Tree-shifts of finite type

Nathalie Aubrun, Marie-Pierre Béal

Tree-shifts of finite type

Nathalie Aubrun and Marie-Pierre Béal *

Abstract

A one-sided (resp. two-sided) shift of finite type of dimension one can be described as the set of infinite (resp. bi-infinite) sequences of consecutive edges in a finite-state automaton. While the conjugacy of shifts of finite type is decidable for one-sided shifts of finite type of dimension one, the result is unknown in the two-sided case.

In this paper, we study the shifts of finite type defined by infinite ranked trees. Indeed, infinite ranked trees have a natural structure of symbolic dynamical systems. We prove a Decomposition Theorem for these tree-shifts, *i.e.* we show that a conjugacy between two tree-shifts can be broken down into a finite sequence of elementary transformations called in-splittings and in-amalgamations. We prove that the conjugacy problem is decidable for tree-shifts of finite type. This result makes the class of tree-shifts closer to the class of one-sided shifts of sequences than to the class of two-sided ones. Our proof uses the notion of bottom-up tree automata.

1 Introduction

Sofic shifts are bi-infinite sequences labeling paths in a finite automaton. Shifts of finite type are a particular important subclass of sofic shifts. Two-sided (resp. one-sided) shifts of finite type are bi-infinite (resp. right-infinite) sequences of consecutive edges in a finite-state automaton (see [12, 13.8], [9]). They are well understood in the one-sided case since the conjugacy is decidable for such shifts of finite type [20]. The proof uses the Decomposition Theorem (see for instance [9]). This theorem states that every conjugacy between two one-sided shifts of finite type can be decomposed into a finite sequence of splittings and amalgamations, which are elementary operations on automata presenting the two shifts.

In the two-sided case, the decidability of the conjugacy problem between two shifts of finite type is still an open question. For class of shifts of sequences larger than the class of shifts of finite type, like sofic shifts, the problem is also open [6]. In higher dimension, many questions become more difficult. The main reason is that there exists no good representation of multidimensional shifts comparable to finite automata in dimension one. Even if there exists a generalization of finite automata to dimension two, which are called textile systems (see [14], see also the automata for tiling systems in [5]), results are more complex than in dimension one. The Decomposition Theorem can be extended to two-sided

*Université Paris-Est, CNRS, Laboratoire d'informatique Gaspard-Monge, 5 boulevard Descartes, 77454 Marne-la-Vallée, France.

multidimensional shifts of finite type, but an additional operation, called an inversion, is needed (see [8], and also [1]).

In this paper, we introduce the notion of shifts of finite type defined on infinite ranked trees, that we call tree-shifts. Indeed, infinite ranked 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 can be described thanks to top-down or bottom-up tree automata which are used in automata theory for many purposes. Tree automata have applications to logic and game theory (see for instance [19], [4], [15], and [16]). The tree automata that we consider here are bottom-up tree automata. They are simpler than Büchi or Muller tree automata since they have all their states final.

We define two elementary operations on tree automata: the in-splitting operation and the in-amalgamation operation. They are very close to those existing on finite (word) automata. In particular two in-amalgamations commute. We prove a Decomposition Theorem for tree-shifts, *i.e.* we show that a conjugacy between two tree-shifts can be broken down into a finite sequence of in-splittings and in-amalgamations. We then prove that the conjugacy problem is decidable for the class of tree-shifts of finite type. The key of the proof is the commutation property of in-amalgamations. We prove that two tree-shifts of finite type are conjugate if and only if they have a same minimal in-amalgamation. Furthermore, the minimal in-amalgamation of a tree automaton can be computed in a polynomial time in the number of states of the automaton.

The paper is organized as follows. In Section 2 we give basic definitions about tree-shifts and tree automata. We also present the pair graph of a tree automaton and give a polynomial-time algorithm for checking the locality of a deterministic tree automaton. The Decomposition Theorem is proved in Section 3. Our main result, the decidability of the conjugacy, together with an example are given in Section 4. We end the paper with some concluding remarks. A preliminary version of this paper was presented in [2].

2 Shifts, automata and infinite trees

2.1 Tree-shifts

We give here some basic definitions from symbolic dynamics which apply to infinite trees. We consider infinite trees whose nodes have a fixed number of children and are labeled in a finite alphabet.

Let $\Sigma = \{0, 1, \ldots, d-1\}$ be a finite alphabet of cardinal d. 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 sometimes denote $t(x)$ by t_x. A *path* in a tree t is a sequence $(t_{x_n})_{n\geq 0}$ where $x_0 = \varepsilon$ and $x_n \in \Sigma x_{n-1}$ for any $n \geq 0$.

When Σ is fixed, we denote by $\mathcal{T}(A)$ the set of all infinite trees on A, hence the set A^{Σ^*}. On this set we have a natural metric. If t, t' are two trees, we define the distance $d(t, t') = \frac{1}{n+1}$, 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 $d(t, t) = 0$. This metric induces a

topology equivalent to the usual product topology, where the topology in A is the discrete one.

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 any $x \in \Sigma^*$. The set $\mathcal{T}(A)$ equipped with these shift transformations is called the *full shift* of infinite trees over A.

A *pattern* is a function $p : L \to A$, where L is a finite prefix closed subset[1] of Σ^* (hence containing the empty word). The set L is called the *support of the pattern*. The elements of L are called the *nodes* of the pattern. A *block* of height n is a pattern with support $\Sigma^{\leq n-1}$, where n is some positive integer, and $\Sigma^{\leq n}$ denotes the words of length at most n with letters in Σ.

We say that a pattern (resp. a block) p of support L is a *pattern of a tree* (resp. *block of a tree*) t if there is a node $x \in \Sigma^*$ such that $t_{xy} = p_y$ for any $y \in \Sigma^*$. We say that p is a pattern (or block) of t rooted at the node x. If p is not a pattern (or block) of t, one says that t *avoids* p. If p is a pattern (or block) of some tree of tree-shift X, it is called an *allowed pattern* (or *allowed block*) of X.

We define a *tree-shift* (or *tree-subshift*) X of $\mathcal{T}(A)$ as the set $\mathsf{X}_\mathcal{F}$ of all trees avoiding each pattern of a set of blocks $\mathcal{F}$. This tree-shift X is closed and for any shift transformation σ_i, $\sigma_i(X) \subseteq X$. A *tree-shift of finite type* (TSFT) 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 allowed blocks of a tree-shift X, and by $\mathcal{L}_n(X)$ the set of allowed blocks of height n of X. If b is a block of height n with $n \geq 2$, we denote by $\sigma_i(b)$ the block of height $n-1$ such that $\sigma_i(b)_x = b_{ix}$ for $x \in \Sigma^{\leq n-2}$. The block b will be written $b = (b_\varepsilon, \sigma_0(b), \ldots, \sigma_{d-1}(b))$. A tree shift of finite type $\mathsf{X}_\mathcal{F}$, where $\mathcal{F}$ is a finite set of blocks of height at most 2, is called a *Markov tree-shift*.

In the sequel, in order to simplify the notations, we restrict us to binary trees ($\Sigma = \{0, 1\}$) but all results extend trivially to the case of trees with d children for any $d \geq 1$.

Example 1. In figure 1 is pictured an infinite tree of a tree-shift of finite type $X = \mathsf{X}_\mathcal{F}$ on the binary alphabet $\{0, 1\}$ defined by a finite set $\mathcal{F}$ of forbidden blocks of height 2. The forbidden blocks are those whose label sum is equal to 1 modulus 2.

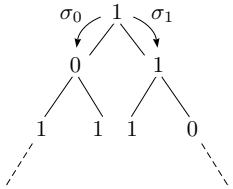


Figure 1: A tree of the tree-shift of finite type $X = \mathsf{X}_\mathcal{F}$ on the alphabet $\{0, 1\}$, where $\mathcal{F}$ is the set of blocks of height 2 whose label sum is 1 modulus 2.

Example 2. In figure 2 is pictured an infinite tree of a tree-shift on the binary alphabet $\{a, b\}$. A block is forbidden if it contains a path with an even number of a between two b.

[1] any prefix of a word of L belongs to L.

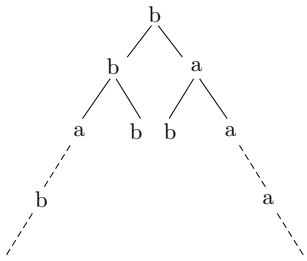


Figure 2: A tree of the tree-shift $Y = \mathsf{X}_{\mathcal{F}'}$ on the alphabet $\{a, b\}$, where $\mathcal{F}'$ is the set of blocks containing a path with an even number of a between two b.

Let A, A' be two finite alphabets, X be a tree-shift of $\mathcal{T}(A)$ and m be a nonnegative 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 $\phi : \mathcal{L}_m(X) \to A'$ such that, for any $x \in \Sigma^*$, $\Phi(t)_x = \phi(b)$, where b is the block of height m of t rooted at x. The smallest integer m satisfying this property is called the *memory* of the block map. A *block map* is a map which is m-local for some nonnegative integer m.

It is known from the Curtis-Lyndon-Hedlund theorem (see [7]) 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 any $t \in X$ and any $i \in \Sigma$. The image of X by a block map is also a tree-shift. 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. It is called a *conjugacy* from X onto Y. The tree-shifts X and Y are then said *conjugate*.

Example 3. Let X the tree-shift of finite type defined in Example 1. Let Y be the tree-shift of finite type over the alphabet $\{a, b, c\}$, where the allowed blocks of height 2 are (a, a, a), (a, b, c), (a, c, b), (a, c, c), (b, b, a), (b, c, a), (c, a, b) and (c, a, c). The 2-block map $\Phi : X \to Y$, defined by $\phi(0, 0, 0) = a$, $\phi(0, 1, 1) = a$, $\phi(1, 1, 0) = b$, and $\phi(1, 0, 1) = c$, is pictured in Figure 3. The map Φ is a conjugacy. Its inverse is a 1-block map Ψ defined by $\psi(a) = 0$ and $\psi(b) = \psi(c) = 1$.

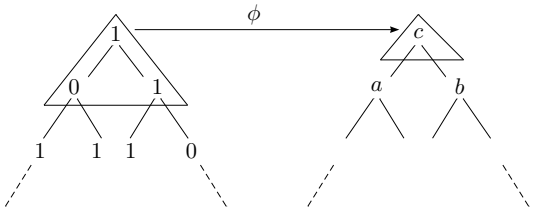


Figure 3: A 2-block map $\Phi : X \to Y$, where X is the tree-shift of Figure 1 and Y a tree-shift of finite type over the alphabet $\{a, b, c\}$. The map Φ is a conjugacy.

Let X be a tree-shift on the alphabet A. Let n be a positive integer. The *higher block presentation* of order n of X is a tree-shift on the alphabet $\mathcal{L}_n(X)$ defined as follows. It is the set of trees t for which there is a tree t' in X such that, for any node x in Σ^*, t_x is the block of height n of t' rooted at x. It is easy to show that a tree-shift is conjugate to any of its higher block presentations.

2.2 Tree automata

In this section we consider bottom-up automata for infinite trees where each node has d children.

A computation in such an automaton goes 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 (or vertices), A is a finite set of input symbols, and Δ is a set of transitions of the form $(q_0, .., q_{d-1}), a \to q$, with $q, q_i \in V$, $a \in A$. A transition $(q_0, .., q_{d-1}), a \to q$ is called a transition *labeled by* a, *going out of* the d-tuple of states $(q_0, .., q_{d-1})$ and *coming in* the state q. Note that no initial nor final states are specified. This means that all states are both initial and final in the setting of classical tree automata (see for instance [4], [16]).

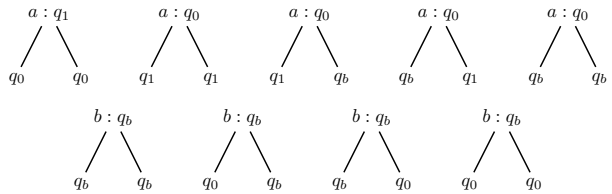
Such an automaton is *deterministic* if for all d-tuple of states $(q_0, .., q_{d-1})$ and for all $a \in A$, there is at most one transition $(q_0, .., q_{d-1}), a \to q$. Then the set of transitions defines a partial function δ from $V^d \times A$ to V. In that case, the automaton is also denoted by $\mathcal{A} = (V, A, \delta)$.

A (bottom-up) *finite computation* of $\mathcal{A}$ on a pattern p is a pattern c on the alphabet V such that, for each node x of p, there is a transition $(c_{x0}, .., c_{x(d-1)}, t_x)$ $\to c_x \in \Delta$. A pattern p is accepted by $\mathcal{A}$ if there is a if there is a finite computation of $\mathcal{A}$ on p.

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 $(c_{x0}, .., c_{x(d-1)}, t_x) \to c_x \in \Delta$. 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. It is equal to $\mathsf{X}_{\mathcal{F}}$, where $\mathcal{F}$ is the set of blocks which are not accepted by $\mathcal{A}$. Indeed, the set of trees accepted by $\mathcal{A}$ is clearly included in $\mathsf{X}_{\mathcal{F}}$. The converse holds by a compacity argument.

Example 4. The tree-shift of Example 2 is accepted by the tree-automaton with three states q_b, q_0, q_1 whose transitions are the followings.



Let m be a nonnegative integer. An *m-local deterministic tree automaton* (or an *m-definite tree automaton*) is a tree automaton $\mathcal{A} = (V, A, \delta)$ such that whenever t and t' are two trees accepted by $\mathcal{A}$ with a same block b of height m rooted at the root ε of t and t', for any computation c of $\mathcal{A}$ on t and any computation c' of $\mathcal{A}$ on t', we have $c_\varepsilon = c'_\varepsilon$. Hence the memory of height m determines the state reached. We say that the block b *focuses* in $\mathcal{A}$ to the state c_ε. A tree automaton is *local* (or *definite*) if it is m-local for some nonnegative integer m.

Proposition 1. *Any tree-shift of finite type is accepted by a deterministic local tree automaton. Conversely any tree-shift accepted by a deterministic local tree automaton is of finite type.*

Proof. Let $X = \mathsf{X}_{\mathcal{F}}$ be a tree-shift of finite type defined by a finite set of forbidden blocks. Without loss of generality, we can assume that $\mathcal{F}$ is the set of all forbidden blocks of height m for some integer $m \geq 2$. We also assume that the arity d of the trees is 2.

We define a deterministic tree automaton $\mathcal{A} = (V, A, \delta)$ such that $V = \mathcal{L}_{m-1}(X)$. For $p_0, p_1 \in V$, $a \in A$, if the block $q = (a, p_0, p_1)$ of height m is an allowed block of X, then $\delta((p_0, p_1), a) = b$, where b is the block of q of height $m-1$ rooted at the node ε. The partial function δ is undefined otherwise. By construction, the automaton $\mathcal{A}$ is deterministic and $(m-1)$-local. It clearly accepts the tree shift X, which proves the first part of the claim.

Let X be a tree-shift and $\mathcal{A} = (V, A, \delta)$ an m-local deterministic tree automaton accepting X. We define $\mathcal{F}$ as the set of forbidden blocks of height $m+1$ of X. One immediately get $X \subseteq \mathsf{X}_{\mathcal{F}}$. Suppose now that $t \in \mathsf{X}_{\mathcal{F}}$. We define a computation c of $\mathcal{A}$ on t as follows. For any $x \in \Sigma^*$, we set c_x as the state focused in $\mathcal{A}$ by the block of height m of t rooted at x. Let b be the block of height $m+1$ of t rooted at x. Since $t \in \mathsf{X}_{\mathcal{F}}$, b is an allowed block of some tree t' in X rooted at some at a node y. Let c' be the computation of $\mathcal{A}$ on t'. We have $\delta(c'_{y0}, c'_{y1}, t'_y) = c'_y$. Since $\mathcal{A}$ is m-local, $c_{xi} = c'_{yi}$ for $0 \leq i \leq 1$, $t_x = t'_y$ and $c_x = c'_y$. It follows that $\delta(c_{x0}, c_{x1}, t_x) = c_x$. Hence c is a computation of $\mathcal{A}$ of t and $t \in X$, which proves the second part of the claim. □

We give below a decision procedure to check whether a deterministic tree automaton is local.

Given a 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 tree 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 [17, p. 647]) are used to check properties of pairs of paths. We extend this notion, together with a notion of pair graph, to trees, to check the locality of tree automata. Seidl [18] uses the notion of branch automata to check some properties on tree automata like the degree of ambiguity of finite tree automata.

Proposition 2. *A deterministic tree automaton is local if and only if there is a computation in its square automaton ending in a non diagonal state.*

Proof. By definition of the square tree automaton $\mathcal{A} \times \mathcal{A}$, the existence of a computation in $\mathcal{A} \times \mathcal{A}$ ending in a state (p, q) with $p \neq q$ implies the existence of two distinct computations of $\mathcal{A}$ on a same tree. Conversely, if there are two distinct computations of $\mathcal{A}$ on a same tree t, they differ at some node $x \in \Sigma^*$. Hence we get two computations of the subtree of t rooted at x ending in two distinct states. □

In order to check the above property, we define the notion of pair graph of a tree automaton. We give the definition for binary trees. Let $\mathcal{A} = (V, A, \Delta)$ be a automaton. The *pair graph* $G = (V_G, E_G)$ 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 $(p',q'), a \to r'$ and each pair (s,s') of states of $\mathcal{A}$, the set of edges E_G contains the following edges

$$((p,p'),(q,q')) \overset{a}{\dashrightarrow} ((r,r'),(s,s')),$$
$$((p,p'),(q,q')) \xrightarrow{a} ((s,s'),(r,r')),$$
$$((p,p'),(q,q')) \overset{a}{\dashrightarrow} (r,r'),$$
$$((p,p'),(q,q')) \xrightarrow{a} (r,r').$$

The A-labels of the edges of G may be removed in order to reduce the complexity of the graph.

A vertex of G is *useful* 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 useful, together with their incoming and outgoing edges. A vertex $((p,q),(r,s))$ (resp. (p,q)) of G is called *non diagonal* if either $p \neq q$ or $r \neq s$ (resp. $p \neq q$).

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 ending in q, the other one in 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 3. *A deterministic tree automaton $\mathcal{A}$ is local if and only if its pair graph contains no non diagonal vertex (r,r').*

Proof. Let G be the pair graph of $\mathcal{A}$. Any vertex $((r,r')(s,s'))$ is in the pair graph if and only if there are two computations c, c' of $\mathcal{A}$ on a same tree t such that c ends in r and c' ends in r', and there are two computations d, d' of $\mathcal{A}$ on a same tree t' such that d ends in s and d' ends in s'. Similarly, a vertex (r,r') is in the pair graph if and only if there are two computations c, c' of $\mathcal{A}$ on a same tree t such that c ends in r and c' ends in r'.

Now $\mathcal{A}$ is not local if and only if there are two computations c (resp. c') of $\mathcal{A}$ on a same tree ending in a state r (resp. ending in a state $r' \neq r$). This is equivalent to the fact that (r,r') is a vertex of G by the above remarks. □

When $\mathcal{A}$ is a deterministic 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 polynomial time as described below. We call 0-predecessor a predecessor of a vertex by a dashed edge (labeled by 0) and 1-predecessor a predecessor of a vertex by a plain edge (labeled by 1). The notions of 0-sucessors and 1-sucessors are defined similarly. We build a queue of vertices called *vertexQueue* containing vertices which have to removed from the pair graph. As soon as such a vertex is removed, one also removes its outgoing edges. A pseudo code of the algorithm is described below. Its time complexity is $O(|V_G| + |E_G|)$.

```
ESSENTIAL(pair graph G_A)
 1  Let n_0(u) be the number of 0-predecessors of the vertex u ∈ V_G
 2  Let n_1(u) be the number of 1-predecessors of the vertex u ∈ V_G
 3  vertexQueue ← the list of vertices u such that n_0(u) = 0 or n_1(u) = 0.
 4  mark the vertices contained in vertexQueue
 5  while vertexQueue is nonempty
 6        do remove the vertex u from vertexQueue
 7           for all 0-successors v de u
 8                 do decrement n_0(v)
 9                    if n_0(v) = 0 and v is unmarked
10                       then add v in vertexQueue
11                            mark v
12           for all 1-successors v de u
13                 do decrement n_1(v)
14                    if n_1(v) = 0 and v is unmarked
15                       then add v in vertexQueue
16                            mark v
17  return the unmarked states
```

As a consequence, it can be checked in $O(|V|^6)$ time whether a deterministic tree automaton is local.

Example 5. A part of the pair graph for the tree automaton of Example 4. It contains the non diagonal state (q_1, q_0) hence the tree automaton is not local.

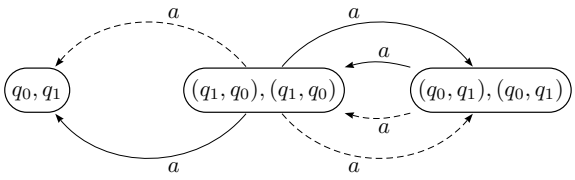


Figure 4: A part of the pair graph for the tree automaton of Example 4.

2.3 Vertex tree-shifts

In this section we consider a particular case of tree shifts of finite type, called vertex tree-shifts.

A *vertex tree-shift* is the tree shift accepted by an automaton $\mathcal{A} = (V, V, \Delta)$ where the transitions have the form $(q_0, q_1), q \to q$. Hence the label of an accepted tree at each node is equal to the corresponding state of the computation. We simplify this setting by saying that a vertex tree-shift is the set of computations of the unlabeled automaton $\mathcal{B} = (V, \Gamma)$ with transitions in $V^2 \times V$ denoted by $(q_0, q_1) \to q$. Note that a vertex shift is a Markov shift and is thus of finite type.

Example 6. The tree-shift X of Example 1 is a vertex tree-shift accepted by the automaton $\mathcal{A} = (V, \Delta)$ with transitions $(0,0) \to 0$, $(1,1) \to 0$, $(1,0) \to 1$ and $(0,1) \to 1$. These transitions are given in the following table t where $(p, q) \to t[p, q]$ is a transition.

	0	1
0	0	1
1	1	0

Proposition 4. *Any tree-shift of finite type is conjugate to a vertex tree-shift.*

Proof. Let $X = \mathsf{X}_{\mathcal{F}}$ be a tree-shift of finite type defined by a finite set of forbidden blocks of height $m+1$ for some nonnegative integer m. Let $\mathcal{A} = (V, A, \delta)$ be the deterministic m-local automaton defined by $V = \mathcal{L}_m(X)$ and, for $p_0, p_1 \in V, a \in A, \delta(p_0, p_1, a)$ is the subblock of height m of $b = (a, p_0, p_1)$ rooted at ε when b is an allowed block of height $m+1$ of X. The automaton $\mathcal{A}$ accepts X. Let Y be the vertex tree-shift made of all computations on $\mathcal{A}$. Note that $\mathcal{A}$ has a unique computation on any tree shift of X.

We define an m-block map Φ from X to Y via $\phi : \mathcal{L}_m(X) \to Y$ by setting $\phi(p) = p$. The map Φ associates to each tree t of X the unique computation of $\mathcal{A}$ on t. The one-block map Ψ from Y to X given by $\psi : Y \to A$ with $\psi(p) = p_\varepsilon$ is the inverse of ϕ. The tree-shifts X and Y are thus conjugate. □

3 Decomposition Theorem

The Decomposition Theorem for shifts of infinite words states that any conjugacy between shifts of finite type can be decomposed into a finite sequence of splittings and amalgamations (see for instance [9]). In this section, we prove an analogous theorem for infinite trees. The crucial lemma will show that the memory of a block map can be reduced using a notion of (input) splittings on tree automata defined below.

Let X be a tree-shift over the alphabet A. For each letter a, we consider a partition $\mathcal{P}_a$ of the subset $\mathcal{L}_2(X)$ of blocks of height 2 labeled by a at their root. Let $[(a, b, c)]_a$ denotes the partition element in $\mathcal{P}_a$ that contains the block (a, b, c), emphasing the fact that it is a class of $\mathcal{P}_a$. Let $\mathcal{P}$ be the partition of $\mathcal{L}_2(X)$ which is the union of all partitions $\mathcal{P}_a$. Let $\Phi : X \to \mathcal{P}^{\Sigma^*}$ be the 2-block map defined by $\phi(a, b, c) = [(a, b, c)]_a$. We denote by $\widetilde{X}$ the tree shift $\Phi(X)$. The map Φ is a 2-block conjugacy whose inverse is a 1-block map. We say that Φ is an *in-splitting map* and that $\widetilde{X}$, and more generally that any shift obtained from $\widetilde{X}$ by renaming symbols, is an *in-splitting* of X. We also say that Φ^{-1} is an *in-amalgamation map* and that X is an *in-amalgamation* of $\widetilde{X}$. When $\mathcal{P}_a$ is the trivial discrete partition for any letter a (*i.e.* $[(a, b, c)]_a = (a, b, c)$ for any a), Φ is called the *complete in-splitting* map and $\widetilde{X}$ is the *complete in-splitting* of X.

A 1-block conjugacy whose inverse is also 1-block is called a *renaming* map. It is just a renaming of symbols.

Theorem 5. *Any conjugacy between tree-shifts can be decomposed as a composition of in-splitting maps and in-amalgamation maps.*

Lemma 6. *Let $\Phi : X \to Y$ be an m-block conjugacy between two tree-shifts with $m \geq 2$. Then there is an in-splitting map Ψ_1 from X to $\widetilde{X}$ and an $(m-1)$-block conjugacy $\widetilde{\Phi}$ from $\widetilde{X}$ onto Y such that $\Phi = \widetilde{\Phi} \circ \Psi_1$.*

Proof. Let us assume that A is the alphabet of the tree-shift X and that B is the alphabet of the tree-shift Y. Let $\phi : \mathcal{L}_m(X) \to B$ be the block function defining Φ. Let $\Psi_1 : X \to \widetilde{X}$ be the 2-block conjugacy defined by $\psi_1(a, b, c) = (a, b, c)$. The map Ψ_1 is a complete in-splitting.

Let f (resp. g, h) be the map from $\mathcal{L}_2(X)$ to A which maps (a, b, c) to a (resp. b, c). Let $\widetilde{\Phi} : \widetilde{X} \to Y$ be the $(m-1)$-block map defined, for any block

b of $\mathcal{L}_{m-1}(\widehat{X})$, by $\widetilde{\phi}(b) = \phi(b')$, where b' is the block of height m defined by $b'_x = f(b_x)$ for any word x of length at most $m-1$ of Σ, and $b'_{x0} = g(b_x)$, and $b'_{x1} = h(b_x)$ for any x of length $m-1$ of Σ. We have $\Phi = \widetilde{\Phi} \circ \Psi_1$. □

Lemma 7. *Let $\Phi : X \to Y$ be a 1-block conjugacy between two tree-shifts such that Φ^{-1} is an m-block map with $m \geq 2$. Then there is an in-splitting map Ψ_1 from X to $\widetilde{X}$, an in-splitting map Ψ_2 from Y to $\widetilde{Y}$, and a 1-block conjugacy $\widetilde{\Phi}$ from $\widetilde{X}$ onto $\widetilde{Y}$ such that $\Phi = \Psi_2^{-1} \circ \widetilde{\Phi} \circ \Psi_1$ and $\widetilde{\Phi}^{-1}$ is an $(m-1)$-block map. This makes the following diagram commute.*

$$\begin{array}{ccc} X & \xrightarrow{\Phi} & Y \\ {\scriptstyle \Psi_1}\downarrow & & \downarrow{\scriptstyle \Psi_2} \\ \widetilde{X} & \xrightarrow{\widetilde{\Phi}} & \widetilde{Y} \end{array}$$

Proof. Let us assume that X (resp. Y) is a tree-shift over the alphabet A (resp. B). Let $\mathcal{P}_a$ be the partition of the blocks of height 2 rooted by a such that two blocks (a,b,c) and (a,b',c') belong to the same class if and only if $\phi(b) = \phi(b')$ and $\phi(c) = \phi(c')$ where $\phi : A \to B$ is the block function defining Φ. Let $[(a,b,c)]_a$ denotes the class of (a,b,c) in $\mathcal{P}_a$. We denote $\phi(X)$ by $\widetilde{X}$. Let $\Psi_1 : X \to \widetilde{X}$ be the 2-block map defined by $\psi_1(a,b,c) = [(a,b,c)]_a$. Then $\widetilde{X}$ is an in-splitting of X.

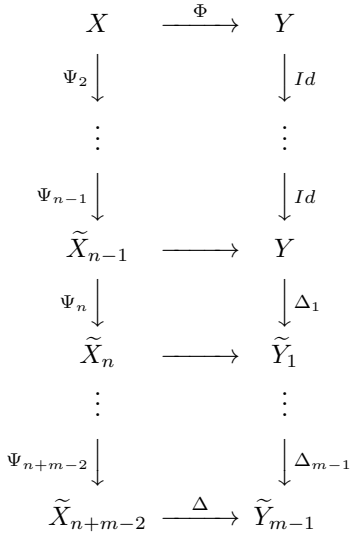
Let B be the alphabet of Y, and $\widetilde{Y}$ be the complete in-splitting of Y over the alphabet $B_2 = \mathcal{L}_2(Y)$. We denote by Ψ_2 the complete in-splitting map from Y to $\widetilde{Y}$.

We define a 1-block map $\widetilde{\Phi}$ from $\widetilde{X}$ onto $\widetilde{Y}$ by $\widetilde{\phi}[(a,b,c)]_a = (\phi(a), \phi(b), \phi(c))$. It is consistent by definition of the partition $\mathcal{P}_a$. We have $\Phi = \Psi_2^{-1} \circ \widetilde{\Phi} \circ \Psi_1$. It remains to check that $\widetilde{\Phi}^{-1} = \Psi_2^{-1} \circ \Phi^{-1} \circ \Psi_1$ is an $(m-1)$-block map. That is, we must show that for any tree t in $\widetilde{Y}$, the coordinates in the block b of height $m-1$ rooted at the node ε of a tree t determines $\widetilde{\Phi}^{-1}(t)_\varepsilon$. But this follows from the observation that the block of height $m-1$ of t rooted at ε determines all $\Psi_2^{-1}(t)_{x0}$ and all $\Psi_2^{-1}(t)_{x1}$, for any word $x \in \Sigma^{m-1}$, and therefore the block of height m at the root ε of $\Psi_2^{-1}(t)$. Hence, if $t' = (\Phi^{-1} \circ \Psi_2^{-1})(t)$, t'_ε is determined by b. Moreover, the block of height $m-1$ of t rooted at ε determines also $\Psi_1(t')_\varepsilon$. □

Proof of Theorem 5. Let $\Phi : X \to Y$ be an n-block conjugacy between two tree-shifts such that ϕ^{-1} is an m-block map, with $n, m \geq 1$.

By Lemma 6 and Lemma 7, there are in-splitting maps $\Psi_1, .., \Psi_{n+m-2}$, $\Delta_1, .., \Delta_{m-1}$, and a renaming map Δ such that $\Phi = \Delta_1^{-1} \circ \Delta_2^{-1} \cdots \circ \Delta_{m-1}^{-1} \circ \Delta^{-1} \circ \Psi_{n+m-2} \circ \Psi_2 \circ \Psi_1$. The result follows from the fact that a renaming map is a particular case of an in-splitting map or of in-amalgamation map. The decomposition into in-splitting maps and in-amalgamation maps is illustrated

in the following diagram.



□

4 Deciding conjugacy for tree-shifts of finite type

In this section we prove that for the particular case of tree-shifts of finite type, the conjugacy problem is decidable. The key point is that for tree shifts of finite type, the in-amalgamation maps commute. We first consider the case of vertex tree-shifts. We will show the following proposition.

Proposition 8. *Suppose X_1 is a vertex tree-shift and X_2, X_3 are vertex tree-shifts obtained from X_1 by in-amalgamations. Then there is a vertex tree-shift X_4 that can be obtained from both X_3 and X_4 by in-amalgamations and such the following diagram commutes.*

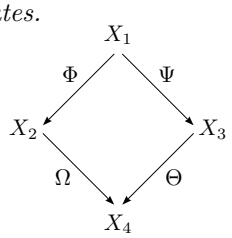


Figure 5: The commutation of in-amalgamation maps. If X_2, X_3 are vertex tree-shifts which are in-amalgamations of X_1, then there is a vertex tree-shift X_4 which is a common amalgamation of X_2, X_3.

In Figure 5, the maps Φ and Ψ are in-amalgamation maps. As a consequence of Proposition 8 the maps Ω and Θ are also in-amalgamation maps.

Let X be a vertex tree-shift accepted by an automaton $\mathcal{A} = (V, \Delta)$ and Φ be an in-splitting map from X to $\widetilde{X}$ defined by partitioning the sets Δ_r of transitions coming in p for each vertex r in V. We denote by $\widetilde{V}$ the alphabet of $\widetilde{X}$. It is the union of the sets $\{[(r, p, q)]_r, \mid (p, q) \to r \in \Delta\}$. The vertices $[(r, p, q)]_r$ are called *in-splitted (or splitted) vertices* of r. One says that the vertices $[(r, p, q)]_r$

are *amalgamated (or merged)* to r. Then $\widetilde{X}$ is a vertex tree-shift accepted by the automaton $\widetilde{\mathcal{A}} = (\widetilde{V}, \widetilde{\Delta})$ defined by $([(r,p,q)]_r, [(r',p',q')]_{r'}) \to [(s,r,r')]_s \in \widetilde{\Delta}$ if and only if $(r,r') \to s \in \Delta$.

For more convenience, the vertices of $\widehat{\Delta}$ corresponding to a partitioning of Δ_r are denoted $r^1 \dots, r^{\ell(r)}$. An in-amalgamation of a vertex tree-shift X accepted by $\mathcal{A}$ is a vertex tree-shift Y accepted by $\widetilde{\mathcal{A}}$ such that Y is an in-splitting of X. The vertices $r^1 \dots, r^{l(p)}$ of $\widetilde{\mathcal{A}}$ are amalgamated to the vertex r. Note that whenever $(p',q') \to r^i \in \widetilde{\Delta}$, then $(p',q') \to r^j \notin \widetilde{\Delta}$, for any $p',q' \in \widetilde{V}$ and any $1 \leq i \neq j \leq \ell(r)$. This also implies that $(r^i,p') \to q' \in \widetilde{\Delta}$ if and only if $(r^j,p') \to q' \in \widetilde{\Delta}$, and $(p',r^i) \to q' \in \widetilde{\Delta}$ if and only if $(p',p^j) \to q' \in \widetilde{\Delta}$ for any vertices $p',q' \in \widetilde{V}$ and any $1 \leq i,j \leq \ell(r)$. Roughly speaking, if a vertex r is splitted into $r^1 \dots, r^{\ell(r)}$, the transitions coming in a r are splitted while transitions going out of r are duplicated after the in-splitting.

Proof of Proposition 8. Suppose that X_n is the vertex tree-shift accepted by (V_n, Δ_n) for $n = 1,2,3$. We assume that there is an in-amalgamation $\Phi : X_1 \to X_2$ and a in-amalgamation $\Psi : X_1 \to X_3$. Let us assume that vertices $p^1, \dots, p^{\ell(p)}$ of V_1 are amalgamated to a vertex p of V_2.

By definition of an in-amalgamation, this implies that if $(q',r') \to p^i \in \Delta_1$, then $(q',r') \to p^j \notin \Delta_1$ for any vertices $q',r' \in V_1$ and any $1 \leq i \neq j \leq \ell(p)$. This implies also that $(p^i,q') \to r' \in \Delta_1$ if an only if $(p^j,q') \to r' \in \Delta_1$, and $(q',p^i) \to r' \in \Delta_1$ if and only if $(q',p^j) \to r' \in \Delta_1$ for any vertices $q',r' \in V_1$ and any $1 \leq i,j \leq \ell(p)$.

Suppose also that vertices $q^1, \dots, q^{\ell(q)}$ of V_1 are amalgamated to a vertex q of V_3. Let us first assume that the vertices $p^1, \dots, p^{\ell(p)}$ and $q^1, \dots, q^{\ell(q)}$ are all distinct. We define X_4 as the in-amalgamation of X_2 obtained by amalgamating the vertices $p, q^1, \dots, q^{l(q)}$ to a vertex q. It is also the in-amalgamation of X_3 obtained by amalgamating the vertices $q, p^1, \dots, p^{l(p)}$ to a vertex q.

Let us now assume that $p^1 = q^1, \dots, p^\ell = q^\ell$ for some integer $1 \leq \ell \leq \min(\ell(p), \ell(q))$. This implies that, for any $1 \leq i \leq \ell(p)$, $1 \leq j \leq \ell(q)$, one has $(p^i,q') \to r \in \Delta_1$ (resp. Δ_2) if and only if $(p^j,q') \to r \in \Delta_1$ (resp. Δ_2), and $(q',p^i) \to r \in \Delta_1$ (resp. Δ_2) if and only if $(q',p^j) \to r \in \Delta_1$ (resp. Δ_2).

We define X_4 as the in-amalgamation of X_2 obtained by amalgamating the vertices $p, q^{l+1}, \dots, q^{l(q)}$ to the vertex p. It is also the in-amalgamation of X_3 obtained by amalgamating the vertices $q, p^{l+1}, \dots, p^{l(p)}$ to a vertex p. Hence, if Φ and Ψ are in-amalgamations, then Ω and Θ also. □

The previous theorem allows us to define the notion of *minimal (in)-amalgamation* of a vertex tree-shift X. It is defined as the vertex tree-shift defined by the smallest automaton (in the number of vertices) which is obtained by in-amalgamations of X.

Corollary 9. *Any vertex tree-shift has a unique minimal in-amalgamation.*

Proof. Let us assume that X has two minimal amalgamations X_2 and X_3. By Proposition 8, X_2 and X_3 have a common in-amalgamation Y. By minimality, $Y = X_2 = X_3$. □

We now describe an algorithm for computing the minimal amalgamation of a vertex tree-shift. Let us assume that X is a vertex tree-shift defined by an n-vertex automaton $\mathcal{A} = (V, \Delta)$. We call *out-degree* of $\mathcal{A}$ the maximal number

of transitions $(p,q) \to r$ or $(q,p) \to r$ for all pair of vertices (p,q). Let us assume that this out-degree of is bounded above by some constant k independent of $|V|$. This implies that the number of transitions is at most $k|V|^3$.

We say that two vertices p, q of V are *pre-mergeable* if p and q have no common incoming transitions, *i.e.* for any pair (r,s) of vertices, $(r,s) \to p \in \Delta$ implies $(r,s) \to q \notin \Delta$, and $(r,s) \to q \in \Delta$ implies $(r,s) \to p \notin \Delta$. We call *signature* of a vertex p the sequence of triples $(0,r,s)$ such that $(p,r) \to s \in \Delta$ concatenated with the sequence of triples $(1,r,s)$ such that $(r,p) \to s \in \Delta$. The triples are sorted in lexicographic order. One can merge two vertices p and q if and only if they are pre-mergeable and $\sigma(p) = \sigma(q)$.

For each pair (r,s), the number the transitions going out of (r,s) is at most k. Whenever this number is greater than or equal to 2, any pair of states ending such transitions are not pre-mergeable. Thus, one can compute in time $O(k^2|V|^2)$ a table of size $|V|^2$ giving, for any pair of states (p,q), whether p and q are pre-mergeable or not.

We deduce from this characterization an algorithm for computing the minimal amalgamation of a vertex tree-shift given by an automaton $\mathcal{A} = (V, \Delta)$. The length of a signature is at most $2k|V|$. It follows that the signatures of all vertices may be computed in time $O(k|V|^2)$. They may then be lexicographically sorted in time $O(k|V|^2)$.

Hence finding (and merging) two vertices which can be merged can be computed in time $O(k^2|V|^2)$. This step being done at most $|V|$ times, the overall complexity for computing the minimal amalgamation is $0(k^2|V|^3)$.

Theorem 10. *Let X_1 and X_2 be two tree-shifts of finite type. It is decidable whether X_1 and X_2 are conjugate.*

Proof. By Proposition 4, one may assume that X_1 and X_2 are vertex tree-shifts. By Theorem 5, there is a sequence a sequence of in-splittings from X_1 to X followed (up to a relabeling of X) of in-amalgamations from X_1 to X_2. This case is illustrated in Figure 6. By Proposition 8, there are vertex tree-shifts at the confluence of any two dashed edges of Figure 6. As a consequence, X_1 and X_2 have a common amalgamation and thus the same minimal amalgamation. Conversely, if X_1 and X_2 have the same minimal amalgamation, there is a sequence of in-splitting and in-amalgamation maps from X_1 to X_2. □

We deduce from this result an algorithm for deciding the conjugacy of two tree-shifts of finite type X_1, X_2 accepted by two deterministic local automata. By Proposition 4, we build two automata $\mathcal{A}_1 = (V_1, \Delta_1)$ and $\mathcal{A}_2 = (V_2, \Delta_2)$ accepting the vertex tree-shifts Y_1, Y_2 such that X_1 and Y_1 (resp. X_2 and Y_2) are conjugate. We compute the minimal amalgamations Z_1 and Z_2 of Y_1 and Y_2 respectively in polynomial time. Then X_1 and X_2 are conjugate if and only if Z_1 and Z_2 are the same vertex tree-shift up to a renaming of the vertices. This test can be done in an exponential time only, making the whole time complexity of this procedure exponential.

Example 7. Let X_1 and X_2 be two vertex tree-shifts over the alphabet $V = \{a, b, c\}$. The tree-shift X_1 is accepted by $\mathcal{A}_1 = (V, \Delta_1)$ and the tree-shift X_2 is accepted by $\mathcal{A}_2 = (V, \Delta_2)$ where Δ_1 and Δ_2 are given in the two following

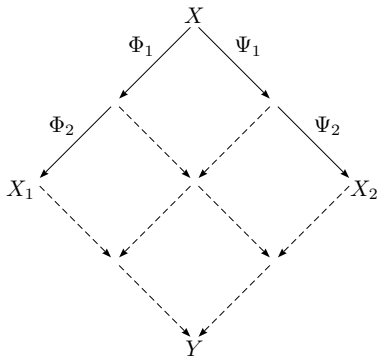


Figure 6: A sequence of tree in-splittings from X_1 to X is followed (up to a relabeling of X), by a sequence of tree in-amalgamations from X to X_2. Any edge represents an in-amalgamation. The tree-shifts X_1 and X_2 have the same minimal amalgamation Y.

tables. If t is a table, $(p, q) \to t[p, q]$ is a transition.

$\Delta_1 =$

	a	b	c
a	a	c	c
b	b	a	a
c	b	a	a

$\Delta_2 =$

	a	b	c
a	c	a	a
b	a	b	b
c	a	b	b

$\Delta_3 =$

	a	b
a	a	b
b	b	a

$\Delta_4 =$

	a	b
a	b	a
b	a	b

Since the second and third row of Δ_1 and the second and third column of Δ_1 are equal, the vertices b and c can be amalgamated. There is an in-amalgamation from $\mathcal{A}_1$ to $\mathcal{A}_3 = (V_3, \Delta_3)$ where $V_3 = \{a, b\}$ and Δ_3 is given by the following tables.

No more in-amalgamation is possible from $\mathcal{A}_3$ and thus $\mathcal{A}_3$ is minimal. Similarly, the second and third row of Δ_2 and the second and third column of Δ_2 are equal, the vertices b and c can be amalgamated. There is an in-amalgamation from $\mathcal{A}_2$ to $\mathcal{A}_4 = (V_4, \Delta_4)$ where $V_4 = \{a, b\}$ and Δ_4 is given by the following tables.

Finally, relabeling the states of $\mathcal{A}_4$ by exchanging a and b gives $\mathcal{A}_3$. Hence, X_1 and X_2 have the same minimal amalgamation and are conjugate.

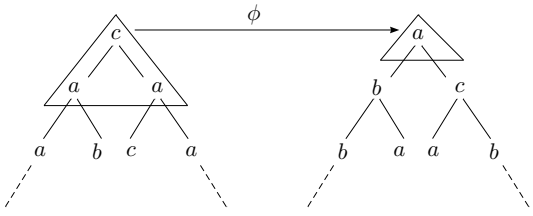


Figure 7: A 2-block map $\Phi : X \to Y$, where X is the tree-shift of Figure 1 and Y a tree-shift of finite type over the alphabet $\{a, b, c\}$.

The 2-block map $\Phi : X_2 \to X_1$ of Figure 7 is a conjugacy. It is defined by $\phi(a, a, b) = b$, $\phi(a, b, a) = c$, $\phi(a, a, c) = b$, $\phi(a, c, a) = c$, $\phi(b, b, b) = a$, $\phi(b, b, c) = a$, $\phi(b, c, b) = a$, $\phi(b, c, c) = a$, $\phi(c, a, a) = a$.

5 Conclusion

We have shown that it decidable whether two tree-shifts of finite type are conjugate. This result makes the class of tree-shifts of finite type close to the class of one-sided shifts of sequences of dimension one. The Decomposition Theorem that we have proved for tree-shifts of finite type may allow us to define a notion of strong tree-shift equivalence between tree-shifts and to deduce that two tree-shifts of finite type are equivalent if and only if their transition matrices are related by a sequence of simple algebraic matrix conditions. In [3], we have started the study of sofic tree-shifts, a class which is larger than tree-shifts of finite type, using tree automata techniques and symbolic dynamic notions similar to the one used for sofic shifts of sequences (see [13], [10], [11]).

References

[1] H. Aso, *Conjugacy of $\mathbf{Z}^2$-subshifts and textile systems*, Publ. Res. Inst. Math. Sci., 36 (2000), pp. 1–18.

[2] 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, Berlin, Heidelberg, 2009, Springer-Verlag, pp. 132–143.

[3] M.-P. Béal and N. Aubrun, *Sofic and Almost of Finite Type Tree-Shifts* , in 5th International Computer Science Symposium in Russia, (CSR'10), E. Mayr and F. Ablayev, eds., no. 6072 in Lecture Notes in Computer Science, Springer-Verlag, 2010, pp. 12–24.

[4] 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.

[5] E. M. Coven, A. Johnson, N. Jonoska, and K. Madden, *The symbolic dynamics of multidimensional tiling systems*, Ergodic Theory Dynam. Systems, 23 (2003), pp. 447–460.

[6] M. Fujiwara, *Conjugacy for one-sided sofic systems*, in Dynamical systems and singular phenomena (Kyoto, 1986), vol. 2 of World Sci. Adv. Ser. Dynam. Systems, World Sci. Publishing, Singapore, 1987, pp. 189–202.

[7] G. Hedlund, *Endomorphisms and automorphisms of the shift dynamical system*, Theory of Computing Systems, 3 (1969), pp. 320–375.

[8] A. S. A. Johnson and K. M. Madden, *The decomposition theorem for two-dimensional shifts of finite type*, Proc. Amer. Math. Soc., 127 (1999), pp. 1533–1543.

[9] B. P. Kitchens, *Symbolic dynamics*, Universitext, Springer-Verlag, Berlin, 1998. One-sided, two-sided and countable state Markov shifts.

[10] W. Krieger, *On sofic systems. I*, Israel J. Math., 48 (1984), pp. 305–330.

[11] ——, *On sofic systems. II*, Israel J. Math., 60 (1987), pp. 167–176.

[12] D. Lind and B. Marcus, *An introduction to symbolic dynamics and coding*, Cambridge University Press, Cambridge, 1995.

[13] M. Nasu, *Topological conjugacy for sofic systems and extensions of automorphisms of finite subsystems of topological markov shifts*, in Proceedings of Maryland special year in Dynamics 1986–87, vol. 1342 of Lecture Notes in Mathematics, Springer verlag, 1988, pp. 564–607.

[14] M. Nasu, *Textile Systems for Endomorphisms and Automorphisms of the Shift*, American Mathematical Society, 1995.

[15] M. Nivat and A. Podelski, eds., *Tree automata and languages*, vol. 10 of Studies in Computer Science and Artificial Intelligence, North-Holland Publishing Co., Amsterdam, 1992. Papers from the workshop held in Le Touquet, June 1990.

[16] D. Perrin and J. Pin, *Infinite words*, Elsevier Boston, 2004.

[17] J. Sakarovitch, *Elements of Automata Theory*, Cambridge University Press, 2009.

[18] H. Seidl, *On the finite degree of ambiguity of finite tree automata*, in Fundamentals of computation theory (Szeged, 1989), vol. 380 of Lecture Notes in Comput. Sci., Springer, New York, 1989, pp. 395–404.

[19] W. Thomas, *Automata on infinite objects*, in Handbook of theoretical computer science, Vol. B, Elsevier, Amsterdam, 1990, pp. 133–191.

[20] R. F. Williams, *Classification of subshifts of finite type*, in Recent advances in topological dynamics (Proc. Conf. Topological Dynamics, Yale Univ., New Haven, Conn., 1972; in honor of Gustav Arnold Hedlund), Springer, Berlin, 1973, pp. 281–285. Lecture Notes in Math., Vol. 318.