ISSN1000-9825 Journal of Software 软件学报 2000,11(7):871～880

# Grid Automata and Grid Grammars for Picture Languages*

SHEN En-shao

(Department of Computer Science and Engineering Shanghai Jiaotong University Shanghai 200030)
E-mail: esshen@mail.sjtu.edu.cn

**Abstract** A two-dimensional grid grammar was designed to fill up the missing ring in the Equivalence Theorem for recognizable picture languages (REC), summarized in a survey paper by Giammarresi and Restivo. Instead of 2-dimensional on-line tessellation automata, grid automata were introduced, which were closer to the traditional binary tree automata, to bridge the grid grammar and other approaches of describing the class of REC. Meanwhile the standard (existential) monadic second order logic was substituted by a weaker logic framework: positive monadic partition logic. A new and complete version of Equivalence Theorem for REC is presented.

**Key words** Formal language, picture language, grammar, automata, restricted second-order logic.

A natural next step in the evolution of the theory of automata over words, trees, and Mazurkiewiz traces as well, is to study finite-state automata over partial orders or directed acyclic graphs (with bounded degree, in short dag's)[1,2]. Among them the topic of two-dimensional rectangular arrays or pictures has attracted much attention[3]. They inherit unusual (w.r.t. the classical theory) while typical features of general dag's, such as the separation of non-determinism from determinism, unclosedness of complement operation, undecidability of empty problem, the inequality of expressive power between first-order definability and star-free expressions, the non-collapsing of monadic second order logic (MSO) down to existential monadic second order logic (EMSO) (infinite hierarchy exists), etc.[2,4,5]. However, their special configuration, which in fact does not constrain themselves much in applications, would facilitate further explorations. Giammarresi and Restivo summarized in Ref. [3] some results up to date and offered an Equivalence Theorem for the recognizable picture languages (REC), reflecting the robustness of this notion. It is shown that the following approaches have the same expressive power:

- 2-dimensional on-line tessellation automata (OTLA, a special type of cellular automata) acceptance;
- (2×2) tiling systems recognizability;
- EMSO definability;
- projection-closed complementation-free regular expressions (PCFRE).

Compared with those in classical situations, obviously there is a missing ring: the array grammars which generate exactly REC. The available 2-dimensional ones such as Siromoney's matrix grammars (SMG), Nivat-Saoudi's image grammars etc., which were initially raised in the problems concerning pattern recognition and image processing[6,7], fail to accomplish this goal. We shall introduce, in this paper, two types of grid automata

* This research is supported by the National Natural Science Foundation of China(国家自然科学基金,No. 69673009).
**SHEN En-shao** was born in 1947. He is a professor at Department of Computer Science, Shanghai Jiaotong University. His current research areas include logic in computer science such as formal languages & automata theory, model-theoretic logic, finite model theory & descriptive complexity, program verification & model checking.

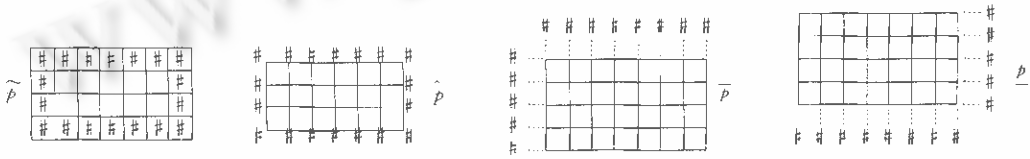
Manuscript received 1998-09-17, accepted 1999-06-22.

(special kind of $S2S$, systems with two successors), and their dual generators called grid grammars. The latter are just the right objects filling above gap. We conclude finally a new version of Equivalence Theorem. These results expose the intimate relationships between different kinds of computing paradigms: Implementation (by automata or grammars) and formal specification (by logical formulae or algebraic expressions); Local operational semantics and global denotational ones; Recognizers (automata or tiling systems) and generators (grammars or re-writing rules).

## 1 Preliminary Notions and Backgrounds

• The idea of looking tiles as transition moves over dag's and tiling systems over dag's as graph recognizers was first introduced by W. Thomas[8], and later adopted widely[3].

• Picture is a labelled rectangular array, denoted by $p$. The equivalence of expressivity between $2\times 2$ tiling systems and $1\times 1$ stencil* systems can be found in Ref. [10]. The underlying idea is simple: just a transformation from cells into vertices and cell-based pictures into vertex-based pictures or grids. In the following we shall follow the stencil based approach. Let $\hat{p}$ denote the extended picture of $p$ by surrounding boundary markers #, and $\tilde{p}$ is the corresponding cell based version as in Ref. [3,11]. $\overline{p}$ and $\underline{p}$ denote partially extended pictures of $p$ respectively, as follows.



• The notion of monadic partition logic and its applications in automata over strings and trees are well-known, while the potential merit of introducing the weaker framework $L(MP)$ instead of the standard MSO can be seen in Refs. [10,12]. Roughly speaking the monadic n-partition quantifier (Ebbinghause quantifier) means the existence of an n-partition of the universe such that the property defined by the formula in the scope of this quantifier is homogeneously true over all partition subsets, i.e. independent of the choice of elements in each partition subset. So it is a special kind of existential monadic second-order quantifier. $L^+(MP)$ is the positive fragment of $L(MP)$ which is a proper sublogic of EMSO. In fact we need only one positive occurrence of partition quantifier.

## 2 Grid Automata

The notion of grid automata was primarily motivated by a local analysis of stencils viewed as transition in stencil systems, and by emphasizing the connection to binary tree automata. While the starting point of most studies on picture languages is based on the analogy to word automata and string languages[13,11,13,17], our motto is,

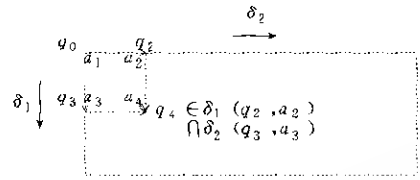
Grid automata $=S2S+$COMMUtativity (a parallel control), where $S2S$ is automata over binary trees.

**Definition 1.** $\mathcal{A}=(\Sigma,Q,q_0,\delta_1,\delta_2,F)$, where each component is the same as binary tree automaton (top-down version) except that $\delta_i(i=1,2)$ are controlled by concurrent constraints, and it runs over $\underline{p}$'s. We introduce a kind of execution control, explaining how to compose compatible transitions (to be moves via $\delta_i$) during the operation. In the following figures the dash arrows represent the given transitions.

* "stencil" represents $1\times 1$ tile. This special name follows from Ref. [9].

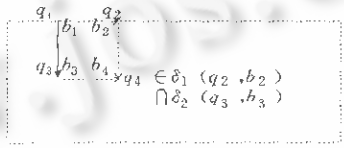
1. Initial moves

For $a_1,a_2,a_3,a_4\in\Sigma$, $q_1\in Q$, if $q_1=q_0$ and there are some $q_2\in\delta_2(q_0,a_1)$ and $q_3\in\delta_1(q_0,a_1)$ such that $\delta_1(q_2,a_2)\cap\delta_2(q_3,a_3)\neq\varnothing$, then the square block in the following figure is a legal initial move.



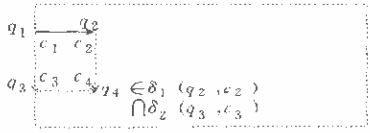

2. Top moves

For $b_1,b_2,b_3,b_4\in\Sigma$ and $q_1,q_3\in Q$ with $q_3\in\delta_1(q_1,b_1)$, if there is some $q_2\in\delta_2(q_1,b_1)$ such that $\delta_2(q_3,b_3)\cap\delta_1(q_1,b_2)\neq\varnothing$, then the showed block is a legal top move.



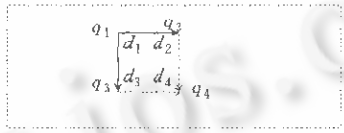

3. Left moves

For $c_1,c_2,c_3,c_4\in\Sigma$, $q_1,q_2\in Q$ with $q_2\in\delta_2(q_1,c_1)$, if there is some $q_3\in\delta_1(q_1,c_1)$ such that $\delta_1(q_2,c_2)\cap\delta_2(q_3,c_3)\neq\varnothing$, then the showed block is a legal left move.



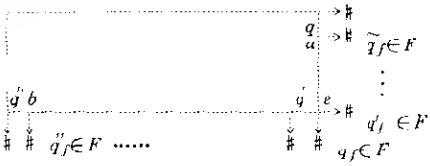

4. Inner moves

For $d_1,d_2,d_3,d_4\in\Sigma$, $q_1,q_2,q_3\in Q$ with $q_2\in\delta_2(q_1,d_1)$, $q_3\in\delta_1(q_1,d_1)$, if there is $q_4\in\delta_1(q_2,d_2)\cap\delta_2(q_3,d_3)$, then the showed block is a legal inner move.




5. Boundary conditions

For right side positions: $\delta_2(q,a)\cap F\neq\varnothing$. For bottom positions: $\delta_1(q'',b)\cap F\neq\varnothing$. Particularly for target (bottom right) position, $\delta_1(q',e)\cap F\neq\varnothing\neq\delta_2(q',e)\cap F$.




A accepts a given picture $p$ of size $m\times n$ iff there is a successful run $\rho:dom(\underline{p})\to Q$ satisfying the following.

- $\rho((1,1))=q_0,\rho((i,n+1))\in F\ni\rho((m+1,j))$, for $1\leqslant i\leqslant m,1\leqslant j\leqslant n$.
- All blocks in the extended graph $(\underline{p},\rho)$ match with the control constraints.

The execution orders of the moves in operation are not unique. Among them the following three choices are of interest:

- diagonal order;
- vertical scanning (from top down and then shift to next right column...);
- horizontal scanning.

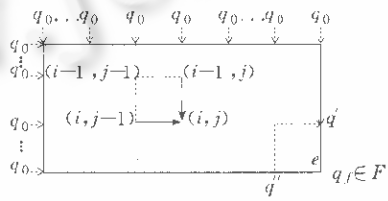
The latter two seem to be more natural and practical.

We call $A$ together with the commutative or concurrent controls (including initial and boundary conditions) a grid automaton of type I.

The different faces about above block traditions (moves) of type I, which simulate directly stencils with some pre-contiguous information, are not substantial. They can be forged into one combination-pattern of two local sequential successors, in type II of grid automata running over $\bar{p}$'s.

**Definition 2.** Let $A=(\Sigma,q_0,Q,\bar{\delta}_1,\bar{\delta}_2,F)$, $\bar{p}$ be an extended picture with label function $l:dom(\bar{p})\to\Sigma\cup\{\#\}$. $A$ accepts $p$ of size $m\times n$ iff there is a run $\bar{\rho}:dom(\bar{p})\to Q$, such that

- $\bar{\rho}((0,j))=q_0=\bar{\rho}((i,0))$ $(1\leqslant i\leqslant m,1\leqslant j\leqslant n)$, and $\bar{\rho}((m,n))\in F$;
- for each proper position $(1\leqslant i\leqslant m,1\leqslant j\leqslant n)$,
  suppose $q_1=\bar{\rho}((i-1,j))$, $q_2=\bar{\rho}((i,j-1))$ and $l((i,j))=a$, then $\bar{\rho}((i,j))=q\in\bar{\delta}_1(q_1,a)\cap\bar{\delta}_2(q_2,a)$.




Call $A$ a grid automaton of type II, in which $\bar{\delta}_i$ is similar to that of leaf-to-root (i.e. bottom up) tree automata except control mechanism.

In principle each block move of type I can be simulated by four related compound transitions of type II. So for each $A_I$ we can construct an $A_{II}$ (construction omitted) s.t. $L^2(A_I)=L^2(A_{II})$. The converse direction is not so obvious along this approach of local simulation. However, if we notice that the connection between two types of grid automata is almost the same as that between two versions (top-down and bottom-up) of tree automata, the equivalence of the expressivity between the two types of grid automata is immediately clear. Later Section 4 will give a more formal but indirect proof.

Note. (i) In the definitions above $q_0$ can be replaced by a set $I_0$ of initial states, without injuring the expressive power as a whole. So can the starting symbols in later grid grammar. This observation is useful in the arguments above as well as later. (ii) Concerning the way of composing local successors the grid automata of type II are similar to Giammarresi-Restivo's revision of 2-dimensional OTLA (see Ref. [3], but there is only one $\delta$). The separation of one $\delta$ into two $\delta_i(i=1,2)$ is crucial in the transition from grid recognizers to grid generators. On the other hand the grid automata of type I are closer to the stencil systems.

## 3 Grid Grammars

The insight of grid grammar was inspired by that of grid automata*, together with some techniques for easier description. A grid grammar consists of a production system $G$, whose form is similar to that of SMG (without intermediate variables), and concurrent constraints (relatively stable independent of $G$) to control the combination of horizontal and vertical production rules, which contain some traces back to Nivat-Saoudi's

* Its control mechanism inherits from type II while its description pattern is similar to that of type I.

table-control[7]. But here we further set the "table-control" free in a symmetric and real parallel way.

**Definition 3.** $G=(\Sigma,V_h,V_v,S_h,S_v,R_h,R_v)$, where

- $\Sigma$, a finite set of terminals;
- $V_h(V_v)$, finite set of variables along horizontal (vertical) direction;
- $S_h\in V_h(S_v\in V_v)$, the starting symbols for $h$-($v$-) direction;
- $R_h(R_v)$, finite set of production rules for two directions respectively. In "regular" case the rules are right linear, such as in forms:

$S_h\to aY$, $Y'\to bY''$, $\overline{Y}\to c$ in $R_h$;

$S_v\to aX$, $X'\to bX''$, $\overline{X}\to c$ in $R_v$.

In this paper we shall focus on such regular rules.

The class of grid grammars (or automata) will be denoted by $GG$ (or $GA$) or simply $G$. To describe the control pattern we introduce some auxiliary symbols:

- an endmark $E$, to replace $X(Y)\to a$ by $X(Y)\to aE$, and denote the corresponding collection as $R_v^*$ ($R_h^*$);
- pairs of numbers $(i,j)\in N\times N$ as a 2-dimensional counter, to record or indicate the coordinates of location where related terminal is just generated.

1. Initial productive moves

$$\begin{array}{c} S_v \\ \downarrow \\ S_h\to(a;1,1)(X,Y) \end{array} \Longleftrightarrow \text{ ex. } \begin{array}{l} S_v\to aX\in R_v \\ S_h\to aY\in R_h \end{array} \quad \text{for some } a\in\Sigma$$

Note if there is no common $a\in\Sigma$ occurring in the initial production rules in $R_v$ and $R_h$ respectively, then $L(G)=\varnothing$.

2. Top production moves

$$\begin{array}{c} \text{(given)} \\ (a;1,j)(X,Y) \end{array} \to \begin{array}{c} S_v \\ \downarrow \\ (a';1,j+1)(X',Y') \end{array} \Longleftrightarrow \text{ ex. } \begin{array}{l} S_v\to a'X'\in R_v \\ Y\to a'Y'\in R_h \end{array} \quad (a'\in\Sigma)$$

Particularly when $Y'=E$, the $h$-derivation rightwards should stop.

3. Left production moves

$$\begin{array}{c} \text{(given) } (a';i,1)(X,Y) \\ \downarrow \\ S_h\to(a';i+1,1)(X',Y') \end{array} \Longleftrightarrow \text{ ex. } \begin{array}{l} X\to a'X'\in R_v \\ S_h\to a'Y'\in R_h \end{array} \quad (a'\in\Sigma)$$

Particularly when $X'=E$, the $v$-derivation downwards should stop.

4. Inner (including final) production moves

$$\begin{array}{c} \text{(given)} \\ (b;i+1,j)(X',Y') \end{array} \to \begin{array}{c} \text{(given)}(a;i,j+1)(X,Y) \\ \downarrow \\ (c;i+1,j+1)(X'',Y'') \end{array} \Longleftrightarrow \text{ ex. } \begin{array}{l} X\to cX''\in R_v \\ Y\to cY''\in R_h \end{array} \quad (c\in\Sigma)$$

with the restrictions that $Y=E\Rightarrow Y''=E$, $X'=E\Rightarrow X''=E$. On other cases $Y$ and $Y''$, $X'$ and $X''$ are independent of each other. When $Y=E=X'$ it's a final production move and the whole derivation or generating process finishes off.

Note. (i) The role of 2-dimensional variables and endmark together with right-bottom control ensure that the $G$-generated graphs are rectangles or pictures. However these coordinated variables are re-written pairwisely along each direction (except the right-bottom control), in order to get sufficient freedom. The coordinated terminals and variables make it unnecessary to write down all symbols generated up to now during each re-writing. These apparatus can also be applied in case of 3 dimension.

(ii) The derivation order, similar to that of grid automata, has choices of horizontal and vertical scanning.

(iii) The notion of grid grammars is in fact a dual form of grid automata of type I (even in view of accepting vs. generating). Nevertheless this relation is obscured somewhat by the outlook of concurrent control mechanism. The introduction of type II automata clears the ambiguity up.

(iv) Observe the algebraic operation $\oplus$ (a kind of $h$-$v$ combination) of word languages. Grid grammar is an operational semantics for this structural characterization of 2-dimensional recognizable picture languages (REC)[3].

## 4 A New Version of Equivalence Theorem for REC

**Theorem 1.** The following items are equivalent:

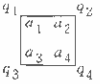
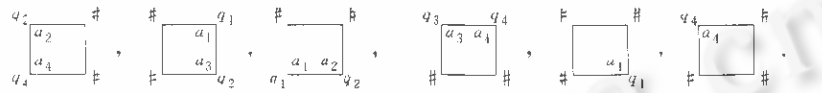
1. Acceptability by grid automata (types I, II); $(GA_I, GA_{II})$
2. Recognizability by stencil systems; $(SS$, similarly $TS$ for tiling system's recognizability$)$
3. $L^1(MP)$-definability;
4. Projection closed, complement-free regular expressability; $(PCFRE)$
5. Derivability by grid grammars, $(GG)$

### 4.1 Proof. 1(*I*)⇔2

First we note that without projection $(S2S+COMM)$ is not equivalent to pure stencil system $\Delta^{(1)}$, the former is weaker. (Counter example exists.) However under projection they have the same recognizability. *

$\Delta^{(1)}$ can simulate $(S2S+COMM)$'s operation, i.e. for each grid automaton (type I) we can construct a corresponding stencil system such that it accepts or recognizes the same set of pictures.

Let $A=(\Sigma, Q, q_0, \delta_1, \delta_2, F)$.

Each legal block move of $A$, say  , constitutes a $1\times1$ stencil. Put it into $\Delta^{(1)}$, together with boundary stencils:  .

Now we construct the stencil system $T=(\Sigma, Q, \Delta^{(1)}, \pi)$, in short $(\Delta^{(1)}, \pi)$, where $\Delta^{(1)}$ is over $Q\times(\Sigma\cup\{\#\})$, $\pi: Q\times\Sigma\to\Sigma$. It's easy to verify the following.

$A$ has a successful run on $\underline{p}$ iff $p$ has an extended expansion $(\hat{p}, \rho)$ over $Q\times(\Sigma\cup\{\#\})$ such that $S((\hat{p}, \rho))\subseteq\Delta^{(1)}$, where $S((\hat{p}, \rho))$ denotes the set of all $1\times1$ subgrids of $(\hat{p}, \rho)$. Hence we have Claim $L(GA_I)\subseteq L(SS)$, $SS$ for stencil systems.

Conversely, given a stencil system $T=(\Delta^{(1)}, \pi)$, we construct a corresponding grid automaton $A=(Q, \Sigma, I, \delta_1, \delta_2, F)$ of type I as follows:

Set $Q=\Delta^{(1)}$ minus boundary stencils (i.e. those containing #.), (called the set of generalized states)

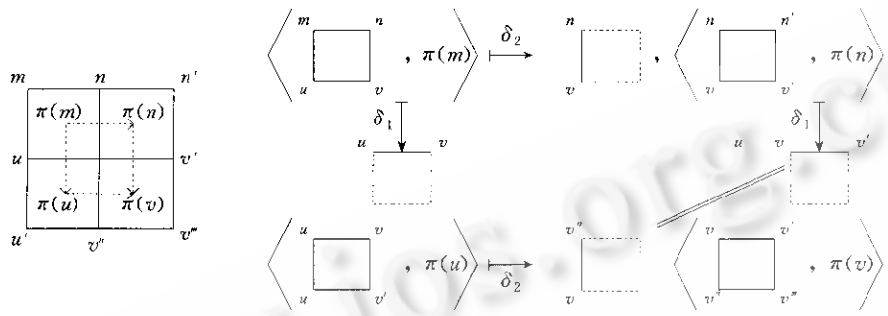
$I=\left\{\begin{array}{|c|}\hline q_0\\ \hline a\\ \hline\end{array}\ \Delta^{(1)}\ \middle|\ \begin{array}{|c|c|}\hline \# & \# \\ \hline a & \# \\ \hline \# & q_0 \\ \hline\end{array}\ \in\Delta^{(1)}\right\}$, $\begin{array}{|c|}\hline q_0\\ \hline a\\ \hline\end{array}$ for those stencils with $(a, q_0)$ as their top-left vertex-labels.

* The situation is quite similar to the relation between Domino systems and tiling systems, see Ref. [16].

$$F=\left\{\begin{array}{|c|}\hline q\\ \hline a\\ \hline q\\ \hline\end{array}\in\Delta^{(1)}\ \middle|\ \begin{array}{|ccc|}\hline \# & q & \# \\ \hline \# & a & \# \\ \hline \# & \# & \# \\ \hline\end{array}\in\Delta^{(1)}\right\}.$$

The idea of defining two transition relations is that contiguous generalized states (via $\delta_i$) simulate the neighbouring stencils in "local successful" tilings of $\Delta^{(1)}$. The definition is showed in the following figures.

For simplicity let $m,n,u,v,\ldots\in\Gamma=Q\times\Sigma$, $a_m=\pi(m)\in\Sigma$, similar for $a_n,a_u,a_v$ etc.



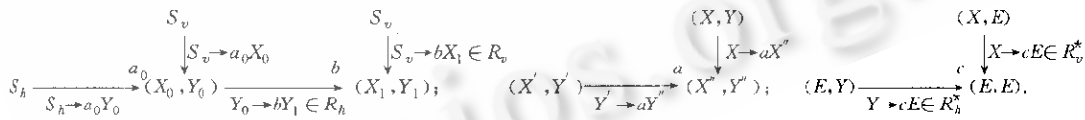

For $\begin{array}{|cc|}\hline m & n\\ u & v\\ \hline\end{array}\in I$, if $\begin{array}{|ccc|}\hline \# & \# & \# \\ \hline \# & m & n \\ \hline\end{array}$, $\begin{array}{|ccc|}\hline \# & \# & \# \\ \hline n & \# & n' \\ \hline\end{array}$, $\begin{array}{|ccc|}\hline \# & \# & \# \\ \hline \# & \# & \# \\ \hline u & \# & \# \\ \hline\end{array}$, $\begin{array}{|ccc|}\hline \# & \# & \# \\ \hline \# & \# & \# \\ \hline \# & u' & \# \\ \hline\end{array}\in\Delta^{(1)}$ then the above block move can be set into an initial move of $A$; for $\begin{array}{|cc|}\hline m & n\\ u & v\\ \hline\end{array}$, $\begin{array}{|cc|}\hline n & n'\\ v & v'\\ \hline\end{array}$, if $\begin{array}{|ccc|}\hline \# & \# & \# \\ \hline m & n & \# \\ \hline\end{array}$, $\begin{array}{|ccc|}\hline \# & \# & \# \\ \hline n & n' & \# \\ \hline\end{array}\in\Delta^{(1)}$, then the above block move can be arranged into a top move of $A$.

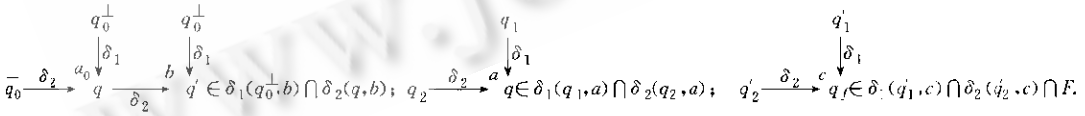
For a fixed picture $p\in\Sigma^{**}$ it's easy to verify that from an accepting tiling of $(\Delta^{(1)},\pi)$ over $p$ we can obtain a successful run of $A$ over $p$, and vice verse. (Using scanning order)

Claim. $L(SS)\subseteq L(GA_I)$

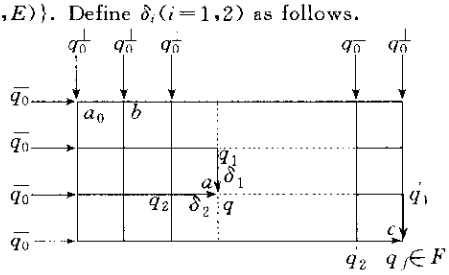
### 4.2 $5\leqslant 1(II)$

For each grid grammar $G$ we construct a grid automaton $A_G$ of type II such that $L(G)=L(A_G)$. The idea is showed in the following figures.




Given $G=(\Sigma,V_h,V_v,S_h,S_v,R_h,R_v)$, construct $A_G=(\Sigma,Q,I,\delta_1,\delta_2,F)$, where $Q=(V_h\cup\{E\})\times(V_v\cup\{E\})\cup\{S_h,S_v\}$, $I=\{S_h,S_v\}$, $F=\{(E,E)\}$. Define $\delta_i(i=1,2)$ as follows.



Suppose $X\in V_v$, $\overline{X}\in\overline{V}_v=V_v\cup\{E\}$, similarly for $Y$, $\overline{Y}$, etc.; $a,b,c\in\Sigma$; $R_v^*$ ($R_h^*$) as modification of $R_v$($R_h$) aforementioned.

Whenever $S_v\to a_0X_0\in R_v^*$ and $S_h\to a_0Y_0\in R_h^*$, put $(S_v,a_0;(\overline{X}_0,\overline{Y}_0))$ in $\delta_1$, $(S_h,a_0;(\overline{X}_0,Y_0))$ in $\delta_2$.

Whenever $S_v\to b\overline{X}_1\in R_v^*$ and $Y\to b\overline{Y}_1\in R_h^*$, put $(S_v,b;(\overline{X}_1,\overline{Y}_1))$ in $\delta_1$, and $((\overline{X},Y),b;(\overline{X}_1,\overline{Y}_1))$ in $\delta_2$, for all $\overline{X}$.

Whenever $X\to b\overline{X}_1\in R_v^*$ and $S_h\to b\overline{Y}_1\in R_h^*$, put $((X,\overline{Y}),b;(\overline{X}_1,Y_1))$ in $\delta_1$, $(S_h,b;(\overline{X}_1,\overline{Y}_1))$ in $\delta_2$, for all $\overline{Y}$.

Whenever $X\to a\overline{X}''\in R_v^*$ and $Y'\to a\overline{Y}''\in R_h^*$, put $((X,\overline{Y}),a;(\overline{X}'',Y''))$ in $\delta_1$, and $((\overline{X}',Y'),a;(\overline{X}'',Y''))$ in $\delta_2$, for all $\overline{Y}$ and all $\overline{X}'$, with the proviso that

$$\overline{Y}=E\Leftrightarrow\overline{Y}''=E,\overline{X}'=E\Leftrightarrow\overline{X}''=E.$$

Clearly $\delta_1$ and $\delta_2$ can "simulate" $R_v$ and $R_h$ respectively, together with their control constraints. From the execution view of automaton's runs and grammar's derivations in scanning orders we have:

Claim. $G$ generates $p$ iff $A_G$ accepts $p$. Hence $L(GG)\subseteq L(GA_{II})$.

Remark. The above approach of local simulation seems unpractical when it is to be applied in the opposite direction. We have to proceed by an indirect way.

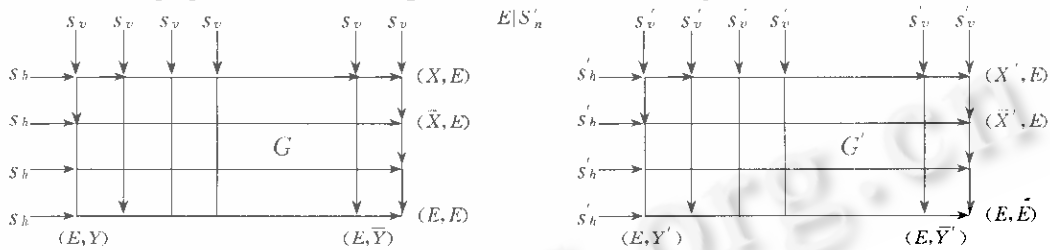
**4.3 4.$\leqslant$5. ($L(PCFRE)\subseteq L(GG)$)**

For this we have to verify that $L(GG)$ contains unit grids $\{a\}$ for all $a\in\Sigma$ and is closed under operations $\ominus$, ①, $\ominus^*$, ①$^*$, $\cap$, $\cup$ and projections.

1. Set $R_h=\{S_h\to a\}$, $R_v=\{S_v\to a\}$. Then the corresponding grid grammar generates $\{a\}$.

2. The closure of $\ominus$, ①, $\ominus^*$, ①$^*$.

The following figures show how to $h$-glue two compatible pictures generated by $G,G'$ respectively.




The indicated combination $E|S'_h$ of $E$ and $S'_h$ possesses both features i.e. vertical synchronization of stopping rightwards in $p$ as well as restarting again in $p'$.

Without loss of generality suppose, except $\Sigma$, symbols in $G$ and $G'$ are disjoint. Let

$G$①$G'=(\Sigma,V_h\cup(V'_h-\{S'_h\})\cup\{E|S'_h\},V_v\cup V'_v,S_h,\{S_v,S'_v\},\overline{R_h\cup R'_h},\overline{R_v\cup R'_v})$,

where except $E$, we introduce a new symbol $E|S'_h$;

$\overline{R_h\cup R'_h}$ is constructed from $R_h^*$ and $(R')_h^*$ by replacing $Y\to aE\in R_h^*$ by $Y\to a(E|S'_h)$ and $S'_h\to aY'\in(R')_h^*$ by $(E|R'_h)\to aY'$;

$\overline{R_v\cup R'_v}:=R_v^*\cup(R')_v^*$.

For control constraints, except the usual ones, we add a synchronization in the $h$-shift of the $h$-production-rules from $G$ to $G'$, which is realized by $E|S'_h$. Concerning the derivation order of $G$◯$G'$ the vertical scanning is preferable.

$G$①$G'$ is still a (special kind of) grid grammar and clearly $L(G)$①$L(G')$ is generated by $G$◯$G'$.

The arguments for other three operations can be done in a similar way.

3. Union

$G=G_1 \bigcup G_2$ through disjoint union of variables (including starting symbols) and production rules. The control constraints have a new restriction: $G_i (i=1,2)$ rules are applied independently.

Claim $G$ is a grid grammar and $L(G)=L(G_1) \bigcup L(G_2)$.

4. Intersection

Define $G=G_1 \times G_2$ through direct product:

$V_h:=V_h^1 \times V_h^2$, $V_v:=V_v^1 \times V_v^2$; $R_h:=R_h^1 \times R_h^2$, $R_v:=R_v^1 \times R_v^2$;

$S_h=(S_h^1, S_h^2)$, $S_v=(S_v^1, S_v^2)$, and endmark is $(E,E)$.

The control pattern is as usual but runs pair-wisely along each direction. Again $G$ is a grid grammar and $L(G)=L(G_1) \bigcup L(G_2)$.

5. Projection

Suppose $\pi: \Sigma \to \Sigma'$. Define $G':=\pi(G)$ by replacing each $a(\in \Sigma)$ in $G$'s production rules by $\pi(a)$ to obtain rules in $G'$. $G'$ remains to be a grid grammar, and $\pi(L(G))=L(G')$.

### 4.4 1(*II*)≤3 (Grid automata's $L^+(MP)$ definability)

Give $A_{II}=(\Sigma, Q, q_0, \delta_1, \delta_2, F)$, where $Q=\{q_0, q_1, \ldots, q_{m-1}\}$.

Similar to the case of words, where we introduce constant symbols for first and last positions[12], here we introduce three 1-ary predicates $Top(x)$, $Lef(x)$ and $Tar(x)$ for *top*-, *left(most)*- and *target*- positions in $dom(\bar{p})$, respectively. They are all $FO$-definable.

Let vocabulary $\tau=\{s_1, s_2, Top, Lef, Tar, P_a: a \in \overline{\Sigma}\}$, where $\overline{\Sigma}=\Sigma \bigcup \{\#\}$, $P_a$'s are 1-ary relation symbols, $s_1, s_2$ are 2-ary relation symbols for vertical and horizontal successors.

$Top(x) = \neg \exists\, y s_1(y,x) \in \forall_1$,

$Lef(x) = \neg \exists\, y s_2(y,x) \in \forall_1$,

$Norm(x) = \neg (Top(x) \vee Lef(x)) = \exists\, y s_1(y,x) \wedge \exists\, z s_2(z,x) \in \exists_1$, i.e. $x \in dom(p)$,

$Tar(x) = \neg \exists\, y s_1(x,y) \wedge \neg \exists\, z s_2(x,y) \in \forall_1$.

Denote

$$\psi_1 = \forall\, x [(Norm(x) \to \bigvee_{a \in \Sigma} P_a(x)) \wedge \bigwedge_{\substack{a \neq b \\ a,b \in \Sigma}} \neg (P_a(x) \wedge P_b(x))],$$

$$\psi_2 = \forall\, x (\neg Norm(x) \longleftrightarrow P_{\#}(x)) \wedge \exists\, x P_{\#}(x),$$

$$\psi_3(y_0, y'_0, \ldots, y_{n-1}, y'_{n-1}) = \forall\, x, y, z \{Norm(x) \wedge s_1(z,x) \wedge s_2(y,x) \wedge$$

$$\bigwedge_{\substack{i,j \in \{0,1,\ldots,n-1\} \\ a \in \Sigma}} (P_a(x) \wedge (y=y_j) \wedge (z=y_i)) \to \bigvee_{q_k \in \delta_1(q_i,a) \cap \delta_2(q_j,a)} [\bigwedge_{l \neq k} x \neq y'_l] \in \forall_1,$$

$$\Psi = \psi_1 \wedge \psi_2 \wedge P^{2,\ldots,2}_{y_0, y'_0, \ldots, y_{n-1}, y'_{n-1}} [\psi_3 \wedge Top(y_0) \wedge Lef(y_0) \wedge \bigvee_{q_i \in F} Tar(y_i)].$$

Now it's clear that $A_{II}$ accepts $p \in \Sigma^{**}$ iff $\bar{p}$ (viewed as $\tau$-structure) satisfies $\Psi$.

Note. (i) In order to guarantee that the interpretation of $P$ quantifier in $\Psi$ is just what we need we should ensure that the formula behind $P$-quantifier is universal ($\forall_1$-formula).

(ii) Using a technique in Ref. [12] we are able to reduce $P^{2,\ldots,2}$ in $\Psi$ down to $P^{1,\ldots,1}$. The $L^+(MP)$ specifications for stencil systems here are more succinct than those in Ref. [10]. We would say that partition logic is more suitable for tiling (stencil) systems and finite-state transition systems.

Now we have proved that (also apply G-R's Thm)

$$(2\times 2)TS=(1\times 1)SS=GA_I \leqslant GA_{II} \leqslant L^+(MP) \leqslant EMSO=PCFRE=TS.$$

$$PCFRE \leqslant GG \leqslant GA_{II}.$$

Hence $GA_I=GA_{II}=GG=(2\times 2)TS=SS=L^+(MP)=PCFRE$.

Final Remark. This work was done in 1997. Later the author was kindly informed of two related works[14,15]. Particularly Matz introduced a more complex combination of operations for pictures (than those treated here and in Ref. [3]), and more powerful notions for regular expressions with operators, and formulated a context-free grammar to characterizing them. However, these grammars can't guarantee in advance their final products to be rectangular picture at all, and it remains open to design a particular class of regular expressions with operators which just capture the class of REC, see Ref. [15]. They have been solved by grid grammars.

## References


1 Inoue K, Nakamura A. A survey of two-dimensional automata theory. LNCS 381, 1990. 72~91
2 Thomas W. Elements of an automata theory over partial orders. DIMACS. Series in Discrete Mathematics and Theoretical Computer Science, 1997,29:25~40
3 Giammarresi D, Restivo A. Two-Dimensional languages. In: Salomma A, Rozenberg G eds. Handbook of Formal Language. Vol.III, Berlin: Springer-Verlag, 1996. 251~268
4 Matz O, Thomas W. The monadic quantifier alternation hierarchy over graphs is infinite. In: Vardi M Y ed. Logics in Computer Science'97, IEEE Symposium. Los Alamitos, California: IEEE Computer Society, 1997. 236~244
5 Thomas Wilke. Star-free picture expressions are strictly weaker than first order logic. LNCS 1256, 1997. 347~35
6 Siromoney G, Siromoney R, Krithivasan K. Picture languages with array rewriting rules. Information and Control, 1973, 22:447~470
7 Nivat M, Saoudi A, Dare V. Parallel generation of finite images. International Journal of Pattern Recognition and Artificial Intelligence, 1989,3:279~294
8 Thomas W. On logics, tilings and automata. LNCS 530, 1991. 441~454
9 Seese D. Interpretability and tree automata: a simple way to solve algorithms on graphs closely related to trees. In: Nivat M *et al* eds. Tree Automata and Languages. Elsevier, 1992. 83~114
10 Shen En-shao. Some notes on graph automata, tiling systems and partition logic. Journal of Computer Science and Technology, 1998,63(6):483~489
11 Giammarresi D, Restivo A, Seibert S *et al*. Monadic second order logic over rectangular pictures and recognizability by tiling systems. Information and Computation, 1995,125:32~45
12 Shen En-shao, Tian Q. Monadic partition logics and finite automata. Theoretical Computer Science, 1996,166:63~81
13 Potthoff A, Seibert S, Thomas W. Nondeterminism vs. determinism on finite automata over direct acyclic graphs. Bulletin of the Belgian Mathematical Society, Simon Stevin 1, 1994. 285~298
14 Latteux M, Simplot D. Context-sensitive string languages and recognizable picture languages. Information and Computation, 1997,138:160~169
15 Matz O. Regular expressions and context-free grammars for picture languages. In: Proceedings of the STACS'97. LNCS 1200, 1997. 283~294
16 Latteux M, Simplot D. Recognizable picture languages and domino tiling. Theoretical Computer Science, 1997,178(1-2): 275~283
17 Simplot D. A charaterization of recognizable picture languages by tiling by finite sets. Theoretical Computer Science, 1999,218:297~323


# 处理图像语言的格点自动机与格点文法


沈恩绍

(上海交通大学计算机科学与工程系 上海 200030)



**摘要** Giammarresi 与 Restivo 在一篇综述中总结出一个关于可识别的图像语言(即 2 维矩形语言)REC 的等价性定理.对比 1 维字语言的相应结果,其中还缺少关于生成文法的相应一环.提出了一种(矩形的)格点文法,正好弥补了这一缺环.而取代 2 维 on-line tesselation 自动机,引入了格点自动机的概念.一方面,它与经典的 2 元树型自动机更相似,另一方面,它也是格点文法与等价性定理中关于 REC 的其他描述方式之间的一座桥梁.同时,标准的 existential monadic 二阶逻辑也被一种更弱的规范框架——positive monadic 分划逻辑所取代.由此导出一个新的更完整的关于 REC 的等价性定理.



**关键词** 形式语言,图像语言,自动机,文法,约束型二阶逻辑.

**中图法分类号** TP301