Generalized Circular One-Way Jumping Finite Automata

Ujjwal Kumar Mishra, Kalpana Mahalingam, and Rama Raghavan

Department of Mathematics, Indian Institute of Technology Madras, Chennai-600036, India.
ma16d030@smail.iitm.ac.in
{kmahalingam,ramar}@iitm.ac.in

Abstract. A discontinuous model of computation called one-way jumping finite automata was defined by H. Chigahara et. al. This model was a restricted version of the model jumping finite automata. One-way jumping finite automata change their states after deleting a letter of an input and jump only in one direction. Allowing a state to delete a subword instead of a letter, we define a new model generalized circular one-way jumping finite automata. These automata work on an input in a circular manner. Similar to one-way jumping finite automata, generalized circular one-way jumping finite automata also jump only in one direction. We show that this newly defined model is powerful than one-way jumping finite automata. We define new variants(right and left) of the model generalized circular one-way jumping finite automata and compare them. We also compare the newly defined model with Chomsky hierarchy. Finally, we explore closure properties of the model.

Keywords: Jumping Finite Automata · One-Way Jumping Finite Automata · Generalized Linear One-Way Jumping Finite Automata · Generalized Circular One-Way Jumping Finite Automata.

1 Introduction

General jumping finite automata (GJFA), very first discontinuous model of computation, were introduced in [14] by Meduna et. al. These automata read and delete an input in a discontinuous manner. The automata can jump in either direction to delete the input. The model was extensively studied in [10,17,18]. Following [14], several jumping transition models have been introduced and studied in the last few years [5,15,16,19,13,11,7].

The model which is of our interest is one-way jumping finite automata (OWJFA), was defined in [5]. The model OWJFA was defined by giving a restriction on jumping behaviour of the model jumping finite automata (JFA), a restricted variant of GJFA. There are two variants of OWJFA: right one-way jumping finite automata (ROWJFA) and left one-way jumping finite automata (LOWJFA). ROWJFA starts processing an input string from the leftmost symbol of the input whereas LOWJFA starts processing an input string from the
rightmost symbol. The models can only jump over symbols which they cannot
process in its current state. Some properties of ROWJFA are given in [3]. The
decision problems of the model ROWJFA are discussed in [1]. Nondeterministic
variant of the model ROWJFA is defined and studied in [2].

OWJFA processes one symbol at a time, i.e., it go from one state to
another by deleting a symbol. One extension of this model that changes its states
by deleting a subword is generalized linear one-way jumping finite automata
(GLOWJFA), was defined in [15]. These automata work on a linear tape and
can process a subword in a state. In this paper, we introduce a new model
called generalized circular one-way jumping finite automata (GCOWJFA). This
model is also a generalization, in deleting an input, of the model OWJFA. Unlike
OWJFA, it can process a subword, i.e., GCOWJFA changes its states by deleting
a subword instead of a symbol. The difference between the model GLOWJFA
and GCOWJFA is that GLOWJFA work on a linear tape whereas GCOWJFA
work on a circular tape. Because of this difference both the models have distinct
computational power. We show that the generalization GCOWJFA increases the
power of OWJFA. Similar to OWJFA, we define two types of GCOWJFA: gen-
eralized right circular one-way jumping finite automata (GRCOWJFA) and gen-
eralized left circular one-way jumping finite automata (GLCOWJFA). The models
GRCOWJFA and GLCOWJFA delete an input in clockwise and anti-clockwise
direction, respectively. Both the models GRCOWJFA and GLCOWJFA can
jump over a part of the input if they cannot delete it.

In this paper, we compare the model GCOWJFA with the models OWJFA
and GLOWJFA. We also compare GRCOWJFA and GLCOWJFA. The language
classes of GRCOWJFA and GLCOWJFA are compared with the language classes
of Chomsky hierarchy. Closure properties of the language classes of GRCOWJFA
and GLCOWJFA are explored.

The paper is organised as follows: In Section 2 we give some basic notion and
notation. We also recall the definitions of ROWJFA, LOWJFA, GLOWJFA,
GLLOWJFA and give examples of these models, in this section. The models
GRCOWJFA and GLCOWJFA have been introduced in Section 3. The models
have been illustrated by examples. In Section 4 the models GLOWJFA and
GCOWJFA have been compared. We compare the models GRCOWJFA and
GLCOWJFA in Section 5. In Section 6 the language classes of the newly intro-
duced model GCOWJFA are compared with the language classes of Chomsky
hierarchy. Finally, we discuss closure properties of the newly introduced models,
in Section 7. We end the paper with few concluding remarks.

2 Preliminaries

In this section, we recall some basic notations and definitions. An alphabet set
is a finite non-empty set $\Sigma$. The elements of $\Sigma$ are called letters or symbols. A
word or string $w = a_1 a_2 \cdots a_n$ is a finite sequence of symbols, where $a_i \in \Sigma$ for
$1 \leq i \leq n$. The reverse of $w$ is obtained by writing the symbols of $w$ in reverse
order and denoted by $w^R$, hence $w^R = a_n \cdots a_2 a_1$. By $\Sigma^*$, we denote the set
of all words over the alphabet $\Sigma$ and by $\lambda$, the empty word. A language $L$ is a subset of $\Sigma^*$ and $L^C = \Sigma^* \setminus L$ denotes the complement of $L$. The symbol $\emptyset$ represents the empty language or empty set. For an arbitrary word $w \in \Sigma^*$, we denote its length or the number of letters in it by $|w|$. For a letter $a \in \Sigma$, $|w|_a$ denotes the number of occurrences of $a$ in $w$. Note that, $\Sigma^+ = \Sigma^* \setminus \{\lambda\}$ and $|\lambda| = 0$. For a finite set $A$, $|A|$ denotes the number of elements in $A$. A word $y \in \Sigma^*$ is a subword or substring of a word $w \in \Sigma^*$ if there exist words $x, z \in \Sigma^*$ such that $w = xyz$. If $w = uv$, then the words $u \in \Sigma^*$ and $v \in \Sigma^*$ are said to be a prefix and a suffix of $w$, respectively. Two sets $A$ and $B$ are comparable if $A \subseteq B$ or $B \subseteq A$. For two sets $A$ and $B$, if $A$ is a proper subset of $B$, then we use the notation $A \subset B$. For definitions of other basic language operations (like union, intersection etc.), the reader is referred to [12].

We now recall the definitions of right and left one-way jumping finite automata [5]. A right one-way jumping finite automaton (ROWJFA) is a quintuple $\mathcal{A} = (\Sigma, Q, q_0, F, R)$, where $\Sigma$ is an alphabet set, $Q$ is a finite set of states, $q_0$ is the starting state, $F \subseteq Q$ is a set of final states and $R \subseteq Q \times \Sigma \times Q$ is a set of rules, where for a state $p \in Q$ and a symbol $a \in \Sigma$, there is at most one $q \in Q$ such that $(p, a, q) \in R$. By a rule $(p, a, q) \in R$, we mean that the automaton goes to the state $q$ from the state $p$ after deleting the symbol ‘$a$’. For $p \in Q$, we set

$$\Sigma_p = \Sigma_{R_p} = \{b \in \Sigma : (p, b, q) \in R \text{ for some } q \in Q\}.$$ 

A configuration of the right one-way jumping automaton $\mathcal{A}$ is a string of $Q \Sigma^*$. The right one-way jumping relation, denoted as $\circlearrowright_\mathcal{A}$, over $Q \Sigma^*$ is defined as follows. Let $(p, a, q) \in R, x \in (\Sigma \setminus \Sigma_p)^*$ and $y \in \Sigma^*$. Then, the ROWJFA $\mathcal{A}$ makes a jump from the configuration $pxay$ to the configuration $qyx$, written as $pxay \circlearrowright_\mathcal{A} qyx$ or just $pxay \circ qyx$ if it is clear which ROWJFA is being referred. Let $\circlearrowright^+$ and $\circlearrowright^*$ denote the transitive and reflexive-transitive closure of $\circlearrowright$, respectively. The language accepted by the automaton $\mathcal{A}$ is

$$L_R(\mathcal{A}) = \{w \in \Sigma^* : q_0w \circlearrowright^* q_f \text{ for some } q_f \in F\}.$$

A left one-way jumping finite automaton (LOWJFA) is similar to that of a ROWJFA except that a configuration of LOWJFA is a string of $\Sigma^*Q$. By a rule $(q, a, p) \in R$, we mean that the automaton goes to the state $q$ from the state $p$ after deleting the symbol ‘$a$’. For $(q, a, p) \in R, x \in (\Sigma \setminus \Sigma_p)^*$ and $y \in \Sigma^*$ the LOWJFA makes a jump from the configuration $yaxp$ to the configuration $xyq$, written as $xyq \circlearrowleft_\mathcal{A} yaxp$ or just $xyq \circ yaxp$ if it is clear which LOWJFA is being referred. The language accepted by the automaton $\mathcal{A}$ is

$$L_L(\mathcal{A}) = \{w \in \Sigma^* : q_f \circlearrowleft wq_0 \text{ for some } q_f \in F\}.$$

**Example 1.** Consider the ROWJFA

![Diagram](https://example.com/diagram.png)

- **Example 2.** Consider the LOWJFA

![Diagram](https://example.com/diagram.png)
Hence, the language accepted by the automaton is 

\[ \{q_0, q_1, q_2\}, R\), the set of rules \( R \) is \( \{\{q_0, a, q_1\}, \{q_1, b, q_2\}, \{q_2, 0\}\}\).

Here, \( \Sigma_{q_0} = \{a\}, \Sigma_{q_1} = \{b, c\}, \Sigma_{q_2} = \emptyset \). Since \( \Sigma_{q_0} = \{a\}\), the automaton will jump over any word from \( (\Sigma \setminus \Sigma_{q_0})^* = \{b, c\}^* \) at \( q_0 \). Also, since \( \Sigma_{q_1} = \{b, c\}\), the automaton will jump over any word from \( (\Sigma \setminus \Sigma_{q_1})^* = \{a\}^* \) at \( q_1 \). It is clear from the rules that if a word is in the language, then it will have exactly one ‘a’ and one ‘c’. Moreover, the automaton has a loop labeled ‘b’ at \( q_1 \). Therefore, the language will have the words of the form \( b^l a^m c^b \) or \( b^l c^b a^m b^m \). We can divide these words in 8 groups \( \{l = m = n = 0\}, \{l = 0, m = 0, n \geq 1\}, \{l = 0, m \geq 1, n = 0\}, \{l = 0, m \geq 1, n \geq 1\}, \{l \geq 1, m = 0, n \geq 1\}, \{l \geq 1, m = 0, n \geq 1\}, \{l \geq 1, m \geq 1, n = 0\}, \{l \geq 1, m \geq 1, n \geq 1\}\).

Using case-by-case analysis, we conclude that the words \( a b^m c \), where \( m \geq 0 \) and \( b^m c \), where \( l, n \geq 0 \) are in the language. For example, consider a string \( b^l c a^m \), where \( l, n \geq 0 \). The sequence of transitions is

\[ q_0, b^l c a^m \cup q_1, b^m c \cup * q_1 c \cup q_2. \]

Hence, the language accepted by the automaton is

\[ L_R(A) = \{a b^m c : m \geq 0\} \cup \{b^l c a^m : l, n \geq 0\}. \]

Note 1. The language of the automaton, when it is considered as jumping finite automaton \([14]\) is \( \{b^l a^m c b^m : l, m, n \geq 0\} \cup \{b^l c b^m a^m : l, m, n \geq 0\} \).

A right one-way jumping finite automaton \([15]\) can be interpreted in two ways: one in which we think it working on linear tape and the other in which we think it working on a tape in a circular manner. When it is thought working on linear tape, in that case, the read head moves in one direction only and starts from the leftmost symbol of an input word. It moves from left to right (and possibly jumps over parts of the input) and upon reaching the end of the input word the automaton jumps to the leftmost symbol of the remaining concatenated word and starts deleting the word freshly from the last visited state. If a transition is defined for the current state and the next letter to be read, then the automaton deletes the letter. If not, but in the remaining input there are letters for which a transition is defined from the current state, the read head jumps to the nearest such letter to the right for its reading. The computation continues until all the letters are deleted or the automaton is stuck in a state in which it cannot delete any letter of the remaining word.

In the light of this interpretation, an extension of \( \text{ROWJFA} \) was given based on reading length of a word and generalized right linear one-way jumping finite automaton was defined in \([15]\). A generalized right linear one-way jumping finite automaton (\( \text{GRLOWJFA} \)) works on a linear tape and deletes an input word from left to right. The automaton starts deleting the input word with the reading head at the leftmost position of the input word. The automaton can jump over a part of the input word which it cannot process in its current state. It reads the nearest available subword of the word present on the input tape. If there is no transition available for current state, the automaton returns (head of the automaton returns) to the leftmost position of the current
input and continues its computation. The computation continues until all the letters are deleted or the automaton is stuck in a state in which it cannot delete any subword of the remaining word. The generalized left linear one-way jumping finite automaton (GROWJFA) work similar to GRLOWJFA. The read head of a GLLOWJFA moves from right to left. We recall the formal definitions of generalized right linear one-way jumping finite automaton and generalized left linear one-way jumping finite automaton from [13], given below.

**Definition 1.** A generalized right linear one-way jumping finite automaton (GROWJFA) is a tuple $A = (\Sigma, Q, q_0, F, R)$, where $\Sigma, Q, q_0, F$ are same as ROWJFA and $R \subseteq Q \times \Sigma^+ \times Q$ is a finite set of rules, where for a state $p \in Q$ and a word $w \in \Sigma^*$, there is at most one $q \in Q$ such that $(p, w, q) \in R$. By a rule $(p, w, q) \in R$, we mean that the automaton goes to the state $q$ from the state $p$ after deleting the word ‘$w$’. The automaton is deterministic in the sense that for a state $p$ and for a word $w \in \Sigma^+$ we have at most one $q \in Q$ such that $(p, w, q) \in R$. A configuration of the automaton $A$ is a string of $\Sigma^*Q\Sigma^*$. The generalized right linear one-way jumping relation, denoted as $\rightleftharpoons_A$, or just $\rightleftharpoons$ if it is clear which GRLOWJFA is being referred, over $\Sigma^*Q\Sigma^*$ is defined as follows: For a state $p \in Q$, set $\Sigma_p = \Sigma_{R,p} = \{w \in \Sigma^* : (p, w, q) \in R \text{ for some } q \in Q\}$.

1. Let $t, u, v \in \Sigma^*$ and $(p, x, q) \in R$. Then the GROWJFA $A$ makes a jump from the configuration $tpuxv$ to the configuration $tuqv$, written as

   $$tpuxv \rightleftharpoons tuqv$$

   if $u$ does not contain any word from $\Sigma_p$ as a subword, i.e., $u \neq u'wu''$, where $u', u'' \in \Sigma^*$, $w \in \Sigma_p$ and $u_2x_1 \neq x$, where $u_2, x_1 \in \Sigma^+$ and $u = u_1u_2$, $x = x_1x_2$ for some $u_1, x_2 \in \Sigma^*$. This is what we mean by nearest available subword.

2. Let $x \in \Sigma^+, y \in \Sigma^*$ and $y$ does not contain any word from $\Sigma_p$ as a subword, i.e., $y \neq y_1wy_2$, where $y_1, y_2 \in \Sigma^*$ and $w \in \Sigma_p$, then the GROWJFA $A$ makes a jump from the configuration $xpy$ to the configuration $pxy$, written as

   $$xpy \rightleftharpoons pxy.$$

The language accepted by the GROWJFA $A$ is

$$L_{\text{GRL}}(A) = \{w \in \Sigma^* : q_0w \rightleftharpoons q_f \text{ for some } q_f \in F\}.$$

Similar to that of GRLOWJFA, the generalized left linear one-way jumping finite automaton was defined.

**Definition 2.** A generalized left linear one-way jumping finite automaton denoted by GLLOWJFA is a tuple $A = (\Sigma, Q, q_0, F, R)$, where $\Sigma, Q, q_0, F$ are same as ROWJFA and $R \subseteq Q \times \Sigma^+ \times Q$ is a finite set of rules, where for a state $p \in Q$ and a word $w \in \Sigma^+$, there is at most one $q \in Q$ such that $(q, w, p) \in R$. By a rule $(q, w, p) \in R$, we mean that the automaton goes to the state $q$ from the state $p$ after deleting the word ‘$w$’. The automaton is deterministic in the sense that
for a state \( p \) and for a word \( w \in \Sigma^+ \) we have at most one \( q \in Q \) such that \((q, w, p) \in R\). A configuration of the automaton \( A \) is a string of \( \Sigma^*Q\Sigma^* \). The generalized left linear one-way jumping relation, denoted as \( \rightarrow \), or just \( \rightarrow \) if it is clear which \( GLLOWJFA \) is being referred, over \( \Sigma^*Q\Sigma^* \) is defined as follows: For a state \( p \in Q \), set \( \Sigma_p = \Sigma_{R,p} = \{ w \in \Sigma^+ : (q, w, p) \in R \text{ for some } q \in Q \} \).

1. Let \( t, u, v \in \Sigma^* \) and \((q, x, p) \in R\). Then, the \( GLLOWJFA \) \( A \) makes a jump from the configuration \( vupt \) to the configuration \( vqut \), written as
   \[ vqut \rightarrow vupt \]
   if \( u \) does not contain any word from \( \Sigma_p \) as a subword, i.e., \( u \neq u'wu'' \), where \( u', u'' \in \Sigma^* \), \( w \in \Sigma_p \) and \( x_2u_1 \neq x \), where \( u_1, x_2 \in \Sigma^+ \) and \( u = u_1u_2 \), \( x = x_1x_2 \) for some \( u_2, x_1 \in \Sigma^* \).

2. Let \( x \in \Sigma^+, y \in \Sigma^* \) and \( y \) does not contain any word from \( \Sigma_p \) as a subword, i.e., \( y \neq y_1y_2 \), where \( y_1, y_2 \in \Sigma^* \) and \( w \in \Sigma_p \), then the \( GLLOWJFA \) \( A \) makes a jump from the configuration \( yxp \) to the configuration \( yxp \), written as
   \[ yxp \rightarrow yxp \]
   if it makes a jump from the configuration \( yxp \) to the configuration \( vxp \).

The language accepted by the \( GLLOWJFA \) \( A \) is
\[
L_{GLL}(A) = \{ w \in \Sigma^* : q_f \rightarrow \rightarrow wq_0 \text{ for some } q_f \in F \}.
\]

We illustrate Definitions [1] and [2] with the following example.

**Example 2.** Consider the automaton \( A = (\{a, b, c\}, \{q_0, q_1\}, q_0, \{q_1\}, R) \), where \( R \) is depicted in the figure below.

![Diagram](#)

We first consider the automaton as a \( GRLOWJFA \). Then, the set of rules is \( R = \{(q_0, c, q_0), (q_0, ab, q_1)\} \). Hence, \( \Sigma_{q_0} = \{c, ab\} \) and \( \Sigma_{q_1} = \emptyset \). Since \( \Sigma_{q_0} = \{c, ab\} \), the automaton at \( q_0 \) can neither jump over a ‘c’ nor over ‘ab’. The automaton can delete arbitrary number of ‘c’s at state \( q_0 \) and can jump ‘a’ or ‘b’ at \( q_0 \). Consider a word \( c^lac^mbc^n \), where \( l, n \geq 0, m \geq 1 \). Then, \( q_0c^lac^mbc^n \rightarrow \rightarrow q_0c^lac^mbc^n \rightarrow aq_0c^{m-1}bc^n \rightarrow \rightarrow aq_0bc^n \rightarrow abq_0c^n \rightarrow q_0ab \rightarrow q_1 \) and hence,
\[
L_{GRL}(A) = \{ c^l | n \geq 0 \} \cup \{ c^lbc^n | l, n \geq 0, m \geq 1 \}.
\]

Now, consider the automaton as \( GLLOWJA \). In this case, the set of rules is \( R = \{(q_0, c, q_0), (q_1, ab, q_0)\} \). It can be shown, as explained above, the language accepted by the automaton \( GLLOWJA \) is
\[
L_{GLL}(A) = \{ abc^n | n \geq 0 \} \cup \{ c^lbc^n | l, n \geq 0, m \geq 1 \}.
\]
3 Generalized Circular One-Way Jumping Finite Automata

The right one-way jumping automata \cite{5} can also be thought/interpreted as working on a tape in a circular manner. In this case, a right one-way jumping finite automaton works on a circular tape and the read head moves in the clockwise direction. The read head starts from the first symbol of an input, i.e., read head is at $w_1$ of the input $w = w_1w_2 \cdots w_n$. The automaton can jump over parts of the input. If a transition is defined for a current state and the next letter to be read, then the automaton deletes the symbol. If not and in the remaining input there are letters for which a transition is defined from the current state, the read head jumps to the nearest such letter to delete it. The computation continues until all the letters are deleted or the automaton is stuck in a state in which it cannot delete any letter of the remaining word.

Keeping this interpretation in mind, we extend the definition of ROWJFA based on reading length of a word and define generalized right circular one-way jumping finite automata. A generalized right circular one-way jumping finite automaton works on a circular tape and deletes an input word in the clockwise direction. The automaton starts deleting the input word with the reading head at the first symbol, the symbol at which reading head is present, of the input, i.e., $w_1$ of $w = w_1w_2 \cdots w_n$. The automaton can jump over a part of the input word that it cannot delete. After deleting a subword of the word present on the input tape, the word is circularly permuted and the reading head goes to the first symbol just after the deleted subword. Suppose $uxv$ is present on the tape and the automaton deletes $x$, then the word is circularly permuted to get $vux$ and the reading head goes to the first symbol of $vux$. The computation continues until all the letters are deleted or the automaton is stuck in a state in which it cannot delete any subword of the remaining word present on the tape. The GLCOWJFA works similar to that of GRCOWJFA. The tape head of a GLCOWJFA moves in the anti-clockwise direction. The formal definitions of generalized right circular one-way jumping finite automaton and generalized left circular one-way jumping finite automaton are given below.

**Definition 3.** A generalized right circular one-way jumping finite automaton (GRCOWJFA) is a tuple $A = (\Sigma, Q, q_0, F, R)$, where $\Sigma, Q, q_0, F$ are same as ROWJFA and $R \subseteq Q \times \Sigma^+ \times Q$ is a finite set of rules, where for a state $p \in Q$ and a word $w \in \Sigma^+$, there is at most one $q \in Q$ such that $(p, w, q) \in R$. By a rule $(p, w, q) \in R$, we mean that the automaton goes to the state $q$ from the state $p$ after deleting the word ‘$w$’. The automaton is deterministic in the sense that for a state $p$ and for a word $w \in \Sigma^+$ we have at most one $q \in Q$ such that $(p, w, q) \in R$. A configuration of the automaton $A$ is a string of $Q\Sigma^*$. The generalized right circular one-way jumping relation, denoted as $\circlearrowright_A$, or just $\circlearrowright$ if it is clear which GRCOWJFA is being referred, over $Q\Sigma^*$ is defined as follows: For a state $p \in Q$, set $\Sigma_p = \Sigma_{R,p} = \{w \in \Sigma^+ : (p, w, q) \in R \text{ for some } q \in Q\}$. Let $u, v \in \Sigma^*$ and $(p, x, q) \in R$. Then, the GRCOWJFA $A$ makes a jump from
the configuration $puxv$ to the configuration $qv$,
written as

$$puxv \mathbin{\bowtie} qvu$$

if $u$ does not contain any word from $\Sigma_p$ as a subword, i.e., $u \neq u'w'u''$, where $w', w'' \in \Sigma^*$, $w \in \Sigma_p$ and $u_2x_1 \neq x$, where $u_2, x_1 \in \Sigma^+$ and $u = u_1u_2$, $x = x_1x_2$ for some $u_1, u_2 \in \Sigma^*$. The language accepted by the GRCOWJFA $A$ is

$$L_{\text{GRC}}(A) = \{w \in \Sigma^* : q_0w \mathbin{\bowtie} q_f^* \text{ for some } q_f \in F\}.$$
So any word in the language will be of the form \( clacmbcn \) or \( clbcma \). We can divide these words in 8 groups:

- \( \{ l = m = n = 0 \} \)
- \( \{ l = 0, m = 1, n = 0 \} \)
- \( \{ l = 0, m = 0, n = 1 \} \)
- \( \{ l = 1, m = 0, n = 0 \} \)
- \( \{ l = 1, m = 1, n = 0 \} \)
- \( \{ l = 1, m = 0, n = 1 \} \)
- \( \{ l = 1, m = 1, n = 1 \} \)
- \( \{ l = 1, m = 1, n = 1 \} \)

And using case-by-case analysis, we conclude that the language accepted by the automaton is

\[
L_{GRC}(A) = \{ clab | l \geq 0 \} \cup \{ clacmbcn | l \geq 0, m, n \geq 1 \} \cup \{ clbcma | l \geq 0, m \geq 1 \}.
\]

Now, consider the automaton as \( GLCOWJA \). In this case, the set of rules is \( R = \{(q_0, c, q_0)\} \). It can be shown, using the above technique, the language accepted by the automaton \( GLCOWJA \) is

\[
L_{GLC}(A) = \{ abc^n | n \geq 0 \} \cup \{ clacmbcn | l \geq 0, m, n \geq 1 \} \cup \{ bcma | m \geq 1, n \geq 0 \}.
\]

Note 2. Note that the language accepted by the automaton of Example 2 when it is considered as \( GJFA \), is \( L = \{ clacmbe^n : l, m, n \geq 0 \} \). The language accepted differs in \( GJFA, GRLOWJFA, GRCOWJFA, GLLOWJFA, GLCOWJFA \) because of different ways of processing the same input.

Now, we give and example that shows that the language classes \( GRCOWJ \) and \( GLCOWJ \) are not disjoint, where \( GRCOWJ \) and \( GLCOWJ \) are the language classes accepted by \( GRCOWJFA \) and \( GLCOWJFA \) respectively.

Example 4. Consider the automaton

```
start
q0
  ab
  \[*\]
q1
```

The automaton cannot make any move without deleting \( ab \) and at the state \( q_1 \) it can delete any number of \( c \)'s. Hence, the language accepted by the automaton is

\[
L_{GRC}(A) = L_{GLC}(A) = L_{GRL}(A) = L_{GLL}(A) = c^*abc^*.
\]

Hence, we conclude the following.

Note 3. \( GRCOWJ \cap GLCOWJ \cap GRLOWJ \cap GLLOWJ \neq \emptyset \).

From the Definition 5 of \( GRCOWJFA \), we observe the following:

**Lemma 1.** Let \( A = (\Sigma, Q, q_0, F, R) \) be a \( GRCOWJFA \). Let \( w_1, w_2 \in \Sigma^* \) and \( (p, x, q) \in R \). If the automaton goes from a configuration \( pw_1xw_2 \) to a configuration \( qw_2w_1 \), then the automaton also goes to the configuration \( qw_2w_1 \) from the configuration \( pw_1xw_2w_1 \), i.e., if \( pw_1xw_2 \circ qw_2w_1 \), then \( pxw_2w_1 \circ qw_2w_1 \).

A similar observation can also be made for the model \( GLCOWJFA \).
4 GCOWJFA and GLOWJFA

In this section, we compare the language class GCOWJ with the language classes ROWJ and GLOWJ, where ROWJ and GLOWJ are the language classes accepted by ROWJFA and GLOWJFA respectively. The comparison between language classes ROWJ and GLOWJ was done in [15] and it was proved that ROWJ ⊆ GLOWJ [15]. Here, we show that the language class ROWJ is properly contained in the language class GCOWJ. Similarly, it can be shown that LOWJ ⊆ GLCOWJ. We also show that GLOWJ is not a part of GCOWJ.

It is clear from the definitions of ROWJFA and GCOWJFA/GLOWJFA that when the rules of a GCOWJFA/GLOWJFA satisfies the condition: if \((p, w, q) \in R\), then \(|w| = 1\). Then, the GCOWJFA/GLOWJFA is nothing but ROWJFA. Hence, ROWJ ⊆ GCOWJFA ∩ GLOWJFA. Similarly, LOWJ ⊆ GLCOWJ ∩ GLLOWJFA. Now, we give a language that is in the class GCOWJ but not in the class ROWJ.

Example 5. ([15]) Consider the GCOWJFA \(A = \langle \{a, b\}, \{q_0, q_1, q_2, q_3, q_4\}, q_0, \{q_0, q_1, q_2, q_3\}, R \rangle\), where the set of rules is \(R = \{\langle q_0, a, q_1 \rangle, \langle q_1, b, q_2 \rangle, \langle q_2, a, q_3 \rangle, \langle q_3, b, q_2 \rangle, \langle q_0, a, q_4 \rangle, \langle q_4, a, q_4 \rangle\}\). The language accepted by the automaton is \(L_{GCOWJ}(A) = \{w \in \{a, b\}^* \mid |w|_a = |w|_b \text{ or } |w|_b = 0\}\).

It was proved in [3] that the language of Example 5 cannot be accepted by any ROWJFA. It was shown in [15] that the language is in the class GLOWJFA. Hence, by Example 5 and definitions of GCOWJFA and GLOWJFA, we have the following result.

Note 4. ROWJ ⊆ GCOWJFA ∩ GLOWJFA.

Similarly, it can be shown that LOWJ ⊆ GLCOWJ ∩ GLLOWJFA.

Now, we give a language that is in the language class GLOWJFA but not in the class GCOWJ.

Example 6. ([15]) Consider the GLOWJFA automaton \(A = \langle \{a, b\}, \{q_0\}, q_0, \{q_0\}, \{(q_0, a, q_0)\} \rangle\). The language accepted by the automaton is the Dyck language, \(D\). Hence, \(D \in GLOWJFA\).

Now, we prove that the Dyck language cannot be accepted by any GCOWJFA.

Lemma 2. There does not exist any GCOWJFA that accepts the Dyck language, i.e., \(D \notin GCOWJFA\).

Proof. If \(D \in GCOWJFA\), then there exists a GCOWJFA \(A = \langle \Sigma, Q, q_0, F, R \rangle\) such that \(L(A) = D\). Let \(l = |Q|\) and \(m = max\{|w| : (p, w, q) \in R\}\). Choose \(n > lm\). We know \(a^n b^n \in D = L(A)\). Then, there exists a sequence of transitions such that \(q_0 a^n b^n \overset{*}{\Rightarrow} q_f\), where \(q_f \in F\). In order to reach the final configuration \(q_f\) from the initial configuration \(q_0 a^n b^n\), there will be intermediate configurations of the form \(q'w_1a\) or \(q''w_2b\) or both, where \(q', q'' \in Q, w_1, w_2 \in \Sigma^*\).
Case 1. If there is an intermediate configuration of the form $q'w_1a$. Then, the automaton will make the following sequence of transitions $q_0a^n b^n \in \ast q'w_1a \in \ast q_f$. Then, by repeated application of Lemma 1, we have a word $w \in \Sigma^*$ such that $q_0ww_1a \in \ast q'w_1a \in \ast q_f$. Hence, $ww_1a \in L(A)$ but $ww_1a \notin D$ because no word of $D$ ends with ‘$a$’. Hence, there cannot be any intermediate configuration of the form $q'w_1a$.

Case 2. Since there cannot be any intermediate configuration of the form $q'w_1a$, all intermediate configurations have the form $q''w_2b$. In this case, the automaton will delete all $a$’s first and then $b$’s, starting from the initial configuration $q_0a^n b^n$, otherwise the automaton will go to a configuration of the form $q'w_1a$ because of Lemma 1. Because of the choice of $n$, there will be a loop consisting of only $a$’s to delete all $a$’s without deleting $b$’s, starting from the initial configuration. The same loop can be used multiple times to create a mismatch in the indices of ‘$a$’ and ‘$b$’. Hence, there will be a word of the form $a^n b^n$, $n_1 \neq n$, in $L(A)$ but $a^n b^n \notin D$ for $n_1 \neq n$.

From the above discussed cases, we conclude that $D \notin \text{GRCOWJ}$.

We conclude the following from Example 5 and Lemma 2.

**Proposition 1.** The language class $\text{GLLOWJ}$ is not contained in the language class $\text{GRCOWJ}$, i.e., $\text{GLLOWJ} \notin \text{GRCOWJ}$. Similarly, it can be proved that $\text{GLLOWJ} \notin \text{GLCOWJ}$.

## 5 GRCOWJFA and GLCOWJFA

In this section, the comparison between the language classes $\text{GRCOWJ}$ and $\text{GLCOWJ}$ has been done. We begin the section with a relationship that connects the languages of the class $\text{GRCOWJ}$ and the languages of the class $\text{GLCOWJ}$.

First we recall a similar result from [15].

**Lemma 3.** For a language $L_1 \in \text{GLLOWJ}$, there exists a language $L_2 \in \text{GLLOWJ}$ such that $L_1^R = L_2$ and vice versa.

Now, we show that for every language $L_1 \in \text{GRCOWJ}$ ($\text{GLCOWJ}$), there exists a language $L_2 \in \text{GRCOWJ}$ ($\text{GLCOWJ}$) resp. such that $L_1^R = L_2$.

**Proposition 2.** For a language $L_1 \in \text{GRCOWJ}$, there exists a language $L_2 \in \text{GLCOWJ}$ such that $L_1^R = L_2$ and vice versa.

**Proof.** Let $L_1 \in \text{GRCOWJ}$. Then, there exists a $\text{GRCOWJFA}$, say $A_1 = (\Sigma, Q, q_0, F, R)$, such that $L_1 = L(A_1)$. We construct a $\text{GLCOWJFA}$, say $A_2$, as: $A_2 = (\Sigma, Q, q_0, F, R^R)$, where $R^R = \{(q, w^R, p) \mid (p, w, q) \in R\}$. Let $L(A_2) = L_2$.

Claim, $L_1^R = L_2$. We have the following lemma:

**Lemma 4.** If $x, y \in \Sigma^*$, $u \in \Sigma^+$ and $p, q \in Q$, then $pxuy \subset A_1$ if and only if $x^R y^R q_{A_2} \subset y^R u^R x^R p$. 
Proof. Let $x, y \in \Sigma^*$, $u \in \Sigma^+$ and $p, q \in Q$. Now, $pxuy \cup_{A_t} qyx$ if and only if $(p, u, q) \in R$, $x$ does not contain any word from $\Sigma_{R,p}$ as a subword and $x''u' \neq u$, where $x''$ is a nonempty suffix of $x$ and $u'$ is a nonempty prefix of $u$ if and only if $(q, u^R, p) \in R^*$, $x'^R$ does not contain any word from $\Sigma_{R,p}$ as a subword and $u^{(2)}u'_1 \neq u^R$, where $x^{(1)}$ is a nonempty prefix of $x^R$ and $u^{(2)}$ is a nonempty suffix of $u^R$ if and only if $x^Ry^Rq_{A_2} \cup y^Ru^Rx^R p$. Hence, for $x, y \in \Sigma^*, u \in \Sigma^+$ and $p, q \in Q$, $pxuy \cup_{A_t} qyx$ if and only if $x^Ry^Rq_{A_2} \cup y^Ru^Rx^R p$.

Now, from Lemma 4, $w \in L_1^R$ if and only if $q_0 w^R \cup_{A_t} q_f$, where $q_f \in F$, if and only if $q_f \cup_{A_2} w q_0$ if and only if $w \in L_2$. Hence, $L_1^R = L_2$.

**Corollary 1.** If a language $L \in \text{GRCOWJ} \cap \text{GLCOWJ}$, then the reversal of the language $L^R \in \text{GRCOWJ} \cap \text{GLCOWJ}$. 

Now, we compare the language class GRCOWJ with the language class GLCOWJ. We give a language that is in the language class GRCOWJ. We show that the language is not in the class GRCOWJ. Hence, we conclude GLCOWJ is not contained in GRCOWJ. Similarly, it can be shown that GRCOWJ is not contained in GLCOWJ.

**Example 7.** Consider the GLCOWJFA $A = \{q_0, q_1, q_2\}$, $q_0, \Sigma_i$, $R$, where $R = \{(q_2, a, q_0), (q_2, b, q_0), (q_1, c, q_0), (q_1, ba, q_1)\}$. Here, $\Sigma_0 = \{a, b, c\}$, $\Sigma_1 = \{ba\}, \Sigma_2 = \emptyset$. We have the following observations regarding the language $L(A)$ of the automaton. Let $L(A) = L_c$.

- Note that all words of $L(A)$ will end with $c$. Suppose there is a word $w$ in $\Sigma^*$ that ends with $a$. Then, $w = w_1 a$, where $w_1 \in \Sigma^*$. Since $a \in \Sigma_0$, the automaton cannot jump over ‘$a$’ at $q_0$ and hence, after deleting ‘$a$’ the automaton will go to the state $q_2$, i.e., $w_1 q_2 \cup w_1 a q_0$. Hence, $w = w_1 a \notin L(A)$. Same argument holds true for words ending with ‘$b$’.
- Note that, $a^n b^n a^n b^n c \in L(A)$ for $n \geq 0$ since, $q_1 \cup \cdots \cup a^{n-1} b^{n-1} a^{n-1} b^{n-1} q_1 \cup a^{n-1} b^{n-1} a^{n-1} b^{n-1} q_1 \cup a^n b^n a^n b^n q_1 \cup a^n b^n a^n b^n c q_0$.
- If $w \in L(A)$, then $|w|^a = |w|^b$. This is because the deletion of $a$ or $b$ will happen using the rule $(q_1, ba, q_1)$ only.

Now, we prove that the language $L_c$ is not in the language class GRCOWJ.

**Lemma 5.** The language $L_c$ of Example 7 cannot be accepted by any GRCOWJFA.

Proof. Let $L_c$ be accepted by a GRCOWJFA $A = (\Sigma, Q, q_0, q_1, q_2)$, then $L(A) = L_c$ and let $l = |Q|, m = \{|w| : (p, w, q) \in R\}$ and $n > m$. From Example 7, we have $a^n b^n a^n b^n c \in L_c = L(A)$. Then, there would be a sequence of transitions such that $q_0 a^n b^n a^n b^n c \cup q_f$, where $q_f \in F$. The automaton can delete $a^i, a^i b^j, b^i, b^i a^i, b^i c$ or $c$, where $i, j \geq 1$, from the initial configuration $q_0 a^n b^n a^n b^n c$.

If it deletes $a^i b^j$, $i, j \geq 1$, from the initial configuration, then it follows the following sequence of transitions: $q_0 a^n b^n a^n b^n c \cup q_1 b^n a^n b^n c a^n i \cup q_f$, where $q_1 \in Q$. In this case, the automaton will also accept the word $a^i b^j a^n b^n c a^n i$.
because \( q_0 a^i b^j a^n b^m c \) is not in \( L(\mathcal{A}) \) but \( a^i b^j a^n b^m c \) is. Hence, it cannot delete \( a^i b^j \) from the initial configuration \( q_0 a^i b^j a^n b^m c \). Similarly, it can be shown that it cannot delete \( b^j, b^j a^i, b^j c \) or \( c, \) where \( i, j \geq 1 \), from the initial configuration.

The automaton cannot delete all \( a \)'s starting from the initial configuration \( q_0 a^i b^j a^n b^m c \) without deleting any ‘\( b \)’ or ‘\( c \)’ because in this case the automaton will have to loop and the loop has only \( a \)'s. Then, that loop can be used multiple times to create a mismatch in the indices of \( a \) and \( b \). Hence, there would be a word \( w \in L(\mathcal{A}) \) such that \( |w|_a \neq |w|_b \) but \( w \notin L_c \). Hence, it cannot delete all \( a \)'s starting from the initial configuration \( q_0 a^i b^j a^n b^m c \) without deleting a ‘\( b \)’ or ‘\( c \)’.

Suppose the automaton deletes \( a^i \) for \( 1 \leq t < n \) without using loop, then starting from the initial configuration \( q_0 a^i b^j a^n b^m c \), i.e., \( q_0 a^i b^j a^n b^m c \), \( a^i \) can only delete \( a \)'s and, hence, it will delete \( a^i b^j \), where \( 0 \leq i < (n - t), 1 \leq j \leq m, \) or \( b^j a^i \), where \( 1 \leq j < m, 1 \leq i < m, \) or \( b^j c \), where \( 0 \leq j < m \). Suppose it deletes \( a^i b^j \), where \( 0 \leq i < (n - t), 1 \leq j \leq m, \) then \( q_0 a^i b^j a^n b^m c \) cannot delete \( a^i b^j \) because \( q_0 a^i+b^n a^n b^m c a^n \) can be a word \( \mathcal{A} \) that \( q_0 a^i b^j a^n b^m c \) cannot delete without using loop in \( \mathcal{A} \) because \( q_0 a^i b^j a^n b^m c \) is not in \( L(\mathcal{A}) \).

Hence, we conclude \( L_c \notin \text{GRCOWJ} \).

From Example \([\text{4}]\) and Lemma \([\text{5}]\), we have that \( \text{GLCOWJ} \notin \text{GRCOWJ} \). Similarly, it can be shown that \( \text{GRCOWJ} \notin \text{GLCOWJ} \). Hence, we have the following result.

**Proposition 3.** The language classes \( \text{GRCOWJ} \) and \( \text{GLCOWJ} \) are incomparable.

### 6 GOWJFA and Chomsky hierarchy

This section compares the language class \( \text{GRCOWJ} \) with the language classes of Chomsky hierarchy. The class of regular languages, context free languages and context sensitive languages are denoted by \( \text{REG} \), \( \text{CF} \) and \( \text{CS} \) respectively. The relation \( \text{REG} \sqsubseteq \text{ROWJ} \) was proved in \([\text{5}]\) and in \([\text{13}]\), the inclusion \( \text{ROWJ} \sqsubseteq \text{GROWJ} \) was shown. Hence, \( \text{REG} \sqsubseteq \text{ROWJ} \sqsubseteq \text{GROWJ} \).

The class \( \text{CF} \) is incomparable with both the language classes \( \text{ROWJ} \) \([\text{3}]\) and \( \text{GROWJ} \) \([\text{15}]\). It was shown in \([\text{13}]\) that \( \text{ROWJ} \sqsubset \text{GROWJ} \sqsubseteq \text{CS} \). In this paper, we show that the language class \( \text{REG} \) is properly contained in the class \( \text{GRCOWJ} \), the language classes \( \text{CF} \) and \( \text{GRCOWJ} \) are incomparable and the class \( \text{GRCOWJ} \) is properly contained in the class \( \text{CS} \). Same results hold good for the class \( \text{GLCOWJ} \).

We recall some results from \([\text{5}]\).
Lemma 6. ROWJ properly includes REG, i.e., REG ⊂ ROWJ.

Lemma 7. CF and ROWJ are incomparable.

From Lemma 6 and Note 4 we have the following result.

Proposition 4. REG ⊂ GRCOWJ.

Proposition 5. The language classes CF and GRCOWJ are incomparable.

Proof. The Dyck language $D \in$ CF but from Lemma 2 we have, $D \not\in$ GRCOWJ. Hence, $CF \not\subseteq GRCOWJ$. From Lemma 7 we conclude $ROWJ \not\subseteq CF$ and hence, from Note 4 we have $GRCOWJ \not\subseteq CF$.

Proposition 6. The language class GRCOWJ is a proper subset of the class CS, i.e., GRCOWJ ⊂ CS.

Proof. Here, we give an idea on how the rules of a GRCOWJFA are simulated by a linear bounded automaton (LBA).

1. For a word $w = w_1 w_2 \cdots w_n$ from the input alphabet, the tape of the LBA contains $w_1 w_2 \cdots w_n$, the tape head is at $w_1$ and the LBA is in the starting state $q_0$, the starting state of GRCOWJFA. We assume, the tape is bounded from the left.

2. Suppose the LBA reaches in a state $p$ during its computation. Then, the LBA searches for the nearest subword, from $\Sigma_p$, of the word available on the tape on the right side of the tape head.

3. Suppose there is a nearest subword $x = x_1 x_2 \cdots x_i$ and the corresponding rule of GRCOWJFA is $(p, x, q)$. Then, the LBA marks it as $\ast \cdots \ast$ ($i$-times). We assume $\ast$ is not a member of the alphabet set. By nearest subword what we mean that suppose there is a word $uxv$ from the input alphabet on the tape on the right side of the tape head, then $u$ does not contain any word from $\Sigma_p$ as a subword and $u'x' \neq x$, where $u'$ is a nonempty suffix of $u$ and $x'$ is a nonempty prefix of $x$.

4. If there is a word from the alphabet left side of $\ast$’s, then the LBA cuts that word and pastes it right side of the word available right side of $\ast$’s, i.e., suppose after marking $x_1 x_2 \cdots x_i$ as $\ast \cdots \ast$, the tape contains $u \ast \cdots \ast v$. Then, the LBA cuts $u$ and pastes after $v$ and it goes to the state $q$ and the tape head is at the first symbol of $vu$.

5. Suppose there are no symbols from the alphabet set present on the tape and the LBA is in a state which is a final state of the GRCOWJFA, then the LBA accepts the input word.

Hence, we conclude that GRCOWJ ⊆ CS. But from Lemma 2 we have $D \not\in$ GRCOWJ. Hence, GRCOWJ ⊂ CS. Similarly, it can be shown that GLCOWJ ⊂ CS.
7 Closure Properties

In this section we discuss closure properties of the language classes GRCOWJ and GLCOWJ under set theoretic as well as language operations. The classes ROWJ [5] and GRLOWJ [15] are not closed under intersection, concatenation and reversal. Moreover, ROWJ is not closed under the operation Kleene star [5], union, complement and homomorphism [3]. In the same line, we show that the class GRCOWJ is not closed under intersection, concatenation, reversal and Kleene star. The same results hold true for the class GLCOWJ. We begin the section with an Example of a GRCOWJFA. The language of the GRCOWJFA will be used to prove non-closure of the class GRCOWJ under intersection, concatenation and Kleene star.

Example 8. Consider the GRCOWJFA $A = (\{a, b\}, \{q_0, q_1\}, q_0, \{q_0, q_1\}, R)$, where $R = \{(q_0, ab, q_1), (q_1, ba, q_0)\}$. Words $a^nb^n \in L(A)$ for all $n \geq 0$ because $q_0a^nb^n \triangleleft q_1b^{n-1}a^{n-1} \triangleleft q_0a^{n-2}b^{n-2} \triangleleft \cdots \triangleleft q_1/q_0$. If $n$ is even, then the automaton will be in the state $q_0$ and if $n$ is odd, then the automaton will be in the state $q_1$. Let $L(A) = L_{ab}$.

First we prove that the class GRCOWJ is not closed under Kleene star. We show that there exists a language $L$ in the class GRCOWJ but Kleene star $L*$ of $L$ is not in the class GRCOWJ.

**Proposition 7.** The language class GRCOWJ is not closed under Kleene star.

**Proof.** It can be shown, by adding some new states and rules in Example 8 that the language $cL_{ab} \in$ GRCOWJ. Let $L = (cL_{ab})^*$. We show that $L \notin$ GRCOWJ. If $L \in$ GRCOWJ, then there exists a GRCOWJFA $A = (\Sigma, Q, q_0, F, R)$ such that $L(A) = L$. Let $l = |Q|$, $m = max\{|w| : (p, w, q) \in R\}$ and $n > lm$. Clearly, $ca^nb^nc \in L(A)$. Hence, there exists a sequence of transitions such that $q_0ca^nb^nc \triangleleft^* q_f$, where $q_f \in F$. The automaton can delete $c$, $ca^i$, $a^ib^i$, $b^i$ or $b^i$ from the initial configuration $q_0ca^nb^nc$.

Case 1. If it deletes $a^i$, then the automaton makes the following sequence of transitions $q_0ca^nb^nc \triangleleft q_1a^{n-i}b^nc \triangleleft^* q_f$, where $q_1 \in Q$. Then, $a^{n-i}b^nc \in L(A)$ because of Lemma 1. But $a^nb^nc \notin L$. Hence, the automaton cannot delete $a^i$ from the initial configuration. Similarly, it can be shown that it cannot delete $aibi$, $b^i$ or $b^i$ from the initial configuration $q_0ca^nb^nc$.

Case 2. Hence, the automaton will have to delete ‘c’ or $ca^i$ from the initial configuration $q_0ca^nb^nc$. We deal with the case ‘c’ only, the case of $ca^i$ can be dealt similarly. If it deletes $c$ from the initial configuration $q_0ca^nb^nc$. Then, $q_0ca^nb^nc \triangleleft q_1a^nb^nc \triangleleft^* q_f$, where $q_1 \in Q$. It can delete $a^{i_1}$, $aibi$, $b^i$, $b^i$ or $c$, where $i_1, j_1 \geq 1$, from the configuration $q_1a^nb^nc$.

Sub-Case 2.1. If it deletes $aibi$ from the configuration $q_1a^nb^nc$, then $q_0ca^nb^nc \triangleleft q_1a^nb^nc \triangleleft q_2a^{n-j_1}b^i \triangleleft^* q_f$, where $q_3 \in Q$. Then, $caibi \notin L(A)$ because of Lemma 1. But $caibi \notin L$. Similarly, it cannot delete $b^i$ or $b^i$ from the configuration $q_1a^nb^nc$.
Sub-Case 2.2. If it deletes $c$ from the configuration $q_1a^nb^n$. Then, $q_0ca^n b^n \circ q_1a^nb^n \circ q_2a^n b^n \circ q_f$, where $q_2 \in Q$. In this case, $b^k \notin \Sigma_q$, for $1 \leq k \leq m$. Now, consider a word $cb^nca^n$. The word $cb^nca^n \in L(A)$ because $q_0cb^nca^n \circ q_1b^nca^n \circ q_2a^nb^n \circ q_f$. But $cb^nca^n \notin L$.

Sub-Case 2.3. Hence, the automaton will have to delete $a^{t_i}$ from the configuration $q_1a^nb^n$. The automaton cannot delete all $a$’s starting from the configuration $q_1a^nb^n$ without deleting $b$ or $c$ because in this case it will use a loop, because of the choice of $n$, consisting of only $a$’s to delete all $a$’s and the same loop can be used multiple times to create a difference in the indices of $a$ and $b$ and, hence there will be a word $w \in L(A)$ such that $|w|_a \neq |w|_b$ but, such $w \notin L$. Suppose it deletes $a^t$, $1 \leq t < n$, without using loop, starting from the configuration $q_1a^nb^n$ without deleting $b$ or $c$, then $q_0ca^n b^n \circ q_1a^nb^n \circ q_2a^n b^n \circ q_f$. From the configuration $q_2a^{n-t}b^n$, it can delete $a^t b^j$, where $0 \leq j < (n-t)$, $1 \leq j' \leq m$, or $b^j c$, where $0 \leq j' < m$. In this case, it will accept $ca^{t+j} b^j ca^{n-t-i}$ or $ca^t b^j ca^{n-t}$, depending on the deletion of $a^t b^j$ or $b^j c$ respectively. But neither $ca^{t+j} b^j ca^{n-t-i}$ nor $ca^t b^j ca^{n-t}$ are elements of $L$.

Hence, from above discussed cases, we conclude $cL_{ab} \in \text{GRCOWJ}$ but $(cL_{ab})^* \notin \text{GRCOWJ}$.

To prove other closure properties of the class \text{GRCOWJ}, we give a Lemma which gives a language that is not in the class \text{GRCOWJ}. The Lemma can be proved similar to Lemma \text{2}.

**Lemma 8.** There does not exist any \text{GRCOWJFA} that accepts the language \{a^n b^n \mid n \geq 0\}.

Now, we discuss other closure properties of the language class \text{GRCOWJ}.

**Proposition 8.** \text{GRCOWJ} is not closed under

1. intersection;
2. concatenation;
3. reversal;
4. Kleene star.

**Proof.** From Proposition \text{4} and Example \text{8} we have $a^* b^*, L_{ab} \in \text{GRCOWJ}$. But $a^* b^* \cap L_{ab} = \{a^nb^n \mid n \geq 0\} \notin \text{GRCOWJ}$, by Lemma \text{8}. Hence, \text{GRCOWJ} is not closed under intersection.

From Example \text{8} and Proposition \text{3} we have $L_{ab}, \{c\} \in \text{GRCOWJ}$. Similar to Lemma \text{5} we can prove $L_{ab}\{c\} \notin \text{GRCOWJ}$. Hence, \text{GRCOWJ} is not closed under concatenation.

It was shown in \text{5} that the languages $c\{w \in \{a, b\}^* \mid |w|_a = |w|_b\} \in \text{ROWJ}$. From Note \text{4} we have \text{ROWJ} $\subseteq \text{GRCOWJ}$ and hence $c\{w \in \{a, b\}^* \mid |w|_a = |w|_b\} \in \text{GRCOWJ}$. Similar to the proof given in Lemma \text{4} one can show that $\{w \in \{a, b\}^* \mid |w|_a = |w|_b\}c \notin \text{GRCOWJ}$. Hence, \text{GRCOWJ} is not closed under reversal.
8 Conclusion

In this paper, we have introduced a new extended version of the model one-way jumping finite automata (OWJFA) called generalized circular one-way jumping finite automata (GCOWJFA). By extended version, we mean that OWJFA change its states by deleting a letter of an input word whereas GCOWJFA change its states by deleting a subword of an input word. The newly introduced version GCOWJFA is different from one of the extended versions generalized linear one-way jumping finite automata (GLOWJFA) of OWJFA. Like OWJFA and GLOWJFA, GCOWJFA also has two variants called generalized right circular one-way jumping finite automata (GRCOWJFA) and generalized left circular one-way jumping finite automata (GLCOWJFA). We have shown that the newly defined version GCOWJFA is more powerful than OWJFA. We also show that its computational power is distinct from that of GLOWJFA. We have compared the computational power of the variants GRCOWJFA and GLCOWJFA. The language classes of GRCOWJFA and GLCOWJFA are compared with the language classes of Chomsky hierarchy. Finally, closure properties of the language classes of GRCOWJFA and GLCOWJFA have been investigated.

References

1. Beier, S., Holzer, M.: Decidability of Right One-Way Jumping Finite Automata. Proc. of LNCS 11088, pp. 109–120, Springer, Cham (2018)
2. Beier, S., Holzer, M.: Nondeterministic Right One-Way Jumping Finite Automata. Proc. of LNCS 11612, pp. 74–85, Springer, Cham (2019)
3. Beier, S., Holzer, M.: Properties of right one-way jumping finite automata. Theoretical Computer Science 798, 78–94 (2019)
4. Beier, S., Holzer, M., Kutrib, M.: Operational State Complexity and Decidability of Jumping Finite Automata. International Journal of Foundations of Computer Science 30(1), 5–27 (2019)
5. Chigahara, H., Fazekas, S. Z., Yamamura, A.: One-Way Jumping Finite Automata. International Journal of Foundations of Computer Science 27(3), 391–405 (2016)
6. Fazekas, S. Z., Hoshi, K., Yamamura, A.: Enhancement of Automata with Jumping Modes. Proc. of LNCS 11525, pp. 62–76, Springer, Cham (2019)
7. Fazekas, S. Z., Hoshi, K., Yamamura, A.: Two-Way Jumping Automata. Proc. of LNCS 12340, pp. 108–120, Springer, Cham (2020)
8. Fernau, H., Paramasivan, M., Schmid, M. L., Vorel, V.: Characterization and complexity results on jumping finite automata. Theoretical Computer Science 679, 31–52 (2017)
9. Kocman, R., Krivka, Z., Meduna, A.: On double-jumping finite automata and their closure properties. RAIRO - Theoretical Informatics and Applications 52(2-3-4), 185–199 (2018)
10. Kocman, R., Meduna, A.: On Parallel Versions of Jumping Finite Automata. Proceedings of the 2015 Federated Conference on Software Development and Object Technologies, pp. 142–149, Springer, Cham (2017)
11. Kocman, R., Nagy, B., Krivka, Z., Meduna, A.: A Jumping $5' \rightarrow 3'$ Watson-Crick Finite Automata Model. NCMA 2018, pp. 117–132, OCG, books@ocg.at, Vienna (2018)

12. Krithivasan, K., Rama, R.: Introduction to Formal Languages, Automata Theory and Computation. Pearson, India (2009)

13. Mahalingam, K., Mishra, U. K., Raghavan, R.: Watson-Crick Jumping Finite Automata. International Journal of Foundations of Computer Science 31(7), 891–913 (2020)

14. Meduna, A., Zemek, P.: Jumping Finite Automata. International Journal of Foundations of Computer Science 23(7), 1555–1578 (2012)

15. Mishra, U. K., Mahalingam, K., Raghavan, R.: Generalized Linear One-Way Jumping Finite Automata. arXiv (2021)

16. Vorel, V.: On basic properties of jumping finite automata. International Journal of Foundations of Computer Science 29(1), 1–15 (2018)

17. Vorel, V.: Two Results on Discontinuous Input Processing. Proc. of LNCS 9777, pp. 205–216, Springer, Cham (2016)