Fundamenta Informaticae X (2) 1–20
IOS Press

The complexity of unary tiling recognizable picture languages: nondeterministic and unambiguous cases [*]

Alberto Bertoni

Massimiliano Goldwurm

Violetta Lonati

Dipartimento di Scienze dell'Informazione, Università degli Studi di Milano
Via Comelico 39/41, 20135 Milano – Italy
$\{$*bertoni, goldwurm, lonati*$\}$*@dsi.unimi.it*

Abstract. In this paper we consider the classes REC_1 and UREC_1 of unary picture languages that are tiling recognizable and unambiguously tiling recognizable, respectively. By representing unary pictures by quasi-unary strings we characterize REC_1 (resp. UREC_1) as the class of quasi-unary languages recognized by nondeterministic (resp. unambiguous) linearly space-bounded one-tape Turing machines with constraint on the number of head reversals. We apply such a characterization in two directions. First we prove that the binary string languages encoding tiling recognizable unary square languages lies between $\text{NTIME}(2^n)$ and $\text{NTIME}(4^n)$; by separation results, this implies there exists a non-tiling recognizable unary square language whose binary representation is a language in $\text{NTIME}(4^n \log n)$. In the other direction, by means of results on picture languages, we are able to compare the power of deterministic, unambiguous and nondeterministic one-tape Turing machines that are linearly space-bounded and have constraint on the number of head reversals.

Keywords: two-dimensional languages, tiling systems, linearly space-bounded Turing machine, head reversal-bounded computation.

Address for correspondence: M. Goldwurm, Dip. Scienze dell'Informazione, Università degli Studi di Milano, Via Comelico 39/41, 20135 Milano, Italy

[*]Partially appeared in *Proc. 24th STACS*, W. Thomas and P. Pascal eds, LNCS 4393, 381–392, Springer, 2007.

1. Introduction

Picture languages have been introduced in the literature as two-dimensional extension of traditional string languages, a picture being a two-dimensional array of elements from a finite alphabet. They have been originally considered as formal models for image processing in connection with problems of pattern recognition. Several classical tools and concepts have been used to classify picture languages and study their properties: regular expressions [10], grammars [16], automata [8], logic formulas [7].

One of the main effort in this area is to capture the notion of recognizability. In particular, various notions of two-dimensional finite automaton have been proposed and studied in the literature [8, 9]. An interesting formal model for the recognition of picture languages is given by the so-called tiling systems introduced in [5], which are based on projection of local properties. A tiling system τ is defined by a finite set Θ of square pictures of size 2 together with a projection between alphabets. Roughly speaking, a language is recognized by τ if each of its elements can be obtained as a projection of a picture whose subpictures of size 2×2 belong to Θ. The class of picture languages recognized by such systems satisfies relevant properties, which resemble classical properties of regular string languages [6].

A special case is represented by pictures over a one-letter alphabet: in this case only the shape of the picture is relevant, and hence a unary picture is simply identified by a pair of positive integers. In this context, a general goal is to define techniques to describe families of recognizable languages, or to construct examples of non-recognizable languages [6, 9]. For instance, families of tiling-recognizable unary picture languages are introduced in [6] by means of integer functions or in [2] by means of special regular expressions, whereas in [9] two-dimensional automata are used to recognize unary languages and several strategies to explore pictures are presented.

In this work we give a complexity result concerning the unary picture languages recognized by tiling systems and by unambiguous tiling systems. We characterize these two classes respectively by means of nondeterministic and unambiguous Turing machines that are space and head reversal-bounded. More precisely, we introduce a notion of quasi-unary string to represent pairs of positive numbers and we prove that a unary picture language L is tiling recognizable (unambiguous tiling recognizable, respectively) if and only if the set of all quasi-unary strings encoding the sizes of the elements of L is recognizable by a nondeterministic (unambiguous, resp.) one-tape Turing machine that works within $\max(n, m)$ space and executes at most $\min(n, m)$ head reversals, on the input representing the pair (n, m).

In particular for the case of squares, this result allows us to relate the recognizability of unary square pictures to nondeterministic time complexity bounds. Informally, it shows that the complexity of the binary encodings of tiling recognizable unary square picture languages is located between $\mathrm{NT{\small IME}}(2^n)$ and $\mathrm{NT{\small IME}}(4^n)$. This yields a large variety of examples of picture languages that are tiling recognizable. For instance, all binary languages in NP correspond to tiling recognizable (unary square) picture languages. The same holds for the binary languages that are $\mathrm{NE{\small XP}T{\small IME}}$-complete [14].

Also, our characterization allows us to use separating results on time complexity classes as a tool for defining recognizable and non-recognizable unary picture languages. In particular, using a property proved in [15], we show the existence of a unary square language that is not tiling recognizable, but corresponds to a binary string language recognizable in nondeterministic time $O(4^n \log n)$.

Finally, we use properties of unambiguous and nondeterministic unary tiling recognizable picture languages to separate complexity classes. In particular, we consider the classes $\mathrm{DS{\small PACE}R{\small EV}}_Q$, $\mathrm{US{\small PACE}R{\small EV}}_Q$ and $\mathrm{NS{\small PACE}R{\small EV}}_Q$ of quasi-unary languages accepted, respectively, by deterministic, unambiguous and nondeterministic one-tape Turing machines working in linear space with constraint on the number of head reversals. Here, our main result is that the inclusions

$$\text{DSPACEREV}_Q \subset \text{USPACEREV}_Q \subset \text{NSPACEREV}_Q$$

are strict. The second strict inclusion is a consequence of a recent result appeared in [4], while the first one follows from a sort of iteration lemma for pictures of crossing sequences of Turing machine computations.

2. Preliminaries on picture languages

Given a finite alphabet Σ, a *picture* (or *two-dimensional string*) over Σ is either a two-dimensional array (i.e., a matrix) of elements of Σ or the *empty picture* λ. The set of all pictures over Σ is denoted by Σ^{**}; a *picture language* (or *two-dimensional language*) over Σ is a subset of Σ^{**}.

Given a picture $p \in \Sigma^{**}$, we use r_p and c_p to denote the number of rows and columns of p, respectively. The pair $(\mathrm{r}_p, \mathrm{c}_p)$ is called the *size* of p. By definition we have $\mathrm{r}_p > 0$ and $\mathrm{c}_p > 0$, except for the empty picture λ that has size $(0, 0)$. The symbol in p with coordinates (i, j) is denoted by $p(i, j)$, for every $1 \le i \le \mathrm{r}_p$ and $1 \le j \le \mathrm{c}_p$. We say that an element $p \in \Sigma^{**}$ is a horizontal rectangle, a vertical rectangle, or a square according whether $\mathrm{r}_p < \mathrm{c}_p$, $\mathrm{r}_p > \mathrm{c}_p$, or $\mathrm{r}_p = \mathrm{c}_p$, respectively. If p is a square, then the size of p is simply r_p. A *square language* is a picture language containing only square pictures. If the alphabet Σ is a singleton, then the pictures over Σ^{**} are called *unary pictures*. A *unary picture language* is a subset of Σ^{**}, where Σ is a singleton.

For any picture $p \in \Sigma^{**}$ of size (m, n), we use $\hat{p}$ to denote a new picture of size $(m + 2, n + 2)$ obtained by surrounding p with a special boundary symbol $\sharp \notin \Sigma$. Such boundary will be useful when describing scanning strategies for pictures.

Many operations can be defined between pictures or picture languages. In particular, we recall the operations of row and column concatenation. Let p and q be pictures over Σ^{**} of size $(\mathrm{r}_p, \mathrm{c}_p)$ and $(\mathrm{r}_q, \mathrm{c}_q)$, respectively. If $\mathrm{r}_p = \mathrm{r}_q$, we define the column concatenation $p \mathbin{\varobar} q$ between p and q as the picture of size $(\mathrm{r}_p, \mathrm{c}_p + \mathrm{c}_q)$ whose i-th row equals the concatenation of the i-th rows of p and q, for every $1 \le i \le \mathrm{r}_p$. If $\mathrm{c}_p = \mathrm{c}_q$, we define the row concatenation $p \ominus q$ analogously. Clearly, $\varobar$ and $\ominus$ are partial operations over the set Σ^{**}. These definitions can be extended to picture languages and then iterated: for every language $L \subseteq \Sigma^{**}$, we set $L^{0\varobar} = L^{0\ominus} = \{\lambda\}$, $L^{i\varobar} = L \varobar L^{(i-1)\varobar}$ and $L^{i\ominus} = L \varobar L^{(i-1)\ominus}$, for every $i \ge 1$. Thus, one can define the *row* and *column closures* as the transitive closures of $\varobar$ and $\ominus$:

$$L^{*\varobar} = \bigcup_{i \ge 0} L^{i\varobar} \qquad L^{*\ominus} = \bigcup_{i \ge 0} L^{i\ominus},$$

which can be seen as a sort of *two-dimensional Kleene star*. Another useful operation is the so-called *rotation*: given $p \in \Sigma^{**}$, its rotation p^R is the picture of size $(\mathrm{c}_p, \mathrm{r}_p)$ defined by

$$p^R(i, j) = p(\mathrm{r}_p + 1 - j, i) \qquad \text{for every } i, j.$$

Several approaches have been proposed to capture the notion of recognizability of picture languages. Here we consider the class REC and its definition in terms of tiling systems [5, 6]. First, we recall the definition of *local picture language*.

Definition 2.1. A *tile* is a square picture of size 2; for every picture p, $T(p)$ denotes the set of all tiles that are subpictures of p. A picture language $L \subseteq \Gamma^{**}$ is called *local* if there exists a finite set Θ of tiles over the alphabet $\Gamma \cup \{\sharp\}$ such that $L = \{p \in \Gamma^{**} \mid T(\hat{p}) \subseteq \Theta\}$. In this case we write $L = \mathcal{L}(\Theta)$.

We also need the notion of projection of pictures and picture languages. Let $\pi : \Gamma \to \Sigma$ be a mapping between two alphabets. Given a picture $p \in \Gamma^{**}$, the *projection* of p by π is the picture $\pi(p) \in \Sigma^{**}$ such that $\pi(p)\ (i, j) = \pi(p(i, j))$ for every pair of coordinates i, j. Analogously, the projection of a language $L \subseteq \Gamma^{**}$ by π is the set $\pi(L) = \{\pi(p) \mid p \in \Gamma^{**}\} \subseteq \Sigma^{**}$.

Definition 2.2. A *tiling system* is a 4-tuple $\tau = \langle \Sigma, \Gamma, \Theta, \pi \rangle$ where Σ and Γ are two finite alphabets, Θ is a finite set of tiles over the alphabet $\Gamma \cup \{\sharp\}$ and $\pi : \Gamma \to \Sigma$ is a projection. A picture language $L \subseteq \Sigma^{**}$ is *tiling recognizable* if there exists a tiling system $\langle \Sigma, \Gamma, \Theta, \pi \rangle$ such that $L = \pi(\mathcal{L}(\Theta))$. We say that τ generates L and denote by REC the class of picture languages that are tiling recognizable.

Notice in particular that any local language is tiling recognizable.

The class REC satisfies some remarkable properties. For instance it can be defined as the class of languages recognized by online tessellation automata, that are special acceptors related to cellular automata [8]; they can be expressed by formulas of existential monadic second order [7]; they can be defined by means of regular-like expressions based on certain composition rules between pictures [6]. In particular we will use the fact that REC is closed with respect to the operations $\cup, \ominus, \oplus, ^{*\ominus}, ^{*\oplus}, ^{R}$.

Since we are interested in unary pictures, we denote by REC$_1$ the class of unary picture languages that are tiling recognizable. Clearly REC$_1 \subseteq$ REC.

Another relevant subclass of REC consists of the tiling recognizable languages whose pictures are the projection of a unique element in the corresponding local language.

Definition 2.3. A tiling system $\tau = \langle \Sigma, \Gamma, \Theta, \pi \rangle$ is *unambiguous* if, for every $q, q' \in \mathcal{L}(\Theta)$, $\pi(q) = \pi(q')$ implies $q = q'$. A language $L \subseteq \Sigma^{**}$ is *unambiguously* tiling recognizable if L is generated by an unambiguous tiling system. The family of all unambiguous tiling recognizable picture languages is denoted by UREC. Moreover, we define UREC$_1$ as the class REC$_1 \cap$ UREC.

The unambiguous tiling recognizable picture languages have been introduced in [5]. In particular, it is known that the inclusion UREC $\subset$ REC is strict and that it is undecidable whether a tiling system is unambiguous [3]. The unary unambiguous languages in REC are also studied in [4], where their relationship with deterministic classes of recognizable picture languages is investigated and, using results appeared in [11, 12], it is shown that the inclusion UREC$_1 \subset$ REC$_1$ is strict.

3. Characterization of REC$_1$ and UREC$_1$

In this section, we state our main result, that is a characterization of the class of unary picture languages that are tiling recognizable, in terms of computational complexity.

To this aim, consider the alphabet $\{\circ\}$ and notice that any unary picture $p \in \{\circ\}^{**}$ is identified by its size, that is by the pair $(\mathrm{r}_p, \mathrm{c}_p)$. Thus, unary pictures (i.e. pairs of positive integers) can be encoded by quasi-unary strings as follows. We consider the set of unary strings over $\{\circ\}$

$$U = \{\circ^n \mid n > 0\}$$

and the following sets of strings that are unary except for one special letter h or v (not occurring in first position):

$$\begin{aligned} Q_h &= \{\circ^n h \circ^k \mid n > 0, k \geq 0\} , \\ Q_v &= \{\circ^n v \circ^k \mid n > 0, k \geq 0\} . \end{aligned}$$

We call *quasi-unary string* over the alphabet $\{\circ, h, v\}$ any string in $Q = U \cup Q_h \cup Q_v$. The length of any quasi-unary string x is denoted as usual by $|x|$, whereas we use ${}_\circ|x|$ to denote the length of the longest prefix of x in $\circ^+$. The use of symbols h and v allows us to distinguish among squares, horizontal and vertical rectangles. Thus, a quasi-unary string $x \in Q_h$ represents the unary horizontal rectangle of size $({}_\circ|x|, |x|)$; $x \in Q_v$ represents the unary vertical rectangle of size $(|x|, {}_\circ|x|)$; whereas $x \in U$ represents the unary square of size $|x|$.

Summarizing the previous definitions, the encoding ϕ from unary pictures to quasi-unary strings can be stated as follows: for every picture $p \in \{\circ\}^{**}$, we have

$$\phi(p) = \begin{cases} \circ^{\mathrm{r}_p}\, h \,\circ^{\mathrm{c}_p - \mathrm{r}_p - 1} & \text{if } \mathrm{r}_p < \mathrm{c}_p \\ \circ^{\mathrm{r}_p} & \text{if } \mathrm{r}_p = \mathrm{c}_p \\ \circ^{\mathrm{c}_p}\, v \,\circ^{\mathrm{r}_p - \mathrm{c}_p - 1} & \text{if } \mathrm{r}_p > \mathrm{c}_p \end{cases} \qquad (1)$$

Notice that $|\phi(p)| = \max(\mathrm{r}_p, \mathrm{c}_p)$, while ${}_\circ|\phi(p)| = \min(\mathrm{r}_p, \mathrm{c}_p)$. Moreover, for every $L \subseteq \{\circ\}^{**}$, we set $\phi(L) = \{\phi(p) \mid p \in L\}$.

Note that function ϕ can be extended to Σ^{**} for an arbitrary alphabet Σ. Thus $\phi(p)$ is defined as in (1) for every $p \in \Sigma^{**}$. Clearly if Σ includes more than one element then ϕ is not an encoding of pictures in Σ^{**}: for every $p \in \Sigma^{**}$, $\phi(p)$ only represents the shape of p.

Now, let us introduce the complexity classes of quasi-unary languages that we shall use to characterize the class of tiling-recognizable unary languages. Our main model of computation is the one-tape nondeterministic Turing machine, that we denote by 1t-NTM for short.

Definition 3.1. NSPACEREV$_Q$ is the class of quasi-unary string languages that can be recognized by a 1t-NTM working within $|x|$ space and executing at most ${}_\circ|x|$ head reversals, for any input x in Q.

In the previous definition we assume that the first and the last symbol of the input are marked in a special way so that the machine only scans the cells of tape containing the input. We call this set of cells the *work portion* of the tape.

We recall that a 1t-NTM is *unambiguous* if every input string accepted by the machine admits a unique accepting computation. We denote such machine by 1t-UTM. Then, we use USPACEREV$_Q$ to denote the class of all quasi-unary string languages that can be recognized by a 1t-UTM working in $|x|$ space with at most ${}_\circ|x|$ head reversals, for any input x in Q.
Our main theorem can then be stated as follows:

Theorem 3.1. A unary picture language L is in REC$_1$ if and only if $\phi(L)$ belongs to NSPACEREV$_Q$. Moreover, L is in UREC$_1$ if and only if $\phi(L)$ belongs to USPACEREV$_Q$.

The proof of Theorem 3.1 is split into two parts presented in Sections 4 and 5, respectively. The first part shows that $L \in$ REC$_1$ and $L \in$ UREC$_1$ imply, respectively, $\phi(L) \in$ NSPACEREV$_Q$ and $\phi(L) \in$ USPACEREV$_Q$. The second part proves the inverse implications.

$$t_1 = \begin{array}{|c|c|}\hline \sharp & \sharp \\ \hline \sharp & a \\ \hline \end{array} \qquad t_2 = \begin{array}{|c|c|}\hline \sharp & \sharp \\ \hline b & b' \\ \hline \end{array} \qquad t_3 = \begin{array}{|c|c|}\hline \sharp & \sharp \\ \hline c & \sharp \\ \hline \end{array}$$

4. From recognizability to complexity bounds

To prove the first part of the main theorem, we show that for every $L \in \text{REC}$ the quasi-unary string language $\phi(L)$ belongs to NSPACEREV_Q. This clearly yields a complexity bound on the shapes of all tiling recognizable languages. The statement is a consequence of the following result concerning the recognition of the size encodings of local languages.

Lemma 4.1. Let Θ be a finite set of tiles over an arbitrary alphabet Γ. Then, $\phi(\mathcal{L}(\Theta))$ belongs to NSPACEREV_Q.

Proof:
We define a 1t-NTM M that, for any input $x \in Q$, nondeterministically tries to generate some $p \in \mathcal{L}(\Theta)$ such that $\phi(p) = x$. First of all, M establishes if $x \in Q_h$, $x \in Q_v$, or $x \in U$. This can be done nondeterministically without head reversals. If $x \in Q_h$ or $x \in U$, then the generation is performed row by row, otherwise the generation has to be done column by column. The input is accepted if and only if such a generating process can be accomplished. We describe the computation in details only when $x \in Q_h$ since the other cases are similar.

The working alphabet Γ' of M contains the symbols $\circ, h, v, \sharp$ $\not b$, all the pairs $(a, b) \in (\Gamma \cup \{\sharp\}) \times (\Gamma \cup \{\sharp\})$, and their marked versions $\overline{(a, b)}$ and $\widetilde{(a, b)}$. The symbols (a, b) shall be used in correspondence with a pair of adjacent symbols in some column of the picture p generated during the computation; the overlined symbols shall be used as bookmarks at the ${}_\circ|x|$-th cell, tildes shall be used to implement a counter.

Given an input x, the machine M behaves as follows:

1. First of all, M reads the tape rightwards till the last input symbol, nondeterministically replacing each input symbol according to Θ, whenever such a replacement is possible. More precisely:

   - the leftmost symbol is replaced by some pair $(\sharp, a)$ such that the tile t_1 in the figure below belongs to Θ;
   - any next symbol is replaced by some pair $(\sharp, b)$ in such a way that, for each pair of consecutive pairs $(\sharp, b)$ and $(\sharp, b')$, the tile t_2 in the figure belongs to Θ (the position of the symbol h is preserved by using overlined pairs);
   - the rightmost symbol $\circ$ is replaced by some pair $(\sharp, c)$ such that the tile t_3 in the figure belongs to Θ. At any position, if no replacement is allowed, then M halts and rejects.

2. M changes direction and reads all the work portion of the tape without head reversals, replacing each symbol (a, b) by (b, c) so that each pair of consecutive symbols do respect Θ (as in point 1). Such a procedure is repeated $({}_\circ|x| - 1)$-many times: this task can be performed by using the first $({}_\circ|x|)$ cells of the tape (those that precede some overlined symbol of Γ') and marking one cell with

a tilde at each repetition. Also during this phase, if no replacement is allowed, then M halts and rejects.

3. After the $({}_{\circ}|x|-1)$-th repetition of step 2, M changes direction and reads all the tape again without head reversals, replacing each symbol (a,b) by $(b,\sharp)$ according to Θ. If no replacement is allowed then M halts and rejects.

The input x is accepted if and only if the procedure can be concluded, that is, if and only if there exists a picture $p \in \mathcal{L}(\Theta)$ of size $({}_{\circ}|x|, |x|)$. Since the machine M works exactly in space $|x|$ and executes exactly ${}_{\circ}|x|$ head reversals, the proof is complete. □

Theorem 4.1. For every $L \in$ REC the quasi-unary string language $\phi(L)$ belongs to NSPACEREV$_Q$.

Proof:
Let $\langle\Sigma,\Gamma,\Theta,\pi\rangle$ be a tiling system generating L and let M be the corresponding Turing machine that recognizes $\phi(\mathcal{L}(\Theta))$, as defined in the proof of Lemma 4.1. Since for each $p \in L$ the word $\phi(p)$ forgets the content of p and only depends on its shape, we have $\phi(L) = \phi(\mathcal{L}(\Theta))$ and hence M just recognizes the set $\phi(L)$. □

The proof of the first part of Theorem 3.1 is completed by the following

Proposition 4.1. For every $L \in$ UREC$_1$, the quasi-unary string language $\phi(L)$ belongs to USPACEREV$_Q$.

Proof:
Let $\langle\{\circ\},\Gamma,\Theta,\pi\rangle$ be an unambiguous tiling system generating L. Then, for each $x \in \phi(L)$, there exists only one picture $q \in \mathcal{L}(\Theta)$ such that $\phi(q) = x$. Now consider the 1t-NTM M that recognizes $\phi(\mathcal{L}(\Theta))$: from the proof of Lemma 4.1 it is clear that the number of accepting computations of M for an input $x \in Q$ equals the number of $q \in \mathcal{L}(\Theta)$ such that $\phi(q) = x$. Therefore, by the previous observation, M is unambiguous and hence $\phi(L) = \phi(\mathcal{L}(\Theta))$ belongs to USPACEREV$_Q$. □

Note that the previous property does not hold for general $L \in$ UREC, since L may contain two different pictures with the same shape.

5. From complexity bounds to recognizability

To prove the second part of Theorem 3.1, we first introduce an auxiliary picture language representing the accepting computations of a 1t-NTM. A similar approach is used in [5] to prove that the emptiness problem for the family REC is undecidable.

5.1. The accepting-computation picture language of a 1t-NTM

Let M be a 1t-NTM and let Σ and Λ be the input and the working alphabet (Λ contains the blank symbol $\not{b}$). We denote by Q the set of states, which includes the initial state q_0 and a unique accepting state q_{yes}. Also let $\delta : Q \times \Lambda \to 2^{Q\times\Lambda\times\{+,-\}}$ be the transition function of M. Without loss of generality, we assume M can never print the blank symbol $\not{b}$, and hence $(q,c,x) \in \delta(p,a)$ implies $c \neq \not{b}$. However, here the machine is not space bounded and it can scan the first blank of the tape: in this case we assume that such a blank is part of the work portion of the tape.

Then, set $\Lambda_Q = \{\sigma_q \mid \sigma \in \Lambda, q \in Q\}$, a configuration of M is a string $C = x\sigma_q y \in \Lambda^*\Lambda_Q\Lambda^*$ which represents the instantaneous description of the machine where $x\sigma y$ is the work portion of the tape, q is the current state and the head scans the cell containing σ on the right of x. If $q = q_0$ and x is the empty string, then C is the initial configuration of M on input σy. If $q = q_{\text{yes}}$ then C is an accepting configuration. We assume the machine halts in every accepting configuration.

Given two configurations C and D of M, we write $C \rhd D$ whenever M can go from C to D without in-between head reversals, possibly by several distinct moves.

Formally, we define an accepting computation[1] of M on input $x \in \Sigma^*$ as a string of the form

$$W = W_1 \rhd W_2 \rhd \cdots \rhd W_n \tag{2}$$

such that all W_j's are configurations of M, W_1 is the initial configuration on input x, W_n is an accepting configuration, $W_i \rhd W_{i+1}$ holds for each $i = 1, \ldots, n-1$, and at configuration W_i the machine executes a head reversal for every $1 < i < n$. Thus, from W_{i-1} to W_i and from W_i to W_{i+1}, the head moves to opposite directions. We say that the sequence of moves from W_i to W_{i+1} is a *run* of the computation, for any $i = 1, \ldots, n-1$ (and hence a computation can be seen as a sequence of runs).

Given an accepting computation W, let $m = \max_i |W_i|$ and consider the picture of size $n \times m$ containing the string W_i (possibly followed by $\not b$'s) on the i-th row, for $1 \le i \le n$. Notice that, from such a picture, one can recover the input and the sequence of runs but not the complete step-by-step computation on the same input. Since M can never print the blank, the last row such a picture does not contain any $\not b$ (but it may include a symbol of the form $\not b_q$).

The *accepting-computation language* of M is defined as the set $A(M)$ of all pictures corresponding to any accepting computation of M. Note that every accepting computation W of M corresponds to a picture $w \in A(M)$ such that $\mathrm{r}_w - 2$ equals the number of head reversals executed in W (corresponding to $W_2, \cdots, W_{n-1}$) and c_w is the space used in W.

Example 5.1. Let M be a 1t-NTM that has working alphabet $\{a, b, c\}$, set of states $Q = \{0, 1, 2, 3, 4, 5, y\}$ and accepting state y. Then, for the input $x = abacbcca$, consider the sequence of moves represented in the following table, where $(\sigma', q', *) \in \delta(\sigma, q)$:

q	0	1	4	2	1	4	5	1	4	0	2	3	2	4	2	0
σ	a	b	a	c	b	c	a	b	c	b	c	a	b	a	a	c
q'	1	4	2	1	4	5	1	4	0	2	3	2	4	2	0	y
σ'	c	a	c	b	a	b	c	a	b	a	b	b	a	c	b	b
$*$	$+$	$+$	$+$	$+$	$+$	$-$	$-$	$+$	$+$	$+$	$+$	$-$	$-$	$+$	$-$	$-$

The picture w associated with such a computation $W = W_1 \rhd W_2 \rhd \cdots \rhd W_7$ is given by

[1] Usually, the computation is a description of the sequence of all single moves executed by the machine on a given input. Rather, here we refer to this concept using the expression *step-by-step computation*.

$$w = \begin{array}{|c|c|c|c|c|c|c|c|}\hline a_0 & b & a & c & b & c & c & a \\ \hline c & a & c & b & a & c_4 & c & a \\ \hline c & a & c & b_1 & c & b & c & a \\ \hline c & a & c & a & b & a & b & a_3 \\ \hline c & a & c & a & b & a_4 & a & b \\ \hline c & a & c & a & b & c & a_2 & b \\ \hline c & a & c & a & b_y & b & b & b \\ \hline \end{array} \begin{array}{l} \to W_1 \\ \to W_2 \\ \to W_3 \\ \to W_4 \\ \to W_5 \\ \to W_6 \\ \to W_7 \end{array}$$

Proposition 5.1. The accepting-computation language of any 1t-NTM belongs to REC. Moreover, the accepting-computation language of any 1t-UTM belongs to UREC.

Proof:
Let L be the accepting-computation language of a 1t-NTM M. We show that L is the projection of a local language L'. To this aim, for every picture $w \in L$, we define a new picture w' obtained from w by changing some symbols, and set $L' = \{w' \mid w \in L\}$. To build w' we proceed as follows, recalling that the i-th row of w corresponds to a configuration W_i defined as in (2), for every $i \geq 1$.

- We mark all symbols of Λ_Q occurring in w by replacing σ_q by $\overrightarrow{\sigma_q}$ when it occurs on odd rows, and by $\overleftarrow{\sigma_q}$ when it occurs on even rows, except for the last row where $\sigma_{q_{yes}}$ remains unchanged. Now, for every $i \geq 1$, let j_i be the column index such that $w(i, j_i)$ is in Λ_Q. Then, for each $1 \leq i < n$, $w'(i, j_i)$ contains an arrow denoting the direction of the head during the run from configuration W_i to configuration W_{i+1}.
- Then, for each $i \geq 2$, we modify the symbols in the i-th row included between the columns of indices j_{i-1} and j_i. Notice that this is the portion of the tape modified during the run from W_{i-1} to W_i. More precisely, we replace each symbol $\sigma \in \Lambda$ in this portion by ${}^q\sigma$, where $q \in Q$ is the state entered by the machine just before printing σ on the tape.

Now, let Γ be the alphabet containing Λ and all symbols ${}^q\sigma, \overrightarrow{\sigma_q}, \overleftarrow{\sigma_q}$ where $\sigma \in \Lambda, q \in Q$, and consider the morphism $\pi : \Gamma \to \Lambda \cup \Lambda_Q$ such that $\pi(\sigma) = \pi({}^q\sigma) = \sigma$ and $\pi(\overrightarrow{\sigma_q}) = \pi(\overleftarrow{\sigma_q}) = \sigma_q$. Clearly, for every picture $w \in L$, we have $w' \in \Gamma^{**}$ and $\pi(w') = w$. Therefore, the result is proved if we provide a representation by tiles for the picture language $L' = \{w' \in \Gamma^{**} \mid w \in L\}$.

The upper left cell of any $w' \in L'$ contains $\overrightarrow{\sigma_{q_0}}$, where q_0 is the initial state and $\sigma \in \Sigma$, while all other cells in the first row are in $\Sigma \cup \{\not b\}$, thus we have the tiles

$$\begin{array}{|c|c|}\hline \sharp & \sharp \\ \hline \sharp & \overrightarrow{a_{q_o}} \\ \hline \end{array} \quad \begin{array}{|c|c|}\hline \sharp & \sharp \\ \hline \overrightarrow{a_{q_o}} & b \\ \hline \end{array} \quad \begin{array}{|c|c|}\hline \sharp & \sharp \\ \hline a & b \\ \hline \end{array} \quad \begin{array}{|c|c|}\hline \sharp & \sharp \\ \hline \not b & \not b \\ \hline \end{array} \quad \begin{array}{|c|c|}\hline \sharp & \sharp \\ \hline b & \sharp \\ \hline \end{array}$$

for every $a \in \Sigma$, $b \in \Sigma \cup \{\not b\}$. The last row of any picture in L' must contain q_{yes} and no $\not b$'s; thus we have to add the tiles

$$\begin{array}{|c|c|}\hline \sharp & a_{q_{yes}} \\ \hline \sharp & \sharp \\ \hline \end{array} \quad \begin{array}{|c|c|}\hline \sharp & a \\ \hline \sharp & \sharp \\ \hline \end{array} \quad \begin{array}{|c|c|}\hline b_{q_{yes}} & \sharp \\ \hline \sharp & \sharp \\ \hline \end{array} \quad \begin{array}{|c|c|}\hline a & \sharp \\ \hline \sharp & \sharp \\ \hline \end{array}$$

$$\begin{array}{|c|c|}\hline a_{q_{yes}} & b \\ \hline \sharp & \sharp \\ \hline \end{array}\ \begin{array}{|c|c|}\hline a & b_{q_{yes}} \\ \hline \sharp & \sharp \\ \hline \end{array}\ \begin{array}{|c|c|}\hline a & b \\ \hline \sharp & \sharp \\ \hline \end{array}\ \begin{array}{|c|c|}\hline \not b_{q_{yes}} & \sharp \\ \hline \sharp & \sharp \\ \hline \end{array}$$

for any $a, b, \in \Lambda$, $a, b \neq \not b$. Moreover, we have to add some tiles corresponding to the moves of the machine. For sake of brevity, from now on we use capital letters to denote a symbol or one of its left-apexed versions, that is A is either the symbol a or ${}^{p}a$ for some $p \in Q$. Thus, for any rightwards move $(q, b, +) \in \delta(p, a)$ such that $q \neq q_{yes}$, we have the tiles

$$\begin{array}{|c|c|}\hline x & \overrightarrow{a_p} \\ \hline x & b \\ \hline \end{array}\ \begin{array}{|c|c|}\hline X & A \\ \hline x & \overrightarrow{a_p} \\ \hline \end{array}, \tag{3}$$

for every $x \in \Lambda \cup \{\sharp\}$, $x \neq \not b$. Moreover, if the previous move $(q, b, +) \in \delta(p, a)$ can be followed by a rightwards move $(r, d, +) \in \delta(q, c)$ or by a leftwards move $(s, f, -) \in \delta(q, e)$, we have to add also the tiles

$$\begin{array}{|c|c|}\hline \overrightarrow{a_p} & C \\ \hline b & {}^{q}d \\ \hline \end{array}\ \begin{array}{|c|c|}\hline A & C \\ \hline {}^{p}b & {}^{q}d \\ \hline \end{array}\ \text{or}\ \begin{array}{|c|c|}\hline \overrightarrow{a_p} & E \\ \hline b & \overleftarrow{e_q} \\ \hline \end{array}\ \begin{array}{|c|c|}\hline A & E \\ \hline {}^{p}b & \overleftarrow{e_q} \\ \hline \end{array}. \tag{4}$$

For the moves of the form $(q_{yes}, b, +) \in \delta(p, a)$, that enter the machine in the accepting state, we add the tiles

$$\begin{array}{|c|c|}\hline \overrightarrow{p_a} & C \\ \hline b & c_{q_{yes}} \\ \hline \end{array}\ \begin{array}{|c|c|}\hline A & C \\ \hline {}^{p}b & c_{q_{yes}} \\ \hline \end{array}\ \begin{array}{|c|c|}\hline C & X \\ \hline c_{q_{yes}} & x \\ \hline \end{array}\ \begin{array}{|c|c|}\hline b & c_{q_{yes}} \\ \hline \sharp & \sharp \\ \hline \end{array}\ \begin{array}{|c|c|}\hline {}^{p}b & c_{q_{yes}} \\ \hline \sharp & \sharp \\ \hline \end{array}\ \begin{array}{|c|c|}\hline c_{q_{yes}} & x \\ \hline \sharp & \sharp \\ \hline \end{array} \tag{5}$$

for every $x \in \Lambda \cup \{\sharp\}$, $x \neq \not b$.

Analogously, if the directions of all previous moves are reversed, we have to consider the symmetric tiles of (3), (4) and (5), that is

$$\begin{array}{|c|c|}\hline \overleftarrow{a_p} & x \\ \hline b & x \\ \hline \end{array}\ \begin{array}{|c|c|}\hline A & X \\ \hline \overleftarrow{a_p} & x \\ \hline \end{array}\ \begin{array}{|c|c|}\hline C & \overleftarrow{a_p} \\ \hline {}^{q}d & b \\ \hline \end{array}\ \begin{array}{|c|c|}\hline C & A \\ \hline {}^{q}d & {}^{p}b \\ \hline \end{array}\ \ldots$$

where $x \in \Lambda \cup \{\sharp\}$. The last tile above can be distinguished from its symmetric version in (4) by adding a further mark to symbols representing the direction of the moves (a mark we here avoid to use for sake of simplicity).

Finally, we have to add the tiles corresponding to the tape portions far from the head, that is the tiles

$$\begin{array}{|c|c|}\hline X & Y \\ \hline x & y \\ \hline \end{array}\ \begin{array}{|c|c|}\hline A & B \\ \hline \sharp & \sharp \\ \hline \end{array}$$

where $x, y \in \Lambda \cup \{\sharp\}$ and $a, b \in \Lambda \backslash \{\sharp, \not b\}$.

The local picture language generated by the set of all tiles defined above is just L' and this proves that $L \in$ REC.

Moreover, if M is unambiguous then there is only one picture $w' \in L'$ corresponding to each input x accepted by the machine. This means that every picture in L admits a unique covering by the set of tiles defined above. Formally, for every $w \in L$ there is a unique $w' \in L'$ such that $w = \pi(w')$. This shows that L belongs to UREC. □

Example 5.2. Consider the 1t-NTM, the sequence of moves and the associated picture w defined in Example 5.1. When transforming w as described in the proof of Proposition 5.1, we obtain the new picture

$$w' = \begin{array}{|c|c|c|c|c|c|c|c|}\hline \overrightarrow{a_0} & b & a & c & b & c & c & a \\ \hline c & {}^1a & {}^4c & {}^2b & {}^1a & \overleftarrow{c_4} & c & a \\ \hline c & a & c & \overrightarrow{b_1} & {}^5c & b & c & a \\ \hline c & a & c & a & {}^4b & {}^0a & {}^2b & \overleftarrow{a_3} \\ \hline c & a & c & a & b & \overrightarrow{a_4} & {}^2a & b \\ \hline c & a & c & a & b & c & \overleftarrow{a_2} & b \\ \hline c & a & c & a & b_y & {}^0b & b & b \\ \hline \end{array}$$

Note that in the proof of Proposition 5.1, if M works in space $n = |x|$ for an accepted input x, then the corresponding picture in L' has n columns and does not contain $\not b$.

5.2. Overlap of picture languages

We now introduce a partial operation on the set of all picture languages (over arbitrary alphabets), that will be used to fix the size of the accepting-computations of Turing machines studied above. Formally, given two pictures p and q of the same size (n, m), let $p \times q$ be the picture such that $(p \times q)(i, j) = (p(i, j), q(i, j))$ for every $1 \leq i \leq n$ and $1 \leq j \leq m$. Then, the *overlap* $L_1 \diamond L_2$ of two picture languages L_1 and L_2 is defined as

$$L_1 \diamond L_2 = \{p \times q \mid p \in L_1, q \in L_2, \mathrm{r}_p = \mathrm{r}_q, \mathrm{c}_p = \mathrm{c}_q, p(1, j) = q(1, j) \text{ for every } 1 \leq j \leq \mathrm{c}_p\} \qquad (6)$$

Proposition 5.2. Given two picture languages in REC, their overlap is still in REC. Moreover, the overlap of two languages in UREC belongs to UREC.

Proof:
Let L_1 and L_2 be two picture languages over the alphabets Σ_1 and Σ_2, respectively, and assume that they are in REC. Then, for each $i \in \{1, 2\}$, there exists a tiling system $\langle \Sigma_i, \Gamma_i, \Theta_i, \pi_i \rangle$ recognizing L_i. Set

$$\text{Top}(\Theta_i) = \{t \in \Theta_i \mid t = \begin{array}{|c|c|}\hline \sharp & \sharp \\ \hline a & b \\ \hline \end{array} \text{ where } a, b \in \Gamma_i \cup \{\sharp\}\}$$

and let Left(Θ_i), Right(Θ_i), and Bottom(Θ_i) be defined analogously. Also, define Inner(Θ_i) as the set of tiles of Θ_i that do not belong to any of the previous set. Now, let $\Gamma = \Gamma_1 \times \Gamma_2$ and define Θ as the union of the sets Inner(Θ), Left(Θ), Right(Θ), Bottom(Θ), Top(Θ), where:

$$\text{Inner}(\Theta) \;=\; \{ \begin{array}{|c|c|}\hline (a_1,a_2) & (b_1,b_2) \\ \hline (c_1,c_2) & (d_1,d_2) \\ \hline \end{array} \;\Big|\; \begin{array}{|c|c|}\hline a_i & b_i \\ \hline c_i & d_i \\ \hline \end{array} \in \text{Inner } (\Theta_i),\; i \in \{1,2\}\},$$

$$\text{Left}(\Theta) = \{ \begin{array}{|c|c|}\hline \sharp & (a_1,a_2) \\ \hline \sharp & (b_1,b_2) \\ \hline \end{array} \Big| \begin{array}{|c|c|}\hline \sharp & a_i \\ \hline \sharp & b_i \\ \hline \end{array} \in \text{Left}(\Theta_i),\; i \in \{1,2\}\;\},$$

Bottom(Θ) and Right(Θ) are defined similarly, whereas Top(Θ) is given by

$$\text{Top}(\Theta) = \{ \begin{array}{|c|c|}\hline \sharp & \sharp \\ \hline (a_1,a_2) & (b_1,b_2) \\ \hline \end{array} \Big| \begin{array}{|c|c|}\hline \sharp & \sharp \\ \hline a_i & b_i \\ \hline \end{array} \in \text{Top}(\Theta_i),\; i \in \{1,2\} \text{ and}$$

$$\pi_1(a_1) = \pi_2(a_2), \pi_1(b_1) = \pi_2(b_2)\}.$$

Finally, set $\pi = \pi_1 \times \pi_2$, that is, for each pair $(a_1, a_2) \in \Gamma$, set $\pi(a_1, a_2) = (\pi_1(a_1), \pi_2(a_2))$. Clearly, $\tau = \langle \Sigma_1 \times \Sigma_2, \Gamma, \Theta, \pi \rangle$ is a tiling system recognizing the overlap of L_1 and L_2. Moreover, if the two tiling systems are unambiguous then also τ is and this concludes the proof. □

We are now able to prove the second part of Theorem 3.1.

Theorem 5.1. Let L be a unary picture language such that $\phi(L)$ is in NSPACEREV$_Q$. Then L is tiling recognizable. Further, if $\phi(L)$ is in USPACEREV$_Q$ then L belongs to UREC$_1$.

Proof:
Since $\phi(L)$ is in NSPACEREV$_Q$, it is recognized by a 1-tape nondeterministic Turing machine 1t-NTM M that works in $|x|$ space for any input $x \in Q$, and executes at most ${}_{\circ}|x|$ head reversals during each computation. Thus, the accepting-computation language $A(M)$ of such a Turing machine is in REC, by Proposition 5.1, and so is the language $\bar{A}$ obtained from $A(M)$ by replacing the symbol $\circ_{q_0}$ by $\circ$ in the upper-leftmost cell of each picture in $A(M)$. Such a language can be further extended by adding rows of blanks to each picture. Thus we can define the language

$$A' = \bar{A} \ominus \left(\not{b}^{*\ominus}\right)^{*\varobar}$$

which is also in REC by the properties of $\ominus$ and $\varobar$.

Now, let us introduce some auxiliary picture languages we shall use to bind the size of pictures in A'. Let E_s be the set of all unary squares over the alphabet $\{\circ\}$ and set

$$E_h = E_s \varobar h^{*\ominus} \varobar \circ^{**} \qquad \text{and} \qquad E_v = E_s \varobar v^{*\ominus} \varobar \circ^{**}.$$

In other words, every $p \in E_s \cup E_h$ contains $\phi(p)$ on each row, while any $p \in E_v$ contains, on each row, the quasi-unary string $\phi(p^R)$. Moreover, consider the picture languages

$$L_s = A' \diamond E_s, \qquad L_h = A' \diamond E_h \qquad \text{and} \qquad L_v = (A' \diamond E_v)^R.$$

and set $L' = L_s \cup L_h \cup L_v$.

By Proposition 5.2, also L' is tiling recognizable and we now show that $L = \pi(L')$, where π is the obvious projection mapping into $\circ$ all symbols of the alphabet of L. Indeed, by the previous definition, we have that any quasi-unary string x representing a picture of $\pi(L')$ is an accepted input of M; hence $x \in \phi(L)$, that is x represents a picture in L. Since both L and $\pi(L')$ are unary, this implies $\pi(L') \subseteq L$.

On the other hand, assume $p \in L$. First of all, notice that $\phi(p)$ is accepted by M, hence there exists $a \in \bar{A}$ having $x = \phi(p)$ on the first row and such that $\mathrm{c}_a = \max(\mathrm{r}_p, \mathrm{c}_p)$ and $\mathrm{r}_a \leq \min(\mathrm{r}_p, \mathrm{c}_p)$. Let $a' \in A'$ be the extension of a that has exactly $\min(\mathrm{r}_p, \mathrm{c}_p)$ rows, and notice that a' always is a horizontal rectangle or a square (i.e. $r_{a'} \leq c_{a'}$). Moreover, consider the picture

$$u_p = \phi(p)^{\circ |x| \ominus}.$$

Notice that, if p is a horizontal rectangle, then $u_p \in E_h$; if p is a vertical rectangle, then $u_p \in E_v$, otherwise, if p is a square, $u_p \in E_s$. In any case, u_p has the same size as a'. Hence, if p is a square or a horizontal rectangle, then we have $p = \pi(a' \diamond u_p)$; otherwise we have $p = \pi\left((a' \diamond u_p)^R\right)$. In all cases, $p \in \pi(L')$ and hence $L \subseteq \pi(L')$. Thus, $L = \pi(L')$ is in REC_1.

Now assume that $\phi(L)$ belongs to $\mathrm{USPACEREV}_Q$. Then, by Proposition 5.2, we have $L' \in \mathrm{UREC}$ and thus, for every $p \in L$, there is a unique $q \in L'$ such that $\pi(q) = p$. This proves that also L belongs to UREC_1. □

6. Square languages

In this section we focus on unary square languages, that is on unary picture languages whose elements are all squares. It is clear that square languages are nothing but sets of positive integers, so far represented by unary strings over the alphabet $\{\circ\}$. In the following definition we introduce a subclass of $\mathrm{NSPACEREV}_Q$ that concerns only square languages and their representation.

Definition 6.1. $\mathrm{NSPACEREV}_1$ ($\mathrm{USPACEREV}_1$, respectively) is the class of unary string languages that can be recognized by a 1t-NTM (1t-UTM, resp.) working within n space and executing at most n head reversals, for any input of length n.

Integers can also be represented by the classical binary encoding and this suggests to define the binary complexity class corresponding to the previous definition.

Definition 6.2. $\mathrm{NSPACEREV}_2$ ($\mathrm{USPACEREV}_2$, respectively) is the class of binary string languages that can be recognized by a 1t-NTM (1t-UTM, resp.) working within 2^n space and executing at most 2^n head reversals, for any input of length n.

Notice that the families $\mathrm{NSPACEREV}_1$ and $\mathrm{NSPACEREV}_2$ are related to the well-known time complexity classification. In particular, denoting by $\mathrm{NTIME}_1(f(n))$ (resp. $\mathrm{NTIME}_2(f(n))$) the class of unary (resp. binary) string languages recognizable by 1t-NTMs working in $f(n)$ time for any input of length n, we have the following relations:

$$\mathrm{NTIME}_1(n) \subseteq \mathrm{NSPACEREV}_1 \subseteq \mathrm{NTIME}_1(n^2),$$
$$\mathrm{NTIME}_2(2^n) \subseteq \mathrm{NSPACEREV}_2 \subseteq \mathrm{NTIME}_2(4^n). \tag{7}$$

As one might guess, the classes $\mathrm{NSPACEREV}_1$ and $\mathrm{NSPACEREV}_2$ define the same family of integer sets, the only difference being their integer representation. To prove this relationship we first show a speed-up property of the number of head reversals that holds for linear space bounded 1t-NTMs.

Lemma 6.1. For every $c > 1$ and every language L recognized by a 1t-NTM M working in n space within n head reversals, there exists a n-space bounded 1t-NTM M' recognizing L within n/c head reversals.

Proof:
We prove the statement only for $c = 2$ and, without loss of generality, we assume that any head reversal of M only occurs at either the first or the last cell of the work portion of the tape. Thus, a computation of M is a sequence of runs as defined in Section 5.1 where, however, in each run the tape head scans all the work portion from one border cell to the other.

We show how a computation of M for an input of length n can be simulated by a machine M' with $n/2$ head reversals, considering only the case where n and $n/2$ are even (the other cases are treated similarly). Thus, in the first and in the $n/2 + 1$-th run of M the head moves rightwards. We can design M' as a 1t-NTM with the tape split in 4 tracks. First, on track 1 M' simulates M starting from the initial configuration while, on track 2, in the first run it guesses nondeterministically a configuration α of M and keeps it in the subsequent runs. Simultaneously, on track 3 M' simulates M starting just from α and, on track 4, it marks 2 cells at each run. Thus, after $n/2$ head reversals the whole track 4 is marked and then M' accepts if and only if the state entered in track 3 is accepting and configuration α kept in track 2 equals the configuration reached in track 1. Note that the number of accepting computations of M and M' (for the same input) are equal.

The case $c > 2$ (or with odd n or $n/2$) are treated by possibly increasing the number of tracks. □

The correspondence between $\mathrm{NSPACEREV}_1$ and $\mathrm{NSPACEREV}_2$ is formally described by the following proposition. Here, we denote by $Bin(n)$ the binary encoding of a positive integer n.

Proposition 6.1. A unary language L is in $\mathrm{NSPACEREV}_1$ (in $\mathrm{USPACEREV}_1$, respectively) if and only if the set $\{Bin(n) \mid 1^n \in L\}$ is in $\mathrm{NSPACEREV}_2$ (in $\mathrm{USPACEREV}_2$, resp.).

Proof:
We prove the statement only in the first direction (the other is similar). Let M be a 1t-NTM accepting L in n space with n head reversals and define $L' = \{Bin(n) \mid 1^n \in L\}$. Then, a 1t-NTM accepting L' can be designed which works in two phases. First, for an input $x \in \{0, 1\}^+$, the unary string 1^n is computed such that $x = Bin(n)$. Then, the machine simulates M on input 1^n and yields the same output. These phases can be carried out in $2^{|x|}$ space by $2^{|x|+1}$ head reversals. However, the number head reversals can be halved by Lemma 6.1. □

Now, applying the proposition above, Theorem 3.1 can be re-stated using these new classes and we get the following corollary.

Corollary 6.1. Given a unary square language L, the following statements are equivalent:

1. L is in REC_1,
2. $\{\circ^{r_p} \mid p \in L\} \in \mathrm{NSPACEREV}_1$,

3. $\{Bin(\mathrm{r}_p) \mid p \in L\} \in \mathrm{NS}\textsc{pace}\mathrm{R}\textsc{ev}_2$,
Moreover, an analogous statement holds in the unambiguous case. That is, for every unary square language L, the following properties are equivalent:
1a. L is in $\mathrm{UR}\textsc{ec}_1$,
2a. $\{\circ^{\mathrm{r}_p} \mid p \in L\} \in \mathrm{US}\textsc{pace}\mathrm{R}\textsc{ev}_1$,
3a. $\{Bin(\mathrm{r}_p) \mid p \in L\} \in \mathrm{US}\textsc{pace}\mathrm{R}\textsc{ev}_2$.

The previous corollary provides a useful tool to verify whether a unary square language is tiling recognizable. For instance, it is well-known that the set of prime numbers is recognizable in polynomial time[1] and hence by the previous corollary we get the following

Proposition 6.2. The set of unary square pictures whose size is a prime number is in $\mathrm{UR}\textsc{ec}_1$.

More generally, if L_π is a binary NP language, then the picture language

$$\{p \in \circ^{**} \mid \mathrm{r}_p = \mathrm{c}_p \ , \ \exists x \in L_\pi \text{ such that } \mathrm{Bin}(\mathrm{r}_p) = 1x\}$$

belongs to $\mathrm{R}\textsc{ec}_1$.

6.1. A tiling recognizable language from an exponential time complete problem

In this subsection we describe an example of tiling recognizable unary square language that corresponds to a decision problem complete in the class

$$\mathrm{NE}\textsc{xp}\mathrm{T}\textsc{ime} = \bigcup_{c \geq 1} \mathrm{NT}\textsc{ime}_2(2^{cn})$$

and hence not included in NP by well-known separation results [15]. This is the inequality problem of regular expressions with squaring INEQ(RE,2), studied by Meyer and Stockmeyer in the framework of the complexity analysis of word problems requiring exponential time [13, 14]. A rather natural binary encoding of INEQ(RE,2) is solvable in $\mathrm{NT}\textsc{ime}_2(2^n)$ and hence the corresponding family of unary square pictures is tiling recognizable. For sake of completeness, we present the construction in detail even if it can be derived by standard arguments from results and material given in [14].

Let us denote by $\mathrm{RE}(0, 1, \cdot, \cup, ^2)$ the family of regular expressions with squaring over the alphabet $\{0, 1\}$. Formally, this is defined by stating that 0, 1 and ε belong to $\mathrm{RE}(0, 1, \cdot, \cup, ^2)$ and, for every $e_1, e_2 \in \mathrm{RE}(0, 1, \cdot, \cup, ^2)$, also the expressions $(e_1 \cup e_2)$, $(e_1 \cdot e_2)$, $(e_1)^2$ are in $\mathrm{RE}(0, 1, \cdot, \cup, ^2)$. Clearly, every $e \in \mathrm{RE}(0, 1, \cdot, \cup, ^2)$ represents a finite language $L(e) \subseteq \{0, 1\}^*$ defined in the obvious way. Moreover, we say that two expressions $e, f \in \mathrm{RE}(0, 1, \cdot, \cup, ^2)$ are equivalent if $L(e) = L(f)$. The decision problem INEQ(RE,2) consists of determining, on input $e, f \in \mathrm{RE}(0, 1, \cdot, \cup, ^2)$, whether e and f are not equivalent. It is well-known that such a problem is log-space complete for the class $\mathrm{NE}\textsc{xp}\mathrm{T}\textsc{ime}$ [14] and hence it does not lie in NP.

Proposition 6.3. The problem INEQ(RE,2) is solvable by a 1t-NTM in $\mathrm{O}(2^{n(\frac{4}{3}+\varepsilon)})$ time and $\mathrm{O}(2^{n(\frac{1}{3}+\varepsilon)})$ space (for any $\varepsilon > 0$).

Proof (outline). Reasoning as in [14] one can design an algorithm that, on input $e, f \in \mathrm{RE}(0, 1, \cdot, \cup, ^2)$, first determines two nondeterministic finite state automata A_e, A_f recognizing $L(e)$ and $L(f)$, respectively. This can be done by eliminating the squaring from both e and f, which leads to a possible exponential increase of the size of the expressions. Then, the procedure nondeterministically generates a word $x \in \{0,1\}^*$ of length at most $2^{\max\{|e|,|f|\}}$ and deterministically checks whether x is accepted by one of the automata and rejected by the other: if this is the case the algorithm accepts the input, otherwise it rejects. The complexity bounds follow from standard Turing machine constructions; in particular, we recall that every multitape TM working in $\mathrm{O}(t(n))$ time and $\mathrm{O}(s(n))$ space can be simulated by a single tape TM in time $\mathrm{O}(t(n)s(n))$. □

Now, given the alphabet $\Sigma = \{0, 1, \varepsilon, (,), \cdot, \cup, 2, \#\}$, consider a standard encoding $cod : \Sigma \longrightarrow \{0,1\}$ associating each symbol of Σ with a string of at least 2 digits. Then, define the language $I \subseteq \{0,1\}^*$ of all strings y such that $y = cod(e\#f)$ for some $e, f \in \mathrm{RE}(0, 1, \cdot, \cup, ^2)$ and $L(e) \neq L(f)$. Since $|y| \geq 2\max\{|e|, |f|\}$, by Proposition 6.3 we obtain the following

Proposition 6.4. The language I is recognizable by a 1t-NTM in $\mathrm{O}(2^{n(\frac{2}{3}+\varepsilon)})$ time and $\mathrm{O}(2^{n(\frac{1}{6}+\varepsilon)})$ space (for any $\varepsilon > 0$).

As a consequence I belongs to $\mathrm{NT{\small IME}}_2(2^n)$ and setting

$$T = \{p \in \circ^{**} \mid r_p = c_p = 1x \text{ for a string } x \in I\}$$

we have that T is in $\mathrm{R{\small EC}}_1$.

6.2. Nonrecognizable picture languages

Another consequence of Corollary 6.1 concerns the construction of unary square languages that are not tiling recognizable. For instance one can prove the existence of a unary square language that is not tiling recognizable, but such that the set of binary encoding of its sizes is not too far (from a complexity view point) from the class $\mathrm{NS{\small PACE}R{\small EV}}_2$. In order to present such an example, for any function $f : \mathbb{N} \to \mathbb{R}^+$, let us define 2t-$\mathrm{NT{\small IME}}_2(f)$ as the class of binary string languages that are recognizable by 2-tape nondeterministic Turing machines working within time $f(n)$ on every input of length n.

Proposition 6.5. There exists a unary square picture language $L \notin \mathrm{R{\small EC}}_1$ such that the string language $S = \{x \in \{0,1\}^* \mid 1x = Bin(\mathrm{r}_p) \text{ for a picture } p \in L\}$ belongs to 2t-$\mathrm{NT{\small IME}}_2(4^n \log n)$.

Proof:
The existence of such language is guaranteed by a result proved in [15]. If $T_1, T_2 : \mathbb{N} \to \mathbb{R}^+$ are two running-time functions such that $T_1(n+1)/T_2(n)$ tends to 0 as n goes to infinity, then there exists a language $S \subseteq \{0,1\}^*$ that belongs to 2t-$\mathrm{NT{\small IME}}_2(T_2(n))$ but does not belong to 2t-$\mathrm{NT{\small IME}}_2(T_1(n))$. Setting $T_1(n) = 4^n$, $T_2(n) = 4^n \log n$, and observing that 2t-$\mathrm{NT{\small IME}}_2(4^n) \supseteq \mathrm{NS{\small PACE}R{\small EV}}_2$, by Theorem 3.1 we have that S is in 2t-$\mathrm{NT{\small IME}}_2(4^n \log n)$ whereas L cannot be tiling recognizable. □

7. Separation results

In the previous section we have shown how properties of complexity classes can be used to determine examples and results concerning the recognizability of picture languages. Here we use our main result in the opposite direction, showing properties of the class NSPACEREV$_Q$ obtained by studying the corresponding family REC$_1$.

A first result concerns the separation between the classes NSPACEREV$_Q$ and USPACEREV$_Q$.

Theorem 7.1. The class USPACEREV$_Q$ is strictly included in NSPACEREV$_Q$.

This property is an immediate consequence of a separation result obtained in [4], where it is proved that there exists a picture language L in REC$_1$ that does not lie in UREC$_1$. Indeed, by Theorem 3.1, this implies that the encoding $\phi(L)$ belongs to NSPACEREV$_Q$\USPACEREV$_Q$.

A further natural subclass of NSPACEREV$_Q$ is defined by considering deterministic Turing machine. In accordance with our previous notation, let 1t-DTM stand for one-tape deterministic Turing machine. Then we denote by DSPACEREV$_Q$ the class of quasi-unary string languages accepted by a 1t-DTM working within $|x|$ space and executing at most $_\circ|x|$ head reversals, for every input x in Q. Clearly such a class is included in USPACEREV$_Q$ and our main purpose in this section is to prove that such inclusion is strict. The result is mainly based on the following property.

Lemma 7.1. Let $L \subseteq Q_h$ be a language in DSPACEREV$_Q$ and let M be a 1t-DTM with s states recognizing L within $|x|$ space and executing at most $_\circ|x|$ head reversals, for every input x in Q. Also assume there exists a word $z \in L$ such that $|z| > m + 1 + 3s^{m+1}$, where $m = {}_\circ|z|$. Then there exists $c \in \mathbb{N}$ such that $z\circ^c$ is in L and all prime divisors of c are smaller or equal to s.

Proof:
As in Lemma 6.1, let M be allowed to reverse the tape head only at the border cells of the work portion of the tape. We also assume that, for any input $x \in Q_h$, M always executes $_\circ|x|$ head inversions and hence $1 + {}_\circ|x|$ runs. This can be done by marking the input symbols that preceed h once at each run and stopping the computation when all the first $_\circ|x|$ symbols are marked.

Now, let $z \in L$ be a string of length $n > m + 1 + 3s^{m+1}$, where $m = {}_\circ|z|$ and let us denote by C the computation of M on input z. For every $j \in \{1, 2, \ldots, n\}$, the *crossing sequence* at position j is an array of states $\underline{p} = (p_0, p_1, \ldots, p_m)$, where each p_i is the state of M while the machine is scanning the j-th cell in the $i + 1$-th run of the computation. Let us first prove the following easy property.

> **Claim.** Assume that, in the computation C on z, the crossing sequences at positions i and j are equal, for two integers i and j such that $m + 1 < i < j \leq n$. Then, the string $z\circ^c$ belongs to L, where $c = j - i$.

Indeed, let C' be the computation of M on input $z\circ^c$. Since the machine is deterministic and the input symbols placed after the position $m + 1$ are all $\circ$, in computation C' between positions j and $n + c$ the machine makes the same moves as in the computation C between the positions i and n. This means that, for every $\ell = j, j + 1, \ldots, n + c$, the crossing sequence of C' at position ℓ equals the crossing sequence of C at position $\ell - c$. Moreover, it is clear that C and C' have the same crossing sequences at positions $\ell \leq i$. As a consequence, the states reached by C and C' at the end of the last run must be equal, and this proves the claim.

Now, let us construct a pair of positions i, j satisfying the hypothesis of the previous claim and let us evaluate their distance. Consider the first two positions i_0, j_0, such that $m+1 < i_0 < j_0$, where M enters the same state (say p_0) during the first run of C. Set $\alpha_0 = j_0 - i_0$ and note that $\alpha_0 \leq s$ (and $j_0 \leq m+s+2$). Consider also the arithmetic progression of positions given by

$$T_0 = \{i_0 + \ell\alpha_0 \mid \ell \in \mathbb{N}, i_0 + \ell\alpha_0 \leq n\}$$

Since, in the first run of C the moves between two consecutive positions in T_0 are the same, M enters state p_0 at each position $j \in T_0$; for the same reason, it prints the same word between any two consecutive positions of T_0. Thus, in the second run, while it moves from right to left, between any two consecutive positions of T_0 the machine reads the same word. In such run, as T_0 has more than s elements, we can consider the first pair of positions $i_1, j_1 \in T_0$, $n - (s\alpha_0) \leq i_1 < j_1 \leq n$, where M enters the same state, say p_1. The difference $c_1 = j_1 - i_1$ satisfies the equality $c_1 = \alpha_0\alpha_1$, for some positive integer $\alpha_1 \leq s$. Again, we can consider the arithmetic progression

$$T_1 = \{j_1 - \ell c_1 \mid \ell \in \mathbb{N}, m+1 < j_1 - \ell c_1\}$$

Thus, during the second run of C, at each position $j \in T_1$ the machine enters the same state p_1; moreover, between any two consecutive positions in T_1 it prints the same word, which will be read backwards in the third run.

The previous reasoning can be repeated for all subsequent runs of C, as the input is long enough. In the last run, we obtain a crossing sequence $\underline{p} = (p_0, p_1, \dots, p_m)$ that occurs in the computation C at all positions belonging to an arithmetic progression T_m defined by

$$T_m = \{i_m + \ell c_m \mid \ell \in \mathbb{N}, i_m + \ell c_m \leq n\} \ ,$$

where i_m is an integer such that $m+1 < i_m \leq m+1+c_m$, $c_m = \alpha_0\alpha_1 \cdots \alpha_m$, with $\alpha_i \in \mathbb{N}$ and $1 \leq \alpha_i \leq s$ for every i. Note that $c_m \leq s^{m+1}$. Thus, the length $n > m+1+3s^{m+1}$ guarantees that T_m contains at least two positions at a distance c_m and hence the lemma follows from the claim above with $c = c_m$. □

Theorem 7.2. The language

$$L = \{\circ^m h \circ^{n-1-m} \mid n = km, k, m \in \mathbb{N}\}$$

belongs to USPACEREV$_Q$\DSPACEREV$_Q$.

Proof:

It is easy to show that L is in USPACEREV$_Q$. Indeed, a 1t-UTM can be designed which, for an input $x \in Q_h$, in the first run nondeterministically marks some symbols of x on the right of h; then, it checks in a deterministic way, by using at most $m = {}_\circ|x|$ head reversals, whether the distance between any two consecutive marks equals m. Note that there is at most one accepting computation for any input.

Now, assume by contradiction that $L \in$ DSPACEREV$_Q$. Then, there is a 1t-DTM M that recognizes L within $|x|$ space and executing at most ${}_\circ|x|$ head reversals, for every input x in Q. Let s be the number of states of M and let m be a prime integer greater than s. Also consider a word $z = \circ^m h \circ^{mk-1}$, where k is an integer such that the length of z satisfies $(k+1)m > m+1+3s^{m+1}$. Since $z \in L$, by Lemma 7.1, there is $c \in \mathbb{N}$ with all prime divisors at most equal to s, such that the string $y = \circ^m h \circ^{mk+c-1}$ is in L. This implies $|y| = m(k+1)+c$ is a multiple of m and hence $m > s$ is a divisor of c: however this is a contradiction since, as stated above, every prime divisor of c must be smaller or equal to s. □

8. Conclusions

In this work we have characterized the families REC_1 and UREC_1 of unary two-dimensional languages that are tiling recognizable and unambiguously tiling recognizable, respectively. We have shown that any unary picture language in REC_1 or in UREC_1 can be represented by a single-tape, linearly space-bounded, nondeterministic or unambiguous Turing machine, with a further input dependent constraint on the number of head reversals. In this construction, each picture in the local language corresponds to an accepting computation of the Turing maching, and viceversa. Exploiting such a correspondence, we use a deep result on picture languages [4, 11, 12] to obtain a separation property between complexity classes. This approach could be also applied to other natural subclasses of NSPACEREV_Q.

Another open problem concerns the comparison between the class DSPACEREV_Q introduced in the present work and the class DREC_1 of deterministic unary picture languages studied in [4].

Concluding, we observe that another natural problem concerns the analysis of classes of languages defined by Turing machines with a bounded number of head reversals. More precisely, one may ask whether a separation property, similar to that one proved in [15] for multitape time-bounded nondeterministic Turing machine, also holds for complexity classes defined by bounding the number of head reversals. This would lead to simpler examples of unary picture languages that are not tiling recognizable.

References

[1] M. Agrawal, N. Kayal, N. Saxena. PRIMES is in P. *Annals of Mathematics*, 160(2): 781-793, 2004.

[2] M. Anselmo, D. Giammarresi, M. Madonia. Regular expressions for two-dimensional languages over one-letter alphabet. In *Proc. 8th DLT*, C.S. Calude, E. Calude and M.J. Dinneen (Eds.), LNCS 3340, 63–75, Springer, 2004.

[3] M. Anselmo, D. Giammarresi, M. Madonia, A. Restivo. Unambiguous recognizable two-dimensional languages. *Theoretical Informatics and Applications* 40(2): 277–293, 2006.

[4] M. Anselmo, M. Madonia. Deterministic and unambiguous two-dimensional languages over one-letter alphabet. To appear in *Theoretical Computer Science*. Preliminary version in *Proc. CAI 2007*, S. Bozapalidis and G. Rahonis (Eds.), LNCS 4728, 147–159, Springer, 2007.

[5] D. Giammarresi, A. Restivo. Recognizable picture languages. *Int. J. Pattern Recognition and Artificial Intelligence* 6: 31–42, 1992.

[6] D. Giammarresi, A. Restivo. Two-dimensional languages. In *Handbook of Formal Languages*, G. Rosenberg and A. Salomaa (Eds.), Vol. III, 215 – 268, Springer-Verlag, 1997.

[7] D. Giammarresi, A. Restivo, S. Seibert, W. Thomas. Monadic second order logic over rectangular pictures and recognizability by tiling system. *Information and Computation*, 125(1):32–45, 1996.

[8] K. Inoue, I. Takanami. A survey of two-dimensional automata theory. In *Proc. 5th Int. Meeting of Young Computer Scientists*, J. Dasson, J. Kelemen (Eds.), LNCS 381, 72–91, Springer-Verlag, 1990.

[9] J. Kari, C. Moore. New results on alternating and non-deterministic two-dimensional finite state automata. In *Proc. 18th STACS*, A. Ferreira, H. Reichel (Eds.), LNCS 2010, 396–406, Springer-Verlag, 2001.

[10] O. Matz. Regular expressions and context-free grammars for picture languages. In *Proc. 14th STACS*, LNCS 1200, 283–294, Springer-Verlag, 1997.

[11] O. Matz. Dot-depth and monadic quantifier alternation over pictures. *Ph.D. thesis*, Technical Report 99-08, RWTH AAchen, 1999.

[12] O. Matz. Dot-depth, monadic quantifier alternation, and first-order closure over grids and pictures. *Theoretical Computer Science*, 270(1-2): 1–70, Elsevier 2002.

[13] A.R. Meyer and L.J. Stockmeyer. The equivalence problem for regular expressions with squaring requires exponential space. In *Proc. 13th Annual IEEE Symp. on Switching and Automata Theory*, 125–129, 1972.

[14] A.R. Meyer and L.J. Stockmeyer. Words problems requiring exponential time. In *Proc. 5th ACM Symp. on Theory of Computing*, 1–9, 1973.

[15] J. I. Seiferas, M. J. Fischer, A. R. Meyer. Separating nondeterministic time complexity classes. *Journal of ACM*, 25(1): 146–167, 1978.

[16] R. Siromoney. Advances in array languages. In *Graph-grammars and their applications to Computer Science*, Ehrig et al. Eds., LNCS 291, 549–563, Springer-Verlag, 1987.

[17] K. Wagner, G. Wechsung. *Computational complexity.* D. Reidel Publishing Company, 1986.