Traces of Term-Automatic Graphs

Antoine Meyer

RAIRO-Theor. Inf. Appl. **42** (2008) 615–630
DOI: 10.1051/ita:2008018

Available online at:
www.rairo-ita.org

TRACES OF TERM-AUTOMATIC GRAPHS

Antoine Meyer[1]

Abstract. In formal language theory, many families of languages are defined using either grammars or finite acceptors. For instance, context-sensitive languages are the languages generated by growing grammars, or equivalently those accepted by Turing machines whose work tape's size is proportional to that of their input. A few years ago, a new characterisation of context-sensitive languages as the sets of traces, or path labels, of rational graphs (infinite graphs defined by sets of finite-state transducers) was established. We investigate a similar characterisation in the more general framework of graphs defined by term transducers. In particular, we show that the languages of term-automatic graphs between regular sets of vertices coincide with the languages accepted by alternating linearly bounded Turing machines. As a technical tool, we also introduce an arborescent variant of tiling systems, which provides yet another characterisation of these languages.

Mathematics Subject Classification. 68Q45, 68Q05.

Introduction

In classical language theory, context-sensitive languages, one of the families of the Chomsky hierarchy [2], are defined as the languages generated by growing grammars. They were later characterised as the languages accepted by linearly space-bounded Turing machines [7], *i.e.* Turing machines whose runs on any input word of length n use at most $k \cdot n$ work tape cells, for some constant k. In [8], it was shown that context-sensitive languages also coincide with the languages accepted by bounded tiling systems.

In 2001, [11] provided yet another characterisation of this family as the set of path languages of rational graphs [9], *i.e.* infinite graphs whose vertices are words and whose sets of edges are defined by finite transducers. This result was

Keywords and phrases. Languages, infinite automata, terms, tiling systems, complexity.

[1] LIAFA, Université Paris Diderot – Paris 7 & CNRS, France; ameyer@liafa.jussieu.fr

Article published by EDP Sciences

later extended in [13] to the more restricted family of automatic graphs (*cf.* [6]), and even to synchronous rational graphs when an infinite number of initial and final vertices are considered (see also [10]). In a way, this provides a "forward", automata-based characterisation of context-sensitive languages, as opposed to linearly bounded machines which are essentially a two-way mechanism. To prove the inclusion of context-sensitive languages in the set of path languages of these families of graphs, these papers use a normal form for growing grammars, due to Penttonen [12]. In [4], these results were reformulated using simpler proof techniques based on tiling systems. This also allowed to investigate interesting subcases, in particular concerning deterministic context-sensitive languages or various sub-classes of rational graphs.

The aim of this work is to extend the results of [8] and [4] to the more general family ETIME of languages accepted by deterministic Turing machines working in time less than $2^{O(n)}$, or equivalently by alternating linearly bounded machines. This family lies between context-sensitive and recursively-enumerable languages in the Chomsky hierarchy. We obtain two new characterisations of ETIME, first as the languages accepted by arborescent tiling systems and second as the traces of infinite graphs defined by various classes of term transducers, namely term-synchronous and term-automatic (or tree-automatic) graphs [1].

After recalling definitions and notations in Section 1, we introduce the notion of arborescent tiling systems in Section 2 and prove that they characterise ETIME. Finally, we extend previously mentioned proofs over rational graphs to the family of term-automatic graphs in Section 3.

1. Notations

1.1. Words, terms and trees

A word u over alphabet Σ can be seen as a tuple $(a_1, \ldots, a_n)$ of elements of Σ, usually written $a_1 \ldots a_n$. Its ith letter is denoted by $u(i) = a_i$. The set of all words over Σ is written Σ^*. The number of letters occurring in u is its length, written $|u|$ (here $|u| = n$). The empty word is written ε. The concatenation of two words $u = a_1 \ldots a_n$ and $v = b_1 \ldots b_m$ is the word $uv = a_1 \ldots a_n b_1 \ldots b_m$. The concatenation operation extends to sets of words: for all $A, B \subseteq \Sigma^*$, AB stands for the set $\{uv \mid u \in A \text{ and } v \in B\}$.

1.1.1. *Terms*

Let $F = \bigcup_{n \geq 0} F_n$ be a finite ranked alphabet, each F_n being the set of symbols of F of arity n, and X be a finite set of *variables* disjoint from F (all sets F_n are also disjoint). We denote the arity of a symbol $f \in F$ by $\mathrm{a}(f)$. Variables are considered of arity 0. The set of finite first-order terms on F with variables in X, written $T(F, X)$, is the smallest set including X such that $f \in F_n \wedge t_1, \ldots, t_n \in T(F, X) \Rightarrow f t_1 \ldots t_n \in T(F, X)$. Words can be seen as terms over a ranked

alphabet whose symbols have arity exactly 1 and whose last symbol is a variable or a special constant. To improve readability, $ft_1 \dots t_n$ will sometimes be written $f(t_1, \dots, t_n)$.

1.1.2. *Trees*

A finite ordered tree t over a set of labels Σ is a mapping from a prefix-closed set $\mathrm{dom}(t) \subseteq \mathbb{N}^*$ into Σ. Elements of $\mathrm{dom}(t)$ are called positions, and for every $p \in \mathrm{dom}(t)$, $t(p)$ is the label of the node at position p. The node at position ε is called the root of the tree, nodes at maximal positions (*i.e.* positions x such that $\nexists y \neq \varepsilon,\ xy \in \mathrm{dom}(t)$) are called leaves, other nodes are called internal.

Any term t over a ranked alphabet F and set of variables X can be represented as a finite ordered ranked tree, whose leaves are labelled with constants in F_0 or variables in X and whose internal nodes are labelled with symbols of arity equal to the number of children of that node. In that case, the domain of t, additionally to being prefix-closed, also has the following properties:

(1) $\forall p \in \mathrm{dom}(t),\ t(p) \in F_{n\geq 1} \implies \{j \mid pj \in \mathrm{dom}(t)\} = [1, n]$,
(2) $\forall p \in \mathrm{dom}(t),\ t(p) \in F_0 \cup X \implies \{j \mid pj \in \mathrm{dom}(t)\} = \emptyset$.

In such a tree, position pi with $i \in \mathbb{N}$ always denotes the ith child of node p. Conversely, any finite ordered tree t labelled over Σ can be represented as a ranked tree t', and hence as a term, by mapping each node label a to a set of symbols (a, n) in $\Sigma \times \mathbb{N}$, with $\mathrm{a}(a, n) = n$, and by renumbering all positions such that $\mathrm{dom}(t')$ verifies the above properties. This will usually be left implicit.

1.1.3. *Finite automata and regular sets*

A finite tree (or term) automaton is a tuple $A = \langle Q, F, q_0, \delta \rangle$, where Q is a set of control states, F a ranked alphabet, q_0 the initial set and δ the set of transition rules of A of the form $(q, f, q_1, \dots, q_n)$ with $\mathrm{a}(f) = n$. A run of A over a tree t is a mapping ρ from $\mathrm{dom}(t)$ to Q such that $\rho(\varepsilon) = q_0$ and for all node $u \in \mathrm{dom}(t)$ of arity $\mathrm{a}(u) = n$, $(\rho(u), t(u), \rho(u1), \dots, \rho(un)) \in \delta$. If A has a valid run on t, we say that t is accepted by A. The set of trees accepted by a finite automaton is called its language, and all such languages are said to be *regular*.

1.2. Graphs

A labelled, directed and simple *graph* is a set $G \subseteq V \times \Sigma \times V$ where Σ is a finite set of labels and V an arbitrary countable set. An element (s, a, t) of G is an *edge* of *source* s, *target* t and *label* a, and is written $s \xrightarrow[G]{a} t$ or simply $s \xrightarrow{a} t$ if G is understood. An edge with the same source and target is called a *loop*. The set of all sources and targets of a graph form its *support* V_G, its elements are called *vertices*.

This vision of graphs as sets of edges allows us to define the union, intersection and inclusion over graphs as the corresponding set operations. A graph included

in another graph G is called a *subgraph* of G. The subgraph of G *induced* by a set of vertices $V' \subseteq V$ is simply the set of edges of G whose source and target both belong to V' (*i.e.* $G \cap (V' \times \Sigma \times V')$). One also speaks of the *restriction* of G to V'.

A sequence of edges $(s_1 \xrightarrow{a_1} t_1, \ldots, s_k \xrightarrow{a_k} t_k)$ with $\forall i \in [2,k],\ s_i = t_{i-1}$ is called a *path*. It is written $s_1 \xrightarrow{u} t_k$, where $u = a_1 \ldots a_k$ is the corresponding *path label*. Vertex s_1 is called the origin of the path, t_k its destination. A path is called a *cycle* if its origin and destination are the same vertex. The language, or set of traces of a labelled graph between two sets I and F of vertices is the set of all words w such that there exists a path labelled by w whose origin is in I and destination in F.

1.3. Turing machines

A Turing machine is a tuple $M = \langle \Gamma, \Sigma, Q, q_0, F, \delta \rangle$ where Σ is the input alphabet, Γ the tape or work alphabet (with $\Sigma \subseteq \Gamma$), Q is a set of states among which q_0 is an initial state and F is a set of final states, and δ is a set of transition rules of the form $pA \to qB\epsilon$ where $p, q \in Q$, $A, B \in \Gamma \cup \{\Box\}$ ($\Box$ being a blank symbol not in Γ) and $\epsilon \in \{+, -\}$.

Configurations of M are denoted as words upv, where uv is the content of the work tape (where prefix and suffix blank symbols are omitted), p is the current control state and the head scans the cell containing the first letter of v. A transition $d = pA \to qB\epsilon$ is enabled on any configuration c of the form $upAv$, and yields a new configuration $d(c) = uBqv'$ (with $v' = v$ if $v \neq \varepsilon$, or $\Box$ otherwise) if $\epsilon = +$ and $u'qCBv$ (with $u'C = u$ if $u \neq \varepsilon$ or $u' = \varepsilon$ and $C = \Box$ otherwise) if $\epsilon = -$. If d is not enabled on c, then $d(c)$ is left undefined.

An *alternating* Turing machine M is defined similarly, with the exception that rules are of the form $d = pA \to \bigwedge_{i \in [1,n]} q_i B_i \epsilon_i$. The alternation degree n of d is written $\mathrm{a}(d)$, by analogy with the notion of arity. For all $i \leq \mathrm{a}(d)$, we write d_i the non-alternating transition $pA \to q_i B_i \epsilon_i$. A run of M on input word w is a tree whose root is labelled by configuration $q_0 w$, and such that the children of any node labelled by configuration c are labelled by $c_1, \ldots, c_n$ if and only if there exists a transition $d \in \delta$ enabled on c such that $\mathrm{a}(d) = n$ and $\forall i \in [1,n]$, $c_i = d_i(c)$. Such a run is successful if all its leaves are labelled by configurations whose control state is in F.

A Turing machine is *linearly bounded* if on every run the total work tape space it uses is at most proportional to the length of its input word. By standard coding techniques, it is sufficient to consider machines whose tape is limited to the cells initially containing the input word. This may be enforced by forbidding transition rules to rewrite the blank symbol $\Box$. The languages of non-alternating linearly bounded machines form the complexity class $\mathrm{SPACE}(O(n))$, which is equivalent to context-sensitive languages [7]. Adding alternation, one obtains the more general class $\mathrm{ASPACE}(O(n))$. By classical complexity results [3], it is also equivalent to the class $\mathrm{DTIME}(2^{O(n)})$, also called ETIME.

2. Arborescent tiling systems

To facilitate the proofs of our main results, this section provides an important technical tool, which was also central to some versions of the corresponding proofs on rational graphs and context-sensitive languages (*cf.* [4]).

Tiling systems were originally defined to recognise or specify picture languages, *i.e.* sets of two-dimensional words on finite alphabets [5], called local picture languages. However, by only looking at the words contained in the first row of each picture of a local picture language, one obtains a context-sensitive language, and the converse is true: for any context-sensitive language there exists a local picture language (and a tiling system accepting it) whose set of upper frontiers is that language [8].

In this section, we extend this result to an arborescent extension of tiling systems, and prove that this new formalism characterises precisely the class ETIME.

2.1. Definitions

Instead of planar pictures, we consider so-called arborescent pictures, which are to ordinary pictures what terms are to words.

Definition 2.1 (arborescent picture)**.** Let Γ be a finite alphabet, an arborescent picture p over Γ is a mapping from the set $X \times [1, m]$ to Γ, where $X \subseteq {\mathbb{N}_+}^*$ is a finite, prefix-closed set of sequences of positive integers (called positions in the framework of trees) and m is a positive integer called the width of p. The set $\mathrm{dom}(p) = X \times [1, m]$ is the domain of p. The set of arborescent pictures over $X \times [1, m]$ is written $\mathrm{AP}(X, m)$.

Like in the case of trees, we assume that X is not only prefix-closed but also left-closed, *i.e.* $\forall i > 0,\ ui \in X \implies \forall j < i,\ uj \in X$. For a given picture $p \in \mathrm{AP}(X, m)$, we write $\mathrm{fr}(p)$ the word $w \in \Gamma^m$ such that $w(i) = p(\varepsilon, i)$, which we call the (upper) frontier of p.

Arborescent pictures of domain $X \times [1, m]$ are isomorphic to ordered trees of domain X with nodes labelled over the set Γ^m. As such, if $m = 1$ they are isomorphic to Γ-labelled ordered trees. One can observe that any branch of an arborescent picture seen as a Γ^m-labelled tree, as well as any arborescent picture whose set of positions X is a subset of 1^*, is an ordinary, planar picture.

Definition 2.2 (sub-picture)**.** For any arborescent picture $p \in \mathrm{AP}(X, m)$, the sub-picture $p' = p|_{x,i,Y,n}$ of p at offset $o = (x, i)$ with $x \in X$ and $i \in [0, m-1]$ is the arborescent picture of domain $Y \times [1, n]$ such that Y is prefix- and left-closed and $\forall (y, j) \in Y \times [1, n],\ (xy, i + j) \in X \times [1, m]$ and $p'(y, j) = p(xy, i + j)$.

We can now define arborescent tiling systems, which allow the specification of sets of arborescent pictures. Similarly to planar tiling systems, in order to be able to recognise meaningful sets of pictures, we first add a border or frame to each picture using a new symbol #.

Definition 2.3 (framed picture)**.** Let p be an arborescent picture of domain $X \times [1,m]$ over Γ and $\# \notin \Gamma$ a new symbol, we define the #-framed picture $p_\#$ as the picture of domain $X' \times [1, m+2]$ with $X' = \{\varepsilon\} \cup \{1\}X \cup X''$ and $X'' = \{1x1 \mid x \in X \wedge \nexists y \in \mathbb{N}, xy \in X\}$ such that

$$\begin{array}{ll} p_\#(\varepsilon, i) = \# & \text{for all } i \in [1, m+2], \\ p_\#(1x, 1) = \# \text{ and } p_\#(1x, m+2) = \# & \text{for all } x \in X, \\ p_\#(x, i) = \# & \text{for all } x \in X'', i \in [1, m+2], \\ p_\#(1x, i+1) = p(x, i) & \text{for all } x \in X, i \in [1, m]. \end{array}$$

An arborescent tiling system is then defined as a set of tiling elements of width and height 2, which can then be combined to form larger framed pictures.

Definition 2.4 (arborescent tiling system)**.** An arborescent tiling system (or ATS) S is a triple $(\Gamma, \#, \Delta)$, where Γ is a finite alphabet, $\# \notin \Gamma$ a frame symbol and Δ is a set of arborescent tiling elements (tiles) in $\{\bar{\Gamma} \times \bar{\Gamma} \times \bar{\Gamma}^n \times \bar{\Gamma}^n \mid n > 0\}$ with $\bar{\Gamma} = \Gamma \cup \{\#\}$.

Each tiling element $d \in \Delta$ is of the form $d = (A, B, \bar{C}, \bar{D})$ with $A, B \in \bar{\Gamma}$ and $\bar{C}, \bar{D} \in \bar{\Gamma}^n$ for some positive integer n. We define additional notations to conveniently manipulate tiling elements. Let $d = (A, B, \bar{C}, \bar{D})$ with $\bar{C} = C_1 \dots C_n$ and $\bar{D} = D_1 \dots D_n$, we write $\mathrm{a}(d) = n$ to denote the arity of d, and d_i with $i \in [1, \mathrm{a}(d)]$ to denote the (planar) tile (A, B, C_i, D_i).

Note that any tiling element $d = (A, B, \bar{C}, \bar{D})$ of arity n is isomorphic to an arborescent picture p_d of domain $X \times [1,2]$, where $X = \{\varepsilon, 1, \dots, n\}$ and $p_d(\varepsilon, 1)$, $p_d(\varepsilon, 2)$, $p_d(i, 1)$ and $p_d(i, 2)$ are respectively equal to A, B, C_i and D_i (for all $i \in [1, n]$). In general we do not distinguish p_d from d and write simply d.

Well-formed tiling systems should obey a certain number of restrictions over their set of tiles, regarding in particular the occurrences of the frame symbol # inside tiles. For all $d = (A, B, \bar{C}, \bar{D})$,

(1) $(A, B) = (\#, \#) \implies \mathrm{a}(d) = 1 \wedge (C_1, D_1) \neq (\#, \#)$,
(2) $\exists i, (C_i, D_i) = (\#, \#) \implies \mathrm{a}(d) = 1 \wedge (A, B) \neq (\#, \#)$,
(3) $\exists i, C_i = \# \wedge D_i \neq \# \implies A = \# \wedge \forall i, C_i = \#$,
(4) $\exists i, D_i = \# \wedge C_i \neq \# \implies B = \# \wedge \forall i, D_i = \#$.

Before defining the set of pictures and the word language accepted by an arborescent tiling system, we define for any arborescent picture p of domain $X \times [1,m]$ over Γ the set $\mathrm{T}(p)$ of tiling elements of p as the set of all sub-pictures $p|_{x,j,X',2}$ of p such that x is an internal position in X, $j \in [1, m-1]$ and $X' = \{\varepsilon\} \cup \{i' > 0 \mid xi' \in X\}$.

Definition 2.5 (language of a tiling system)**.** The set of arborescent pictures accepted by an arborescent tiling system $S = (\Gamma, \#, \Delta)$ is the set $\mathrm{P}(S) = \{p \in \mathrm{AP} \mid \mathrm{T}(p_\#) \subseteq \Delta\}$. The (word) language accepted by S is the set $\mathrm{L}(S) = \{w \in \Gamma^* \mid \exists p \in \mathrm{P}(S), w = \mathrm{fr}(p)\}$ of all upper frontiers of pictures of $\mathrm{P}(S)$.

As previously, note that arborescent tiling systems are a syntactical generalisation of planar tiling systems: framed pictures with a domain $X \subseteq 1^*$ or branches

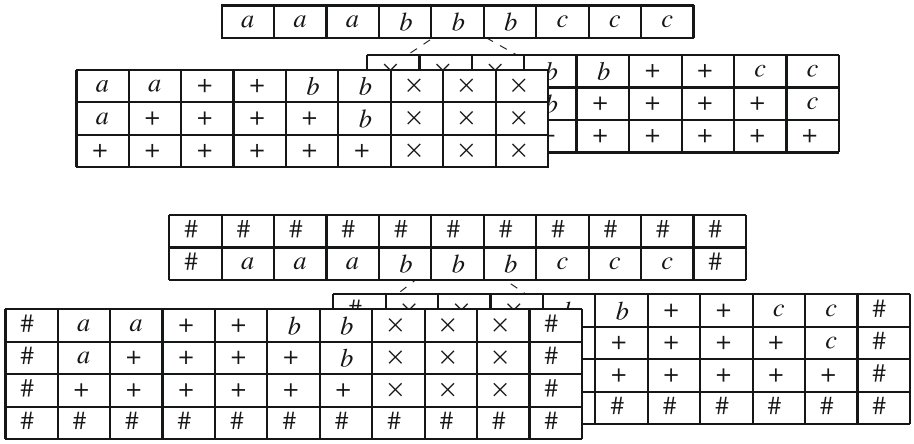

FIGURE 1. Arborescent picture p and the corresponding framed picture $p_{\#}$.

of framed arborescent pictures are planar framed pictures, and arborescent tiling systems whose elements all have arity 1 are ordinary, planar tiling systems.

Example 2.6. Figure 1 represents an arborescent picture p whose frontier is the word $a^3b^3c^3$, as well as the corresponding framed picture. By considering all subpictures of height and width 2 of that framed picture, one obtains a set of tiling elements Δ, which contains, among others, tiling elements $(\#, \#, a, b)$, $(a, +, +, +)$ and $(+, +, \#, \#)$ of arity 1 and $(b, c, b+, \times+)$ of arity 2, but not $(b, c, b+, c+)$ or $(\#, +, \#, +)$ for instance.

One can see that the tiling system $S = (\{a, b, c, +, \times\}, \#, \Delta)$ accepts all arborescent pictures similar to p whose frontiers are words of the form $a^nb^nc^n$ with $n \geq 2$: the left branch of each such picture ensures that the number of a's and b's is equal by replacing at each successive row one occurrence of a and one occurrence of b by some symbol $+$. Occurrences of c are irrelevant and are replaced with symbol $\times$. A lower frame borders can only be generated once all occurrences of a and b have been replaced. A similar check is performed by the right branch for symbols b and c.

Note that S does not accept the word abc, since accepting a similar picture with frontier abc would require some additional tiling elements, like for instance $(a, b, +\times, ++)$ and $(b, c, ++, \times+)$. Consequently, the language $\mathrm{L}(S)$ is $\{a^nb^nc^n \mid n \geq 2\}$.

2.2. Languages of arborescent tiling systems

In this section, we prove that arborescent tiling systems and alternating linearly bounded machines define the same family of languages, namely $\mathrm{ASPACE}(O(n))$, also equal as previously mentioned to $\mathrm{DTIME}(2^{O(n)}) = \mathrm{ETIME}$.

Proposition 2.7. *For every arborescent tiling system S, there exists an alternating linearly bounded machine M such that* $\mathrm{L}(M) = \mathrm{L}(S)$.

Proof. Let $S = (\Gamma, \#, \Delta)$ be an arborescent tiling system. We build an alternating linearly bounded machine $M = (\Gamma, \Gamma', Q, q_\#, f, \delta)$ accepting $\mathrm{L}(S)$. Its work alphabet Γ' is the union of all $\bar{\Gamma}^k$ for $k \in [1, \mathrm{a}(S)]$, where $\bar{\Gamma} = \Gamma \cup \{\#\}$ and $\mathrm{a}(S) = \max\{\mathrm{a}(d) \mid d \in \Delta\}$. The control states and transition rules of M are (only) those appearing in the following description of M's behaviour.

(1) M starts in configuration $[q_\# w]$, where $w \in \Gamma^*$ is the input word. For all $(\#, \#, C, D) \in \Delta$ with $D \neq \#$, δ contains a rule $q_C D \to q_D D+$. For every tile $(\#, \#, C, \#) \in \Delta$ and positive integer $n \leq a(S)$, M can reverse its head with a rule $q_C] \to p_1 {}^{\#}_{\#^n}]-$. This first sweep checks that w is a possible upper frontier of a picture accepted by S.

(2) In the next sweep, M generates at once a possible n-tuple of next rows based on the current configuration and the tiles in Δ. For all $\bar{A} = A_1 \dots A_m$, $\bar{C} = C_1 \dots C_n$ and $\bar{D} = D_1 \dots D_n$ with $m, n \in [1, \mathrm{a}(S)]$ and for all $k \in [1, m]$, δ contains the rules

$$
\begin{aligned}
p_k {}^{\#}_{\#^n} \bar{A} &\to p_k {}^{A_k}_{\bar{C}} \bar{C}- && \text{for all } (A_k, \#, \bar{C}, \#^n) \in \Delta,\\
p_k {}^{B_k}_{\bar{D}} \bar{A} &\to p_k {}^{A_k}_{\bar{C}} \bar{C}- && \text{for all } (A_k, B_k, \bar{C}, \bar{D}) \in \Delta,\\
p_k {}^{B_k}_{\bar{D}} [&\to \bigwedge_{i \in [1,n]} q_i {}^{\#}_{\#^l} [+ && \text{for all } (\#, B_k, \#^n, \bar{D}) \in \Delta,\ l \in [1, \mathrm{a}(S)].
\end{aligned}
$$

Rules of the latter type perform a head reversal with universal branching at the end of a sweep. On each new computation branch, proceed to the next step.

(3) The last row generated on component k consists of a sequence of frame symbols # if and only if the last symbol written to the right of the left border symbol is #.

If this is the case on the current computation branch, reach accepting state f with rules of the form $q_k {}^{\#}_{\#^n} \bar{B} \to f \bar{B}+$ for all $m \in [1, \mathrm{a}(S)]$, $k \in [1, m]$ and $\bar{B} = B_1 \dots B_m$ with $B_k = \#$. Otherwise, proceed to the next step.

(4) This step is the right-to-left counterpart of step 2. For all $\bar{B} = B_1 \dots B_m$, $\bar{C} = C_1 \dots C_n$ and $\bar{D} = D_1 \dots D_n$ with $m, n \in [1, \mathrm{a}(S)]$ and for all $k \in [1, m]$, δ contains the rules

$$
\begin{aligned}
q_k {}^{\#}_{\#^n} \bar{B} &\to q_k {}^{B_k}_{\bar{D}} \bar{D}+ && \text{for all } (\#, B_k, \#^n, \bar{D}) \in \Delta,\\
q_k {}^{A_k}_{\bar{C}} \bar{B} &\to q_k {}^{B_k}_{\bar{D}} \bar{D}+ && \text{for all } (A_k, B_k, \bar{C}, \bar{D}) \in \Delta,\\
q_k {}^{A_k}_{\bar{C}}] &\to \bigwedge_{i \in [1,n]} p_i {}^{\#}_{\#^l}]- && \text{for all } (A_k, \#, \bar{C}, \#^n) \in \Delta, l \in [1, \mathrm{a}(S)].
\end{aligned}
$$

As previously, rules of the latter type perform a head reversal with universal branching at the end of a sweep. On each new computation branch, proceed to the next step.

(5) Conversely to step 3, if the last symbol written on component k to the left of the right border symbol is $\#$, then reach accepting state f with rules of the form $p_k \frac{\#}{\#^n} \bar{A} \to f\bar{A}-$ for all $m \in [1, \mathrm{a}(S)]$, $k \in [1, m]$ and $\bar{A} = A_1 \dots A_m$ with $A_k = \#$. Otherwise, proceed to step 2.

Steps 2 to 5 are repeated until all computation branches have reached the accepting state f. It then only remains to check that this happens if, and only if, the input word w is accepted by S.

A control state of M of the form $p_k \frac{B_k}{\bar{D}}$, occurring in a right-to-left sweep of the tape (step 2), carries the information that at the previous step, the tape cell to the right of the head contained some word $\bar{B}$ of k-th letter B_k, and that this cell now contains $\bar{D}$. Transitions from this control state to a control state $p_k \frac{A_k}{\bar{C}}$ are now enabled if, and only if, the current cell contains a word $\bar{A}$ whose k-th letter is A_k, and there exists in Δ a tiling element $(A_k, B_k, \bar{C}, \bar{D})$, which M can nondeterministically try to simulate. If such a transition is used, $\bar{A}$ is then replaced with $\bar{C}$ and the simulation continues.

Once a complete sweep is achieved, it is ensured that the new tape content is consistent with the tiling elements in Δ as far as component k of the previous tape content is concerned. Once the left tape delimiter is reached, either the whole tape contains border symbols, in which case the current execution branch should reach an accepting state (step 3), or M has to launch as many alternating computations as necessary to proceed with the simulation. Similar explanations then hold for states of type $q_k \frac{A_k}{\bar{C}}$ during step 4.

Consider a run ρ of M over word w with $|w| = n$. Let w_x be the tape content reached after $|x|$ complete sweeps of the tape where x is a sequence of indices denoting the computation branch where this tape content appears (*i.e.* x is the sequence of indices k of control states of M encountered after each head reverse on this particular execution branch), and let X be the set of all such x. Define $w_x(k)$ as the word $\bar{A}^{(x,1)}(k) \dots \bar{A}^{(x,n)}(k)$, where $w_x = \bar{A}^{(x,1)} \dots \bar{A}^{(x,n)}$. Let p be the arborescent picture of domain $Y = \{y \mid 1y \in X\}$ whose row at position ε is w and whose row at position yk is precisely $w_{1y}(k)$. It is tedious but straightforward to prove that p belongs to $\mathrm{P}(S)$ (and hence w to $\mathrm{L}(S)$) if and only if ρ is an accepting run (and hence $w \in \mathrm{L}(M)$). □

Proposition 2.8. *For every alternating linearly bounded machine M, there exists an arborescent tiling system S such that* $\mathrm{L}(S) = \mathrm{L}(M)$.

Proof. Let $M = (\Sigma, \Gamma, Q, q_0, F, \delta)$ be an alternating linearly bounded machine. We build an arborescent tiling system $S = (\Gamma', \#, \Delta)$ such that $\mathrm{L}(S) = [\mathrm{L}(M)]$, where [and] are two new symbols. We suppose without loss of generality that every rule d of M is given under the form $d = pA \to q_1 B_1 \mu_1 \wedge \dots \wedge q_n B_n \mu_n$. By d_i we refer to the fragment $pA \to q_i B_i \mu_i$ of d, and we let $\mathrm{a}(d) = n$.

The construction goes as follows. S first needs to set an input word w as the upper frontier of any picture it accepts. It then encodes the initial configuration of M on w as the second row. Subsequent tiles simulate the application of a transition of M on the configuration encoded by the current row, and check that the previous transition was correctly simulated. Arity n tiling elements will be used to simulate a rule of alternation degree n. This process goes on until an accepting state is reached by M on a given execution branch. In that case, a bottom border is generated by S on the corresponding picture branch.

This process requires additional information, in particular about the position of the head and the index of the last simulated transition, to be added to the picture alphabet. To this end, bracketed superscripts of the form (d) with $d \in \delta$ will be used to indicate that the transition last performed by M before reaching the configuration represented by the current row is d. Subscripts ℓ and r are used to indicate whether a given cell lies to the left or to the right of the read head. Subscripts of the form p with $p \in Q$ denote that the read head currently stands on the position represented by the current cell and is in state p.

Tiling elements of S consist in the following sets. First, we need a set of tiles of arity 1 to set the input word as upper frontier. For all $a, b \in \Sigma$, Δ contains

$$(\#, \#, \#, [\,),\ (\#, \#, [\,, a),\ (\#, \#, a, b),\ (\#, \#, b, \,]),\ (\#, \#, \,], \#).$$

In the second row we then need to encode the initial configuration $[q_0 w]$ of M. Thus for all $a, b \in \Sigma$, Δ contains the arity 1 tiles

$$\begin{gathered}(\#, [\,, \#, [_{\ell}^{(\perp)}),\ ([\,, a, [_{\ell}^{(\perp)}, a_{q_0}^{(\perp)}),\ (a, b, a_{q_0}^{(\perp)}, b_r^{(\perp)}),\\ (a, b, a_r^{(\perp)}, b_r^{(\perp)}),\ (b, \,], b_r^{(\perp)}, \,]^{(\perp)}),\ (\,[\,, \#, \,]_r^{(\perp)}, \#),\end{gathered}$$

where $\perp$ denotes a dummy transition of arity considered as 1. Subsequent tiles check the consistency of the previously applied transition d throughout a row, and simulate the application of a new transition d' of M. Arity n tiles are used when d' is of alternation degree n. For all $A, B, B', C, C' \in \Gamma$ and $d \in \delta$, Δ thus contains

$$\begin{array}{ll}
(\#, [_{p}^{(d)}, \#^n, ([_{\ell}^{(d')})^n) & \text{for all } d' = (p[\to q_1[+ \wedge \ldots \wedge q_n[+) \in \delta,\\
(\#, [_{\ell}^{(d)}, \#^n, Y) & \text{for all } d' \in \delta \text{ with } \mathrm{a}(d') = n, \text{ where } \forall i \in [1, n],\\
& Y_i = \begin{cases} [_{q_i}^{(d')} \text{ or } [_{\ell}^{(d')} \text{ if } d'_i = pC \to q_i C' - \\ [_{\ell}^{(d')} \text{ otherwise,} \end{cases}\\
(A_{\ell}^{(d)}, B_{\ell}^{(d)}, (A_{\ell}^{(d')})^n, Y) & \text{for all } d' \in \delta \text{ with } \mathrm{a}(d') = n, \text{ where } \forall i \in [1, n],\\
& Y_i = \begin{cases} B_{q_i}^{(d')} \text{ or } B_{\ell}^{(d')} \text{ if } d'_i = pC \to q_i C' - \\ B_{\ell}^{(d')} \text{ otherwise,} \end{cases}
\end{array}$$

$$(A_\ell^{(d)}, B_p^{(d)}, X, Y) \qquad \text{for all } d' \in \delta \text{ with } \mathrm{a}(d') = n, \text{where } \forall i \in [1,n],$$
$$\begin{cases} X_i = A_{q_i}^{(d')} \text{ and } Y_i = {B'_r}^{(d')} & \text{if } d'_i = pB \to q_i B'- \\ X_i = A_\ell^{(d')} \text{ and } Y_i = {B'_\ell}^{(d')} & \text{if } d'_i = pB \to q_i B'+ \end{cases}$$

as well as all tiling elements of the form $(\rfloor_p^{(d)}, \#, (\rfloor_p^{(d')})^n, \#^n)$, $([\,{}_\ell^{(d)}, \#, X, \#^n)$, $(A_r^{(d)}, B_r^{(d)}, X, (B_r^{(d')})^n)$ and $(A_p^{(d)}, B_r^{(d)}, X, Y)$, defined dually.

Furthermore, if the last simulated transition ends in a final control state, tiles of Δ should allow one to generate a lower border: we thus have copies of all the previous rules with $X = Y = \#$, with the additional constraint that transition d reaches an accepting state of M.

It remains to prove that one indeed has $\mathrm{L}(S) = [\mathrm{L}(M)]$, which can be done by proving that in any picture p accepted by S, if a row r encodes some configuration c of M, then the children of r accurately encode the direct successors of c through an alternating transition $d \in \delta$. Consequently, each picture of upper frontier $[w]$ accepted by S is a faithful encoding of an accepting run of M on w. Conversely, whenever M has an accepting run over some word w, its encoding as a picture has to be accepted by S.

Defining a tiling system which accepts precisely $L(M)$ and not $[L(M)]$, *i.e.* removing the border symbols from $\mathrm{L}(S)$, is then a simple exercise. □

From Propositions 2.7 and 2.8, we deduce the announced theorem.

Theorem 2.9. *The languages of arborescent tiling systems form the complexity class* ETIME.

Note that the language accepted by the tiling system of Example 2.6 is a context-sensitive language, which could also be accepted by a non-arborescent tiling system.

3. Traces of term-automatic graphs

We now turn to the main result of this paper, which is the study of languages of graphs characterised by automata-defined binary relations over terms, and in particular term-automatic graphs. We define these relations and the graphs they generate, then present a two-steps proof that the languages of term-automatic graphs indeed coincide with ASPACE($O(n)$). First, we establish this result for the simpler term-synchronous graphs in Section 3.2, then generalise it to term-automatic graphs in Section 3.3.

3.1. Definitions

Let $s = f(s_1 \dots s_m)$ and $t = g(t_1 \dots t_n)$ be two terms over some ranked alphabet F. We define the overlap $[st]$ of s and t as a term over domain $\mathrm{dom}(s) \cup \mathrm{dom}(t)$ and extended alphabet $(F \cup \{\bot\})^2$ (each element (f,g) of this alphabet being

written simply fg), such that $\forall p \in \mathrm{dom}(s) \cup \mathrm{dom}(t)$, $[st](p) = fg$ with $f = s(p)$ if $p \in \mathrm{dom}(s)$ or $\perp$ otherwise, and $g = t(p)$ if $p \in \mathrm{dom}(t)$ or $\perp$ otherwise. This notation is extended to sets in the natural way.

We can now define term-automatic and term-synchronous relations. We say a binary relation R is *term-* (or *tree-*)*automatic* if the term language $[R] = \{[st] \mid (s,t) \in R\}$ is regular. If furthermore for all $(s,t) \in R$, $\mathrm{dom}(s) = \mathrm{dom}(t)$, it is called *synchronous*. In other words, a synchronous relation is an automatic relation which only associates terms with the same domain. Both families of relations are closed under relational composition. Term-automatic and term-synchronous relations are syntactical extensions of the corresponding families of relations over words. As such, they also define extended families of graphs.

Definition 3.1. A Σ-graph G is term-automatic (resp. term-synchronous) if it is isomorphic to a graph $\{u \xrightarrow{a} v \mid a \in \Sigma,\ (u,v) \in R_a\}$, where $(R_a)_{a\in\Sigma}$ is a family of term-automatic (resp. term-synchronous) relations.

3.2. Term-synchronous graphs

This section presents direct simulations of alternating tiling systems by synchronous graphs and conversely, showing that the languages of term-synchronous graphs between regular sets of vertices form the class ETIME.

Proposition 3.2. *For every term-synchronous graph G and regular sets I and F there exists an arborescent tiling system S such that* $\mathrm{L}(S) = \mathrm{L}(G, I, F)$.

Proof. Let $G = (R_a)_{a\in\Sigma}$ be a term-synchronous graph, and I, F two regular sets of vertices of G. We build an arborescent tiling system $S = (\Gamma, \#, \Delta)$ such that $\mathrm{L}(S) = \mathrm{L}(G, I, S)$.

For all $a \in \Sigma$, let A_a be a finite top-down term automaton accepting the language $[R_a]$ (as defined in Section 3.1), and A_I, A_F similar automata for I and F respectively. For every $a \in \Sigma$, we also define relations $R_{I\circ a} = Id_I \circ R_a$ and $R_{a\circ F} = R_a \circ Id_F$, where Id_L denotes the identity relation over some set L. Let also $A_{I\circ a}$ and $A_{a\circ F}$ be two automata accepting the languages $[R_{I\circ a}]$ and $[R_{a\circ F}]$ respectively. The control state sets of all these automata are supposed disjoint.

For every path $t_0 \xrightarrow{a_1} t_1 \dots \xrightarrow{a_n} t_n$ in G with $t_0 \in I$, $t_n \in F$ and $\forall i,\ \mathrm{dom}(t_i) = X$, we want S to accept an arborescent picture p such that

- $p|_{\varepsilon,1,X,1}$ is isomorphic to the term $a_1, (\rho_1)$, where ρ_1 is an accepting run of $A_{I\circ a_1}$ over $[t_0 t_1]$;
- $p|_{\varepsilon,i,X,1}$ is isomorphic to the term $a_i(\rho_i)$ for all $i \in [2, n-1]$, where ρ_i is an accepting run of A_{a_i} over $[t_{i-1} t_i]$;
- $p|_{\varepsilon,n,X,1}$ is isomorphic to the term $a_n(\rho_n)$, where ρ_{n+1} is an accepting run of $A_{a_n\circ F}$ over $[t_{n-1} t_n]$,

and conversely, S should only accept all such pictures which correspond to paths in G between I and F.

These conditions are sufficient for S to accept the word language $\mathrm{L}(G, I, F)$. To ensure they indeed hold, we define Δ as containing the following tiles. For

the leftmost columns of pictures, we simulate for every a the possible runs of automaton $A_{I\circ a}$ with tiles

$$\begin{array}{ll}
(\#,\#,\#,a), & \\
(\#,a,\#,px) & \text{if } p \text{ initial in } A_{I\circ a},\\
(\#,px,\#^k,p_1x_1\dots p_kx_k) & \text{if } px\to p_1x_1\dots p_kx_k\in A_{I\circ a},\\
(\#,px,\#,\#) & \text{if } px\to\varepsilon\in A_{I\circ a}.
\end{array}$$

For pairs of intermediate columns inside a picture, we simulate two automata side by side while checking for consistency. Hence for all $a,b\in\Sigma$, we have tiles

$$\begin{array}{ll}
(\#,\#,a,b), & \\
(a,b,px,qy) & \text{with } p \text{ initial in } A_a,\ q \text{ initial in } A_b\\
 & \text{and } x=fg \text{ and } y=gh \text{ for some } f,g,h,\\
(px,qy,p_1x_1\dots p_kx_k,q_1y_1\dots q_ky_k) & \text{if } \begin{cases} px\to p_1x_1\dots p_kx_k\in A_a\\ qy\to q_1y_1\dots q_ky_k\in A_b\end{cases}\\
 & \text{and } \forall i\in[0,k],\ x_i=fg \text{ and } y_i=gh \text{ for some } f,g,h\\
(px,qy,\#,\#) & \text{if } px\to\varepsilon\in A_a,\ qy\to\varepsilon\in A_b\\
 & \text{and } x=fg \text{ and } y=gh \text{ for some } f,g,h.
\end{array}$$

Finally, for the rightmost columns we have the following set of tiles, which is analogous to the leftmost case. For every letter b, Δ contains

$$\begin{array}{ll}
(\#,\#,b,\#), & \\
(b,\#,qy,\#) & \text{if } q \text{ initial in } A_{b\circ F},\\
(qy,\#,q_1y_1\dots q_ky_k,\#^k) & \text{if } qy\to q_1y_1\dots q_ky_k\in A_{b\circ F},\\
(qy,\#,\#,\#) & \text{if } qy\to\varepsilon\in A_{b\circ F}.
\end{array}$$

It then only remains to check that given this set of tiles, S indeed enjoys the properties cited above, and hence accepts $\mathrm{L}(G,I,F)$. □

Proposition 3.3. *For every arborescent tiling system S, there exists a term-synchronous graph G and regular sets I and F such that $\mathrm{L}(G,I,F)=\mathrm{L}(S)$.*

Proof. Let $S=(\Gamma,\#,\Delta)$ be an arborescent tiling system. We build a term-synchronous graph G such that $\mathrm{L}(S)=\mathrm{L}(G,I,F)$ for some regular sets I and F. In the following, symbol $\#$ is overloaded to make the notation less cumbersome, and represents functional symbols of varying arities, which can be deduced from the context. In particular, we write $\#_X$ for a given prefix-closed set X the term of domain X whose nodes are all labelled with $\#$.

Let R_a, $a\in\Sigma$, be the binary relation between all terms $\#(s)$ and $\#(t)$ (*i.e.* s and t with an additional unary $\#$ at the root) such that a labels the root of t

and for a given $p \in P(S)$, either $s = p|_{\varepsilon,i,X,1}$ and $t = p|_{\varepsilon,i+1,X,1}$ for some $i > 0$ or $s = \#_X$ and $t = p|_{\varepsilon,0,X,1}$.

Let G be the graph defined by $(R_a)_{a\in\Sigma}$, we show that G is term-synchronous by constructing automata $(A_a)_{a\in\Sigma}$ such that $\mathrm{L}(A_a) = [R_a] = \{[st] \mid (s,t) \in R_a\}$. For all a, A_a has transitions:

$$\begin{aligned}
&q_0\#\# \to q_{AB,1} && \text{if } (\#,\#,A,B) \in \Delta,\\
&q_{\bar{A}\bar{B},i}AB \to q_{\bar{C}\bar{D},1}\dots q_{\bar{C}\bar{D},k} && \text{if } d = (A_i, B_i, \bar{C}, \bar{D}) \in \Delta,\ k = \mathrm{a}(d),\\
& && \quad A_i = \bar{A}(i) \text{ and } B_i = \bar{B}(i),\\
&q_{\bar{A}\bar{B},i}A_iB_i \to \varepsilon && \text{if } (A_i, B_i, \#, \#) \in \Delta,\\
& && \quad A_i = \bar{A}(i) \text{ and } B_i = \bar{B}(i).
\end{aligned}$$

We define I as the regular set of all terms labelled over $\{\#\}$, and F as the set of all possible rightmost columns of pictures accepted by S. This set of terms is accepted by the automaton A_F whose transitions are:

$$\begin{aligned}
&q_0\# \to q_{A,1} && \text{if } (\#,\#,A,\#) \in \Delta,\\
&q_{\bar{A},i}A_i \to q_{\bar{C},1}\dots q_{\bar{C},k} && \text{if } d = (A_i, \#, \bar{C}, \#^k) \in \Delta \text{ and } A_i = \bar{A}(i),\\
&q_{\bar{A},i}A_i \to \varepsilon && \text{if } (A_i, \#, \#, \#) \in \Delta \text{ and } A_i = \bar{A}(i).
\end{aligned}$$

By construction of each of the A_a, I and A_F, there exists a path in G labelled by a word w between a vertex in I and a vertex in F if and only if the vertices along that paths are the successive columns of a picture in $P(S)$ whose frontier is w. □

3.3. Term-automatic graphs

In this section, we show that the more general family of term-automatic graphs defines the same family of languages as their synchronous counterparts.

Proposition 3.4. *For every term-automatic graph G and regular sets of terms I and F, there exists a term-synchronous graph G' and regular sets I' and F' such that* $\mathrm{L}(G', I', F') = \mathrm{L}(G, I, F)$.

Proof. Let G be a term-automatic graph defined by a family $(R_a)_{a\in\Sigma}$ of automatic relations and I, F be two regular languages, each $[R_a]$ being accepted by an automaton A_a, I by A_I and F by A_F. We define a synchronous graph $G' = (R'_a)_{a\in\Sigma}$ and two regular sets I' and F' such that $\mathrm{L}(G, I, F) = \mathrm{L}(G', I', F')$.

Recall that term-automatic relations are defined using a notion of overlap between terms (*cf.* Sect. 3.1). Two terms s and t with different domains belong to a term-automatic relation R defined by automaton A if the overlap $[st]$ of s and t is accepted by A. This notion of overlap consists in "unifying" the domains of s and t, and padding undefined positions with a special symbol $\perp$.

We wish to reuse this idea, but instead of unifying the domains of two terms only, we have to unify the domains of all vertices along a given path. Indeed, in

a term-synchronous graph, edges can only exist between terms with precisely the same domain. For every term s standing for a vertex in G, we will thus have to consider an infinite set of "versions" of s in G', one for each possible term domain larger than that of s.

Let Γ be a ranked alphabet, we define alphabet Γ' as $\Gamma' = \Gamma_0 \cup \Gamma_n$ with $\Gamma'_0 = \#_0$ and $\Gamma'_n = \Gamma \cup \#_n$, where n is the maximal arity of symbols in Γ. Let ϕ be a mapping from $T(\Gamma)$ to $2^{T(\Gamma')}$ such that for any term $t \in T(\Gamma)$,

$$\phi(t) = \big\{t' \in T(\Gamma') \mid \mathrm{dom}(t) \subset \mathrm{dom}(t'),\ \forall p \in \mathrm{dom}(t), t'(p) = t(p) \text{ and } \forall p \in \mathrm{dom}(t') \setminus \mathrm{dom}(t), t'(p) \in \{\#_0, \#_n\}\big\}.$$

In other words, to any term t, ϕ associates the set of all terms obtained by "padding" t with silent symbols $\#_0$ and $\#_n$. This mapping is extended to sets of terms in the natural way. Note that, given any $t' \in F(\Gamma')$, there exists at most one term $t \in T(\Gamma)$ such that $t' \in \phi(t)$.

We now define, for every $a \in \Sigma$, relation R'_a as $\{(s', t') \mid (s, t) \in R_a,\ s' \in \phi(s),\ t' \in \phi(t)$ and $\mathrm{dom}(s') = \mathrm{dom}(t')\}$. This synchronous relation can be characterised by a finite tree automaton A'_a defined from A_a. For every transition $px \to q_1 \dots q_k$ in A_a, with $0 \leq k \leq n$, A'_a has a transition $p'x \to q'_1 \dots q'_k (q_\#)^{n-k}$, as well as transitions $q_\# \#_n \to (q_\#)^n$ and $q_\# \#_0 \to \varepsilon$. The initial state of A'_a is q'_0. We also let $I' = \phi(I)$ and $F' = \phi(F)$, for which automata can be similarly defined from A_I and A_F.

Let G' be the term-synchronous graph defined by $(R'_a)_{a \in \Sigma}$. For every path $i' = t'_0 \xrightarrow{a_1} t'_1 \dots t'_{n-1} \xrightarrow{a_n} t'_n = f'$ in G' with $i' \in I'$ and $f' \in F'$, by definition of G' and ϕ, and for all $i \in [1, n]$, there must exist t_{i-1} and t_i such that $t_{i-1} \xrightarrow{a_i} t_i \in G$, $t'_{i-1} \in \phi(t_{i-1})$ and $t'_i \in \phi(t_i)$. Also, by definition of I and F, $t_0 \in I$ and $t_n \in F$. Hence $a_1 \dots a_n \in \mathrm{L}(G, I, F)$, and more generally $\mathrm{L}(G', I', F') \subseteq \mathrm{L}(G, I, F)$.

Conversely, consider any path $i = t_0 \xrightarrow{a_1} t_1 \dots t_{n-1} \xrightarrow{a_n} t_n = f$ in G with $i \in I$ and $f \in F$. One can easily see that for some sufficiently large domain X, for all $i \in [0, n]$ there exists $t'_i \in \phi(t_i)$ with $\mathrm{dom}(t'_i) = X$. From there, it is not difficult to conclude that there is a path in G' labelled by $a_1 \dots a_n$, hence that $\mathrm{L}(G, I, F) \subseteq \mathrm{L}(G', I', F')$. □

Remark 3.5. Note that for every term-automatic graph G and regular sets I and F, there exists a term-automatic graph G' and finite sets I' and F' such that $\mathrm{L}(G', I', F') = \mathrm{L}(G, I, F)$. Indeed, for any regular I and F and finite I' and F' the relations $I' \times I$ and $F \times F'$ are automatic. Since term-automatic relations are closed under union and composition, this can be used to build G' from G.

This, however, does not hold in the term-synchronous case. Indeed, since each connected component of a term-synchronous graph is finite, the language of any such graph from a finite set of initial vertices is regular.

Combining Theorem 2.9, Propositions 3.2, 3.3 and 3.4, as well as Remark 3.5, we obtain the following result concerning the family of languages accepted by term-synchronous and term-automatic graphs.

Theorem 3.6. *The languages of term-synchronous graphs between regular sets of vertices and of term-automatic graphs between regular or finite sets of vertices form the complexity class* ETIME.

4. Conclusion

We have proved that the class of languages accepted by alternating linearly bounded machines (ETIME) can also be characterised as the sets of first rows of pictures accepted by arborescent tiling systems, as well as the sets of path labels of term-automatic graphs between regular or finite sets of initial and final vertices.

A natural extension of this work would be to generalise Theorem 3.6 to graphs defined by more expressive classes of tree transducers, in order to fully extend the existing results on rational graphs. In practice, this would require extending the construction for Proposition 3.4 to more general padding techniques.

Further points of interest concern the extension of other results from [4] to term-automatic graphs, in particular regarding structural restrictions of these graphs, like finite or bounded degree, or the restriction to a single initial vertex, as well as a similar study of related complexity classes or families of languages.

References

[1] A. Blumensath and E. Grädel, Automatic structures, in *Proceedings of the 15th IEEE Symposium on Logic in Computer Science (LICS 2000)*, IEEE (2000) 51–62.
[2] N. Chomsky, On certain formal properties of grammars. *Inform. Control* **2** (1959) 137–167.
[3] A. Chandra, D. Kozen and L. Stockmeyer, Alternation. *J. ACM* **28** (1981) 114–133.
[4] A. Carayol and A. Meyer, Context-sensitive languages, rational graphs and determinism. *Log. Meth. Comput. Sci.* **2** (2006).
[5] D. Giammarresi and A. Restivo, *Handbook of Formal Languages*, Vol. **3**, Chap. Two-dimensional languages. Springer (1996).
[6] B. Khoussainov and A. Nerode, Automatic presentations of structures, in *International Workshop on Logical and Computational Complexity (LCC '94)*, Springer (1995) 367–392.
[7] S. Kuroda, Classes of languages and linear-bounded automata. *Inform. Control* **7** (1964) 207–223.
[8] M. Latteux and D. Simplot, Context-sensitive string languages and recognizable picture languages. *Inform. Comput.* **138** (1997) 160–169.
[9] C. Morvan, On rational graphs, in *Proceedings of the 3rd International Conference on Foundations of Software Science and Computation Structures (FoSSaCS 2000). Lect. Notes Comput. Sci.* **1784** (2000) 252–266.
[10] C. Morvan and C. Rispal, Families of automata characterizing context-sensitive languages. *Acta Informatica* **41** (2005) 293–314.
[11] C. Morvan and C. Stirling, Rational graphs trace context-sensitive languages, in *Proceedings of the 26th International Symposium on Mathematical Foundations of Computer Science (MFCS 2001). Lect. Notes Comput. Sci.* **2136** (2001) 548–559.
[12] M. Penttonen, One-sided and two-sided context in formal grammars. *Inform. Control* **25** (1974) 371–392.
[13] C. Rispal, The synchronized graphs trace the context-sensitive languages, in *Proceedings of the 4th International Workshop on Verification of Infinite-State Systems (INFINITY 2002). Elect. Notes Theoret. Comput. Sci.* **68** (2002).