Lattice Regular Grammar-Automata

A. John Kaspar, D. K. Sheena Christy*, D. G. Thomas

Abstract—**This paper studies some of the closure properties namely, homomorphism, inverse homomorphism, quotient and reversal of lattice languages. The appropriate tools to generate lattice languages such as lattice regular expressions, lattice regular grammar, lattice linear grammar and lattice regular grammar in normal form are defined. Also, the equivalences between lattice regular grammar, lattice left linear grammar, lattice right linear grammar and lattice grammar in normal form have been shown. The pumping lemma for lattice regular languages is established and used to prove that certain lattice languages are not lattice regular languages. Further, the equivalence of lattice finite automata and lattice regular grammar has been demonstrated.**

Index Terms—**Finite automata, Lattice automata, Lattice grammar, Lattice languages.**

1. Introduction

Finite automata are conceptual machines that determine whether or not a string (i.e., a sequence of characters) is part of a language. The automata-theoretic approach applies automata theory as a unifying paradigm for system verification, synthesis and specification [16], [17], [20]. Also, automata allows the algorithmic and logical parts of reasoning about systems to be separated, yielding asymptotically optimal algorithms. For reasoning on Boolean-valued systems, the approach of automata-theory has proven to be very useful as well as powerful. Automata are the key to some techniques namely partial-order verification, modular verification, on-the-fly verification, open systems, infinite-state systems, hybrid systems and verification of real time. There are automata-based solutions to many decision and synthesis problems for which no alternative solution exists. The academic as well as in industrials, automated-verification tools have used automata-based methodologies (for example, COSPAN and SPIN). But, in a number of new verification approaches involving reasoning about multi-valued Kripke structures, an atomic proposition is regarded as an element from a lattice rather than an element of Boolean value at a given state.

The multi-valued setting appears as a matter of course in systems where the designer can assign rich values to atomic propositions such as unknown, uninitialized, high impedance, logic 1, logic 0, don't care, etc.,[13]. This has indirect applications such as abstraction methods, where as the abstract system allows the atomic propositions and transitions to have unknown assignments [1], [7], verification of systems from varying viewpoints, where the value of the atomic propositions is the composition of their values in the different viewpoints [12] and query checking, where query checking reduced as a model checking over multi-valued Kripke structures[3]. Different forms of lattices are used for different purposes. To illustrate, in the application of abstraction, researchers have employed three values ordered as in L_3[6]. They also ordered its generalisation to linear ordering[4]. The elements of lattice are sets of formulas ordered by inclusion order in query checking[2]. When considering varying viewpoints, every viewpoint is represented by Boolean and composition of these viewpoints yields Boolean lattice products, such as $L_{2,2}$[6]. Finally, in systems having a wide range of atomic proposition values, different orders may be applied to the individual values that may not always result in a lattice.

It is acknowledged that traditional automata are Boolean because they accept or reject their input. On the other hand, weighted automata assign a value to each word taken from a semiring over a large domain. There is special case of weighted automata called lattice automata (multi valued objects) introduced by Kupferman and Lustig, in which the semiring is a finite lattice. They developed lattice automata for finite and infinite words. It has intriguing theoretical features as well as applications in formal languages. Closure properties namely, join (union), meet (intersection) and complementation are proved and decision problems for lattice languages through lattice automata have been studied. Also, it is proved that the results of lattice automata are distinct and superior to those of semi-ring and weighted automata. They have also investigated the complexity of constructions as well as decision problems for lattice automata with respect to the size of both the automaton and its corresponding lattice[15]. Some other theoretical properties of lattice automata such as minimization, approximation and bisimulation relation have been studied [5], [8], [9]. Whenever an automaton is used to define a family of languages, one gets interested in knowing what type of grammar is associated with it. Therefore, another popular technique to specify languages is called 'Grammars', used to describe the languages mathematically and has many interesting applications [10], [11], [18], [21]. Hence, to study lattice languages there is a need for grammar generating the lattice languages, which has not been studied in recent years. With this understanding, the common and powerful mechanism called lattice grammar for lattice language is introduced and studied along with their algebraic properties in this paper.

The paper is organized as follows: The basic notions and some closure properties of lattice languages are presented in Section 2. The closure properties such as homomorphism, inverse homomorphism, quotient and reversal of lattice languages are proved in Section 3, in Section 4, lattice regular expressions, lattice regular grammar, lattice left linear grammar, lattice right linear grammar and lattice

Manuscript received December 08, 2021; revised September 19, 2022.

A. John Kaspar is a Research Scholar in the Department of Mathematics, SRM Institute of Science and Technology, Kattankulathur - 603 203, India, e-mail: ja8952@srmist.edu.in.

*Corresponding Author - D. K. Sheena Christy is an Assistant Professor(Sr.G) in the Department of Mathematics, SRM Institute of Science and Technology, Kattankulathur - 603 203, India, e-mail: sheena.lesley@gmail.com.

D. G. Thomas is a Former Professor in the Department of Mathematics, Madras Christian College, Chennai - 600 059, India, e-mail: dgthomasmcc@yahoo.com.

regular grammar in normal form are introduced and studied. Also, the equivalence of lattice regular grammars, lattice left linear grammars, lattice right linear grammars and lattice regular grammars in normal form except for an empty string are proved. In addition to this, pumping lemma for lattice languages and the equivalence of lattice finite automata and lattice regular grammars have been demonstrated.

2. Preliminaries

The basic notions of finite automata and formal languages can be found in [14], [19]. Definitions with appropriate examples of lattices and its operations are discussed in [15]. The required notions and definitions of lattice automata have been recalled in this Section.

Definition 2.1. *[15]* Let $\mathcal{L}$ be a lattice and Z be a set of elements. An $\mathcal{L}$-set over Z is a function S from Z to $\mathcal{L}$, that is assigning a value from $\mathcal{L}$ to each element of Z.

Definition 2.2. *[15]* Consider a lattice $\mathcal{L}$ and Σ, a set of elements called alphabet. A lattice language L is a $\mathcal{L}$-set over Σ^*. Therefore, a lattice language L from Σ^* to $\mathcal{L}$ assigns a value from $\mathcal{L}$ to each word of L over Σ.

Definition 2.3. *[15]* A non-deterministic lattice automaton on finite words (LNFW) is a 6-tuple $\mathcal{A} = \{\mathcal{L}, \Sigma, Q, Q_0, \delta, F\}$, where $\mathcal{L}$, Σ and Q are a lattice, an alphabet and a finite set of states respectively, Q_0 in $\mathcal{L}^Q$ is a $\mathcal{L}$-set of start states, δ in $\mathcal{L}^{Q\times\Sigma\times Q}$ is a $\mathcal{L}$-transition-relation and F in L^Q is a $\mathcal{L}$-set of final states.

A run of an LNFW on a word $w = \sigma_1\sigma_2\ldots\sigma_n$ is a sequence of $n+1$ number of states $r = q_0.q_1\ldots q_n$. $val(r,w) = Q_0(q_0) \wedge \bigwedge_{j=0}^{n-1} \delta(q_j, \sigma_{j+1}, q_{j+1}) \wedge F(q_n)$ is the value of r on w. Clearly, $Q_0(q_0)$ is the value of q_0, q_0 being a start state, $\delta(q_j, \sigma_{j+1}, q_{j+1})$ is the value of q_{j+1} being the next state of q_j, when σ_{j+1} is an input alphabet, $F(q_n)$ is the value of q_n, q_n being the final state and the meet of all these values is the value of r with $0 \le j \le n-1$. We denote the value of traversal of r by $Q_0(q_0) \wedge \bigwedge_{j=0}^{n-1} \delta(q_j, \sigma_{j+1}, q_{j+1})$ and its acceptance value by $F(q_n)$. The value of LNFW $\mathcal{A}$ on a word w is denoted by $\mathcal{A}(w)$ and obtained by the join of the values of all the possible runs of $\mathcal{A}$ on the word w. i.e., $val(\mathcal{A}, w) = \vee\{val(r,w)/$ a run on w of $\mathcal{A}$ is $r\}$. The lattice language of LNFW $\mathcal{A}$ is denoted by $L(\mathcal{A})$, which maps each word w to its corresponding value in $\mathcal{L}$. That is, $L(\mathcal{A})(w) = val(\mathcal{A}, w)$.

Note: It is obvious that, in some cases the transition of the lattice automata is still on the same state after reading an input alphabet. i.e., $\delta(q_j, \sigma_{j+1}, q_j)$ is also possible. The extended transition function is given by $\delta^*(q_j, xa, q_i) = \delta^*(q_j, x, q_k)\delta(q_k, a, q_i)$, where $x \in \Sigma^*, a \in \Sigma$ and $q_i, q_j, q_k \in Q$.

Definition 2.4. *[15]* A deterministic lattice automaton on finite words (LDFW) is an LNFW, where there is only one state $q_0 \in Q$ such that $Q_0(q_0) \neq \perp$ and $\forall q' \in Q$ and $\sigma \in \Sigma$ there is only one state $q'' \in Q$ such that $\delta(q', \sigma, q'') \neq \perp$.

Note: $l_j : 0 \le j \le n-1$ is the value of q_{j+1} being the next state of q_j when σ_{j+1} is the input alphabet for the corresponding transition $\delta(q_j, \sigma_{j+1}, q_{j+1})$.

Theorem 2.1. *[15] Let $\mathcal{A}$ be a Non-deterministic lattice automaton on finite words (or Deterministic lattice automaton on finite words) with n number of states over $\mathcal{L}$ with m number of elements. There is a simple Non-deterministic lattice automaton on finite words (respectively, Deterministic lattice automaton on finite words) $\mathcal{A}'$ with nm number of states such that $L(\mathcal{A}') = L(\mathcal{A})$.*

Theorem 2.2. *[15] Let $\mathcal{A}$ be a Non-deterministic lattice automaton on finite words with n number of states, over $\mathcal{L}$ with m number of elements. There is a simple Deterministic lattice automaton on finite words $\mathcal{A}'$ with m^n number of states such that $L(\mathcal{A}') = L(\mathcal{A})$.*

3. Closure properties of lattice languages

In this section, the closure operations namely, homomorphism, inverse homomorphism, quotient and reversal of lattice languages have been defined and proved that the lattice languages are closed under homomorphism, inverse homomorphism, quotient with arbitrary sets and right quotient by any set. This section starts with recalling the closure properties namely, union, intersection and complementation of lattice languages studied in [15].

Theorem 3.1. *[15] Let $\mathcal{A}$ be an Non-deterministic lattice automaton on finite words with n number of states. There is a Non-deterministic lattice automaton on finite words $\mathcal{A}'$ with 2^n number of states such that $L(\mathcal{A}') = comp(L(\mathcal{A}))$. (i.e., lattice languages are closed under complementation).*

Theorem 3.2. *[15] Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be Deterministic lattice automata on finite words over $\mathcal{L}$. There are Deterministic lattice automata on finite words $\mathcal{A}_\wedge$ and $\mathcal{A}_\vee$ such that $L(\mathcal{A}_\vee) = L(\mathcal{A}_1) \vee L(\mathcal{A}_2)$ and $L(\mathcal{A}_\wedge) = L(\mathcal{A}_1) \wedge L(\mathcal{A}_2)$. If $\mathcal{A}_1$ and $\mathcal{A}_2$ has n_1 and n_2 number of states and $\mathcal{L}$ has m number of elements then $\mathcal{A}_\wedge$ has atmost $n_1n_2m^2$ and atleast n_1n_2m number of states. Also, $\mathcal{A}_\vee$ has n_1n_2 number of states.*

Definition 3.1. An onto function $f : \Sigma \to \Delta^*$ is called a homomorphism if for all $x, y \in \Sigma$, $f(xy) = f(x)f(y)$, in which Σ and Δ are alphabets. This homomorphism can be naturally extended to $f : \Sigma^* \to \Delta^*$ as $f(\lambda) = \lambda$, $f(\sigma x) = f(\sigma)f(x), \sigma \in \Delta$ and $x \in \Sigma^*$.

Theorem 3.3. *The class of all lattice languages is closed under homomorphism and inverse homomorphism.*

Proof: Let $f : \Sigma \to \Delta^*$ be a homomorphism and $L \subseteq \Sigma^*$ be a lattice language then there exists a lattice automaton $\mathcal{A} = (\mathcal{L}, Q, Q_0, \Sigma, \delta, F)$ such that $L(\mathcal{A}) = L$.

Now, construct a lattice automaton $\mathcal{A}' = (\mathcal{L}, Q, Q_0, \Delta, \delta', F)$, where $\delta' \in \mathcal{L}^{Q\times\Delta\times Q}$ is defined by $\delta'(q_0, \alpha, q) = l$ if and only if there is a word $w \in \Sigma^*$ such that $f(w) = \alpha$ and $\delta(q_0, w, q) = l$, where $q_0, q \in Q\alpha \in \Delta^*$ and $l \in \mathcal{L}$.

Let $\alpha \in L(\mathcal{A}')$. Then, $val(\mathcal{A}', \alpha) = \vee\{val(r, \alpha)/$ a run on α of $\mathcal{A}'$ is $r\}$, where $val(r, \alpha) = Q_0(q_0) \wedge \bigwedge_{j=1}^{n-1} \delta(q_j, \alpha_j, q_{j+1}) \wedge F(q_n)$ for some $q_0, q_n \in Q$.

Thus, there exists $w \in \Sigma^* \ni f(w) = \alpha$ and $val(r, w) = Q_0(q_0) \wedge \bigwedge_{j=1}^{n-1} \delta(q_j, w_j, q_{j+1}) \wedge F(q_n)$ for some $q_0, q_n \in Q$. Therefore, $val(\mathcal{A}, w) = \vee\{val(r, w)/$ a run on w of $\mathcal{A}$ is $r\}$.

Hence, $w \in L(\mathcal{A}) = L$

$\Rightarrow f(w) \in f(L)$
$\Rightarrow \alpha \in f(L)$
Similarly, the converse part also can be proved.

Let $L \subseteq \Delta^*$ be a lattice language accepted by the lattice automaton. Let $\mathcal{A}$ be a lattice automaton such that $L(\mathcal{A}) = L$.

Construct a lattice automaton $\mathcal{A}' = (\mathcal{L}, Q, Q_0, \Sigma, \delta', F)$, where $\delta' \in \mathcal{L}^{Q\times\Sigma\times Q}$ is defined by $\delta'(q_0, w, q) = \delta(q_0, f(w), q)$, for all $q_0, q \in Q, w \in \Sigma^*$. Then $L(\mathcal{A}) = L$.

Now, let $w \in L(\mathcal{A})$ then $val(\mathcal{A}, w) = \vee\{val(r,w)/$ a run on w of $\mathcal{A}$ is $r\}$, where

$val(r,w) = Q_0(q_0) \wedge \bigwedge_{j=1}^{n-1} \delta(q_j, w_j, q_{j+1}) \wedge F(q_n)$

$\Leftrightarrow val(\mathcal{A}, f(w)) = \vee\{val(r, f(w))/$ a run on $f(w)$ of $\mathcal{A}'$ is $r\}$, where $val(r, f(w)) = Q_0(q_0) \wedge \bigwedge_{j=1}^{n-1} \delta(q_j, f(w_j), q_{j+1}) \wedge F(q_n)$

$\Leftrightarrow f(w) \in L(\mathcal{A}) = L$
$\Leftrightarrow w \in f^{-1}(L)$. ∎

Definition 3.2. Let L_1 and L_2 be two lattice languages over the alphabet Σ then the quotient of L_1 and L_2 is defined as $L_1/L_2 = \{x \in \Sigma^* |$ there exists $y \in L_2$ such that $xy \in L_1\}$.

Theorem 3.4. *The class of lattice languages is closed under quotient with arbitrary sets.*

Proof: Let $L_1 \subseteq \Sigma^*$ be a lattice language and a set $L_2 \subseteq \Sigma^*$. Then there exists a lattice automaton $\mathcal{A} = (\mathcal{L}, Q, Q_0, \Sigma, \delta, F)$ such that $L(\mathcal{A}) = L_1$.

Construct a lattice automaton $\mathcal{A}' = (\mathcal{L}, Q, Q_0, \Sigma, \delta, F')$, where $F' : Q \to \mathcal{L}$ is defined by $\bigvee_{y\in L_2} \{\delta(q, y, q') \wedge F(q')\}$

Now, $x \in L(\mathcal{A}') \Leftrightarrow val(\mathcal{A}', x) = \vee\{val(r,x)/$ a run on x of $\mathcal{A}'$is $r\}$, where,

$val(r,x) = Q_0(q_0) \wedge \bigwedge_{j=1}^{n-1} \delta(q_j, \alpha_j, q_{j+1}) \wedge F'(q_n)$ for some $q_0, q_n \in Q$.

$\Leftrightarrow$ there exists $val(\mathcal{A}', xy) = \vee\{val(r, xy)/$ a run on xy of $\mathcal{A}'$ is $r\}$, where

$val(r,\alpha) = Q_0(q_0) \wedge \bigwedge_{j=1}^{n-1} \delta(q_j, xy, q_{j+1}) \wedge F(q_n')$ for some $q_0, q_n \in Q$ and $y \in L_2$.

$\Leftrightarrow xy \in L(\mathcal{A}) = L_1$, for some $y \in L_2$
$\Leftrightarrow x \in L_1/L_2$.
Similarly, the converse part also can be proved. ∎

Definition 3.3. Let $L_1, L_2 \subseteq \Sigma^*$. The right quotient of L_1 by L_2 is defined as $L_2^{-1}L_1 = \{x \in \Sigma^* |$ there exsist $y \in L_1$ such that $xy \in L_2\}$.

Theorem 3.5. *The class of lattice languages is closed under right quotient by arbitrary set.*

Definition 3.4. Let L be a lattice language over an alphabet Σ then the reversal of L is denoted by L^R and is defined by $L^R = \{w^R : \forall w \in \Sigma^*\}$.

Theorem 3.6. *The class of lattice regular is closed under reversal. i.e., If L is lattice regular language then L^R is also a lattice regular language.*

Proof: Let $L \subseteq \Sigma^*$ be a lattice language and $\mathcal{L}$ be a given lattice then there exists a lattice automaton $\mathcal{A} = (\mathcal{L}, Q, Q_0, \Sigma, \delta, F)$ such that $L(\mathcal{A}) = L$.

Now construct a lattice automaton $\mathcal{A}^R = (\mathcal{L}, Q^R, Q_0^R, \Sigma, \delta^R, F^R)$ such that $L(\mathcal{A}^R) = L^R$ where $Q^R = Q$, $Q_0^R = F$, $F^R = Q_0$ and δ^R is defined as by $\delta^R(q, w^R, p) = \delta(p, w, q) \forall p, q \in Q, w \in \Sigma^*$.

Let $w^R \in L(\mathcal{A}^R)$ then $val(\mathcal{A}^R, w^R) = \vee\{val(r, w^R)/$ a run on w^R of $\mathcal{A}^R$ is $r\}$, where

$val(r, w^R) = Q_0^R(q_0) \wedge \bigwedge_{j=1}^{n-1} \delta^R(q_j, w_j, q_{j+1}) \wedge F^R(q_n)$ for some $q_0, q_n \in Q^R$.

Thus, there exists $w \in \Sigma^*$ such that $w^R = w$ and $val(r,w) = Q_0(q_0) \wedge \bigwedge_{j=1}^{n-1} \delta(q_j, w_j, q_{j+1}) \wedge F(q_0)$ for some $q_0, q_n \in Q$. Therefore, $val(\mathcal{A}, w) = \vee\{val(r,w)/$ a run on w of $\mathcal{A}$ is $r\}$.

Hence, $w^R \in L(\mathcal{A}^R) \Rightarrow w^R \in L^R$. Similarly, the converse part also can be proved.

Therefore, the class of lattice regular language is closed under reversal. ∎

4. Lattice Regular Expressions and Grammar

In this section, lattice regular expressions, lattice grammar, lattice regular grammar, lattice right linear grammar, lattice left linear grammar and lattice grammar in normal form are defined. Also, described the pumping lemma for lattice languages, used to establish a necessary and sufficient condition for a given lattice language to be regular. Further, proved the equivalence between lattice finite automaton and lattice regular grammar by showing that they generate the same lattice language.

Definition 4.1. Let Σ be an given alphabet and $\mathcal{L}$ be an lattice, then the family of lattice regular expressions $\mathcal{R}$ over Σ is defined by the following ways:

- $\emptyset$ (empty set) $\in \mathcal{R}$
- λ (empty word) $\in \mathcal{R}$
- $x \in \mathcal{R}; \forall x \in \Sigma$
- $lx \in \mathcal{R}$ and $xl \in \mathcal{R}$ for all $l \in \mathcal{L}$ and $x \in \mathcal{R}$

If $x_1,\ x_2 \in \mathcal{R}$ then

- $x_1 + x_2 \in \mathcal{R}$
- $x_1x_2 \in \mathcal{R}$ and
- $x_1^* \in \mathcal{R}$.

Definition 4.2. Let $L_\mathcal{R}$ be the lattice language represented by lattice regular expressions $\mathcal{R}$ and is defined as follows:

- If $\emptyset \in \mathcal{R}$ then $L_\mathcal{R} = \{\emptyset\}$
- If $\lambda \in \mathcal{R}$ then $L_\mathcal{R} = \{\lambda\}$
- For all $x \in \mathcal{R}$, $L_\mathcal{R} = \{(x,l) : x$ drived from $\mathcal{R}$ and $l = val(\mathcal{R}, w) \in \mathcal{L}\}$

If x_1 and x_2 are lattice regular expressions then

- $L_\mathcal{R}(x_1 + x_2) = L_\mathcal{R}(x_1) \cup L_\mathcal{R}(x_2)$
- $L_\mathcal{R}(x_1x_2) = L_\mathcal{R}(x_1)L_\mathcal{R}(x_2)$ and
- $L_\mathcal{R}(x_1^*) = (L_\mathcal{R}(x_1))^*$.

Theorem 4.1. *If L is an lattice automaton language over Σ then $L = L_x$ for some lattice regular expression $x \in \mathcal{R}$.*

Proof: Let L be a lattice automaton language over Σ then there exists a lattice automaton $\mathcal{A} = (\mathcal{L}, Q, Q_0, \Sigma, \delta, F)$ such that $L = L(\mathcal{A})$. Consider that there are n number of states in Q and m number of alphabets in Σ such that $Q = \{q_1, q_2, \ldots, q_n\}$ and $\Sigma = \{w_1, w_2, \ldots, w_n\}$ respectively. For every $i, j = 1, 2, \ldots, n$, let $x_{ij}^0 = \lambda + w_s$, if $\delta(q_i, w_s, q_j)$ exists and $val(\mathcal{R}, w_s) = (q_{ij}) \vee (\bigwedge_{s=1}^{m} (q_i, w_s, q_j)); 1 \le s < m$ and $x_{ij}^k = x_{ij}^{k-1} + (x_{ik}^{k-1}(x_{kk}^{k-1})^* x_{kj}^{k-1}), k = 1, 2, \ldots, n$. By the principle of mathematical induction on k, it can be reduced that $L_{x_{ij}^k}(\lambda) = \top$ if $i = j$ and $\perp$ otherwise, where $\top, \perp \in \mathcal{L}$ represent top and bottom of the lattice $\mathcal{L}$ respectively. $L_{x_{ij}^k}(w) = \delta(q_1, w, q_2)$ for each $w \in \Sigma$ and for $m \in \mathbb{N}$,

$$L_{x_{ij}^k}(w_{i_0} w_{i_1} \dots w_{i_m}) = \vee_{i_1 \le k} \vee_{i_2 \le k} \cdots \vee_{i_m \le k} \delta(q_i, w_{i_0}, q_{i_1}) \wedge \delta(q_{i_1}, w_{i_1}, q_{i_2}) \wedge \cdots \wedge \delta(q_{i_m}, w_{i_m}, q_j).$$

Thus for all $w \in \Sigma^*$, $L_{x_{ij}^n}(w) = \delta^*(q_i, w, q_j)$.

Therefore, $L = L_x$ for some $x \in \mathcal{R}$.

■

Definition 4.3. A lattice grammar is a 5-tuple $\mathcal{G} = (\mathcal{L}, V, T, S, P)$, where

- $\mathcal{L}$ - a lattice
- V - finite set of alphabet called non-terminal symbols
- T - finite set of alphabet called terminal symbols
- S - lattice set($\mathcal{L}$-set) of V called start variables such that $S : V \to \mathcal{L}$
- P - finite set of lattice production rules ($\mathcal{L}$-production rules) over $V \cup T$ such that

$$P = \{A \xrightarrow{l} B : A, B \in V \cup T, l \in \mathcal{L} \text{ and atleast one Non-terminal must occur on the left side of the rule}\}$$

Definition 4.4. If $\alpha \xrightarrow{l} \beta$, $l \in \mathcal{L}$ is a $\mathcal{L}$-production rule then $A\alpha B \overset{l}{\Rightarrow} A\beta B$ means $A\alpha B$ directly drives $A\beta B$, where $\alpha, \beta, A, B \in (V \cup T)^*$.

If $A, A_1, \dots, A_n \in (V \cup T)^*$ and $A \xrightarrow{l_0} A_1$, $A_1 \xrightarrow{l_1} A_2$, $\dots$, $A_{n-1} \xrightarrow{l_n} A_n$ are $\mathcal{L}$-production rules of $\mathcal{G}$, where $l_0, l_1, \dots, l_n \in \mathcal{L}$ then A drives A_n can be written as $A \overset{l}{\underset{*}{\Rightarrow}} A_n$, where l is the meet of all $l_j : 0 \le j \le n$ and the expression $A \overset{l_0}{\Longrightarrow} A_1 \overset{l_1}{\Longrightarrow} A_2 \overset{l_2}{\Longrightarrow} \dots \overset{l_n}{\Longrightarrow} A_n$ is called derivation chain from A to A_n.

The value of the word of terminals w derivable from $A \in S$ is denoted by $val(A, w)$ and obtained by $val(A, w) = S(A) \wedge \bigwedge_{j=0}^{n} \{l_j\}$, where $l_j \in \mathcal{L}$ and $S(A)$ is the value of the Non-terminal of A, A being a star variable.

Definition 4.5. A lattice grammar $\mathcal{G}$ is said to be regular if each of its lattice production rules are of the form $A \xrightarrow{l} aB$ or $A \xrightarrow{l} a$, where $A, B \in V, a \in T$ and $l \in \mathcal{L}$.

Definition 4.6. A word w is said to be generated by the lattice regular grammar $\mathcal{G}$, if there exists atleast one derivation chain from A to w and the value of the word w is defined as

$$val(\mathcal{G}, w) = \vee\{val(A, w) : \text{ for all } A \in S.\}$$

Definition 4.7. The $\mathcal{L}$- regular language of the lattice regular grammar $\mathcal{G}$ is the set of all words generated by $\mathcal{G}$ and is denoted by $L(\mathcal{G})$. That is,

$$L(\mathcal{G}) = \{val(\mathcal{G}, w) : w \text{ generated by } \mathcal{G}\}$$

Definition 4.8. A lattice grammar $\mathcal{G} = (\mathcal{L}, V, T, S, P)$ is said to be in normal form if the $\mathcal{L}$-production rules is either of the form:

$A \xrightarrow{l} aB$ or $A \xrightarrow{l} \lambda$, where $A, B \in V$, $a \in T$, λ-empty alphabet and $l \in \mathcal{L}$.

Note: Each lattice grammar can be reduced to a lattice grammar in normal form by changing its $\mathcal{L}$-production rules of the form $A \xrightarrow{l} a$ by $A \xrightarrow{l} aC$ and $C \xrightarrow{l} \lambda$, where C is a newly added non-terminal, which is not in V.

Definition 4.9. A lattice grammar $\mathcal{G} = (\mathcal{L}, V, T, S, P)$ is called linear grammar, if the $\mathcal{L}$-productions rules are of the form $A \xrightarrow{l} w_1 B w_2$ or $A \xrightarrow{l} w$, where, $A, B \in V$, $l \in \mathcal{L}$ and $w_1, w_2, w \in T^*$.

If the $\mathcal{L}$-productions rules are of the form $A \xrightarrow{l} Bw$ or $A \xrightarrow{l} w$ then $\mathcal{G}$ is called left linear grammar and if the $\mathcal{L}$-production rules are in the form $A \xrightarrow{l} wB$ or $A \xrightarrow{l} w$ then $\mathcal{G}$ is called left linear grammar, where, $A, B \in V$, $l \in \mathcal{L}$ and $w_1, w_2, w \in T^*$.

Definition 4.10. A language $L \subseteq T^*$ is called lattice linear language, if there is lattice linear grammar $\mathcal{G}$ such that $L(\mathcal{G}) = L$.

The class of lattice regular language is a subclass of the class of lattice language.

Theorem 4.2. *Lattice left linear grammar and lattice right linear grammar generates the same language.*

Proof: Let $\mathcal{G} = (\mathcal{L}, V, T, S, P)$ be a lattice left linear grammar. Construct a lattice right linear grammar $\mathcal{G}' = (\mathcal{L}, V', T', S', P')$ with $\mathcal{L}$-production rules P' as follows:

1. $q_0 \xrightarrow{l} w$ in P' iff $q_0 \xrightarrow{l} w \in P$, $S(q_0) = l$.
2. $q_0 \xrightarrow{l} wA$ in P', for $S(q_0) = l$ iff $A \xrightarrow{l} w \in P$.
3. $A \xrightarrow{l} w$ and $A \xrightarrow{l} wq_0$ in P' iff $q_0 \xrightarrow{l} Aw \in P$, $S(q_0) = l$.
4. $A \xrightarrow{l} wB$ iff $B \xrightarrow{l} Aw \in P$, where $A, q_0 \in V$ and $w \in T^*$.

To prove $L(\mathcal{G}') = L(\mathcal{G})$:

Let $w \in L(\mathcal{G})$, where $w = w_1 w_2 \dots w_n; w_j \in T^*$ for $j = 1, 2, \dots, n$. Then, $val(\mathcal{G}, w) = \vee\{val(q_0, w) : \text{ for all } A \in S\}$, thus there exists $q_0 \in V$ such that $Q_0(q_0) \in \mathcal{L}$ and $val(q_0, w) = S(q_0) \wedge \bigwedge_{j=0}^{n-1} \{l_j\}$, in which each $l_j \in \mathcal{L}$.

If $q_0 \xrightarrow{l} w$ is a production in P for some $l \in \mathcal{L}$ then $q_0 \xrightarrow{l} w$ is in P' and $w \in L(\mathcal{G}')$. Otherwise, there exists $q_0, q_1, \dots, q_n \in V$ and $l_1, l_2, \dots, l_{n-1} \in \mathcal{L}$ such that $q_0 \overset{l_0}{\Longrightarrow} q_1 w_1 \overset{l_1}{\Longrightarrow} q_2 w_1 w_2 \overset{l_2}{\Longrightarrow} \dots \overset{l_{n-1}}{\Longrightarrow} w_1 w_2 \dots w_n = w$.

Now, corresponding to the above derivation chain, P must have the following $\mathcal{L}$-production rules $q_0 \xrightarrow{l_0} q_1 w_1$, $q_1 \xrightarrow{l_1} q_2 w_2$, $\dots$, $q_{n-2} \xrightarrow{l_{n-2}} q_{n-1} w_{n-1}$ and $q_{n-1} \xrightarrow{l_{n-1}} w_n$.

Therefore, P' should have the following lattice $\mathcal{L}$-production rules $q_0 \xrightarrow{l_0} w_1 q_1$, $q_1 \xrightarrow{l_1} w_2 q_2$, $\dots$, $q_{n-2} \xrightarrow{l_{n-2}} w_{n-1} q_{n-1}$ and $q_{n-1} \xrightarrow{l_{n-1}} w_n$.

Thus, there is a derivation chain for w in $\mathcal{G}'$ such that $q_0 \overset{l_0}{\Longrightarrow} w_1 q_1 \overset{l_1}{\Longrightarrow} w_1 w_2 q_2 \overset{l_2}{\Longrightarrow} \dots \overset{l_{n-1}}{\Longrightarrow} w_1 w_2 \dots w_n = w$.

Therefore, $val(\mathcal{G}', w) = \vee\{val(q_0, w) : \text{ for all } q_0 \in S'\}$, thus there exists $q_0 \in V'$ such that $Q_0(q_0) \in \mathcal{L}$ and $val(A, w) = Q_0(q_0) \wedge \bigwedge_{j=0}^{n-1} \{l_j\}$, in which each $l_j \in \mathcal{L}$. i.e., $w \in L(\mathcal{G}')$. Similarly, the converse of the theorem can be proved. ■

Theorem 4.3. *For every lattice grammar $G = (\mathcal{L}, V, T, P, S)$ in normal form, there is a lattice automata $\mathcal{A} = (\mathcal{L}, Q, Q_0, \Sigma, \delta, F)$.*

Proof: Consider $Q = V$, $Q_0 = S$, $\mathcal{L}$ be any lattice, define the set F such that $F : Q \to \mathcal{L}$ by $F(q_f) = l$ iff $q_f \xrightarrow{l} \lambda$ is a lattice production in P and $\delta \in \mathcal{L}^{Q\times\Sigma\times Q}$ by $\delta(q,a,p)=l$ iff $q \xrightarrow{l} ap$ in P.

Now, $\mathcal{A}=(\mathcal{L},Q,Q_0,\Sigma,\delta,F)$ is a lattice automaton.

Let $x \in L(\mathcal{G})$ and $w = w_1w_2\ldots w_n, \forall w_j \in T$. Then $val(\mathcal{G},w)=\vee\{val(A,w):$ for all $A\in S\}$, where $val(A,w)= S(A)\wedge \bigwedge_{j=0}^{n-1}\{l_j\}$, in which each $l_j\in\mathcal{L}$.

Now, $val(A,w)$ implies that there exists $q_0,q_1,\ldots,q_n\in V$ and $l_1,l_2,\ldots,l_{n-1}\in\mathcal{L}$ such that $q_0 \overset{l_0}{\Longrightarrow} w_1q_1 \overset{l_1}{\Longrightarrow} w_1w_2q_2 \overset{l_2}{\Longrightarrow} \ldots \overset{l_{n-1}}{\Longrightarrow} w_1w_2\ldots w_n = w$.

Then corresponding to the above derivation chain, the $\mathcal{L}$-production rule P has the following production rules:

$q_0 \xrightarrow{l_0} w_1q_1$, $q_1 \xrightarrow{l_1} w_2q_2$, $\ldots$, $q_{n-2} \xrightarrow{l_{n-2}} w_{n-1}q_{n-1}$ and $q_{n-1} \xrightarrow{l_{n-1}} w_n$, which gives the following transitions in δ of $\mathcal{A}$ such that $\delta(q_0,w_1,q_1)=l_0$, $\delta(q_1,w_2,q_2)=l_1$, $\ldots$, $\delta(q_{n-1},w_n,q_f)=l_{n-1}$.

Therefore, $val(\mathcal{A},w)=\vee\{val(r,w)/$ a run on w of $\mathcal{A}$ is $r\}$, where $val(r,w)=Q_0(q_0)\wedge\bigwedge_{j=0}^{n-1}\delta(q_j,w_{j+1},q_{j+1})\wedge F(q_f)$, for some $q_0\in Q_0$ and $q_f\in F$.

Thus, $w\in L(\mathcal{A})$.

Similarly, the converse part also can be proved.

■

Corollary 4.1. For every lattice grammar $\mathcal{G}=(\mathcal{L},V,T,S,P)$ in normal form, there is a lattice regular grammar $\mathcal{G}$ such that $L(\mathcal{G})=L(\mathcal{G})-\lambda$.

Theorem 4.4. *Every lattice right linear language can be generated by a lattice grammar in normal form.*

Proof: Let $\mathcal{G}=(\mathcal{L},V',T,S,P')$ be given lattice right linear grammar.

To prove the theorem, first construct a lattice regular grammar $\mathcal{G}_1=(\mathcal{L},V_1,T,S,P_1)$ such that $\mathcal{G}\sim\mathcal{G}_1$ and then construct a lattice grammar in normal form $\mathcal{G}'=(\mathcal{L},V',T,S,P')$ such that $\mathcal{G}'\sim\mathcal{G}_1$.

case(i):

If P has $\mathcal{L}$-production rules of the form $A\xrightarrow{l}wB$ or $A\xrightarrow{l}w$ of P with $|w|\le 1$ then put these rules in the set of $\mathcal{L}$-production rules P_1.

For the $\mathcal{L}$-production rules of P are in the form $A\xrightarrow{l}wB$ with $|w|>1$ and $w=w_1w_2\ldots w_n$ the $\mathcal{L}$-set production rules P_1 has following $\mathcal{L}$-production rules $A\xrightarrow{l}w_1Z_1$, $Z_1\xrightarrow{l}w_2Z_2$, $\ldots$, $Z_{n-1}\xrightarrow{l}w_nB$, where $Z_1,Z_2,\ldots,Z_{n-1}\in V_1$ which are not in V.

For the $\mathcal{L}$-production rules of P are in the form $A\xrightarrow{l} w_1w_2\ldots w_m, m\ge 2$ and $l\in\mathcal{L}$, the $\mathcal{L}$-productions rules P_1 has the following production rules $A\xrightarrow{l}w_1Y_1$, $Y_1\xrightarrow{l}a_2Y_2$, $\ldots$, $Y_m\xrightarrow{l}\lambda$, where $Y_1,Y_2,\ldots,Y_m\in V_1$ which are not in V.

Therefore, V_1 has set of all variables in V and also possesses new variables used in the above $\mathcal{L}$-production rules.

Thus, the lattice grammar $\mathcal{G}_1=(\mathcal{L},V_1,\Sigma,S,P_1)$ with P_1 contains the following types of lattice productions:

1. $A\xrightarrow{l}aB$
2. $A\xrightarrow{l}B$
3. $A\xrightarrow{l}\lambda$ $A,B\in V_1,\lambda\in\Sigma^*,a\in\Sigma$

To prove $\mathcal{G}\sim\mathcal{G}_1$, Let $w\in L(\mathcal{G})$ then $val(\mathcal{G},w)=\vee\{val(A,w):$ for all $A\in S\}$, where, $val(A,w)=Q_0(q_0)\wedge\bigwedge_{j=0}^{n-1}\{l_j\}$, in which each $l_j\in\mathcal{L}$ for some $q_0\in V$ such that $Q_0(q_0)\in\mathcal{L}$.

If $q_0\xrightarrow{l}x$ is a lattice production in P then $|w|=1$ then clearly $q_0\xrightarrow{l}w$ in P_1.

Now, if $|w|>1$ and $w=a_1a_2\ldots a_n;\forall a_i\in\Sigma$ then there exists $q_1,q_2,\ldots,q_{n-1}\in V'$ such that $q_0\xrightarrow{l}a_1q_1\xrightarrow{l} a_1a_2q_2,\ldots,q_{n-1}\xrightarrow{l}a_1a_2\ldots a_n=w$.

i.e., $q_0\overset{l}{\Rightarrow}x$ in $\mathcal{G}_1$

Hence, $w\in L(\mathcal{G}_1)$

Consider a word $w\in L(\mathcal{G})$ then $val(\mathcal{G},w)=\vee\{val(A,w):$ for all $A\in S\}$, where, $val(A,w)=Q_0(q_0)\wedge\bigwedge_{j=0}^{n-1}\{l_j:\forall l_j\in\mathcal{L}\}$ for some $q_0\in V'$ such that $Q_0(q_0)\in\mathcal{L}$.

Therefore, there is a derivation chain of w in $\mathcal{G}_1$ as given below

$q_0\xrightarrow{l_1}a_1A_1'\xrightarrow{l_2}a_1a_2A_2'\xrightarrow{l_3}\ldots\xrightarrow{l_k}a_1a_2\ldots a_nA_n'\xrightarrow{l_{l+1}}w_1q_1\xrightarrow{r_1} w_1b_1B_1'\xrightarrow{r_2}w_1b_1b_2B_2'\xrightarrow{r_3}\ldots\xrightarrow{r_m}w_1b_1b_2\ldots b_mB_m'\xrightarrow{r_{m+1}} w_1w_2q_2\xrightarrow{s_1}\ldots\xrightarrow{t_n}w_1w_2\ldots w_nq_n\xrightarrow{t_{n+1}}w_1w_2\ldots w_n\lambda=w$

Thus, the productions

$q_0\xrightarrow{l_1}a_1A_1'$, $A_1'\xrightarrow{l_2}a_2A_2'$, $\ldots$, $A_{i-1}'\xrightarrow{l_n}a_iA_{l_i}'$, $A_{i+1}'\xrightarrow{l_{i+1}} a_{i+1}A_{i+2}^i$, $\ldots$, $A_n'\xrightarrow{l_{i+1}}q_i$, $q_1\xrightarrow{r_1}b_1B_1'$, $B_1'\xrightarrow{r_2}b_2B_2'$, $\ldots$, $B_m'\xrightarrow{r_{m+1}}q_2$, $\ldots$, $C_{n-1}'\xrightarrow{t_n}w_nq_n$, $q_n\xrightarrow{t_{n+1}}\lambda$ are in the $\mathcal{L}$-production rules P_1.

Therefore, there is a derivation chain for w in $\mathcal{G}$ as shown below:

$q_0\xrightarrow{l}w_1q_1\xrightarrow{r}w_1w_2q_2\xrightarrow{s}\ldots\xrightarrow{t}w_1w_2\ldots w_nq_n\xrightarrow{t_{n+1}} w_1w_2\ldots w_n\lambda=w$, where $l=l_1\wedge l_2\wedge\cdots\wedge l_{k+1}$, $r=r_1\wedge r_2\wedge \cdots\wedge r_{m+1}$, $s=s_1\wedge s_2\wedge\cdots\wedge s_{u+1}$, $\ldots$, $t=t_1\wedge t_2\wedge\cdots\wedge t_{n+1}$.

Thus, there exists $q_0\in V_1$ such that $Q_0(q_0)$ and $val(A,w)$ in $\mathcal{G}$.

Therefore, $w\in L(\mathcal{G})$, hence $\mathcal{G}\sim\mathcal{G}_1$.

case(ii):

It is easy to see that V_1 contains variables in V and some new variables added in the above procedure for finding $\mathcal{G}_1$. Use an algorithm to eliminate all lattice rules of the form $A\xrightarrow{l}B$ as by given below:

Construct the set $U_i(A)=\{A\}$, for $A\in V'$, and $U_{i+1}= U_i(A)\cup\{B|B\xrightarrow{l}Z\in P_1$ for some $Z\in U_i(A),l\in\mathcal{L}\}$.

Since, V' is finite set, there exists an integer j such that $U_{j+k}(A)=U_k(A);k=1,2,\ldots$ and $U_j(A)$ denoted by $U(A)$; $\forall A\in V'$.

Now construct the lattice grammar $\mathcal{G}'=(V',\Sigma,S,P')$. P' is defined as follows:

$A\xrightarrow{r\wedge l}aB$ is in P' iff $\exists\, Z\in V'\ \ni\ A\in U(Z)$ and $Z\xrightarrow{l}aB\in P_1$, where $r=\wedge(r_1,r_2,\ldots,r_{m+1})$.

$A\xrightarrow{r\wedge l}\lambda$ is in P' iff $\exists\, Z\in V'\ \ni\ A\in U(Z)$ and $Z\xrightarrow{l}\lambda\in P_1$, where $r=\wedge(r_1,r_2,\ldots,r_{m+1})$.

Clearly, the lattice grammar $\mathcal{G}'=(\mathcal{L},V',T,S,P')$ is in normal form. Therefore, it is simple to show that $\mathcal{G}_1\sim\mathcal{G}'$.

■

Corollary 4.2. *Lattice right linear grammar is equivalent to lattice regular grammar except for λ.*

Theorem 4.5. *The following statements are equivalent except that for λ.*

- *lattice regular grammar*
- *lattice left linear grammar*

- *lattice right linear grammar*
- *lattice grammar in normal form.*

The proof of the above theorem can be easily obtained from results of the theorem 4.2, theorem 4.4, corollary 3.1 and corollary 3.2. The following lemma gives necessary and sufficient condition for a given lattice language to be regular.

Lemma 4.1. *(Pumping lemma) Let Σ be a given alphabet. If $L \subseteq \Sigma^*$ be a lattice language over a lattice $\mathcal{L}$ then there exits a positive integer m, where if for each word $w \in \Sigma^*$ with $|w| \geq m$ then w can be decomposed as $w = xyz$ in such a way that $|y| \geq 0$, $|xy| \leq m$ and for each $i \geq 0$, $L(\mathcal{A})(xyz) = L(\mathcal{A})(xy^iz)$.*

Proof: If L is lattice language then there exists a lattice automaton $\mathcal{A} = \{\mathcal{L}, \Sigma, Q, Q_0, \delta, F\}$ such that $L(\mathcal{A}) = L$. Let $m \geq 0$ be the number of states in $\mathcal{A}$.

Let $w = w_1w_2 \ldots w_i \ldots w_n$ be a word in Σ^* with $n \geq m$ and $L(\mathcal{A})(w) = l$, $l \in \mathcal{L}$. For $s = 1, 2, \ldots, n$, let $\delta^*(q_0, w_1w_2 \ldots w_s) = q_s$. There exists two integers j and k, $0 \leq j < k \leq m$ such that $q_j = q_k$, since there are $m+1$ states in the sequence of states $q_0, \ldots, q_m$. Thus $x = w_1w_2 \ldots w_j$, $y = w_{j+1}w_{j+2} \ldots w_k$ and $x = w_{k+1}w_{k+2} \ldots w_n$ then $w = xyz$ with $|y| \geq 1$ and $|xy| = k \leq m$ and from

$$\begin{aligned}
&\delta^*(q_0, w_1w_2 \ldots w_jw_{j+1} \ldots w_kw_{k+1} \ldots w_n) \\
&= \delta^*(\delta^*(\delta^*(q_0, (w_1w_2 \ldots w_j))w_{j+1} \ldots w_k)w_{k+1} \ldots w_n) \\
&= \delta^*(\delta^*(q_j, w_{j+1}w_{j+2} \ldots w_k)w_{k+1} \ldots w_n) \\
&= \delta^*(q_k, w_{k+1} \ldots w_n) \\
&= q_n
\end{aligned}$$

and for any $i \geq 0$

$$\begin{aligned}
&\delta^*(q_0, w_1w_2 \ldots w_j(w_{j+1} \ldots w_k)^iw_{k+1} \ldots w_n) \\
&= \delta^*(\delta^*(\delta^*(q_0, (w_1w_2 \ldots w_j))(w_{j+1} \ldots w_k)^i)w_{k+1} \ldots w_n) \\
&= \delta^*(\delta^*(q_j, (w_{j+1}w_{j+2} \ldots w_k)^i)w_{k+1} \ldots w_n) \\
&= \delta^*(q_k, w_{k+1} \ldots w_n) \\
&= q_n
\end{aligned}$$

Then for $i \geq 0$, $L(\mathcal{A})(xyz) = l = L(\mathcal{A})(xy^iz)$.

■

The example given below uses the above stated pumping lemma and illustrates how a given lattice language is not regular.

Example 4.1. Consider a lattice language $L \subseteq \Sigma^*$ over a lattice $\mathcal{L}$, where L consists of strings of the form $w = a^nb^n$ with lattice value l in which $a, b \in \Sigma, n \in \mathbb{N}$ and $l \in \mathcal{L}$.

Now, consider a string $w = a^nb^n$ with $n \geq m$, where $L(w) = l, l \in \mathcal{L}$. By pumping lemma, decompose w into xyz with $|y| \geq 1$ and $|xy| \leq m \leq n, n, m \in \mathbb{N}$ such that $L(xyz) = L(xy^iz)$ for every $i \geq 0$. Therefore, $x = a^r$, $y = a^s$, $z = a^tb^n$ with $r \geq 0$, $s \geq 0$, $t \geq 0$ and $r+s+t = n$, since the condition implies that $|xy| \leq m \leq n$. A contradiction arise because $L(xy^0z) = L(a^{n-s}b^n) \neq l$, $l \in \mathcal{L}$. Hence, the lattice language L is not regular.

Theorem 4.6. *Let $\mathcal{G} = (\mathcal{L}, V, T, S, P)$ be a lattice grammar, then there exists a lattice finite automaton $\mathcal{A}(\mathcal{L}, \Sigma, Q, Q_0, \delta, F)$ such that $L(\mathcal{A}) = L(\mathcal{G})$.*

Proof: Given a lattice regular grammar $\mathcal{G} = (\mathcal{L}, V, T, S, P)$. Now construct a lattice automaton $\mathcal{A} = \{\mathcal{L}, \Sigma, Q, Q_0, \delta, F\}$, where $Q = V \cup \{q_f\}$, $\Sigma = T$, $S = Q_0$, $\mathcal{L}$ be any lattice, $F = \{q_f : F(q_f) = \top\}$ and the $\mathcal{L}$-transition-relation $\delta \in \mathcal{L}^{Q \times \Sigma \times Q}$ is defined as

$\delta(q, a, p) = l$ iff $q \xrightarrow{l} ap$, where $q, p \in V, a \in T$ and $l \in \mathcal{L}$

$\delta(q, a, q_f) = l$ iff $q \xrightarrow{l} a$, where $q \in V, a \in T$ and $l \in \mathcal{L}$.

To prove $L(\mathcal{G}) = L(\mathcal{A})$, let $w = w_1w_2 \ldots w_n \in L(\mathcal{G})$ and $val(\mathcal{G}, w) \in \mathcal{L}$ then there exists $q_0 \in V$ such that $S(q_0) \in \mathcal{L}$.

Now, $val(\mathcal{G}, w)$ implies that there exists at least one derivation chain of the form $q_0 \xRightarrow{l_0} w_1q_1 \xRightarrow{l_1} w_1w_2q_2 \xRightarrow{l_2} \ldots \xRightarrow{l_{n-1}} w_1w_2 \ldots w_n = w$, where, $l_j \in \mathcal{L}$, $q_j \in Q$ for $0 \leq j \leq n-1$ and $q_0 \xrightarrow{l_0} w_1q_1$, $q_1 \xrightarrow{l_1} w_2q_2$, $\ldots$, $q_{n-2} \xrightarrow{l_{n-2}} w_{n-1}q_{n-1}$ and $q_{n-1} \xrightarrow{l_{n-1}} w_n$ all are in P of $\mathcal{G}$.

That is, $q_0 \overset{l}{\underset{*}{\Rightarrow}} w$, $l = \wedge\{l_j : 0 \leq j \leq n\}$.

Therefore, there is a sequence of $\mathcal{L}$-transition-relations such that $\delta(q_0, w_1, q_1) = l_0$, $\delta(q_1, w_2, q_2) = l_1$, $\ldots$, $\delta(q_{n-1}, w_n, q_f) = l_{n-1}$.

Therefore, for a word w, $val(r, w) = Q_0(q_0) \wedge \bigwedge_{j=0}^{n-1} \delta(q_j, w_{j+1}, q_{j+1}) \wedge F(q_n)$, where $q_0 \in Q_0$ and $q_n \in F$.

Thus, $val(\mathcal{A}, w) = \vee\{val(r, w)/$ a run on w of $\mathcal{A}$ is $r\}$.

Therefore $w \in L(\mathcal{A})$.

The converse is similarly proved. ■

Example 4.2. Consider the lattice regular grammar $\mathcal{G} = (\mathcal{L}, V, T, S, P)$ where,

$\mathcal{L} = \langle 0, 1, 2, 3, \leq \rangle$, $V = \{q_0, q_1, q_2, q_3\}$, $T = \{a, b\}$, $S = \{q_0\}$ with $S(q_0) = 3$ and the $\mathcal{L}$- production rules P is defined as follows:

$P = \{q_0 \xrightarrow{1} aq_2, q_2 \xrightarrow{2} bq_2, q_2 \xrightarrow{2} bq_3, q_3 \xrightarrow{3} a, q_0 \xrightarrow{1} bq_1, q_1 \xrightarrow{2} aq_1, q_1 \xrightarrow{2} bq_1\}$.

Construct a lattice finite automaton $\mathcal{A} = (\mathcal{L}, Q, \Sigma, \delta, Q_0, F)$ where,

$\mathcal{L} = \langle 0, 1, 2, 3, \leq \rangle$, $\Sigma = T$, $Q = V \cup \{q_f\}$, $Q_0 = F$, $F = \{q_f\}$ with $F(q_f) = 3$ and the $\mathcal{L}$-transition-relations are defined by P as follows:

$\delta(q_0, a, q_2) = 1$, $\delta(q_2, b, q_2) = 2$, $\delta(q_2, b, q_3) = 2$, $\delta(q_3, a, q_f) = 3$, $\delta(q_0, b, q_1) = 1$, $\delta(q_1, a, q_1) = 2$, $\delta(q_1, b, q_1) = 2$.

Theorem 4.7. *Given a lattice finite automaton $\mathcal{A} = (\mathcal{L}, \Sigma, Q, Q_0, \delta, F)$, there exists a lattice grammar $\mathcal{G}(\mathcal{L}, V, T, S, P)$ such that $L(\mathcal{G}) = L(\mathcal{A})$.*

Proof: Given a lattice automaton $\mathcal{A} = \{\mathcal{L}, \Sigma, Q, Q_0, \delta, F\}$, construct a lattice regular grammar $\mathcal{G} = (\mathcal{L}, V, T, S, P)$, where $V = Q$, $T = \Sigma$, $Q_0 = S$, $\mathcal{L}$ be any lattice and the $\mathcal{L}$-production rules of P are defined as

$P = \{q \xrightarrow{l} ap$ iff $\delta(q, a, p) = l$, where $q, p \in Q$, $a \in \Sigma$ and $l \in \mathcal{L}\}$

$P = \{q \xrightarrow{l} a$ iff $\delta(q, a, q_f) = l$, where $q, q_f \in Q$, $a \in \Sigma$ and $l \in \mathcal{L}\}$

To prove $L(\mathcal{A}) = L(\mathcal{G})$.

Let $w \in L(\mathcal{A})$ and $val(\mathcal{A}, w) = \vee\{val(r, w)/$ a run on w of $\mathcal{A}$ is $r\}$, where $val(r, w) = Q_0(q_0) \wedge \bigwedge_{j=0}^{n-1} \delta(q_j, w_{j+1}, q_{j+1}) \wedge F(q_f)$, for some $q_0 \in Q_0$ and $q_f \in F$. Then there exists $q_0, q_1, \ldots, q_f \in Q$ such that $\delta(q_0, w_1, q_1) = l_0$, $\delta(q_1, w_2, q_2) = l_1$, $\ldots$, $\delta(q_{n-1}, w_n, q_f) = l_{n-1}$ are all in δ of $\mathcal{A}$.

Therefore, corresponding to the above transitions we have the $\mathcal{L}$-production rules as follows:

$$q_0 \overset{l_0}{\Longrightarrow} w_1q_1 \overset{l_1}{\Longrightarrow} w_1w_2q_2 \overset{l_2}{\Longrightarrow} \ldots \overset{l_{n-1}}{\Longrightarrow} w_1w_2\ldots w_n = w$$

where, $l_j \in \mathcal{L}$ for $0 \leq j \leq n-1$ and $q_0 \xrightarrow{l_0} w_1q_1$, $q_1 \xrightarrow{l_1} w_2q_2$, $\ldots$, $q_{n-2} \xrightarrow{l_{n-2}} w_{n-1}q_{n-1}$ and $q_{n-1} \xrightarrow{l_{n-1}} w_n$ all are in P of $\mathcal{G}$.

That is, $q_0 \overset{l}{\underset{*}{\Rightarrow}} w$, $l = \wedge\{l_j : 0 \leq j \leq n-1\}$ and $val(\mathcal{G}, w) = \vee\{val(A,w) :$ for all $A \in S\}$, where $val(A,w) = S(A) \wedge \bigwedge_{j=0}^{n-1}\{l_j\}$, in which each $l_i \in \mathcal{L}$.

Therefore, $w \in L(\mathcal{G})$.

The converse is similarly proved. ■

5. Conclusion

Lattice automata and the lattice languages accepted by it have interesting theoretical characteristics as well as applications in various fields such as query checking, abstraction method and quantitative verification. In this paper, the generating mechanism called lattice grammar for lattice languages has been introduced and certain specific closure properties of lattice languages have been proved. Lattice regular grammar, lattice left linear grammar, lattice right linear grammar and lattice grammar in normal form are defined and proved that they are equivalent. Also defined lattice regular expressions for lattice languages. Further, pumping lemma for lattice languages, used to establish a necessary and sufficient condition for a given lattice language to be regular has been proved. The equivalence between lattice finite automata and lattice regular grammar has also been proven.

References

[1] G. Bruns and P. Godefroid, "Model Checking Partial State Spaces with 3-Valued Temporal Logics," *Proceedings of the 11th Conference on Computer Aided Verification volume 1633 of Lecture Notes in Computer Science*, 1999, pp. 274 - 287.

[2] G. Bruns and P. Godefroid, "Temporal logic query checking," *Proceedings 16th Annual IEEE Symposium on Logic in Computer Science*, 2001, pp. 409 - 417.

[3] W. Chan, "Temporal-logic queries," *Proceedings of the 12th Conference on Computer Aided Verification volume 1855 of Lecture Notes in Computer Science*, 2000, pp. 450 – 463.

[4] M. Chechik, B. Devereux and A. Gurfinkel, "Model-checking infinite state-space systems with fine-grained abstractions using SPIN," in *SPIN Workshop in model-checking software volume 2057 of Lecture Notes in Computer Science*, 2001, pp. 16 - 36.

[5] M. Ciric, J. Ignjatovic, N. Damljanovic and M. Basic, "Bisimulations for fuzzy automata," *Fuzzy Sets and Systems*, vol. 186, no. 1 pp. 100 – 139, 2012.

[6] S. Easterbrook and M. Chechik, "A framework for multi-valued reasoning over inconsistent viewpoints," *Proceedings of the 23rd International Conference on Software Engineering*, 2001, pp. 411-420.

[7] S. Graf and H. Saidi, "Construction of abstract state graphs with PVS," *Proceedings of the 9th Conference on Computer Aided Verification volume 1254 of Lecture Notes in Computer Science*, 1997, pp. 72–83.

[8] S. Halamish and O. Kupferman, "Minimizing deterministic lattice automata," *Proceedings of the 14th FoSSaCS volume 6604 of Lecture Notes in Computer Science*, 2011, pp. 199 - 213.

[9] S. Halamish and O. Kupferman, "Approximating deterministic lattice automata," In 10th International Symposium on Automated Technology for Verification and Analysis volume 7561 of Lecture Notes in Computer Science, 2012, pp. 27 – 41.

[10] S. Hamdi, A. B. Abdallah and M. H. Bedoui, "A robust QRS complex detection using regular grammar and deterministic automata," *Biomed. Signal Process. Control*, vol. 40, pp. 263 - 274, 2018.

[11] V. X. Heavens, "Metamorphism, formal grammar and undecidable code mutation," *International Journal of Computer Science*, vol. 2, no. 1, pp. 70 - 75, 2007.

[12] A. Hussain and M. Huth, "On model checking multiple hybrid views," *Technical Report TR-2004-6*, University of Cyprus, 2004.

[13] IEEE standard multivalue logic system for VHDL model interoperability (std logic 1164), 1993.

[14] John E. Hopcroft, Rajeev Motwani and Jeffrey D. Ullman, *Introduction to Automata Theory, Languages, and Computation*, Pearson Education Ltd., 3rd edition, 2014.

[15] O. Kupferman, Y. Lustig, *Lattice automata*, in: Proc. VWCAI2007, Lecture Notes in Computer Science, Vol. 4349, Springer, Berlin, 2007, pp. 199 – 213.

[16] O. Kupferman, M. Y. Vardi and P. Wolper, "An automata-theoretic approach to branching-time model checking," *Journal of the ACM*, vol. 47, no. 2, pp. 312 – 360, 2000.

[17] R. P. Kurshan, *Computer-aided verification of coordinating processes: the automata-theoretic approach,*. Princeton Univ. Press, 1994.

[18] R. Nakano, "Error correction of enumerative induction of deterministic context-free L-systems grammar," *IAENG International Journal of Computer Science*, vol. 40, no. 1, pp. 47 - 52, 2013.

[19] Peter Linz, *An introduction to formal languages and automata*, Jones & Bartlett Learning, 2006.

[20] M. Y. Vardi, P. Wolper, "Reasoning about infinite computations," *Information and Computation*, vol. 115, no. 1, pp. 1 – 37, 1994.

[21] J. Zhang and Z. Qian, "The equivalent conversion between regular grammar and finite automata,"*Journal of Software Engineering and Applications*, vol. 6, no. 1, pp. 33 - 37, 2013.