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, 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 impendence, 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 [11], [17], 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 $L_2$ [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 $L_2$ [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: jas9952@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: dgthomasnmc@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 \( L \) be a lattice and \( Z \) be a set of elements. An \( L \)-set over \( Z \) is a function \( S \) from \( Z \) to \( L \), that is assigning a value from \( L \) to each element of \( Z \).

Definition 2.2. [15] Consider a lattice \( L \) and \( \Sigma \), a set of elements called alphabet. A lattice language \( L \) is a \( L \)-set over \( \Sigma \). Therefore, a lattice language \( L \) from \( \Sigma \) to \( L \) assigns a value from \( L \) to each word of \( L \) over \( \Sigma \).

Definition 2.3. [15] A non-deterministic lattice automaton on finite words (LNFW) is a 6-tuple \( A = (\Sigma, Q, Q_0, \delta, F) \), where \( L, \Sigma \) and \( Q \) are a lattice, an alphabet and a finite set of states respectively, \( Q_0 \in L^Q \) is a \( L \)-set of start states, \( \delta \) in \( L^{Q \times \Sigma \times Q} \) is a \( L \)-transition relation and \( F \) in \( L \) is a \( L \)-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 \( r = q_0, q_1, \ldots, q_n \), \( \text{val}(r, w) = Q_0(q_0) \land \bigwedge_{j=0}^{n-1} \delta(q_j, \sigma_{j+1}, q_{j+1}) \land F(q_n) \) 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 \( q_j \) being the next state of \( q_j \), when \( \sigma_{j+1} \) 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 \( 0 \leq j \leq n-1 \). We denote the value of traversal of \( r \) by \( Q_0(q_0) \land \bigwedge_{j=0}^{n-1} \delta(q_j, \sigma_{j+1}, q_{j+1}) \) and its acceptance value by \( F(q_n) \). The value of LNFW \( A \) on a word \( w \) is denoted by \( A(w) \) and obtained by the join of the values of all the possible runs of \( A \) on the word \( w \), i.e., \( \text{val}(A, w) = \bigvee \{ \text{val}(r, w) / a \) run on \( w \) of \( A \) is \( r \}. \) The lattice language of LNFW \( A \) is denoted by \( L(A) \), which maps each word \( w \) to its corresponding value in \( L \). That is, \( L(A)(w) = \text{val}(A, w) \).

Note: It is obvious that, in some cases the transition of the lattice automaton 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 \( \delta'(q_j, x, q_j) = \delta'(q_j, x, q_j) = \delta(q_j, x, q_j) = \delta(q_j, \sigma_{j+1}, q_{j+1}) \), where \( x \in \Sigma, a \in \Sigma \) and \( q_j, q_{j+1} \in Q \).

Definition 2.4. [15] A deterministic lattice automaton on finite words (DLFW) is an LNFW, where there is only one state \( q_0 \in Q \) such that \( Q_0(q_0) = q_0 \) 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: If \( L \leq j \leq n-1 \) is the value of \( q_{j+1} \) being the next state of \( q_j \) when \( \sigma_{j+1} \) is the input alphabet for the corresponding transition \( \delta(q_j, \sigma_{j+1}, q_{j+1}) \).

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

Theorem 2.2. [15] Let \( A \) be a Non-deterministic lattice automaton on finite words with \( n \) number of states, over \( \Sigma \) with \( m \) number of elements. There is a simple Deterministic lattice automaton on finite words \( A' \) with \( mn^2 \) number of states such that \( L(A') = L(A) \).

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 \( A \) be a Non-deterministic lattice automaton on finite words with \( n \) number of states. There is a Non-deterministic lattice automaton on finite words \( A' \) with \( 2^n \) number of states such that \( L(A') = \text{comp}(L(A)) \), (i.e., lattice languages are closed under complementation).

Theorem 3.2. [15] Let \( A_1 \) and \( A_2 \) be Deterministic lattice automata on finite words over \( \Sigma \). There are Deterministic lattice automata on finite words \( A_3 \) and \( A_4 \) such that \( L(A_1) \lor L(A_2) \) and \( L(A_3) \land L(A_2) \). If \( A_1 \) and \( A_2 \) has \( n_1 \) and \( n_2 \) number of states and \( L \) has \( m \) number of elements then \( A_3 \) has almost \( n_1n_2m^2 \) and at most \( n_1n_2n^2 \) number of states. Also, \( A_4 \) has \( n_1n_2m \) number of states.

Definition 3.1. An onto function \( f : \Sigma \rightarrow \Delta^* \) is called a homomorphism if for all \( x, y \in \Sigma \), \( f(xy) = f(x)f(y) \), in which \( \Sigma \) and \( \Delta \) are alphabets. This homomorphism can be naturally extended to \( f : \Sigma^* \rightarrow \Delta^* \) as \( f(\lambda) = f(1) \), \( f(\sigma x) = f(\sigma)f(x) \), \( \sigma \in \Delta \) and \( x \in \Sigma^* \).

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

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

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

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

Thus, there exists \( w \in \Sigma^* \), \( f(w) = \alpha \) and \( \text{val}(r, w) = Q_0(q_0) \land \bigwedge_{j=1}^{n-1} \delta(q_j, \sigma_j, q_{j+1}) \land F(q_n) \) for some \( q_0, q_n \in Q \).

Therefore, \( \text{val}(A, w) = \bigvee \{ \text{val}(r, w) / a \) run on \( w \) of \( A \) is \( r \}. \) Hence, \( w \in L(A)(L) \).
\[ f(w) \in f(L) \]
\[ \alpha \in f(L) \]

Similarly, the converse part also can be proved.

Let \( L \subseteq \Sigma^* \) be a lattice language accepted by the lattice automaton. Let \( A \) be a lattice automaton such that \( L(A) = L \).

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

Now, let \( w \in L(A) \) then \( val(A, w) = \bigvee \{val(r, w) / r \text{ is a run on } A \text{ is } r \} \)
\[ \Rightarrow \exists q_0, q \in Q \text{ such that } \delta(q_0, f(w), q) \in Q \text{ for some } q_0, q \in Q. \]

Theorem 3.3. Let \( A \) be a lattice language over an alphabet \( \Sigma \) then the reversal of \( A \) is defined as \( A^R = \{s \in \Sigma^* \mid \exists r \in A, r = l^R \} \).

Theorem 4.1. Let \( \Sigma \) be an given alphabet and \( L \) be an lattice, then the family of lattice regular expressions \( \mathcal{R} \) over \( \Sigma \) is defined by the following ways:
- \( \emptyset \) (empty set) \( \in \mathcal{R} \)
- \( \lambda \) (empty word) \( \in \mathcal{R} \)
- \( x \in \Sigma, x \in \mathcal{R} \)
- \( a \) (number of \( \mathcal{R} \)) \( \in \mathcal{R} \)
- \( L \) (length of \( \mathcal{R} \)) \( \in \mathcal{R} \)

Definition 4.2. Let \( L_1, L_2 \in \Sigma^* \) be two lattice languages over \( \Sigma \) then the reversal of \( L_1 \) is defined as \( L_1^R = \{s \in \Sigma^* \mid \exists r \in L_1, r = l^R \} \).

Definition 4.3. Let \( R \) be the lattice language represented by lattice regular expressions \( \mathcal{R} \) and is defined as follows:
- If \( \emptyset \) \( \in \mathcal{R} \)
- If \( \lambda \) \( \in \mathcal{R} \)
- For all \( x \in \Sigma, x \in \mathcal{R} \)
- For all \( a \in \mathcal{R} \), \( a \) \( \in \mathcal{R} \)
- For all \( l \in \mathcal{R} \), \( l \) \( \in \mathcal{R} \)

Theorem 4.4. Let \( L_1, L_2 \in \Sigma^* \) be two lattice languages over \( \Sigma \) then the reversal of \( L_1 \) is defined as \( L_1^R = \{s \in \Sigma^* \mid \exists r \in L_1, r = l^R \} \).

Theorem 4.5. Let \( L \) be a lattice language over an alphabet \( \Sigma \) then the reversal of \( L \) is defined by \( L^R = \{s \in \Sigma^* \mid \exists r \in L, r = l^R \} \).

Theorem 4.6. The class of lattice regular languages is closed under reversal by arbitrary set.

Theorem 4.7. Let \( L \) be a lattice language over an alphabet \( \Sigma \) then the reversal of \( L \) is defined by \( L^R = \{s \in \Sigma^* \mid \exists r \in L, r = l^R \} \).

Theorem 4.8. The class of lattice regular languages is closed under reversal by lattice regular grammar by showing that they generate the same lattice language.

4. LATTICE REGULAR EXPRESSIONS AND GRAMMAR

In this section, lattice regular expressions, lattice grammar, lattice regular grammar, lattice regular language, 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.
principle of mathematical induction on $k$, it can be reduced that $L_{\delta}(\lambda) = \tau$ if $i = j$ and $\bot$ otherwise, where $\tau, \bot \in \mathcal{L}$ represent top and bottom of the lattice $\mathcal{L}$ respectively.

$L_{\delta}(w) = \delta(q_1, w, q_2)$ for each $w \in \Sigma$ and for $m \in \mathbb{N}$,

$L_{\delta}(w_0w_1\ldots w_n) = \bigvee_{i_1 \leq k} \bigvee_{i_2 \leq k} \cdots \bigvee_{i_m \leq k} \delta(q_i, w_{i_1}, \ldots, q_j).

Thus for all $w \in \mathcal{L}$, $L_{\delta}(w) = \delta(w, q, q_j).

Therefore, $L = L_{\delta}$ for some $x \in \mathbb{R}$.

---

**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
- $S$ - set of lattice production rules (L-production rules) over $\mathcal{L}$ such that $S \in \mathcal{L}$
- $P$ - finite set of lattice production rules (L-production rules) over $\mathcal{L}$

Thus for all $w \in \mathcal{L}$, $L_{\delta}(w) = \delta(w, q, q_j)$.

Therefore, $L = L_{\delta}$ for some $x \in \mathbb{R}$.

---

**Definition 4.4.** If $\alpha \xrightarrow{t} \beta$, $l \in \mathcal{L}$ is a L-production rule then $\alpha \beta \xrightarrow{t} \beta \alpha$ means $\alpha \beta$ directly drives $\beta \alpha$, $\alpha, \beta, \gamma \in \mathcal{V}$.

If $A_1, A_2, \ldots, A_n \in \mathcal{L}$ and $A_1 \xrightarrow{t} A_2, A_2 \xrightarrow{t} A_3, \ldots, A_{n-1} \xrightarrow{t} A_n$ are $\mathcal{L}$-production rules of $\mathcal{G}$, then there exists $A_{n+1} \xrightarrow{t} A_{n+2}$.

The value of the word terminals $w$ derivable from $S \in \mathcal{S}$ is denoted by $val(A, w)$ and obtained by $val(A, w) = S(A) \wedge \bigwedge_{j=0}^{n} \{i_j\}$, where $i_j \in \mathcal{L}$ 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 L-production rules are of the form $A \xrightarrow{t} aB$ or $A \xrightarrow{t} a$, where $A, B \in \mathcal{V}$, $a \in T$, 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 at least one derivation chain from $A$ to $w$ and the value of the word $w$ is defined as

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

**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 $L(\mathcal{G})$.

**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 are either of the form:

$A \xrightarrow{t} aB$ or $A \xrightarrow{t} \lambda$, where $A, B \in \mathcal{V}$, $a \in T$, $\lambda$-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 $A \xrightarrow{t} a$ by $A \xrightarrow{t} aC$ and $C \xrightarrow{t} \lambda$, 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 $A \xrightarrow{t} w_1Bw_2$ or $A \xrightarrow{t} w$, where $A, B \in \mathcal{V}$, $l \in \mathcal{L}$ and $w_1, w_2, w \in T^*$.

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

**Definition 4.10.** A language $L \subseteq T^*$ is called a 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. $q_0 \xrightarrow{t} w$ in $P'$ iff $q_0 \xrightarrow{t} w$ in $P$, $S(q_0) = l$.
2. $q_0 \xrightarrow{t} wa$ in $P'$, for $S(q_0) = l$ iff $A \xrightarrow{t} w$ in $P$.
3. $A \xrightarrow{t} w$ and $A \xrightarrow{t} wq_0$ in $P'$ iff $q_0 \xrightarrow{t} Aw$ in $P$, $S(q_0) = l$.
4. $A \xrightarrow{t} wB$ in $P'$ iff $B \xrightarrow{t} Aw$ in $P$, where $A, q_0 \in V$ and $w \in T^*$.

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

Let $w \in L(\mathcal{G})$, then $w = w_1w_2\ldots w_{j-1}w_j$ for $j = 1, 2, \ldots, n$. Then, $\text{val}(\mathcal{G}, w) = \bigvee \{\text{val}(q_0, w) \wedge \bigwedge_{j=0}^{n} \{i_j\} \}$, in which each $i_j \in \mathcal{L}$.

If $q_0 \xrightarrow{t} w$ is a production in $P$ for some $l \in \mathcal{L}$ then $q_0 \xrightarrow{t} w$ is in $P'$ and $w \in L(\mathcal{G}')$. Otherwise, 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 $q_0 \xrightarrow{t} q_1w_1 \xrightarrow{t} q_2w_1w_2 \xrightarrow{t} \ldots \xrightarrow{t} w_1w_2\ldots w_n = w$.

Now, corresponding to the above derivation chain, $P'$ should have the following $\mathcal{L}$-production rules $q_0 \xrightarrow{t} w_1q_1 \xrightarrow{t} q_2w_1q_2 \xrightarrow{t} \ldots \xrightarrow{t} w_1w_2\ldots w_n = w$.

Therefore, $P'$ should have the following $\mathcal{L}$-production rules $q_0 \xrightarrow{t} w_1q_1 \xrightarrow{t} w_1w_2q_2 \xrightarrow{t} \ldots \xrightarrow{t} w_1w_2\ldots w_n = w$.

Thus, there is a derivation chain for $w \in \mathcal{G}'$ such that $q_0 \xrightarrow{t} w_1 \xrightarrow{t} w_1w_2w_3 \xrightarrow{t} \ldots \xrightarrow{t} w_1w_2…w_n = w$.

Therefore, $\text{val}(\mathcal{G}', w) = \bigvee \{\text{val}(q_0, w) \wedge \bigwedge_{j=0}^{n-1} \{i_j\} \}$, in which each $i_j \in \mathcal{L}$, i.e., $w \in L(\mathcal{G}')$. Similarly, the converse of the theorem can be proved.

**Theorem 4.3.** For every lattice grammar $\mathcal{G} = (\mathcal{L}, V, T, S, P)$ 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 $F : Q \rightarrow \mathcal{L}$ by $F(q_j) = i$ iff $q_j \xrightarrow{t} \lambda$ is a
lattice production in $P$ and $\delta \in \mathbb{L}^{Q \times \Sigma \times \mathbb{Q}}$ by $\delta(q,a,p) = t$ iff $q \xrightarrow{a} p$ in $P$. 

Now, $A = (\mathbb{L}, Q, \Sigma, \mathbb{D}, F)$ is a lattice automaton.

Let $x \in L(\mathbb{S})$ and $w = w_1 \cdots w_n, \forall w_j \in T$. Then $\text{val}(\mathbb{S}, w) = \bigvee \{ \text{val}(A, w) : \text{for all } A \in S \}$, where $\text{val}(A, w) = S(A) \wedge \bigwedge_{j=0}^{n-1} \{ l_j \}$, in which each $l_j \in \mathbb{L}$. 

Now, $\text{val}(A, w)$ implies that there exists $q_0, q_1, \ldots, q_n \in V$ and $l_1, l_2, \ldots, l_{n-1} \in \mathbb{L}$ such that $q_0 \xrightarrow{b} q_1 \xrightarrow{b} \cdots \xrightarrow{b} q_{n-1} \xrightarrow{b} w_n$, which gives the following transitions in $\mathbb{S}$ of $A$ 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, $\text{val}(A, w) = \bigvee \{ \text{val}(A, w) : \text{for all } A \in S \}$, where $\text{val}(A, w) = Q_0(q_0) \wedge \bigwedge_{j=0}^{n-1} \{ l_j \}$. 

Similarly, the converse part also can be proved.

Corollary 4.1. For every lattice grammar $\mathbb{S} = (\mathbb{L}, V, T, S, P)$ in normal form, there is a lattice regular grammar $\mathbb{S}_1$ such that $L(\mathbb{S}) = L(\mathbb{S}_1) \setminus \lambda$.

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

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

To prove the theorem, first construct a lattice regular grammar $\mathbb{S}_1 = (\mathbb{L}, V_1, T, S, P_1)$ such that $\mathbb{S}_1 \sim \mathbb{S}_1$ and then construct a lattice grammar in normal form $\mathbb{S}_2 = (\mathbb{L}, V', T, S, P')$ such that $\mathbb{S}_2 \sim \mathbb{S}_1$.

case(i):

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

For the $\mathbb{L}$-production rules of $P$ are in the form $A \xrightarrow{t} wB$ with $|w| > 1$ and $w = w_1w_2 \cdots w_n$ the $\mathbb{L}$-set production rules $P_1$ has following $\mathbb{L}$-production rules $A \xrightarrow{t} w_1 Z_1, Z_1 \xrightarrow{t} w_2 Z_2, \ldots, Z_{n-1} \xrightarrow{t} w_n B$, where $Z_1, Z_2, \ldots, Z_{n-1} \in V_1$ which are not in $V$.

For the $\mathbb{L}$-production rules of $P$ are in the form $A \xrightarrow{t} w_1w_2 \cdots w_m$ with $m \geq 2$ and $l \in \mathbb{L}$, the $\mathbb{L}$-productions rules $P_1$ has the following production rules $A \xrightarrow{t} w_1 Y_1, Y_1 \xrightarrow{t} a_2 Y_2, \ldots, Y_m \xrightarrow{t} \lambda$, 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 $\mathbb{L}$-production rules.

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

1. $A \xrightarrow{a} aB$
2. $A \xrightarrow{a}$
3. $A \xrightarrow{A, B} V \backslash \lambda \in \Sigma^*, \alpha \in \Sigma$

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

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

Now, if $|w| > 1$ and $w = a_1a_2 \cdots a_n, \forall a_i \in \Sigma$ then there exists $q_1, q_2, \ldots, q_{n-1} \in V'$ such that $q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} q_2 \cdots \xrightarrow{a_n} q_f = a_n = w$.

i.e., $q_0 \xrightarrow{t} x$ in $\mathbb{S}_1$.

Hence, $w \in L(\mathbb{S}_1)$. Consider a word $w \in L(\mathbb{S}_1)$ then $\text{val}(\mathbb{S}_1, w) = \bigvee \{ \text{val}(A, w) : \text{for all } A \in S \}$, where, $\text{val}(A, w) = Q_0(q_0) \wedge \bigwedge_{j=0}^{n-1} \{ l_j \}$ for some $q_0 \in V'$ such that $Q_0(q_0) \in L$. 

Therefore, there is a derivation chain of $w$ in $\mathbb{S}_1$ as given below:

$\text{val}(\mathbb{S}_1, w) = Q_0(q_0) \wedge \bigwedge_{j=0}^{n-1} \{ l_j \}$

Thus, the productions $\text{val}(\mathbb{S}_1, w) = Q_0(q_0) \wedge \bigwedge_{j=0}^{n-1} \{ l_j \}$, where $\lambda$ are in the $\mathbb{L}$-production rules $P_1$. 

Therefore, there is a derivation chain of $w$ in $\mathbb{S}_1$ as shown above:

Thus, there exists $g \in V_1$ such that $Q_0(q_0)$ and $\text{val}(A, w)$ in $\mathbb{S}_1$.

Therefore, $w \in L(\mathbb{S}_1)$, hence $\mathbb{S}_1 \sim \mathbb{S}_1$.

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 $\mathbb{S}_1$. 

Use an algorithm to eliminate all lattice rules of the form $A \xrightarrow{t} B$ as by given below:

Construct the set $U_i(A) = \{ A \}$, for $A \in V'$, and $U_{i+1} = U_i(A) \cup \{ B | B \xrightarrow{t} Z \in P_1 \}$ for some $Z \in U_i(A), \ell \in \mathbb{L}$. 

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

Now construct the lattice grammar $\mathbb{S}_2 = (V', \Sigma, S, P')$. $P'$ is defined as follows:

$A \xrightarrow{a} AB$ in $P'$ iff $\exists Z \in V \ni A \in U(Z)$ and $Z \xrightarrow{a} AB$ in $P_1$, where $r = \ell_1 \wedge \ell_2 \wedge \cdots \wedge \ell_{i+1}$. 

$A \xrightarrow{a} \lambda$ is in $P'$ iff $\exists Z \in V \ni A \in U(Z)$ and $Z \xrightarrow{a} \lambda$ in $P_1$, where $r = \ell_1 \wedge \ell_2 \wedge \cdots \wedge \ell_{i+1}$.

Clearly, the lattice grammar $\mathbb{S}_2 \sim (V', T, S, P')$ is in normal form. Therefore, it is simple to show that $\mathbb{S}_1 \sim \mathbb{S}_2$.

Corollary 4.2. Lattice right linear grammar is equivalent to lattice regular grammar except for $\lambda$.

Theorem 4.5. The following statements are equivalent except for $\lambda$.

1. lattice regular grammar
2. lattice left linear grammar
\[ \text{Lemma 4.1. (Pumping lemma) Let } \Sigma \text{ be a given alphabet. If } L \subseteq \Sigma^* \text{ be a lattice language over a lattice } \mathcal{L} \text{ then there exists a positive integer } m, \text{ where if for each word } w \in \Sigma^* \text{ with } |w| \geq m \text{ then we can decompose } w = xyz \text{ in such a way that } |y| \geq 0, |xy| \leq m \text{ and for each } i \geq 0, L(A)(xyz) = L(A)(xy^iz). \]

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

Let \( w = w_1w_2 \ldots w_{xy}w_{xz} \in \Sigma^* \) be a word in \( \Sigma^* \) with \( n \geq m \) and \( L(A)(w) = l \in \mathcal{L} \). For \( s = 1, 2, \ldots, n \), let \( \delta^s(q_0, w_1w_2 \ldots w_{xy}) = q_s \). There exists two integers \( j, k \), \( 0 \leq j < k \leq m \) such that \( q_j = q_k \). Since there are \( m+1 \) states in the sequence of states \( q_0, q_1, \ldots, q_m \). Thus \( x = w_1w_2 \ldots w_{xy}j, y = w_{xy+j+1} \ldots w_{xz} \) and \( y = w_{xy+j+1} \ldots w_{xz} \) then \( w = xyz \) with \( |y| \geq 1 \) and \( |xy| = k - m \) and from

\[ \begin{align*}
\delta^s(q_0, w_1w_2 \ldots w_{xy}) &= \delta^s(q_0, w_1w_2 \ldots w_{xy}) \cup \{ 0 \} \\
\delta^s(q_0, w_1w_2 \ldots w_{xy}) &= \delta^s(q_0, w_1w_2 \ldots w_{xy}) \cup \{ 0 \} \\
\delta^s(q_0, w_1w_2 \ldots w_{xy}) &= \delta^s(q_0, w_1w_2 \ldots w_{xy}) \cup \{ 0 \} \\
\delta^s(q_0, w_1w_2 \ldots w_{xy}) &= \delta^s(q_0, w_1w_2 \ldots w_{xy}) \cup \{ 0 \}
\end{align*} \]

and for any \( i \geq 0 \)

\[ \delta^i(q_0, w_1w_2 \ldots w_{xy}) = \delta^i(q_0, w_1w_2 \ldots w_{xy}) \cup \{ 0 \} \]

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

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 \( w = ab^n \) with lattice value \( l \) in which \( a, b \in \Sigma, n \in \mathbb{N} \) and \( l \in \mathcal{L} \).

Now, consider a string \( w = ab^n \) with \( n \geq 2 \). By pumping lemma, decompose \( w \) into \( xyz \) with \( |y| \geq 1 \) and \( |xy| \leq m \) for some \( i \), \( j \), \( k \), such that \( L(xyz) = L(\mathcal{L}(l)) \) for every \( i \geq 0 \). Therefore, \( x = a^i, y = a^j, z = b^n \) with \( r \geq 0 \), \( s \geq 0 \), \( t \geq 0 \) and \( r + s + t = n \), the condition implies that \( |xy| \leq m \). A contradiction arise because \( L(xyz) = L(\mathcal{L}(l)) \neq L(\mathcal{L}(l)) \) if \( l \neq \mathcal{L}(l) \). Hence, the lattice language \( L \) is not regular.

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

**Proof:** Given a lattice regular grammar \( \mathcal{L} = (\mathcal{L}, \Sigma, \Sigma, \Sigma, \Sigma, \Sigma) \), where \( \Sigma = V \cup \{ q \} \), \( \Sigma = T \), \( S = Q_0 \), \( L \) be any lattice, \( F = \{ q_f : F(q_f) = T \} \) and the \( \mathcal{L} \)-transition relation \( \delta \in L \times \Sigma \times Q \) is defined as \( \delta(q, a, p) = l \) if \( q \rightarrow ap \), where \( q, p \in V, a \in \Sigma \) and \( l \in \mathcal{L} \).

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

Now, \( \mathcal{L}(w) \) implies that there exists at least one derivation chain of the form \( q_0 \delta_0 \rightarrow w_1q_1 \delta_1 \rightarrow w_2q_2 \delta_2 \rightarrow \ldots \rightarrow w_nq_n \) where \( l_j \in L \) \( q_j \in Q \) for \( 0 \leq j \leq n \) and \( q_0 \delta_0 \rightarrow w_1q_1 \delta_1 \rightarrow w_2q_2 \delta_2 \rightarrow \ldots \rightarrow w_nq_n \) \( \delta_0 \rightarrow \delta_2 \rightarrow \ldots \rightarrow \delta_{n-1} \rightarrow \delta_n \) \( w_n \), \( q_{n-1} \delta_{n-1} = \delta_n \) \( q_n \) are all in the \( \mathcal{L} \)-transition relations.

That is, \( q_0 \delta_0 \rightarrow w_1q_1 \delta_1 \rightarrow w_2q_2 \delta_2 \rightarrow \ldots \rightarrow w_nq_n \)

Therefore, there is a sequence of \( \mathcal{L} \)-transition relations such that \( |\delta(q_0, w_1q_1) = l_0, \delta(q_1, w_2q_2) = l_1, \ldots, \delta(q_{n-1}, w_nq_n) = l_{n-1} \).

Thus, \( \mathcal{L}(w) \) a run on \( w \) of \( A \) is \( r \).

Therefore, \( L(\mathcal{L}) = L(\mathcal{L}) \).

The converse is similarly proved.

**Example 4.2.** Consider the lattice regular grammar \( \mathcal{L} = (\mathcal{L}, \Sigma, Q, Q_0, \delta, \mathcal{L}) \), where \( \mathcal{L} = (0, 1, 2, 3) \), \( V = \{ q_0, q_1, q_2, q_3 \}, \mathcal{L} = (a, b) \), \( S = (\{ q_0 \} \cap S(\{ q_0 \}) = 3 \) and the \( \mathcal{L} \)-production rules of \( P \) is defined as follows:

\[ P = \{ q_0 \rightarrow aq_2q_2 \rightarrow bq_2q_2 \rightarrow bq_3, q_3 \rightarrow aq_1 \rightarrow bq_1, q_1 \rightarrow bq_1 \} \]

Construct a lattice finite automaton \( A = (\mathcal{L}, \Sigma, Q, \delta, F) \), where \( \mathcal{L} = (0, 1, 2, 3) \), \( \Sigma = T \), \( Q = V \cup \{ q_f \}, \mathcal{L} = (a, b) \), \( S = (\{ q_0 \} \cap S(\{ q_0 \}) = 3 \) and the \( \mathcal{L} \)-transition relations are defined by \( P \) as follows:

\[ \delta(q_0, a, q_2) = 1, \delta(q_2, b, q_2) = 2, \delta(q_2, b, q_3) = 2, \delta(q_3, b, q_3) = 3, \delta(q_0, b, q_1) = 1, \delta(q_1, a, q_2) = 2, \delta(q_1, b, q_1) = 2 \]

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

**Proof:** Given a lattice automaton \( A = (\mathcal{L}, \Sigma, Q, Q_0, \delta, F) \), construct a lattice regular grammar \( \mathcal{L} = (\mathcal{L}, \Sigma, \mathcal{L}, \Sigma, \Sigma, \Sigma) \), where \( \mathcal{L} = (a, b) \), \( \mathcal{L} = (0, 1, 2, 3) \), \( \Sigma = T \), \( Q = V \cup \{ q_f \} \), \( \mathcal{L} = (a, b) \), \( S = (\{ q_0 \} \cap S(\{ q_0 \}) = 3 \) and \( \mathcal{L} \)-productions of \( P \) are defined as

\[ P = \{ q \rightarrow ap, a \in V, p \in Q, a \in \Sigma, l \in \mathcal{L} \} \]

\[ P = \{ q_0 \rightarrow a, q_f \rightarrow b \} \]
\[ q_0 \xrightarrow{b_1} w_1q_1 \xrightarrow{b_2} w_1w_2q_2 \xrightarrow{b_3} \ldots \xrightarrow{b_{n-1}} w_1w_2\ldots w_n = w \]

where, \( l_j \in L \) for \( 0 \leq j \leq n - 1 \) and \( q_0 \xrightarrow{b_1} w_1q_1, q_1 \xrightarrow{b_2} w_2q_2, \ldots, q_{n-2} \xrightarrow{b_{n-3}} w_{n-1}q_{n-1} \) and \( q_{n-1} \xrightarrow{b_{n-1}} w_n \) all are in \( P \) of \( \mathcal{S} \).

That is, \( q_0 \xrightarrow{w} l = \land \{ l_j : 0 \leq j \leq n - 1 \} \) and \( \text{val}(\mathcal{S}, w) = \lor \{ \text{val}(A, w) : \text{for all } A \in \mathcal{S} \} \) where \( \text{val}(A, w) = S(A) \land \land_{i=1}^{n-1} \land_{j=0}^1 \{ l_j \}, \) in which each \( l_j \in L \).

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

The converse is similarly proved. \( \blacksquare \)

5. CONCLUSION

Lattice automata and the lattice languages accepted by them 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


