between the class of of irreducible tree-shifts of finite type and the classe of irreducible sofic tree-shifts and both inclusions are strict. As shown below, almost of finite type tree-shifts are also factors of irreducible tre-shifts of finite type by a biclosing map.

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 $0 \leq i \leq (a + m)$, and $s_x = t_x$ for all $x \in \Sigma^i$ with $0 \leq i \leq m - 1$, then $s_x = t_x$

for all $x \in \Sigma^m$. Note that the trees have to be pictured horizontally with their root on the right side to see the left closing property.

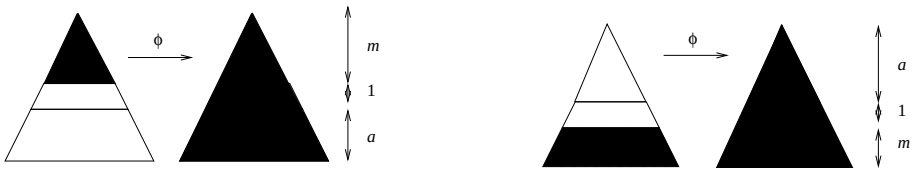


Fig. 3 (left) The notion of left closing map. (right) The notion of right closing map.

A tree automaton $\mathcal{A} = (V, A, \Delta)$ is *left closing* if any two computations of $\mathcal{A}$ on a same tree and with the same root, are equal. Equivalently, there is a nonnegative integer a such any two finite computations c, c' of $\mathcal{A}$ on a same block $u \in \mathcal{B}_a(X)$ such that $c_\varepsilon = c'_\varepsilon$ satisfy $c_x = c'_x$ for all $x \in \{0, 1\}$.

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 $0 \leq i \leq (a + m)$, and $s_x = t_x$ for all $x \in \Sigma^i$ with $a + 2 \leq i \leq (a + m)$, then $s_x = t_x$ for all $x \in \Sigma^{a+1}$. The map Φ is *right resolving* if it is a one-block map and if one can choose $m = 1$ and $a = 0$.

We now introduce the notion of resolving block (see [21]). Let Φ be a 1-block map from X to Y. A *resolving block* of Φ is a block z of Y of height h such that if $\phi(u) = \phi(v) = z$ where u and v are two blocks of X of height h and ϕ is the block function of Φ extended to the blocks, then $u_\varepsilon = v_\varepsilon$.

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 one-block map which is right resolving, left closing, and has a resolving block.

The tree-shift of Example 1 is an almost of finite type tree-shift. The tree-shift of trees such that any path of the tree is a path of the automaton of Figure 4.

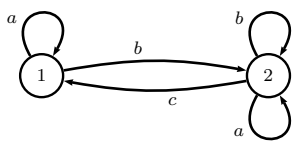


Fig. 4 A finite automaton defining the constraint of a tree-shift which is not almost of finite type.

Proposition 14 *Let S and T be two conjugate irreducible sofic tree-shifts. Then S is AFT if and only if T is AFT.*

Proof We assume that S is AFT. Let Θ be a one-block map from an irreducible tree-shift of finite type X onto a sofic tree-shift S, which is right resolving, left

closing, and has a resolving block. Since X is irreducible, a tree of X can be extended up and one may assume that Θ is left closing with memory $m = 1$ and anticipation a. Let Φ be a conjugacy from S to T. We can assume that Φ is an m block map with an n-block inverse. Let $X^{(n+m)}$ (resp. $S^{(n+m)}$) be the higher block presentation of X (resp. S) of order $n+m$, and τ (resp. τ') be the natural conjugacy from X to $X^{(n+m)}$ (resp. from S to $S^{(n+m)}$). The tree-shift $X^{(n+m)}$ is an irreducible tree-shift of finite type. The one-block map Θ extends in a natural way to a one-block map Θ' from $X^{(n+m)}$ to $S^{(n+m)}$. Let $\Theta' = \Theta \circ \tau'^{(-1)}$ and $\Psi = \Theta' \circ \Phi'$. One can check that the map Ψ' is a one-block map which is right resolving. One can check that it is left closing with memory 1 and anticipation $a + m + 2n$, and that it has a resolving block. Indeed, if u is a resolving block of Θ of height h which is the top of a tree t of S, then one can choose, as a resolving block of Ψ, a block of height $\max(h, n+m) + a + n$ at the top of the tree $t' = \Phi(t)$. Thus the tree-shift T is AFT.

$$\begin{tikzcd}
X \arrow[r, "\tau"] \arrow[d, "\Phi"'] & X^{(n+m)} \arrow[d, "\Phi'"'] \arrow[dr, "\Psi"] & \\
S \arrow[r, "\tau'"] & S^{(n+m)} \arrow[r, "\Theta'"] & T
\end{tikzcd}$$

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

Proposition 15 *A tree-shift is AFT if and only if it is accepted by an AFT tree automaton.*

Proof Let S be an AFT tree-shift on the alphabet A. Let $\Phi : X \to S$ be a one-block map from an irreducible tree-shift of finite type X onto S which is right resolving, left closing and has a resolving block.

Since X is irreducible, a tree of X can be extended up (*i.e.* is a child of another one) in X. Thus, without loss of generality, we may assume that Φ is left closing with parameters $m = 1$ and a.

Let $X = \mathsf{X}_{\mathcal{F}}$ where $\mathcal{F}$ is a set of forbidden blocks of height n for X. Let $\mathcal{A} = (V = \mathcal{B}_{n-1}(X), E, \Delta)$ where Δ contains the transitions $(u, v, e, (e, u', v')$, with $u, v \in \mathcal{B}_{n-1}(X)$, $(e, u, v) \in \mathcal{B}_n(X)$, and (e, u', v') is the block of height $n-1$ at the top of (e, u, v). The tree automaton $\mathcal{A}$ is deterministic and accepts X. Let $\mathcal{B} = (V, A, \Delta')$ where $(u, v, \phi(e), w) \in \Delta'$ if and only if $(u, v, e, w) \in \Delta$. The tree automaton $\mathcal{B}$ accepts S. Since $\mathcal{A}$ is irreducible, $\mathcal{B}$ also.

Since Φ is a one-block right resolving map, $\mathcal{B}$ is a deterministic tree automaton. Indeed, if $u, v, \phi(e)$ are known, there is at most one symbol e such that $\phi(e, u_\varepsilon, v_\varepsilon) = (\phi(e), \phi(u_\varepsilon), \phi(v_\varepsilon))$ and thus at most one block (e, u', v') whose image by ϕ is $(\phi(e), \phi(u'), \phi(v'))$.

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 having the

same root u. These computations define two distinct trees s, s' of X such that $s_\varepsilon = s'_\varepsilon = u_\varepsilon$ and such that $\Phi(s) = \Phi(s') = t$. Since Φ is left closing with parameters $m = 1$, we get $s = s'$ and thus $c = c'$.

If Φ has a resolving block z of height h and is left closing with parameters $m = 1$ and a, we define a block z' of S of height $h + a$ extending z. If $\phi(w) = \phi(w') = z'$, where w, w' are blocks of X of height $h + a$, then $w_x = w'_x$ for $x \in \Sigma^{\leq (h-1)}$. Thus z' is a synchronizing block of $\mathcal{B}$.

Conversely, if S is accepted by an AFT tree automaton $\mathcal{A} = (V, A, \Delta)$, let $X_\mathcal{A}$ be the irreducible transition tree-shift of $\mathcal{A}$ accepted by $\mathcal{T} = (V, \Delta, \Delta')$, whose transitions are $(p, q) \xrightarrow{(p,q,a,r)} r$ for $(p, q, a, r) \in \Delta$. Let $\Phi_\mathcal{A}$ be the one-block map from $X_\mathcal{A}$ 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 has a resolving block since $\mathcal{A}$ is synchronized. As a consequence, S is an AFT.

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

Proof Let S be an irreducible sofic tree-shift. By Proposition 9, any irreducible sofic tree-shift is accepted by an irreducible deterministic synchronized tree automaton $\mathcal{F}$ equal to the Fischer automaton of S.

Let us assume that S is accepted by an AFT tree automaton $\mathcal{A}$. Since $\mathcal{A}$ and $\mathcal{F}$ are irreducible and synchronized, there is a same block z which is synchronizing for both. Morover, the context of each state in each tree automaton is the context of some block tree in S.

Since $\mathcal{A}$ and $\mathcal{F}$ are irreducible and deterministic, it follows that for each state p of $\mathcal{A}$, there is a state $f(p)$ in $\mathcal{F}$ which has the same context. Moreover if $(p, q) \xrightarrow{a} r$ is a transition of $\mathcal{A}$ then $(f(p), f(q)) \xrightarrow{a} f(r)$ is a transition of $\mathcal{F}$. If $(p', q') \xrightarrow{a} r'$ is a transition of $\mathcal{F}$, there is a transition $(p, q) \xrightarrow{a} r$ of $\mathcal{A}$ such that $p' = f(p)$, $q' = f(q)$ and $r' = f(r)$.

Let us show that $\mathcal{F}$ is left closing. Let c, c' be two computations of $\mathcal{F}$ on t with a same root p. Since $\mathcal{F}$ is irreducible, there is an hyperpath from p to q labelled by some pattern w such that z is a subtree of w rooted at ε. Let P be the complete prefix code defining this hyperpath. Thus there are two distinct computations d, d' of $\mathcal{F}$ on a tree s such that, for any node x in P, the tree rooted at x in s is t, $d_x = p$, and z is a block rooted at the root of s. Let e, e' be two computations of $\mathcal{A}$ on s such that $f(e_x) = d_x$ and $f(e'_x) = d'_x$ for any $x \in \Sigma^*$. These computations have the same root since z is synchronizing for $\mathcal{A}$. Since $\mathcal{A}$ is left closing, these computations are equal. As a consequence, for any $x \in \Sigma^*$, $f(e_x) = f(e'_x)$, implying $d_x = d'_x$ and thus $c_y = c'_y$ for any $y \in \Sigma^*$. This proves that the Fischer automaton of an AFT is left closing.

Conversely, if $\mathcal{F}$ is AFT, then X is AFT by Proposition 15.

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

Proof One can compute the Fischer automaton of the sofic tree-shift and check whether it is AFT as explained in Section 6.

5 Conjugacy invariants

In the case of one-dimensional shifts, Krieger proved that the Fischer automaton is a topological conjugacy invariant of the underlying irreducible sofic shift [18] (see also [8]). In this section, we extend Krieger's theorem to sofic tree-shifts.

Lemma 2 *Let $\mathcal{A} = (V, A, \Delta)$ be the Fischer automaton of the irreducible sofic tree-shift $S_{\mathcal{A}}$ with transition tree-shift $X_{\mathcal{A}}$. Let $\Phi_{\mathcal{A}} : X_{\mathcal{A}} \to S_{\mathcal{A}}$ the one-block map assigning to each transition its label. Let us assume that there is a block map Θ such that the following diagram commutes.*

$$\begin{array}{ccc} X_{\mathcal{A}} & \xrightarrow{\Theta} & X_{\mathcal{A}} \\ {\scriptstyle \Phi_{\mathcal{A}}}\downarrow & & \downarrow{\scriptstyle \Phi_{\mathcal{A}}} \\ S_{\mathcal{A}} & \xrightarrow{id} & S_{\mathcal{A}} \end{array}$$

Then Θ is the identity map.

Proof Let u be a synchronizing block of height n of $S_{\mathcal{A}}$. Let z be a block of height n of $X_{\mathcal{A}}$ such that $\Phi_{\mathcal{A}}(z) = u$. Let Z be the subset of $X_{\mathcal{A}}$ containing the trees t of $X_{\mathcal{A}}$ such that

- z is a subtree of t rooted at ε;
- for any node $x \in \Sigma^*$, there is a finite complete prefix code $P \subset \Sigma^{\geq n}$ depending on x such that, for anyany $y \in P$, z is a subtree of t rooted at the position xy.

Note that Z is not a tree-subshift of $X_{\mathcal{A}}$. It is clear that $\Phi_{\mathcal{A}}$ is one-to-one on Z. Indeed, the condition on the occurrences of synchronized blocks implies that there is only one run on the trees in Z. Hence Θ is one-to-one on Z. Let us assume that Θ is an m-block map which is not the identity map on $X_{\mathcal{A}}$. Then there is a block v of height m such that $\theta(v) \neq v_\varepsilon$. Since $X_{\mathcal{A}}$ is irreducible, the block v is a subblock of some tree in Z, which is a contradiction.

Proposition 16 *Let $\mathcal{A} = (V, A, \Delta)$ (resp. $\mathcal{B} = (V', B, \Delta')$) be the Fischer automaton of the irreducible sofic tree-shift $S_{\mathcal{A}}$ (resp. $S_{\mathcal{B}}$). Let $X_{\mathcal{A}}$ (resp. $X_{\mathcal{B}}$) be the transition tree-shift of $\mathcal{A}$ (resp. $\mathcal{B}$). If $S_{\mathcal{A}}$ and $S_{\mathcal{B}}$ are conjugate, then $\mathcal{A}$ and $\mathcal{B}$ are symbolic conjugate. As a consequence $X_{\mathcal{A}}$ and $X_{\mathcal{B}}$ are conjugate.*

Proof There is one-block and onto map $\Phi_{\mathcal{A}} : X_{\mathcal{A}} \to S_{\mathcal{A}}$ (resp. $\Phi_{\mathcal{B}} : X_{\mathcal{B}} \to S_{\mathcal{B}}$). Let Φ be a conjugacy from $S_{\mathcal{A}}$ to $S_{\mathcal{B}}$. We may assume that Φ is an m-block map and Φ^{-1} is an n-block map with $m+n \geq 2$. Since $\mathcal{A}$ is deterministic, the map Φ_A is a one-block right resolving map. Let $X_{\mathcal{A}}^{(m+n-1)}$ (resp. $S_{\mathcal{A}}^{(m+n-1)}$) be the higher block presentation of $X_{\mathcal{A}}$ (resp. $\mathcal{S}_{\mathcal{A}}$) of order $(m+n-1)$, and $\tau_{\mathcal{A}}$ (resp. $\mu_{\mathcal{A}}$) be the $(m+n-1)$-block conjugacy from $X_{\mathcal{A}}$ to $X_{\mathcal{A}}^{(m+n-1)}$ (resp. from $S_{\mathcal{A}}$ to $S_{\mathcal{A}}^{(m+n-1)}$) assigning to each tree t the tree t' whose nodes are

blocks of height $m+n-1$ of t. Let $\Phi'_{\mathcal{A}}$ be the natural one-block map such that $\Phi'_{\mathcal{A}} \circ \tau_{\mathcal{A}} = \mu_{\mathcal{A}} \circ \Phi_{\mathcal{A}}$.

Let $\Phi' = \mu_{\mathcal{A}}^{(-1)} \circ \Phi$. Since Φ is an m-block map, the map $\Psi = \Phi' \circ \Phi'_{\mathcal{A}}$ is a one-block map from $X_{\mathcal{A}}^{(m+n-1)}$ to $S_{\mathcal{B}}$. Since $\Phi^{(-1)}$ is an n-block map, Ψ is right resolving.

Let $\mathcal{C} = (\mathcal{B}(X_{\mathcal{A}}^{(m+n-2)}), B, \Theta)$ be the tree automaton on the alphabet B whose transitions are $(u,v), b \to w$, where $(w_\varepsilon, u, v) \in \mathcal{B}(X_{\mathcal{A}}^{(m+n-1)})$, w is the subblock of size $m+n-2$ rooted at ε of (w_ε, u, v), and $\psi(w_\varepsilon, u, v)) = b$. Since Ψ is right resolving, $\mathcal{C}$ is a deterministic tree automaton accepting $S_{\mathcal{B}}$.

One can check that the tree automaton $\mathcal{C}$ is also irreducible and synchronized. Thus, we consider the partitioning of the states of $\mathcal{C}$ into classes of states having the same context in $\mathcal{S}_{\mathcal{B}}$. We denote by $[u]$ the class of the state u. We define a one-block and onto map Γ from $X_{\mathcal{A}}^{(m+n-1)}$ to $X_{\mathcal{B}}$ by $\gamma((w_\varepsilon, u, v) = ([u],[v]) \xrightarrow{b} [w]$, where $b = \psi(w_\varepsilon, u, v))$. We have $\Phi_{\mathcal{B}} \circ \Gamma = \Psi$. Hence the block map $\theta_1 = \Gamma \circ \tau_{\mathcal{A}}$ is onto from $X_{\mathcal{A}}$ to $X_{\mathcal{B}}$. We get the following commutative diagram.

$$\begin{tikzcd}
X_{\mathcal{A}} \arrow[r, "\tau_{\mathcal{A}}"] \arrow[d, "\Phi_{\mathcal{A}}"'] & X_{\mathcal{A}}^{(m+n-1)} \arrow[r, "\Gamma"] \arrow[d, "\Phi'_{\mathcal{A}}"'] \arrow[dr, "\Psi"] & X_{\mathcal{B}} \arrow[d, "\Phi_{\mathcal{B}}"] \\
S_{\mathcal{A}} \arrow[r, "\mu_{\mathcal{A}}"] & S_{\mathcal{A}}^{(m+n-1)} \arrow[r, "\Phi'"] & S_{\mathcal{B}}
\end{tikzcd}$$

By exchanging the roles played by $\mathcal{A}$ and $\mathcal{B}$, we also get a block map $\theta_2 : X_{\mathcal{B}} \to X_{\mathcal{A}}$ such that following diagram commutes.

$$\begin{tikzcd}
X_{\mathcal{A}} \arrow[r, "\theta_1"] \arrow[d, "\Phi_{\mathcal{A}}"'] & X_{\mathcal{B}} \arrow[r, "\theta_2"] \arrow[d, "\Phi_{\mathcal{B}}"'] & X_{\mathcal{A}} \arrow[d, "\Phi_{\mathcal{A}}"] \\
S_{\mathcal{A}} \arrow[r, "\Phi' \circ \mu_{\mathcal{A}} = \Phi"] & S_{\mathcal{B}} \arrow[r, "\Phi'^{-1} \circ \mu_{\mathcal{B}} = \Phi^{-1}"] & S_{\mathcal{A}}
\end{tikzcd}$$

By Lemma 2, $\theta_2 \circ \theta_1$ is the identity map. As a consequence the maps θ_1 and θ_2 are conjugacies.

Proposition 16 provides a conjugacy invariant for irreducible sofic tree-shifts. Indeed, if two irreducible sofic tree-shifts are conjugate, the underlying tree-shifts of finite type defined by their Fischer automata are also conjugate. Note that it is unknonw whether two irreducible sofic tree-shifts are conjugate since the problem is already open for one-sided sofic shifts of sequences.

6 The algorithmic issue

In this section, we design algorithms to check whether a tree automaton is synchronized, and whether it is left closing. We also describe an algorithm to

compute the Fischer automaton of an irreducible sofic tree-shift given by a tree automaton that accepts it.

6.1 Computation of the Fischer automaton

Let $\mathcal{A} = (V, A, \Delta)$ be a tree automaton. We define the deterministic tree automaton $\mathcal{D}(\mathcal{A})$, called the (uncomplete) *determinized tree automaton* of $\mathcal{A}$, as the accessible part from the state V of the tree automaton $(\mathfrak{P}(V), A, \Delta')$ where, for $P, Q \in \mathfrak{P}(V)$, $(P, Q) \xrightarrow{a} R$ if the set R of states r such that $(p, q) \xrightarrow{a} r \in \Delta$ for some $p \in P$, $q \in Q$ is nonempty.

Proposition 17 *It can be checked in polynomial time whether a tree automaton is irreducible.*

Proof Let $\mathcal{A} = (V, A, \Delta)$ be a tree automaton. For any state p in V, and any positive integer n, we define the sets

$$P_0 = \{p\},$$
$$P_n = P_{n-1} \cup \{q \in V \mid \exists a \in A, r, s \in P_{n-1} \text{ such that } (r, s) \xrightarrow{a} q \in \Delta\}.$$

The sequence of subsets $(P_n)_{n \geq 0}$ is increasing for the inclusion. Thus there is an integer n_0 such that $P_n = P_{n_0}$ for any $n \geq n_0$. By construction, there is an hyperpath from p to q if and only if $q \in P_{n_0}$. Thus $\mathcal{A}$ is irreducible if and only if $P_{n_0} = V$ for any state p. The set P_{n_0} is computed with at most $|V|$ steps, the time complexity of each step being at most the number of transitions. Hence the global time complexity is $O(|V| \times |\Delta|)$.

Proposition 18 *It is decidable whether a tree automaton is synchronized.*

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

Proposition 19 *Let $\mathcal{A} = (V, A, \Delta)$ be a tree automaton accepting an irreducible sofic tree-shift S. The Fischer automaton of S is computable from $\mathcal{A}$.*

Proof We first compute the determinized tree automaton $\mathcal{D}(\mathcal{A})$ of $\mathcal{A}$. Let R be a state of $\mathcal{D}(\mathcal{A})$ which is minimal for the inclusion. Let u the label of a hyperpath from V to R. Then u is a synchronizing pattern of $\mathcal{D}(\mathcal{A})$. Indeed, any finite computation of $\mathcal{D}(\mathcal{A})$ on u has its root in R by minimality of R. We now keep in $\mathcal{D}(\mathcal{A})$ only the states accessible from the state R and get an irreducible and synchronized tree automaton accepting S. Its minimization gives the Fischer automaton of S by Proposition 9. It is computed in polynomial time in the size of $\mathcal{D}(A)$ (see for instance [10]).

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

6.2 The pair graph of a tree automaton

The algorithm to check whether a tree automaton is AFT basically consists in computing the productive states of the product tree automaton of the tree automaton. This algorithm is known in principle and is used to check the emptiness of tree languages (see [23] and [28]). We give below a direct algorithm for the sake of completeness.

Given a tree automaton $\mathcal{A} = (V, A, \Delta)$, we define the *square tree automaton* of $\mathcal{A}$, denoted by $\mathcal{A} \times \mathcal{A} = (V \times V, A, \Delta')$, as the deterministic tree automaton whose transitions are $((p, p'), (q, q')) \xrightarrow{a} (r, r')$ if and only if $(p, q) \xrightarrow{a} r$ and $(p', q') \xrightarrow{a} r'$ 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 [25, p. 647]) are used to check properties of pairs of paths. This notion of square tree automaton of a tree automaton, together with a notion a pair graph, is used to check properties of pairs of computations. We used it in [3] to check properties of locality. Seidl [26] used branch automata to check the degree of ambiguity of finite tree automata.

Proposition 20 *A tree automaton is not left closing if and only if there is a computation in the square tree 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 labelled by 0 or 1 and a letter from A. For more convenience, an edge labelled by 1 is noted by a plain arrow $\longrightarrow$ and is called a plain edge, and an edge labelled by 0 is noted by a dashed arrow $\dashrightarrow$ and is called a dashed edge. For each pair of transitions $(p, q) \xrightarrow{a} r$ and $(p', q') \xrightarrow{a} r'$ of $\mathcal{A}$,

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

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

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

When $\mathcal{A}$ is a deterministic tree automaton, we have $|V_G| = O(|V|^4)$ and $|E_G| = O(|V|^6)$. The essential part of the pair graph can be computed in time $O(|V_G| + |E_G|)$ as described in [3].

It is easy to verify that a vertex $((p, p'), (q, q'))$ is a vertex of the (essential part of the) pair graph if and only if there are two computations of $\mathcal{A}$ on a tree s one ending in p, the other one in p', and there are two computations of $\mathcal{A}$ on a tree t one with a root q, the other one with the root q'.

Note also that there is an edge $((p, p'), (q, q')) \xrightarrow{0} ((r, r'), (s, s'))$ in the pair graph if and only if there is a letter a and transitions $(p, q) \xrightarrow{a} r$ and $(p', q') \xrightarrow{a} r'$ in $\mathcal{A}$ (or transitions $(p, q) \xrightarrow{a} s$ and $(p', q') \xrightarrow{a} s'$ in $\mathcal{A}$). There is an edge $((p, p'), (q, q')) \xrightarrow{0} (r, r')$ in the pair graph if and only if there is a letter a and transitions $(p, q) \xrightarrow{a} r$ and $(p', q') \xrightarrow{a} r'$ in $\mathcal{A}$.

Proposition 21 *A tree automaton is not left closing if and only its there is in its pair graph an edge from a non diagonal vertex and ending to a vertex* (p, p).

Proof Let $\mathcal{A}$ be a tree automaton and $G_{\mathcal{A}}$ its pair graph. Suppose that there is a path in $G_{\mathcal{A}}$ from vertex $((p, p'), (q, q'))$ with $p \neq p'$ or $q \neq q'$ to a vertex (r, r). Then there are two computations c, c' of $\mathcal{A}$ on a same tree t such that the root of c is p and the root of c' is p', and there are two computations d, d' of $\mathcal{A}$ on a same tree s such that the root of d is q and the root of d' is q'. There are also transitions $(p, q) \xrightarrow{a} r$ and $(p', q') \xrightarrow{a} r'$. This implies that there are two distinct computations of $\mathcal{A}$ on a same tree whose root is r. Thus $\mathcal{A}$ is not left closing.

Conversely, if $\mathcal{A}$ is not left closing, there are two distinct computations c, c' of $\mathcal{A}$ on a same tree t sharing a same root r. Let x be a node of the tree t such that $c_x \neq c'_x$. Then there is path in $G_{\mathcal{A}}$ labelled by (x, w) going from some vertex $((c_x, c'_x), (s, s'))$ or some vertex $((s, s'), (c_x, c'_x))$ (depending on the last letter of x) to (r, r), where w is is the label of the path of the tree t from the root to the node x. The existene of such a path implies the existence of an edge going from a non digonal vertex to a diagonal one. Hence the conclusion.

If $\mathcal{A}$ is a deterministic tree automaton, 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 21 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 Fischer automaton of an irreducible sofic tree-shift is AFT. Note that Seidl's check of the finite degree of ambiguity of tree automata in [26] 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 5.

Conclusion

In this article, we have shown that sofic tree-shifts differ from one-sided sofic shifts of infinite sequences at least concerning the following property: there

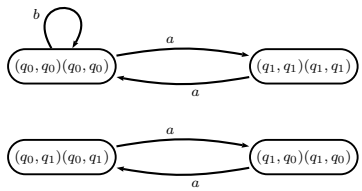


Fig. 5 The pair graph for the tree automaton of Example 2. A thick edge represents a plain edge and a dashed edge with the same label. Each vertex (p, q) is identified with the vertex $(p, q)(p, q)$. The tree-shift S accepted by $\mathcal{A}$ satisfies the property of Proposition 21 since there is no edge from a non diagonal vertex to diagonal one. Then $\mathcal{A}$ is a left closing tree automaton and as a consequence the tree-shift S is AFT. It is not a tree-shift of finite type since there is a loop around a non-diagonal state (see [3]).

may be more than one reduced deterministic irreducible tree automata accepting the same irreducible sofic tree-shift. The reason is that reduced tree automata may not have a synchronizing block. For irreducible sofic tree-shifts, the Fischer automaton remedy for this lack. There is also a natural extension of the class of almost of finite type shifts of sequences to tree-shifts.

Acknowledgements The authors would like to thank the anonymous reviewers for their valuable comments and suggestions.

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. N. Aubrun and M.-P. Béal. Sofic and almost of finite type tree-shifts. In *CSR*, volume 6072 of *Lecture Notes in Computer Science*, pages 12–24. Springer, 2010.
3. N. Aubrun and M.-P. Béal. Tree-shifts of finite type. *Theor. Comput. Sci.*, 459:16–25, 2012.
4. T. Bates, S. Eilers, and D. Pask. Reducibility of covers of AFT shifts. *Israel J. Math.*, 185:207–234, 2011.
5. M.-P. Béal, J. Berstel, S. Eilers, and D. Perrin. Symbolic dynamics. *CoRR*, abs/1006.1265, 2010.
6. M.-P. Béal, F. Fiorenzi, and D. Perrin. A hierarchy of shift equivalent sofic shifts. *Theor. Comput. Sci.*, 345(2-3):190–205, 2005.
7. M.-P. Béal and D. Perrin. Symbolic dynamics and finite automata. In *Handbook of formal languages, Vol. 2*, pages 463–505. Springer, Berlin, 1997.
8. M. Boyle, B. Kitchens, and B. Marcus. A note on minimal covers for sofic systems. *Proc. Amer. Math. Soc.*, 95(3):403–411, 1985.
9. T. Ceccherini-Silberstein, M. Coornaert, F. Fiorenzi, and Z. Sunic. Cellular automata on regular rooted trees. In *CIAA*, pages 101–112, 2012.
10. H. Comon, M. Dauchet, R. Gilleron, C. Löding, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. Tree automata techniques and applications. http://www.grappa.univ-lille3.fr/tata, 2007. release October, 12th 2007.
11. E. Coven, A. Johnson, N. Jonoska, and K. Madden. The symbolic dynamics of multidimensional tiling systems. *Ergodic Theory Dynam. Syst.*, 23(02):447–460, 2003.
12. G. Fici and F. Fiorenzi. Topological properties of cellular automata on trees. In *DCM*, pages 255–266, 2012.

13. M. Fujiwara and M. Osikawa. Sofic systems and flow equivalence. *Math. Rep. Kyushu Univ.*, 16(1):17–27, 1987.
14. G. Hedlund. Endomorphisms and automorphisms of the shift dynamical system. *Theory of Computing Systems*, 3(4):320–375, 1969.
15. 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.
16. N. Jonoska and B. Marcus. Minimal presentations for irreducible sofic shifts. *IEEE Trans. Inform. Theory*, 40(6):1818–1825, 1994.
17. B. P. Kitchens. *Symbolic Dynamics.* Universitext. Springer-Verlag, Berlin, 1998. One-sided, two-sided and countable state Markov shifts.
18. W. Krieger. On sofic systems. I. *Israel J. Math.*, 48(4):305–330, 1984.
19. D. Lind and K. Schmidt. Symbolic and algebraic dynamical systems. In *Handbook of Dynamical Systems, Vol. 1A*, pages 765–812. North-Holland, Amsterdam, 2002.
20. D. A. Lind and B. H. Marcus. *An Introduction to Symbolic Dynamics and Coding.* Cambridge University Press, 1995.
21. B. H. Marcus. Sofic systems and encoding data. *IEEE Trans. Inform. Theory*, 31(3):366–377, 1985.
22. M. Nasu. *Textile Systems for Endomorphisms and Automorphisms of the Shift.* American Mathematical Society, 1995.
23. J. Neumann, A. Szepietowski, and I. Walukiewicz. Complexity of weak acceptance conditions in tree automata. *Inform. Process. Lett.*, 84(4):181–187, 2002.
24. D. Perrin and J. Pin. *Infinite words.* Elsevier Boston, 2004.
25. J. Sakarovitch. *Elements of Automata Theory.* Cambridge University Press, 2009.
26. 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.
27. W. Thomas. Automata on infinite objects. In *Handbook of Theoretical Computer Science, Vol. B*, pages 133–191. Elsevier, Amsterdam, 1990.
28. W. Thomas. Languages, automata, and logic. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages, Vol. III*, pages 389–455. Springer, New York, 1997.
29. J. Verdú-Mas, R. Carrasco, and J. Calera-Rubio. Parsing with probabilistic strictly locally testable tree languages. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(7):1040–1050, 2005.
30. 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.
31. S. Williams. Covers of non-almost-finite type sofic systems. *Proc. Amer. Math. Soc.*, 104(1):245–252, 1988.