TWO-DIMENSIONAL REGULAR LANGUAGES AND THEIR SYNTACTIC CHARACTERIZATION

Tomokazu Arita\textsuperscript{1,}§, Kensei Tsuchida\textsuperscript{2}, Takeo Yaku\textsuperscript{3}

\textsuperscript{1}Division of Integrated Sciences
J.F. Oberlin University
3758 Tokiwa-machi, Machida-shi, Tokyo, 194-0294 JAPAN
e-mail: arita@obirin.ac.jp

\textsuperscript{2}Department of Information and Computer Sciences
Toyo University
2100 Kujirai, Kawagoe-shi, Saitama, 350-8585 JAPAN
e-mail: kensei@eng.toyo.ac.jp

\textsuperscript{3}Department of Computer Science and System Analysis
Nihon University
3-25-40 Sakurajosui, Setagaya-Ku, Tokyo, 156-8550 JAPAN
e-mail: yaku@cs.chs.nihon-u.ac.jp

Abstract: In this paper, we deal with two-dimensional rectangular arrays for representing row and column structures such as symbol tables, spreadsheets, etc. Firstly we formalize the two-dimensional regular languages by pairs of regular expressions for rows and columns. Next we also introduce context-sensitive graph grammars called labeled grid graph grammars, which characterize the two-dimensional regular languages. We note that the labeled grid graph grammars directly identify the two-dimensional regular languages. Furthermore, we show examples.

AMS Subject Classification: 68Q42, 68Q45
Key Words: two-dimensional regular expression, grid graphs, graph grammars, symbol tables, cellular automata

Received: November 25, 2008 © 2008, Academic Publications Ltd.

§Correspondence author
1. Introduction

Many tables contain repetitions of same patterns of rows and columns. Data types of cells in tables depend on structures of rows and columns in them. For example, it is general that symbol tables have same data types of fields. In this paper, our targets are tables with repetition of pattern of rows and columns.

Two-dimensional rectangular arrays and grid graphs are able to represent table structures and two-dimensional images. Several systems that recognize sets of two-dimensional rectangular arrays are proposed for pattern recognitions, etc. For example, two-dimensional regular expressions, tiling systems, four-way automata and so on [18] recognize several types of sets of two-dimensional rectangular arrays. D. Giammarresi et al proposed two-dimensional regular expression in [18]. They proposed two-dimensional regular expression by using horizontal and vertical concatenations. Furthermore, M. Anselmo et al proposed diagonal concatenation for two-dimensional regular expressions over one-letter [1], [2]. K. Inoue et al proposed automata that recognize languages of two-dimensional rectangular arrays, called on-line tessellation automata [12]. Graph grammars have succeeded in characterizing graph structures and generating several types of graphs [8], [9], [10], [13], [15], [16], [17], [19], [20]. We proposed a graph grammar for grid graphs in [5]. This grammar derives grid graphs with no node labels by synchronizing using state propagation among nodes in graph rewriting rules [4]. Label propagation is based on state propagation of cellular automata (e.g. [14], [21]).

In this paper, we consider two-dimensional rectangular arrays and grid graphs by paying attention to row and column structures.

In this paper, we introduce a system, called a pair regular expression, for representing two-dimensional arrays as tables with dependence between rows and columns. A pair regular expression consists of two (one-dimensional) regular expressions. One regular expression represents order of symbols of rows, and the other regular expression represents order of symbols of columns. Next, we introduce graph grammars for generating grid graphs with node labels representing two-dimensional rectangular arrays, called labeled grid graph grammars. Labeled grid graph grammars are constructed by extending grid graph grammar without node labels [4], [5] based on state propagation of cellular automata (e.g. [14], [21]). The two-dimensional rectangular array is identified from the set of labeled graphs by the following steps. (1) The grid graph grammar [5] identifies the graph structure of two-dimensional regular arrays from the graphs.
(2) Certain algorithms recognize the orders of labels. However, simultaneous recognition method such as graph grammar for structures and the label order seems to be useful for table processing. We also show that there exists a graph grammar corresponding to two-dimensional rectangular arrays represented by pair regular expressions.

This paper is organized as follows. In Section 2 we review some definitions and basic notations. In Section 3 we propose definitions for two-dimensional rectangular arrays and pair regular expressions. In Section 4 we propose graph grammars for pair regular expressions. In Section 5 we propose maps from two-dimensional rectangular arrays to pictures. Finally Section 6 concludes.

Part of the result in this paper appeared in [3].

2. Preliminaries

In this paper, we follow the basic notation and properties of the theory of one-dimensional language found in [11].

We firstly review a definition of finite automata [11] as follows. A deterministic finite automaton is a 5-tuple \((Q, \Sigma, \delta, q_0, F)\), where \(Q\) is a finite set of states, \(\Sigma\) is a finite input alphabet, \(q_0\) in \(Q\) is the initial state, \(F \subseteq Q\) is the set of final states, and \(\delta\) is the transition function mapping \(Q \times \Sigma\) to \(Q\) [11]. Here \(\delta\) may be extended to the function mapping \(Q \times \Sigma^*\) to \(Q\). A string \(x\) in \(\Sigma^*\) is accepted by a finite automaton \(M = (Q, \Sigma, \delta, q_0, F)\) if \(\delta(q_0, x) = p\) for some \(p\) in \(F\). The language accepted by \(M\) is the set \(\{x | \delta(q_0, x) \in F\}\) [11].

A nondeterministic finite automata is a 5-tuple \((Q, \Sigma, \delta, q_0, F)\), where \(Q, \Sigma, q_0,\) and \(F\) have the same meaning as for a deterministic finite automaton, and \(\delta\) is the function mapping \(Q \times \Sigma\) to \(2^Q\) [11].

We next also review definitions about two-dimensional languages [1].

Let \(\Sigma\) be a finite alphabet. Then \(\Sigma^*\) denotes all strings of symbols in \(\Sigma\) (see [11]), and \(\Sigma^{**}\) denotes the set of two-dimensional strings (pictures) over \(\Sigma\) (see [1], [2]). We here denote an element of a picture \(p\) over a coordinate \((i, j)\) by \(p(i, j)\), where \(i\) and \(j\) are positive integers [18].

**Example 1.** We illustrate a picture over \(\{0, 1\}\) in Figure 1.

Thirdly, we review several definitions about graph grammars found in [5], [17].

In [5], we dealt with graphs with multi node labels and multi edge labels.
In this paper, we consider that graphs have multi node labels and single edge label. That is, we deal with only graphs without edge labels. Furthermore, we also deal with graph grammars without edge labels.

Let $\Sigma$ be an alphabet of node labels. A graph over $\Sigma$ is $G = (V, E, \psi)$, where $V$ is the finite set of nodes, $E \subseteq \{(v, w) | v, w \in V\}$ is the set of edges, and $\psi : V \rightarrow \Sigma$ is the labeling function. In this paper, we always assume that $G$ is connected. Graph $G = (V, E, \psi)$ is an undirected graph if, for every $(v, w) \in E$, there is also $(w, v) \in E$ (cf. [5], [17]).

A production $p$ is of the form $M \rightarrow (D, C)$, where $M$ is a graph over $\Sigma$, $D$ is a graph over $\Sigma$ called an embedded graph, and $C \subseteq \Sigma \times V_M \times V_D \times \{\text{in}, \text{out}\}$, called a connection relation, is a set of connection instructions (cf. [5], [17]).

A production $p : M \rightarrow (D, C)$ removes a subgraph $M'$ of its applied graph $G$, such that $M$ and $M'$ are isomorphic, and replaces $M'$ with $D'$, such that $D'$ and $D$ are isomorphic and the set of nodes in $G$ and $D'$ are pairwise disjoint. Then the connection instructions of $C$ replace neighboring edges around $M'$. A connection instruction $(\alpha, x, y, \text{out})$ of $C$ means that if there was an edge from the node $x$ in $G$ for which $(D, C)$ is substituted to a node $w$ with label $\alpha$, then the embedding process will establish an edge from $y$ to $w$. Similarly, a connection instruction $(\alpha, x, y, \text{in})$ of $C$ means that if there was an edge from a node $w$ with label $\alpha$ to the $x$ in $G$ for which $(D, C)$ is substituted, then the embedding process will establish an edge from $w$ to $y$.

A graph grammar is $GG = (\Sigma, \Delta, P, S)$, where $\Sigma$ is the finite set of node labels, $\Delta \subseteq \Sigma$ is the set of terminal node labels, $P$ is the set of productions, and $S$ is the start graph (cf. [5], [17]).

The language generated by graph grammar $GG = (\Sigma, \Delta, P, S)$ is $L(GG) = \{ g | g \text{ is derived from } S \text{ by } GG , \text{ all node labels of } g \text{ are terminal node labels } \}$ (cf. [17]).

Example 2. Figure 2 shows an example of a production and a derivation of a graph. Node labels $\sharp$, $C_s$, $C_k$, $W_1$, $B$, and $A$ are non-terminal. Node label $C_t$ is terminal. Furthermore, a box denotes a nonterminal node,
Two-dimensional regular languages and... 283

Figure 2: An application of a production of a context-sensitive graph grammar

The figure shows an application of a production of a context-sensitive graph grammar. A black circle denotes a terminal node, a line denotes an undirected edge, and a dotted line is part of a connection instruction. Production $p$ is $M \rightarrow (D,C)$, where $M = (V,E,\psi)$ such that $V = \{x_1, x_2\}$, $E = \{(x_1, x_2), (x_2, x_1)\}$, $\psi(x_1) = C_k$, and $\psi(x_2) = A$, and $D = (V_D, E_D, \psi_D)$ such that $V_D = \{x_3, x_4, x_5, x_6\}$, $E_D = \{(x_3, x_4), (x_4, x_5), (x_5, x_6), (x_6, x_3)\}$, $\psi(x_3) = C_k$, $\psi(x_4) = B$, $\psi(x_5) = A$, $\psi(x_6) = C_t$ and $C = \{c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8\}$ such that $c_1 = (C_s, x_1, x_3, \text{in})$, $c_2 = (\#, x_1, x_3, \text{in})$, $c_3 = (W_1, x_2, x_4, \text{in})$, $c_4 = (B, x_2, x_4, \text{in})$, $c_5 = (C_s, x_1, x_3, \text{out})$, $c_6 = (\#, x_1, x_3, \text{out})$, $c_7 = (W_1, x_2, x_4, \text{out})$, and $c_8 = (B, x_2, x_4, \text{out})$.

By applying $p$ to graph $G_1$, $G_2$ is obtained in Figure 2.

Here we review a definition of grid graphs ([5], [7]). A (undirected) graph is called a **two-dimensional grid graph** iff the graph consists of the product of two path graphs [7]. Let $P_n$ and $P_m$ be path graphs, then $P_n \square P_m$ denotes the product of $P_n$ and $P_m$. If $P_n$ is a path graph with $n$ nodes and $P_m$ is a path graph with $m$ nodes, then $T_{n,m}$ denotes a grid graph for the product of the $P_n$ and $P_m$. 


3. Pair Regular Expressions for Two-Dimensional Regular Languages

In this section, we introduce notations and definitions for a model of two-dimensional regular expressions.

**Definition 1.** A two-dimensional \( m \times n \) rectangular array over \( \Sigma \times \Gamma \) represents a rectangular placement of \( mn \) elements \((s_i, g_j)\) in \( \Sigma \times \Gamma\), \((i = 1, ..., m; j = 1, ..., n)\) as follows:

\[
\begin{array}{cccc}
(s_1, g_1) & \cdots & (s_1, g_j) & \cdots & (s_1, g_n) \\
\vdots & \ddots & \vdots & \ddots & \vdots \\
(s_i, g_1) & \cdots & (s_i, g_j) & \cdots & (s_i, g_n) \\
\vdots & \ddots & \vdots & \ddots & \vdots \\
(s_m, g_1) & \cdots & (s_m, g_j) & \cdots & (s_m, g_n)
\end{array}
\]

The set of all two-dimensional \( m \times n \) rectangular arrays over \( \Sigma \times \Gamma \) such that \( m \geq 1 \) and \( n \geq 1 \) is denoted by \((m, n) - RA_{\Sigma \times \Gamma}\). The set of two-dimensional rectangular arrays over \( \Sigma \times \Gamma \), denoted \( RA_{\Sigma \times \Gamma}\), is the set \( RA_{\Sigma \times \Gamma} = \bigcup_{m \geq 1, n \geq 1} (m, n) - RA_{\Sigma \times \Gamma}\). A two-dimensional language over \( \Sigma \times \Gamma \) is a subset of \( RA_{\Sigma \times \Gamma}\).

Given a two-dimensional rectangular array \( a \in RA_{\Sigma \times \Gamma}\), each element \((s, g)\) of \( a \) is called a *cell* of \( a \), a horizontally arrangement of cells of \( a \) is called a *row* of \( a \), and a vertically arrangement of cells of \( a \) is called a *column* of \( a \). Furthermore, the number of rows of \( a \) is called the *height* of \( a \) and the number of columns of \( a \) is called the *width* of \( a \).

Let \( m \) be the height of \( a \) and let \( n \) be the width of \( a \). \( m \cdot n \) is called the *size* of \( a \).

In this paper, we illustrate two-dimensional rectangular arrays by drawing lines among cells (see Example 3).

**Example 3.** Let \( \Sigma = \{x, y\} \) and \( \Gamma = \{0, 1\} \). We now illustrate a two-dimensional \( 3 \times 5 \) rectangular array \( \alpha_1 \) where for each cell \((s_i, g_j)\) of \( \alpha_1 \) \((i = 1, 2, 3; j = 1, ..., 5)\), \((s_i, g_j) = (x, 0)\) such that \( i = j \), and \((s_i, g_j) = (y, 1)\) such that \( i \neq j \).

\[
\begin{array}{cccccc}
(x, 0) & (y, 1) & (y, 1) & (y, 1) & (y, 1) \\
(y, 1) & (x, 0) & (y, 1) & (y, 1) & (y, 1) \\
(y, 1) & (y, 1) & (x, 0) & (y, 1) & (y, 1)
\end{array}
\]

\( \alpha_1 = \)
Here we define a new notation that denotes two-dimensional rectangular arrays.

**Definition 2.** Let $\Sigma$ be a finite alphabet of symbols, and $\Gamma$ be a finite alphabet of symbols. A pair string over $\Sigma \times \Gamma$ is a pair $s = \langle r, c \rangle$, where a string $r = r_1r_2...r_m$ in $\Sigma^+$ and a string $c = c_1c_2...c_n$ in $\Gamma^+$, and we call them a row string and a column string, respectively. Each symbol $r_i$ in $r$ is called a row label and each symbol $c_j$ in $c$ is called a column label. The set of all pair strings over $\Sigma \times \Gamma$ is denoted by $E_{\Sigma \times \Gamma}$.

Given a $s = \langle r, c \rangle \in E_{\Sigma \times \Gamma}$ such that $r = r_1r_2...r_m$, $c = c_1c_2...c_n$, $m \geq 1$, and $n \geq 1$, a two-dimensional rectangular array over $\Sigma \times \Gamma$ represented by $s$, denoted $a(s)$, is a two-dimensional $m \times n$ rectangular array such that $(s_i, g_j) = (r_i, c_j)$ for each cell $(s_i, g_j)$ in $a(s)$, $i = 1, 2, ..., m$, and $j = 1, 2, ..., n$.

**Example 4.** Let $\Sigma = \{x, y\}$, $\Gamma = \{0, 1, 2\}$, and a pair string $s_1 = \langle xy, 012 \rangle$ over $\Sigma$ and $\Gamma$. We now illustrate the two-dimensional rectangular array $a(s_1)$ represented by $s_1$.

$$a(s_1) = \begin{array}{ccc}
(x, 0) & (x, 1) & (x, 2) \\
(y, 0) & (y, 1) & (y, 2)
\end{array}$$

We identify the boundary of a pair string by using special symbols. For any pair string $s$ over $\Sigma \times \Gamma$, we define $\hat{s}$ as the pair string obtained by surrounding $s$ with a special boundary symbol $\# \notin \Sigma \cup \Gamma$ (see [1]). That is, for a pair string $s$ such that $s = \langle r, c \rangle$, $\hat{s} = \langle \#r\#, \#c\# \rangle$.

**Example 5.** As an example of a pair string with boundary symbols, we show the two-dimensional rectangular array $a(\hat{s}_1)$ represented by the pair string $\hat{s}_1$ obtained by surrounding $s_1$ of Example 4 with the symbol $\#$.

$$a(\hat{s}_1) = \begin{array}{cccc}
(\#, \#) & (\#, 0) & (\#, 1) & (\#, 2) \\
(\#, \#) & (\#, 0) & (\#, 1) & (\#, 2) \\
(\#, \#) & (\#, 0) & (\#, 1) & (\#, 2) \\
(\#, \#) & (\#, 0) & (\#, 1) & (\#, 2)
\end{array}$$

We here extends pair strings and define pair regular expressions.

**Definition 3.** Let $R$ be a (one-dimensional) regular expression over $\Sigma$, and $C$ be a (one-dimensional) regular expression over $\Gamma$. A pair regular expression over $\Sigma \times \Gamma$, denoted by $\langle R, C \rangle$, represents the set $L(\langle R, C \rangle) = \{ \langle r, c \rangle | r$ is a string in the set represented by $R$, $c$ is a string in the set represented by $C$, and
A pair regular expression over $\Sigma \times \Gamma$ represents a subset of $E_{\Sigma \times \Gamma}$.

We here show examples for pair regular expressions.

**Example 6.** Let $\Sigma = \{0,1\}$ and $\Gamma = \{0,1\}$ The pair regular expression $\langle 00,11 \rangle$ over $\Sigma \times \Gamma$ denotes $\{\langle 00,11 \rangle\}$. The pair regular expression $\langle 0^+,1^+ \rangle$ over $\Sigma \times \Gamma$ denotes $\{\langle 0,1 \rangle, \langle 00,1 \rangle, \langle 0,11 \rangle, \langle 00,11 \rangle, \ldots \}$. The pair regular expression $\langle (0+1)^+, (0+1)^+ \rangle$ over $\Sigma \times \Gamma$ denotes $\{\langle r,c \rangle | r$ and $c$ are elements in the set of all strings of 0’s and 1’s, respectively $\}.$

**Example 7.** Let $\Sigma = \{d,h,i,m\}$ and $\Gamma = \{k,s,t\}$. Let $L_1$ be a two-dimensional regular expression $L_1 = \langle h(i+d+m)^*,(skt)^+ \rangle$. A pair string $s_2 = \langle hidmiimi,sktsktskt \rangle$ is an element of $L_1$ and Figure 3 shows the two-dimensional rectangular array $a(s_2)$.

**Definition 4.** The language represented by a pair regular expression $\langle R,C \rangle$ over $\Sigma \times \Gamma$ is the set $L(\langle R,C \rangle)$. A language $L$ over $\Sigma \times \Gamma$ is pair regular if $L$ is the set represented by some pair regular expression $\langle R,C \rangle$ over $\Sigma \times \Gamma$. $L$ is called a pair regular language.

### 4. Graph Grammars for Two-Dimensional Regular Languages

In this section, we construct graph grammars for characterizing syntactically two-dimensional regular languages. Firstly, we propose labeled grid graphs corresponding to two-dimensional rectangular arrays. Next, we propose graph
Remarks. The two-dimensional rectangular array is identified from the set of labeled graph by the following steps. (1) The grid graph grammar [5] identifies the graph structure of two-dimensional regular arrays from the graphs, or the algorithm of Theorem 2 in [7] recognizes the graphs as grid graphs. (2) Certain algorithms recognize the orders of labels.

However, simultaneous recognition method such as graph grammar for structures and the label order seems to be useful for table processing.

We consider several types of node labels in order to construct grammars. For symbols $x$ and $y$, we use node labels $(x, y)$, $[x, y]$, and $[[x, y]]$ as different labels. We especially use $(x, y)$ for representing a cell in two-dimensional rectangular arrays.

We here introduce graphs for representing two-dimensional rectangular arrays.

**Definition 5.** A labeled grid graph over $\Sigma \times \Gamma$ is a 3-tuple $g = (V, E, \psi)$, where $V$ is a finite set of nodes, $E$ is a finite set of edges, $\psi$ is a node labeling function mapping $V$ to $\Sigma \times \Gamma$, and $g$ is a grid graph.

We define labeled grid graphs representing two-dimensional rectangular arrays.

**Definition 6.** Let $a$ be a two-dimensional $m \times n$ rectangular array over $\Sigma \times \Gamma$. The labeled grid graph representing $a$, denoted by $g(a)$, is defined as the labeled grid graph $g(a) = (V_{g(a)}, E_{g(a)}, \psi_{g(a)})$, where $g(a)$ is $p_1\Box p_2$ such that $p_1$ is a path of length $m$ from node $v_1$ to $v_m$ and $p_2$ is a path of length $n$ from $w_1$ to $w_n$, and $\psi((v_i, w_j)) = (s_i, g_j)$ for $(v_i, w_j) \in V_{g(a)}$, cell $(s_i, g_j)$ of $a$, $i = 1, \ldots, m$, and $j = 1, \ldots, n$.

**Example 8.** Figure 4 shows the labeled grid graph representing $a(s_1)$ in Example 4.

Next, we introduce graph grammars for pair regular expressions by extending the grid graph grammar in [5] based on labeled grid graphs.

**Definition 7.** Let $\Sigma$ be a set of row labels and $\Gamma$ be a set of column labels. A labeled grid graph grammar for a pair regular expression $(R, C)$ over $\Sigma \times \Gamma$ is a tuple $(\Sigma_G, \Gamma_G, P_G, S_G)$, where $\Sigma_G$ is a set of node labels; $\Gamma_G$ is a set of terminal node labels that is $\{ t \mid t$ is a node label for $(s, q)$ such as $s \in \Sigma \cup \{^\sharp\}$.
and \( q \in \Gamma \cup \{\sharp\} \); \( P_G \) is a set of productions; \( S_G \) is the start graph that has single node and it is labeled by symbol \( S \).

We show all production types for labeled grid graph grammars in Figure 5, 6, and 7.

Here the node label \( S \) is the node label of the start graph, labels \( A \) and \( B \) are used for generating the first row, label \( C \) is used for generating middle rows, label \( X \) is used for state propagations of width, and label \( F \) is used for generating the bottom boundary. Symbols \( s \) and \( t \) denote symbols in \( \Sigma \), respectively. Symbols \( a \) and \( b \) denote symbols in \( \Gamma \), respectively. Symbols \( q \) and \( r \) are not contained in \( \Sigma \) and \( \Gamma \). Furthermore, a special symbol \( * \) used in connection instructions represents all node labels.

Productions of types \( P_7 \), \( P_{7-2} \), \( P_{7-3} \), \( P_{7-4} \), \( P_8 \), \( P_9 \), \( P_{10} \), \( P_{13} \), \( P_{13-2} \), \( P_{13-3} \), \( P_{13-4} \), \( P_{14} \), \( P_{15} \), \( P_{16} \), \( P_{19} \), \( P_{20} \), \( P_{21} \), and \( P_{22} \) consist of combinations of all symbols for \( \Sigma \) and \( \Gamma \). For example, node label \( a \) and \( s \) in type \( P_7 \) denote elements of \( \Sigma \) and \( \Gamma \), respectively. Productions for type \( P_7 \) needs \( |\Sigma| \times |\Gamma| \) concrete productions by replacing symbols \( a \) and \( s \) in \( P_7 \).

We note that a labeled grid graph grammar generates labeled grid graphs representing two-dimensional arrays with boundary. For a labeled grid graph grammar \( G \) for \( \langle R, C \rangle \) over \( \Sigma \times \Gamma \), the language \( L(G) \) is the set of labeled grid graphs that represent two-dimensional rectangular arrays for pair strings obtained by surrounding elements in \( L(\langle R, C \rangle) \) with the boundary symbol \( \sharp \).

**Definition 8.** Let \( \Sigma \) be a set of row labels and \( \Gamma \) be a set of column labels. \( G \) is a labeled grid graph grammar over \( \Sigma \times \Gamma \) if there exists some pair regular expression \( \langle R, C \rangle \) over \( \Sigma \times \Gamma \) such that \( G \) is a labeled grid graph grammar for \( \langle R, C \rangle \).

We show properties of labeled grid graph grammars.

**Proposition 1.** Let \( G = (\Sigma_G, \Gamma_G, P_G, S_G) \) be a labeled grid graph grammar over \( \Sigma \times \Gamma \). \( G \) generates only labeled grid graphs over \( \Sigma \cup \{\sharp\} \times \Gamma \cup \{\sharp\} \).
Proof. This is shown by forms of productions and from derivation steps of grammars.

We show all order of applications of productions in $P_G$ from the start graph.
Figure 6: Production types of labeled grid graph grammars (2)

$S_G$ in Figure 8. We consider the generations of labeled grid graphs for $m \times n$ rectangular arrays corresponding to pair strings such that $m \geq 2$ and $n \geq 2$. The first column is generated by a production of the $P_1$ type. The other columns
are generated by productions of the $P_2$ type. Furthermore, the first row is generated by a production of the $P_6$ type. The other rows are generated by productions of the $P_{12}$ type. The other productions are used for changing node labels and inheriting row and column labels from neighbor nodes.

Each production types $P_6$, $P_6-2$, $P_7$, $P_7-2$, $P_8$, $P_9$, $P_{12}$, $P_{13}$, $P_{13-2}$, $P_{14}$, $P_{15}$, and $P_{17}$ has a triangle cycle in the graph of the right hand sides. Thus, graphs applied these productions to have a triangle cycle. However, the triangle cycle is replaced to a rectangular cycle by application of a next production to these graphs. Finally, graphs generated form applications of productions of $P_G$ become grid graphs.

Similarly, $G$ generates only grid graphs representing $1 \times 1$, $m \times 1$, and $1 \times n$
rectangular arrays, $m \geq 2$ and $n \geq 2$, by order of applications of productions as shown in Figures 8.

Furthermore, terminal node labels are pairs $(s,g)$ such that $s \in \Sigma \cup \{\sharp\}$ and $g \in \Gamma \cup \{\sharp\}$. Thus, labeled grid graph grammars generate only labeled grid graphs.

**Proposition 2.** Let $\Sigma$ and $\Gamma$ be finite alphabets and $e$ be a pair regular expression over $\Sigma \times \Gamma$. There exists a labeled grid graph grammar $G_e$ over $\Sigma \times \Gamma$ and $L(G_e)$ contains only labeled grid graphs for all elements in $L(e)$.

**Proof.** Let $e = \langle R, C \rangle$ over $\Sigma \times \Gamma$. Let $A = (Q_A, \Sigma, \delta_A, q_{A0}, F_A)$ be a deterministic finite automaton corresponding to $R$ and let $B = (Q_B, \Gamma, \delta_B, q_{B0}, F_B)$ be a deterministic finite automaton corresponding to $C$.

We define a labeled grid graph grammar $G_e = (\Sigma_e, \Gamma_e, P_e, S_e)$ over $\Sigma \times \Gamma$ for $e$, as follows.

Let $S_1 = \{S, A, B, C, D_1, D_2, F, X, \sharp, E_1, E_2, W_1, W_2, W_3\}$, $S_2 = \{[s,t] | s \in \Sigma \text{ and } t \in \Gamma\}$, $S_3 = \{[s,q_A] | s \in \Sigma \text{ and } q_A \in Q_A\}$, $S_4 = \{[s,q_B] | s \in \Gamma \text{ and } q_B \in Q_B\}$, $S_5 = \{([s,g]) | s \in \Sigma \text{ and } g \in \Gamma\}$, and $S_6 = \{(s,t) | s \in \Sigma \cup \{\sharp\} \text{ and } t \in \Gamma \cup \{\sharp\}\}$.

Here we assume that $S_i$ for $i = 1, 2, ..., 6$ and $\Gamma$ are pairwise disjoint.

$\Sigma_e = \bigcup_{1 \leq i \leq 6} S_i \cup \Sigma \cup \Gamma$. $\Gamma_e = S_6$.

$P_e$ is constructed as follows. Firstly let $P_e$ be an empty set. If there exists $\delta_B(q_{B0}, s) = q$ then a following production $p_1$ is added to $P_e$: $p_1 : G_1 \rightarrow (D_1, \emptyset)$ such that $G_1 = (\{v_1\}, \emptyset, \psi_{G_1})$ where $\psi_{G_1}(v_1) = S$, $S_1 = \{[w_i | i = 1, 2, 3, 4]\}$, $\delta_{D_1} = \{(e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, \psi_{D_1} | e_1 = (w_1, w_2), e_2 = (w_2, w_3), e_3 = (w_3, w_4), e_4 = (w_4, w_1), e_5 = (w_3, w_1), e_6 = (w_4, w_3), e_7 = (w_1, w_4), e_8 = (w_1, w_4), \psi_{D_1}(w_1) = \sharp, \psi_{D_1}(w_2) = [s, g], \psi_{D_1}(w_3) = A, \text{ and } \psi_{D_1}(w_4) = W_1\}$. That is, production $p_1$ is the $P_1$ type.

If there exists $\delta_B(q_1, s) = q_2$ and $\delta_B(q_2, t) = q_3$ then a following production $p_2$ is added to $P_e$: $p_2 : G_2 \rightarrow (D_2, C_2)$ such that $G_2 = (\{v_1, v_2\}, \{(v_1, v_2), (v_2, v_1)\}, \psi_{G_2})$ where $\psi_{G_2}(v_1) = [s, q_2]$ and $\psi_{G_2}(v_2) = A$, $D_2 = ([w_i | i = 1, 2, 3, 4], \{e_{ij} | j = 1, 2, \ldots, 8\}, \psi_{D_2})$ where $e_1 = (w_1, w_2)$, $e_2 = (w_2, w_3)$, $e_3 = (w_3, w_4)$, $e_4 = (w_4, w_1)$, $e_5 = (w_3, w_1)$, $e_6 = (w_4, w_3)$, $e_7 = (w_1, w_4)$, $e_8 = (w_1, w_4)$, $\psi_{D_2}(w_1) = s$, $\psi_{D_2}(w_2) = [t, q_3]$, $\psi_{D_2}(w_3) = A$, and $\psi_{D_2}(w_4) = B$, and $C_2 = \{(*, v_1, w_1, \text{in}), (*, v_1, w_1, \text{out}), (*, v_2, w_4, \text{in}), (*, v_2, w_4, \text{out})\}$. That is, production $p_2$ is the $P_2$ type.

If there exists $\delta_B(q, s) = q_F$ such that $q_F \in F_B$ then a following production $p_3$ is added to $P_e$: $p_3 : G_3 \rightarrow (D_3, C_3)$ such that $G_3 = (\{v_1, v_2\}, \{(v_1, v_2), (v_2, v_1)\}, \psi_{G_3})$ where $\psi_{G_3}(v_1) = [s, q_F]$ and $\psi_{G_3}(v_2) = A$, $D_3 = ([w_i | i =
1. Derivation steps for 1x1 rectangular arrays

\[ P_1, type \rightarrow P_1, type \rightarrow P_2, type \rightarrow P_3, type \rightarrow P_4, type \rightarrow P_5, type \rightarrow P_6, type \rightarrow P_7, type \rightarrow P_8, type \rightarrow P_9, type \rightarrow P_{10}, type \rightarrow P_{11}, type \rightarrow P_{12}, type \rightarrow P_{13}, type \rightarrow P_{14}, type \rightarrow P_{15}, type \rightarrow P_{16}, type \rightarrow P_{17}, type \rightarrow P_{18}, type \rightarrow P_{19}, type \rightarrow P_{20}, type \rightarrow P_{21}, type \rightarrow P_{22}, type \]  

2. Derivation steps for 1xn rectangular arrays

\[ P_1, type \rightarrow P_1, type \rightarrow \cdots \rightarrow P_n, type \rightarrow P_1, type \rightarrow P_2, type \rightarrow P_3, type \rightarrow P_4, type \rightarrow P_5, type \rightarrow P_6, type \rightarrow P_7, type \rightarrow P_8, type \rightarrow P_9, type \rightarrow P_{10}, type \rightarrow P_{11}, type \rightarrow P_{12}, type \rightarrow P_{13}, type \rightarrow P_{14}, type \rightarrow P_{15}, type \rightarrow P_{16}, type \rightarrow P_{17}, type \rightarrow P_{18}, type \rightarrow P_{19}, type \rightarrow P_{20}, type \rightarrow P_{21}, type \rightarrow P_{22}, type \]  

3. Derivation steps for mx1 rectangular arrays

\[ P_1, type \rightarrow P_1, type \rightarrow P_2, type \rightarrow \cdots \rightarrow P_n, type \rightarrow P_1, type \rightarrow P_2, type \rightarrow P_3, type \rightarrow P_4, type \rightarrow P_5, type \rightarrow P_6, type \rightarrow P_7, type \rightarrow P_8, type \rightarrow P_9, type \rightarrow P_{10}, type \rightarrow P_{11}, type \rightarrow P_{12}, type \rightarrow P_{13}, type \rightarrow P_{14}, type \rightarrow P_{15}, type \rightarrow P_{16}, type \rightarrow P_{17}, type \rightarrow P_{18}, type \rightarrow P_{19}, type \rightarrow P_{20}, type \rightarrow P_{21}, type \rightarrow P_{22}, type \]  

4. Derivation steps for mxn rectangular arrays

\[ P_1, type \rightarrow P_1, type \rightarrow \cdots \rightarrow P_n, type \rightarrow P_1, type \rightarrow P_2, type \rightarrow P_3, type \rightarrow P_4, type \rightarrow P_5, type \rightarrow P_6, type \rightarrow P_7, type \rightarrow P_8, type \rightarrow P_9, type \rightarrow P_{10}, type \rightarrow P_{11}, type \rightarrow P_{12}, type \rightarrow P_{13}, type \rightarrow P_{14}, type \rightarrow P_{15}, type \rightarrow P_{16}, type \rightarrow P_{17}, type \rightarrow P_{18}, type \rightarrow P_{19}, type \rightarrow P_{20}, type \rightarrow P_{21}, type \rightarrow P_{22}, type \]  

Figure 8: The order of applications of productions in \( P_G \)

\[ 1, 2, 3, 4 \}, \{ e_j | j = 1, 2, \ldots, 8 \}, \psi_{D_3} \) where \( e_1 = (w_1, w_2) \), \( e_2 = (w_2, w_1) \), \( e_3 = (w_2, w_3) \), \( e_4 = (w_3, w_2) \), \( e_5 = (w_3, w_4) \), \( e_6 = (w_4, w_3) \), \( e_7 = (w_4, w_1) \), \( e_8 = (w_1, w_4) \), \( \psi_{D_3}(w_1) = s \), \( \psi_{D_3}(w_2) = \emptyset \), \( \psi_{D_3}(w_3) = E_1 \), and \( \psi_{D_3}(w_4) = B \), and
\[C_3 = \{(*, v_1, w_1, \text{in}), (*, v_1, w_1, \text{out}), (*, v_2, w_4, \text{in}), (*, v_2, w_4, \text{out})\}.\] That is, production \(p_3\) is the \(P_3\) type.

If there exists \(\delta_A(q_{A_0}, s) = q\) where \(q \notin F_A\) then a following production \(p_4\) is added to \(P_e: p_4\) is \(G_4 \rightarrow (D_4, C_4)\) such that \(G_4 = \{v_1, v_2, v_3\}, \{v_1, v_2\}, (v_2, v_1), (v_2, v_3), (v_3, v_2)\}, \psi_{G_4}\) where \(\psi_{G_4}(v_1) = \sharp, \psi_{G_4}(v_2) = W_1, \psi_{G_4}(v_3) = X,\) and \(D_4 = \{\{w_i | i = 1, 2, 3, 4\}, \{e_j | j = 1, 2, \ldots, 8\}\}, \psi_{D_4}\) where \(e_1 = (w_1, w_2), e_2 = (w_2, w_1), e_3 = (w_2, w_3), e_4 = (w_3, w_2), e_5 = (w_2, w_4), e_6 = (w_4, w_2), e_7 = (w_3, w_4), e_8 = (w_4, w_3)\), \(\psi_{D_4}(w_1) = [s, \sharp], \psi_{D_4}(w_3) = s,\) \(\text{and} \psi_{D_4}(w_4) = W_2,\) and \(C_4 = \{(*, v_1, w_1, \text{in}), (*, v_1, w_1, \text{out}), (*, v_3, w_3, \text{in}), (*, v_3, w_3, \text{out})\}.\) That is, production \(p_4\) is the \(P_6\) type.

If there exists \(\delta_A(q_{A_0}, s) = q\) where \(q \in F_A\) then following productions \(p_5\) and \(p_6\) are added to \(P_e: p_5\) is \(G_5 \rightarrow (D_5, C_5)\) such that \(G_5 = \{v_1, v_2, v_3\}, \{v_1, v_2\}, (v_2, v_1), (v_2, v_3), (v_3, v_2)\}, \psi_{G_5}\) where \(\psi_{G_5}(v_1) = \sharp, \psi_{G_5}(v_2) = W_1, \psi_{G_5}(v_3) = X,\) and \(D_5 = \{\{w_i | i = 1, 2, 3, 4\}, \{e_j | j = 1, 2, \ldots, 8\}\}, \psi_{D_5}\) where \(e_1 = (w_1, w_2), e_2 = (w_2, w_1), e_3 = (w_2, w_3), e_4 = (w_3, w_2), e_5 = (w_2, w_4), e_6 = (w_4, w_2), e_7 = (w_3, w_4), e_8 = (w_4, w_3)\), \(\psi_{D_5}(w_1) = [s, \sharp], \psi_{D_5}(w_3) = s,\) \(\text{and} \psi_{D_5}(w_4) = W_3,\) \(\text{and} \)\(C_5 = \{(*, v_1, w_1, \text{in}), (*, v_1, w_1, \text{out}), (*, v_3, w_3, \text{in}), (*, v_3, w_3, \text{out})\}.\) Furthermore, \(p_6\) is \(G_6 \rightarrow (D_6, C_6)\) such that \(G_6 = \{v_1, v_2, v_3\}, \{v_1, v_2\}, (v_2, v_1), (v_2, v_3), (v_3, v_2)\}, \psi_{G_6}\) where \(\psi_{G_6}(v_1) = \{s, q\}, \psi_{G_6}(v_2) = W_3, \psi_{G_6}(v_3) = X,\) and \(D_6 = \{\{w_i | i = 1, 2, 3\}, \{w_1, w_2, w_2, w_1\}, \psi_{D_6}\) where \(\psi_{D_6}(w_1) = [s, \sharp], \psi_{D_6}(w_2) = \sharp,\) \(\text{and} \psi_{D_6}(w_3) = F,\) \(\text{and} \)\(C_6 = \{(*, v_1, w_1, \text{in}), (*, v_1, w_1, \text{out}), (*, v_3, w_3, \text{in}), (*, v_3, w_3, \text{out})\}.\) That is, production \(p_5\) is the \(P_6\) type and \(p_6\) is the \(P_{18}\) type.

If there exists \(\delta_A(q_{A_1}, s) = q_2\) and \(\delta_A(q_{A_2}, t) = q_3\) then a following production \(p_7\) is added to \(P_e: p_7\) is \(G_7 \rightarrow (D_7, C_7)\) such that \(G_7 = \{(v_1, v_2, v_3), (v_1, v_2, v_1), (v_2, v_3), (v_3, v_2)\}, \psi_{G_7}\) where \(\psi_{G_7}(v_1) = [s, q_2], \psi_{G_7}(v_2) = W_2, \psi_{G_7}(v_3) = X,\) and \(D_7 = \{\{w_i | i = 1, 2, 3, 4\}, \{e_j | j = 1, 2, \ldots, 8\}\}, \psi_{D_7}\) where \(e_1 = (w_1, w_2), e_2 = (w_2, w_1), e_3 = (w_2, w_3), e_4 = (w_3, w_2), e_5 = (w_2, w_4), e_6 = (w_4, w_2), e_7 = (w_3, w_4), e_8 = (w_4, w_3), \psi_{D_7}(w_1) = [s, \sharp], \psi_{D_7}(w_3) = t,\) \(\text{and} \psi_{D_7}(w_4) = W_2,\) \(\text{and} \)\(C_7 = \{(*, v_1, w_1, \text{in}), (*, v_1, w_1, \text{out}), (*, v_3, w_3, \text{in}), (*, v_3, w_3, \text{out})\}.\) That is, production \(p_7\) is the \(P_{10}\) type.

If there exists \(\delta_A(q_{A_3}, s) = q_2\) and \(\delta_A(q_{A_4}, t) = q_F\) such that \(q_F\) is in \(F_A\) then following productions \(p_8\) and \(p_9\) is added to \(P_e: p_8\) is \(G_8 \rightarrow (D_8, C_8)\) such that \(G_8 = \{(v_1, v_2, v_3), (v_1, v_2), (v_2, v_1), (v_2, v_3), (v_3, v_2)\}, \psi_{G_8}\) where \(\psi_{G_8}(v_1) = [s, q_2], \psi_{G_8}(v_2) = W_2, \psi_{G_8}(v_3) = X,\) and \(D_8 = \{\{w_i | i = 1, 2, 3, 4\}, \{e_j | j = 1, 2, \ldots, 8\}\}, \psi_{D_8}\) where \(e_1 = (w_1, w_2), e_2 = (w_2, w_1), e_3 = (w_2, w_3), e_4 = (w_3, w_2), e_5 = (w_2, w_4), e_6 = (w_4, w_2), e_7 = (w_3, w_4), e_8 = (w_4, w_3), \psi_{D_8}(w_1) = [s, \sharp], \psi_{D_8}(w_2) = [t, q_F], \psi_{D_8}(w_3) = t,\) \(\text{and} \psi_{D_8}(w_4) = W_3,\) \(\text{and} \)\(C_8 = \{(*, v_1, w_1, \text{in}), (*, v_1, w_1, \text{out}), (*, v_1, w_1, \text{out}), (*, v_3, w_3, \text{in}), (*, v_3, w_3, \text{out})\}.\) That is, production \(p_8\) is the \(P_{12}\) type.
labeled grid graphs for all elements in $A$ in order of symbols of strings accepted by automaton $B$. By productions constructed above, row labels are derived in order of symbols of strings accepted by automaton $B$. Similarly, column labels are derived in order of symbols of strings accepted by automaton $B$.

Thus, we obtain a labeled grid graph grammar $G_e$ that $L(G_e)$ contains only labeled grid graphs for all elements in $L(e)$. \[\square\]

**Proposition 3.** Let $\Sigma$ and $\Gamma$ be finite alphabets. If $L$ is the set of graphs generated by a labeled grid graph grammar over $\Sigma \times \Gamma$, then the set of two-dimensional rectangular arrays corresponded to $L$ is denoted by a pair regular expression over $\Sigma \times \Gamma$.

**Proof.** Let $L$ be the set generated by a labeled grid graph grammar $G=(\Sigma_G, \Gamma_G, P_G, S_G)$ over $\Sigma \times \Gamma$.

Here we consider types of productions in $P_G$. Productions that are types of $P_1$, $P_2$, and $P_3$ decide columns labels. Productions that are types of $P_6$, $P_6-2$, $P_{12}$, $P_{17}$, and $P_{18}$ decide rows labels. The other productions are used for inheriting row labels and column labels from neighbor nodes. Accordingly, we construct a regular expression for column strings from $P_1$, $P_2$, and $P_3$. Furthermore we also construct a regular expression for row strings from $P_6$, $P_6-2$, $P_{12}$, $P_{17}$, and $P_{18}$.

Firstly we construct a nondeterministic finite automaton $B = (Q_B, \Gamma, \delta_B, q_{B0}, F_B)$ as follows. Let $Q_B = \{q_{B0}\}$, and let $F_B$ be an empty set. If there exists a production $p_1$ in $P_G$ that is the $P_1$ type in Figure 5, then the state for $q$ is added to $Q_B$ and the set of $\delta_B(q_{B0}, s)$. If there exists a production $p_2$ in $P_G$ that is the $P_2$ type in Figure 5, then the state for $r$ is added to $Q_B$ and the set of $\delta_B(q, t)$. If there exists a production $p_3$ in $P_G$ that is the $P_3$ type in Figure 5, then the state for $q$ is added to $F_B$.

Next we construct a nondeterministic finite automaton $A = (Q_A, \Sigma, \delta_A,$
$q_{A0}, F_A$) as follows. Let $Q_A$ be $\{q_{A0}\}$, and let $F_A$ be an empty set. If there exists a production $p_4$ in $P_G$ that is the $P_6$ type in Figure 5, then the state for $q$ is added to $Q_A$ and the set of $\delta_A(q_{A0}, s)$. If there exists a production $p_5$ in $P_G$ that is the $P_{6-2}$ type in Figure 7, then the state for $q$ is added to $Q_A$ and the set of $\delta_A(q_{A0}, s)$. If there exists a production $p_6$ in $P_G$ that is the $P_{12}$ type in Figure 5, then the state for $r$ is added to $Q_A$ and the set of $\delta_A(q, t)$. If there exists a production $p_7$ in $P_G$ that is the $P_{17}$ type in Figure 6, then the state for $r$ is added to $Q_A$ and the set of $\delta_A(q, t)$. If there exists a production $p_8$ in $P_G$ that is the $P_{18}$ type in Figure 6, then the state for $q$ is added to $F_A$.

To finish this proof, we construct a pair regular expression $e = \langle R, C \rangle$ that $R$ is a regular expression corresponding to $A$ and $C$ is a regular expression corresponding to $B$.

**Theorem 1.** Let $\Sigma$ and $\Gamma$ be finite alphabets. The languages generated by labeled grid graph grammars over $\Sigma \times \Gamma$ are the languages denoted by pair regular expressions over $\Sigma \times \Gamma$.

**Proof.** This proof is shown by Proposition 2 and 3.

We show an example of a labeled grid graph grammar.

**Example 9.** Let $\Sigma = \{d, h, i, m\}$ be an alphabet of row labels, $\Gamma = \{s, k, t\}$ be an alphabet of column labels, and $L_1 = \langle h(i + d + m)^*, (skt)^+ \rangle$ be a pair regular expression over $\Sigma$ and $\Gamma$ as shown in Example 7.

We consider a construction of a labeled grid graph grammar $GG_{st}$ for $L_1$. We now construct $GG_{st}$ by using the method as shown in the proof of Proposition 2.

Firstly, we construct an automaton $A$ that is equivalent to $h(i + d + m)^*$ as follows. An automaton $A$ is $(Q_A, \Sigma, \delta_A, q_{A0}, F_A)$ where $Q_A = \{q_{A0}, q_{A1}\}$, $\delta_A(q_{A0}, h) = q_{A1}, \delta_A(q_{A1}, i) = q_{A1}, \delta_A(q_{A1}, d) = q_{A1}, \delta_A(q_{A1}, m) = q_{A1}$, and $F_A = \{q_{A1}\}$.

Similarly, we construct an automaton $B$ that is equivalent to $(skt)^+$ as follows. An automaton $B$ is $(Q_B, \Gamma, \delta_B, q_{B0}, F_B)$ where $Q_B = \{q_{B0}, q_{B1}, q_{B2}, q_{B3}\}$, $\delta_B(q_{B0}, s) = q_{B1}, \delta_B(q_{B1}, k) = q_{B2}, \delta_B(q_{B2}, t) = q_{B3}, \delta_B(q_{B3}, s) = q_{B1}$, and $F_B = \{q_{B3}\}$.

Here we construct $GG_{st}$ as follows. A labeled grid graph grammar $GG_{st}$ for $L_1$ is $(\Sigma_{st}, \Gamma_{st}, P_{st}, S_{st})$ where, $\Sigma_{st} = \{S, A, B, C, D_1, D_2, F, X, \sharp, E_1, E_2, W_1, W_2, W_3\} \cup \{[s, t]|s \in \Sigma \text{ and } t \in \Gamma\} \cup \{[s, q]|s \in \Sigma \text{ and } q \in Q_A\} \cup \{[s, q]|s \in \Gamma \text{ and } q \in Q_B\} \cup \Sigma \cup \Gamma \cup \{([s, t]|s \in \Sigma \text{ and } t \in \Gamma\} \cup \{(s, t)|s \in \Sigma \cup \{\sharp\} \text{ and } t \in \Gamma \cup \{\sharp\}\}; \Gamma_{st} = \{(s, q)|s \in \Sigma \cup \{\sharp\} \text{ and } q \in \Gamma \cup \{\sharp\}\}; P_{st}$ is a set
of productions; and $S_{st} = \{v_1, \emptyset, \psi_S\}$ where $\psi_S(v_1) = S$.

$P_{st}$ is constructed as follows. Productions of $P_1$ type, $P_2$ type, $P_3$ type, $P_6$ type, $P_{6-2}$ type, $P_{12}$ type, $P_{17}$ type, and $P_{18}$ type are prepared based on automata $A$ and $B$. Productions of $P_7$ type, $P_{7-2}$ type, $P_{7-3}$ type, $P_{7-4}$ type, $P_8$ type, $P_9$ type, $P_{10}$ type, $P_{13}$ type, $P_{13-2}$ type, $P_{13-3}$ type, $P_{13-4}$ type, $P_{14}$ type, $P_{15}$ type, $P_{16}$ type, $P_{19}$ type, $P_{20}$ type, $P_{21}$ type, $P_{21-2}$ type, and $P_{22}$ type are prepared for all symbols $\Sigma$ and $\Gamma$. Productions of $P_4$ type, $P_5$ type, and $P_6$ type are added to $P_{st}$.

We illustrate the derivation of the labeled grid graph for $a(\hat{s}_2)$ in Example 7 based on $GG_{st}$ in Figure 9.

5. Maps for Pictures

We here define a mapping from two-dimensional rectangular arrays to pictures.

**Definition 9.** Let $\Sigma$ and $\Gamma$ be two finite alphabets, $\Theta$ be a finite alphabet. Let $a$ be a two-dimensional rectangular array in $RA_{\Sigma \times \Gamma}$, $m$ be the height of $a$, and $n$ be the width of $a$. A $\pi: \Sigma \times \Gamma \rightarrow \Theta$ is a mapping from two-dimensional rectangular arrays to pictures. The projection by mapping $\pi$ of $a$ is the picture $p$ in $\Theta^{**}$ such that $p(i, j) = \pi(s_i, g_j)$ for each cell $(s_i, g_j)$ of $a$, $i = 1, 2, ..., m$ and $j = 1, 2, ..., n$.

We now give examples of the projection.

**Example 10.** Let $\alpha_1$ be the two-dimensional rectangular array shown in Example 3 and $\pi_1$ be the mapping such that $\pi_1(x, 0) = 1$ and $\pi_1(y, 1) = 0$. Then the picture $\pi(\alpha_1)$ is the following.

$$
\pi_1(\alpha_1) = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0
\end{bmatrix}
$$

**Example 11.** We illustrate a projection by mapping $\pi_2$ of $a(s_2)$ in Example 7 such that $\pi_2(h, s) =$ “symbol”, $\pi_2(h, k) =$ “kind”, $\pi_2(h, t) =$ “type”, $\pi_2(i, s) =$ string, $\pi_2(i, k) =$ variable, $\pi_2(i, t) =$ integer, $\pi_2(d, s) =$ string, $\pi_2(d, k) =$ variable, $\pi_2(d, t) =$ double, $\pi_2(m, s) =$ string, $\pi_2(m, k) =$ method, and $\pi_2(m, t) =$ types.

The fields “string”, “variable”, “integer”, “double”, “method”, and “types” have values or are empty.
Figure 9: The derivation for the labeled grid graph representing $s_2$ based on $GG_{st}$
Figure 10: A projection by mapping $\pi_2$ of $a(s_2)$ with values

<table>
<thead>
<tr>
<th>Level 0</th>
<th>Level 1</th>
<th>Level 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>symbol</td>
<td>kind</td>
<td>type</td>
</tr>
<tr>
<td>a</td>
<td>var</td>
<td>int</td>
</tr>
<tr>
<td>b</td>
<td>var</td>
<td>double</td>
</tr>
<tr>
<td>setA</td>
<td>method</td>
<td>int</td>
</tr>
<tr>
<td>F</td>
<td>method</td>
<td>void</td>
</tr>
<tr>
<td></td>
<td>x</td>
<td>var</td>
</tr>
<tr>
<td></td>
<td>y</td>
<td>var</td>
</tr>
</tbody>
</table>

Figure 10 illustrates a picture added some values for a projection by mapping $\pi_2$ of $s_2$. Each field in Figure 10 is empty or has a value that is “var”, “int”, “a”, “b”, “setA” etc.

We expect that various types of tables are formalized by pair regular expressions as well as Examples 7, 9, and 11. Furthermore we also expect that structures of those tables are identified syntactically based on labeled grid graph grammars.

6. Conclusions

We defined two-dimensional regular expressions by pairs of one-dimensional regular expressions for representing row and column structures. Furthermore we constructed labeled grid graph grammars for pair strings representing pair regular expressions. We expect that various types of tables are formalized by pair regular expressions and identified based on labeled grid graphs for them.

Acknowledgements

The authors would like to thank Professor Kimio Sugita of Tokai University for valuable discussions and helpful comments.
References


