Simulating Parallel Internal Column Contextual Array Grammars Using Two-Dimensional Parallel Restarting Automata with Multiple Windows

Abhisek Midya[1(✉)], Frits Vaandrager[1], D. G. Thomas[2], and Chandrima Ghosh[3]

[1] Institute for Computing and Information Sciences, Radboud University, Nijmegen, The Netherlands
abhisekmidyacse@gmail.com, F.Vaandrager@cs.ru.nl

[2] Department of Science and Humanities, Saveetha School of Engineering, Chennai, India
dgthomasmcc@yahoo.com

[3] Computer Science and Engineering, Presidency University, Bangalore, India
chandrimaghosh211098@gmail.com

Abstract. The connection between picture languages and restarting automata has been established in Otto (2014). An interesting class of picture languages generated by parallel contextual array grammars was studied with application in image generation and analysis in Subramanian et al. (2008). In this paper, we introduce a variant of two dimensional restarting automata that accepts a subclass of parallel internal contextual array languages. We show that these automata can simulate parallel internal column contextual array grammars in reverse order.

Keywords: Parallel internal column contextual array grammars · Membership problem · Restarting automaton

1 Introduction

Syntactic approaches, on account of their structure-handling capability, have played an important role in the problem of description of picture patterns considered as connected digitized, finite arrays of symbols. Using the techniques of formal string language theory, various types of picture or array grammars have been introduced and investigated in [3,9,10,24,25]. Most of the array grammars are based on Chomskian string grammars. Some recent results on picture languages can be found in [2,8,18]. Another interesting class of string grammars, called the class of contextual grammars, was proposed by Marcus in [16]. A contextual grammar defines a string language by starting from a given finite set of strings and adjoining iteratively pairs of strings (called contexts) associated

T. Lukić et al. (Eds.): IWCIA 2020, LNCS 12148, pp. 106–122, 2020.
https://doi.org/10.1007/978-3-030-51002-2_8

to sets of words (called selectors), to the strings already obtained. These contextual grammars [6,23] are known to provide new approaches for a number of basic problems in formal language theory. Recently, extension of string contextual grammars to array structures has been attempted in [1,7,15,23]. A new method of description of pictures of digitized rectangular arrays, through parallel contextual array grammars, was introduced [4,26]. In this paper, we establish a relationship between *two dimensional restarting automata* and *parallel internal column contextual array grammars*.

The concept of restarting automaton was introduced in [14], in order to model the 'analysis by reduction', which is a technique used in linguistics to analyze sentences of natural languages. Analysis by reduction consists of stepwise simplifications (reductions) of a given (lexically disambiguated) extended sentence unless a correct simple sentence is obtained. A word is accepted until an error is found - the process continues until either the automaton accepts or an error is detected. Each simplification replaces a short part of the sentence by an even shorter one. The one dimensional restarting automaton contains a finite control unit, a head with a look-ahead window attached to a tape.

It has been shown in [12], that restarting automaton with delete (simply, DRA) can represent the analyzer for characterizing the class of *contextual grammars with regular selector* (CGR). Also [13] showed that restarting automata recognize a family of languages which can be generated by certain type of contextual grammars, called *regular prefix contextual grammars with bounded infix* (RPCGBI).

Here, we focus on two dimensional parallel restarting automata as we are dealing with rectangular picture languages and bring the concept of multiple windows in order to capture the parallel application of rules of parallel internal column contextual array grammars. A two dimensional parallel restarting automaton can delete adjoined sub-arrays in a cycle and followed by restart (*DEL-RST*). We exploit the *DEL-RST* operation to reverse the adjoining contexts that take place in a derivation of a parallel internal column contextual array grammar. We use two dimensional parallel restarting automaton with multiple windows to simulate parallel internal column contextual array grammars in reverse order.

The *membership problem* for a language asks whether, for a given grammar G and a string w, w belongs to the language generated by G or not.

The remainder of this paper is organized as follows. Section 2 describes the basic classes of contextual grammars in more detail which is followed by an example in Subsect. 2.1. Section 3 presents the new variant of two dimensional parallel restarting automata with multiple windows. In Sect. 4, we describe the connection between parallel internal column contextual array grammars and two dimensional parallel restarting automata with multiple windows, also an example is given in Subsect. 4.1 for better understanding. Subsection 4.2 presents some interesting properties of the proposed automata and in Subsect. 4.3 we discuss about the complexity of membership problem for parallel internal column contextual array languages, also we introduce some new definitions. Section 5 concludes the work and shows a future direction of work.

2 Preliminaries

Let V be a finite alphabet. We write V^* for the set of all finite strings over V, which includes the empty string λ. An *image* or *picture* over V is a rectangular $m \times n$ array of elements of V or in short $[a_{ij}]_{m\times n}$. The set of all images over V is denoted by V^{**}. A *picture language* or *two dimensional language* over V is a subset of V^{**}. We define $V^{m,n} = \{A \in V^{**} \mid A \text{ has } m \text{ rows and } n \text{ columns}\}$. If $a \in V$, then $[a^{m,n}]$ is the array over $\{a\}$ with m rows and n columns. In this paper, Λ denotes any empty array. The notion of *column concatenation* is defined as follows: if A and B are two arrays where

$$A = \begin{bmatrix} a_{1,j} & \dots & a_{1,k} \\ a_{2,j} & \dots & a_{2,k} \\ \dots & \dots & \dots \\ a_{l,j} & \dots & a_{l,k} \end{bmatrix}, B = \begin{bmatrix} b_{1,m} & \dots & b_{1,n} \\ b_{2,m} & \dots & b_{2,n} \\ \dots & \dots & \dots \\ b_{l,m} & \dots & b_{l,n} \end{bmatrix} \text{ then } A\Phi B = \begin{bmatrix} a_{1,j} & \dots & a_{1,k} & b_{1,m} & \dots & b_{1,n} \\ a_{2,j} & \dots & a_{2,k} & b_{2,m} & \dots & b_{2,n} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ a_{l,j} & \dots & a_{l,k} & b_{l,m} & \dots & b_{l,n} \end{bmatrix}.$$

If L_1, L_2 are two picture languages over an alphabet V, the *column concatenation* $L_1 \Phi L_2$ of L_1, L_2 is defined by $L_1 \Phi L_2 = \{A\Phi B \mid A \in L_1, B \in L_2\}$. Column concatenation is only defined for pictures that have the same number of rows. Note that operation Φ is associative. If X is an array, the set of all sub-arrays of X is denoted by $sub(X)$. We now recall the notion of column array context [4,26].

Definition 1. *Let V be an alphabet. A* column array context *c over V is of the form*

$$c = \left[\begin{smallmatrix} u_1 \\ u_2 \end{smallmatrix}\right] \psi \left[\begin{smallmatrix} v_1 \\ v_2 \end{smallmatrix}\right] \in V^{**}\psi V^{**},$$

where u_1, u_2 are arrays of sizes $1 \times p$, and v_1, v_2 are arrays of sizes $1 \times q$, for some $p, q \geq 1$, and ψ is a special symbol not in V.

The next definition deals with the parallel internal column contextual operation.

Definition 2. *Let V be an alphabet, C a finite set of column array contexts over V, and $\varphi : V^{**} \to 2^C$ a mapping, called choice mapping. For an array $A = \begin{bmatrix} a_{1,j} & \dots & a_{1,k} \\ a_{2,j} & \dots & a_{2,k} \\ \dots & \dots & \dots \\ a_{l,j} & \dots & a_{l,k} \end{bmatrix}$, $j \leq k, a_{ij} \in V$, we define $\hat{\varphi} : V^{**} \to 2^{V^{**}\psi V^{**}}$ such that $L\psi R \in \hat{\varphi}(A)$, where*

$$L = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_l \end{bmatrix}, R = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_l \end{bmatrix},$$

and $c_i = \left[\begin{smallmatrix} u_i \\ u_{i+1} \end{smallmatrix}\right] \psi \left[\begin{smallmatrix} v_i \\ v_{i+1} \end{smallmatrix}\right] \in \varphi \left[\begin{smallmatrix} a_{i,j} & \dots a_{i,k} \\ a_{i+1,j} & \dots a_{i+1,k} \end{smallmatrix}\right]$, with $c_i \in C, (1 \leq i \leq l-1)$, not all need to be distinct. Given an array $X = [a_{ij}]$ of size $m \times n$, $a_{ij} \in V, X = X_1 \Phi X_2 \Phi X_3$ where

$$X_1 = \begin{bmatrix} a_{1,1} & \dots & a_{1,p-1} \\ a_{2,1} & \dots & a_{2,p-1} \\ \vdots & \vdots & \vdots \\ a_{m1} & \dots & a_{m,p-1} \end{bmatrix}, X_2 = \begin{bmatrix} a_{1,p} & \dots & a_{1,q} \\ a_{2,p} & \dots & a_{2,q} \\ \vdots & \vdots & \vdots \\ a_{m,p} & \dots & a_{m,q} \end{bmatrix}, X_3 = \begin{bmatrix} a_{1,q+1} & \dots & a_{1,n} \\ a_{2,q+1} & \dots & a_{2,n} \\ \vdots & \vdots & \vdots \\ a_{m,q+1} & \dots & a_{m,n} \end{bmatrix}$$

and $1 \leq p \leq q \leq n$, we write $X \Rightarrow_{in} Y$ if $Y = X_1 \Phi L \Phi X_2 \Phi R \Phi X_3$ such that $L\psi R \in \hat{\varphi}(X_2)$. Here L and R are called left and right contexts respectively.

We say that Y is obtained from X by parallel internal column contextual operation ($\Rightarrow$).

Now we consider the notion of parallel internal column contextual array grammar [4,26].

Definition 3. *A* parallel internal column contextual array grammar (PICCAG) *is an ordered system $G = (V, \mathbf{A}, C, \varphi)$, where V is an alphabet, $\mathbf{A}$ is a finite subset of V^{**} called the axiom set, C is a finite set of column array contexts over V, $\varphi : V^{**} \to 2^C$ is the choice mapping which performs the parallel internal column contextual operation. When φ is omitted we call G a parallel internal contextual array grammar without choice.*
We already discussed the notion of $X \Rightarrow_{in} Y$ in the previous definition. Here we denote by $\Rightarrow^*_{in}$ the reflexive transitive closure of $\Rightarrow_{in}$. The parallel internal column contextual array language (PICCAL) *generated by G is defined as the set $L_{in}(G) = \{Y \in V^{**} \mid \exists X \in \mathbf{A}$ such that $X \Rightarrow^*_{in} Y\}$.*

2.1 Example

Let $G = (V, \mathbf{A}, C, \varphi)$ be a parallel internal column contextual array grammar (*PICCAG*) where $V = \{a, b\}$, $\mathbf{A} = \left\{ B = \begin{bmatrix} a & a & b & b \\ a & a & b & b \\ b & b & a & a \\ b & b & a & a \end{bmatrix} \right\}$,
$C = \left\{ \begin{bmatrix} a \\ b \end{bmatrix} \psi \begin{bmatrix} b \\ a \end{bmatrix}, \begin{bmatrix} a \\ a \end{bmatrix} \psi \begin{bmatrix} b \\ b \end{bmatrix}, \begin{bmatrix} b \\ b \end{bmatrix} \psi \begin{bmatrix} a \\ a \end{bmatrix} \right\}$, φ is a choice mapping satisfying

$$\varphi \begin{bmatrix} a & b \\ b & a \end{bmatrix} = \begin{bmatrix} a \\ b \end{bmatrix} \psi \begin{bmatrix} b \\ a \end{bmatrix}, \varphi \begin{bmatrix} a & b \\ a & b \end{bmatrix} = \begin{bmatrix} a \\ a \end{bmatrix} \psi \begin{bmatrix} b \\ b \end{bmatrix}, \varphi \begin{bmatrix} b & a \\ b & a \end{bmatrix} = \begin{bmatrix} b \\ b \end{bmatrix} \psi \begin{bmatrix} a \\ a \end{bmatrix}.$$

Then,
$L_{in}(G) = \left\{ \begin{bmatrix} (a^n \ b^n)_m \\ (b^n \ a^n)_m \end{bmatrix} \mid n \geq 2, m = 2 \right\}$, where $a^n = aa...a$ (n times) and $a_m = \underset{a}{\vdots}$, with m rows. A simple derivation of a member of $L_{in}(G)$ is as follows:

$$B = \begin{bmatrix} a & a & b & b \\ a & a & b & b \\ b & b & a & a \\ b & b & a & a \end{bmatrix} \Rightarrow \begin{bmatrix} a & a & a & b & b & b \\ a & a & a & b & b & b \\ b & b & b & a & a & a \\ b & b & b & a & a & a \end{bmatrix} = \begin{bmatrix} (a^3 \ b^3)_2 \\ (b^3 \ a^3)_2 \end{bmatrix} \in L_{in}(G).$$

Now, if we consider $a =$ white box and $b =$ black box, we get a nice rectangular picture, see Fig. 1 and Fig. 2.

3 Two Dimensional Parallel Restarting Automata with Multiple Windows

It is interesting to find the connection between picture languages with deterministic two-dimensional three-way ordered restarting automata (det-2D-3W-ORWW) and deterministic two-dimensional extended two-way ordered restarting automata (det-2D-x2W-ORWW) in [19,20].

In this section we present a variant of two dimensional restarting automaton called a two dimensional parallel restarting automaton with multiple windows

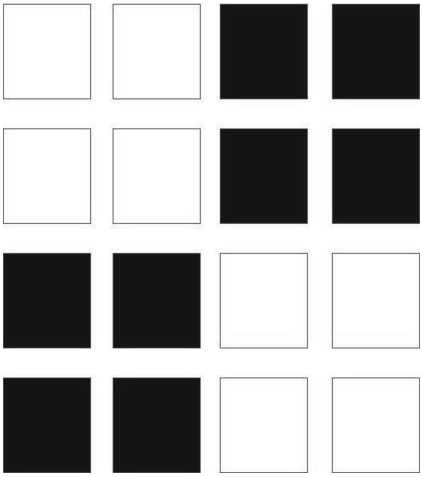

Fig. 1. A rectangular picture of size 4×4

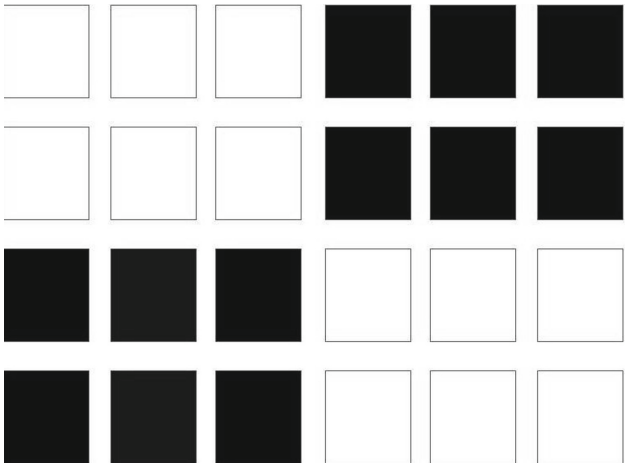

Fig. 2. A rectangular picture of size 4×6

(2D-PRA-W_m) in order to simulate *PICCAL* in reverse order. Here we introduce multiple windows to deal with the parallel application of rules of parallel internal column contextual array grammars. This automaton works when *PICCAG* is given.

We describe the basic working nature of 2D-PRA-W_m. It contains finite control unit and multiple tapes and each tape is associated with individual head and they work in a parallel way. At several points, it cuts-off sub-arrays from each sub-window using DEL operation followed by *restart* (RST) operation, that

is, DEL-RST. Here in each sub-window the same number of columns are deleted, this happens in exactly the same positions.

All the heads move together right along the individual tape until it takes any DEL-RST operation. RST implies that the restarting automaton places all the windows over the left border of the individual tape and it completes one *cycle*. After performing a DEL-RST operation, the restarting automaton is unable to remember any step of computation that was performed already.

Let W be an array of size $m \times n$, with $m \geq 2$. Assume

$$W = \begin{bmatrix} a_{1,1} & \cdots & a_{1,p-1} \\ a_{2,1} & \cdots & a_{2,p-1} \\ \vdots & \vdots & \vdots \\ a_{m1} & \cdots & a_{m,p-1} \end{bmatrix}$$

$$\text{Now W can be viewed as } [W]^{m,n} = \begin{bmatrix} [W_i]^{2,k} \\ [W_{i+1}]^{2,k} \\ \vdots \\ [W_f]^{2,k} \end{bmatrix} \textit{where}$$

$$[W_1]^{2,n} = \begin{bmatrix} a_{1,1} & \cdots & a_{1,n} \\ a_{2,1} & \cdots & a_{2,n} \end{bmatrix}, [W_2]^{2,n} = \begin{bmatrix} a_{2,1} & \cdots & a_{2,n} \\ a_{3,1} & \cdots & a_{3,n} \end{bmatrix}, [W_3]^{2,n} = \begin{bmatrix} a_{3,1} & \cdots & a_{3,n} \\ a_{4,1} & \cdots & a_{4,n} \end{bmatrix},$$

$$[W_{m-1}]^{2,n} = \begin{bmatrix} a_{m-1,1} & \cdots & a_{m-1,n} \\ a_{m,1} & \cdots & a_{m,n} \end{bmatrix}$$

Now we present the concept of super window and sub-window.

$$[W]^{f+1,k} = \begin{bmatrix} [W_i]^{2,k} \\ [W_{i+1}]^{2,k} \\ \vdots \\ [W_f]^{2,k} \end{bmatrix}$$

where $[W]^{f+1,k}$ is called the super window (array) of size $((f+1) \times k)$ which contains sub-windows (arrays) $[W_i]^{2,k}$ of sizes $(2 \times k)$ where $[W_i]^{2,k} = ([\lhd^{2,1}]\Phi V^{2,k-1}) \cup (V^{2,k}) \cup (V^{2,k-1}\Phi[\rhd^{2,1}]) \cup ([\lhd^{2,1}]\Phi V^{2,k-2}\Phi[\rhd^{2,1}])$. Here f denotes the number of sub-windows. The second row of each ith sub-window $[W_i]^{2,k}$ overlaps with the first row of each $(i+1)$th sub-window $[W_{i+1}]^{2,k}$. Now we show the transition function δ of 2D-PRA-W$_m$ for set of sub-windows. Here $[\lhd^{2,1}], [\rhd^{2,1}]$ denote one column and 2 rows of $\lhd, \rhd$ marker respectively.

Suppose $W = \begin{bmatrix} a & a & a & a & b & b & b & b \\ a & a & a & a & b & b & b & b \\ b & b & b & b & a & a & a & a \\ b & b & b & b & a & a & a & a \end{bmatrix}$.

Now W can be viewed in the following way with the help of sub window and super window. The size of each sub window depends on the given grammar. Interestingly, size of super window depends on the number of sub window. (Discussed in detail in Theorem 1). We have shown below sub window $[W_1], [W_2], [W_3]$ and the super window $[W]$ which contains $[W_1], [W_2], [W_3]$.

$$[W_1] = \begin{bmatrix} a & a & a & a & b & b \\ a & a & a & a & b & b \end{bmatrix}, [W_2] = \begin{bmatrix} a & a & a & a & b & b \\ b & b & b & b & a & a \end{bmatrix}, [W_3] = \begin{bmatrix} b & b & b & b & a & a \\ b & b & b & b & a & a \end{bmatrix},$$

$$[W] = \begin{bmatrix} [W_1] \\ [W_2] \\ [W_3] \end{bmatrix},$$

Let $G = PICCAG = (V, \mathbf{A}, C, \varphi)$.

Definition 4. *A* two dimensional parallel restarting automaton with multiple window (2D-PRA-W_m)*, is given through a* 6*-tuple,* $M = (Q, V, \triangleleft, \triangleright, q_0, \delta)$ *where*

- Q *is a finite set of states,*
- V *is the input alphabet,*
- $\triangleleft, \triangleright$ *are left border, right border markers respectively,*
- $q_0 \in Q$ *is the initial state,*
- $\delta : Q \times [W_i]^{2,k} \rightarrow 2^{((Q \times \{MVR, DEL-RST\}) \cup \{Accept, Reject\})}$ *is the transition function. This function describes four different types of transition steps:*
  - *MVR:* $(q, MVR) \in \delta(q, ([W_i]^{2,k}, [W_{i+1}]^{2,k}, ..., [W_f]^{2,k}))$*. Thus each sub-window of* M *sees the sub-array of size* $2 \times k$*. Applying the transition function* δ*, each sub-window of* M *moves through left to right using* MVR *until it takes DEL-RST or* $\triangleright \notin [W_i]^{2,k}$*.*
  - *DEL-RST:* $(q_0, DEL - RST) \in \delta(q, ([W_i]^{2,k}, [W_{i+1}]^{2,k}, ..., [W_f]^{2,k}))$ *: For possible contents of each sub-window, it deletes a subarray and causes* M *to move its sub-window to the left border marker* $\triangleleft$ *and re-enters into the initial state* q_0*.*
  - *ACCEPT: Accept* $\in \delta(q, ([W_i]^{2,k}, [W_{i+1}]^{2,k}, ..., [W_f]^{2,k}))$*, it gets into an accepting state.*
  - *REJECT: Reject* $\in \delta(q, ([W_i]^{2,k}, [W_{i+1}]^{2,k}, ..., [W_f]^{2,k})) = \emptyset$ *(i.e., when* δ *is undefined), then* M *will reject the input.*
- *Let* $P \in V^{r,s}$ *is accepted by* $2D - PRA - W_m$ M*, if there is a computation, which starts from the initial configuration* $q_0[\triangleleft^{r,1}]\Phi P \Phi[\triangleright^{r,1}]$*, and reaching the Accept state. By* $L(M)$*, we denote the language consisting of all arrays accepted by* M*. In formal notation* $L(M) = \{P \in V^{r,n} \mid q_0[\triangleleft^{r,1}]\Phi P^{r,s}\Phi[\triangleright^{r,1}] \vdash^* Accept\}$*. Here* $[\triangleleft^{r,1}], [\triangleright^{r,1}]$ *denote one column and* r *rows of* $\triangleleft, \triangleright$ *marker respectively.*

In general, the 2D-PRA-W_m is nondeterministic, that is, there can be two or more instructions with the same left-hand side. If this is not the case, the automaton is deterministic.

Proposition 1 *(Error preservation of* 2D-PRA-W_m*). If* $[W]^{f+1,k} \vdash_M^* [W']^{f+1,k'}$ *and* $[W]^{f+1,k} \notin L(M)$ *then* $[W']^{f+1,k'} \notin L(M)$ *where* $[W]^{f+1,k}, [W']^{f+1,k'} \in V^{**}, k > k'$*.*

4 2D-PRA-W_m and PICCAG

Before we analyze the relationship between 2D-PRA-W_m and *PICCAG*, which is the objective of this section, we first need to understand the relationship of DRA with string contextual grammars [12].

External contextual grammars were introduced by Marcus in 1969 [16]. Internal contextual grammars [22] produce strings starting from an *axiom* and in each step *left context* and *right context* are adjoined to the string based on certain string called selector present as a sub-string in the derived string. u, v are called *left context* and *right context* respectively. For more details on contextual grammars, we refer to [21].

- The selector in a contextual grammar can be of arbitrary type in nature, like regular, context-free etc, but the strings u, v are finite.
- Normal DRA works in the opposite way of contextual grammars in accepting strings [12]. In a normal DRA M, w is given as an input, it checks the items of the window with the contextual grammar G that any given rule has been used or not.
- If it finds that any rule has been used then the automaton deletes the left and right context u, v and takes the RST operation, otherwise takes MVR and checks whether any rule in G can be applied.
- In this way, the automaton simulates the derivation of contextual grammar in reverse order and if the input string can be reduced back to the axiom B[1], it implies that the string w can be generated using the given grammar G, thus $w \in L(G)$.
- Here the *size* of the tape of the automaton M is same as the size of the array w. Step by step, the automaton M only deletes subarrays of w, so the size of the tape becomes smaller and smaller.

In this paper, we adapt the working nature of DRA to solve membership problem for *PICCAL*. We show that the membership problem for *PICCAL* is solvable by the introduced 2D-PRA-W$_m$. The paradigm of this version of 2D-PRA-W$_m$ is closely related to *PICCAG*. A *PICCAG* works just in the *opposite direction* of 2D-PRA-W$_m$. The connection is established based on the following observation. For a *PICCAG* rule $\varphi[x_i] = L_i \psi R_i$, now if we present that in two-dimensional form-

$$\varphi[x_i] = \begin{bmatrix} l_{i,1} & \dots & l_{i,m} \\ l_{i+1,1} & \dots & l_{i+1,m} \end{bmatrix} \psi \begin{bmatrix} R_{i,1} & \dots & R_{i,n} \\ R_{i+1,1} & \dots & R_{i+1,n} \end{bmatrix}$$

where 2D-PRA-W$_m$ has to delete the left context L_i and right context R_i, that is, $\varphi[x_i] = L_i \psi R_i$ is occurred as a subarray in the given input array. In that case, we informally say that a *PICCAG* rule is found in the window as a subarray.

Let M be 2D-PRA-W$_m$. A reduction system induced by M is $RS(M) = (V^{**}, \vdash_M)$. For each *PICCAG* G, we define a reduction system induced by G as $RS(G) = (V^{**}, \Rightarrow_G^{-1})$ where $([W]^{f+1,k} \vdash_M^{-1} [W']^{f+1,k'})$ iff $[W']^{f+1,k'} \Rightarrow_G [W]^{f+1,k}$.

With the above detail we will construct a 2D-PRA-W$_m$ in such a way that if $B \Rightarrow_G^* P$ then $P \vdash_M^* B$ for $P, B \in V^{**}, B \in \mathbf{A}$, thus $RS(G) = RS(M)$. Also $\mathcal{L}$2D-PRA-W$_m$ denotes the class of languages accepted by 2D-PRA-W$_m$.

Theorem 1. *For a PICCAG G, a* 2D-PRA-W$_m$ *automaton M can be constructed in such a way that $RS(G) = RS(M)$ and $L_{in}(G) = L(M)$.*

Proof. Given a *PICCAG* $G = (V, \mathbf{A}, C, \varphi)$ we have to construct a 2D-PRA-W$_m$ automaton $M = (Q, V, \triangleleft, \triangleright, q_0, \delta)$, that accepts $L_{in}(G)$ where

- $Q = \{q_0, q, Accept, Reject\}$
- V is the input alphabet

[1] We consider $\mathbf{A}$ is a singleton axiom set and $B \in A$.

– $\triangleleft, \triangleright$ are left and right borders respectively and $\triangleleft, \triangleright \notin V$
– q_0 is the initial state.

$$[W]^{f+1,k} = \begin{bmatrix} [W_i]^{2,k} \\ [W_{i+1}]^{2,k} \\ \vdots \\ [W_f]^{2,k} \end{bmatrix}$$

– Here number of columns in each window of M will be $k = max(Rule_{max}, k_b + 2)$ where $Rule_{max}$ is the maximum size given rule - $Rule_{max} = max\{|Rule_1|_c, |Rule_2|_c, ..., |Rule_n|_c\}$ where $|Rule_i|_c$ denotes the number of columns in the ith rule and $1 \leq i \leq n, n \geq 1$.
– Let k_b be axiom size. 2 is added there for the left border $\triangleleft$ and the right border $\triangleright$. The reason for 2 is added with k_b is to satisfy the accepting condition - ACCEPT - $Accept \in \delta(q, ([W_i]^{2,k}, [W_{i+1}]^{2,k}, ..., [W_f]^{2,k}))$ where $[W]^{f+1,k} = [\triangleleft^{f+1,1}]\Phi B \Phi[\triangleright^{f+1,1}]$ where $B \in \mathbf{A}$.
– If the rule is $Rule_i = (\varphi[x_i] = [L_i]\psi[R_i])$ where x_i, L_i, R_i are arrays of size $2 \times k, k \geq 1$, then we define $|Rule_i|_c = |x_i|_c + |L_i|_c + |R_i|_c$, $Rule_{max} = max\{|Rule_1|_c, |Rule_2|_c, ..., |Rule_n|_c\}$

Lemma 1. *If the input is $P^{m,n}$, then number of windows will be $m - 1$.*

Proof. Each window will take care of each rule in a parallel way. According to Definition 2, we know that if the input is P of size $m \times n$ then the number of parallel rules will be $m - 1$, from this fact we can conclude this. (see example for better understanding)

- *DEL-RST*: The DEL-RST instruction of the 2D-PRA-W_m for solving membership problem of *PICCAL*, works in the following manner:
  – Now M works in a parallel way on, $(q, ([W_i]^{2,k}, [W_{i+1}]^{2,k}, ..., [W_f]^{2,k}))$ where $i \geq 1$, and applies DEL-RST on each sub-window from state q to arrive at $((q_0, [W'_i]^{2,k'}), (q_0, [W'_{i+1}]^{2,k'}), ..., (q_0, [W'_f]^{2,k'}))$ and eventually reaching $(q_0, ([W'_i]^{2,k'}, [W'_{i+1}]^{2,k'}, ..., [W'_f]^{2,k'}))$ where $[W']^{f+1,k'}, [W'_i]^{2,k'}$ are scattered sub-array of $[W]^{f+1,k}, [W_i]^{2,k}$ respectively and $k > k'$, immediately followed by a RST instruction: RST $\in \delta(q, [W_i]^{2,k})$ for any possible contents $[W_i]^{2,k}$ of the window. If no *PICCAG* rule does belong to window as a subarray and $\triangleright$ does not belong to window ($\triangleright \notin [W_i]^{2,k}$) then the automaton takes MVR operation.
- ACCEPT: $Accept \in \delta(q, ([W_i]^{2,k}, [W_{i+1}]^{2,k}, ..., [W_f]^{2,k}))$ where $[W]^{f+1,k} = [\triangleleft^{f+1,1}]\Phi B \Phi[\triangleright^{f+1,1}]$ where $B \in \mathbf{A}$. Here $[\triangleleft^{f+1,1}], [\triangleright^{f+1,1}]$ denote one column of $\triangleleft, \triangleright$ marker respectively, here we deal with singleton axiom set.
- REJECT: $Reject \in \delta(q, ([W_i]^{2,k}, [W_{i+1}]^{2,k} ..., [W_f]^{2,k})) = \emptyset$. That is when δ is undefined. In other words, when $2D - PRA - W_m$ is unable to take any of the DEL-RST or MVR operation, then the transition becomes undefined.

2D-PRA-W$_{\mathrm{m}}$ simulates the derivation of *PICCAG* in reverse order, in case of any *PICCAG* rule it deletes the left and right contexts using DEL-RST instruction which is defined already. For *PICCAG*, the derivation starts from the axiom to the generated array, the automaton starts the reduction from the generated array to the axiom. If $B \Rightarrow_G^* P$ then $P \vdash_M^* B$ where $P, B \in V^{**}, B \in \mathbf{A}$ is axiom, thus $RS(G) = RS(M)$.

Corollary 1. *The membership problem for PICCAL can be solved by 2D-PRA-W$_{\mathrm{m}}$.*

Proof. We conclude this important result from Theorem 1.

4.1 Example

Consider the *PICCAG* G given in Example 2.1. Suppose $P = \begin{bmatrix} a & a & a & a & b & b & b & b \\ a & a & a & a & b & b & b & b \\ b & b & b & b & a & a & a & a \\ b & b & b & b & a & a & a & a \end{bmatrix}$ is given as an input and we note that $P \in L_{in}(G)$. Now we can construct a 2D-PRA-W$_{\mathrm{m}}$ automaton $M = (Q, V, \triangleleft, \triangleright, q_0, \delta)$, that accepts P where

- $Q = \{q_0, q, Accept, Reject\}$
- V is the input alphabet
- $\triangleleft, \triangleright$ are left and right borders respectively and $\triangleleft, \triangleright \notin V$
- q_0 is the initial state
- The number of columns in each window is $k = 6$ and the number of windows is 3 respectively.
- In the first cycle, rule and $\triangleleft$ are not found in the window and so it takes MVR: $(q, MVR) \in \delta(q, ([W_1]^{2,6}, [W_2]^{2,6}, [W_3]^{2,6}))$, where

$$[W_1]^{2,6} = \begin{bmatrix} \triangleleft & a & a & a & a & b \\ \triangleleft & a & a & a & a & b \end{bmatrix}, [W_2]^{2,6} = \begin{bmatrix} \triangleleft & a & a & a & a & b \\ \triangleleft & b & b & b & b & a \end{bmatrix}, [W_3]^{2,6} = \begin{bmatrix} \triangleleft & b & b & b & b & a \\ \triangleleft & b & b & b & b & a \end{bmatrix},$$

- After taking the MVR operation the elements of windows get changed and M takes $DEL - RST$: $(q_0, ([W_1']^{2,4}, [W_2']^{2,4}, [W_3']^{2,4})) \in \delta(q, [W_1]^{2,6}, [W_2]^{2,6}, [W_3]^{2,6}))$, where $[W_i']^{2,4}$ is the scattered sub-array of $[W_i]^{2,6}, i \geq 1$ and

$$[W]^{4,6} = \begin{bmatrix} [W_1]^{2,6} \\ [W_2]^{2,6} \\ [W_3]^{2,6} \end{bmatrix}, [W']^{4,4} = \begin{bmatrix} [W_1']^{2,4} \\ [W_2']^{2,4} \\ [W_3']^{2,4} \end{bmatrix}$$

$$[W_1]^{2,6} = \begin{bmatrix} a & a & a & a & b & b \\ a & a & a & a & b & b \end{bmatrix}, [W_1']^{2,4} = \begin{bmatrix} a & a & a & b \\ a & a & a & b \end{bmatrix}$$

$$[W_2]^{2,6} = \begin{bmatrix} a & a & a & a & b & b \\ b & b & b & b & a & a \end{bmatrix}, [W_2']^{2,4} = \begin{bmatrix} a & a & a & b \\ b & b & b & a \end{bmatrix}$$

$$[W_3]^{2,6} = \begin{bmatrix} b & b & b & b & a & a \\ b & b & b & b & a & a \end{bmatrix}, [W_3']^{2,4} = \begin{bmatrix} b & b & b & a \\ b & b & b & a \end{bmatrix}$$

- In the next cycle, again M can take $DEL - RST$: $(q_0, ([W_1']^{2,4}, [W_2']^{2,4}, [W_3']^{2,4})) \in \delta(q_0, [W_1]^{2,6}, [W_2]^{2,6}, [W_3]^{2,6}))$ where $[W_i']^{2,4}$ is the scattered sub-array of $[W_i]^{2,6}, i \geq 1$ and

$$[W]^{4,6} = \begin{bmatrix} [W_1]^{2,6} \\ [W_2]^{2,6} \\ [W_3]^{2,6} \end{bmatrix}, [W']^{4,4} = \begin{bmatrix} [W'_1]^{2,4} \\ [W'_2]^{2,4} \\ [W'_3]^{2,4} \end{bmatrix}$$

$$[W_1]^{2,6} = \begin{bmatrix} \lhd & a & a & a & b & b \\ \lhd & a & a & a & b & b \end{bmatrix}, [W'_1]^{2,4} = \begin{bmatrix} \lhd & a & a & b \\ \lhd & a & a & b \end{bmatrix}$$

$$[W_2]^{2,6} = \begin{bmatrix} a & a & a & a & b & b \\ b & b & b & b & a & a \end{bmatrix}, [W'_2]^{2,4} = \begin{bmatrix} \lhd & a & a & b \\ \lhd & b & b & a \end{bmatrix}$$

$$[W_3]^{2,6} = \begin{bmatrix} b & b & b & b & a & a \\ b & b & b & b & a & a \end{bmatrix}, [W'_3]^{2,4} = \begin{bmatrix} \lhd & b & b & a \\ \lhd & b & b & a \end{bmatrix}$$

– In the next cycle, $Accept \in \delta(q_0, [W_1]^{2,6}, [W_2]^{2,6}, [W_3]^{2,6})$ where

$$[W]^{4,6} = \begin{bmatrix} \lhd & a & a & b & b & \rhd \\ \lhd & a & a & b & b & \rhd \\ \lhd & b & b & a & a & \rhd \\ \lhd & b & b & a & a & \rhd \end{bmatrix}$$

In this way, every member of $L_{in}(G)$ is accepted by M. Now we consider the input $P' = \begin{bmatrix} a & a & a & a & b & b & b \\ a & a & a & a & b & b & b \\ b & b & b & b & a & a & a \\ b & b & b & b & a & a & a \end{bmatrix}$ and we note that $P' \notin L_{in}(G)$.

In the first cycle, rule and $\lhd$ are not found in the window, so it takes MVR: $(q, MVR) \in \delta(q, ([W_1]^{2,6}, [W_2]^{2,6}, [W_3]^{2,6}))$, where

$$[W_1]^{2,6} = \begin{bmatrix} \lhd & a & a & a & a & b \\ \lhd & a & a & a & a & b \end{bmatrix}, [W_2]^{2,6} = \begin{bmatrix} \lhd & a & a & a & a & b \\ \lhd & b & b & b & b & a \end{bmatrix}, [W_3]^{2,6} = \begin{bmatrix} \lhd & b & b & b & b & a \\ \lhd & b & b & b & b & a \end{bmatrix},$$

– Now M takes $DEL - RST : (q_0, ([W'_1]^{2,4}, [W'_2]^{2,4}, [W'_3]^{2,4})) \in \delta(q, ([W_1]^{2,6}, [W_2]^{2,6}, [W_3]^{2,6}))$, where $[W'_i]^{2,4}$ is the scattered sub-array of $[W_i]^{2,6}, i \geq 1$ and

$$[W]^{4,6} = \begin{bmatrix} [W_1]^{2,6} \\ [W_2]^{2,6} \\ [W_3]^{2,6} \end{bmatrix}, [W']^{4,4} = \begin{bmatrix} [W'_1]^{2,4} \\ [W'_2]^{2,4} \\ [W'_3]^{2,4} \end{bmatrix}$$

$$[W_1]^{2,6} = \begin{bmatrix} a & a & a & a & b & b \\ a & a & a & a & b & b \end{bmatrix}, [W'_1]^{2,4} = \begin{bmatrix} a & a & a & b \\ a & a & a & b \end{bmatrix}$$

$$[W_2]^{2,6} = \begin{bmatrix} a & a & a & a & b & b \\ b & b & b & b & a & a \end{bmatrix}, [W'_2]^{2,4} = \begin{bmatrix} a & a & a & b \\ b & b & b & a \end{bmatrix}$$

$$[W_3]^{2,6} = \begin{bmatrix} b & b & b & b & a & a \\ b & b & b & b & a & a \end{bmatrix}, [W'_3]^{2,4} = \begin{bmatrix} b & b & b & a \\ b & b & b & a \end{bmatrix}$$

In the next cycle, again M can take $DEL - RST : (q_0, ([W'_1]^{2,4}, [W'_2]^{2,4}, [W'_3]^{2,4})) \in \delta(q_0, ([W_1]^{2,6}, [W_2]^{2,6}, [W_3]^{2,6}))$ where $[W'_i]^{2,4}$ is the scattered sub-array of $[W_i]^{2,6}, i \geq 1$ and

$$[W]^{4,6} = \begin{bmatrix} [W_1]^{2,6} \\ [W_2]^{2,6} \\ [W_3]^{2,6} \end{bmatrix}, [W']^{4,4} = \begin{bmatrix} [W'_1]^{2,4} \\ [W'_2]^{2,4} \\ [W'_3]^{2,4} \end{bmatrix}$$

$$[W_1]^{2,6} = \begin{bmatrix} \lhd & a & a & a & b & b \\ \lhd & a & a & a & b & b \end{bmatrix}, [W'_1]^{2,4} = \begin{bmatrix} \lhd & a & a & b \\ \lhd & a & a & b \end{bmatrix}$$

$$[W_2]^{2,6} = \begin{bmatrix} a & a & a & a & b & b \\ b & b & b & b & a & a \end{bmatrix}, [W'_2]^{2,4} = \begin{bmatrix} \lhd & a & a & b \\ \lhd & b & b & a \end{bmatrix}$$

$$[W_3]^{2,6} = \begin{bmatrix} b & b & b & b & a & a \\ b & b & b & b & a & a \end{bmatrix}, [W'_3]^{2,4} = \begin{bmatrix} \lhd & b & b & a \\ \lhd & b & b & a \end{bmatrix}$$

In the next cycle, it rejects because this time transition function is undefined. $Reject \in \delta(q_0, ([W_1]^{2,6}, [W_2]^{2,6}, [W_3]^{2,6}))$ where

$$[W]^{4,6} = \begin{bmatrix} \lhd & a & a & b & \rhd \\ \lhd & a & a & b & \rhd \\ \lhd & b & b & a & \rhd \\ \lhd & b & b & a & \rhd \end{bmatrix}$$

4.2 Properties of 2D-PRA-W$_m$

In this section, we introduce some important properties of 2D-PRA-W$_m$.

Lemma 2. *The language class $\mathcal{L}$(2D-PRA-W$_m$) is closed under* 180° *rotation.*

Proof. Let 2D-PRA-W$_m$ be $M = (Q, V, \triangleleft, \triangleright, q_0, \delta)$, that accepts a language $L \subseteq V^{*,*}$ where $Q = \{q_0, q, Accept, Reject\}$, V is the input alphabet, $\triangleleft, \triangleright$ are left and right borders respectively and $\triangleleft, \triangleright \notin V$, q_0 is the initial state, $Accept \in \delta(q, ([W_i]^{2,k}, [W_{i+1}]^{2,k}, ..., [W_f]^{2,k}))$, $Reject \in \delta(q, ([W_i]^{2,k}, [W_{i+1}]^{2,k}, ..., [W_f]^{2,k})) = \emptyset$ (i.e., when δ is undefined), then M will reject.

Now, from M we can construct M_R (after 180° rotation of M) where $M_R = (Q, V, \triangleleft, \triangleright, q_0, \delta)$, that accepts a language $L \subseteq V^{*,*}$ where $Q = \{q_0, q, Accept', Reject\}$, V is the input alphabet, $\triangleleft, \triangleright$ are left and right borders respectively and $\triangleleft, \triangleright \notin V$, q_0 is the initial state, $Accept' \in \delta(q, ([W_i]_R^{2,k}, [W_{i+1}]_R^{2,k}, ..., [W_f]_R^{2,k}))$ where $[W_i]_R^{2,k}$ is the ith sub-window after 180° rotation of $[W_i]^{2,k}$ and $1 \leq i \leq f$, $Reject \in \delta(q, ([W_i]_R^{2,k}, [W_{i+1}]_R^{2,k}, ..., [W_f]_R^{2,k})) = \emptyset$ (i.e., when δ is undefined), then M will reject the input.

Lemma 3. *The language class $\mathcal{L}$*(2D-PRA-W$_m$) *is closed under complement.*

Proof. Let M be a 2D-PRA-W$_m$, that accepts a language $L \subseteq V^{*,*}$. Now, from M we can construct M_c (complement of M) by interchanging undefined and accepting transitions.

Lemma 4. *The language class $\mathcal{L}$*(2D-PRA-W$_m$) *is closed under column concatenation.*

Proof. Let M_1 be a 2D-PRA-W$_m$, on V, that accepts a language $L_1 \subseteq V^{*,*}$ where $Accept \in \delta(q, ([W_i]^{2,k}, [W_{i+1}]^{2,k}, ..., [W_f]^{2,k}))$. Consider M_2 be another 2D-PRA-W$_m$ which accepts a language $L_2 \subseteq V^{*,*}$ where $Accept \in \delta(q, ([W_i'']^{2,k}, [W_{i+1}'']^{2,k}, ..., [W_f'']^{2,k}))$. Now, we can construct M which can accept $L_1 \Phi L_2$ by modifying the accepting state, that is, $Accept \in \delta(q, ([W_i]^{2,k} \Phi [W_i'']^{2,k}, [W_{i+1}]^{2,k} \Phi [W_{i+1}'']^{2,k}, ..., [W_f]^{2,k} \Phi [W_f'']^{2,k}))$.

Lemma 5. *The language class $\mathcal{L}$*(2D-PRA-W$_m$) *with auxiliary special symbol is closed under intersection.*

Proof. Let 2D-PRA-W$_m$ be a $M_1 = (Q_1, V, \Gamma_1, \triangleleft, \triangleright, q_0', \delta_1)$ with auxiliary special symbols. Let $M_2 = (Q_2, V, \Gamma_2, \triangleleft, \triangleright, q_0'', \delta_2)$ be another 2D-PRA-W$_m$ with special symbols where $\Gamma_1, \Gamma_2 \supseteq V$. Now we can construct $M = (Q, V, \Gamma, \triangleleft, \triangleright, q_0, \delta)$ such that $L(M) = L_(M_1) \bigcap L_(M_2)$. Essentially, M will work as follows:

- M first simulates M_1, that is, it behaves exactly like M_1. If M_1 should get stuck on the given input, that is, M_1 does not accept, then neither does M. If, however, M_1 accepts, then instead of accepting, M marks the position (i, j) at which M_1 accepts, using a special symbol.

– Now only M should start simulating M_2. So, it is understood that we need to mark the last position by special symbol and because of that we introduced $\Gamma = V \cup T$ is the tape alphabet, $\Gamma \supseteq V$.

Lemma 6. *The language class* $\mathcal{L}$(2D-PRA-W$_{\mathrm{m}}$) *is not closed under transposition.*

Proof. Let $G = (V, A, C, \varphi)$ be a parallel internal column contextual array grammar where $V = \{a, b\}$,

$$A = \left\{ B = \begin{bmatrix} a & a & a & a & b \\ a & b & a & b & b \\ a & b & a & b & b \end{bmatrix} \right\}, C = \left\{ \begin{bmatrix} a \\ b \end{bmatrix} \psi \begin{bmatrix} a \\ b \end{bmatrix}, \begin{bmatrix} b \\ b \end{bmatrix} \psi \begin{bmatrix} b \\ b \end{bmatrix} \right\},$$

where $B \in A$, φ is a choice mapping,

$$\varphi \begin{bmatrix} a & a & a \\ b & a & b \end{bmatrix} = \begin{bmatrix} a \\ b \end{bmatrix} \psi \begin{bmatrix} a \\ b \end{bmatrix}, \varphi \begin{bmatrix} b & a & b \\ b & a & b \end{bmatrix} = \begin{bmatrix} b \\ b \end{bmatrix} \psi \begin{bmatrix} b \\ b \end{bmatrix}$$

We can construct 2D-PRA-W$_{\mathrm{m}}$ such that $L(M) = L(G)$ where the configuration of acceptance , ACCEPT- $[W]^{3,7} = [\triangleleft]^{3,1} B [\triangleright]^{3,1}$. where $B \in A$. Now, if we consider the transposition of B, we obtain

$$B_T = \begin{bmatrix} a & a & a \\ a & b & b \\ a & a & a \\ a & b & b \\ b & b & b \end{bmatrix}.$$

Clearly, $B_T \notin L(M)$ because in this case we cannot construct M_T. If $B_T \in L(M)$ then the content of each sub-window w_j in each cycle c_i should be transposed to w_{j_T} such that $\forall i \forall j \; Transpose(w_j, w_{j_T})$. In order to do that, the working procedure of our M for given G, needs to be changed.

4.3 Complexity of Membership Problem for PICCAL

In this section, we discuss the complexity of solving the membership problem for *PICCAL*. Let us start with *internal contextual string languages with finite choice* ($ICSL(FIN)$). $ICSL(FIN)$ is contained in the family of languages generated by *growing context-sensitive grammars* ($GCSG$), and from this scenario we will comment on the time complexity of solving membership problem for *PICCAL*. So here, first we recall the definition of $ICSL(FIN)$ and $GCSG$.

Definition 5 [17]**.** *For an alphabet* Σ*, we denote by* Σ^* *the free monoid generated by* Σ*, by* λ *its identity, and* $\Sigma^+ = \Sigma^* - \{\lambda\}$*. The family of finite languages is denoted by* FIN*. Contextual grammar is a construct,* $G = (\Sigma, A, (sel_1, C_1), (sel_2, C_2), ..., (sel_k, C_k))$*, for some* $k \geq 1$*, where* Σ *is an alphabet,* $A \subset \Sigma^*$ *is a finite set, called the axiom set,* $sel_i \subseteq \Sigma^*, 1 \leq i \leq k$*, are the sets of selectors, and* $C_i \subset \Sigma^* \times \Sigma^*$ *where* $1 \leq i \leq k$*, and* C_i *is a finite set of contexts. There are two basic modes of derivation, the internal mode of derivation as follows. For two words* $x, y \in \Sigma^*$*, we have the internal mode of derivation:*

$x \Longrightarrow_{in} y$ *iff* $x = x_1x_2x_3, y = x_1ux_2vx_3, x_2 \in sel_i, (u, v) \in C_i$, *for some* $1 \leq i \leq k$.
The language generated by internal mode of derivation is: $L_{in}(G) = \{w \in \Sigma^* \mid x \in A, x \Longrightarrow_{in}^* w\}$, *where* $\Longrightarrow_{in}^*$ *denotes the reflexive - transitive closure of* $\Longrightarrow_{in}$.

If the sets $sel_1, sel_2, ..., sel_k$ *are languages in a given family* FIN, *then* G *is said to be with* FIN *choice. The family of languages generated by contextual grammars with* FIN *choice in the internal mode of derivation is denoted by* $ICSL(FIN)$.

Now we recall the definition from [11].

Definition 6. *A* context-sensitive grammar (CSG) *is a tuple* $G = (V, T, P, S)$, *where* V *is a set of alphabets,* T *is a finite set of terminal symbols,* P *is a finite set of production rules, and* S *is the starting symbol. We say that* G *is* growing *if* S *does not appear on the right and* $|\alpha| < |\beta|$ *for any* $(\alpha \to \beta)$, *with* $\alpha \neq S$, *from* P.

Definition 7. *A CSG* $G = (V, T, P, S)$ *is QGCSG if there exists a function* $f : (V \cup T)^* \mapsto \mathbb{Z}^+$ *such that, for all* $p \in P$, $f(\alpha) > f(\beta)$.

Lemma 7. $ICSL(FIN) \subset GCSG$

Proof. $G = (\Sigma, A, (sel_1, C_1), (sel_2, C_2), ..., (sel_k, C_k))$ be $ICSL(FIN)$. We can assume that $(\lambda, \lambda) \notin C_i$ for all $1 \leq i \leq k$. The problem in developing a $QGCSG$, is to simulate an insertion step $x \Rightarrow_{in} y$ if $x = x_1x_3, \lambda \in S_i, y = x_1uvx_3$, and $(u, v) \in C_i$ for some $1 \leq i \leq k$. In order to avoid this, we do as follows:
We define homomorphism $h : \Sigma^* \to \Sigma'^*$ where $\Sigma' = \{a' \mid a \in \Sigma\}$ such that $\Sigma \cap \Sigma' = \phi$ and $h(a) = a'$ for $a \in \Sigma$. Now we are ready to construct $QGCSG$ $G' = (\Sigma' \cup S, \Sigma, P, S)$ where $S \notin \Sigma'$, the P is given below.
$P = \{S \to h(x) \mid x \in A\} \cup \{S \to h(u, v) \mid \lambda \in A \cap S_i, (u, v) \in C_i, 1 \leq i \leq k\} \cup \{h(x) \to h(uxv) \mid x \in S_i \setminus \{\lambda\}, (u, v) \in C_i, 1 \leq i \leq k\} \cup \{h(a) \to h(uva), h(a) \to h(auv) \mid a \in \Sigma, \lambda \in S_i, (u, v) \in C_i, 1 \leq i \leq k\} \cup \{h(a) \to a \mid a \in \Sigma\}$ with the valuation $f(S) = 1$ and $f(h(a)) = 2, f(a) = 3$, if $a \in \Sigma$. So, here the constructed grammar is $QGCSG$ and $L(G') = L_{in}(G)$.

Since $QGCSG = GCSG$, we can state that $ICSL(FIN) \subset GCSG$. Here the inclusion is strict because the cross-dependency language $L_{cross-dependency} = \{a^n b^m c^n d^m \mid n, m \geq 1\} \notin ICSL(FIN)$ but $L_{cross-dependency} \in GCSL$.

Lemma 8 [11]**.** *The membership problem for internal contextual string languages with finite choice* $(ICSL(FIN))$ *is* $LOG(CFL) - hard$.

Proof. From Lemma 7, we concluded that $ICSL(FIN) \subset GCSG$. In [5], it is shown that $GCSL$ family of languages, is contained in $LOG(CFL)$. This shows that the upper bound for membership problem for $ICSL(FIN)$ is $LOG(CFL)$.

Lemma 9. *The membership problem for parallel internal column contextual array languages (PICCAL) is contained in* NP.

Proof. Let $VERIFIER(W, C)$ be a procedure where W, C are given inputs and denote a word and certificate respectively. Here C is a certificate, i.e., a derivation. The procedure $VERIFIER(W, C)$ returns "YES" if the given certificate C is correct, otherwise "NO". In other words, $VERIFIER(W, C)$ verifies the correctness of C. Moreover the running time of $VERIFIER(W, C)$ is bounded by a polynomial in $|W|$ where $|W|$ denotes the size of W. See Algorithm 1.

Algorithm 1. Polynomial Time Verifier

```
1: procedure VERIFIER(W, C)
2:     Initialize w_i ← Axiom                    ▷ w_i stores the Axiom
3:     Initialize k ← |W|                        ▷ k stores the length of W
4:     Initialize N ← |C|                        ▷ Nstores the length of C
5:     for i = 1 to N do
6:         w_i ⇒_ithstep w_{i+1}
7:         if w_{i+1} == W then                  ▷ ith step of the derivation
8:             print YES return                  ▷ C is correct
9:         else
10:            i ← i + 1
11:        end if
12:    end for
13:    if |w_i| > k then
14:        Print NO                              ▷ C is incorrect
15:    end if
16: end procedure
```

Corollary 2. *The membership problem for PICCAL is at least $LOG(CFL)$ − hard and is contained in NP.*

Proof. From Lemma 8 and Lemma 9, we can easily conclude this Corollary 2.

5 Conclusion and Future Work

In this paper, we have introduced a non-deterministic 2D-PRA-W_m to solve the membership problem of *PICCAL*. Here we have introduced multiple windows in order to capture the parallel application of the parallel column contextual array rules. Also we discussed some of the important properties of 2D-PRA-W_m and commented on the complexity of membership problem for *PICCAL*.

Here our focus was on column concatenation only. We can extend our work to take care of row concatenation too. In terms of future direction of work, it could be also interesting, if we can define a powerful subclass of *PICCAG* and solve the membership problem using deterministic 2D-PRA-W_m.

Acknowledgement. Supported by NWO TOP project 612.001.852 Grey-box learning of Interfaces for Refactoring Legacy Software (GIRLS).

References

1. Alhazov, A., Fernau, H., Freund, R., Ivanov, S., Siromoney, R., Subramanian, K.: Contextual array grammars with matrix control, regular control languages, and tissue P systems control. Theor. Comput. Sci. **682**, 5–21 (2017)
2. Anselmo, M., Giammarresi, D., Madonia, M.: A common framework to recognize two-dimensional languages. Fundamenta Informaticae **171**(1–4), 1–17 (2020)
3. Bunke, H., Sanfeliu, A.: Syntactic and Structural Pattern Recognition: Theory and Applications, vol. 7. World Scientific, New York (1990)
4. Chandra, P.H., Subramanian, K., Thomas, D.: Parallel contextual array grammars and languages. Electron. Notes Discrete Math. **12**, 106–117 (2003)
5. Dahlhaus, E., Warmuth, M.K.: Membership for growing context-sensitive grammars is polynomial. J. Comput. Syst. Sci. **33**(3), 456–472 (1986)
6. Ehrenfeucht, A., Păun, G., Rozenberg, G.: Contextual grammars and formal languages. In: Rozenberg, G., Salomaa, A. (eds.) Handbook of Formal Languages, pp. 237–293. Springer, Heidelberg (1997). https://doi.org/10.1007/978-3-662-07675-0_6
7. Fernau, H., Freund, R., Siromoney, R., Subramanian, K.: Non-isometric contextual array grammars and the role of regular control and local selectors. Fundamenta Informaticae **155**(1–2), 209–232 (2017)
8. Fernau, H., Paramasivan, M., Schmid, M.L., et al.: Simple picture processing based on finite automata and regular grammars. J. Comput. Syst. Sci. **95**, 232–258 (2018)
9. Firschein, O.: Syntactic pattern recognition and applications. Proc. IEEE **71**(10), 1231–1231 (1983)
10. Giammarresi, D., Restivo, A.: Two-dimensional languages. In: Rozenberg, G., Salomaa, A. (eds.) Handbook of Formal Languages, pp. 215–267. Springer, Heidelberg (1997). https://doi.org/10.1007/978-3-642-59126-6_4
11. Holzer, M.: On fixed and general membership for external and internal contextual languages. In: Developments in Language Theory: Foundations, Applications, and Perspectives, pp. 351–361. World Scientific, New York (2000)
12. Janar, P., Mráz, F., Páltek, M., Procházka, M., Vogel, J.: Deleting automata with a restart operation. In: 3rd International Conference Developments in Language Theory, DLT, pp. 191–202 (1997)
13. Jancar, P., Mraz, F., Plátek, M., Procházka, M., Vogel, J.: Restarting automata, marcus grammars and context-free languages. In: Developments in Language Theory, pp. 102–111. World Scientific, New York (1995)
14. Jančar, P., Mráz, F., Plátek, M., Vogel, J.: Restarting automata. In: Reichel, H. (ed.) International Symposium on Fundamentals of Computation Theory, pp. 283–292. Springer, Heidelberg (1995). https://doi.org/10.1007/3-540-60249-6_60
15. Krithivasan, K., Balan, M.S., Rama, R.: Array contextual grammars. In: Recent Topics in Mathematical and Computational Linguistics, pp. 154–168 (2000)
16. Marcus, S.: Contextual grammars. In: International Conference on Computational Linguistics Coling 1969: Preprint No. 48 (1969)
17. Midya, A., Thomas, D., Malik, S., Pani, A.K.: Polynomial time learner for inferring subclasses of internal contextual grammars with local maximum selectors. In: Hung, D., Kapur, D. (eds.) International Colloquium on Theoretical Aspects of Computing, pp. 174–191. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-67729-3_11

18. Mráz, F., Průša, D., Wehar, M.: Two-dimensional pattern matching against basic picture languages. In: Hospodar, M., Jiraskova, G. (eds.) International Conference on Implementation and Application of Automata, pp. 209–221. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-23679-3_17
19. Otto, F.: Restarting automata for picture languages: a survey on recent developments. In: Holzer, M., Kutrib, M. (eds.) International Conference on Implementation and Application of Automata, pp. 16–41. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-08846-4_2
20. Otto, F., Mráz, F.: Extended two-way ordered restarting automata for picture languages. In: Dediu, A.H., Martin-Vide, C., Sierra-Rodriguez, J.L., Truthe, B. (eds.) International Conference on Language and Automata Theory and Applications. pp. 541–552. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-04921-2_44
21. Paun, G.: Marcus Contextual Grammars, vol. 67. Springer, Dordrecht (2013). https://doi.org/10.1007/978-94-015-8969-7
22. Paun, G., Nguyen, X.M.: On the inner contextual grammars. Rev. Roum. Math. Pures Appl. **25**(4), 641–651 (1980)
23. Rama, R., Smitha, T.: Some results on array contextual grammars. Int. J. Pattern Recogn. Artif. Intell. **14**(04), 537–550 (2000)
24. Rosenfeld, A.: Picture Languages: Formal Models for Picture Recognition. Academic Press, Cambridge (2014)
25. Rosenfeld, A., Siromoney, R.: Picture languages: a survey. Lang. Design **1**(3), 229–245 (1993)
26. Subramanian, K., Van, D.L., Chandra, P.H., Quyen, N.D.: Array grammars with contextual operations. Fundamenta Informaticae **83**(4), 411–428 (2008)