

# A unifying approach to picture grammars<sup>☆</sup>

Matteo Pradella<sup>a,\*</sup>, Alessandra Cherubini<sup>b</sup>, Stefano Crespi Reghizzi<sup>a</sup>

<sup>a</sup>*Dipartimento di Elettronica e Informazione, Politecnico di Milano,  
P.zza L. da Vinci, 32, 20133 Milano, Italy*

<sup>b</sup>*Dipartimento di Matematica, Politecnico di Milano,  
P.zza L. da Vinci, 32, 20133 Milano, Italy*

---

## Abstract

Several old and recent classes of picture grammars, that variously extend context-free string grammars in two dimensions, are based on rules that rewrite arrays of pixels. Such grammars can be unified and extended using an approach, whereby the right part of a rule is formalized by means of a finite set of permitted tiles. We focus on a simple type of tiling, named *regional*, and define the corresponding regional tile grammars. They include both Siromoney's (or Matz's) Kolam grammars and their generalization by Průša, as well as Drewes's grid grammars. Regionally defined pictures can be recognized with polynomial-time complexity by an algorithm extending the CKY one for strings. Regional tile grammars and languages are strictly included into our previous tile grammars and languages, and are incomparable with Giammarresi-Restivo tiling systems (or Wang systems).

*Key words:* picture language, tiling, picture grammar, 2D language, CKY algorithm, syntactic pattern recognition

---

## 1. Introduction

Since the early days of formal language theory, considerable research effort has been spent towards the objective of extending grammar based approaches from one to two dimensions (2D), i.e., from string languages to picture languages. Several approaches have been proposed (and sometimes re-proposed) in the course of the years, which in different ways take inspiration from regular expressions and from Chomsky's string grammars, but, to the best of our knowledge, no general classification or detailed comparison of picture grammars has been attempted. It is fair to say that the immense

---

<sup>☆</sup>A preliminary version is [1]. Work partially supported by PRIN Project *Mathematical aspects and emerging applications of automata and formal languages*, ESF Programme *Automata: from Mathematics to Applications (AutoMathA)*, and CNR IEIIT.

\*Corresponding author.

*Email addresses:* pradella@elet.polimi.it (Matteo Pradella),  
alessandra.cherubini@polimi.it (Alessandra Cherubini), crespi@elet.polimi.it  
(Stefano Crespi Reghizzi)

success of grammar-based approaches for strings, e.g. in compilation and natural language processing, is far from being matched by picture grammars. Several causes for this may exist. First, the lack of broadly accepted reference models has caused a dispersion of research efforts. Second, the algorithmic complexity of parsing algorithm for 2D languages has rarely been considered, and very few efficient algorithms, and fewer implementations, exist. Last, but not least, most grammar types have been invented by theoreticians and their applicability in picture or image processing remains to be seen.

We try to remove, or at least to partially offset, the first two causes, thus hoping to set in this way the ground for applied research on picture grammars. First, we outline how several classical models of picture grammars based on array rewriting rules can be unified by a tiling based approach. A typical rewriting rule replaces a pixel array, occurring in some position in the picture, by a right part, which is a pixel array of equal size. Each grammar type considers different forms of rewriting rules, that we show how to formalize using more or less general sets of tiles. Then, we focus on a simple type of tile sets, those of *regional tile grammars*. This new class generalizes some classical models, yet it is proved to permit polynomial-time recognition of pictures by an approach extending the classical Cocke-Kasami-Younger (CKY) algorithm [2] of context-free (CF) string languages.

From the standpoint of more powerful grammar models, regional tile grammars correspond to a natural restriction of our previous *tile (rewriting) grammars* (TG) [3, 4]. For such grammars, a rule replaces a rectangular area filled with a nonterminal symbol with a picture belonging to the language defined by a specified set of tiles over terminal or nonterminal symbols. It is known that the TG family dominates the family of languages defined by the *tiling systems* (TS) of Giammarresi and Restivo [5] (which are equivalent to Wang systems [6][7]), and that the latter are NP-complete with respect to picture recognition time complexity. The new model enforces the constraint that the local language used to specify the right part of a rule is made by assembling a finite number of homogeneous rectangular pictures. Such tiling is related to Simplot's [8] interesting closure operation on pictures.

Regional tile grammars are then shown to dominate other grammar types. The first is the classical Kolam grammar type of Siromoney [9] (which, in its context-free form, is equivalent to the grammars of Matz [10]); it is less general because the right parts of grammar rules must be tiled in ways decomposable as vertical and horizontal concatenations. Three other grammar families are then shown to be less general: *Průša's type* [11], *grid* [12], and *context-free matrix grammars* [13]. The language inclusion properties for all the above families are thus clarified.

The presentation continues in Section 2 with preliminary definitions, then in Sections 3 and 4 with the definition of tile grammars, their regional variant, and relevant examples. In Section 4.1 we present the parsing algorithm and prove its correctness and complexity. In Section 5 we compare regional tile grammars and languages with other picture language families. The paper concludes by summarizing the main results.

## 2. Basic definitions

The following notation and definitions are mostly from [14] and [3].

**Definition 2.1.** Let  $\Sigma$  be a finite alphabet. A two-dimensional array of elements of  $\Sigma$  is a picture over  $\Sigma$ . The set of all pictures over  $\Sigma$  is  $\Sigma^{++}$ . A picture language is a subset of  $\Sigma^{++}$ .

For  $h, k \geq 1$ ,  $\Sigma^{(h,k)}$  denotes the set of pictures of size  $(h, k)$  (we will use the notation  $|p| = (h, k)$ ,  $|p|_{\text{row}} = h$ ,  $|p|_{\text{col}} = k$ ).  $\# \notin \Sigma$  is used when needed as a boundary symbol;  $\hat{p}$  refers to the bordered version of picture  $p$ . That is, for  $p \in \Sigma^{(h,k)}$ , it is

$$\begin{array}{cccccc}
 & & & & \# & \# & \dots & \# & \# \\
 & p(1,1) & \dots & p(1,k) & \# & p(1,1) & \dots & p(1,k) & \# \\
 p = & \vdots & \ddots & \vdots & \hat{p} = & \vdots & \ddots & \vdots & \vdots \\
 & p(h,1) & \dots & p(h,k) & \# & p(h,1) & \dots & p(h,k) & \# \\
 & & & & \# & \# & \dots & \# & \#
 \end{array}$$

A pixel is an element  $p(i, j)$  of  $p$ . If all pixels are identical to  $C \in \Sigma$  the picture is called  $C$ -homogeneous or  $C$ -picture.

Row and column concatenations are denoted  $\ominus$  and  $\oplus$ , respectively.  $p \ominus q$  is defined iff  $p$  and  $q$  have the same number of columns; the resulting picture is the vertical juxtaposition of  $p$  over  $q$ .  $p^{k \ominus}$  is the vertical juxtaposition of  $k$  copies of  $p$ ;  $p^{+ \ominus}$  is the corresponding closure.  $\oplus, {}^{k \oplus}, {}^{+ \oplus}$  are the column analogues.

**Definition 2.2.** Let  $p$  be a picture over  $\Sigma$ . The domain of a picture  $p$  is the set  $\text{dom}(p) = \{1, 2, \dots, |p|_{\text{row}}\} \times \{1, 2, \dots, |p|_{\text{col}}\}$ . A subdomain of  $\text{dom}(p)$  is a set  $d$  of the form  $\{x, x+1, \dots, x'\} \times \{y, y+1, \dots, y'\}$  where  $1 \leq x \leq x' \leq |p|_{\text{row}}$ ,  $1 \leq y \leq y' \leq |p|_{\text{col}}$ . We will often denote a subdomain by using its top-left and bottom-right coordinates, in the previous case the quadruple  $(x, y; x', y')$ .

The set of subdomains of  $p$  is denoted  $D(p)$ . Let  $d = \{x, \dots, x'\} \times \{y, \dots, y'\} \in D(p)$ , the subpicture  $\text{spic}(p, d)$  associated to  $d$  is the picture of size  $(x' - x + 1, y' - y + 1)$  such that  $\forall i \in \{1, \dots, x' - x + 1\}$  and  $\forall j \in \{1, \dots, y' - y + 1\}$ ,  $\text{spic}(p, d)(i, j) = p(x + i - 1, y + j - 1)$ .

A subdomain is called  $C$ -homogeneous (or homogeneous) when its associated subpicture is a  $C$ -picture.  $C$  is called the label of the subdomain.

Two subdomains  $d_a = (i_a, j_a; k_a, l_a)$  and  $d_b = (i_b, j_b; k_b, l_b)$  are horizontally adjacent (resp. vertically adjacent) iff  $j_b = l_a + 1$ , and  $k_b \geq i_a, k_a \geq i_b$  (resp.  $i_b = k_a + 1$ , and  $l_b \geq j_a, l_a \geq j_b$ ). We will call two subdomains adjacent, if they are either vertically or horizontally adjacent.

The translation of a subdomain  $d = (x, y; x', y')$  by displacement  $(a, b) \in \mathbb{Z}^2$  is the subdomain  $d' = (x + a, y + b; x' + a, y' + b)$ . We will write  $d' = d \oplus (a, b)$ . We will also sometimes apply  $\oplus$  to a set  $W$  of subdomains, meaning the set containing the translations of all the elements of  $W$ .

**Definition 2.3.** A homogeneous partition of a picture  $p$  is any partition  $\pi = \{d_1, d_2, \dots, d_n\}$  of  $\text{dom}(p)$  into homogeneous subdomains  $d_1, d_2, \dots, d_n$ .

The unit partition of  $p$ , written  $\text{unit}(p)$ , is the homogeneous partition of  $\text{dom}(p)$  defined by single pixels.

An homogeneous partition is called strong if adjacent subdomains have different labels.

We observe that if a picture  $p$  admits a strong homogeneous partition of  $\text{dom}(p)$  into subdomains, then the partition is unique and will be denoted by  $\Pi(p)$ . To illustrate, in Figure 2 are depicted pictures with outlined borders of subdomains. The marked partitions of the last two pictures are homogeneous but not strong, because some adjacent subdomains hold the same letter.

We now introduce the central concepts of *tile*, and *local language*.

**Definition 2.4.** We call *tile* a square picture of size  $(2,2)$ . We denote by  $\llbracket p \rrbracket$  the set of all tiles contained in a picture  $p$ .

Let  $\Sigma$  be a finite alphabet. A (two-dimensional) language  $L \subseteq \Sigma^{++}$  is *local* if there exists a finite set  $\theta$  of tiles over the alphabet  $\Sigma \cup \{\#\}$  such that  $L = \{p \in \Sigma^{++} \mid \llbracket \hat{p} \rrbracket \subseteq \theta\}$ . We will refer to such language as  $LOC(\theta)$ .

Locally testable languages in the strict sense (*LT*) are analogous to local languages, but are defined through square tiles with side possibly bigger than 2. In the rest of the paper we will call these variants of tiles *k-tiles*, to avoid confusion with standard  $2 \times 2$  tiles. For instance, *3-tiles* are square pictures of size  $(3,3)$ .

Last, we define *tiling systems* (TS). Tiling systems define the closure w.r.t. alphabetic projection of local languages, and are presented and studied extensively in [14].

**Definition 2.5.** A tiling system (*TS*) is a 4-tuple  $\mathcal{T} = (\Sigma, \Gamma, \theta, \pi)$ , where  $\Sigma$  and  $\Gamma$  are two finite alphabets,  $\theta$  is a finite set of tiles over the alphabet  $\Gamma \cup \{\#\}$ , and  $\pi : \Gamma \rightarrow \Sigma$  is an alphabetic projection.

The language defined by the tiling system  $\mathcal{T}$  (in the rest of the paper denoted by  $L(\mathcal{T})$ ) is the set of pictures  $\{\pi(p) \mid \hat{p} \in LOC(\theta)\}$ .

### 3. Tile grammars

We are going to introduce and study a very general grammar type specified by a set of rewriting rules (or productions). A typical rule has a left and a right part, both pictures of unspecified but equal (isometric) size. The left part is an  $A$ -homogeneous picture, where  $A$  is a nonterminal symbol. The right part is a picture of a local language over nonterminal symbols. Thus a rule is a scheme defining a possibly unbounded number of isometric pairs: left picture, right picture. In addition there are simpler rules whose right part is a single terminal.

The derivation process of a picture starts from a  $S$  (axiom)-homogeneous picture. At each step, an  $A$ -homogeneous subpicture is replaced with an isometric picture of the local language, defined by the right part of a rule  $A \rightarrow \dots$ . The process terminates when all nonterminals have been eliminated from the current picture.

For simplicity, this presentation focuses on nonterminal rules, thus excluding for instance that both terminal and nonterminal symbols are in the same right part. This normalization has a cost in terms of grammar dimension and readability, but does not lose generality. Indeed, more general kinds of rules (e.g. like those used in [3]), can be easily normalized by introducing some auxiliary nonterminals and rules. We will present and use analogous transformations when comparing with other grammar devices in Section 5, where we will talk about *nonterminal normal forms*.

**Definition 3.1.** A tile grammar (TG) is a tuple  $(\Sigma, N, S, R)$ , where  $\Sigma$  is a set of terminal symbols,  $N$  is a set of nonterminal symbols,  $S \in N$  is the starting symbol,  $R$  is a set of rules.

Let  $A \in N$ . There are two kinds of rules:

$$\text{Fixed size: } A \rightarrow t, \quad \text{where } t \in \Sigma; \quad (1)$$

$$\text{Variable size: } A \rightarrow \omega, \quad \omega \text{ is a set of non-concave tiles over } N \cup \{\#\}. \quad (2)$$

Concave tiles are like:

$$\begin{array}{|c|c|} \hline B & B \\ \hline C & B \\ \hline \end{array}$$

or a rotation thereof, where  $B \neq \#$  (so we use tiles having this structure only for borders). It is easy to see that all pictures in  $LOC(\omega)$ , where  $\omega$  is a set of non-concave tiles, admit a strong homogeneous partition.

Picture derivation is next defined as a relation between partitioned pictures.

**Definition 3.2.** Consider a tile grammar  $G = (\Sigma, N, S, R)$ , let  $p, p' \in (\Sigma \cup N)^{(h,k)}$  be pictures of identical size. Let  $\pi = \{d_1, \dots, d_n\}$  be a homogeneous partition of  $\text{dom}(p)$ . We say that  $(p', \pi')$  derives in one step from  $(p, \pi)$ , written

$$(p, \pi) \Rightarrow_G (p', \pi')$$

iff, for some  $A \in N$ , there exist in  $\pi$  an  $A$ -homogeneous subdomain  $d_i = (x, y; x', y')$ , called application area, and a rule  $A \rightarrow \alpha \in R$  such that  $p'$  is obtained substituting  $\text{spic}(p, d_i)$  in  $p$  with:

- $\alpha \in \Sigma$ , if  $A \rightarrow \alpha$  is of type (1),<sup>1</sup>
- $s \in LOC(\alpha)$ , if  $A \rightarrow \alpha$  is of type (2).

Moreover,  $\pi' = (\pi \setminus \{d_i\}) \cup (\Pi(s) \oplus (x - 1, y - 1))$ .

We say that  $(p', \pi')$  derives from  $(p, \pi)$  in  $n$  steps, written  $(p, \pi) \xRightarrow{n}_G (p', \pi')$ , iff  $p = p'$  and  $\pi = \pi'$ , when  $n = 0$ , or there are a picture  $p''$  and a homogeneous partition  $\pi''$  such that  $(p, \pi) \xRightarrow{n-1}_G (p'', \pi'')$  and  $(p'', \pi'') \Rightarrow_G (p', \pi')$ . We use the abbreviation  $(p, \pi) \xRightarrow{*}_G (p', \pi')$  for a derivation with a finite number of steps.

Roughly speaking, at each step of the derivation an  $A$ -homogeneous subpicture is replaced with an isometric picture of the local language, defined by the right part of a rule  $A \rightarrow \alpha$ , that admits a strong homogeneous partition. The process terminates when all nonterminals have been eliminated from the current picture.

In the rest of the paper, and when considering also other grammatical devices, we will drop the  $G$  symbol when it is clear from the context, writing e.g.  $(p, \pi) \xRightarrow{*} (p', \pi')$ .

---

<sup>1</sup>In this case,  $x = x'$  and  $y = y'$ .

**Definition 3.3.** The picture language defined by a grammar  $G$  (written  $L(G)$ ) is the set of  $p \in \Sigma^{++}$  such that

$$\left( S^{|p|}, \{\text{dom}(p)\} \right) \xRightarrow{*}_G (p, \text{unit}(p))$$

For short we also write  $S \xRightarrow{*}_G p$ .

We emphasize that, to generate a picture of a certain dimension, one must start from a picture of the same dimension.

We also will use the notation  $\mathcal{L}(X)$  to denote the class of languages generated by some formal device  $X$ , e.g.  $\mathcal{L}(TG)$  will denote the class of languages generated by tile grammars.

The following examples will be used later for comparing language families.

**Example 1.** One row and one column of  $b$ 's.

The set of pictures having one row and one column (both not at the border) that hold  $b$ 's, and the remainder of the picture filled with  $a$ 's is defined by the tile grammar  $G_1$  in Figure 1, where the nonterminals are  $\{A_1, A_2, A_3, A_4, V_1, V_2, H_1, H_2, X, A, B\}$ . We recall

$$G_1 : S \rightarrow \left[ \begin{array}{cccccccc} \# & \# & \# & \# & \# & \# & \# & \# \\ \# & A_1 & A_1 & V_1 & A_2 & A_2 & \# & \# \\ \# & A_1 & A_1 & V_1 & A_2 & A_2 & \# & \# \\ \# & H_1 & H_1 & V_1 & H_2 & H_2 & \# & \# \\ \# & A_3 & A_3 & V_2 & A_4 & A_4 & \# & \# \\ \# & A_3 & A_3 & V_2 & A_4 & A_4 & \# & \# \\ \# & \# & \# & \# & \# & \# & \# & \# \end{array} \right]$$

$$A_i \rightarrow \left[ \begin{array}{cccc} \# & \# & \# & \# \\ \# & X & X & \# \\ \# & A_i & A_i & \# \\ \# & A_i & A_i & \# \\ \# & \# & \# & \# \end{array} \right] \mid \left[ \begin{array}{cccc} \# & \# & \# & \# \\ \# & X & X & \# \\ \# & \# & \# & \# \end{array} \right], \text{ for } 1 \leq i \leq 4$$

$$X \rightarrow \left[ \begin{array}{cccccc} \# & \# & \# & \# & \# & \# \\ \# & A & X & X & \# & \# \\ \# & \# & \# & \# & \# & \# \end{array} \right] \mid a; \quad H_i \rightarrow \left[ \begin{array}{cccccc} \# & \# & \# & \# & \# & \# \\ \# & B & H_i & H_i & \# & \# \\ \# & \# & \# & \# & \# & \# \end{array} \right] \mid b, \text{ for } 1 \leq i \leq 2$$

$$A \rightarrow a; \quad B \rightarrow b; \quad V_i \rightarrow \left[ \begin{array}{ccc} \# & \# & \# \\ \# & B & \# \\ \# & V_i & \# \\ \# & V_i & \# \\ \# & \# & \# \end{array} \right] \mid b, \text{ for } 1 \leq i \leq 2.$$

$$p_1 = \begin{array}{|c|c|c|c|c|} \hline a & a & b & a & a \\ \hline b & b & b & b & b \\ \hline a & a & b & a & a \\ \hline a & a & b & a & a \\ \hline \end{array}$$

Figure 1: Tile grammar  $G_1$  (top) and a picture  $p_1$  (bottom) of Example 1.

that  $\left[ \begin{array}{c} \square \end{array} \right]$  denotes the set of tiles contained in the argument picture. This notation is

preferable to the listing of all tiles, shown next:

$$S \rightarrow \left\{ \begin{array}{|c|c|} \hline \# & \# \\ \hline \# & A_1 \\ \hline \end{array}, \begin{array}{|c|c|} \hline \# & \# \\ \hline A_1 & A_1 \\ \hline \end{array}, \dots, \begin{array}{|c|c|} \hline A_1 & V_1 \\ \hline H_1 & V_1 \\ \hline \end{array}, \begin{array}{|c|c|} \hline V_1 & A_2 \\ \hline V_1 & H_2 \\ \hline \end{array}, \dots, \begin{array}{|c|c|} \hline A_4 & A_4 \\ \hline \# & \# \\ \hline \end{array}, \begin{array}{|c|c|} \hline A_4 & \# \\ \hline \# & \# \\ \hline \end{array} \right\}.$$

An example of derivation is shown in Figure 2, where partitions are outlined for readability.

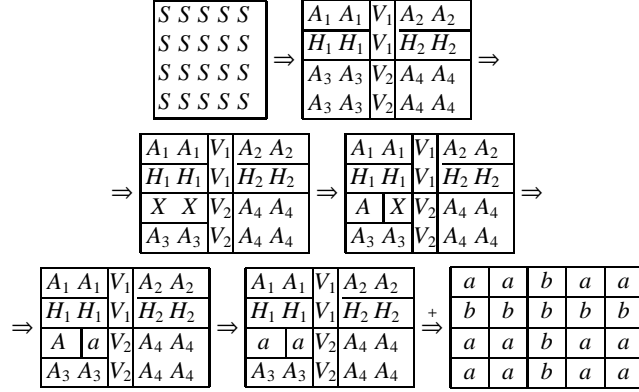


Figure 2: Derivation using grammar  $G_1$  of Example 1, Figure 1, with outlined partitions.

**Example 2.** Pictures with palindromic rows. Each row is an even palindrome over  $\{a, b\}$ . The grammar  $G_2$  is shown in Figure 3.

$$G_2 : S_P \rightarrow \left[ \begin{array}{|c|c|c|c|} \hline \# & \# & \# & \# \\ \hline \# & R & R & \# \\ \hline \# & S_P & S_P & \# \\ \hline \# & S_P & S_P & \# \\ \hline \# & \# & \# & \# \\ \hline \end{array} \right] \mid \left[ \begin{array}{|c|c|c|c|} \hline \# & \# & \# & \# \\ \hline \# & R & R & \# \\ \hline \# & \# & \# & \# \\ \hline \end{array} \right]$$

$$R \rightarrow \left[ \begin{array}{|c|c|c|c|c|c|} \hline \# & \# & \# & \# & \# & \# \\ \hline \# & A & R & R & A' & \# \\ \hline \# & \# & \# & \# & \# & \# \\ \hline \end{array} \right] \mid \left[ \begin{array}{|c|c|c|c|c|c|} \hline \# & \# & \# & \# & \# & \# \\ \hline \# & B & R & R & B' & \# \\ \hline \# & \# & \# & \# & \# & \# \\ \hline \end{array} \right]$$

$$R \rightarrow \left[ \begin{array}{|c|c|c|c|} \hline \# & \# & \# & \# \\ \hline \# & A & A' & \# \\ \hline \# & \# & \# & \# \\ \hline \end{array} \right] \mid \left[ \begin{array}{|c|c|c|c|} \hline \# & \# & \# & \# \\ \hline \# & B & B' & \# \\ \hline \# & \# & \# & \# \\ \hline \end{array} \right]$$

$$A \rightarrow a; B \rightarrow b; A' \rightarrow a; B' \rightarrow b.$$

$$p_2 = \begin{array}{|c|c|c|c|} \hline a & b & b & a \\ \hline b & a & a & b \\ \hline a & a & a & a \\ \hline \end{array}$$

Figure 3: Tile grammar  $G_2$  (top) and a picture  $p_2$  (bottom) of Example 2.

### 3.1. Properties of tile grammars

First, we state a language family inclusion between tiling systems (Definition 2.5) and tile grammars, proved in [3]. We will illustrate it with an example, both to give the reader an intuitive idea of the result, and to later re-use the example.

**Proposition 3.1.**  $\mathcal{L}(TS) \subset \mathcal{L}(TG)$ .

Consider a TS  $T = (\Sigma, \Gamma, \theta, \pi)$ , where  $\Sigma$  is the terminal alphabet,  $\theta$  is a tile-set,  $\Gamma$  is the tile-set alphabet, and  $\pi : \Gamma \rightarrow \Sigma$  is an alphabetic projection. It is quite easy to define a TG  $T'$  such that  $L(T') = L(T)$ . Informally, the idea is to take the tile-set  $\theta$  and add two markers, e.g.  $\{b, w\}$  in a “chessboard-like” fashion to build up a tile-set suitable for the right part of the variable size starting rule; other straightforward fixed size rules are used to encode the projection  $\pi$ .

We note how both  $\mathcal{L}(TS)$  and  $\mathcal{L}(TG)$  are closed under intersection with the class of all height-1 pictures: the classes resulting in that intersection are the well-known classes of recognizable and context-free, respectively, string languages. The inclusion is hence proper: any context-free, non-recognizable string language is also (when considered as a picture language) in  $\mathcal{L}(TG)$ , but not in  $\mathcal{L}(TS)$ .

The next example illustrates the reduction from a TS to a TG.

**Example 3.** *Square pictures of a's.*

The TS  $T_3$  is based on a local language over  $\{0, 1\}$  such that all pixels of the main diagonal are 1 and the remaining ones are 0, and on the projection  $\pi(0) = \pi(1) = a$ .  $T_3$  and the equivalent TG  $G_3$  are shown in Figure 4.

The “chessboard-like” construction is used to ensure that the only strong homogeneous partition obtained in applying a rule is the one in which partitions correspond to single pixels. This allows the application of terminal rules encoding projection  $\pi$ . Note that in the first rule of grammar  $G_3$  we used tiles arising from the two possible chessboard structures, i.e. the one with a “black” in top-left position, and the one with a “white” in the same place. Indeed, to fill areas above and below the diagonal with 0's we need both tiles

$$\begin{array}{|c|c|} \hline 0_b & 0_w \\ \hline 0_w & 0_b \\ \hline \end{array} \quad \text{and} \quad \begin{array}{|c|c|} \hline 0_w & 0_b \\ \hline 0_b & 0_w \\ \hline \end{array}.$$

Note also that the construction is applied in a straightforward way, just by imposing the two complementary chessboard patterns on it. We could simplify it in this particular case, because it is not necessary to distinguish  $1_w$  and  $1_b$ , as they appear only on the diagonal so they are never horizontally or vertically adjacent.

The following complexity property will be used to separate the TG language family from several subfamilies to be introduced.

In this paper as “parsing problem” we consider the problem of deciding if a given input picture is in  $L(G)$ , for a fixed grammar  $G$  (i.e. the also called *non-uniform membership problem*). The complexity of parsing algorithms is thus expressed in term of the size of the input picture.

**Proposition 3.2.** *The parsing problem for  $\mathcal{L}(TG)$  is NP-complete.*



$$\begin{aligned}
T_3 : \theta = & \left[ \begin{array}{cccccc} \# & \# & \# & \# & \# & \# \\ \# & 1 & 0 & 0 & 0 & \# \\ \# & 0 & 1 & 0 & 0 & \# \\ \# & 0 & 0 & 1 & 0 & \# \\ \# & 0 & 0 & 0 & 1 & \# \\ \# & \# & \# & \# & \# & \# \end{array} \right], \pi(0) = a, \pi(1) = a. \\
G_3 : S \rightarrow & \left[ \begin{array}{cccccc} \# & \# & \# & \# & \# & \# \\ \# & 1_b & 0_w & 0_b & 0_w & \# \\ \# & 0_w & 1_b & 0_w & 0_b & \# \\ \# & 0_b & 0_w & 1_b & 0_w & \# \\ \# & 0_w & 0_b & 0_w & 1_b & \# \\ \# & \# & \# & \# & \# & \# \end{array} \right] \cup \left[ \begin{array}{cccccc} \# & \# & \# & \# & \# & \# \\ \# & 1_w & 0_b & 0_w & 0_b & \# \\ \# & 0_b & 1_w & 0_b & 0_w & \# \\ \# & 0_w & 0_b & 1_w & 0_b & \# \\ \# & 0_b & 0_w & 0_b & 1_w & \# \\ \# & \# & \# & \# & \# & \# \end{array} \right] \\
& 1_w \rightarrow a, 1_b \rightarrow a, 0_w \rightarrow a, 0_b \rightarrow a.
\end{aligned}$$

Figure 4: For Example 3 the TS defining  $\{a^{(n,n)} \mid n > 1\}$  (top), and the equivalent TG grammar (bottom).

PROOF. Since the construction, illustrated in Example 3, used for proving Proposition 3.1 can be done in polynomial time, and thanks to the fact that the parsing problem for  $\mathcal{L}(TS)$  is NP-complete (see [15] where tiling systems are called *homomorphisms of local lattice languages*, or [16]) it follows that parsing  $\mathcal{L}(TG)$  is NP-hard.

For NP-completeness, we show that parsing  $\mathcal{L}(TG)$  is in NP. First, we assume without loss of generality that a TG  $G$  does not contain any *chain rule*, i.e. a rule of the form

$$A \rightarrow \left[ \begin{array}{cccc} \# & \# & \# & \# \\ \# & B & B & \# \\ \# & B & B & \# \\ \# & \# & \# & \# \end{array} \right], \quad B \in N$$

that corresponds to a renaming rule of a string grammar.

If this is not the case, it is possible to discard chain rules by directly using the well-known (e.g. [17]) approach for context-free string grammars.

We assume to have a candidate derivation

$$(S^{(h,k)}, \{\text{dom}(p)\}) \Rightarrow_G (p_1, \pi_1) \Rightarrow_G (p_2, \pi_2) \Rightarrow_G \cdots \Rightarrow_G (p_{n-1}, \pi_{n-1}) \Rightarrow_G (p, \text{unit}(p))$$

and we are going to prove that checking its correctness takes polynomial time in  $h, k$  (size of the picture), by considering the dominant parameters of time complexity.

First, the length  $n$  of this derivation, since there are no chain rules, is at most  $h \cdot k$ . In fact, we start from a partition with only one element coinciding with  $\text{dom}(p)$ , and at each step at least one element is added, arriving at step  $n$ , where the number of elements is  $h \cdot k$ , each corresponding to a pixel.

For each step, we must find the application area in  $(p_i, \pi_i)$ , and the corresponding rewritten nonterminal  $A$ , by comparing  $(p_i, \pi_i)$  with  $(p_{i+1}, \pi_{i+1})$ . The number of comparisons to be performed is at most  $h \cdot k$ .

Then, we have to find a rule  $A \rightarrow \omega$  in  $R$  which is compatible with the rewritten subpicture of  $p_{i+1}$  corresponding to the application area. So, at most we must check every rule in  $R$ , and every tile of its right part, on a subpicture, given by the application area, that has size at most  $h \cdot k$ . Hence, we have to consider for this step a number of checks that is at most

$$h \cdot k \cdot |R| \cdot \max_{(A \rightarrow \omega) \in R} |\omega|$$

Each of these considered steps can be done in polynomial time in every reasonable machine model, hence the resulting time complexity remains polynomial.  $\square$

From [3] it is known that the family of TG languages is closed w.r.t. union, column/row concatenations, column/row closure operations, rotation, and alphabetic projection.

As strings can always be seen as pictures having only one row, we mention that all the families presented in this work, that exactly define the context-free string languages if restricted to one dimension (i.e. all but tiling systems and grid grammars, presented in Section 5.3), are not closed w.r.t. intersection and complement.

#### 4. Regional tile grammars

We now introduce the central concept of *regional language*, and a corresponding specialization of tile grammars. The adjective “regional” is a metaphor of geographical political maps, where different regions are filled with different colors; of course, regions are rectangles.

Regional tile grammars are central to this work, because they are the most general among the polynomial-time parsable grammar models considered in this paper. We will see that it is easy to define the other kinds of 2D grammars by restricting the tiles used in regional tile grammars.

**Definition 4.1.** *A homogeneous partition is regional (HR) iff distinct (not necessarily adjacent) subdomains have distinct labels. A picture  $p$  is regional if it admits a HR partition. A language is regional if all its pictures are so.*

For example, consider Figure 5: the partitions in subdomains of the picture on the left is homogeneous and strong, but not regional, since four different subdomains bear the same symbol  $A$ . On right, a picture with regional partitions outlined is depicted.

A	A	B	A	A
A	A	B	A	A
D	D	B	D	D
A	A	C	A	A
A	A	C	A	A

$A_1$	$A_1$	B	$A_2$	$A_2$
$A_1$	$A_1$	B	$A_2$	$A_2$
$D_1$	$D_1$	B	$D_2$	$D_2$
$A_3$	$A_3$	C	$A_4$	$A_4$
$A_3$	$A_3$	C	$A_4$	$A_4$

Figure 5: Pictures with outlined partitions in subdomains: strong homogeneous partition (left), and regional (right).

Another (negative) example is in Figure 4: a “chessboard-like” picture admits a unique homogeneous partition, in which every subdomain corresponds to a single pixel. Note that in general these partitions are strong (adjacent subdomains have different symbols, like in a chessboard), but are not regional (e.g. in the variable size rule of grammar  $G_3$  there are multiple  $0_b$  symbols).

**Definition 4.2.** A regional tile grammar (RTG) is a tile grammar (see Definition 3.1), in which every variable size rule  $A \rightarrow \omega$  is such that  $LOC(\omega)$  is a regional language.

We note that the tile grammars presented in Examples 1 and 2 are regional, while the one of Example 3 ( $G_3$ ) is not. Another RTG is presented in the following example.

**Example 4.** Misaligned palindromes.

A picture is a “ribbon” of two rows, divided into four fields: at the top-left and at the bottom right of the picture are palindromes as in Example 2 (where rules for  $S_p$  are defined). The other two fields are filled with  $c$ 's and must not be adjacent. The corresponding regional tile grammar  $G_4$  is shown in Figure 6.

$$G_4 : S \rightarrow \left[ \begin{array}{cccccccc} \# & \# & \# & \# & \# & \# & \# & \# \\ \# & P_1 & P_1 & P_1 & P_1 & C_1 & C_1 & \# \\ \# & C_2 & C_2 & P_2 & P_2 & P_2 & P_2 & \# \\ \# & \# & \# & \# & \# & \# & \# & \# \end{array} \right]; P_1 \rightarrow S_p; P_2 \rightarrow S_p$$

$$C_i \rightarrow \left[ \begin{array}{ccccc} \# & \# & \# & \# & \# \\ \# & C & C_i & C_i & \# \\ \# & \# & \# & \# & \# \end{array} \right] \mid c, \text{ for } 1 \leq i \leq 2; C \rightarrow c.$$

$$p_4 = \begin{array}{cccccccc} a & a & b & b & a & a & c & c & c & c \\ c & c & b & a & b & a & a & b & a & b \end{array}$$

Figure 6: Regional tile grammar  $G_4$  (top) and a picture  $p_4$  (bottom) of Example 4.

Next, we study the form of tiles occurring in a regional local language.

**Definition 4.3.** Consider a tile set  $\theta$  over the alphabet  $\Sigma \cup \{\#\}$ .

We define the horizontal and vertical adjacency relations  $\mathcal{H}_\theta, \mathcal{V}_\theta \subseteq (\Sigma \cup \{\#\})^2$  as

$$A\mathcal{H}_\theta B \Leftrightarrow A \neq B \wedge \exists t \in \theta, \exists i \in \{1, 2\} : t(i, 1) = A \wedge t(i, 2) = B;$$

$$A\mathcal{V}_\theta B \Leftrightarrow A \neq B \wedge \exists t \in \theta, \exists j \in \{1, 2\} : t(1, j) = A \wedge t(2, j) = B.$$

Then, the adjacency relations are  $\mathcal{A}_\theta = \mathcal{H}_\theta \cup \mathcal{V}_\theta$  and  $\mathcal{A}'_\theta = \mathcal{H}_\theta^{-1} \cup \mathcal{V}_\theta$ .

**Proposition 4.1.** Let  $p \in \Sigma^{++}$  and  $\theta = \llbracket \hat{p} \rrbracket$ ; picture  $p$  is regional iff the incidence graphs of both  $\mathcal{A}_\theta \cap \Sigma^2$  and  $\mathcal{A}'_\theta \cap \Sigma^2$  are acyclic.

PROOF. First of all, we note that tiles occurring in  $\hat{p}$  for a regional picture  $p$  have the following form (or a rotation thereof):

$$\begin{array}{|c|c|} \hline A & A \\ \hline A & A \\ \hline \end{array}, \begin{array}{|c|c|} \hline A & A \\ \hline B & B \\ \hline \end{array}, \begin{array}{|c|c|} \hline A & A \\ \hline B & C \\ \hline \end{array}, \begin{array}{|c|c|} \hline A & B \\ \hline C & D \\ \hline \end{array}, \begin{array}{|c|c|} \hline \# & \# \\ \hline A & \# \\ \hline \end{array}, \begin{array}{|c|c|} \hline \# & \# \\ \hline A & A \\ \hline \end{array}, \begin{array}{|c|c|} \hline \# & \# \\ \hline A & B \\ \hline \end{array},$$

with  $A, B, C, D \in \Sigma$  all different. The incidence graphs of the adjacency relations of this tile-set are clearly all acyclic. Moreover, a picture exclusively made of these kinds of tiles admits a unique strong homogeneous partition. So, if we start from a regional picture  $\hat{p}$ , we obtain acyclic incidence graphs for the tile-set made of all its tiles.

Vice versa, if we consider a tile set  $\theta$  such that its adjacency relations are both acyclic, then tiles in  $\theta$  must be like those considered in the previous paragraph. Also, for any picture in  $LOC(\theta)$ , an acyclic  $\mathcal{A}_\theta$  means that any path going from the top-left corner and arriving to the bottom-right corner and performing only down and right movements cannot traverse two distinct subdomains bearing the same label. For  $\mathcal{A}'_\theta$  it is analogous, but starting from the top-right corner, arriving to the bottom-left corner and performing only left and down movements. But this means that  $LOC(\theta)$  is a regional language.  $\square$

Notice that this result uses the adjacency relations for tile-sets just described, i.e.  $\mathcal{A}_\theta$ , and  $\mathcal{A}'_\theta$ , in which the movements intuitively go from left to right and from top to bottom, and from right to left and from top to bottom. The same results hold also for different choices, e.g. we could consider a  $\mathcal{A}''_\theta = \mathcal{H}_\theta \cup \mathcal{V}_\theta^{-1}$ , i.e. moving from left to right and from bottom to top, instead of  $\mathcal{A}'_\theta$ .

**Definition 4.4.** A tile set  $\theta$  is called simple regional iff there exists a regional picture  $p$  such that  $\theta = \llbracket \hat{p} \rrbracket$ .

**Proposition 4.2.** For every simple regional tile set  $\theta$ , the language  $LOC(\theta)$  is regional.

PROOF. First, let us suppose that  $L = LOC(\theta)$  is non-regional. But this means that there exists  $p \in L$  that is not regional. Then,  $\theta$  is not simple regional, because  $\llbracket \hat{p} \rrbracket$  is not (by Proposition 4.1).  $\square$

**Proposition 4.3.** A local language  $L$  is regional iff there exist some simple regional tile sets  $\theta_1, \theta_2, \dots, \theta_n$ ,  $n \geq 1$ , such that  $L = \bigcup_{1 \leq i \leq n} LOC(\theta_i)$ .

PROOF. If  $L$  is regional, then by Proposition 4.1 it suffices to set

$$\{\theta_1, \theta_2, \dots, \theta_n\} = \{\llbracket \hat{p} \rrbracket \mid p \in L\}.$$

The other direction is a consequence of Proposition 4.2 and the fact that a finite union of regional languages is regional.  $\square$

Thanks to this result and without loss of generality<sup>2</sup>, in the rest of the paper we will always consider regional tile grammar where the right parts of type (2) rules are simple regional. In practice, right parts will be written as  $\llbracket q \rrbracket$ , where  $q$  is a bordered regional picture.

#### 4.1. Parsing for regional tile grammars

To present our version of the Cocke-Kasami-Younger (CKY) algorithm [2], we have to generalize from substrings to subpictures. Like the CKY algorithm for strings,

---

<sup>2</sup> $X \rightarrow \theta$  generates the same language as the rules  $X \rightarrow \theta_1 \mid \theta_2 \mid \dots \mid \theta_n$ .

our algorithm works bottom-up, by considering all subpictures of the input picture, starting from single pixels (i.e.  $1 \times 1$  subpictures), and then increasing their size. As a substring is identified by the positions of its first and last characters, a subpicture is conveniently identified by its subdomain. For simplicity and without loss of generality, we assume that the regional tile grammar considered does not contain chain rules.

The algorithm's main data structure is the *recognition matrix*, a four-dimensional matrix, holding lists of nonterminals, that the algorithm fills during its run. A non-terminal  $A$  is put into the matrix entry corresponding to subdomain  $d$ , if the same nonterminal can derive the subpicture  $\text{spic}(p, d)$ .

To decide if a rule can be used to derive the subpicture corresponding to subdomain  $d$ , the right part of the rule is examined, together with all the subdomains contained in  $d$ . Type (1) rules are easily managed, because they can only generate single terminal pixels, therefore they are considered only at the beginning with unitary subdomains. For example, let us consider grammar  $G_1$  of Example 1 (Figure 1), and its derivation shown in Figure 2. The pixel at position  $(3, 2)$  is an  $a$ , and the only possible generating terminal rules are  $X \rightarrow a$  and  $A \rightarrow a$ . So we enter both  $X$  and  $A$  into the recognition matrix at  $(3, 2; 3, 2)$ .

For a type (2) rule  $A \rightarrow \omega$  we need to check all the pictures in  $LOC(\omega)$ , isometric to the considered subpicture. Thanks to the regional constraint, every nonterminal used in the right part of the rule corresponds to a unique homogeneous rectangular area, if the rule is applicable. So we examine all the sets of nonterminals stored in the recognition matrix for all the subdomains contained in  $d$ : if we are able to find a set of subdomains which comply with the adjacency relations of the right part of the rule, then the rule is applicable. For example, let us consider the subdomain  $(3, 1; 3, 2)$  for the derivation of Figure 2. Subdomains  $(3, 1; 3, 1)$  and  $(3, 2; 3, 2)$  have already been considered, being "smaller", and the set  $\{A, X\}$  has been entered at positions  $(3, 1; 3, 1)$  and  $(3, 2; 3, 2)$ . This means that, if we consider  $X$  at  $(3, 1; 3, 1)$ , and  $A$  at  $(3, 2; 3, 2)$ , then all the adjacency relations of the type (2) rule for  $X$  in Figure 1 are satisfied (namely,  $\# \mathcal{H}A, A \mathcal{H}X, X \mathcal{H} \#, \# \mathcal{V}A, A \mathcal{V} \#, \# \mathcal{V}X, X \mathcal{V} \#$ ). So the algorithm places  $X$  into  $(3, 1; 3, 2)$ , since subpicture  $(3, 1; 3, 2)$  can be parsed to  $X$ .

**Remark** *In the pseudo-code, loops on Cartesian products are to be executed in lexicographic order. For example, in loop*

**for each**  $(i, j) \in \{1, \dots, 10\} \times \{3, 5, \dots, 11\}$ : ...

*the control variables (i.e.  $i$  and  $j$  in this case) will go through the following sequence of values:  $(1, 3), (1, 5), \dots, (1, 11), (2, 3), (2, 5), \dots, (10, 11)$ .*

We now present the details of the algorithm. Let  $p$  be a picture of size  $(m, n)$ , to be parsed with a regional tile grammar  $G = (\Sigma, N, S, R)$ .

**Definition 4.5.** *A recognition matrix  $\mathfrak{M}$  is a 4-dimensional  $m \times n \times m \times n$  matrix over the powerset of  $N$ .*

Being a generalization of the CKY algorithm for strings, the meaning of  $A \in \mathfrak{M}(i, j; h, k)$  is that  $A$  can derive the subpicture  $\text{spic}(p, (i, j; h, k))$ . In fact, only cells  $(i, j; h, k)$ , with  $h \geq i, k \geq j$ , are used: these cells are the four-dimensional counterpart of the upper triangular matrix used in classical CKY algorithm.

We introduce another data structure, the *subdomains vector*, to be used for recognizing the applicability of type (2) rules.

**Definition 4.6.** Consider a recognition matrix  $\mathfrak{M}$ , and a subdomain  $d = (i, j; k, l)$ . Let the nonterminal set  $N$  be arbitrarily ordered as  $A_1, A_2, \dots, A_{|N|}$ . The subdomains vector  $\mathfrak{D}(\mathfrak{M}, d)$  is a tuple  $(D_1, D_2, \dots, D_{|N|})$ , where every  $D_t$  is the set of subdomains  $d'$  such that  $A_t \in \mathfrak{M}(d')$  and  $d'$  is a subdomain contained in  $d$ ; if  $D_t$  is empty, then its conventional value is set to  $(0, 0; 0, 0)$ .

For any nonterminal  $A$ , the notation  $\mathfrak{D}(\mathfrak{M}, d)|_A$  denotes the component of the vector corresponding to  $A$ .

To simplify the notation, we shall write  $\mathfrak{D}(d)$  instead of  $\mathfrak{D}(\mathfrak{M}, d)$  at no risk of confusion, because the algorithm refers to a unique recognition matrix  $\mathfrak{M}$ . Moreover, we use the notation  $\overline{\mathfrak{D}(d)}$  for referring to the set of all possible vectors of subdomains present in  $\mathfrak{D}(d)$ , i.e.  $\overline{\mathfrak{D}(d)} := D_1 \times D_2 \times \dots \times D_{|N|}$ .

The main role of this ancillary data structure is to assign all the subdomains contained in a given subdomain  $d$ , to nonterminals, if possible, by considering the already filled portion of  $\mathfrak{M}$ . Using  $\mathfrak{D}$ , we are able to check if the adjacency relations of rules are satisfied. For example, if a rule  $A \rightarrow \alpha$  demands  $A_2 \mathcal{H}_\alpha A_8$ , then we only have to check if one of the elements of  $\overline{\mathfrak{D}(d)}$  has components 2 and 8 that are horizontally adjacent, with the domain corresponding to nonterminal  $A_2$  to the left. Figure 7 shows the procedure used to compute vector  $\mathfrak{D}$ .

It is important to remark that  $\mathfrak{D}$  is central for keeping the time of the parsing algorithm polynomial w.r.t. the input size. Indeed, in a regional tile grammar the number of homogeneous subdomains to be considered for a candidate application area is at most  $|N|$ , because the number of different homogeneous areas arising from the application of a rule is at most the number of nonterminals of the grammar. Hence  $\overline{\mathfrak{D}}$  has size less than  $(m^2 n^2)^{|N|}$ . In principle, it would be possible to adapt this algorithm also to an unrestricted tile grammar, but in this case the number of elements to be considered could be exponential, as the number of different homogeneous subdomains could be as big as the number of pixels of the application area, i.e. up to  $m \cdot n$  (see e.g. grammar  $G_3$  in Figure 4).

The actual procedure for checking if a rule of the grammar can be applied to a given rectangle  $(i, j; k, l)$  is presented in Figure 8. Based on vector  $\mathfrak{D}$ , computed for the relevant subdomain  $(i, j; k, l)$ , the procedure checks, for a right part  $\omega$  of a variable-size rule, if all adjacency constraints are satisfied.

The *Main* procedure, presented in Figure 9, is structured as a straightforward generalization to two dimensions of the CKY parsing algorithm. The input picture  $p$  is in  $L(G)$  iff  $S \in \mathfrak{M}(1, 1; |p|_{row}, |p|_{col})$ .

#### 4.1.1. Correctness and complexity of parsing

We start with a technical lemma, used to prove the correctness of the CheckRule procedure.

**Lemma 4.1.** Let  $\omega$  be a regional set of tiles and  $d$  a subdomain.  $\text{CheckRule}(\omega, d)$  returns true iff there exists a rule  $C \rightarrow \omega$ , such that  $(p_0, \pi_0) \Rightarrow_G (p_1, \pi_1)$ , where  $d \in \pi_0$ , and  $\text{spic}(p_0, d)$  is a  $C$ -picture.

**Procedure** *Compute* $\mathfrak{D}(\mathfrak{M}, (i, j; k, l))$ :  
Every set in  $\mathfrak{D}$  is empty;  
**for each**  $(i', j') \in \{i, \dots, k\} \times \{j, \dots, l\}$ :  
    **for each**  $(k', l') \in \{i', \dots, k\} \times \{j', \dots, l\}$ :  
        **for each**  $A \in \mathfrak{M}(i', j'; k', l')$ :  
            put  $(i', j'; k', l')$  into the set  $\mathfrak{D}|_A$ ;  
**for each**  $A \in N$ :  
    **if**  $\mathfrak{D}|_A = \emptyset$  **then** put  $(0, 0; 0, 0)$  into the set  $\mathfrak{D}|_A$ ;  
**return**  $\mathfrak{D}$ .

Figure 7: Compute $\mathfrak{D}$

PROOF. By construction, a true output of  $\text{CheckRule}(\omega, d)$  is equivalent to the fact that there exist  $q \in \text{LOC}(\omega)$  and a partition of  $d$  into the subdomains  $d_1, d_2, \dots, d_r$ , such that:

1. every  $\text{spic}(q, d_j)$  is an  $A$ -picture, for some nonterminal  $A \in \mathfrak{M}(d_j)$ ;
2. if  $\text{spic}(q, d_j)$  is an  $A$ -picture, then for all  $d_k \neq d_j$  the subpicture  $\text{spic}(q, d_k)$  is not an  $A$ -picture.

This means that  $\Pi(q) \oplus (x-1, y-1)$ , where  $d = (x, y; x', y')$ , is the HR partition  $\{d_1, d_2, \dots, d_r\}$ . Moreover, starting from  $(p_0, \pi_0)$ , where  $\text{spic}(p_0, d)$  is a  $C$ -picture, it is possible to apply a rule  $C \rightarrow \omega$  in a derivation step  $(p_0, \pi_0) \Rightarrow_G (p_1, \pi_1)$ , where  $\pi_0 = \{d, d'_1, d'_2, \dots, d'_n\}$ ,  $\pi_1 = \{d'_1, d'_2, \dots, d'_n\} \cup \{d_1, d_2, \dots, d_r\}$ , and  $q = \text{spic}(p_1, d) \in \text{LOC}(\omega)$ .  $\square$

After this, the correctness is easy to prove, analogously to the 1D case [2].

**Theorem 4.1.**  $\mathfrak{M}(d) = \{A \in N \mid A \xRightarrow{*}_G \text{spic}(p, d)\}$ , for every subdomain  $d$ .

PROOF. The proof is by induction on the size of subdomain  $d$ .

*Base:*  $d = (i, j; i, j)$ . This means that  $|\text{spic}(p, d)| = (1, 1)$ . Hence,  $A \xRightarrow{*}_G \text{spic}(p, d)$  iff  $A \rightarrow \text{spic}(p, d) \in R$ . This case is handled by the first loop of procedure Main, the one over each pixel  $p(i, j)$ . If  $\text{spic}(p, d) = t$ , and there exists a rule  $A \rightarrow t$ , then the algorithm enters  $A$  into  $\mathfrak{M}(d)$ . Vice versa,  $A \in \mathfrak{M}(d)$  means that the algorithm has put  $A$  in the set, therefore there must exist a rule  $A \rightarrow \text{spic}(p, d)$ .

*Induction:* let us consider  $d = (i, j; i+v-1, j+h-1)$ ,  $v > 1$ , or  $h > 1$ , or both. We prove that  $A \xRightarrow{*}_G \text{spic}(p, d)$  implies  $A \in \mathfrak{M}(d)$ . In this case, the size of the subpicture is not  $(1, 1)$ , therefore the first rule used in the derivation  $A \xRightarrow{*}_G \text{spic}(p, d)$  is a variable size rule  $A \rightarrow \omega$ . Thanks to the two nested loops with control variables  $(v, h)$  and  $(i, j)$ , when the algorithm considers  $d$ , it has already considered all its subdomains  $d_1, d_2, \dots, d_k$ . By the induction hypothesis, for every  $1 \leq j \leq k$ ,  $B \xRightarrow{*}_G \text{spic}(p, d_j)$  implies  $B \in \mathfrak{M}(d_j)$ . Hence (Lemma 4.1),  $\text{CheckRule}(\omega, d)$  must be true, and the algorithm puts  $A$  in  $\mathfrak{M}(d)$ .

```

Procedure CheckRule ( $\mathfrak{D}, \omega, (i, j, k, l)$ ) :
for each  $(d_1, d_2, \dots, d_{|N|}) \in \overline{\mathfrak{D}}$ ;
     $f := True$ ;
    for each  $(N_a, N_b) \in \mathcal{H}_\omega$ :
        if  $d_a = (i_a, j_a; k_a, l_a)$  and  $d_b = (i_b, j_b; k_b, l_b)$  are not such that
             $j_b = l_a + 1$ , and  $k_b \geq i_a, k_a \geq i_b$ ,
        then  $f := False$ ;
    for each  $(N_a, N_b) \in \mathcal{V}_\omega$ :
        if  $d_a = (i_a, j_a; k_a, l_a)$  and  $d_b = (i_b, j_b; k_b, l_b)$  are not such that
             $i_b = k_a + 1$ , and  $l_b \geq j_a, l_a \geq j_b$ ,
        then  $f := False$ ;
    for each  $(\#, N_a) \in \mathcal{H}_\omega$ :
        if  $d_a = (i_a, j_a; k_a, l_a)$  and  $j_a \neq j$  then  $f := False$ ;
    for each  $(N_a, \#) \in \mathcal{H}_\omega$ :
        if  $d_a = (i_a, j_a; k_a, l_a)$  and  $l_a \neq l$  then  $f := False$ ;
    for each  $(\#, N_a) \in \mathcal{V}_\omega$ :
        if  $d_a = (i_a, j_a; k_a, l_a)$  and  $i_a \neq i$  then  $f := False$ ;
    for each  $(N_a, \#) \in \mathcal{V}_\omega$ :
        if  $d_a = (i_a, j_a; k_a, l_a)$  and  $k_a \neq k$  then  $f := False$ ;
    if  $f$  then return True;
return False.

```

Figure 8: CheckRule



**Procedure Main:**Every set in  $\mathfrak{M}$  is empty;**for each** pixel  $p(i, j) = t$ :    **if** there exists a fixed size rule  $A \rightarrow t \in R$ ,    **then** put  $A$  into the set  $\mathfrak{M}(i, j; i, j)$ ;**for each**  $(v, h) \in \{1, \dots, m\} \times \{1, \dots, n\}$ :    **for each**  $(i, j) \in \{1, \dots, m - v\} \times \{1, \dots, n - h\}$ :         $\mathfrak{D} := \text{Compute}\mathfrak{D}(\mathfrak{M}, (i, j; i + v - 1, j + h - 1))$ ;        **for each** variable size rule  $(A \rightarrow \omega) \in R$ :            **if**  $\text{CheckRule}(\mathfrak{D}, \omega, (i, j; i + v - 1, j + h - 1))$ ,            **then** put  $A$  into the set  $\mathfrak{M}(i, j; i + v - 1, j + h - 1)$ ;**return**  $\mathfrak{M}$ .

Figure 9: Main

Conversely, we prove that  $A \in \mathfrak{M}(d)$  implies  $A \xRightarrow{*}_G \text{spic}(p, d)$ .  $A \in \mathfrak{M}(d)$  means that procedure Main has put  $A$  in the set. Therefore,  $\text{CheckRule}(\omega, d)$  must be true. Thanks to Lemma 4.1, this is equivalent to the existence of an applicable variable size rule  $A \rightarrow \omega$  for the first step of the derivation  $A \xRightarrow{*}_G \text{spic}(p, d)$ . The rest of the derivation holds by induction hypothesis.  $\square$

**Theorem 4.2.** *The parsing problem for  $\mathcal{L}(\text{RTG})$  has temporal complexity that is polynomial with respect to the input picture size.*

PROOF. First, it is straightforward to see that  $\text{Compute}\mathfrak{D}$  performs a number of operations that is  $O(|N| \cdot m^2 n^2)$ .

Let us now consider the  $\text{CheckRule}$  procedure. This procedure performs a loop for each element of  $\overline{\mathfrak{D}}$ , which contains a number of elements that is less than  $(m^2 n^2)^{|\mathfrak{D}|}$ , and nested loops on  $\mathcal{H}_\omega$  and  $\mathcal{V}_\omega$ . Therefore the number of checks performed by it is dominated by a value that is

$$O\left((m^2 n^2)^{|\mathfrak{D}|} \cdot \max_{(A \rightarrow \omega) \in R} \{|\mathcal{H}_\omega|, |\mathcal{V}_\omega|\}\right).$$

Coming finally to the *Main* procedure, we note that its core part consists of two nested loops, over two sets that are at most  $m \cdot n$  each. The body of these two loops consists in a call to  $\text{Compute}\mathfrak{D}$ , and then another loop over the grammar rules, comprising a call to  $\text{CheckRule}$  (hence the dominant part).

Therefore, the number of operations performed is at most

$$O\left(|R| \cdot \max_{(A \rightarrow \omega) \in R} \{|\mathcal{H}_\omega|, |\mathcal{V}_\omega|\} \cdot (m^2 n^2)^{|\mathfrak{D}|} \cdot m^2 n^2\right).$$

$x'$	1	$x$	1	$x$	0	$x$
0	$1'$	0	1	0	0	0
$x$	1	$x'$	1	$x$	0	$x$
1	1	1	$1'$	1	1	1
$x$	1	$x$	1	$x'$	0	$x$
0	1	0	1	0	$0'$	0
$x$	1	$x$	1	$x$	0	$x'$

Figure 10: A picture of the language  $L_{lt}$  of Example 5

Each of these operations can be done in polynomial time in every reasonable machine model, therefore the resulting time complexity is polynomial w.r.t. the picture size.  $\square$

The property of having polynomial time complexity for picture recognition, united with the rather simple and intuitively pleasing form of RTG rules, should make them a worth addition to the series of array rewriting grammar models conceived in past years.

## 5. Comparison with other language families

In this section we prove or recall some inclusion relations between grammar models and corresponding language families. To this end we rely on the examples of Section 4, and on the separation of complexity classes.

In presenting other grammatical models we have been faced with a dilemma: to stick to the original formulation, or to reformulate the definition in terms more comparable with our own. We have opted for the former, because otherwise we would have incurred the penalty of proving that the old and new formulations are equivalent.

We start by comparing regional tile grammars and tiling systems. To this end, we adapt a proof and an example introduced by Průša in [11].

**Example 5.** Consider a language  $L_{lt}$  over the alphabet  $\Sigma = \{0, 0', 1, 1', x, x'\}$  where the “primed” symbols are used on the diagonal. A picture  $p$  is in  $L_{lt}$  if, and only if:

1.  $p$  is a square picture of odd size;
2.  $p(i, j) \in \{0, 1, x\}$ , when  $i \neq j$ ;  $p(i, j) \in \{0', 1', x'\}$ , otherwise.
3.  $p(i, j) \in \{x, x'\}$  iff  $i$  and  $j$  are odd;
4. if  $p(i, j) \in \{1, 1'\}$  then the  $i$ -th row or the  $j$ -th column (or both) is made of symbols taken from  $\{1, 1'\}$ .

An example picture is shown in Figure 10. It is quite easy to see that  $L_{lt}$  is a locally testable language, definable through a set of 3-tiles. Primed symbols by definition appear only on the main diagonal, and are used to have only square pictures.

**Proposition 5.1.**  $\mathcal{L}(RTG)$  and  $\mathcal{L}(LT)$  are incomparable.

PROOF. First, we know from [14] that  $\mathcal{L}(LT) \subset \mathcal{L}(TS)$ , and that  $\mathcal{L}(RTG)$  define context-free string languages, if restricted to one dimension, so there are languages in  $\mathcal{L}(RTG)$  that are not in  $\mathcal{L}(LT)$ .

To end the proof, we need a language that is in  $\mathcal{L}(LT)$  but not in  $\mathcal{L}(RTG)$ . We suppose, by contradiction, that there exists a RTG  $G = (\Sigma, N, S, R)$  such that  $L(G) = L_{lt}$  of Example 5. Without loss of generality, we assume that  $R$  does not contain chain rules, and that all right sides of rules in  $R$  are simple regional. We consider a natural number  $n = 2k + 1$  big enough to comply with the requirements presented in the rest of the proof.

First, let  $L_1$  be  $\{p \in L_{lt} \mid |p| = (n, n)\}$ . Clearly,  $|L_1| = 2^{n-1}$ , and it contains at least  $\lceil 2^{n-1}/|R| \rceil$  pictures that can be generated in the first step by the same rule.

We now fix such a rule, e.g.  $S \rightarrow \alpha$ , and let  $L_2$  be the subset of  $L_1$  generated by this rule. In a  $n \times n$  picture, the number of possible partitions in homogeneous subpictures is less than  $(n^4)^{|N|}$ . This means that there exists a set  $L_3 \subseteq L_2$ , having size  $|L_3| \geq \frac{2^{n-1}}{|R| \cdot n^{|N|}}$  such that every picture in it was generated by  $G$  starting with the same rule  $S \rightarrow \alpha$ , and such that the initial  $S$ -homogeneous picture was replaced by the same  $s \in LOC(\alpha)$ .

Depending on the chosen rule's right part, i.e.  $\alpha$ , we now identify a row or a column of the picture in an odd position, and call it  $\lambda$ . We have two cases: either (1) every  $s \in LOC(\alpha)$  is made of homogeneous subpictures having all both width and height less than  $n$ ; or (2) in every  $s \in LOC(\alpha)$  there is at least one homogeneous subpicture  $s'$  having width or height equal to  $n$  (but clearly not both, because we are not considering chain rules). In case (1), let  $\lambda$  be the first row. In case (2), let  $\lambda$  be one of the rows or columns in an odd position and completely contained in  $s'$ .

Let  $L_4$  be a subset of  $L_2$  such that every picture in it has the same  $\lambda$ . Because of its definition, if we fix an odd row of pictures in  $L_{lt}$ , then columns of even indexes that are completely filled by 1 and 1' are determined by it (if we fix an odd column, it is analogous but with rows). Hence,  $|L_4| \leq 2^{\frac{n-1}{2}}$ .

We can assume that  $n$  is sufficiently large so that  $|L_3| > |L_4|$ , i.e. there is at least a picture in  $L_3$  which is not present in  $L_4$ . So we are able to find in  $L_3$  two pictures  $p$  and  $q$  that are generated by the same initial rule,  $S \rightarrow \alpha$ , with the same initial strong homogeneous partition (the one determined by  $s$ ), and such that  $\lambda$  in  $p$  is different from  $\lambda$  in  $q$ . Now consider all the subpictures of  $p$  and  $q$  that are in the positions corresponding to the initial strong homogeneous partition. Of these subpictures, we consider only the sets  $P' = \{p'_1, p'_2, \dots, p'_i\}$ , and  $Q' = \{q'_1, q'_2, \dots, q'_i\}$ , with  $i \leq |N|$ , that contain subpictures that intersect with  $\lambda$  in  $p$  and in  $q$ , respectively. If we replace in  $p$  all the elements of  $P'$  with the elements in  $Q'$ , we obtain a picture which is derivable from  $S \rightarrow \alpha$ , but it is not in  $L_{lt}$ , because it contains columns (or rows in some cases (2)) that are not compatible with the fixed  $\lambda$ .  $\square$

The fact that  $\mathcal{L}(LT) \subset \mathcal{L}(TS)$  implies the following statement.

**Corollary 5.1.**  $\mathcal{L}(RTG)$  and  $\mathcal{L}(TS)$  are incomparable.

This last result, together with the facts that RTG rules are a restricted form of TG rules, and that  $\mathcal{L}(TS) \subset \mathcal{L}(TG)$ , gives us the following:

**Corollary 5.2.**  $\mathcal{L}(RTG) \subset \mathcal{L}(TG)$ .

### 5.1. Context-free Kolam grammars

This class of grammars has been introduced by Siromoney et al. [9] under the name “Array grammars”, later renamed “Kolam Array grammars” in order to avoid confusion with Rosenfeld’s homonymous model. Much later Matz reinvented the same model [10] (considering only CF rules). We prefer to keep the historical name, CF Kolam grammars (CFKG), and to use the more succinct definition of Matz.

**Definition 5.1.** A sentential form over an alphabet  $V$  is a non-empty well-parenthesized expression using the two concatenation operators,  $\ominus$  and  $\oplus$ , and symbols taken from  $V$ .  $\mathcal{SF}(V)$  denotes the set of all sentential forms over  $V$ . A sentential form  $\phi$  defines either one picture over  $V$  denoted by  $\langle\!\langle\phi\rangle\!\rangle$ , or none.

For example,  $\phi_1 = ((a \oplus b) \ominus (b \oplus a)) \in \mathcal{SF}(\{a, b\})$  and  $\langle\!\langle\phi_1\rangle\!\rangle$  is the picture  $\begin{array}{|c|c|} \hline a & b \\ \hline b & a \\ \hline \end{array}$ .

On the other hand  $\phi_2 = ((a \oplus b) \ominus a)$  denotes no picture, since the two arguments of the  $\ominus$  operator have different column numbers.

CF Kolam grammars are defined analogously to CF string grammars. Derivation is similar: a sentential form over terminal and nonterminal symbols results from the preceding one by replacing a nonterminal with some corresponding right hand side of a rule. The end of a derivation is reached when the sentential form does not contain any nonterminal symbols. If this resulting form denotes a picture, then that picture is generated by the grammar.

**Definition 5.2.** A context-free Kolam grammar (CFKG) is a tuple  $G = (\Sigma, N, S, R)$ , where  $\Sigma$  is the finite set of terminal symbols, disjoint from the set  $N$  of nonterminal symbols;  $S \in N$  is the start symbol; and  $R \subseteq N \times \mathcal{SF}(N \cup \Sigma)$  is the set of rules. A rule  $(A, \phi) \in R$  will be written as  $A \rightarrow \phi$ .

For a grammar  $G$ , we define the *derivation* relation  $\Rightarrow_G$  on the sentential forms  $\mathcal{SF}(N \cup \Sigma)$  by  $\psi_1 \Rightarrow_G \psi_2$  iff there is some rule  $A \rightarrow \phi$ , such that  $\psi_2$  results from  $\psi_1$  by replacing an occurrence of  $A$  by  $\phi$ . As usual,  $\overset{*}{\Rightarrow}_G$  denotes the reflexive and transitive closure of  $\Rightarrow_G$ . Notice that the derivation thus defined rewrites strings, not pictures.

From the derived sentential form, one then obtains the denoted picture. The picture language generated by  $G$  is the set

$$L(G) = \{\langle\!\langle\psi\rangle\!\rangle \mid \psi \in \mathcal{SF}(\Sigma), S \overset{*}{\Rightarrow}_G \psi\}.$$

With a slight abuse of notation, we will often write  $A \overset{*}{\Rightarrow}_G p$ , with  $A \in N, p \in \Sigma^{++}$ , instead of  $\exists \phi : A \overset{*}{\Rightarrow}_G \phi, \langle\!\langle\phi\rangle\!\rangle = p$ .

It is convenient to consider a normal form with exactly two or zero nonterminals in the right part of a rule [10].

**Definition 5.3.** A CF Kolam grammar  $G = (\Sigma, N, S, R)$ , is in Chomsky Normal Form (CNF) iff every rule in  $R$  has the form either  $A \rightarrow t$ , or  $A \rightarrow B \ominus C$ , or  $A \rightarrow B \oplus C$ , where  $A, B, C \in N$ , and  $t \in \Sigma$ .

We know from [10] that for every CFKG  $G$ , if  $L(G)$  does not contain the empty picture, there exists a CNF CFKG  $G'$ , such that  $L(G) = L(G')$ . Also, the classical algorithm to translate a string grammar into CNF can be easily adapted to CFKGs.

**Example 6.** The following Chomsky Normal Form grammar  $G_5$  defines the set of pictures such that each column is an odd length palindrome.

$$\begin{aligned}
S &\rightarrow V \oplus S \mid A_1 \ominus A_2 \mid B_1 \ominus B_2 \mid a \mid b \\
V &\rightarrow A_1 \ominus A_2 \mid B_1 \ominus B_2 \mid a \mid b \\
A_2 &\rightarrow V \ominus A_1 \mid a \\
B_2 &\rightarrow V \ominus B_1 \mid b \\
A_1 &\rightarrow a \\
B_1 &\rightarrow b.
\end{aligned}$$

### 5.1.1. Comparison with other models

First, we sketchily and intuitively show that the original CF Kolam definition is equivalent to the one introduced by Matz. The following description is directly taken from [9].

Let  $G = (\Sigma, N, S, R)$ , be a Kolam context-free grammar, where  $N = N_1 \cup N_2$ ,  $N_1$  a finite set of nonterminals,  $N_2$  a finite set of intermediates,  $\Sigma$  a finite set of terminals,  $R = R_1 \cup R_2 \cup R_3$ ,  $R_1$  a finite set of nonterminal rules,  $R_2$  a finite set of intermediate rules,  $R_3$  a finite set of terminal rules.  $S \in N_1$  is the start symbol.

$R_1$  is a set of pairs  $(A, B)$  (written  $A \rightarrow B$ ),  $A \in N_1$ ,  $B \in (N_1 \cup N_2)^{+\oplus}$  or  $B \in (N_1 \cup N_2)^{+\ominus}$ .

$R_2$  is a set of pairs  $(B, C)$ ,  $B \in N_2$ ,  $C \in (N_2 \cup \{x_1, x_2, \dots, x_k\})^{+\oplus}$ ,

with  $x_1, \dots, x_k \in \Sigma^{++}$ ,  $|x_i|_{row} = |x_{i+1}|_{row}$ ,  $1 \leq i < k$ ;

or  $C \in (N_2 \cup \{x_1, x_2, \dots, x_k\})^{+\ominus}$ , with  $x_1, \dots, x_k \in \Sigma^{++}$ ,  $|x_i|_{col} = |x_{i+1}|_{col}$ ,  $1 \leq i < k$ .

$R_3$  is a set of pairs  $(A, t)$ ,  $A \in (N_1 \cup N_2)$  and  $t \in \Sigma^{++}$ .

(Derivation) If  $A$  is an intermediate, then the intermediate language generated by  $A$  is  $M_A = \{x \mid A \xrightarrow{*} x, x \in \{x_1, \dots, x_k\}^{+\oplus}, x_j \in \Sigma^{++}, |x_i|_{row} = |x_{i+1}|_{row}, 1 \leq i < k\}$  or  $M_A = \{x \mid A \xrightarrow{*} x, x \in \{x_1, \dots, x_k\}^{+\ominus}, x_j \in \Sigma^{++}, |x_i|_{col} = |x_{i+1}|_{col}, 1 \leq i < k\}$ . Derivation proceeds as follows. Starting from  $S$ , nonterminal rules are applied without any restriction as in a string grammar, till all the nonterminals are replaced, introducing parentheses whenever necessary. Now replace for each intermediate  $A$  in  $N_2$  elements from  $M_A$ , subject to the conditions imposed by  $\oplus$ ,  $\ominus$ . The replacements start from the innermost parentheses and proceeds outwards. The derivation comes to an end if the condition for  $\ominus$  or  $\oplus$  is not satisfied.

Grammar  $G_5$  of Example 6 complies with this definition. In it,  $A_1$  and  $B_1$  are intermediates.

It is very easy to see that the original definition of CF Kolam grammars is equivalent to the new one given by Matz. Right part of rules are made of vertical or horizontal concatenations of nonterminals or fixed terminal pictures. So we can define an equivalent grammar that is as stated in Definition 5.2, by translating the right part of rules that contain terminal pictures  $x_1, x_2, \dots, x_p$ , decomposing each picture  $x_i$  in a sentential form  $\phi$  such that  $x_i = \langle \phi \rangle$ . Vertical or horizontal concatenations are then treated analogously (e.g. we translate  $AB$  into  $(A \oplus B)$ ). Clearly, we do not need to distinguish nonterminals from intermediate symbols.

**Proposition 5.2.**  $\mathcal{L}(CFKG) \subset \mathcal{L}(RTG)$ .

PROOF. In [3] a construction is given to prove that a CF Kolam grammar (in the form defined by Matz [10]) can be transformed into a TG. It turns out that the TG thus constructed is a RTG.

Sketchily, consider a CF Kolam grammar  $G$  in CNF. Rules  $A \rightarrow t, t \in \Sigma$  are identical in the two models and generate the same kind of languages (i.e. single terminal symbols). Rules  $A \rightarrow B \oplus C$  of  $G$  are equivalent to RTG rules having the following form:

$$A \rightarrow \left[ \begin{array}{cccccc} \# & \# & \# & \# & \# & \# \\ \# & B & B & C & C & \# \\ \# & B & B & C & C & \# \\ \# & \# & \# & \# & \# & \# \end{array} \right]$$

Rules  $A \rightarrow B \ominus C$  of  $G$  are equivalent to RTG rules having the following form:

$$A \rightarrow \left[ \begin{array}{cccc} \# & \# & \# & \# \\ \# & B & B & \# \\ \# & B & B & \# \\ \# & C & C & \# \\ \# & C & C & \# \\ \# & \# & \# & \# \end{array} \right]$$

The inclusion is strict, because the language of Example 1 was shown by Matz [10] to trespass the generative capacity of his grammars.  $\square$

The fact that the picture recognition problem for CF Kolam grammars has been recently proved [18] to be polynomial in time of course follows from the above inclusion property and from Theorem 4.2.

For the special case of CF Kolam grammars in CNF, we note that the parsing time complexity is  $O(m^2n^2(m+n))$  [18]. Some of the reasons of this significant difference are the following. Kolam grammars in CNF are much simpler, because in the right part of a rule there are at most two distinct nonterminals. So, checking if a rule is applicable has complexity which is linear with respect to the picture width or height.

## 5.2. Průša's context-free grammars

In the quest for generality, D. Průša [11] has recently defined a grammar model that extends CF Kolam rules, gaining some generative capacity. The model is for instance able to generate the language of Example 1.

### 5.2.1. Definitions

The following definitions are taken and adapted from [19, 11].

**Definition 5.4.** A 2D CF Průša grammar (PG) is a tuple  $(\Sigma, N, S, R)$ , where  $\Sigma$  is the finite set of terminal symbols, disjoint from the set  $N$  of nonterminal symbols;  $S \in N$  is the start symbol; and  $R \subseteq N \times (N \cup \Sigma)^{++}$  is the set of rules.

**Definition 5.5.** Let  $G = (\Sigma, N, S, R)$  be a PG. We define a picture language  $L(G, A)$  over  $\Sigma$  for every  $A \in N$ . The definition is given by the following recursive descriptions:

- (i) If  $A \rightarrow w$  is in  $R$ , and  $w \in \Sigma^{++}$ , then  $w \in L(G, A)$ .
- (ii) Let  $A \rightarrow w$  be a production in  $R$ ,  $w = (N \cup \Sigma)^{(m,n)}$ , for some  $m, n \geq 1$ . Let  $p_{i,j}$ , with  $1 \leq i \leq m$ ,  $1 \leq j \leq n$ , be pictures such that:
1. if  $w(i, j) \in \Sigma$ , then  $p_{i,j} = w(i, j)$ ;
  2. if  $w(i, j) \in N$ , then  $p_{i,j} \in L(G, w(i, j))$ ;
  3. let  $P_k = p_{k,1} \oplus p_{k,2} \oplus \cdots \oplus p_{k,n}$ . For any  $1 \leq i < m$ ,  $1 \leq j \leq n$ ,  $|p_{i,j}|_{col} = |p_{i+1,j}|_{col}$ ; and  $P = P_1 \ominus P_2 \ominus \cdots \ominus P_m$ .

Then  $P \in L(G, A)$ .

The set  $L(G, A)$  contains all and only the pictures that can be obtained by applying a finite sequence of rules (i) and (ii). The language  $L(G)$  generated by grammar  $G$  is defined as the language  $L(G, S)$ .

Informally, rules can either be terminal rules, in this case managed exactly as tile grammars or Kolam grammars, or have a picture as right part. In this latter case, the right part is seen as a “grid”, where nonterminals can be replaced by other pictures, but maintaining its grid-like structure. Note that the grid meshes may differ in size.

**Example 7.** The grammar  $G_6$  of Figure 11 generates the language of pictures with one row and one column of  $b$ 's in a background of  $a$ 's (see Example 1).

$$\begin{array}{c}
 A \quad V \quad A \\
 S \rightarrow \begin{array}{ccc} H & b & H \\ A & V & A \end{array}, \quad A \rightarrow AM \mid M, \quad M \rightarrow \begin{array}{c} a \\ M \end{array} \mid a, \\
 \\
 V \rightarrow \begin{array}{c} b \\ V \end{array} \mid b, \quad H \rightarrow bH \mid b.
 \end{array}$$

Figure 11: PG  $G_6$  of Example 7.

It would be simple to prove that every Průša grammar admits the following normal form:

**Definition 5.6.** A Průša grammar  $G = (\Sigma, N, S, R)$ , is in Nonterminal Normal Form (NNF) iff every rule in  $R$  has the form either  $A \rightarrow t$ , or  $A \rightarrow w$ , where  $A \in N$ ,  $w \in N^{++}$ , and  $t \in \Sigma$ .

### 5.2.2. Comparison with other models

To compare Průša grammars with tile grammars, we note that the two models are different in their derivations. Tile grammars start from a picture made of  $S$ 's having a fixed size, and being every derivation step isometric, the resulting picture, if any, has the same size. On the other hand, Průša grammars start from a single  $S$  symbol, and then “grow” the picture derivation step by derivation step, obtaining, if any, a usually larger picture.

First, we prove that the language of Example 4 cannot be defined by Průša grammars, so the language families are different. To this aim, we use a technique analogous to the one introduced for proving Proposition 5.1.

**Proposition 5.3.**  $\mathcal{L}(PG) \neq \mathcal{L}(RTG)$ .

PROOF. Let  $G = (\Sigma, N, S, R)$  be a PG such that  $L(G) = L(G_4)$ , where  $G_4$  is the RTG presented in Example 4. Without loss of generality we assume that  $R$  does not contain chain rules, and that for every rule  $A \rightarrow \omega$ , it is  $|\omega|_{row} \leq 2$ .

In the rest of the proof we classify the derivations, depending on the rule that is applied first, call it  $S \rightarrow \omega$ , where  $|\omega| = (x, y)$ ,  $1 \leq x \leq 2$ ,  $y \geq 1$ . Moreover, we will consider the subset  $L' \subseteq L(G_4)$ , such that every picture has two rows,  $3n$  columns for any  $n \geq 1$ , and is such that the two  $c$ -homogeneous subpictures in it have size  $(1, n)$ :

$$L' = \left\{ \begin{array}{|ccc|} \hline w & w^R & c^n \\ \hline c^n & w' & w'^R \\ \hline \end{array} \in L(G_4) \mid n > 0, |w| = |w'| = n \right\}.$$

We will call  $L'_\omega$  the set of pictures in  $L'$  generated by applying  $S \rightarrow \omega$  first.

First, we consider the case in which  $\omega$  has the form  $\frac{A}{B}$ . In this case both  $A$  and  $B$  must generate CF string languages. Since the language  $ww^Rc^{|w|}$  is not context free,  $A$  cannot generate exactly, and for any  $n$ , such strings, and the same holds *mutatis mutandis* for  $B$ .

Indeed, if we consider the string languages  $\{ww^Rc^h\}$  and  $\{c^kw'w'^R\}$ , we can apply the pumping lemma for CF string languages by considering for “pumping” either the  $ww^R$  part, or the  $w'w'^R$ , or the parts made of  $c$  symbols, or a combination thereof. If we keep  $h$  or  $k$  bounded, we can nonetheless generate an unbounded number of pictures of  $L(G_4)$ , but there will also be an unbounded number of pictures of  $L'$  not generable in such a way (i.e. those having a number of  $c$ 's greater than the chosen bound). Analogously, if we keep one of both the parts  $ww^R$  and  $w'w'^R$  bounded, there will be an unbounded number of pictures of  $L'$  not generable as well.

So, either  $L'_\omega$  is finite (or empty), or  $A$  and  $B$  generate CF languages that properly contain  $\{ww^Rc^{|w|}\}$  and  $\{c^{|w|}w'w'^R\}$ , respectively.  $A$  and  $B$  must generate strings having respectively the form  $ww^Rc^h$ , and  $c^kw'w'^R$ , where  $h$  and  $k$  are not bounded by any constants. Being  $h$  and  $k$  unbounded, we can take a string generated by  $A$  and one generated by  $B$  such that  $h > 2|w'|$ , and  $h + 2|w| = k + 2|w'|$ . But in this case the corresponding picture is not in  $L(G_4)$ . Hence, we can safely assume that  $y > 1$ .

Now we have to consider starting rules having  $1 \leq x \leq 2$ ,  $y > 1$ . We fix  $n$ , so that there are not any pictures of  $L'$  generable starting with a rule with  $x = 2$  and  $y = 1$ , and the value of  $n$  is big enough to comply with the requirements of the rest of the proof.

Clearly, the number of elements in the set  $X(n)$  defined as the one of pictures in  $L'$  for the fixed  $n$  is  $2^{2n}$ , and  $X(n)$  contains at least  $\lceil 2^{2n}/|R| \rceil$  pictures that are generated in the first step by the same rule  $S \rightarrow \omega$ . We call this subset  $L'_\omega(n)$ , because it corresponds to the finite subset of  $L'_\omega$  for the chosen value of  $n$ .

Without loss of generality, we assume that  $n > y$ , so each nonterminal in  $w$  generates a subpicture (that in the rest of the proof we will index by  $p_{i,j}$ ,  $1 \leq i \leq x$ ,  $1 \leq j \leq y$ ) having at most two rows and at least one column. Being the number of different sequences  $|p_{1,1}|_{col}, |p_{1,2}|_{col}, \dots, |p_{1,y}|_{col}, |p_{2,1}|_{row}, |p_{2,2}|_{row}$  limited by  $2(3n)^y$  (each  $|p_{1,i}|_{col}$  is less than  $3n$  and at most there are two rows), there exists a subset  $Y(n)$  of  $L'_\omega(n)$ , having cardinality  $|Y(n)| \geq 2^{2n}/(2|R|(3n)^y)$ , in which for any two pictures  $p$  and  $p'$ , and for every  $i, j$ , the size  $|p_{i,j}|$  is equal to  $|p'_{i,j}|$ .



Let  $W(n)$  be a subset of  $L'_\omega(n)$  such that every picture in it is like  $\begin{bmatrix} q^R & q & c^n \\ c^n & q & q^R \end{bmatrix}$  (i.e. the central third of the picture is made of two equal rows). Clearly,  $|W(n)| \leq 2^n$ .

We can assume that  $n$  is large enough so that  $|Y(n)| > |W(n)|$ . But this means that in  $Y(n)$  there are two different pictures  $p = \begin{bmatrix} q^R & q & c^n \\ c^n & s & s^R \end{bmatrix}$ , and  $p' = \begin{bmatrix} q'^R & q' & c^n \\ c^n & s' & s'^R \end{bmatrix}$ , with  $q \neq s$ ,  $q' \neq s'$ , and (1)  $q \neq q'$  or (2)  $s \neq s'$ . We know that  $y > 1$ , so if we replace  $p_{1,1}$  and  $p_{2,1}$  (if  $x = 2$ ) in  $p$  with  $p'_{1,1}$  and  $p'_{2,1}$ , in case (1), we obtain a picture generated by  $G$  that is not in  $L(G_4)$ . Case (2) is analogous, but considers the right part of  $p$ , i.e.  $p_{1,y}$  and  $p_{2,y}$ .  $\square$

Indeed, Průša grammars can be seen as a restricted form of regional tile grammars, as stated by the following proposition.

**Proposition 5.4.**  $\mathcal{L}(PG) \subset \mathcal{L}(RTG)$ .

PROOF. Consider a PG in NNF  $G$ . First of all, we assume without loss of generality that for any rule, nonterminals used in its right part are all different. If this is not the case, e.g. assume that we have a rule

$$A \rightarrow \begin{array}{cc} X & Y \\ Z & X \end{array},$$

then we can rename one of the  $X$  symbols to a freshly introduced nonterminal  $X'$ , and then add the chain rule  $X' \rightarrow X$ .

Let us define a RTG  $G'$  equivalent to  $G$ . Since the conversion of terminal rules is obvious we only discuss nonterminal rules. For a nonterminal rule of  $G$ , e.g.

$$A \rightarrow \begin{array}{ccc} B_{1,1} & \dots & B_{1,k} \\ \vdots & \ddots & \vdots \\ B_{h,1} & \dots & B_{h,k} \end{array}$$

we introduce the following rule in  $G'$ :

$$A \rightarrow \left[ \begin{array}{ccccccc} \# & \# & \# & \dots & \# & \# & \# \\ \# & B_{1,1} & B_{1,1} & \dots & B_{1,k} & B_{1,k} & \# \\ \# & B_{1,1} & B_{1,1} & \dots & B_{1,k} & B_{1,k} & \# \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ \# & B_{h,1} & B_{h,1} & \dots & B_{h,k} & B_{h,k} & \# \\ \# & B_{h,1} & B_{h,1} & \dots & B_{h,k} & B_{h,k} & \# \\ \# & \# & \# & \dots & \# & \# & \# \end{array} \right].$$

Note that each nonterminal  $B_{i,j}$  is repeated four times in the right part of the rule, so to have the tile  $\begin{bmatrix} B_{i,j} & B_{i,j} \\ B_{i,j} & B_{i,j} \end{bmatrix}$ , that can be used to “cover” a rectangular area of any size. Notice that the original grid alignments are preserved by RTG derivations.

Essentially, Průša grammars can be seen as RTG's with the additional constraint that tiles used in the right parts of rules must not have one of these forms:

$$\begin{array}{|c|c|} \hline A & B \\ \hline C & C \\ \hline \end{array}, \begin{array}{|c|c|} \hline A & C \\ \hline B & C \\ \hline \end{array}, \begin{array}{|c|c|} \hline C & C \\ \hline A & B \\ \hline \end{array}, \begin{array}{|c|c|} \hline C & A \\ \hline C & B \\ \hline \end{array}$$

with  $A, B, C$  all different. □

**Proposition 5.5.**  $\mathcal{L}(CFKG) \subset \mathcal{L}(PG)$ .

PROOF. For containment, it suffices to note that the constraints on tiles of the corresponding tile grammar, introduced in the proof of Proposition 5.4, are a weaker form of the constraints used for proving Proposition 5.2.

The containment is strict, since Průša grammar can generate the language of one column and one row of  $b$ 's in a field of  $a$ 's (see Example 7), while CF Kolam grammar cannot [10]. □

### 5.3. Grid grammars

Grid grammars are an interesting formalism defined by Drewes [20],[12]. Grid grammars are based on an extension of quadrees [21], in which the number of “quadrants” is not limited to four, but can be  $k^2$ , with  $k \geq 2$  (thus forming a square “grid”).

Following the tradition of quadrees, and differently from the other formalisms presented here, grid grammars generate pictures which are seen as sets of points on the “unit square” delimited by the points  $(0,0)$ ,  $(0,1)$ ,  $(1,0)$ ,  $(1,1)$  of the Cartesian plane. The following definitions are taken (and partially adapted) from [12].

*Let the unit square be divided by a evenly spaced grid into  $k^2$  squares, for some  $k \geq 2$ . A production of a grid picture grammar consists of a nonterminal symbol on the left-hand side and the square grid on the right-hand side, each of the  $k^2$  squares in the grid being either black or white or labelled with a nonterminal.*

*A derivation starts with the initial nonterminal placed in the unit square. Then productions are applied repeatedly until there is no nonterminal left, finally yielding a generated picture. As usual, a production is applied by choosing a square containing a nonterminal  $A$  and a production with left-hand symbol  $A$ . The nonterminal is then removed from the square and the square is subdivided into smaller black, white, and labelled squares according to the right-hand side of the chosen production. The set of all pictures generated in this manner constitutes the picture language generated by the grammar.*

*A picture generated by a grid picture grammar can be written as a string expression. Let the unit black square be represented by the symbol  $B$ , and the white unit square by  $W$ . By definition, each of the remaining pictures in the generated language consists of  $k^2$  subpictures  $\pi_{1,1}, \dots, \pi_{1,k}, \dots, \pi_{k,1}, \dots, \pi_{k,k}$ , each scaled by the factor  $1/k$ , going from bottom-left  $\pi_{1,1}$  to top right  $\pi_{k,k}$ . If  $t_{i,j}$  is the expression representing  $\pi_{i,j}$  (for  $1 \leq i, j \leq k$ ), then  $[t_{1,1}, \dots, t_{1,k}, \dots, t_{k,1}, \dots, t_{k,k}]$  represents the picture itself (for  $k = 2$  it is a quadtree).*

In order to compare such model, in which a picture is in the unit square and back-and-white, with the ones presented in this work, we introduce a different but essentially compatible formalization, in which the generated pictures are square arrays of symbols, and the terminal alphabet is not limited to black and white.

### 5.3.1. Definitions

To define grid grammars and their languages, we introduce a new definition that is similar to the one used for Kolam grammars in Section 5.1.

**Definition 5.7.** For a fixed  $k \geq 2$ , a *sentential form over an alphabet  $V$*  is either a symbol  $a \in V$ , or  $[t_{1,1}, \dots, t_{1,k}, \dots, t_{k,1}, \dots, t_{k,k}]$ , and every  $t_{i,j}$  being a sentential form.  $\mathcal{SF}(V)$  denotes the set of all sentential forms over  $V$ .

A sentential form  $\phi$  defines a set of pictures  $\langle\!\langle\phi\rangle\!\rangle$ :

- $\langle\!\langle a \rangle\!\rangle$ , with  $a \in V$ , represents the set  $\{a\}^{(n,n)}$ ,  $n \geq 1$  of all  $a$ -homogeneous square pictures;
- $\langle\!\langle [t_{1,1}, \dots, t_{1,k}, \dots, t_{k,1}, \dots, t_{k,k}] \rangle\!\rangle$ , represents the set of all square grid pictures where every  $\langle\!\langle t_{i,j} \rangle\!\rangle$  has the same size  $n \times n$ , for  $n \geq 1$ , and  $\langle\!\langle t_{1,1} \rangle\!\rangle$  is at the bottom-left corner,  $\dots$ ,  $\langle\!\langle t_{1,k} \rangle\!\rangle$  is at the bottom right corner,  $\dots$ , and  $\langle\!\langle t_{k,k} \rangle\!\rangle$  is at the top right corner.

Note that we maintained in the sentential forms the original convention of starting from the bottom-left position. For example, consider the sentential form

$$\phi = [[a, b, [a, b, b, a], c], a, B, [b, a, a, b]].$$

The smallest picture in  $\langle\!\langle\phi\rangle\!\rangle$  is depicted in Figure 12.

B	B	B	B	a	a	b	b
B	B	B	B	a	a	b	b
B	B	B	B	b	b	a	a
B	B	B	B	b	b	a	a
b	a	c	c	a	a	a	a
a	b	c	c	a	a	a	a
a	a	b	b	a	a	a	a
a	a	b	b	a	a	a	a

Figure 12: Example picture generated by the form  $[[a, b, [a, b, b, a], c], a, B, [b, a, a, b]]$ .

**Definition 5.8.** A *grid grammar (GG)* is a tuple  $G = (\Sigma, N, S, R)$ , where  $\Sigma$  is the finite set of terminal symbols, disjoint from the set  $N$  of nonterminal symbols;  $S \in N$  is the start symbol; and  $R \subseteq N \times \mathcal{SF}(N \cup \Sigma)$  is the set of rules. A rule  $(A, \phi) \in R$  will be written as  $A \rightarrow \phi$ .

For a grammar  $G$ , we define the *derivation relation*  $\Rightarrow_G$  on the sentential forms  $\mathcal{SF}(N \cup \Sigma)$  by  $\psi_1 \Rightarrow_G \psi_2$  iff there is some rule  $A \rightarrow \phi$ , such that  $\psi_2$  results from  $\psi_1$  by replacing an occurrence of  $A$  by  $\phi$ . As usual,  $\overset{*}{\Rightarrow}_G$  denotes the reflexive and transitive closure of  $\Rightarrow_G$ . As with Kolam grammars, the derivation thus defined rewrites strings, not pictures.

The derived sentential form denotes a set of pictures. Formally, the picture language generated by  $G$  is the set

$$L(G) = \left\{ p \in \langle\!\langle\psi\rangle\!\rangle \mid \psi \in \mathcal{SF}(\Sigma), S \overset{*}{\Rightarrow}_G \psi \right\}.$$

In the literature, parameter  $k$  is fixed for a grid grammar  $G$ , i.e. all the right parts of rules are either terminal or  $k$  by  $k$  grids. This constraint could be relaxed, by allowing different  $k$  for different rules: the results that are shown next still hold for this generalization.

It is trivial to see that grid grammars admit the following normal form:

**Definition 5.9.** A grid grammar  $G = (\Sigma, N, S, R)$ , is in Nonterminal Normal Form (NNF) iff every rule in  $R$  has the form either  $A \rightarrow t$ , or  $A \rightarrow [B_{1,1}, \dots, B_{1,k}, \dots, B_{k,1}, \dots, B_{k,k}]$ , where  $A, B_{i,j} \in N$ , and  $t \in \Sigma$ .

**Example 8.** A simple example of a grid grammar in NNF is:

$$S \rightarrow [S, B, S, B, B, B, S, B, S], \quad S \rightarrow a, \quad B \rightarrow b.$$

The generated language is that of “recursive” crosses of  $b$ 's in a field of  $a$ 's. Figure 13 shows an example picture of the language.

$a$	<b><math>b</math></b>	$a$	<b><math>b</math></b>	<b><math>b</math></b>	<b><math>b</math></b>	$a$	$a$	$a$
<b><math>b</math></b>	<b><math>b</math></b>	<b><math>b</math></b>	<b><math>b</math></b>	<b><math>b</math></b>	<b><math>b</math></b>	$a$	$a$	$a$
$a$	<b><math>b</math></b>	$a$	<b><math>b</math></b>	<b><math>b</math></b>	<b><math>b</math></b>	$a$	$a$	$a$
<b><math>b</math></b>	<b><math>b</math></b>	<b><math>b</math></b>	<b><math>b</math></b>	<b><math>b</math></b>	<b><math>b</math></b>	<b><math>b</math></b>	<b><math>b</math></b>	<b><math>b</math></b>
<b><math>b</math></b>	<b><math>b</math></b>	<b><math>b</math></b>	<b><math>b</math></b>	<b><math>b</math></b>	<b><math>b</math></b>	<b><math>b</math></b>	<b><math>b</math></b>	<b><math>b</math></b>
<b><math>b</math></b>	<b><math>b</math></b>	<b><math>b</math></b>	<b><math>b</math></b>	<b><math>b</math></b>	<b><math>b</math></b>	<b><math>b</math></b>	<b><math>b</math></b>	<b><math>b</math></b>
$a$	<b><math>b</math></b>	$a$	<b><math>b</math></b>	<b><math>b</math></b>	<b><math>b</math></b>	$a$	$a$	$a$
<b><math>b</math></b>	<b><math>b</math></b>	<b><math>b</math></b>	<b><math>b</math></b>	<b><math>b</math></b>	<b><math>b</math></b>	$a$	$a$	$a$
$a$	<b><math>b</math></b>	$a$	<b><math>b</math></b>	<b><math>b</math></b>	<b><math>b</math></b>	$a$	$a$	$a$

Figure 13: A picture of Example 8;  $b$  symbols are written in boldface for better readability.

### 5.3.2. Comparison with other models

First, we note that this is the only 2D grammatical model presented in this paper which cannot generate string (i.e. 1D) languages, since all the generated pictures, if any, have the same number of rows and columns by definition.

It is easy to see that the class of languages generated by grid grammars are a proper subset of the one of Průša grammars. In fact, a grid grammar can be seen as a particular kind of Průša grammar, in which symbols in right part of rules generate square pictures having the same size.

Surprisingly, the same reasoning can be applied also to prove inclusion w.r.t. CF Kolam grammars.

**Proposition 5.6.**  $\mathcal{L}(GG) \subset \mathcal{L}(CFKG)$ .

PROOF. Given a grid grammar  $G = (\Sigma, N, S, R)$  for simplicity in NNF, we construct an equivalent CFKG.

- (i) For terminal rules  $A \rightarrow t, t \in \Sigma$ , we introduce the following rules in the equivalent CF Kolam grammar  $G'$ :

$$A \rightarrow (A \oplus A_v) \ominus (A_h \oplus t) \mid t, \quad A_h \rightarrow A_h \oplus t \mid t, \quad A_v \rightarrow t \ominus A_v \mid t$$

where  $A_h, A_v$  are freshly introduced nonterminals, not used in other rules. It is easy to see that these rules can only generate all the square pictures made of  $t$ 's.

- (ii) For nonterminal rules  $A \rightarrow [B_{1,1}, \dots, B_{1,k}, \dots, B_{k,1}, \dots, B_{k,k}]$ , we add the following “structurally equivalent” kind of rules:

$$A \rightarrow \begin{array}{c} (B_{k,1} \oplus \dots \oplus B_{k,k}) \\ \ominus \\ \dots \\ \ominus \\ (B_{1,1} \oplus \dots \oplus B_{1,k}) \end{array}$$

To show the equivalence  $L(G) = L(G')$ , we use induction on derivation steps. As base case, we note that terminal rules of  $G$  are equivalent to the rules of  $G'$  introduced at (i).

Induction step: consider a nonterminal rule like in (ii). By induction hypothesis, all  $B_{j,i}$  of  $G'$  generate languages equivalent to their homonym in  $G$ , and all made of square pictures. We will use the notation  $b_{j,i}$  for referring to pictures generated by  $B_{j,i}$ .

By definition of  $\ominus$ ,  $|(b_{j,1} \oplus \dots \oplus b_{j,k})|_{col} = |(b_{j+1,1} \oplus \dots \oplus b_{j+1,k})|_{col}$ , for all  $1 \leq j < k$ . Moreover, by definition of  $\oplus$ ,  $|b_{j,i}|_{row} = |b_{j,i-1}|_{row}$ , for all  $1 \leq i < k$ . Being all squares, this means that the sentential form  $(b_{k,1} \oplus \dots \oplus b_{k,k}) \ominus \dots \ominus (b_{1,1} \oplus \dots \oplus b_{1,k})$  of  $G'$  generates a picture iff all  $b_{j,i}$  have the same size. But this also means that it is equivalent to the sentential form  $[B_{1,1}, \dots, B_{1,k}, \dots, B_{k,1}, \dots, B_{k,k}]$  of  $G$ .

The inclusion is proper, because by definition grid grammars cannot generate non-square pictures (e.g. string languages).  $\square$

#### 5.4. Context-free matrix grammars

The early model of CF matrix grammars [13] is a very limited kind of CF Kolam grammars. The following definition is taken and adapted from [22].

**Definition 5.10.** Let  $G = (H, V)$  where  $H = (\Sigma', N, S, R)$  is a string grammar, where  $N$  is the set of nonterminals,  $R$  is a set of productions,  $S$  is the starting symbol,  $\Sigma' = \{A_1, A_2, \dots, A_k\}$ ,  $V$  is a set of string grammars,  $V = \{V_1, V_2, \dots, V_k\}$  where each  $A_i$  is the start symbol of string grammar  $V_i$ . The grammars in  $V$  are defined over a terminal alphabet  $\Sigma$ , which is the alphabet of  $G$ . A grammar  $G$  is said to be a context-free matrix grammar (CFMG) iff  $H$  and all  $V_i$  are CF grammars.

Let  $p \in \Sigma^{++}$ ,  $p = c_1 \oplus c_2 \oplus \dots \oplus c_n$ .  $p \in L(G)$  iff there exists a string  $A_{x_1} A_{x_2} \dots A_{x_n} \in L(H)$  such that every column  $c_j$ , seen as a string, is in  $L(V_{x_j})$ ,  $1 \leq j \leq n$ . The string  $A_{x_1} A_{x_2} \dots A_{x_n}$  is said to be an intermediate string deriving  $p$ .

Informally, the grammar  $H$  is used to generate a horizontal string of starting symbols for the “vertical grammars”  $V_j$ ,  $1 \leq j \leq k$ . Then, the vertical grammars are used to generate the columns of the picture. If every column has the same height, then the generated picture is defined, and is in  $L(G)$ .

**Example 9.** *The language of odd-width rectangular pictures over  $\{a, b\}$ , where the first row, the last row, and the central column are made of  $b$ 's, the rest is filled with  $a$ 's is defined by the CFMG  $G_7$  of Figure 14.*

$$\begin{aligned}
G_7 &= (H, \{V_1, V_2\}) \text{ where} \\
H &: S \rightarrow A_1 S A_1 \mid A_2 \\
V_1 &: A_1 \rightarrow bA; \quad A \rightarrow aA \mid b; \\
V_2 &: A_2 \rightarrow bA_2 \mid b.
\end{aligned}$$

$$p_7 = \begin{array}{|cccccc|}
\hline
b & b & b & b & b & b \\
a & a & a & b & a & a \\
a & a & a & b & a & a \\
a & a & a & b & a & a \\
a & a & a & b & a & a \\
b & b & b & b & b & b \\
\hline
\end{array}$$

Figure 14: CF matrix grammar  $G_7$  of Example 9 (top), and an example picture (bottom).

#### 5.4.1. Comparison with other grammar families

First, we note that it is trivial to show that the class of CFMG languages is a proper subset of CF Kolam languages.

**Proposition 5.7.**  $\mathcal{L}(CFMG) \subset \mathcal{L}(CFKG)$ .

Intuitively, it is possible to consider the string sub-grammars  $G$ , and  $G_j$ , of a CF matrix grammar  $M$ , all in Chomsky Normal Form. This means that we can define an equivalent CF Kolam grammar  $M'$ , in which rules corresponding to those of  $G$  use only the  $\oplus$  operator, while rules corresponding to those of  $G_j$  use only the  $\ominus$  operator.

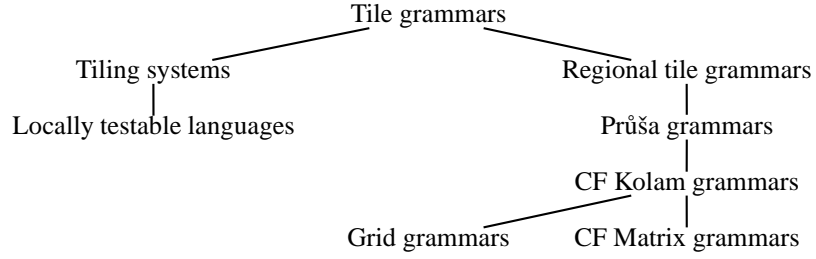
Also, it is easy to adapt classical string parsing methods to matrix grammars [22].

**Proposition 5.8.**  $\mathcal{L}(CFMG)$  and  $\mathcal{L}(GG)$  are incomparable.

PROOF. First, we know that by definition Grid grammars can generate only square pictures. On the other hand, it is impossible to define a CF matrix grammar generating infinitely many and only squares. This is because classical string pumping lemmata can be applied both to  $G$  (the ‘‘horizontal component’’ of the grammar), and to  $G_j$ ,  $1 \leq j \leq k$  (see e.g. [23]). Therefore the two language classes are incomparable.  $\square$

## 6. Summary

We finish with a synopsis of the previous language family inclusions, and a presentation of the constraints on the tile set of tile grammars corresponding to each class.



### *Průša grammars*

Průša grammars in Nonterminal Normal Form are regional tile grammars with the constraint that tiles used in right part of rules must not have one of these forms:

$$\begin{bmatrix} A & B \\ C & C \end{bmatrix}, \begin{bmatrix} A & C \\ B & C \end{bmatrix}, \begin{bmatrix} C & C \\ A & B \end{bmatrix}, \begin{bmatrix} C & A \\ C & B \end{bmatrix}$$

with  $A, B, C$  all different nonterminals. (See Proposition 5.4.)

### *CF Kolam grammars*

CF Kolam grammars in Chomsky Normal Form can be seen as regional tile grammars such that the tile-sets used in the right parts of rules must have one of the following forms:

$$\left[ \begin{array}{cccccc} \# & \# & \# & \# & \# & \# \\ \# & A & A & B & B & \# \\ \# & A & A & B & B & \# \\ \# & \# & \# & \# & \# & \# \end{array} \right], \left[ \begin{array}{cccc} \# & \# & \# & \# \\ \# & A & A & \# \\ \# & A & A & \# \\ \# & B & B & \# \\ \# & B & B & \# \\ \# & \# & \# & \# \end{array} \right]$$

with  $A \neq B$ . (See Proposition 5.2.) Clearly, this is also compatible with the constraint of Průša grammars.

### *Grid grammars*

For grid grammars in Nonterminal Normal Form, we have the same constraints on nonterminal rules as in CF Kolam grammars. Moreover, there is a different treatment of terminal rules of the grid grammar, i.e. rules like  $A \rightarrow t, t \in \Sigma$ . The corresponding regional tile grammar rules (still maintaining the CF Kolam grammars constraints) are used to generate from  $A$  square  $t$ -homogeneous pictures of any size, and are the following:

$$A \rightarrow \left[ \begin{array}{cccc} \# & \# & \# & \# \\ \# & A_1 & A_1 & \# \\ \# & A_1 & A_1 & \# \\ \# & A_2 & A_2 & \# \\ \# & \# & \# & \# \end{array} \right], A_1 \rightarrow \left[ \begin{array}{cccc} \# & \# & \# & \# \\ \# & A & A & A_3 \\ \# & A & A & A_3 \\ \# & \# & \# & \# \end{array} \right],$$

$$A_2 \rightarrow \left[ \begin{array}{cccccc} \# & \# & \# & \# & \# & \# \\ \# & A_4 & A_4 & A_5 & \# & \# \\ \# & \# & \# & \# & \# & \# \end{array} \right] \mid \left[ \begin{array}{ccc} \# & \# & \# \\ \# & A_5 & \# \\ \# & \# & \# \end{array} \right], \quad A_5 \rightarrow t.$$

$$A_3 \rightarrow \left[ \begin{array}{ccc} \# & \# & \# \\ \# & A_5 & \# \\ \# & A_3 & \# \\ \# & A_3 & \# \\ \# & \# & \# \end{array} \right] \mid \left[ \begin{array}{ccc} \# & \# & \# \\ \# & A_5 & \# \\ \# & \# & \# \end{array} \right],$$

with  $A_1, \dots, A_5$  all freshly introduced nonterminals. In practice, we are using the CF Kolam grammar rules corresponding to terminal rules of grid grammars of Proposition 5.6, translated into regional tile grammar rules following the construction of Proposition 5.2.

#### *CF matrix grammars*

Following the construction sketched in Proposition 5.7 for proving that CF matrix grammars define a subset of the class defined by CF Kolam grammars, we note that the constraints as for CF Kolam grammars apply. The added constraint is that if a nonterminal  $C$  is used as left part of a “horizontal” rule

$$C \rightarrow \left[ \begin{array}{cccccc} \# & \# & \# & \# & \# & \# \\ \# & A & A & B & B & \# \\ \# & A & A & B & B & \# \\ \# & \# & \# & \# & \# & \# \end{array} \right]$$

then it shall not be used as left part of a “vertical” rule

$$C \rightarrow \left[ \begin{array}{cccc} \# & \# & \# & \# \\ \# & A & A & \# \\ \# & A & A & \# \\ \# & B & B & \# \\ \# & B & B & \# \\ \# & \# & \# & \# \end{array} \right]$$

and vice versa. (This is a direct consequence of the informal considerations at the beginning of Section 5.4.1 and the proof of Proposition 5.2.)

From all that, regional tile grammars prove to be useful as a unifying, not overly general, concept for hitherto separated grammar models.

**Acknowledgments** We thank the anonymous referees for many suggestions, in particular the structure of proof of Proposition 4.3, and various improvements of the parsing algorithm.



## References

- [1] A. Cherubini, S. Crespi Reghizzi, M. Pradella, Regional languages and tiling: A unifying approach to picture grammars, in: *Mathematical Foundations of Computer Science (MFCS 2008)*, Vol. 5162 of *Lecture Notes in Computer Science*, Springer, 2008, pp. 253–264.
- [2] D. H. Younger, Recognition of context-free languages in time  $n^3$ , *Information and Control* 10 (2) (1967) 189–208.
- [3] S. Crespi Reghizzi, M. Pradella, Tile Rewriting Grammars and Picture Languages, *Theoretical Computer Science* 340 (2) (2005) 257–272.
- [4] A. Cherubini, S. Crespi Reghizzi, M. Pradella, P. San Pietro, Picture languages: Tiling systems versus tile rewriting grammars, *Theoretical Computer Science* 356 (1-2) (2006) 90–103.
- [5] D. Giammarresi, A. Restivo, Recognizable picture languages, *International Journal Pattern Recognition and Artificial Intelligence* 6 (2-3) (1992) 241–256, special Issue on *Parallel Image Processing*.
- [6] C. Allauzen, B. Durand, Tiling problems, in: E. Börger, E. Grädel, Y. Gurevich (Eds.), *The classical decision problem*, Springer-Verlag, 1997.
- [7] L. de Prophetis, S. Varricchio, Recognizability of rectangular pictures by Wang systems, *Journal of Automata, Languages and Combinatorics* 2 (4) (1997) 269–288.
- [8] D. Simplot, A characterization of recognizable picture languages by tilings by finite sets, *Theoretical Computer Science* 218 (1999) 297–323.
- [9] G. Siromoney, R. Siromoney, K. Krithivasan, Picture languages with array rewriting rules, *Information and Control* 23 (5) (1973) 447–470.
- [10] O. Matz, Regular expressions and context-free grammars for picture languages, in: *14th Annual Symposium on Theoretical Aspects of Computer Science*, Vol. 1200 of *Lecture Notes in Computer Science*, 1997, pp. 283–294.
- [11] D. Průša, *Two-dimensional Languages (PhD Thesis)*, Charles University, Faculty of Mathematics and Physics, Czech Republic, 2004.
- [12] F. Drewes, S. Ewert, R. Klempien-Hinrichs, H.-J. Kreowski, Computing raster images from grid picture grammars, *Journal of Automata, Languages and Combinatorics* 8 (3) (2003) 499–519.
- [13] G. Siromoney, R. Siromoney, K. Krithivasan, Abstract families of matrices and picture languages, *Computer Graphics and Image Processing* 1 (1972) 284–307.
- [14] D. Giammarresi, A. Restivo, Two-dimensional languages, in: A. Salomaa, G. Rozenberg (Eds.), *Handbook of Formal Languages*, Vol. 3, *Beyond Words*, Springer-Verlag, Berlin, 1997, pp. 215–267.

- [15] K. Lindgren, C. Moore, M. Nordahl, Complexity of two-dimensional patterns, *Journal of Statistical Physics* 91 (5-6) (1998) 909–951.
- [16] H. Lewis, Complexity of solvable cases of the decision problem for predicate calculus, in: *Proc. 19th Symposium on Foundations of Computer Science*, 1978, pp. 35–47.
- [17] M. A. Harrison, *Introduction to Formal Language Theory*, Addison Wesley, 1978.
- [18] S. Crespi Reghizzi, M. Pradella, A CKY parser for picture grammars, *Information Processing Letters* 105 (6) (2008) 213–217.
- [19] D. Průša, Two-dimensional context-free grammars, in: G. Andrejkova, S. Krajci (Eds.), *Proceedings of ITAT 2001*, 2001, pp. 27–40.
- [20] F. Drewes, Language theoretic and algorithmic properties of d-dimensional colages and patterns in a grid, *Journal of Computer and System Sciences* 53 (1) (1996) 33–66.
- [21] R. A. Finkel, J. L. Bentley, Quad trees: A data structure for retrieval on composite keys, *Acta Informatica* 4 (1974) 1–9.
- [22] V. Radhakrishnan, V. T. Chakaravarthy, K. Krithivasan, Pattern matching in matrix grammars, *Journal of Automata, Languages and Combinatorics* 3 (1) (1998) 59–72.
- [23] M. Nivat, A. Saoudi, V. R. Dare, Parallel generation of finite images, *International Journal Pattern Recognition and Artificial Intelligence* 3 (3-4) (1989) 279–294.