The Shuffle on Trajectories of Infinite Arrays

Devi Velayutham
Department of Mathematics, Hindustan College of Arts and Science, India

Abstract: *In this paper authors study and investigate the shuffle on trajectories on infinite array languages. Like finite array languages this approach is applicable to concurrency providing a method to define parallel composition of processes. It is also applicable to parallel computation. The operations are introduced using a uniform method based on the notion of ωω-trajectory. Authors introduce an Array Grammar with Shuffle on Trajectories (AGST) and compare it with other array grammars for generative poauthorsr. Authors prove closure properties for different classes of array languages with respect to the shuffle on trajectories.*

Keywords: *Büchi two-dimensional online tessellation automaton, ωω-trajectory, ωω-recognizable array language, column shuffle ωω-recognizable array language.*

Received April 16, 2015; accepted May 22, 2016

1. Introduction

Infinite arrays are digitized images of symbols occupying a quadrant of a plane. They can be thought of as extensions of infinite words to two dimensions. The motivation for considering infinite arrays lies in the fact that pictures of functions can be considered as infinite digitized images. An early work in this direction is done by Nakamura and Ono in [13]. Also authors study ω-languages and automata in [15, 19] and authors refer [1, 10, 14, 17, 18] for initiating this work. Infinite arrays is an array with infinite number of columns and rows.

Shuffle on trajectories is an interesting tool for the generation of languages of finite words as authorsll as infinite words. It is found in [4, 5, 6, 12]. Parallel composition of words and languages appears as a fundamental operation in parallel computation and in the theory of concurrency. This operation is modeled by shuffle operation or restrictions of this operation such as literal shuffle and inserton. A trajectory is a segment of a line in a plane starting in the origin of axes. Continuing parallel with the axes OX and OY. The line can change its direction only in points of non negative integers. In [16] trajectory plays an important role. The shuffle of two words has a natural geometrical interpretation related to lattice points in the plane [11]. On the other hand theoretical models for generating two dimensional arrays authorsre proposed in [9]. Shuffle operation on finite arrays with trajectories has been introduced in [8]. To develop the study on parallel contextual array grammars [2]. Based on the study of [3] authors consider the shuffle on trajectories as a tool for obtaining various infinite array languages.

The paper is organized as follows. In section 2, authors review necessary notions related to infinite arrays, trajectories and shuffle on trajectories. In section 3 authors give the notion of shuffle operation to infinite array languages and proved closure and associative properties for ωω-recognizable languages. In section 4 authors study the literal shuffle for infinite array languages and obtain an interesting result.

2. Preliminaries

In this section authors recall some necessary notions and definitions for infinite arrays languages. For finite or infinite array considerd in this paper the bottom most row is the first row and the left most column is the first column.

- *Definition* 1. An infinite word is a mapping from $\mathbb{N}$ to Σ. The set of all finite (respectively infinite) words on Σ is denoted by Σ^* (respectively Σ^ω). A ω-language L is a subset of Σ^ω.
- *Definition* 2. Let $\alpha=a_0\, a_1\, a_2\, ...$, where $a_i \in \Sigma$, for all $i \geq 0$ be an ω-word over Σ. A run of $\mathcal{A}$ on α is a sequence of states $s = s_0\, s_1\, s_2\, ...$, such that $s_0 = q_0$ and $(s_i, a_i, s_{i+1}) \in \delta$, for all $i \geq 0$. The run is successful if and only if $\inf(s) \cap F \neq \phi$ (where $\inf(s)$ is the set of all states which repeat infinitely many times in s). α is accepted by $\mathcal{A}$ if and only if there exists a successful run of $\mathcal{A}$ on α.

The ω-language recognized by $\mathcal{A}$ is $L(\mathcal{A})=\{\alpha \in \Sigma^\omega \mid \alpha$ is accepted by $\mathcal{A}\}$. An ω-language L is referred to as ω-regular or Büchi recognizable if and only if there exists a Büchi automaton $\mathcal{A}$ such that $L(\mathcal{A})=L$.

- *Definition* 3. An infinite array has an infinite number of rows and an infinite number of columns. The collection of all infinite arrays over Σ is denoted by $\Sigma^{\omega\omega}$. If an array $p \in \Sigma^{\omega\omega}$ has entry a_{ij} in the ith row and jth column, $a_{ij} \in \Sigma$ then authors write $p=(a_{ij})$, $i=1, 2, ..., j=1, 2, ...$ For $p \in \Sigma^{\omega\omega}$, $\hat{p}$ is the infinite array obtained by placing a row of boundary symbol # below the first row of p and to the left of the first column of p.

- *Definition* 4. A Büchi automaton is a quintuple $\mathcal{A} = (\Sigma, Q, q_0, \delta, F)$, where Σ is the input alphabet; Q is a finite set of states; $q_0 \in Q$ is the initial state; δ is the transition relation, $\delta \subseteq Q \times \Sigma \times Q$, and $F \subseteq Q$ is the set of final states.
- *Definition* 5. A non-deterministic Büchi Two-Dimensional Online Tessellation Automaton (2DBOTA) is $\mathcal{A} = (\Sigma, Q, Q_0, F, \delta)$ where Σ is an input alphabet, Q is the finite set of states, $Q_0 \subseteq Q$ is a set of initial states, $F \subseteq Q$ is a set of final states and $\delta : Q \times Q \times \Sigma \rightarrow 2^Q$ is a transition function.

A computation by a 2DBOTA on an infinite array p with

$$\hat{p} = \begin{matrix} \cdot & \cdot & \cdot & \cdot & \cdot \\ \# & \cdot & \cdot & \cdot & \cdot \\ \# & a_{21} & a_{22} & \cdot & \cdot \\ \# & a_{11} & a_{12} & \cdot & \cdot \\ \# & \# & \# & \# & \cdot \end{matrix}$$

Where $a_{ij} \in \Sigma$ and $\# \notin \Sigma$ is done as follows:

At time t=0, an initial state $q_0 \in Q_0$ is associated with all the positions of $\hat{p}$ holding #. At time t=1, a state from $\delta(q_0, q_0, a_{11})$ is associated with the position (1, 1) holding a_{11}. At time t=2, states are associated simultaneously with the positions (1, 2) and (2, 1) respectively holding a_{12} and a_{21}. If s_{11} is the state associated with the position (1, 1), then the state associated with the position (2, 1) is an element of $\delta(s_{11}, q_0, a_{21})$ and to the position (1, 2) is an element of $\delta(q_0, s_{11}, a_{12})$. Authors then proceed to the next diagonal. The states associated with each position (i, j) by the transition function δ depends on the states already associated with the positions (i–1, j), (i, j–1) and the symbol a_{ij}. Let s_{ij} be the state associated with the position (i, j) where the entry is a_{ij}. A run (or a computation) of an infinite array is an element of Q^ω. A run for an infinite array is a sequence of states $s_{11}\ s_{12}\ s_{21}\ s_{31}\ s_{22}\ s_{13}$ ... and it is denoted by r(p).

If $\mathcal{A}$ is non-deterministic, a run for an infinite array is a set of sequences of states containing only one sequence of state, but in the deterministic case it is a singleton set. If $p \in \Sigma^{\omega\omega}$, the set of runs of p is denoted by Run(p).

The language of infinite arrays recognized by the non-deterministic Büchi online tessellation automaton $\mathcal{A}$ is $L^{\omega\omega}(\mathcal{A}) = \{p \in \Sigma^{\omega\omega} : \inf(r(p)) \cap F \neq \phi$, for some $r(p) \in \text{Run}(p)\}$.

A language of infinite arrays is called a $\omega\omega$-array language. A $\omega\omega$-array language K is called recognizable if there exists a Büchi online tessellation automaton A such that $L^{\omega\omega}(A) = K$.

2.1. Shuffling on Trajectories over Infinite Arrays

In this section authors prove that the shuffling of two $\omega\omega$-recognizable languages is again $\omega\omega$-recognizable array language.

- *Definition* 6. The column shuffle operation on an infinite array denoted by $\text{ш}^{c\omega}$ is defined recursively by

$$\begin{aligned} P\ \text{ш}^{c\omega}\ Q &= ((A \obar X)\ \text{ш}^{c\omega}\ (B \obar Y)) \\ &= A \obar (X\ \text{ш}^{c\omega}\ (B \obar Y)) \cup \\ &\qquad B \obar ((A \obar X)\ \text{ш}^{c\omega}\ Y) \end{aligned}$$

Where $P = A \obar X$ and $Q = B \obar Y$, where $P, Q \in \Sigma^{\omega\omega}$, A is the first column of array P and B is the first column of array Q. If A is empty then X=P. Likewise if B is empty then Y=Q. Also $P\ \text{ш}^{c\omega}\ \Lambda = \Lambda\ \text{ш}^{c\omega}\ P = P$.

- *Definition* **7**. The row shuffle operation on an infinite array denoted by $\text{ш}^{r\omega}$ is defined recursively by

$$\begin{aligned} P\ \text{ш}^{r\omega}\ Q &= ((A \ominus X)\ \text{ш}^{r\omega}\ (B \ominus Y)) \\ &= A \ominus (X\ \text{ш}^{r\omega}\ (B \ominus Y)) \cup \\ &\qquad B \ominus ((A \ominus X)\ \text{ш}^{r\omega}\ Y) \end{aligned}$$

Where $P = A \ominus X$ and $Q = B \ominus Y$, $P, Q \in \Sigma^{\omega\omega}$, A is the first row of array P and B is the first row of array Q. Also $P\ \text{ш}^{r\omega}\ \Lambda = \Lambda\ \text{ш}^{r\omega}\ P = P$.

Let $V_1 = \{r, u\}$ and $V_2 = \{\ell, d\}$ be the set of versors in the plane. ℓ, r, u and d stand for the left, right, up and down directions respectively. A trajectory is an element $t \in V_1^\omega \cup V_2^\omega$.

- *Definition* 8. Let Σ be a finite alphabet, $v \in \{r, u\}$, $t \in \{r, u\}^\omega$ and $P, Q \in \Sigma^{\omega\omega}$. The column shuffle of P with Q on the trajectory vt, denoted by $P\ \text{ш}_{vt}^{c\omega}\ Q$ is recursively defined as follows:

If $P = A \obar X$ and $Q = B \obar Y$ where $A, B, X, Y \in \Sigma^{\omega\omega}$, A and B are the first columns of P and Q respectively, then

$$\begin{aligned} P\ \text{ш}_{vt}^{c\omega}\ Q &= (A \obar X)\ \text{ш}_{vt}^{c\omega}\ (B \obar Y) \\ &= \begin{cases} A \obar (X\ \text{ш}_{vt}^{c\omega}\ (B \obar Y)) & \text{if } v = r, \\ B \obar ((A \obar X)\ \text{ш}_{vt}^{c\omega}\ Y) & \text{if } v = u. \end{cases} \end{aligned}$$

$$\text{If } P = \Lambda,\ \Lambda\ \text{ш}_{vt}^{c\omega}\ (B \obar Y) = \begin{cases} \phi & \text{if } v = r \\ B \obar (\Lambda \text{ш}_{vt}^{c\omega}\ Y) & \text{if } v = u \end{cases}$$

$$\text{If } Q = \Lambda,\ (A \obar X)\ \text{ш}_{vt}^{c\omega}\ \Lambda = \begin{cases} A \obar (X\ \text{ш}_{vt}^{c\omega}\ \Lambda) & \text{if } v = r \\ \phi & \text{if } v = u \end{cases}$$

$$\text{and } \Lambda\ \text{ш}_{vt}^{c\omega}\ \Lambda = \begin{cases} \Lambda & \text{if } t = \Lambda \\ \phi & \text{otherwise} \end{cases}$$

The row shuffle of P with Q on the trajectory vt, $v \in \{\ell, d\}$ and $t \in \{\ell, d\}^\omega$ is defined in a similar way with r, u replaced by ℓ, d and the catenation $\obar$ is replaced by $\ominus$ catenation. Also if $|P|_c \neq |t|$ or $|P|_c \neq |t|_u$ then $P\ \text{ш}_{vt}^{c\omega}\ Q = \phi$. Similarly if $|P| \neq |t|$ or $|Q| \neq |t|_d$ then

P ш$_{vt}^{r\omega}$ Q = ϕ. If T is a set of trajectories, then P ш$_{vt}^{c\omega}$ Q = $\bigcup_{t\in T}$ P ш$_{vt}^{c\omega}$ Q and P ш$_{vt}^{r\omega}$ Q = $\bigcup_{t\in T}$ P ш$_{vt}^{r\omega}$ Q.

- *Example* 1. Let

$$P=\begin{array}{cccccccc}\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot\\\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot\\b&b&\cdot&\cdot&b&b&b&\cdot\\b&b&\cdot&\cdot&b&b&b&\cdot\\a&a&\cdot&\cdot&a&b&b&\cdot\\\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot\\\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot\\a&a&\cdot&\cdot&a&b&b&\cdot\\a&a&\cdot&\cdot&a&b&b&\cdot\end{array}$$

$$Q=\begin{array}{cccccccc}\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot\\\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot\\a&a&\cdot&\cdot&a&b&b&\cdot\\a&a&\cdot&\cdot&b&b&b&\cdot\\a&a&\cdot&\cdot&a&b&b&\cdot\\\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot\\\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot\\a&a&\cdot&\cdot&a&b&b&\cdot\\a&a&\cdot&\cdot&a&b&b&\cdot\end{array}\quad\text{and}\quad R=\begin{array}{cccccccc}\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot\\\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot\\b&b&\cdot&\cdot&b&b&b&\cdot\\b&b&\cdot&\cdot&b&b&b&\cdot\\a&a&\cdot&\cdot&a&a&a&\cdot\\\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot\\\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot\\a&a&\cdot&\cdot&a&a&a&\cdot\\a&a&\cdot&\cdot&a&a&a&\cdot\end{array}$$

be three infinite arrays in Σ$^{\omega\omega}$. In P, the entries in the (i, j)th positions 1 ≤ i ≤ n, 1 ≤ j ≤ m are "a" whereas in other places the entries are "b". In Q the (i, j)th positions i ≥ 1, 1 ≤ j ≤ m are "a" whereas in other places the entries are "b". In R, the entries in the (i, j)th positions j ≥ 1, 1 ≤ i ≤ n are "a" whereas in other places the entries are "b". Now for t = (ru)$^{\omega}$,

$$P\ \text{ш}_{vt}^{c\omega}\ R=\begin{array}{cccccccccccc}\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot\\\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot\\b&b&b&b&\cdot&\cdot&b&b&b&b&\cdot&\cdot\\b&b&b&b&\cdot&\cdot&b&b&b&b&\cdot&\cdot\\a&a&a&a&\cdot&\cdot&a&a&a&a&\cdot&\cdot\\\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot\\\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot\\a&a&a&a&\cdot&\cdot&a&a&b&a&\cdot&\cdot\\a&a&a&a&\cdot&\cdot&a&a&b&a&\cdot&\cdot\end{array},$$

Similarly t = (ℓd)ω, P ш$_{vt}^{r\omega}$ Q =

$$\begin{array}{cccccccccc}\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot\\\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot\\a&a&\cdot&\cdot&\cdot&a&b&b&\cdot&\cdot\\b&b&\cdot&\cdot&\cdot&b&b&b&\cdot&\cdot\\a&a&\cdot&\cdot&\cdot&a&b&b&\cdot&\cdot\\b&b&\cdot&\cdot&\cdot&a&b&b&\cdot&\cdot\\a&a&\cdot&\cdot&\cdot&a&b&b&\cdot&\cdot\\a&a&\cdot&\cdot&\cdot&a&b&b&\cdot&\cdot\\\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot\\a&a&\cdot&\cdot&\cdot&a&b&b&\cdot&\cdot\\a&a&\cdot&\cdot&\cdot&a&b&b&\cdot&\cdot\\a&a&\cdot&\cdot&\cdot&a&b&b&\cdot&\cdot\\a&a&\cdot&\cdot&\cdot&a&b&b&\cdot&\cdot\end{array}.$$

Now, authors give a result concerning the shuffle of two ωω-recognizable array languages. If L_1, $L_2 \subseteq \Sigma^{\omega\omega}$, then L_1 ш$^{\omega}$ L_2 = {P ш$^{c\omega}$ Q, P ш$^{c\omega}$ Q | P ∈ L_1, Q ∈ L_2} and L_1 ш$_T^{\omega}$ L_2 = {P ш$_t^{c\omega}$ Q, P ш$_t^{r\omega}$ Q | P ∈ L_1, Q ∈ L_2, t ∈ T}

- *Theorem* 1. If L_1 and L_2 are ωω-recognizable array languages then L_1 ш$^{\omega}$ L_2 is a ωω-recognizable array language.

- *Proof*. Let L_1 and L_2 be two ωω-recognizable array languages over the same alphabet Σ. Let $\mathcal{A}_i$ = (Σ, Q_i, Q_0^i, F_i, δ_i) be a Büchi online tessellation automaton such that L($\mathcal{A}_i$) = L_i, i = 1, 2. Authors define a Büchi online tessellation automaton $\mathcal{A}$ = (Σ, Q, Q_0, δ, F) such that L($\mathcal{A}$) = L_1 ш$^{\omega}$ L_2 as follows: Q = $Q_1 \times Q_2 \times$ {0, 1, 2}. Elements in Q are denoted as (q_1^1, q_1^2, k) where $q_1^1 \in Q_1$, $q_1^2 \in Q_2$ and 0 ≤ k ≤ 2. The initial state is $q_0 = (q_0^1, q_0^2, 0)$ and the set of final states is F =$Q_1 \times Q_2 \times$ {2}. The transition function δ is defined in such a way that it simulates non-deterministically on the first component, the automaton $\mathcal{A}_1$ or on the second component, the automaton $\mathcal{A}_2$. The third component of a state is used to record an occurrence of a final state from F_1 if the value 1 is stored. The value 2 is stored if at some stage later a final state from F_2 does occur. The value 0 is stored in the third component whenever the first two components are not the final states.

Formally, δ is defined as follows:

$$\delta((q_1^1,q_1^2),(q_2^1,q_2^2),0,a)=\{(\delta_1((q_1^1,q_2^1),a),q_1^2,0),$$
$$(q_1^1,\delta_2((q_1^2,q_2^2),a),0)\}\quad\text{if}\quad \delta_1(q_1^1,q_2^1,a)\notin F_1$$
$$\delta((q_1^1,q_1^2),(q_2^1,q_2^2),0,a)=\{(\delta_1((q_1^1,q_2^1),a),q_1^2,1),$$
$$(q_1^1,\delta_2((q_1^2,q_2^2),a),0)\}\quad\text{if}\quad \delta_1(q_1^1,q_2^1,a)\in F_1$$
$$\delta((q_1^1,q_1^2),(q_2^1,q_2^2),1,a)=\{(\delta_1((q_1^1,q_2^1),a),q_1^2,1),$$
$$(q_1^1,\delta_2((q_1^2,q_2^2),a),1)\}\quad\text{if}\quad \delta_2(q_1^1,q_2^1,a)\notin F_2$$
$$\delta((q_1^1,q_1^2),(q_2^1,q_2^2),1,a)=\{(\delta_1((q_1^1,q_2^1),a),q_1^2,1),$$
$$(q_1^1,\delta_2((q_1^2,q_2^2),a),2)\}\quad\text{if}\quad \delta_2(q_1^1,q_2^1,a)\in F_2$$
$$\delta((q_1^1,q_1^2),(q_2^1,q_2^2),2,a)=\{(\delta_1((q_1^1,q_2^1),a),q_1^2,0),$$
$$(q_1^1,\delta_2((q_1^2,q_2^2),a),0)\}\quad\text{if}\quad \delta_1(q_1^1,q_2^1,a)\notin F_1$$
$$\delta((q_1^1,q_1^2),(q_2^1,q_2^2),2,a)=\{(\delta_1((q_1^1,q_2^1),a),q_1^2,1),$$
$$(q_1^1,\delta_2((q_1^2,q_2^2),a),0)\}\quad\text{if}\quad \delta_1(q_1^1,q_2^1,a)\in F_1$$

Clearly L($\mathcal{A}$) = L_1 ш$^{\omega}$ L_2.

- *Definition* 9. A column shuffle ωω-array language over Σ is the set $C=\left\{\begin{bmatrix}u_1\\u_2\\\vdots\\u_m\end{bmatrix}\middle|\, u_i\in\Sigma^{r\times k}, r,k,m\geq 1\right\}$ and row shuffle array language over Σ is the set R={[u_1 u_2 ... u_m] | $u_i \in \Sigma^{k\times c}$, c, k, m ≥ 1}.
- *Definition* 10. An array grammar with shuffle on trajectories (AGST) is a construct G = (Σ, B, C, R, T_C, T_R) where Σ is an alphabet, B is a finite subset of Σ$^{\omega\omega}$ called the base of G, C and R are called column and row shuffle array languages over Σ respectively. $T_C \subset V_1^{\omega}$ and $T_R \subset V_2^{\omega}$ are sets of trajectories over the column and row shuffle array of C and R respectively.

The direct derivation with respect to G is a binary relation ⇒ ш$_T^{\omega}$ where X, Y ∈ Σ$^{\omega\omega}$ if and only if Y = X ш$_{T_C}^{\omega}$ U with U∈C or Y=X ш$_{T_R}^{\omega}$ U with U ∈ R. ⇒ ш$_T^{*\omega}$ is the reflexive transitive closure of ⇒ ш$_T^{\omega}$.

The language generated by G, denoted as L(G) is defined as follows

L(G)={Y∈Σ$^{\omega\omega}$ | X∈B such that X ⇒ ш$_T^{*\omega}$ Y}.

The family of all languages generated by AGST is denoted by AGST. AGST$_{REG}$, AGST$_{CF}$, AGST$_{CS}$

denote the family of all languages generated by AGST grammars with regular, context free and context sensitive languages of trajectories respectively.

Authors now give some examples from the family AGST$_{REG}$.

- *Example* 2. Consider the infinite array in $\Sigma^{\omega\omega}$

$$P = \begin{matrix} \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ b & b & b & b & b & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ b & b & b & b & b & \cdot & \cdot \\ b & b & b & b & b & \cdot & \cdot \\ a & a & a & b & b & \cdot & \cdot \\ a & a & a & b & b & \cdot & \cdot \\ a & a & a & b & b & \cdot & \cdot \end{matrix}$$

In P, the entries in the (i, j)th positions $1 \le i \le 3$, $1 \le j \le 3$ are a whereas in other places the entries are b. P is generated by the following AGST of AGST$_{REG}$. G = (Σ, B, C, R, T$_C$, T$_R$) where $B = \begin{pmatrix} a & a & a \\ a & a & a \\ a & a & a \end{pmatrix}$,

R = $\{(b)^n / n \ge 3\}$, C = $\{(b)^n / n \ge 4\}$
T$_R$ = $\{\lambda^n d / n \ge 3\}$, T$_C$ = $\{r^4 u / n \ge 4\}$

A sample derivation is shown below:

$\begin{matrix} a & a & a \\ a & a & a \\ a & a & a \end{matrix}$	$ш_{T_R}$ $\Rightarrow \ell^3 d$	$\begin{matrix} b & b & b \\ a & a & a \\ a & a & a \\ a & a & a \end{matrix}$	$ш_{T_C}$ $\Rightarrow r^4 u$	$\begin{matrix} b & b & b & b \\ a & a & a & b \\ a & a & a & b \\ a & a & a & b \end{matrix}$

$ш_{T_R}$ $\Rightarrow \ell^4 d$	$\begin{matrix} b & b & b & b \\ b & b & b & b \\ a & a & a & b \\ a & a & a & b \\ a & a & a & b \end{matrix}$	$ш_{T_C}$ $\Rightarrow r^5 u$	$\begin{matrix} b & b & b & b & b \\ b & b & b & b & b \\ a & a & a & b & b \\ a & a & a & b & b \\ a & a & a & b & b \end{matrix}$

Repeatedly applying row shuffle and column shuffle operations authors get the infinite array language.

- *Theorem* 2. The AGST$_{REG}$ intersects (R:R)AG.

- *Proof.* The L-token of all sizes of a fixed proportion is generated by (R : R)AG. In fact G = (V, I, P, S) where V = V$_1$ $\cup$ V$_2$, V$_1$ = {S}, V$_2$ = {A, B}, I ={a, b} and P= P$_1$ $\cup$ P$_2$ generates L-tokens of all sizes, the ratio betauthorsen the two arrays of L is 1.

Here P$_1$ $\to$ {S $\to$ (S A) B},
R = $\{(b)^n / n \ge 3\}$, C = $\{(b)^n / n \ge 4\}$ and P$_2$=
$\left\{ S \to \begin{matrix} a & a & a \\ a & a & a \\ a & a & a \end{matrix} \right\}$.

A sample derivation is shown below.

$$S \Rightarrow \begin{matrix} a & a & a \\ a & a & a \\ a & a & a \end{matrix} \Rightarrow \begin{matrix} b & b & b & b \\ a & a & a & b \\ a & a & a & b \\ a & a & a & b \end{matrix} \Rightarrow \begin{matrix} b & b & b & b & b \\ b & b & b & b & b \\ a & a & a & b & b \\ a & a & a & b & b \\ a & a & a & b & b \end{matrix}$$

Repeatedly applying row and column shuffle operations authors get the infinite array language. This language is also generated by a AGST of AGST$_{REG}$.

Now authors have a result with respect to two $\omega\omega$-array languages L$_1$ and L$_2$ and a regular language T.

- *Theorem* 3. Let T $\subseteq$ $\{r, u\}^\omega \cup \{\ell, d\}^\omega$ be a set of trajectories. For all recognizable $\omega\omega$-array languages L$_1$ and L$_2$, the array language L$_1$ ш$_T$ L$_2$ is $\omega\omega$-recognizable.

- *Proof.* Let T be a regular language. L$_1$, L$_2$ are two recognizable $\omega\omega$-array languages over the same alphabet Σ. Let A$_i$ = (Σ, Q$_i$, δ_i, q_0^i, F$_i$) be a deterministic two direction Buchi online tessellation automaton such that L(A$_i$) = L$_i$ for i=1, 2. Also A$_T$ =({r, u, ℓ, d}, Q$_T$, δ_T, q_0^T, F$_T$) be a deterministic finite state automaton such that L(A$_T$) =T.

Authors define a Buchi online tessellation automaton A=(Σ, Q, δ, Q$_0$, F) such that L(A)=L$_1$ ш$_T^\omega$ L$_2$. Formally A on an input p $\in$ $\Sigma^{\omega\omega}$ simulates non deterministically A$_1$ or A$_2$ or from A$_2$ to A$_1$. Each change determines a transition in A$_T$ as follows: a change from A$_1$ to A$_2$ is interpreted as u or d and a change from A$_2$ to A$_1$ is interpreted as r or ℓ respectively.

The input p is accepted by A if and only if each of A$_1$, A$_2$ and A$_T$ accepts L$_1$, L$_2$ and T respectively.

Formally Q=Q$_1$×Q$_T$×Q$_2$×{0, 1, 2}. Elements of Q are denoted as (q_1^1, q_T, q_1^2, k) where $q_1^1 \in$ Q$_1$, $q_1^2 \in$ Q$_2$ and $0 \le k \le 2$, q$_0$ $\in$ $(q_0^1, q_T, q_0^2, 0)$ and the set of final states is F = (Q$_1$,×Q$_T$×Q$_2$×{2}). The transition function δ is defined as follows: $\delta : Q \times Q \times \Sigma \to 2^Q$. The transition function is defined in such a way that it simulates non-deterministically in the first component, the automaton A$_1$ or on the second component the automaton A$_2$. The third component of a state is record an occurrence of a final state from F$_1$ if the value 1 is stored. The value 2 is stored if at some stage later a final state from F$_2$ does occur. The value 0 is stored in the third component whenever the first two components are not the final states. Now for t = (ru)^w where t $\in$ T. δ is defined as follows.

$$\delta((q_1^1, q_T, q_1^2), (q_2^1, q_T, q_2^2), 0, a) = \{(\delta_1(q_1^1, q_2^1), a), \delta_T(q_T, r), q_1^2, 0),$$
$$(q_1^1, \delta_T(q_T, u), \delta_2((q_1^2, q_2^2), a), 0)\} \quad \text{if} \quad \delta_1(q_1^1, q_T, q_2^1, a) \notin F_1$$
$$\delta((q_1^1, q_T, q_1^2), (q_2^1, q_T, q_2^2), 0, a) = \{(\delta_1(q_1^1, q_2^1), a), \delta_T(q_T, r), q_1^2, 1),$$
$$(q_1^1, \delta_T(q_T, u), \delta_2((q_1^2, q_2^2), a), 0)\} \quad \text{if} \quad \delta_1(q_1^1, q_T, q_2^1, a) \in F_1$$
$$\delta((q_1^1, q_T, q_1^2), (q_2^1, q_T, q_2^2), 1, a) = \{(\delta_1(q_1^1, q_2^1), a), \delta_T(q_T, r), q_1^2, 1),$$
$$(q_1^1, \delta_T(q_T, u), \delta_2((q_1^2, q_2^2), a), 1)\} \quad \text{if} \quad \delta_2(q_1^2, q_T, q_2^2, a) \notin F_2$$
$$\delta((q_1^1, q_T, q_1^2), (q_2^1, q_T, q_2^2), 1, a) = \{(\delta_1(q_1^1, q_2^1), a), \delta_T(q_T, r), q_1^2, 1),$$
$$(q_1^1, \delta_T(q_T, u), \delta_2((q_1^2, q_2^2), a), 2)\} \quad \text{if} \quad \delta_2(q_1^2, q_T, q_2^2, a) \in F_2$$
$$\delta((q_1^1, q_T, q_1^2), (q_2^1, q_T, q_2^2), 2, a) = \{(\delta_1(q_1^1, q_2^1), a), \delta_T(q_T, r), q_1^2, 0),$$
$$(q_1^1, \delta_T(q_T, u), \delta_2((q_1^2, q_2^2), a), 0)\} \quad \text{if} \quad \delta_1(q_1^1, q_T, q_2^1, a) \notin F_1$$
$$\delta((q_1^1, q_T, q_1^2), (q_2^1, q_T, q_2^2), 2, a) = \{(\delta_1(q_1^1, q_2^1), a), \delta_T(q_T, r), q_1^2, 1),$$
$$(q_1^1, \delta_T(q_T, u), \delta_2((q_1^2, q_2^2), a), 0)\} \quad \text{if} \quad \delta_1(q_1^1, q_T, q_2^1, a) \in F_1$$

$\therefore$ L(A) = L$_1$ ш$_T^\omega$ L$_2$.

Similarly for t = $(\ell d)\omega$ authors can prove the above result.

$\therefore L_1 \,\text{ш}_T^{\omega}\, L_2$ is $\omega\omega$-recognizable.

- *Example* 3. Let

$$L_1 = \begin{array}{ccccccccc}
\# & . & . & . & . & . & . & . & . \\
\# & . & . & . & . & . & . & . & . \\
\# & b & b & . & . & b & b & b & . \\
\# & b & b & . & . & b & b & b & . \\
\# & a & a & . & . & a & b & b & . \\
. & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . \\
\# & a & a & . & . & a & b & b & . \\
\# & \# & \# & \# & \# & \# & \# & \# & \#
\end{array}$$

where $L_1=\{P_{n,m} / n, m \geq 1\}$. In $P_{n,m}$ the entries in the (i, j)th positions $1 \leq i \leq n$, $1 \leq j \leq m$ are a and the remaining entries are b. L_1 is recognizable [5] and $L(A_1)=L_1$ where $A_1=(\Sigma_1, Q_1, \delta_1, q_0^1, F_1)$ with $Q_1=\{q_0^1, q_1^1, q_2^1\}$ and $F_1=\{q_2^1\}$. The transition function $\delta_1 : Q_1 \times Q_2 \times \Sigma \rightarrow 2^Q$ is defined as follows:

$$\delta_1(q_0^1, q_0^1, a) = q_1^1 \tag{1}$$

$$\delta_1(q_0^1, q_1^1, a) = q_1^1 \tag{2}$$

$$\delta_1(q_1^1, q_0^1, a) = q_1^1 \tag{3}$$

$$\delta_1(q_1^1, q_1^1, a) = q_1^1 \tag{4}$$

$$\delta_1(q_0^1, q_1^1, b) = q_2^1 \tag{5}$$

$$\delta_1(q_1^1, q_0^1, b) = q_2^1 \tag{6}$$

$$\delta_1(q_1^1, q_2^1, b) = q_2^1 \tag{7}$$

$$\delta_1(q_2^1, q_1^1, b) = q_2^1 \tag{8}$$

$$\delta_1(q_2^1, q_2^1, b) = q_2^1 \tag{9}$$

$$\delta_1(q_0^1, q_2^1, b) = q_2^1 \tag{10}$$

$$\delta_1(q_2^1, q_0^1, b) = q_2^1. \tag{11}$$

Consider another infinite array language L_2.

$$L_2 = \begin{array}{cccccccccc}
\# & . & . & . & . & . & . & . & . & . \\
\# & b & b & . & . & . & b & b & b & . \\
\# & b & b & . & . & . & b & b & b & . \\
. & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . \\
\# & a & a & . & . & . & a & a & a & . \\
\# & a & a & . & . & . & a & a & a & . \\
\# & \# & \# & . & . & . & \# & \# & \# & .
\end{array}$$

where $L_2 = \{R_n / n \geq 1\}$

In R_n the entries in the (i, j)th positions $j \geq 1$, $1 \leq i \leq n$ are a, other entries are b. L_2 is recognizable in [5]. Since $L_2=L(A_2)$ where $A_2=(\Sigma_2, Q_2, \delta_2, q_0^2, F_2)$, $\Sigma_2=\{a, b\}$, $Q_2=\{q_0^2, q_1^2, q_2^2\}$ and $F_2=\{q_2^2\}$.

The transition function δ_2 is given as follows:

$$\delta_2(q_0^2, q_0^2, a) = q_1^2 \tag{12}$$

$$\delta_2(q_1^2, q_0^2, a) = q_1^2 \tag{13}$$

$$\delta_2(q_0^2, q_1^2, a) = q_1^2 \tag{14}$$

$$\delta_2(q_1^2, q_1^2, a) = q_1^2 \tag{15}$$

$$\delta_2(q_1^2, q_0^2, b) = q_2^2 \tag{16}$$

$$\delta_2(q_1^2, q_2^2, b) = q_2^2 \tag{17}$$

$$\delta_2(q_2^2, q_0^2, b) = q_2^2 \tag{18}$$

$$\delta_2(q_2^2, q_1^2, b) = q_2^2 \tag{19}$$

$$\delta_2(q_2^2, q_2^2, b) = q_2^2. \tag{20}$$

Let $T=T_C=\{(ru)^{\omega}\}$ be the trajectory. Clearly $L(A_T)=T$ where $A_T=(\Sigma_T, Q_T, \delta_T, q_0^T, F_T)$ is a finite deterministic automaton such that $\Sigma_T=\{r, u\}$, $Q_T=(q_0^T, q_r^T, q_u^T)$ and $F_T=\{q_u^T\}$. The transition function δ_T is defined as $\delta_T(q_0^T, r) = q_r^T$, $\delta_T(q_r^T, u) = q_u^T$, $\delta_T(q_u^T, r) = q_r^T$, $\delta_T(q_u^T, \not{b}) = q_u^T$, $\delta_T(q_r^T, \not{b}) = q_r^T$.

Now

$L(A)=L(A_1) \,\text{ш}_T^{\omega}\, L(A_2)=P \,\text{ш}_t^{C\omega}\, R$

$$= \begin{array}{cccccccccccc}
. & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . \\
b & b & b & b & . & . & b & b & b & b & . & . \\
b & b & b & b & . & . & b & b & b & b & . & . \\
a & a & a & a & . & . & a & a & b & a & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . \\
a & a & a & a & . & . & a & a & b & a & . & . \\
a & a & a & a & . & . & a & a & b & a & . & .
\end{array}$$

Clearly $L(A_1) \,\text{ш}_T^{\omega}\, L(A_2)=L(A)$ where $A=(\Sigma, Q, \delta, q_0, F)$ with $Q=Q^1 \times Q_T \times Q_2$, $Q_0=\{(q_0^1, q_0^T, q_0^2)\}$, $F=F_1 \times F_T \times F_2$ and $\delta : Q \times Q \times \Sigma \rightarrow 2^Q$.

The transition function δ is given as follows:

$$\begin{aligned}
&\delta((q_0^1, q_0^T, q_0^2), (q_0^1, q_0^T, q_0^2), a) \\
&= \{(\delta_1(q_0^1, q_0^1, a), \delta_T(q_0^T, r), q_0^2)\} \\
&= \{(q_1^1, q_r^T, q_0^2)\}
\end{aligned} \tag{21}$$

$$\begin{aligned}
&\delta((q_1^1, q_r^T, q_0^2), (q_0^1, q_0^T, q_0^2), a) \\
&= \{(q_1^1, \delta_T(q_r^T, u), \delta_2(q_0^2, q_0^2, a)\} \\
&= \{(q_1^1, q_u^T, q_1^2)\}
\end{aligned} \tag{22}$$

$$\begin{aligned}
&\delta((q_0^1, q_0^T, q_0^2), (q_1^1, q_r^T, q_0^2), a) \\
&= \{(\delta_1(q_0^1, q_1^1, a), \delta_T(q_u^T, \not{b}), q_0^2)\} \\
&= \{(q_1^1, q_u^T, q_0^2)\}
\end{aligned} \tag{23}$$

$$\begin{aligned}
&\delta((q_1^1, q_u^T, q_1^2), (q_0^1, q_0^T, q_0^2), a) \\
&= \{(\delta_1(q_1^1, q_0^1, a), \delta_T(q_u^T, r), q_1^2)\} \\
&= \{(q_1^1, q_r^T, q_1^2)\}
\end{aligned} \tag{24}$$

$$\begin{aligned}
&\delta((q_1^1, q_u^T, q_0^2), (q_1^1, q_u^T, q_1^2), a) \\
&= \{(q_1^1, \delta_T(q_r^T, \not{b}), \delta_2(q_0^2, q_1^2, a)\} \\
&= \{(q_1^1, q_r^T, q_1^2)\}
\end{aligned} \tag{25}$$

$$\begin{aligned}
&\delta((q_0^1, q_0^T, q_0^2), (q_1^1, q_r^T, q_0^2), b) \\
&= \{(\delta_1(q_0^1, q_1^1, b), \delta_T(q_r^T, \not{b}), q_0^2)\} \\
&= \{(q_2^1, q_r^T, q_0^2)\}
\end{aligned} \tag{26}$$

$$\begin{aligned}
&\delta((q_1^1, q_u^T, q_1^2), (q_0^1, q_0^T, q_0^2), b) \\
&= \{(\delta_1(q_1^1, q_0^1, b), \delta_T(q_u^T, r), q_1^2)\} \\
&= \{(q_2^1, q_r^T, q_1^2)\}
\end{aligned} \tag{27}$$

$$\begin{aligned}&\delta((q_2^1,q_r^T,q_1^2),(q_0^1,q_0^T,q_0^2),a)\\&=\{(q_2^1,\delta_T(q_r^T,u),\delta_2(q_1^2,q_0^2,a)\}\\&=\{(q_2^1,q_u^T,q_1^2)\}\end{aligned} \tag{28}$$

$$\begin{aligned}&\delta((q_1^1,q_u^T,q_1^2),(q_2^1,q_r^T,q_1^2),b)\\&=\{(\delta_1(q_1^1,q_2^1,b),\delta_T(q_u^T,r),q_1^2)\}\\&=\{(q_2^1,q_r^T,q_1^2)\}\end{aligned} \tag{29}$$

$$\begin{aligned}&\delta((q_2^1,q_r^T,q_0^2),(q_1^1,q_u^T,q_1^2),b)\\&=\{(q_2^1,\delta_T(q_r^T,\not{b}),\delta_2(q_0^2,q_1^2,b)\}\\&=\{(q_2^1,q_r^T,q_2^2)\}\end{aligned} \tag{30}$$

$$\begin{aligned}&\delta((q_2^1,q_r^1,q_0^2),(q_2^1,q_r^T,q_2^2),b)\\&=\{(q_2^1,\delta_T(q_r^T,\not{b}),\delta_2(q_0^2,q_2^2,b)\}\\&=\{(q_2^1,q_r^T,q_1^2)\}\end{aligned} \tag{31}$$

$$\begin{aligned}&\delta((q_2^1,q_r^T,q_1^2),(q_2^1,q_u^T,q_1^2),a)\\&=\{(q_2^1,\delta_T(q_r^T,\not{b}),\delta_2(q_1^2,q_1^2,a)\}\\&=\{(q_2^1,q_r^T,q_1^2)\}\end{aligned} \tag{32}$$

- *Theorem* 4. Let $T \subseteq \{r, u\}^{\omega} \cup \{\ell, d\}^{\omega}$ be a set of ω-trajectories. If T is a context free ωω-language and L_1, L_2 are all Regular Languages (RL), then $L_1 \text{ш}_T^{\omega} L_2$ is a context free ωω-language.

Authors construct a pushdown automaton $A=(\Sigma, \Gamma_1, \Gamma_2, Q, \delta, \delta^1, S, F, F', z_0, \$)$ such that $L(A)=L_1 \text{ш}_T^{\omega} L_2$, where Q is the set of states $Q=\overline{Q} \cup Q_1 \cup \ldots$ *Proof.* Let T, the set of ω-trajectories be a context free language. Consider two regular ωω-array languages L_1, L_2 over the same alphabet. Let $A_i=(\Sigma, \Gamma_i, Q_i, \delta_i, \delta_i^1, S_i, F_i, F_i^1, \$)$ be a finite state automata such that $L(A_i) =L_i$, for i=1, 2. Let $A_p=(\{r, u, \ell, d\}^{\omega}, \Gamma_T, Q_T, q_0^T, z_0^T, \Gamma_T, \delta_T)$ be a pushdown automaton such that $L(A_p)=T$, where Q_T is the set of states. Γ_T is the stack alphabet, $q_0^T \in Q_T$ is the initial state, $z_0^T \in \Gamma_T$ is the initial stack symbol, $\Gamma_T \subseteq Q_T$ is the set of states and δ_T is the transition mapping defined as $\delta_T : Q_T\times(\{(ru)^{\omega}, (\ell d)\omega\} \cup \{\lambda\}) \times\Gamma_T\to 2^{Q_T\times\Gamma_T}$.

$\cup\, Q_k$, $Q_i \cap Q_j = \phi$ if $i \neq j$. Each Q_i has an initial state q_i and a final state f_i. $F^1=\bigcup_{i=1}^{k}\{f_i\}$, $S=\bigcup_{i=1}^{k}\{q_i\}$, $\overline{Q}$ has an initial state q_0. $F \subseteq \overline{Q}$ is the set of final states, Γ_1 is the finite set of storage symbols. $|\Gamma_1|=k$ and each member of Γ_1 corresponds to one and only one Q_i. Γ_2 is the set of second storage symbols. $z_0 \in \Gamma_2$ is the initial symbol of the second storage. $\$ \notin \Sigma$ is the end marker.

Informally A on an input $p\in\Sigma^{\omega\omega}$ simulates non deterministically A_1 or A_2 and from time to time changes the simulation from A_1 to A_2 or from A_2 to A_1. Each change determines a transition in A_T as follows: A change from A_1 to A_2 is interpreted as u or d and a change from A_2 to A_1 is interpreted as r or ℓ respectively. The input p is accepted by A if and only if each of A_1, A_2 and A_p accepts L_1, L_2 and T respectively.

Formally $Q=Q_1\times Q_T\times Q_2$, $\overline{Q}=\{(q_0^1,q_T,q_0^2)\}$, $F=F_1\times F_T\times F_2$ and $F'=\{F_1^1\times Q_T\times k_{2i}\}\cup\{k_{1i}\times Q_T\times F_2^1\}$. The transition function δ is defined as $\delta : Q\times\{\Sigma \cup \{\varepsilon\}\} \cup \{\$\}\times\Gamma_1\to 2^{Q\times\Gamma_1\times\Gamma_2}$. The transition function δ is given as

$$\begin{aligned}&\delta((q_1^1,q_T,q_1^2),(q_2^1,q_T,q_2^2),a,a)\\&=\cup\{(\delta_1(q_1^1,q_2^1),a),\delta_T(q_T,\alpha_1),q_1^2),\varepsilon,\beta\}\\&\cup\{(q_1^1,\delta_T(q_T,\alpha_2),\delta_2((q_1^2,q_2^2),a),\varepsilon),\beta\}\end{aligned}$$ and

$$\begin{aligned}&\delta((q_1^1,q_T,q_1^2),(q_2^1,q_T,q_2^2),\$,a)\\&=\cup\{(\delta_1(q_1^1,q_2^1),\$),\delta_T(q_T,\alpha_1),q_1^2),s,\beta\}\\&\cup\{(q_1^1,\delta_T(q_T,\alpha_2),\delta_2((q_1^2,q_2^2),\$),s),\beta\}\end{aligned}$$

Where $q_1^1 \in Q_1$, $q_1^2 \in Q_2$, $q_T \in Q_T$, $\alpha_1 \in \{r, \ell\}$, $\alpha_2 \in \{u, d\}$, $a \in \Sigma \cup \{\lambda\}$, X, $\beta \in \Gamma_1^{*\omega}$. δ^1 is the mapping from $\overline{Q} \times (\Gamma_1 \cup \{\varepsilon_1\}) \times \Gamma_2$ into finite subsets of $\overline{Q} \times \Gamma_2^{*\omega}$. Clearly $L_1 \text{ш}_T^{\omega} L_2$ is a context free language. □

- *Example* 4. Consider the two recognizable infinite array languages L_1 and L_2 as in Example 3. The transition function for L_1 is $\delta_1 : Q_1\times Q_2\times\Sigma\to 2^Q$ is defined as follows.

$$\delta_1(q_{01}^1,q_{01}^2,a)=q_{11}^1 \tag{33}$$

$$\delta_1(q_{01}^1,q_{01}^1,a)=q_{11}^1 \tag{34}$$

$$\delta_1(q_{11}^1,q_{01}^1,a)=q_{11}^1 \tag{35}$$

$$\delta_1(q_{11}^1,q_{11}^1,a)=q_{11}^1 \tag{36}$$

$$\delta_1(q_{01}^1,q_{11}^1,b)=q_{21}^1 \tag{37}$$

$$\delta_1(q_{11}^1,q_{01}^1,b)=q_{21}^1 \tag{38}$$

$$\delta_1(q_{11}^1,q_{21}^1,b)=q_{21}^1 \tag{39}$$

$$\delta_1(q_{21}^1,q_{11}^1,b)=q_{21}^1 \tag{40}$$

$$\delta_1(q_{21}^1,q_{21}^1,b)=q_{21}^1 \tag{41}$$

$$\delta_1(q_{01}^1,q_{21}^1,b)=q_{21}^1 \tag{42}$$

$$\delta_1(q_{21}^1,q_{01}^1,b)=q_{21}^1 \tag{43}$$

Similarly the transition function for L_2 is δ_2 given as follows.

$$\delta_2(q_{01}^2,q_{01}^2,a)=q_{11}^2 \tag{44}$$

$$\delta_2(q_{11}^2,q_{01}^2,a)=q_{11}^2 \tag{45}$$

$$\delta_2(q_{01}^2,q_{11}^2,a)=q_{11}^2 \tag{46}$$

$$\delta_2(q_{11}^2,q_{11}^2,a)=q_{11}^2 \tag{47}$$

$$\delta_2(q_{11}^2,q_{01}^2,b)=q_{21}^2 \tag{48}$$

$$\delta_2(q_{11}^2,q_{21}^2,b)=q_{21}^2 \tag{49}$$

$$\delta_2(q_{21}^2,q_{01}^2,b)=q_{21}^2 \tag{50}$$

$$\delta_2(q_{21}^2, q_{11}^2, b) = q_{21}^2 \tag{51}$$

$$\delta_2(q_{21}^2, q_{21}^2, b) = q_{21}^2 \tag{52}$$

Authors take T=T$_C$={r^{m}u^m / m ≥ 1} a context free language. Corresponding to T is $\mathcal{A}_p$ =({r, u}, Γ$_T$, Q$_T$, q_0^T, z_0^T, F$_T$, δ$_T$). That is L(A$_p$)=T. The transition function is defined as

$$\delta_T(q_0^T, r, z_0^T) = (q_1^T, rz_0^T) \tag{53}$$

$$\delta_T(q_2^T, \cancel{b}, z_0^T) = (q_2^T, z_0^T) \tag{54}$$

$$\delta_T(q_1^T, \cancel{b}, r) = (q_1^T, r) \tag{55}$$

$$\delta_T(q_2^T, \cancel{b}, r) = (q_2^T, r) \tag{56}$$

$$\delta_T(q_1^T, r, r) = (q_1^T, rr) \tag{57}$$

$$\delta_T(q_1^T, u, r) = (q_2^T, \lambda) \tag{58}$$

$$\delta_T(q_2^T, u, r) = (q_2^T, \lambda) \tag{59}$$

$$\begin{gathered}\delta_T(q_2^T, \cancel{b}, u) = q_1^T \\ F_T = \{q_2^T\}.\end{gathered} \tag{60}$$

Let L = L$_1$ ш$_T$ L$_2$

$$= \begin{array}{cccccccccccc} \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ b & b & b & \cdot & \cdot & \cdot & b & b & b & b & \cdot & \cdot \\ b & b & b & \cdot & \cdot & \cdot & b & b & b & b & \cdot & \cdot \\ a & a & a & \cdot & \cdot & \cdot & a & a & b & a & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ a & a & a & \cdot & \cdot & \cdot & a & a & b & a & \cdot & \cdot \\ a & a & a & \cdot & \cdot & \cdot & a & a & b & a & \cdot & \cdot \end{array}$$

The automaton that recognizes L is the pushdown automaton $\mathcal{A}$ such that L($\mathcal{A}$) = L, where $\mathcal{A}$ = (Σ, Γ$_1$, Γ$_2$, Q, δ, δ′, S, F, F′, z$_0$, \$), Q = $\overline{Q}$ ∪ Q$_1$ ∪ ... ∪ Q$_k$, Q$_i$ ∩ Q$_j$ = ϕ if i ≠ j. The transition function \delta is defined as follows.

$$\begin{gathered}\delta((q_{01}^1, q_0^T, q_{01}^2), (q_{01}^1, q_0^T, q_{01}^2), a, z_0) \\ = \{(\delta_1(q_{01}^1, q_{01}^1, a), \delta_T(q_0^T, r, z_0^T), az_0, \varepsilon)\} \\ = \{((q_{11}^1, q_{11}^T, q_{01}^2), az_0, \varepsilon)\}\end{gathered} \tag{61}$$

$$\begin{gathered}\delta((q_{11}^1, q_1^T, q_{01}^2), (q_{11}^1, q_1^T, q_{01}^2), a, a) \\ = \{(q_{11}^1, \delta_T(q_1^T, u, r), \delta_2(q_{01}^2, q_{01}^2, a), a, \varepsilon)\} \\ = \{((q_{11}^1, q_2^T, q_{11}^2), a, \varepsilon)\}\end{gathered} \tag{62}$$

$$\begin{gathered}\delta((q_{11}^1, q_1^T, q_{01}^2), (q_{01}^1, q_1^T, q_{11}^2), a, a) \\ = \{(\delta_1(q_{11}^1, q_{01}^1, a), \delta_T(q_2^T, \cancel{b}, u), q_{11}^2, aa, \varepsilon)\} \\ = \{((q_{11}^1, q_1^T, q_{11}^2), aa, \varepsilon)\}\end{gathered} \tag{63}$$

$$\begin{gathered}((q_{11}^1, q_1^T, q_{01}^2), (q_{11}^1, q_1^T, q_{01}^2), a, b)) \\ = \{(q_{11}^1, \delta_T(q_1^T, u, r), \delta_2(q_{01}^2, q_{01}^2, a), b, \varepsilon)\} \\ = \{((q_{11}^1, q_2^T, q_{11}^2), b, \varepsilon)\}\end{gathered} \tag{64}$$

$$\begin{gathered}\delta((q_{11}^1, q_1^T, q_{01}^2), (q_{01}^1, q_1^T, q_{11}^2), a, b) \\ = \{(\delta_1(q_{11}^1, q_{01}^1, a), \delta_T(q_2^T, \cancel{b}, u), q_{11}^2, ab, \varepsilon)\} \\ = \{((q_{11}^1, q_1^T, q_{11}^2), ab, \varepsilon)\}\end{gathered} \tag{65}$$

$$\begin{gathered}\delta((q_{11}^1, q_1^T, q_{01}^2), (q_{01}^1, q_1^T, q_{11}^2), b, a) \\ = \{\delta_1(q_{11}^1, q_{01}^1, b), \delta_T(q_2^T, \cancel{b}, u), q_{11}^2, ba, \varepsilon\} \\ = \{(q_{21}^1, q_1^T, q_{11}^2), ab, \varepsilon\}\end{gathered} \tag{66}$$

$$\begin{gathered}\delta((q_{11}^1, q_1^T, q_{01}^2), (q_{01}^1, q_1^T, q_{11}^2), b, b) \\ = \{(\delta_1(q_{11}^1, q_{01}^1, b), \delta_T(q_2^T, \cancel{b}, u), q_{11}^2), bb, \varepsilon\} \\ = \{(q_{21}^1, q_2^T, q_{11}^2), bb, \varepsilon\}\end{gathered} \tag{67}$$

3. Applications

Shuffle on trajectories provides a useful tool to the study of a variety of problems in the area of parallel computation and in the theory of concurrency. there are many new problems of both theoretical and practical interest. An important problem seems to be the problem of parallelization of languages. Shuffle on trajectories offers a suitable theoretical framework to investigate this problem. Also the problem can be investigated with the turing complexity classes (time and space). Finding good parallelizations of problems can produce significant improvements with respect to the time used by a (one processor) computer to solve the problem. In this case the problem can be solved faster on a parallel computer.

Other aspects from the theory of concurrency and parallel computation such as priorities, the existence of critical sections, communication, the use of re-entrant routines are studied using semantic constraints on the shuffle operation. Of the special interest is to extend these operations for more complex objects such as graphs, networks or different types of automata.

4. Future Work

Shuffle on trajectories offers a suitable theoretical framework to investigate the problem of parallelization of languages. The Examples 3 and 4 deal with regular and context free languages respectively. For instance one can consider other intermediate classes of languages locally testable languages, linear languages, context sensitive languages, matrix languages, etc. Also the authors will study the literal shuffle on infinite arrays in future and fairness of shuffle on infinite trajectories in future.

5. Conclusions

Shuffle on infinite array languages provides a useful tool for the study of parallel computation and the theory of concurrency. The use of shuffle operation in the theory of concurrency and parallel composition is well known. This operation is used to yield formal languages. The shuffle on finite and infinite words have been investigated extensively but the study of shuffle operation on infinite arrays is in the initial stage. So in this paper the authors have made a attempt to study in depth the use of shuffle operation in different languages of infinite arrays to provide new classes of language of arrays and images. Authors have defined an array grammar with shuffle on trajectories over infinite arrays and obtained interesting results. Based on the studies [7, 11] authors have made an attempt to examine the effect of shuffle operation on ωω-recognizable languages and extended the shuffle operation to ωω-array languages.

References

[1] Arulprakasam R., Dare V., Gnanasekaran S., and Radhakrishnan M., "Local ω-Partial Languages," *in Proceedings of National Seminar on Algebra and Analysis Gateway to Modern Technology*, Kodambakkam, pp. 69-73, 2013.

[2] Chandra P., Martin-Vide C., Subramanian K., Van D., and Wang P., *Handbook of Pattern Recognition and Computer Vision*, World Scientific, 2004.

[3] Chandra P., Subramanian K., and Thomas D., "Parallel Contextual Array Grammars and Languages," *Electronic Notes in Discrete Mathematics*, vol. 12, pp.106-117, 2000.

[4] Christy D., Masilamani V., Thomas D., Atulya K., Nagar A., and Thamburaj R., "Shuffle on Array Languages Generated by Array Grammars," *Math Application*, vol. 3, pp. 17-31, 2014.

[5] Dare V., Annadurai S., Kalyani T., and Thomas D., "Partial Trajectories" *in Proceedings on National Conference of Mathematical and Computational Models*, Narosa Publications, pp. 418-426, 2007.

[6] Dare V., Annadurai S., Kalyani T., and Thomas D., "Trajectories P Systems," *Progress in Natural Sciences*, vol. 18, no. 5, pp. 611-616, 2008.

[7] Dare V., Subramanian K., Thomas D., and Siromoney R., "Infinite Arrays and Recognisability," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 14, no. 4, pp. 525-536, 2000.

[8] Geetha H., Thomas D., Kalyani T., and Venkatesan A., "Shuffle on Trajectories Over Finite Array Languages," *in Proceedings of International Workshop on Combinatorial Image Analysis*, Madrid, pp. 261-274, 2011.

[9] Giammaresi D. and Restivo A., *Handbook of Formal Languages*, Springer-Verlag, 1997.

[10] Jansirani N. and Dare V., "A Study on Diminishing Cells Infinite Array," *in Proceedings of 6th International Conference on Bio-Inspired Computing: Theories and Applications*, Penang, pp. 329-332, 2011.

[11] Kadrie A., Dare V., Thomas D., and Subramanian K., "Algebraic Properties of The Shuffle Over Omega Trajectories," *Information Processing Letters*, vol. 80, no. 3, pp. 139-144, 2001.

[12] Matescu A., Rozenberg G., and Salomaa A., "Shuffle on Trajectories Systactic Constraints," *Theoretical Computer Science*, vol. 197, no. 1-2, pp. 1-56, 1998.

[13] Nakamura A. and Ono H., "Pictures of Functions and their Acceptability by Automata," *Theoretical Computer Science*, vol. 23, no. 1, pp. 37-48, 1988.

[14] Park D., "Concurrency and Automata on Infinite Sequences," *in Proceedings of Theoretical Computer Science*, Karlsruhe, pp. 167-183, 1981.

[15] Perrin D. and Pin J., "Mots Infinis," *Report LITP*, Institut Blaise Pascal, 1993.

[16] Samira C. and Selma B., "Unmanned Vehicle Trajectory Tracking By Neural Networks," *The International Arab Journal of Information Technology*, vol. 13, no. 3, pp. 272-275, 2016.

[17] Siromoney R., Dare V., and Subramanian K., "Infinite Arrays and Infinite Computations," *Theoretical Computer Science*, vol. 24, no. 2, pp. 195-205, 1983.

[18] Siromoney R., Subramanian K., and Dare V., "On Infinite Arrays Obtained By Deterministic Controlled Table L-Array Systems," *Theoretical Computer Science*, vol. 33, pp. 3-11, 1984.

[19] Thomas W., *Handbook of Theoretical Computer Science*, MIT Press, 1990.

Devi Velayutham, M.Sc., M.Phil., B.Ed., Research Scholar, Sathyabama University, Chennai, India. She has 15 years of teaching experience in engineering colleges. Her research interest is automata theory and theory of computations.