Two Dimensional Picture Languages: Tiling Systems, Domino Systems and Weighted Finite Automata

Benedikt Lüken-Winkels

September 2020

Introduction

This report provides an overview and a summary of the works of Gimmarresi and Restivo [1] and Culik and Kari [2] on two-dimensional languages, especially focussing on Tiling Systems, Domino Systems and Weighted Finite Automata.

1 Tiling Systems

The basic idea of a tiling system is to use the ability of finite automata to recognize a string language by projecting a local language in the two-dimensional world. Therefore the local language can be obtained from a finite set Θ of square *tiles* of size 2×2. Then a two-dimensional language is 'tiling recognizable', if it can be projected from the local picture language.

A tiling system $\mathcal{T}$ is defined as a 4-tuple $\mathcal{T} = \{\Sigma, \Gamma, \Theta, \pi\}$. Let Σ and Γ be finite alphabets. A language $L \subseteq \Sigma^{**}$ is tiling recognizable, if there is a local language $L' \subseteq \Gamma^{**}$ that can be obtained exactly from tiles taken from the finite set of tiles Θ over the alphabet $\Gamma \cup \{\#\}$ and a projection $\pi : \Theta \to \Gamma$. When L' can be obtained from Θ it can be written as $L' = L(\Theta)$. A computational approach on verifying the locality of L' is to scan a picture from L' with a 2×2 window and checking, if every tile is included in Θ.

The recognition of a Language L by a tiling system $\mathcal{T}$ can be written as $L = L(\mathcal{T})$, when L is a projection of some local language. and $L \in \mathcal{L}(TS)$ (L lies in the family of two-dimensional languages recognizable by tiling systems).

Example 1 Let $\Sigma = \{a\}$ be a one-letter alphabet and let L be the language of all pictures over Σ with 3 rows.
Claim: Language L is tiling recognizable.

$$Let\ \Theta = \left\{ \begin{array}{c} \boxed{\begin{matrix} \# & \# \\ \# & 1 \end{matrix}}, \boxed{\begin{matrix} \# & \# \\ 1 & 1 \end{matrix}}, \boxed{\begin{matrix} \# & \# \\ 1 & \# \end{matrix}} \\ \boxed{\begin{matrix} \# & 1 \\ \# & 2 \end{matrix}}, \boxed{\begin{matrix} 1 & 1 \\ 2 & 2 \end{matrix}}, \boxed{\begin{matrix} 1 & \# \\ 2 & \# \end{matrix}} \\ \boxed{\begin{matrix} \# & 2 \\ \# & 3 \end{matrix}}, \boxed{\begin{matrix} 2 & 2 \\ 3 & 3 \end{matrix}}, \boxed{\begin{matrix} 2 & \# \\ 3 & \# \end{matrix}} \\ \boxed{\begin{matrix} \# & 3 \\ \# & \# \end{matrix}}, \boxed{\begin{matrix} 3 & 3 \\ \# & \# \end{matrix}}, \boxed{\begin{matrix} 3 & \# \\ \# & \# \end{matrix}} \end{array} \right\}$$

Then a picture $p \in L(\Theta)$ can look like this:

$$\boxed{\begin{matrix} \# & \# & \# & \# & \# & \# & \# \\ \# & 1 & 1 & 1 & 1 & 1 & \# \\ \# & 2 & 2 & 2 & 2 & 2 & \# \\ \# & 3 & 3 & 3 & 3 & 3 & \# \\ \# & \# & \# & \# & \# & \# & \# \end{matrix}}$$

Using the previously explained method a 2×2 window traversing p will always contain a tile from Θ. Now with π being defined as $\pi(1) = \pi(2) = \pi(3) = a$, one can see, that L is *tiling recognizable*, so $L \in \mathcal{L}(TS)$. $\square$

This approach works for languages with any number of rows, with a corresponding size of Γ, since for each row there has to be a dedicated symbol in the alphabet to keep track of the number of rows in each picture from $L' = L(\Theta)$.

1.1 Closure Properties

Projection Let Σ_1 and Σ_2 be finite alphabets and $\rho : \Sigma_1 \to \Sigma_2$ a projection. If $L_1 \subseteq \Sigma_1^{**}$ is tiling recognizable, then $L_2 = \rho(L_1)$ $(L_2 \subseteq \Sigma_2^{**})$ is too. Let $\mathcal{T}_1 = (\Sigma_1, \Gamma, \Theta, \pi_1)$ be recognizing tiling system of L_1 and $\mathcal{T}_2 = (\Sigma_2, \Gamma, \Theta, \pi_2)$. Now if π_2 is defined as $\pi_2 = \rho \circ \pi_1 : \Gamma \to \Sigma_2$, one can see, that L_2 is recognized by $\mathcal{T}_2$. Therefore $L_1, L_2 \in \mathcal{L}(TS)$.

$\Rightarrow \mathcal{L}(TS)$ is closed under projection.

Row and column concatenation Let L_1 and L_2 be picture languages over an alphabet Σ and let $L = L_1 \ominus L_2$ be the language corresponding to the row concatenation of L_1 and L_2. Furthermore let $\mathcal{T}_1 = (\Sigma, \Gamma_1, \Theta_1, \pi_1)$ and $\mathcal{T}_2 = (\Sigma, \Gamma_2, \Theta_2, \pi_2)$ be tiling systems for L_1 and L_2. In a tiling system $\mathcal{T} = (\Sigma, \Gamma, \Theta, \pi)$ for L with $\Gamma = \Gamma_1 \cup \Gamma_2$, Θ has to contain

each tile from Θ_1 without its bottom borders

$$\Theta_1' = \left\{ \boxed{\begin{matrix} a_1 & b_1 \\ c_1 & d_1 \end{matrix}} \;\middle|\; \boxed{\begin{matrix} a_1 & b_1 \\ c_1 & d_1 \end{matrix}} \in \Theta_1 \ and\ c_1, d_1 \neq \# \right\},$$

each tile from Θ_2 without its upper borders

$$\Theta_2' = \left\{ \boxed{\begin{array}{cc} a_1 & b_1 \\ c_1 & d_1 \end{array}} \mid \boxed{\begin{array}{cc} a_1 & b_1 \\ c_1 & d_1 \end{array}} \in \Theta_2 \textit{ and } a_1, b_1 \neq \# \right\} \text{ and}$$

connecting (gluing) tiles

$$\Theta_{12} = \left\{ \boxed{\begin{array}{cc} \# & a_1 \\ \# & a_2 \end{array}}, \boxed{\begin{array}{cc} b_1 & \# \\ b_2 & \# \end{array}}, \boxed{\begin{array}{cc} c_1 & d_1 \\ c_2 & d_2 \end{array}} \mid \begin{array}{l} \boxed{\begin{array}{cc} \# & a_1 \\ \# & \# \end{array}}, \boxed{\begin{array}{cc} b_1 & \# \\ \# & \# \end{array}}, \boxed{\begin{array}{cc} c_1 & d_1 \\ \# & \# \end{array}} \in \Theta_1 \\ \boxed{\begin{array}{cc} \# & \# \\ \# & a_2 \end{array}}, \boxed{\begin{array}{cc} \# & \# \\ b_2 & \# \end{array}}, \boxed{\begin{array}{cc} \# & \# \\ c_2 & d_2 \end{array}} \in \Theta_2 \end{array} \right\}.$$

Then $\Theta = \Theta_1 \cup \Theta_2 \cup \Theta_{12}$ and $\pi : \Gamma \to \Sigma$ maps the tile's symbols corresponding to their 'origins':

$$\forall a \in \Gamma, \qquad \pi(a) = \begin{cases} d & \pi_1(a) \qquad \text{if } a \in \Gamma_1 \\ & \pi_2(a) \quad \text{if } a \in \Gamma_2 \backslash \{\Gamma_1 \cap \Gamma_2\} \end{cases}$$

For the *column concatenation* $L = L_1 \oplus L_2$ there is a similar approach, but in this case with Θ_1 including all but the right and Θ_2 all but the left bordered tiles. The 'gluing' set of tiles Θ_{12} corresponds to the columns where the pictures are concatenated.

$\Rightarrow \mathcal{L}(TS)$ is closed row and column concatenation.

Row and column closure For a tiling system $(\Sigma, \Gamma, \Theta, \pi)$ for $L^{*\ominus}$, two different tiling systems can be used to build Γ as the union of sets of tiles without upper and lower borders and the set of gluing tiles, as shown above. With the same technique a tiling system for $L^{*\oplus}$ can be defined.

$\Rightarrow \mathcal{L}(TS)$ is closed row and column closure operations.

Union and intersection Let L_1 and L_2 be picture languages over an alphabet Σ and let $L = L_1 \cup L_2$ be the language corresponding to the union of L_1 and L_2. Furthermore let $(\Sigma, \Gamma_1, \Theta_1, \pi_1)$ and $(\Sigma, \Gamma_2, \Theta_2, \pi_2)$ be tiling systems for L_1 and L_2, respectively. A tiling system $(\Sigma, \Gamma, \Theta, \pi)$ for L has $\Gamma = \Gamma_1 \cup \Gamma_2$ with $\Theta = \Theta_1 \cup \Theta_2$. The projection π is defined corresponding to π_1 and π_2 as above.

$(\Sigma, \Gamma, \Theta, \pi)$ for the language $L = L_1 \cap L_2$ uses as local alphabet Γ a subset of $\Gamma_1 \times \Gamma_2$ to identify tiles that occur in both Θ_1 and Θ_2. Two symbols that map to the same symbol in Σ create a pair, that belongs to Γ:

$$(a_1, a_2) \in \Gamma \Leftrightarrow \pi_1(a_1) = \pi_2(a_2)$$

Now Θ consists of tiles, where these pairs of symbols correspond to their origins:

$$\Theta = \left\{ \boxed{\begin{array}{cc} (a_1, a_2) & (b_1, b_2) \\ (c_1, c_2) & (d_1, d_2) \end{array}} \mid \boxed{\begin{array}{cc} a_1 & b_1 \\ c_1 & d_1 \end{array}} \in \Theta_1 \textit{ and } \boxed{\begin{array}{cc} a_2 & b_2 \\ c_2 & d_2 \end{array}} \in \Theta_2 \right\},$$

$\Rightarrow \mathcal{L}(TS)$ is closed under union and intersection.

Complement Let $\Sigma = \{a, b\}$ and let $L = \{p \in \Sigma^{**} | p = s \ominus s$ wheres is a square$\}$ be the language of rectangles with identical upper and lower halves. If $L \in \mathcal{L}(TS)$, then there is a local L' over an alphabet Γ from which L can be obtained by projection. In general $|\Sigma| \leq |\Gamma|$, as seen in previous examples. Now, let L_n be the language of rectangles with identical squares as upper and lower halves of size n and L'_n the local language over Γ that can be projected to L_n by π. There are $|\Sigma|^{n^2}$ possible pictures of L_n and $|\Gamma|^{2n}$ possible n-th and $(n+1)$-th rows in pictures of L'_n. So for a large n there will be two pictures $p' = s'_p \ominus s''_p$ and $q' = s'_q \ominus s''_q$ in L'_n with identical n-th and $(n+1)$-th rows and projections $p = s_p \ominus s_p$ and $q = s_q \ominus s_q$ in L_n with $s_p \neq s_q$. Since the n-th and $(n+1)$-th rows are identical Θ contains tiles that allow to obtain the picture $v' = s'_p \ominus s''_q$ with $\pi(v') = s_p \ominus s_q$, creating a picture not in L_n.

The complement of L ${}^cL$ can be written as ${}^cL = L_1 \cup L_2$, the union of the language with rectangles of size $\neq (2n, n)$ and the language of rectangles of size $(2n, n)$ with different top and bottom halves. L_1 is recognizable by a tiling system, that creates a rectangle using descending stairs, two by one, starting in the top left corner and then missing the last square on the bottom right. L_2 can be decomposed in several languages that are recognized by tiling systems using the same technique as for L_1.
Now one can see, that $L \notin \mathcal{L}(TS)$, while ${}^cL \in \mathcal{L}(TS)$.

$\Rightarrow \mathcal{L}(TS)$ is not closed under complement.

Rotation To obtain a tiling system for the rotation L^R of a language L with tiling system $(\Sigma, \Gamma, \Theta, \pi)$, the tiles in Θ are rotated. $(\Sigma, \Gamma, \Theta^R, \pi)$ recognizes L^R.

$\Rightarrow \mathcal{L}(TS)$ is closed under rotation.

2 Domino Systems

In domino systems, the definition of the local language is altered and called hv-local language. Instead of using (2×2)-tiles, horizontal and vertical dominoes of sizes $(1, 2)$ and $(2, 1)$ define the new set of tiles. A two-dimensional language $L \subseteq \Gamma^{**}$ is hv-local, if there exists a finite set of dominoes Δ over $\Gamma \cup \{\#\}$ that exactly 'produces' L. A domino system is a 4-tuple $\mathcal{D} = (\Sigma, \Gamma, \Delta, \pi)$ where Σ and Γ are finite alphabets, Δ is the set of dominoes over $\Gamma \cup \{\#\}$ and $\pi : \Gamma \to \Sigma$.

Proposition L is an hv-local language, then L is a local language, too.

Let $L = L(\Delta)$ and Θ be a finite set of tiles, that consist of dominoes from Δ. To show, that also $L = L(\Theta)$, let $p \in L(\Theta)$ and $q \in L(\Delta)$. As defined, in p any (1×2) sub-block $B_{1,2}(\widehat{p})$ and (2×1) sub-block in $B_{1,2}(\widehat{p})$ is included in Δ. So $p \in L(\Theta)$.

Any (2×2) sub-block $B_{2,2}(\widehat{q})$ of q is a tile from Θ. Since those tiles are made of dominoes from Δ, $q \in L(\Delta)$. $\square$
There are local languages that are not hv-local, so the converse of this proposition does not hold.

Theorem $\mathcal{L}(TS) = \mathcal{L}(DS)$

3 Weighted Finite Automata

A WFA describes a grey-scale picture by focussing on the relation between parts and subparts of the image. Therefore the image is recursively partitioned into four quadrants which can be addressed by a quadtree.
Let $\Sigma = \{(0,0),(0,1),(1,0),(1,1)\}$ be the alphabet for words $w \in \Sigma^*$ which form a path through the quadtree to either find a *node*, i.e. a subsquare in the image, or a *leaf*, i.e. the smallest unit in the image, a *pixel* for images with finite resolution. A finite finite-resolution image usually consists of $(2^n \times 2^n)$ pixels whith a fixed n, while a multi-resolution image is a collection of several images, with $n = 0, 1, \dots$. Then the addressing scheme works as seen in figure 1, starting from the bottom left corner with the entire image having the address ε.

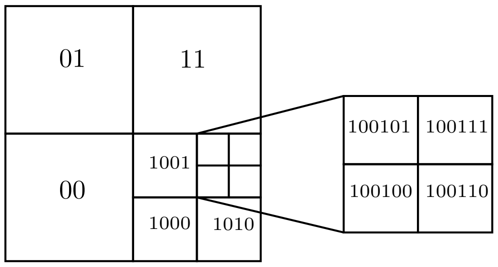

Figure 1: Example for addressing subsquares in images for WFA

Let $f : \Sigma^* \to \mathbb{R}$ be the greyness function that assigns a greyness value to a subsquare. To describe an rgb color picture, three images are necessary.
A function $f : \Sigma^* \to \mathbb{R}$ is average preserving, if

$$f(w) = \frac{1}{4}[f(w(0,0)) + f(w(0,1)) + f(w(1,0)) + f(w(1,1))]$$

for each $w \in \Sigma^*$. This means, that the average greyness of the subsquare with the address w of a picture p is equal to the average greyness of its subsquares.
A weighted finite automaton is a 5-tuple $\mathcal{A} = (Q, \Sigma, f, \alpha, \beta)$, with

- Q the finite set of states,
- Σ the finite alphabet to form the addresses of the subsquares ($\Sigma = \{(0,0),(0,1),(1,0),(1,1)\}$),
- $f : Q \times \Sigma \times Q \to [-\infty, \infty]$ the function assigning weights to the transitions in the automaton,
- $\alpha : Q \to [-\infty, \infty]$ the initial distribution and

- $\beta \to [-\infty, \infty]$ the final distribution on the greyness scale.

If $f(p,a,q) \neq 0$, (p,a,q) is a transition. Now the distribution function $\delta : Q \times \Sigma^* \to [-\infty, \infty]$ is defined as

$$\begin{aligned} \delta(q,\varepsilon) &= \alpha(q),\ \forall q \in Q \\ \delta(q,wa) &= \sum_{p \in Q} \delta(p,w) \cdot f(p,a,q),\ \forall p \in Q, w \in \Sigma^* \text{ and } a \in \Sigma \end{aligned}$$

With this, the WFA $\mathcal{A}$ defines the function $\phi_{\mathcal{A}} : \Sigma^* \to [-\infty, \infty]$, which returns the greyness value of a subsquare with a given word (address) $w \in \Sigma^*$, by summing up the initial distribution of the starting state, the final distribution of the final state and the weights of the transitions for each path in the automaton that forms the input word w:

$$\phi_{\mathcal{A}}(w) = \sum_{q_0,\ldots,q_n \in Q} \alpha(q_0) \cdot f(q_0,a_1,q_1) \cdot \cdots \cdot f(q_{n-1},a_n,q_n) \cdot \beta(q_n)$$

In other words, the sum of the weights on all paths leading from state p to q corresponds to the effect, the state q has on the image.

A WFA $\mathcal{A}$ is average preserving, iff

$$\forall p \in Q : \sum_{a \in \Sigma, q \in Q} (f(p,a,q) \cdot \beta(q)) = 4 \cdot \beta(p)$$

so the sum of the greyness values (distribution) after all possible state transitions from state p is equal to four times its final distribution value $\beta(p)$. Then, if $\mathcal{A}$ is average perserving, $\phi_{\mathcal{A}}$ is average preserving.

3.1 Encoding an image with a WFA

The encoding of an image with a WFA means being able to reconstruct the image up to a certain resolution. A 2^k by 2^k resolution would create a quadtree of height k, where all values are on one level. Then, propagating towards the root each node is assigned the average value of its children.

The encoding algorithm generates a WFA $\mathcal{A}$ and $\phi_{\mathcal{A}}$ for a given distribution function ϕ, such that $\phi_{\mathcal{A}} = \phi$. The goal is to find an automaton with as few states as possible. There are several ways to improve the algorithm, when used in practise. By introducing an error margin e, the amount of transitions can be reduced. If the transition weight lies below a certain value it is set to 0 and is no longer taken into account. The result is, that the encoded image is no longer equal to image to be encoded, but is still similar (depending on the size of e). In addition the size of the automaton decreases.

Algorithm 1 shows the encoding procedure of an image given by ϕ to a WFA $\mathcal{A}$ and the corresponding $\phi_{\mathcal{A}}$.

Algorithm 1: Construct WFA $\mathcal{A}$ and $\phi_{\mathcal{A}}$ for image given by ϕ

```
input: Distribution function $\phi$ for the given image
$N \leftarrow 0$
$i \leftarrow 0$
$\beta(q_0) \leftarrow \phi(\varepsilon)$
$\gamma(q_0) \leftarrow \varepsilon$
do
    $w \leftarrow \gamma(q_i)$
    foreach $a \in \{(0,0),(0,1),(1,0),(1,1)\}$ do
        Find $c := c_0, \dots, c_N$ such that $\phi_{wa} = c_0\phi_0 + \cdots + c_N\phi_N$ (for
         each $a$), where $\phi_j$ corresponds to the subsquare of the $j$-th
         state $q_j$ ($\phi_{\gamma(q_j)}$)
        if $c$ exists then
            for $j = 0, \dots, N$ do
                $f(q_i, a, q_j) \leftarrow c_j$
            end
        else
            $\gamma(q_{N+1}) \leftarrow wa$
            $\beta(q_{N+1}) \leftarrow \phi(wa)$
            $f(q_i, a, q_{N+1}) \leftarrow 1$
            $N \leftarrow N + 1$
        end
    end
    $i \leftarrow i + 1$
while $i \leq N$;
$\alpha(q_0) \leftarrow 1$
for $j = 1, \dots, N$ do
    $\alpha(q_j) \leftarrow 0$
end
```

Example 2 Figure 2 shows the automaton, that can be used to generate the image shown in figure 3. Using the WFA-encoding algorithm (algorithm 1) this automaton is created in as follows:

1. q_0 is assigned to the entire square ε and $\beta(q_0) := 0.5$.

2. Now with $w = \varepsilon$, iterate over the subsquares $\{(0,0),(0,1),(1,0),(1,1)\}$ and try to find $c = c_0, \dots, c_N$.

   (a) $\varepsilon 00$: For $\phi_{\varepsilon 00} = c_0\phi_0$ c_0 gets the value 1.25 $\Rightarrow f(q_0, 00, q_0) := 1.25$

   (b) $\varepsilon 01$: For $\phi_{\varepsilon 01} = c_0\phi_0$ c_0 gets the value 1 $\Rightarrow f(q_0, 01, q_0) := 1$

   (c) $\varepsilon 10$: For $\phi_{\varepsilon 10} = c_0\phi_0$ c_0 gets the value 1 $\Rightarrow f(q_0, 10, q_0) := 1$

   (d) $\varepsilon 11$: For $\phi_{\varepsilon 11} = c_0\phi_0$ c_0 gets the value 0.5 $\Rightarrow f(q_0, 11, q_0) := 0.5$

3. $\alpha(q_0) := 1$

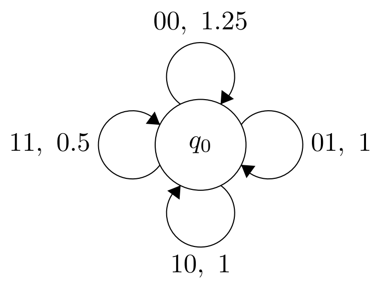

Figure 2: Automaton to generate figure 3

$\frac{1}{4}$	$\frac{1}{2}$
$\frac{3}{4}$	$\frac{1}{2}$

Figure 3: Example image with greyscale

References

[1] Dora Giammarresi and Antonio Restivo. Two-dimensional languages. In Grzegorz Rozenberg and Arto Salomaa, editors, *Handbook of Formal Languages, Volume 3: Beyond Words*, pages 215–267. Springer, 1997.

[2] Karel Culik II and Jarkko Kari. Image compression using weighted finite automata. In Andrzej M. Borzyszkowski and Stefan Sokolowski, editors, *Mathematical Foundations of Computer Science 1993, 18th International Symposium, MFCS'93, Gdansk, Poland, August 30 - September 3, 1993, Proceedings*, volume 711 of *Lecture Notes in Computer Science*, pages 392–402. Springer, 1993.

[3] U. S. N. Raju, Irlanki Sandeep, Nattam Sai Karthik, Rayapudi Siva Praveen, and Mayank Singh Sachan. Weighted finite automata based image compression on hadoop mapreduce framework. In Barbara Carminati and Latifur Khan, editors, *2015 IEEE International Congress on Big Data, New York City, NY, USA, June 27 - July 2, 2015*, pages 653–656. IEEE Computer Society, 2015.