
Tile-rewriting grammars for picture languages and associated parsing techniques



PhD Minor research

Student : Daniele Paolo Scarpazza
Affiliation : Politecnico di Milano
Advisor : Prof. Stefano Crespi Reghizzi
Date : February 16th, 2005

In this presentation:

- 1) What are pictures languages
- 2) What are Tile-Rewriting Grammars
- 3) We have a polynomial parsing technique for Tile-Rewriting Grammars!

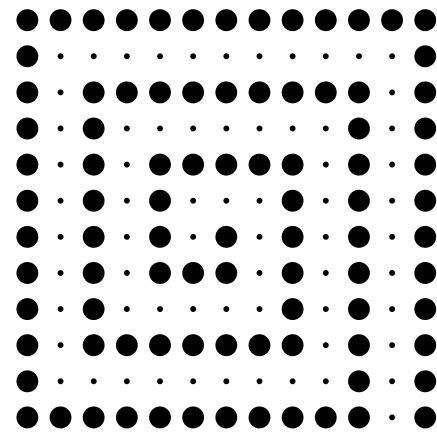
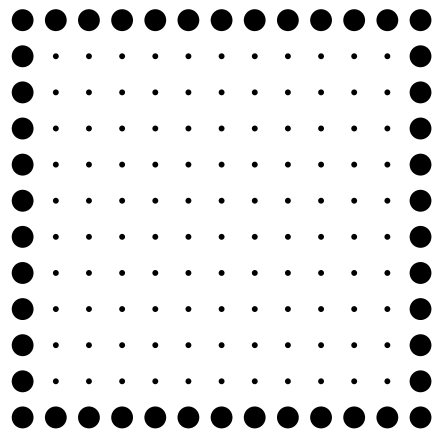
In this presentation:

- 1) What are pictures languages
- 2) What are Tile-Rewriting Grammars
- 3) We have a polynomial parsing technique for Tile-Rewriting Grammars!

Picture

Given a finite alphabet Σ , a *picture* over Σ is a rectangular array of elements of Σ .

Example: $\Sigma = \{\bullet, \cdot\}$



We call Σ^{**} the set of pictures is over Σ .

For $h, k \geq 1$, $\Sigma^{(h,k)}$ denotes the set of pictures of size (h, k) .

We will use the notation $|p| = (h, k)$, $|p|_{row} = h$, $|p|_{col} = k$.

A *pixel* is an element $p(i, j)$.

If all pixels are identical to $C \in \Sigma$ the picture is called *homogeneous* and denoted as C -picture.

Subpicture

Let p and q be pictures. q can be a *subpicture* of p at position (i, j) , and we write:

$$q \trianglelefteq_{(i,j)} p.$$

Example: if $p = \begin{pmatrix} a & d & g & j & m \\ b & e & h & k & n \\ c & f & i & l & o \end{pmatrix}$ and $q = \begin{pmatrix} e & h & k \\ f & i & l \end{pmatrix}$, then: $q \trianglelefteq_{(2,2)} p$.

Substitution

If p, q, q' are pictures, $q \trianglelefteq_{(i,j)} p$, and q, q' have the same size, then

$$p[q'/q]_{(i,j)}$$

is the picture obtained by replacing in p the occurrence of q at (i, j) with q' .

$$\text{If } p = \begin{pmatrix} a & d & g & j & m \\ b & e & h & k & n \\ c & f & i & l & o \end{pmatrix}, \quad q = \begin{pmatrix} e & h & k \\ f & i & l \end{pmatrix}, \quad q' = \begin{pmatrix} Z & Z & Z \\ Z & Z & Z \end{pmatrix},$$

then:

$$p[q'/q]_{(2,2)} = \begin{pmatrix} a & d & g & j & m \\ b & Z & Z & Z & n \\ c & Z & Z & Z & o \end{pmatrix}.$$

Coordinates

A *coordinate* is a couple of positive integers.

If $q \trianglelefteq_{(i,j)} p$, we call $coor_{(i,j)}(q, p)$
the set of coordinates in p where q is located.

$$p = \begin{pmatrix} a & d & g & j & m \\ b & e & h & k & n \\ c & f & i & l & o \end{pmatrix}, \quad q = \begin{pmatrix} e & h & k \\ f & i & l \end{pmatrix},$$

$$coor_{(2,2)}(q, p) = \{(2, 2), (2, 3), (2, 4), (3, 2), (3, 3), (3, 4)\}.$$

Rectangle

Intuitively, we call *rectangle* a set of coordinates of a rectangular area.

We write rectangles in the following form $r \boxtimes c \boxplus (i, j)$.

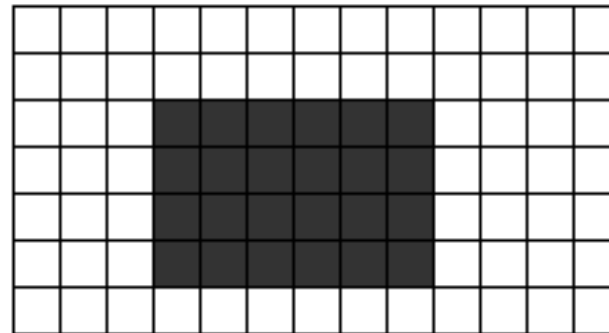
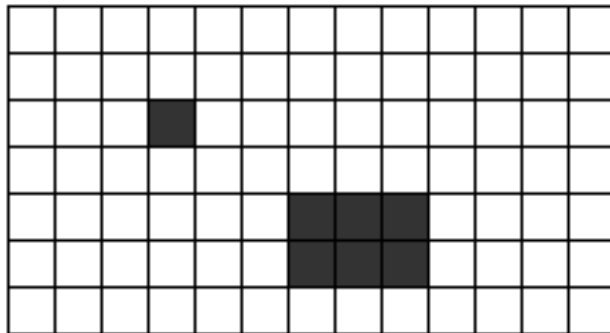
That rectangle is r rows high, c columns wide and starts at position (i, j) .

$$p = \begin{pmatrix} a & d & g & j & m \\ b & e & h & k & n \\ c & f & i & l & o \end{pmatrix}, \quad q = \begin{pmatrix} e & h & k \\ f & i & l \end{pmatrix},$$

$$\text{coord}_{(2,2)}(q, p) = \{(2, 2), (2, 3), (2, 4), (3, 2), (3, 3), (3, 4)\} = 2 \boxtimes 3 \boxplus (2, 2)$$

Ceiling

Given a set of coordinates C , the *ceiling* of C , denoted as $\lceil C \rceil$, is the smallest rectangle which is either a superset or equal to C .



$$\begin{aligned} & \lceil (1 \boxtimes 1 \boxplus (3, 4)) \cup (2 \boxtimes 3 \boxplus (5, 7)) \rceil = \\ & = \lceil \{(3, 4), (5, 7), (5, 8), (5, 9), (6, 7), (6, 8), (6, 9)\} \rceil = \\ & = 4 \boxtimes 6 \boxplus (3, 4) \end{aligned}$$

Locally testable language (LOC).

Given a finite set of tiles $\omega = \{t_1, t_2, \dots\} \subseteq \Sigma^{(i,j)}$,

$LOC(\omega)$ is the set of those pictures which use all the tiles in ω at least once.

$$\omega = \left\{ \begin{array}{cc} A & A & A & B & B & B \\ A & A & ' & A & B & ' & B & B \end{array} \right\},$$

$$\begin{array}{cc} A & A & B & B & B \\ A & A & B & B & B \end{array} \in LOC(\omega)$$

$$\begin{array}{cc} A & A & A & B & B \\ A & A & B & B & B \end{array} \notin LOC(\omega), \quad \begin{array}{cc} A & A & A & B \\ A & A & A & B \end{array} \notin LOC(\omega)$$

In this presentation:

- 1) What are pictures languages
- 2) What are Tile-Rewriting Grammars
- 3) We have a polynomial parsing technique for Tile-Rewriting Grammars!

Tile Rewriting Grammar (TRG)

It is a tuple (Σ, N, S, R) , where Σ is the *terminal* alphabet, N is a set of *nonterminal* symbols, $S \in N$ is the *starting symbol*, R is a set of *rules*. R may contain two kinds of rules:

Fixed size: $A \rightarrow t$

Variable size: $A \rightarrow \omega$

A fixed size rule rewrites a A -homogeneous subpicture as t . A variable size rule rewrites a A -homogeneous subpicture as one of the pictures in $LOC(\omega)$.

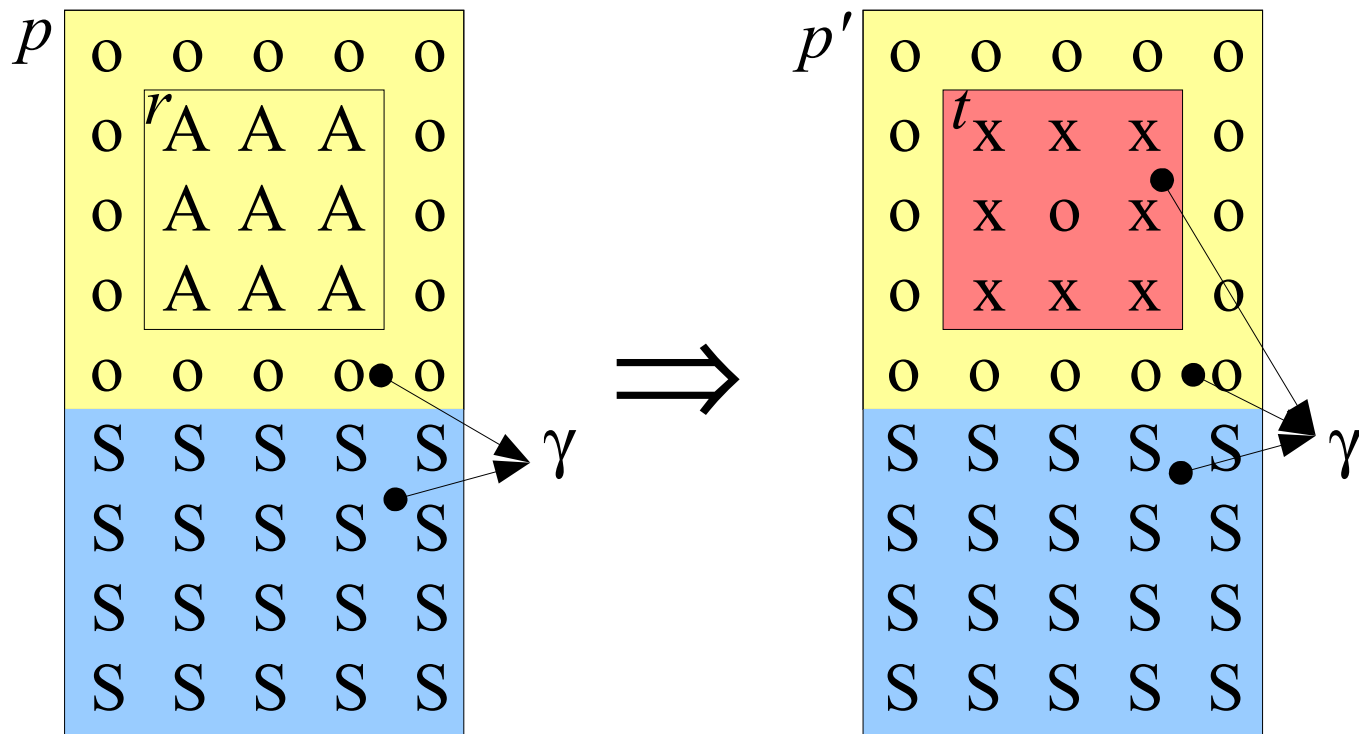
Fixed size rules are not a special case of variable size rules.

Equivalence relation, Maximal subpicture

Let γ be an equivalence relation on $coord(p)$, written $(x, y) \sim (x', y')$.

Two subpictures q and q' are equivalent with respect to γ iff their coordinates are equivalent: $(x, y) \sim (x', y')$.

A homogeneous C -subpicture $q \trianglelefteq p$ is *maximal with respect to* γ iff every equivalent C -subpicture q' is completely included in q or does not overlap with q .



Derivation in one step $(p, \gamma) \Rightarrow_G (p', \gamma')$:

- there is a rule $A \rightarrow t$ or $A \rightarrow \omega$ in grammar G ;
- there is an A -homogeneous subpicture $r \trianglelefteq p$, maximal with respect to γ
- p' is obtained substituting r with picture t , i.e. $p' = p[t/r]$
- in case of variable size rule $t \in LOC(\omega)$
- γ' is equal γ except for the eq. class containing $z = \text{coord}_{(i,j)}(r, p)$, split into z and its complement w.r.t. its equivalence class.

The subpicture r is named the *application area* in the derivation step.

Derivation in n steps is a trivial extension.

The *picture language* defined by a grammar G (written $L(G)$) is the set of $p \in \Sigma^{**}$ such that, if $|p| = (h, k)$, then

$$\left(S^{(h,k)}, \text{coord}(p) \times \text{coord}(p) \right) \xRightarrow{*}_G (p, \gamma) \quad (1)$$

where γ is arbitrary.

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

Example: Chinese boxes.

$G = (\Sigma, N, S, R)$, where $\Sigma = \{\ulcorner, \urcorner, \llcorner, \lrcorner, \circ\}$, $N = \{S\}$, and R consists of the following rules:

$$S \rightarrow \begin{array}{cc} \ulcorner & \urcorner \\ \llcorner & \lrcorner \end{array};$$

$$S \rightarrow \left\{ \begin{array}{cc} \ulcorner & \circ \\ \circ & S \end{array}, \begin{array}{cc} \circ & S \\ \llcorner & \circ \end{array}, \begin{array}{cc} \circ & S \\ \circ & S \end{array}, \begin{array}{cc} S & S \\ \circ & \circ \end{array}, \begin{array}{cc} \circ & \circ \\ S & S \end{array}, \begin{array}{cc} S & S \\ S & S \end{array}, \begin{array}{cc} \circ & \urcorner \\ S & \circ \end{array}, \begin{array}{cc} S & \circ \\ S & \circ \end{array}, \begin{array}{cc} S & \circ \\ \circ & \lrcorner \end{array} \right\}$$

Example picture in $L(G)$:

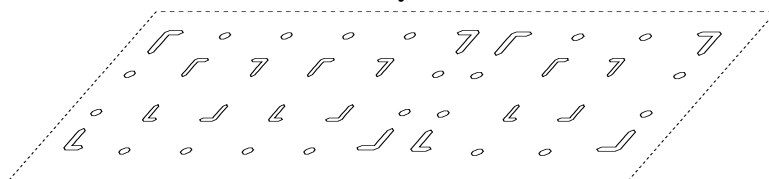
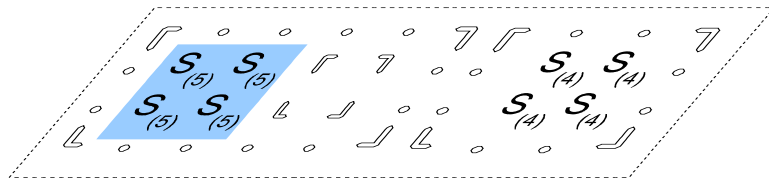
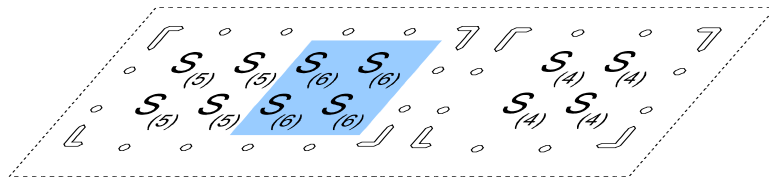
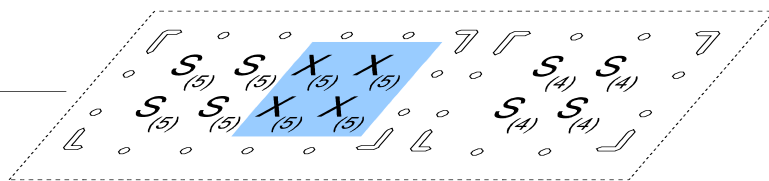
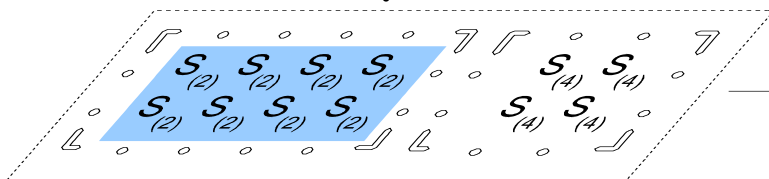
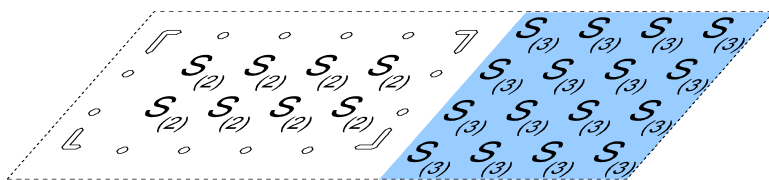
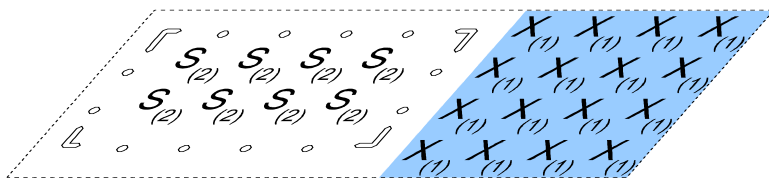
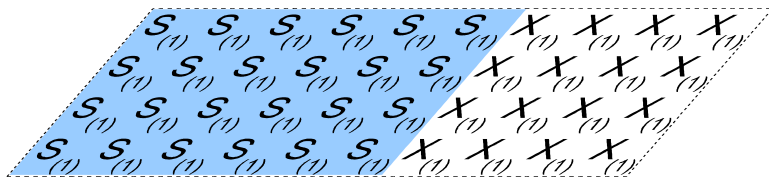
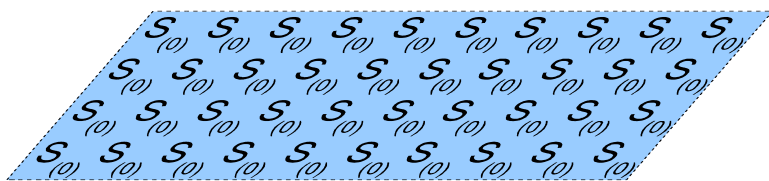
\ulcorner	\circ	\circ	\circ	\circ	\urcorner
\circ	\ulcorner	\circ	\circ	\urcorner	\circ
\circ	\circ	\ulcorner	\urcorner	\circ	\circ
\circ	\circ	\llcorner	\lrcorner	\circ	\circ
\circ	\llcorner	\circ	\circ	\lrcorner	\circ
\llcorner	\circ	\circ	\circ	\circ	\lrcorner

For convenience, we will often specify a set of tiles by a sample picture exhibiting the tiles as its subpictures.

We write $|$ to separate alternative right parts of rules.

The previous grammar becomes:

$$S \rightarrow \begin{array}{cc} \ulcorner & \urcorner \\ \llcorner & \lrcorner \end{array} \mid B_{2,2} \left(\begin{array}{cccc} \ulcorner & \circ & \circ & \urcorner \\ \circ & S & S & \circ \\ \circ & S & S & \circ \\ \llcorner & \circ & \circ & \lrcorner \end{array} \right)$$



In this presentation:

- 1) What are pictures languages
- 2) What are Tile-Rewriting Grammars
- 3) We have a polynomial parsing technique for Tile-Rewriting Grammars!

In this presentation:

- 1) What are pictures languages
- 2) What are Tile-Rewriting Grammars
- 3) We have a polynomial parsing technique for Tile-Rewriting Grammars!