Traces of term-automatic graphs

Antoine Meyer
Institute of Mathematical Sciences, Chennai, India

ameyer@imsc.res.in

Abstract

In formal language theory, many families of languages are defined using grammars or finite acceptors like pushdown automata and Turing machines. For instance, context-sensitive languages are the languages generated by growing grammars, or equivalently the languages accepted by Turing machines whose work tape size is proportional to that of their input.

A few years ago, a new characterization of context-sensitive languages as the path languages of rational graphs (infinite graphs defined by sets of finite-state transducers) was established. We investigate a similar characterization 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, which form the complexity class ETIME. As a technical tool, we also introduce an arborescent variant of tiling systems, which provides yet another characterization of these languages.

Introduction

In classical language theory, context-sensitive languages are defined as the languages generated by growing grammars, and as such they are one of the families of the well-known Chomsky hierarchy [Cho59]. Later, they were also characterized as the languages accepted by linearly space-bounded Turing machines [Kur64], i.e. Turing machines whose runs on any input word of length n use at most $k \cdot n$ cells of their work tape, where k is a constant. In [LS97], it was shown by direct simulations of linearly bounded machines that context-sensitive languages also coincide with the languages accepted by tiling systems.

In 2001, Morvan and Stirling [MS01] provided yet another characterization of this family as the set of path languages of rational graphs [Mor00], a family of infinite graphs whose vertices are words and whose sets of edges are defined using finite transducers. This result was later extended in [Ris02] to the more restricted families of automatic graphs (Cf. [KN95]), and even to synchronous rational graphs when an infinite number of initial and final vertices are considered (for a summary, see also [MR05]). In a way, this provides a "forward", automata-based characterization of context-sensitive languages, as opposed for instance to grammars, or 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, the above-mentioned papers use a normal form for growing grammars, due to Penttonen [Pen74]. In [CM06], these results were reformulated using simpler proof techniques based on tiling systems. Beyond the benefit of mere simplification, this permitted the investigation of interesting sub-cases, in particular concerning deterministic context-sensitive languages or rational graphs with various structural constraints.

The aim of this work is to extend the results in both [LS97] and [CM06] to a more general family of languages, which complexity theorists call ETIME. This family corresponds to those languages accepted by alternating linearly bounded machines, or equivalently by deterministic Turing machines working in time less than $2^{c.n}$, where n is the size of the input word and c is a constant. This family of languages

sits in-between context-sensitive and recursively-enumerable languages in the Chomsky hierarchy. The results we obtain are two new characterizations of ETIME, first as the languages accepted by arborescent tiling systems and second as the traces of infinite graphs defined by various families of term transducers, namely term-synchronous and term-automatic (or tree-automatic) graphs [BG00].
This paper is organized as follows. After recalling relevant definitions and notations in Section 1, we introduce our notion of arborescent tiling systems in Section 2 and prove that they characterize ETIME. Finally, using this technical tool, we extend previously mentioned proofs over rational graphs to the family of term-automatic graphs in Section 3 before concluding with ongoing perspectives.

1 Notations

Words. 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 i-th 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 $\{uv \mid u \in A \text{ and } v \in B\}$.

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 $f \in F_n \wedge t_1, \ldots, t_n \in T(F, X)$ implies $ft_1 \ldots t_n \in T(F, X)$. 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 make the reading easier, $ft_1 \ldots t_n$ will sometimes be written $f(t_1, \ldots, t_n)$.

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, and nodes at maximal positions (i.e. positions x such that $\nexists y \neq \varepsilon,\ xy \in \mathrm{Dom}(t)$) are called leaves, and nodes which are not leaves 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 labeled with constants in F_0 or variables in X and whose internal nodes are labeled with symbols whose arity is 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 ranked tree, position pi with $i \in \mathbb{N}$ always denotes the i-th child of the node at position p. Conversely, any finite ordered tree t labeled over Σ can be represented as a ranked tree t', and hence as a term, simply by mapping each node label a to a corresponding symbol (a, n) in the alphabet $\Sigma \times \mathbb{N}$, where n is the number of children of the considered node, and by renumbering all positions such that the domain of t' verifies properties 1 and 2 above. In the following, we will not distinguish terms from trees. When considering a finite ordered tree as a term, the above transformations will be left implicit.

Finite automata and regular sets. A finite tree (or term) automaton is a tuple $A =< Q, F, q_0, \delta >$, 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 $qf'q_1 \ldots q_n$ with $\mathrm{a}(f) = n$. A run of A over a tree t is a tagging of all nodes of t by

control states such that the root is tagged with q_0 and the children of a node labeled by f are tagged with $q_1 \dots q_n$ if and only if $qf \to q_1 \dots q_n \in \delta$. A run over t is successful is for every leaf labeled by a and tagged with control state q, there exists a rule $qa \to \varepsilon \in \delta$, and we say that t is accepted. The set of trees accepted by any automaton is called its language, and all such languages are said to be *regular*.

Graphs. A labeled, 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*. A sequence of edges $(s_1 \xrightarrow{a_1} t_1, \dots, s_k \xrightarrow{a_k} t_k)$ 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 $u = a_1 \dots a_k$ 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 labeled graph between two sets I and F of vertices is the set of all words w such that there exists a path labeled by w whose origin is in I and destination in F.

Alternating Turing machines. A Turing machine is a tuple $M =< \Gamma, \Sigma, Q, q_0, F, \delta >$ 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 labeled by configuration $q_0 w$, and such that the children of any node labeled by configuration c are labeled by $c_1, \dots, c_n$ 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 of its leaves are labeled by configurations whose control state is in F.

Linearly bounded machines. 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 where the usable tape is limited to the cells initially containing the input word. This may be enforced by forbidding transition rules to rewrite the blank symbol $\Box$ into anything else.
The languages of non-alternating linearly bounded machines form a complexity class noted SPACE($O(n)$). It was shown by Kuroda [Kur64] that this class is equivalent to that of context-sensitive languages. Adding alternation, one obtains the more general class ASPACE($O(n)$). By classical complexity theory results [CKS81], it is also equivalent to the class 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 [CM06].

Tiling systems were originally defined to recognize or specify picture languages, i.e. sets of two-dimensional words on finite alphabets [GR96], called local picture languages. However, by only looking at the words contained in the first row of each picture, one obtains a context-sensitive language [LS97]. In this section, we extend this result to an arborescent extension of tiling systems. We prove that the word languages characterized by this new formalism are precisely the languages accepted by alternating linearly bounded machines.

2.1 Definitions

Instead of planar pictures, we consider so-called arborescent pictures, which are to ordinary pictures what terms are to words. Intuitively, an arborescent picture can be seen as a book with partially open pages.

Definition 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 $\mathrm{AP}(X, m)$.

Again, we assume that X is not only prefix-closed but also left-closed, i.e. $\forall i > 0,\ ui \in X \implies \forall j < i,\ uj \in X$. Arborescent pictures of domain $X \times [1, m]$ are isomorphic to ordered trees of domain X with nodes labeled over the set Γ^m. As such, if $m = 1$ they are isomorphic to Γ-labeled ordered trees. Arborescent pictures whose set of positions X is a subset of 1^* are ordinary, planar pictures. 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.

Definition 2 (Subpicture). 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 recognize meaningful sets of pictures, we first add a border or frame to each picture using a new symbol $\#$.

Definition 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

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

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 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 $X = \{\varepsilon, 1, \ldots, n\}$ and $p_d(\varepsilon, 1)$, $p_d(\varepsilon, 2)$, $p_d(i, 1)$ and $p_d(i, 2)$ are respectively equal to A, B, $\bar{C}(i)$ and $\bar{D}(i)$ (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. $\exists i, C_i = \# \implies A = \# \wedge \forall i, C_i = \#$,
3. $\exists i, D_i = \# \implies B = \# \wedge \forall i, D_i = \#$,
4. $A \neq \# \wedge B \neq \# \implies (C_i = \# \iff D_i = \#)$.

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 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 $\mathrm{P}(S)$.

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 6. *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 behavior.

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 $\bar{A} = A_1 \ldots A_m$, $\bar{C} = C_1 \ldots C_n$ and $\bar{D} = D_1 \ldots D_n$ with $m, n \in [1, \mathrm{a}(S)]$ and for all $k \in [1, m]$, δ contains the rules

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

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 $p_k{}^{\#}_{\#^n}\bar{A} \to f\bar{A}-$ 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, which is tedious but straightforward. □

Proposition 7. *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 $d = pA \to q_1B_1\mu_1 \wedge \dots \wedge q_nB_n\mu_n$. By d_i we refer to the fragment $pA \to q_iB_i\mu_i$ of d, and we let $\mathrm{a}(d) = n$.
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 $[q_0w]$ of M. Thus for all $a, b \in \Sigma$, Δ contains the arity 1 tiles

$$\begin{aligned}
&(\#, [\,, \#, [^{(\perp)}_l),\ ([\,, a, [^{(\perp)}_l, a^{(\perp)}_{q_0}),\ (a, b, a^{(\perp)}_{q_0}, b^{(\perp)}_r),\\
&(a, b, a^{(\perp)}_r, b^{(\perp)}_r),\ (b,], b^{(\perp)}_r,]^{(\perp)}),\ (\,], \#,]^{(\perp)}_r, \#),
\end{aligned}$$

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

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

as well as all dually defined tiling elements of the form

$$
(]^{(d)}_{p}, \#, (]^{(d')}_{p})^n, \#^n), \ ([^{(d)}_{l}, \#, X, \#^n), \ (A^{(d)}_{r}, B^{(d)}_{r}, X, (B^{(d')}_{r})^n) \ \text{ and } \ (A^{(d)}_{p}, B^{(d)}_{r}, X, Y).
$$

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 induction on the length of computations. Removing the border symbols from $\mathrm{L}(S)$ is then a simple exercise. □

From Propositions 6 and 7, we deduce the announced theorem.

Theorem 8. *The languages accepted by arborescent tiling systems form the complexity class* $\mathrm{ASPACE}(O(n)) = \mathrm{DTIME}(2^{O(n)})$.

3 Traces of term-automatic graphs

We now turn to the main result of this paper, which is the study of languages of graphs characterized 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 $\mathrm{ASPACE}(O(n))$. First, we establish this result for the simpler term-synchronous graphs in Section 3.2, then generalize it to term-automatic graphs in Section 3.3.

3.1 Definitions

Let $s = f(s_1 \ldots s_m)$ and $t = g(t_1 \ldots t_n)$ be two terms over some ranked alphabet F. We define the overlap $[st]$ of s and t as the following 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):

$$
\begin{array}{ll}
[st] = fg([s_1 t_1] \ldots [s_k t_k]) & \text{with } k = \max(m, n) \\
 & \text{and } \forall i \in [m, k], j \in [n, k], s_i = t_j = \bot, \\
[s\bot] = f\bot([s_1 \bot] \ldots [s_m \bot]), & \\
[\bot t] = \bot g([\bot t_1] \ldots [\bot t_n]). &
\end{array}
$$

This notation is extended to sets in the natural way. We can now define term-automatic and term-synchronous relations.

Definition 9. A binary relation R is term-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 term-synchronous.

In other words, a term-synchronous is a term-automatic relation which only associates pairs of terms over the same domain (terms with the same structure). Both families of relations are closed under relational composition.

Proposition 10. *Let R_1 and R_2 be two term-automatic (resp. term-synchronous) relations, the relation $R_1 \circ R_2$ is also term-automatic (resp. term-synchronous).*

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 11. 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

Proposition 12. *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 $G = (R_a)_{a\in\Sigma}$ 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_i be a finite top-down term automaton accepting $[R_a]$, 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 is a shorthand notation for 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, as defined in the previous section. For every path $t_0 \xrightarrow{a_1} t_1 \ldots \xrightarrow{a_n} t_n$ 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

- $p|_{\varepsilon,0,X,1}$ is isomorphic to the term $[\,(\rho_0)$, where $[$ is considered a unary symbol and ρ_0 is an accepting run of $A_{I\circ a}$ 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 $[t_{i-1} t_i]$
- $p|_{\varepsilon,n+1,X,1}$ is isomorphic to the term $]\,(\rho_{n+1})$, where $]$ is considered a unary symbol and ρ_{n+1} is an accepting run of $A_{a\circ F}$ 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

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

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 13. *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 $S = (\Gamma, \#, \Delta)$ 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 labeled 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:

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

We define I as the regular set of all terms labeled 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:

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

By construction of each of the A_a, of I and of A_F, there exists a path in G labeled 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. □

Combining Propositions 12 and 13, we obtain the following result concerning the family of languages accepted by term-synchronous graphs.

Theorem 14. *The languages of term-synchronous graphs between regular sets of initial and final vertices are the languages accepted by arborescent tiling systems.*

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 15. *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 and I, F by A_I and A_F respectively. We define a synchronous graph $G' = (R'_a)_{a\in\Sigma}$ and two regular sets I' and F' such that $\mathrm{L}(G,I,F) = \mathrm{L}(G',I',F')$. 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 $t \in T(\Gamma)$,

$$\begin{aligned} \phi(t) \;=\; \{t' \in T(\Gamma') \mid \mathrm{Dom}(t) \subset \mathrm{Dom}(t'),\ \forall p \in \mathrm{Dom}(t), t'(p) = t(p) \\ \text{and } \forall p \in \mathrm{Dom}(t') \setminus \mathrm{Dom}(t), t'(p) \in \{\#_0, \#_n\}. \end{aligned}$$

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 $a \in \Sigma$, the relation R'_a as $\{(s',t') \mid (s,t) \in R_a,\ s' \in \phi(s),\ t' \in \phi(t) \text{ and } \mathrm{Dom}(s') = \mathrm{Dom}(t')\}$. This synchronous relation can be characterized 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 $0 \le k \le n$, 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 *unique* 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' labeled by $a_1 \dots a_n$, hence that $\mathrm{L}(G,I,F) \subseteq \mathrm{L}(G',I',F')$. □

Remark 16. It can easily be shown 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. Hence, since term-automatic relations are closed under composition, this can be used to build G' from G.

4 Conclusion

We have presented the proof that ETIME, the class of languages accepted by alternating linearly bounded machines, can also be characterized 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. However, to fully extend the existing results on rational graphs, one would have to investigate graphs defined by a family of term transducers at least as general as word transducers. Ongoing work focuses on extending the construction in Section 3.3 to graphs defined by linear top-down tree transducers with regular look-ahead and ε-transitions. Further points of interest concern the extension of other results from [CM06] to term-automatic graphs, in particular regarding the traces of structural restrictions of these graphs (finite or bounded degree, single initial vertex), as well as the similar study of other complexity classes or families of languages around context-sensitive languages.

References

[BG00] A. Blumensath and E. Grädel. Automatic structures. In *Proceedings of the 15th IEEE Symposium on Logic in Computer Science (LICS 2000)*, pages 51–62. IEEE, 2000.

[Cho59] N. Chomsky. On certain formal properties of grammars. *Information and Control*, 2:137–167, 1959.

[CKS81] A. Chandra, D. Kozen, and L. Stockmeyer. Alternation. *Journal of the ACM*, 28(1):114–133, 1981.

[CM06] A. Carayol and A. Meyer. Context-sensitive languages, rational graphs and determinism. *Logical Methods in Computer Science*, to appear (preliminary version available at `http://www.liafa.jussieu.fr/~ameyer/`), 2006.

[GR96] D. Giammarresi and A. Restivo. *Handbook of Formal Languages*, volume 3, chapter Two-dimensional languages. Springer, 1996.

[KN95] B. Khoussainov and A. Nerode. Automatic presentations of structures. In *International Workshop on Logical and Computational Complexity (LCC '94)*, pages 367–392. Springer, 1995.

[Kur64] S. Kuroda. Classes of languages and linear-bounded automata. *Information and Control*, 7(2):207–223, 1964.

[LS97] M. Latteux and D. Simplot. Context-sensitive string languages and recognizable picture languages. *Information and Computation*, 138(2):160–169, 1997.

[Mor00] C. Morvan. On rational graphs. In *Proceedings of the 3rd International Conference on Foundations of Software Science and Computation Structures (FoSSaCS 2000)*, volume 1784 of *Lecture Notes in Computer Science*, pages 252–266. Springer, 2000.

[MR05] Christophe Morvan and Chloé Rispal. Families of automata characterizing context-sensitive languages. *Acta Informatica*, 41(4-5):293–314, 2005.

[MS01] 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)*, volume 2136 of *Lecture Notes in Computer Science*, pages 548–559. Springer, 2001.

[Pen74] M. Penttonen. One-sided and two-sided context in formal grammars. *Information and Control*, 25(4):371–392, 1974.

[Ris02] 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)*, volume 68 of *Electronic Notes in Theoretical Computer Science*, 2002.