INVESTIGATION OF VARIOUS PROPERTIES OF JUMPING FINITE AUTOMATA

Thesis submitted in partial fulfilment of the requirements for the award of degree of

Master of Engineering

in

Computer Science and Engineering

Submitted By

SHWETA SHARMA

(801332026)

Under the supervision of:

Dr. Ajay Kumar

Assistant Professor

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT

THAPAR UNIVERSITY

PATIALA – 147004

JULY 2015

CERTIFICATE

I hereby certify that the work which is being presented in the thesis entitled, "*Investigation of various properties of jumping finite automata* ", in partial fulfilment of the requirements for the award of degree of Master of Engineering in *Computer Science and Engineering* submitted in Computer Science and Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of *Dr. Ajay Kumar* and refers other researcher's work which are duly listed in the reference section.

The matter presented in the thesis has not been submitted for the award of any other degree of this or any other University.

Signature:
(Shweta Sharma)

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

13/7/15
(Dr. Ajay Kumar)
Assistant Professor
Computer Science and Engineering Department
Thapar University

Countersigned by:

(Dr. Deepak Garg)
Head
Computer Science and Engineering Department
Thapar University
Patiala

(Dr. S. S. Bhatia)
Dean (Academic Affairs)
Thapar University
Patiala

Acknowledgement

he thesis turns into reality with support, help and motivation of many individuals. I xtend my sincere thanks to all of them. I express my deepest gratitude to my upervisor **Dr. Ajay Kumar**, Assistant Professor, Computer Science and Engineering epartment, Thapar University, for his guidance and support. No volume of words is nough to thank him for his motivation and concern that he added to carry out this esearch-oriented venture. His meticulous and priceless supervision during thesis ork inspired me in innumerable ways.

am highly grateful to **Dr. Deepak Garg**, Head of Department, CSED and **Dr. shutosh Mishra**, P.G Coordinator for their motivation and inspiration that helped to omplete the thesis work. Their support helped me to explore this thesis topic more in n organized manner and provided me with all required facilities during the need of he hour while completing this endeavor.

My thanks and appreciations go to staff members and my colleague who helped me out with their abilities.Most importantly, I am highly indebted to my parents whose love and constant support helped me to be calm even in oddest of times and showing me the right direction when there was no hope.

Shweta Sharma

(801332026)

Abstract

In earlier days, information processing was done using classical methods in a continuous manner. However, today the retrieval of information is through modernized methods in a discontinuous manner. Symbolizing the discontinuous processing of information correctly gave the conclusion to accept the formal automata in discontinuous manner; therefore a new category came in the field of automata called jumping finite automata.This automaton presented a new area; that needs to be explored as it was diverged from the previously existing models. In jumping finite automata, the string is read symbol by symbol and in left to right direction in a discontinuous manner. It demonstrates the jump from one location to other over the tape and can move in either direction; it performs the further computational steps from that position. If a symbol is read or processed in a computational step, then the automata cannot reread the same symbol again.

The presented thesis work investigates different properties of jumping finite automata and gives a valuable insight of this newly investigated topic. We described various operations on languages while some operation on words and examined results on their closure properties. We have also provided the results on decision problems of jumping finite automata. Based on the heuristics we had developed a java tool to demonstrate the working of jumping finite automata. This software has the capability of discontinuous reading; it recognizes whether given pattern is of GJFA or JFA and whether a string belongs to any of the patterns.

Table of Contents

List of Figures

List of Keywords

FA Finite Automata

DFA Deterministic Finite Automata

JFA Jumping Finite Automata

GJFA General Jumping Finite Automata

DCFL Deterministic Context Free Languages

VPL Visibly Push-down Languages

Chapter 1

Introduction

This chapter provides a basic description of automata and finite automata. It describes the operations on finite automata. At last in this chapter; the need for discontinuous automata is being discussed.

1.1 Finite automata

The branch of computer science and mathematics called the theory of computation deals with how effectively problems can be solved using an algorithm on the model of computation. An automaton is a mathematical study of computing machine or device and its computing capabilities. Practical application of automata is in the field of lexical analysis, verification of software and designing of the digital circuit.

A finite automaton consists of finite states. It is a computing device that acts as language acceptor partly because it produces output "yes" or "no" in response to a given input string. The response generated by automata depends on current input symbol and current state. A finite automaton announces the acceptance of string if it reads complete string and its current state is accepting state. For e.g. to accept the language $L_1 = \{x \in \{a,b\}^* \mid x \text{ ends with } aa\}$ finite automata operate in three states; to accept string with two consecutive aa .

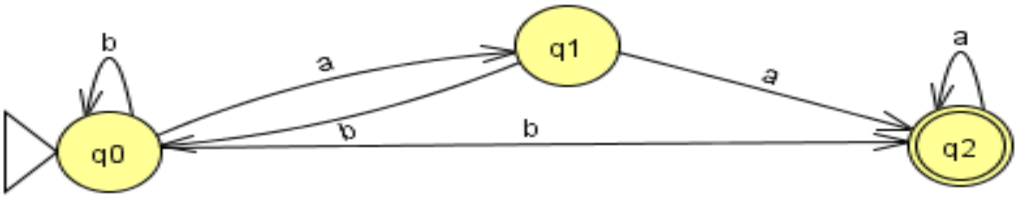


Fig. 1.1: A Finite automata accepting input string ending with consecutive aa .

In the above Fig 1.1, the input symbol b on state q_0 does not represent any move or progress, and this causes finite automata to remain in the state q_0 whereas input symbol a shows movement to state q_1 .In q_1 the state, input symbol makes a

backward move taking back to state q_0 while the input symbol a makes progress to state q_2 showing acceptance to string present in the language. Therefore, we can simply trace the arrows corresponding to input symbol to diagnose the string of given language.It is compulsory to indicate final or accepting state in finite automata; because it defines which particular input symbol sequence is accepted by finite automata. Following example shows that it accepts the string "start".

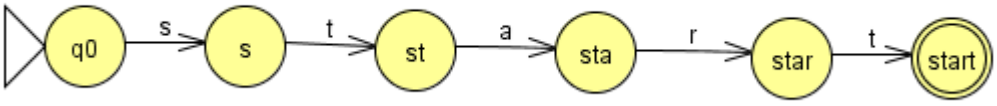


Fig. 1.2: A finite automaton for recognizing input string "start".

Therefore, it is justified to say that it is an informally state machine that can comprehensively capture all possible states and transitions that the machine can make while responding to input symbol sequence. Finite automata recognize regular languages. The deterministic nature of finite automata can make only one transition for a given input symbol whereas in case of non-deterministic behavior there can be multiple transitions for a given input symbol. Regular language that is recognized by non-deterministic finite automata can also be described by its corresponding deterministic nature [26].

The regular expression denotes the same pattern of strings that is described by finite automata which means every formal language that can be defined by finite automata can be equally defined by its corresponding regular expression.Language is defined on the finite non-empty set of symbols called alphabet. They are denoted by $\sum$ .Language is defined as a subset of $\sum *$. φ Denotes null language and $\in$ denotes empty string respectively.

1.2 Operations on languages

Following are some operations that can be applied on languages [27].

- **Union:** Let L_1 and L_2 be two languages. The union of both languages contains the set of strings that are present in L_1 or in L_2 or in both. For

example If $L_1 = \{a, b\}$ and $L_2 = \{a, ab, ba\}$. Therefore Union of L_1 and L_2 $\{a, b, ab, ba\}$

- **Concatenation:** This operation is performed by concatenating string present in L_1 with every string present in L_2 . For example let $L_1 = \{a, b\}$ and $L_2 = \{a, ab, ba\}$. After applying concatenation operation resulting language becomes $L_1 . L_2 = \{aa, ba, aab, bab, aba, bba\}$
- **Intersection:** Let L_1 and L_2 be two languages. Applying intersection on both languages contains the strings that are common in both languages. For example if $L_1 = \{a, b\}$ and $L_2 = \{a, ab, ba\}$ Therefore intersection of L_1 and $L_2 = \{a\}$.
- **Shuffle:** Let $w \in L_1$ and $v \in L_2$ be two words. If $w = x_1x_2..x_n$ and $v = y_1y_2..y_n$ .The word formed by shuffling these two words is $shuffle(w, v) = \{x_1y_1x_2y_2....x_ny_n\}$
- **Reverse:** Let L_1 be the language. After applying reverse operation, all the words present in a language get reversed. In finite automata, it can be applied by interchanging initial and final states and by reversing the transition direction. For example $L_1 = \{aab, abb, aab\}$ applying reverse operation, the language becomes reverse (L_1) $= \{baa, bba, baa\}$ [27].
- **Difference:** Let L_1 and L_2 be two languages. Applying difference on both languages contains the strings that are present only in a first language not in a second language. For example If $L_1 = \{a, b\}$ and $L_2 = \{a, ab, ba\}$ Therefore difference of L_1 and $L_2 = \{b\}$.

1.3 Need of Discontinuous processing

The real time application requires discontinuous processing of information to meet the limitation of unregulated mechanism in modern computation scenario. Earlier it was virtually immaterial therefore it remained neglected for information processing but today it has become the need of computation and it play a vital role in the field of automation. Modern information methods are based on processing of information in the discontinuous way and also consider dynamic behavior. Their formalized models are though complexin comparison to the regulated mechanism, but they can solve vide variety of research area problems [7]. They provide beneficial insight for typical and

less inferred problems. These models dynamic behavior is based on their desired input symbol and current state which helps to analyze, model and control processing of information in an efficient manner. Characterizing discontinuous information processing induced the idea to accept already existing classical formalized model with a feature of discontinuous processing. Moreover, it has already provided a framework in many areas [4].

Chapter 2

Discontinuous Automata

In this chapter, discontinuous automata like nested word automata, deep pushdown automata, input revolving automata, etc. are being discussed. This chapter describes the functionality of already existing discontinuous automata.

2.1Nested word automata

Finite and infinite words contain a hierarchical nested structure that was introduced by Madhusudan and Alur known as Nested words. They provide with the intuitive idea to model programs with recursive procedure calls, by matching the pair of the calls and returns with the help of the nesting relation while internal operations correspond to other elements. Such kind of structure containing dual behavior is represented by the nested word. This dual nature can be represented by HTML and XML documents, linguistic annotated data and structured programs [17].Nested word automata are similar to finite automata, and it reads input according to a linear sequence from left to right.On a call operation the propagation is along the outgoing nested - linear edge and on return operation; nested-linear incoming edge determines the new state. Nested word automata and nested words have property to share logical characterization with unnested words. Their finite automata model shares same expressive power as MSO monadic second order. Natural and Robust nature of decision procedure, logical characterization and closure properties can be obtained from it [1].

Modal μ calculus is used to describe regularity as an alternating way for infinite and finite unnested words. The motivating areas for nested words are software verification. The sequential computation control flow in programming languages is modeled through pushdown automata using recursive, nested, implementing methods and procedure calls [17].

For a given language of nested word over some alphabet, another language L' can be obtained by linear encoding of these nested words. The new language consists of symbols that are tagged according to position.Therefore L' is context free language if it obtained from L which is a regular language. Push-down automata that accept L' has a special kind of structure that is if it reads a call operation then the automation

has to push one symbol on stack and if it is reading a return symbol then it must apply pop operation whereas in case of reading an internal symbol it only needs to update its control state [8]. Such automata are called as visibly pushdown automata and the languages accepted by such automata are known as visibly pushdown languages.

Since these automata can be deterministic therefore visibly pushdown, languages can correspond to DCFL (deterministic context-free languages) subclass. VPLs generalize the balanced languages, bracketed languages, parenthesis languages and they also correspond to have better closure properties than parenthesis languages [1].For checking properties; that are not expressed in existing specification logics like pre-post conditions, stack inspection, etc. can be modeled through regular languages of nested words. Push down automata does two purpose that is to discover the hierarchy -matching corresponding to hierarchy and the second purpose is processing the matching.

2.1.1 Relation to ordered trees

Data with dual structure (linear and hierarchical) is modeled using either binary or ordered unranked trees and can be queried with the help of tree automata. In case of ordered tree; nodes that share the same parent are ordered linearly and to define the ordering of nodes classical tree traversal is used [17].

For document processing, the nested word has more advantages as compared to ordered trees. Suffix, Prefix, and concatenation are some of word operations that are used for document processing that do not perform analogous tree operations.

Secondly tree automata also can express the constraints along a hierarchical path on the labels and also along siblings in left to right order but they face problem to capture the constraints of the global linear order. NWA gives an illusion as that of tree automata. Top –down, Bottom –up tree automata are its special cases. These outputs define that a query can be clearly transformed in nested word view with some benefits in complexity.

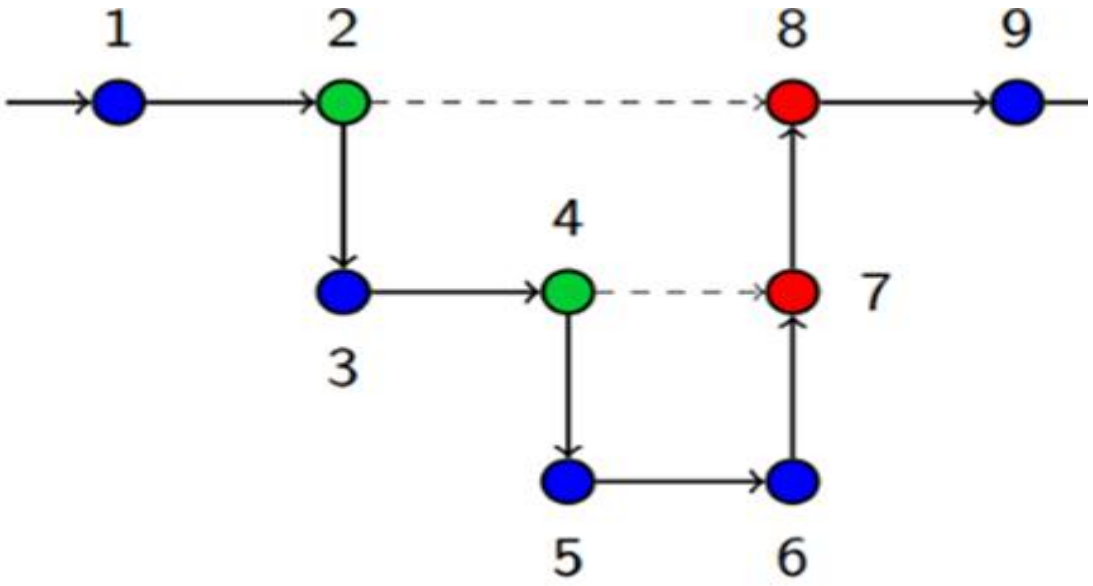


Fig. 2.1: Nested word structure of $i'c'i'c'i'i'r'r'i'$ [1]

In the above figure $\{1,3,5,6\}$ represents the internal operations, $\{2,4\}$ represents the call operation and $\{7,8\}$ represents corresponding return positions.

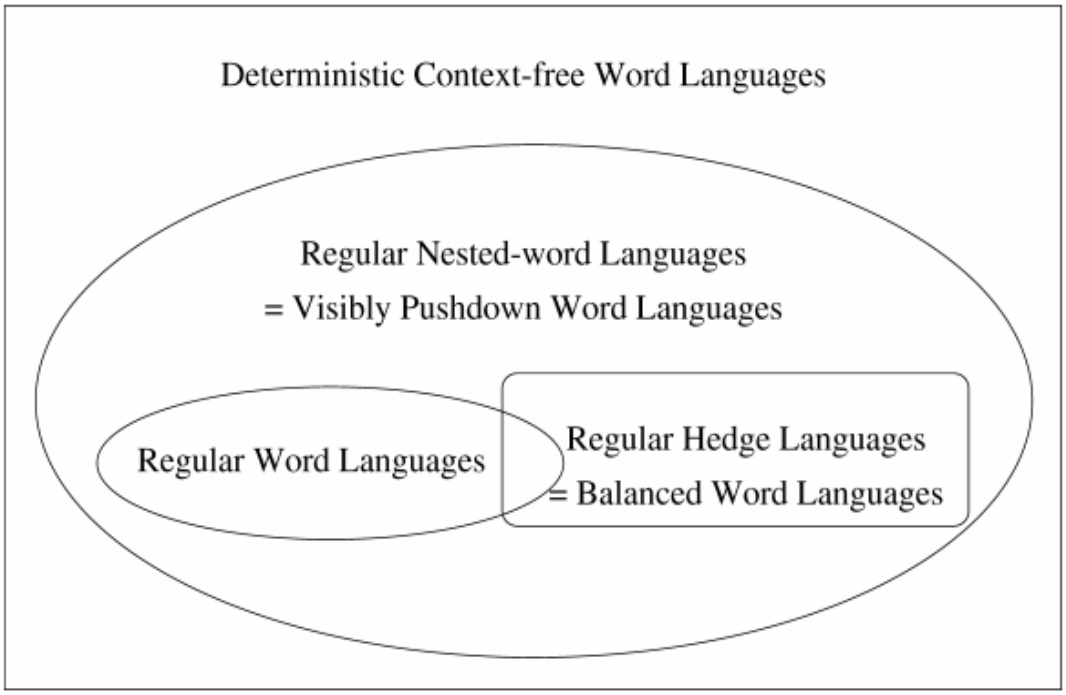


Fig. 2.2: Relationship among language classes [1]

2.2 Input revolving automata

In recent years, finite state automata have been extended, and its different variants came in the picture. With this, succeeding came one variant called input revolving automata. This automation also resembles to forgetting automata. What led to differences among these two is the inspiration, technical details, and many more results. The similarities between them; is how symbols are processed, and it depends upon the condition or present state of automata. The other similarity that exists is that they both have the power to read a symbol again [2].

Their capability to shift input and non-deterministic behavior make them competent to other discontinuous automata. The three operational functionality of this automaton is left revolving, right revolving or bidirectional revolving. In left revolving last symbol comes in the first position, in right revolving first symbol goes in the end, and bi-revolving extreme position symbols are swapped. This automaton is very practical in manifesting non-determinism over determinism conduct.

2.3 Quantum Automata

They are formed by allowing the matrices to have complex entries with a necessary condition of each of matrices to be unitary [9]. Unitary nature means the sum of the square of all the entries of the matrix comes out to be 1 and product of any two columns should be 0.

2.4 Deep pushdown automata

A deep pushdown automaton is a stereotype of existing push down automata. This generalization allows making deeper expansion; it replaces the nth topmost symbol of the stack with a string [6].

Chapter 3

Problem Statement

This chapter defines the motivation behind the thesis. It also contains problem statement with the steps followed to solve the problem.

3.1 Motivation

The real-time application requires discontinuous processing of information to meet the limitation of unregulated mechanism in modern computation scenario. Characterizing discontinuous information processing induced the idea to accept already existing classical formalized model with the feature of discontinuous processing. Their formalized models are though complex in comparison to the regulated mechanism, but they can solve a wide variety of research area problems. They provide beneficial insight for typical and less inferred problems [7].

Alexander and Peter [5] presented a new investigation area in the field of automation called jumping finite automata. This automaton is modified the version of finite classical automata where input symbol can be read discontinuously, which means after reading an input symbol; it can jump in either direction and can begin its further computation from there. This paper presents some of the results on closure properties and decidability.

The presented thesis work provides an extension to closure properties of some operation. The analysis of decision problems like universality, inclusion and equivalence under general jumping finite automata and jumping finite automata is described in the thesis. A tool in Java is made to test whether given grammar is GJFA or JFA and whether entered string is accepted by the grammar.

3.2 Problem Statement

The problem is to check whether given operator is closed or not for GJFA and JFA. With the help of different language combination, our purpose was to find at least one such example where the operator does not satisfies the properties of general jumping finite automata or jumping finite automata. So we had carried out an operation on all permutations of languages to find such a contradictory example to prove operator

closure property as not closed. Secondly we worked to find an algorithm to prove whether universality, equivalence and inclusion are decidable or not.

3.2.1 Objectives and Methodology

Following are the general objectives of the presented thesis work

- Analysis and comparisons of different discontinuous automata.
- Various properties of GJFA and JFA are investigated.
- A tool is designed that represent the working of GJFA and JFA

Analysis and comparison were carried out by studying the different type of discontinuous automata available in the literature. On the basis of theorems of GJFA and JFA, closure properties are examined. Together with this; we worked to find an algorithm to solve the decision problem. Based on the heuristics we had also developed a Java tool to demonstrate the working of jumping finite automata. This software has the capability of discontinuous reading; it recognizes whether given pattern is of GJFA or JFA and whether a string belongs to any of the patterns.

Chapter 4

Definition and Properties

Here in this chapter we first discuss the definition and later we describe properties of jumping finite automata.

4.1Preliminaries

Definition 1: GJFA (General jumping finite automata) is defined as $M = (Q, \Sigma, \delta, q_0, F)$

Q *is finite set of state*

Σ *is input alphabet*

$\Sigma \cap Q = \phi, \delta \subseteq Q \times \Sigma^* \times Q$

$q0$ *is inital state*

F *is final state*

Any string in $\Sigma^* Q \Sigma^*$ provideconfiguration of M .Let $m, n, m', n' \in \Sigma^*, mn = m'n'$ and $ry \to t$ then M can make a jump from $mryn \to m'tn'$ .The language accepted by $L(M)$ becomes [5].

$$L(M) = \{u'v' \mid u'v' \in \Sigma^*, u'sv' \xrightarrow{*} f', f' \in F \}$$

Let $w \in \Sigma^*$.By analogy, we can say that w is accepted by M if $w \in L(M)$ and w is rejected if it belongs to $w \in \Sigma^* - L(M)$ .Two GJFA M and M' are said to be equivalent if $L(M) = L(M')$

Definition 2: $M = (Q, \Sigma, \delta, q_0, F)$ be GJFA. M is $\in$-free GJFA if $ry \to t \in \delta$ which [5] denotes $|y| \geq 1$. M Has a degree as n, where $|n| \geq 0$ if $ry \to t \in \delta$ signifies that $|y| \leq n$.

Definition 3: M is JFA (jumping finite automata) if it has degree 1[5].

Definition 4: Let $M = (Q, \Sigma, \delta, q_0, F)$ be JFA. By analogy to GJFA, M is $\in$-free jumping finite automata if $ry \to t \in \delta$ which signifies M is deterministic jumping finite automata and complete jumping finite automata for short (DJFA and CJFA) if:

- Deterministic jumping finite automata[5]
    1. If it is $\in$-free JFA
    2. For each $r \in q$ and for each $y \in \sum$,only one $t \in q$
- Complete jumping finite automata[5]
    1. If it is DJFA
    2. For each $r \in q$ and for each $y \in \sum$ there is precisely one $t \in q$ such that $ry \to t \in \delta$.

Definition 5: Let $M = (Q, \Sigma, \delta, q_0, F)$ be GJFA.

Graphical representation of the transitional graph of M is denoted by $\Delta(M)$ and $ry \to t \in \delta$ where r and t represents the node and the edge between is labeled by y .

Consequently without loss of generality labeled directed multi-graph state $q' \in q$ is terminating [5] if final state reachable from s . If there exists path from r to $t \in q$ using $t1, t2, ...tn$ for $n >= 2$ where $t_i\ y_i = t_{i+1} \in \delta$ for all $i = 1,2,3..n-1$ therefore $r\ y1\ y2\ y3..... yn \to t$

Definition 6: Consider the JFA

$$M = (\{s, r, t\}, \{a, b, c\}, R, s, \{f\})$$

Where R define rules as $sa \to r, rb \to t$ and $tc \to s$ .It begins reading from s , M first reads a followed by b and then it reads c and it enters again into the start state. The occurrence of the symbol a, b, c can be present anywhere over the given input string. Therefore, the accepted language becomes [5] $L(M) = \{w \in \Sigma^* \mid w_a \mid = \mid w_b \mid = \mid w_c \mid\}$

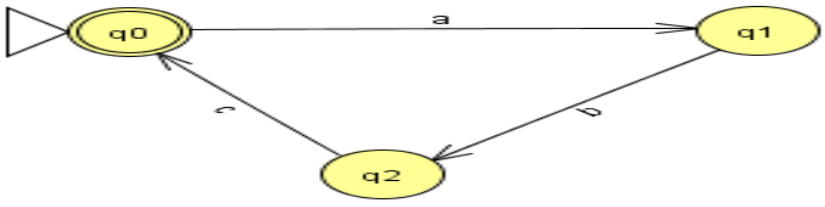


Fig. 4.1: Jumping finite automata

Definition 7: Consider the GJFA

$$M = (\{s,t,f\},\{a,b,c\},R,s,\{f\})$$

Where R denote rules as $sabc \rightarrow r, rcc \rightarrow f$, $fa \rightarrow f$ .It starts reading from s, then M first read string abc, which can appear anywhere in the given input string. Then it begins searching for cc and it jumps to the location in either direction to read it [5]. Finally, it reads a .Therefore rules describe accepted language as $L(M) = (a)*(abc)(cc)(a)*$.

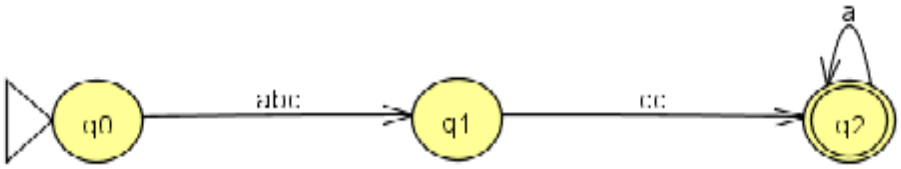


Fig. 4.2: General jumping finite automata

Definition 8: Consider the GJFA

$$M = (\{s,t,f\},\{a,b\},R,s,\{f\})$$

Where R defined by $sab \rightarrow f$, $fa \rightarrow f$, $fb \rightarrow f$ be GJFA.It begins to read from s, M first read the string ab, which appear anywhere in the input string. Then it randomly read the symbols a,b .Therefore, these rules describe the accepted language [5] $L(M) = (a+b)*ab(a+b)*$

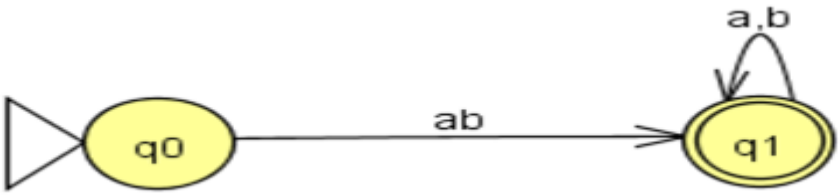


Fig. 4.3: General jumping finite automata

Lemma[5]: Let $M = (Q1,\sum,\delta,s1,F)$ be GJFA and $M' = (Q2,\sum,\delta,s2,F2)$ be $\in$-free GJFA, this lemma can be proved with the help of standard conversion of finite automata into $\in$-free finite automata. Therefore $L(M) = L(M')$

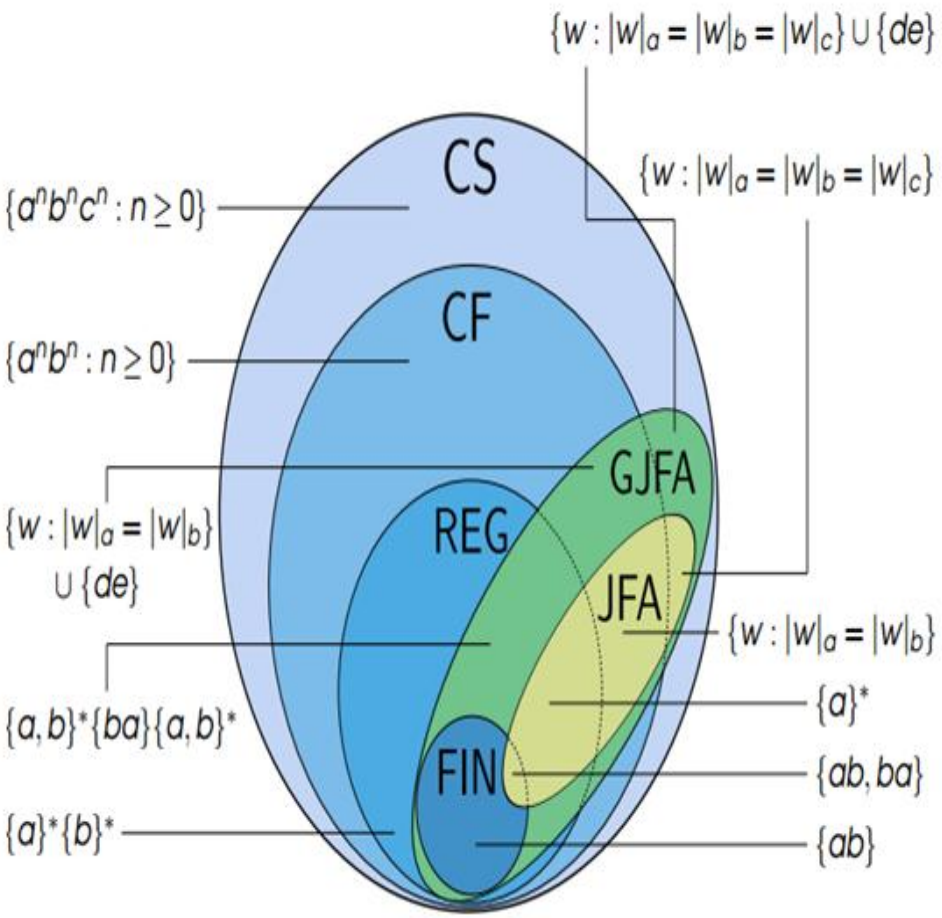


Fig. 4.4: String acceptance power [5]

4.2 Properties of JFA and GJFA

Theorem 1[5]**:** DJFA can be converted into an equivalent CJFA.

Proof: Let $M = (q, \sum, \delta, q0, F)$ be DJFA. Its analogue CJFA M' without loss of generality can be written

$$M' = (q \cup \{\partial\}, \sum, \delta'', q0, F')$$

Where $\delta = \delta'$ .If for each $y \in \sum$ and $r \in q$ there doesn't exist $ry \to t \in \delta$ .For converting into CJFA add (∂)dead state for $\sum$ transitions that do not exist. Add a new transition $ry \to \partial$ in δ' to convert DJFA to CJFA. The above newly added transition makes $L(M) = L(M')$

Theorem 2 [5]**:** JFA=DJFA=$\in$-free JFA

Proof: JFA=$\in$-free JFA by above Lemma. DJFA$\subseteq\in$-free JFA demonstrated by the postulate of the deterministic nature of jumping finite automata. Contrariwise inclusion $\in$-free JFA$\subseteq$DJFA is proved by FA to DFA. As (JFA=$\in$-free JFA) and (DJFA $\subseteq\in$-free JFA)and($\in$-free JFA$\subseteq$ DJFA) satisfies. Therefore JFA=$\in$-free JFA=DJFA.

Theorem 3 [5]**:** If $|\sum|$=1 and GJFA are $M=(Q1,\sum,\delta,s1,F)$ then $L(M)$ is regular

Proof: Let $M=(Q1,\sum,\delta,s1,F)$ be GJFA where $|\sum|$=1.Therefore, if string formed by $w\in\sum^*$ can contain only one symbol in it. Therefore *sw* kind of configuration is obtained where no jumps over input string. Therefore $L(M)$ is regular.

Theorem 4[5]**:** Let K' be arbitrary language. Then $K'\in$ JFA only if $K'=perm(K')$

Proof: Let $M=(Q1,\sum,\delta,s1,F)$ is jumping finite automata. By analogy, let us consider M is DJFA.If $w'\in L(M)$ with no loss of generality $perm(w')\subseteq L(M)$.For e.g. $perm(\in)=\in$and it belongs $L(M)$ so assume $w\neq\in$.

Without generality loss considering $w=p1,p2,p3..pn$ where $pi\in\sum$ for all $i=1,2,3...n$ for $n\geq1$Therefore $w\in L(M)$

$$s1p1\rightarrow s2,s2p2\rightarrow s3.....s(i-1)pn\rightarrow sn\in\delta$$

Where $s1,s2....sn\in Q1$.In jumping finite automata, the essential part is first searched, processed and then the optional part is traversed. For e.g. grammar $(p)*(pqr)(rr)(p)*$ accepts $ppqrrr\,or\,rrpqrp\,or\,pqrprr$. As requisite strings of grammar are (pqr) and (rr) so first they are searched, processed then optional part. Therefore, the order does not matter of input string.

Theorem 5 [5]: JFA$\subseteq$GJFA

Proof: From definition and GJFA-JFA $\neq0$ because $(ab)*$is accepted by GJFA and rejected by JFA

$$M=(\{s\},\{a,b\},\{sab\rightarrow s\},s,\{s\})$$

Theorem 6[5]**:**FIN $\subset$ GJFA

Proof: Let $K \in$ FIN.Since K is finite, therefore $|K| = n$ .Representation of K is $K \equiv \{w1, w2, ...wn\}$.GJFA defined as $M \equiv (\{s, f\}, \Sigma, \delta, s, \{f\})$ where $\Sigma = alph(K)$ and $\delta = sw1 \rightarrow f, sw2 \rightarrow f, sw3 \rightarrow f ... swn \rightarrow f$ .Therefore $L(M) = K$ and FIN $\subset$ GJFA

Theorem 7 [5]**:** No GJFA accepts $(a)*(b)*$

Proof: In $M = (Q1, \Sigma, \delta, s1, F)$ GJFA acceptance order is not fixed.It also accepts $(b)*(a)*$.

Theorem 8 [5]**:** Let Σ denote input alphabet such that cardinality $(\Sigma) \geq 2$, Then there exists a GJFA $M = (Q1, \Sigma, \delta, s1, F)$ of degree n for any $n \geq 1$ such that $L(M)$ cannot be accepted by any GJFA with a degree $n-1$.

Proof: Let Σ be input alphabet with cardinality (Σ)>2 and let there be input $a, b \in \Sigma$ such that both of them are not equal $a \neq b$.

The case where n=1 directly follows jumping finite automata. Therefore, we define GJFA with degree greater than 1 as

$$M_n = (\{s, f\}, \Sigma, \{sw \rightarrow r\}, s, \{r\})$$

$w = ab(a)n-2$. Clearly this can be analyzed that $L(M_n) = \{w\}$ .However, we now try to prove that whether a same language with degree n-1 of GJFA is accepted or rejected [5].Suppose for contradiction there be a GJFA with degree n-1, $L(H) = (Q, \Sigma, R, s', F)$ such that $L(H) = L(M_n) = \{w\}$ and |w|>n-1 there has to be

$$us'xv \rightarrow f$$

This rule has to be present in H where $w = uxv$ $u, v \in \Sigma*$, x$\in \Sigma+$, $f \in F$ and m $\geq$ 2.However then also $s'xuv \rightarrow f$ and $uvs'x \rightarrow f$ are present in H which contradicts the assumption that $L(H) = \{w\}$ .Therefore $L(M_n)$ cannot be accepted by any of GJFA with degree less than n.

Theorem 9[5]**:** $GJFA_n \subset GJFA_{n+1}$

Proof: $GJFA_n \subset GJFA_{n+1}$ it can be stated directly from the definition of General jumping finite automata with degree n for all $n \geq 0$.Therefore, it can be observed that $GJFA_n - GJFA_{n+1} \neq \varphi$

Theorem 10[5]**:** Let $M = (Q, \Sigma, R, s, F)$ be a general jumping finite automata. Let $w, x, y, z \in \Sigma^*$ and $py \rightarrow q \in R$ then

- M makes a left jump that symbolically can be written as

$$wxpyz \rightarrow_l wqxz$$

$$L(M)_L = \{u'v' \mid u, v \in \Sigma^*, usv \rightarrow_{l^*} f \text{ with } f \in F\}$$

- M makes a right jump that symbolically can be written as

$$wpyxz \rightarrow_r wxqz$$

$$L(M)_r = \{u'v' \mid u, v \in \Sigma^*, usv \rightarrow_{r^*} f \text{ with } f \in F\}$$

$GJFA_l, JFA_l, GJFA_r$ *and* JFA_r all four of them denotes the family of languages general jumping finite automata with a left jump and right jump together with jumping finite automata with a left jump and right jump.

Theorem 11 [5]**:** $GJFA_r = JFA_r = REG$

Proof: We first try to prove that jumping finite automata with right jumps equivalent to regular grammar. Consider a jumping finite automata $M = (Q, \Sigma, R, s, F)$.Observe it that since M can make only right jumps; if it can occur in the configuration of the form xpy where $x \in \Sigma^*$, $p \in Q$, $y \in \Sigma^*$.In such cases, the symbol cannot be read anymore. If jumping finite automata want to read complete string than it should be in the configuration of the form $sw$and it cannot make any jumps.

Consequently, the scenario is behaved like that of ordinary finite automata, reading the input in left to right direction, therefore, $L(M)$ is regular and so it can be stated as $JFA_r \subseteq REG$. Accordingly any finite automata can be viewed as a jumping finite automaton with the configuration of the form sw and it does not jump at all. Therefore, it can be said $REG \subseteq JFAr$ which proves that $REG = JFAr$.

Similarly, general jumping finite automata with right jumps can be proved equivalent to regular by the same method using lazy finite automation instead of finite automata.

Theorem 12 [5]**:** $JFA_a \subset JFA_b$

Proof: Let $M = (Q, \Sigma, R, s, F)$ be a jumping finite automata. The JFA

$$M' = (Q, \Sigma, R \cup \{s \rightarrow s\}, s, F)$$

Clearly satisfies $L(M)_a = L(M')_b$, therefore $JFA_a \subseteq JFA_b$.So we now try to prove this that inclusion is proper. Without loss of generality consider language $K = (a)^*(b)^*$ .The jumping finite automata

$$H = (\{s, f\}, \{a, b\}, \{sa \rightarrow f, fb \rightarrow f\}, s, \{f\})$$

$L(H)_b = K$. However observing the fact that differs from K .Therefore, it can be stated for every jumping finite automata $L(M)_a \neq K$ .Hence jumping finite automata reading string from anywhere is properly included in jumping finite automata reading string from beginning

Theorem 13 [5]**:** $JFA_l - REG \neq \varphi$

Proof: Consider the jumping finite automata

$M = (\{s, p, q\}, \{a, b\}, R, s, \{s\})$ Where R contains the rules as

$$sa \rightarrow r, rb \rightarrow s, sb \rightarrow t, ta \rightarrow s$$

We argue that

$$L(M)_l = \{w \parallel w_a| = |w_b|\}$$

With input word, M starts reading the last symbol. The symbol is read by using $sa \rightarrow r \, or \, sb \rightarrow t$, and it makes a jump to the left before the existence of the rightmost occurrence of $a \, or \, b$ Later it consumes it by using the rules respectively. Moreover, if this read left symbol was at the rightmost position, then it makes a jump one letter to the left, and again it repeats the process. Alternatively, in other case it does not jump at all [5].

Therefore in every configuration having the form urv; where r denotes the state's such that $r \in \{s,r,t\}, u \in \{a,b\}^* \; v \in \{a,\in\}\{b\}^*$.Based on observation it can be stated that $L(M)_l$ not regular because $L(M)_l = \{w \| w_a |=| w_b |\}$ is not accepted by regular.

Chapter 5

Operators and Decision problems

The decision problems are discussed in this chapter followed by examples describing the closure properties of some of the well-known operators.

5.1 Decision Problems

If there is an algorithm to solve a problem, then the decision problem is decidable, and if there is not such type of existence of an algorithm, then it is not decidable.

5.1.1 Universality

The universality problem is fundamental in the field of automata theory, and several beneficial problems of verification reduce in polynomial time to this problem. The universality problem states that, given an automaton A over some alphabet $\sum$ if the language A contains all existing words over $\sum$, that means $Lang(A) = \sum^*$ .The standard algorithm for this problem suggest for converting automata into deterministic automata using subset construction method and then to look for reachability of set, which contain only non-accepting states [28]. The subset construction can even construct deterministic automata that are exponentially large than original automata. The following algorithm demonstrates that universality is decidable for both the GJFA and JFA [26].

For verifying the universality problem in jumping finite automata we consider the following steps:

- Let $M = (Q, \Sigma, R, s, F)$ be the given expression of general jumping finite automata or jumping finite automata.
- Convert this expression in the graphical form.
  Graphical representation of the transitional graph of M is denoted by $\Delta(M)$ and $ry \to t \in \delta$ where r and t represents the node and the edge between is labeled by y .

$$M = (Q, \Sigma, R, s, F) \longrightarrow \Delta(M)$$

- For this given graph, construct power set for all nodes of the given graph using the combinatorial algorithm $P(Q)$ or 2^Q.
- For each subset created, check if it is connected or not using breadth-first search algorithm.
- If for every power set member, it is connected then all strings are accepted by given jumping finite automata reason being, if one string ends in its final state then every string made from those alphabets becomes accepted because this automation can accept permutation of given string. If one string is rejected, then none of its permutation is accepted.

5.1.2 Inclusion

The inclusion problem consists of two expression one called the left-hand expression and the other one right-hand expression [26]. Now the question is whether the language, which exists on the left hand, is being included within the language of the right-hand side expression. There exists an algorithm to check the inclusion of two jumping finite automata [28]. Therefore, it can be stated, as inclusion is decidable for both GJFA and JFA. The following steps are followed to check inclusion:

- Construct the corresponding jumping finite automata for $M_1 = (Q_1, \sum_1 R_1, s_1, F_1)$ and $M_2 = (Q_2, \sum_2 R_2, s_2, F_2)$
- Compute the complement for given automata as $L(M_2)$ .We first need to make M_2 complete if it is not complete. Therefore $M_2' = (Q_2, \sum_2, R_2, s, Q_2 - F_2)$
- Construct automation such that $L(B) = L(M_1) \cap L(M_2')$ .The construction of this automation is possible by product construction

$$B = (Q_1 \times Q_2, \ \Sigma_1 \bigcup \Sigma_2, \delta_{1,2}, F_1 \times (Q_2 \quad F_2))$$

- Check the condition if $L(B) \neq \varphi$?

  This condition is checked by a search algorithm on graph $G = (Q, E)$ where $Q = Q_1 \times Q_2$ and transitions or edges are represented by E , $L(B) \neq \varphi$ If there is a path from starting state to end state

5.1.3 Equivalence

Given $M_1 = (Q_1, \sum_1 R_1, s_1, F_1)$ and $M_2 = (Q_2, \sum_2 R_2, s_2, F_2)$ be two JFA or GJFA. Do both of them define the same set of strings? It can be proved applying inclusion on both sides; that is $L(M_1) \subseteq L(M_2) and\ L(M_2) \subseteq L(M_1)$

5.2Cut operator

Many software packages extend the operators like interleaving, counting for regular expression matching. Some of them indeed have the ability to control the nature of non-deterministic behavior inherent in the regular expression. Such kind of operators exists in real world programming. Regular expressions even have the ability to handle the processing of text-oriented problems. For such kind of problem, the regular expression that denotes language is extended by an understanding of how to apply a regular expression to string. The default case in libraries of regular expression matching is that it begins to search for the leftmost substring, and it choose the longest one[19]. This behavior is often repeatedly used to match the different regular expression against the string using the help of program control flow loop to decide about the next expression to be matched. For this type of repeated matching, pseudo code is given below, and we assume matcher function matches to the longest available prefix:

M=matcher (“(a+b)*”, s);

If (M! =null) then

If(matcher (“a b *c”, remainder)!=null) then

Return remainder == “”;

Return false;

However, this above code gives contradictory results. Consider the following examples:

1. Consider the string $s = abac$ the program first try to match the regular expression to its substring ab and it leaves the remainder that is again matched to a regular expression $R2 = a.(b*).c$, and therefore finally it return true [19]. Moreover, if

program return true for some set of string then that is a regular language but, in this case, regular language [26] cannot be defined by $R_1.R_2$

2. Consider another example to prove the above thing. Let there be a string $s = aababcc$ that can be matched by the rule $R_1.R_2$. However when execution of program takes place then R_1 will match the longest sequence of $aabcb$ and it leaves the remainder cc and this cannot be matched to R_2 .Therefore, the longest[26] matching strategy loses the non-deterministic behavior of the expression if "if statement" is combined explicitly.

5.2.1 Definition

Cut operation is a binary operation that is denoted by symbol (!) and it definition is as follows:

$$L_1!L_2 \equiv \{uv | u \in L_1, v \in L_2, uv_1 \notin L_1 \ \text{for all} \ v_1 \in prefix(v)\}$$

Cut expression is built using the operators that are allowed in regular expressions [19]. For example applying cut operator on yields to the empty language. The reason being is, for every string that's present in $L(ab*b)$ is also in $L(ab*)$ as well; it means the longest matching strategy will never leave the remainder b for second. Therefore, no such string exists, and it yields to empty language

5.2.2 Cut operator with general jumping finite automata

Consider $M_1 = (\{s\},\{a,b\}\{sab \rightarrow s\},s,\{s\})$ and $M_2 = (\{s,t,f\},\{a,b\},R,s,\{f\})$ where R defined by $sab \rightarrow f$, $fa \rightarrow f$, $fb \rightarrow f$ is GJFA. Without loss of generality if we apply cut operation on M_1 and M_2 then for some set of string as $ababab$ it produces output empty languages and for another set of strings $\{aba, abbb, ababa, ababb\}$ the output produced gives a regular expression $ab(a+b)*$ that is accepted by GJFA .Therefore, GJFA is closed under the cut operator.

5.2.3. Cut operator with jumping finite automata

Let $M_1 = (\{s,r\},\{a,b\},\{sa \rightarrow r, rb \rightarrow t\},s,\{s\})$ be a jumping finite automata. Let M_2 be $(a+b)*$ .To prove it by analogy there exists a specified set of strings $\{ab, aabb, abab, baba, bbaa\}$ .On applying cut operator, it produces output empty

language but there also exists another set of string $\{a,b,abb,aba,bab,bba,bbab,abba,baaa,bbab\}$ .Such set of string on applying cut operator with strings of M_1; gives output strings whose regular expression is $(a+b)*$ and it is accepted by jumping finite automata.Therefore, the cut operator is closed for jumping finite automata.

5.3. Cyclic shift

The cyclic shift of a language L is defined as [27] L_{CS} it is as follows:

$$L_{CS} = \{vu \mid uv \in L\}$$

5.3.1 Cyclic shift on General jumping finite automata

Let $M_1 = (\{s,t,f\},\{a,b\},R,s,\{f\})$ where R defined by $sab \to f$, $fa \to f$, $fb \to f$ be GJFA. By analogy, we prove that after applying cyclic shift operation, accepted language become $L_{CS} = (a+b)*ab$ and this language is defined by rules $sab \to f$, $fa \to f$, $fb \to f$ .Therefore, $L_{CS} = L$ both languages are similar.

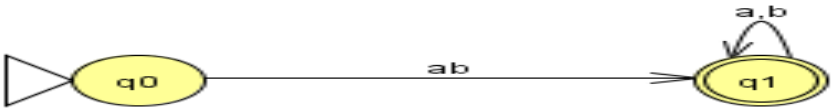


Fig. 5.1: GJFA for $L_{CS} = (a+b)*ab$

5.3.2 Cyclic shift on jumping finite automata

Let $M = (\{s,r,t\},\sum,R,s,\{s\})$ be jumping finite automata where $\Sigma = \{a,b,c\}$ and it consists of the rules defined as $sa \to r, rb \to t, tc \to s$ .Without loss of generality, we state that after applying cyclic shift the accepted language is[25]

$$L_{CS}(M) = \{w \in \Sigma^* \mid w_a| = |w_b| = |w_c|\}$$

Therefore, we conclude $L_{CS}(M) = L(M)$.

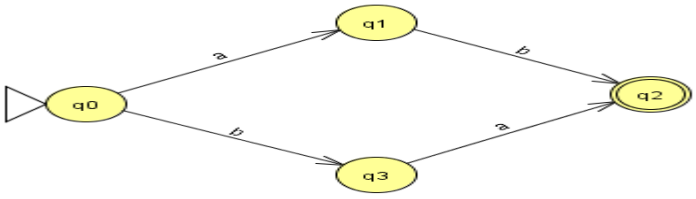
5.4. Difference

5.4.1 JFA is not closed under difference

Let $M_1 = (Q_1, \sum_1 R_1, s_1, F_1)$ and $M_2 = (Q_2, \sum_2 R_2, s_2, F_2)$ be two JFA. With no loss of generality, we first calculate the complement of $L(M_2)$ .We assume that M_2 is CJFA (complete jumping finite automata) and as already stated above every JFA =DJFA=CJFA [24] .Therefore, the JFA can be written as

$$M_2' = (Q_2, \sum_2, R_2, s, Q_2 - F_2)$$

The above JFA states that it is closed under complement operation. Now we check whether $L(M_1) \bigcap L(M_2')$ is closed. We know JFA is closed under intersection operation [5]. Therefore, JFA should be closed under difference operator. However, there exists a contradictory example. Let $L(M_1) = (ab, ba)$ and $L(M_2) = (a+b)*$ .When we apply complement on $(a+b)*$ the language becomes $(a)*(b)*$.If we then apply intersection operation, the language becomes (ab) which is not accepted by JFA.



.

Fig. 5.2: Jumping Finite automata (ab, ba)

Fig. 5.3: Jumping Finite automata $(a+b)*$

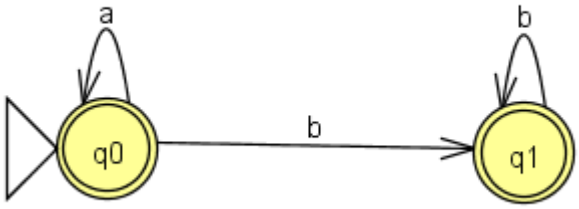


Fig. 5.4: Complement of $(a+b)*$

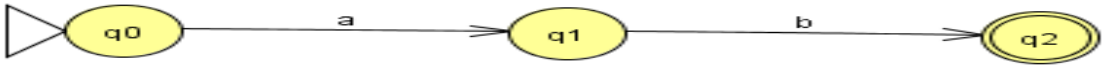


Fig. 5.5: Only *ab* as remainder, rejected by JFA

5.4.2 GJFA is not closed under difference operator

Let $M_1 = (Q_1, \sum_1 R_1, s_1, F_1)$ and $M_2 = (Q_2, \sum_2 R_2, s_2, F_2)$ be two GJFA. We next check for the complement of $L(M_2)$ .However, GJFA is not closed under complement operation. Let $L(M_2) = (a+b)*$ and therefore gives the expression that is not accepted by GJFA. Hence, GJFA cannot be stated as

$$M_2' \neq (Q_2, \sum_2, R_2, s, Q_2 - F_2)$$

GJFA is also not closed under intersection operation, $L(M_1) \bigcap L(M_2')$ is not accepted. Therefore, GJFA is not closed under difference operator.

5.4.3. JFA is closed under difference with regular

Let $M_1 = (Q_1, \sum_1 R_1, s_1, F_1)$ be JFA. Consider $L(M_1) = (ab, ba)$ and consider the regular language as $L(M_2) = (a)*(b)*$. The complement of $L(M_2')$ is $(a,b)*$ .This complement is accepted by JFA. Intersection of two JFA is always closed. Thus $L(M_1) \bigcap L(M_2')$ is always closed. Therefore, it can be stated as JFA is closed under the difference with regular.

5.4.4 GJFA is not closed under difference with regular

Let $M_1 = (Q_1, \sum_1 R_1, s_1, F_1)$ be GJFA and consider the regular language $L(M_2) = (a)*(b)*$. This regular language is not accepted by GJFA. However, we have to take the complement of $(a)*(b)*$ that is $(a+b)*$ and GJFA accepts $(a+b)*$.However, GJFA is not closed under intersection operation.Therefore $L(M_1) \bigcap L(M_2')$ is not closed.Finally, GJFA is not closed under difference with regular.

5.5. Symmetric difference

The symmetric difference of two languages can be stated as L_1, L_2 can be stated as [27]:

$$(L_1 \bigcup L_2) - (L_1 \bigcap L_2)$$

5.5.1 JFA is closed under the symmetric difference

Consider $M_1 = (Q_1, \sum_1 R_1, s_1, F_1)$ and $M_2 = (Q_2, \sum_2 R_2, s_2, F_2)$ be two JFA. Union of two JFA can be stated

$$L(M_1) \bigcup L(M_2) = (Q_1 \bigcup Q_2 \bigcup \{s\}, \sum_1 \bigcup \sum_2, R_1 \bigcup R_2 \bigcup \{s \to s_1, s \to s_2\}, s, F_1 \bigcup F_2)$$

Where we assume $s \notin (Q_1 \bigcup Q_2)$.Union of two JFA is a closed operation. Similarly as stated above JFA is also closed under intersection and difference operator. Therefore, it can be clearly analyzed that JFA is closed under symmetric difference operator

5.5.2GJFA is not closed under the symmetric difference

Consider $M_1 = (Q_1, \sum_1 R_1, s_1, F_1)$ and $M_2 = (Q_2, \sum_2 R_2, s_2, F_2)$ be two GJFA. Union of two GJFA can be stated

$$L(M_1) \bigcup L(M_2) = (Q_1 \bigcup Q_2 \bigcup \{s\}, \sum_1 \bigcup \sum_2, R_1 \bigcup R_2 \bigcup \{s \to s_1, s \to s_2\}, s, F_1 \bigcup F_2)$$

Where we assume $s \notin (Q_1 \bigcup Q_2)$.Union of two GJFA is a closed operation. By analogy as above stated GJFA is not closed under intersection and difference operator. Therefore, it can be analyzed that GJFA is not closed under symmetric difference operator

5.6 Reverse operator

Following two steps are done to convert jumping finite automata into its corresponding reverse [24,25,27] jumping finite automata.

1) Interchanging final and initial state

2) Reverse the traversal direction

5.6.1 JFA is closed under reversal operation

Let $M = (\{s,r,t\},\sum,R,s,\{s\})$ be jumping finite automata where $\sum = \{a,b,c\}$ and it consists of the rules defined as $sa \to r, rb \to t, tc \to s$ .Therefore, language accepted becomes

$$L(M) = \{w \in \sum^* || w|_a = |w|_b = |w|_c\}$$

If we reverse the transition direction and interchange the final and initial state then rules changes to $sc \to t, tb \to r, ra \to s$ Therefore the language accepted becomes.

$$Lrs(M) = \{w \in \sum^* || w|_a = |w|_b = |w|_c\}$$

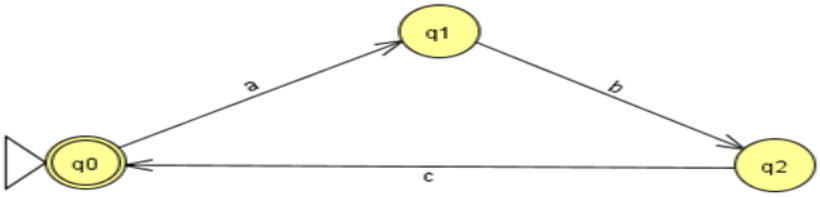


Fig. 5.6: JFA before reverse operation

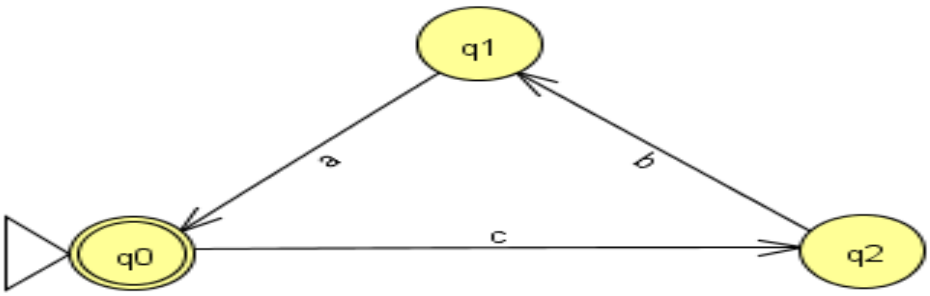


Fig. 5.7: JFA after reverse operation

5.6.2 GJFA is closed under reverse operation

Consider $M = (\{s,t,f\},\{p,q,r\},spqr \to t,trr \to f, fp \to f,\{s\},f)$ be a GJFA. The accepted language becomes $L(M) = (p)^*(pqr)(rr)(p)^*$ .When the reverse operation is applied then language becomes $Lrs(M) = (p)^*(rr)(rqp)(p)^*$and this language is accepted by GJFA.

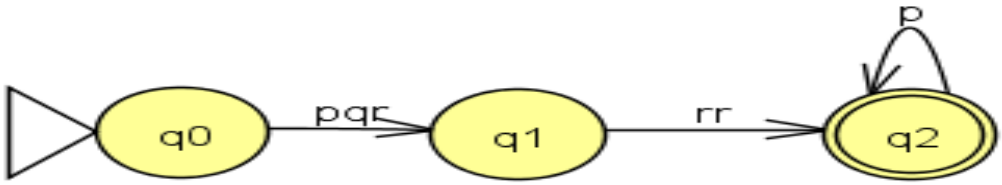


Fig. 5.8: Before reverse operator

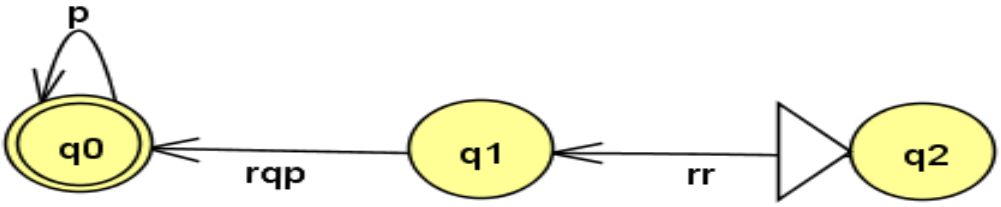


Fig. 5.9: After reverse operator

5.7. Mirror Image

Consider $M = (\{s,t,f\},\{p,q,r\},spqr \to t,trr \to f, fp \to f,\{s\},f)$ be a GJFA Therefore language accepted becomes $L(M) = (p)^*(pqr)(rr)(p)^*$[25,27].In contrast to reverse operation after applying the mirror image operation, even the rules does not change; in fact rules also remains the same $spqr \to t,trr \to f, fp \to f$.

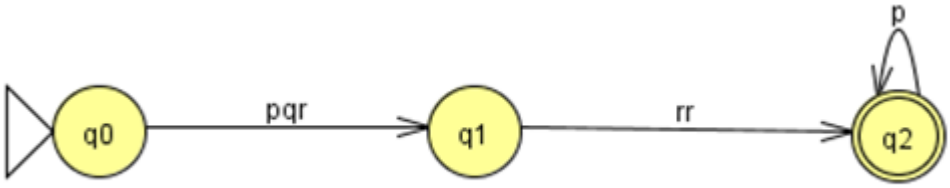


Fig. 5.10: Before mirroring image operation

Fig. 5.11: After mirror image operation

5.8. Kleene Star

Kleene Star of GJFA is closed:

5.8.1 Let $M_1 = (\{s,t,f\},\{a,b\},R,s,\{f\})$ where R defined by $sab \to f$, $fa \to f$ *and* $fb \to f$ be GJFA [5, 25]. The language accepted becomes

$$L(M) = (a,b)^*(ab)(a,b)^*$$

Without loss of generality, we now prove the effect of applying Kleene star to given GJFA

Fig. 5.12: Before Kleene star

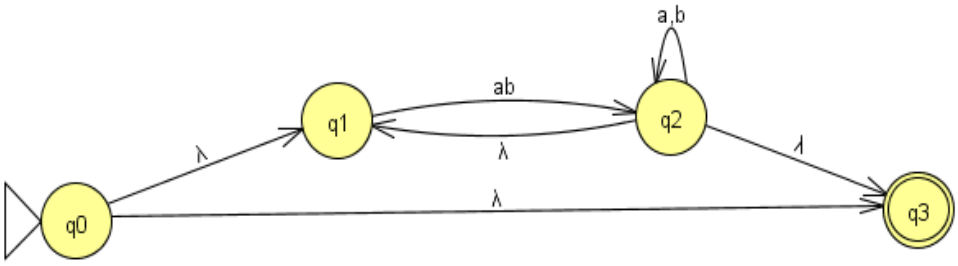


Fig. 5.13: After Kleene star

Fig. 5.14: After removal of $\in$ moves the language accepted becomes

$$L(M) = (a,b)^*(ab)(a,b)^*$$

Consider $M = (\{s,t,f\},\{p,q,r\},spqr \to t, trr \to f, fp \to f,\{s\},f)$ be a GJFA Therefore language accepted becomes $L(M) = (p)^*(pqr)(rr)(p)^*$ .Without loss of generality, we now prove the effect of applying Kleene star to given GJFA.

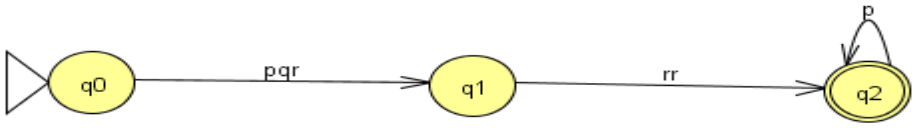


Fig. 5.15: Before Kleene star

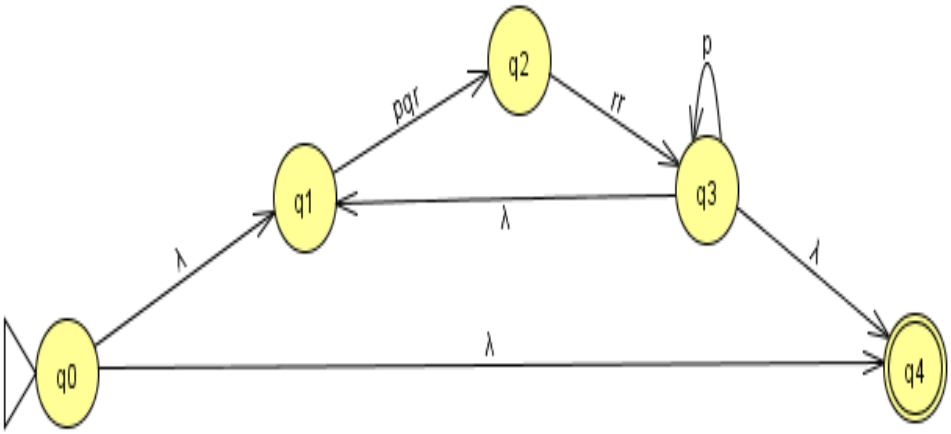


Fig. 5.16: After Kleene star

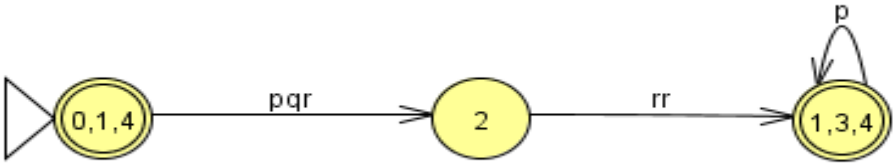


Fig. 5.17: After removal of $\in$ moves the accepted language becomes

$$L(M) = (p)^*(pqr)(rr)(p)^*$$

5. 9. Kleene plus of GJFA is closed

By analogy similar to Kleene star. Let $M_1 = (\{s,t,f\},\{a,b\},R,s,\{f\})$ where R defined by $sab \to f$, $fa \to f$ *and* $fb \to f$ be GJFA. The language accepted becomes

$$L(M) = (a,b)^*(ab)(a,b)^*$$

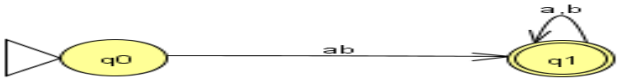


Fig. 5.18: Before Kleene plus

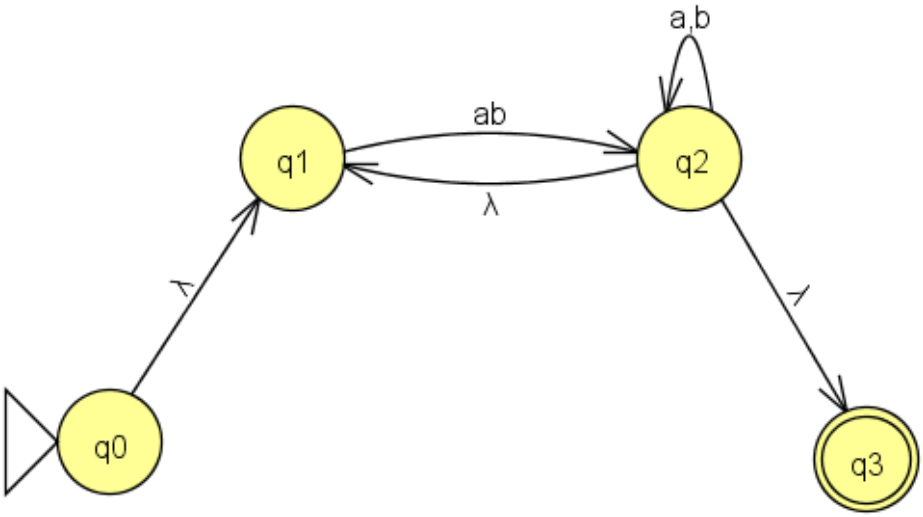


Fig. 5.19: After Kleene plus

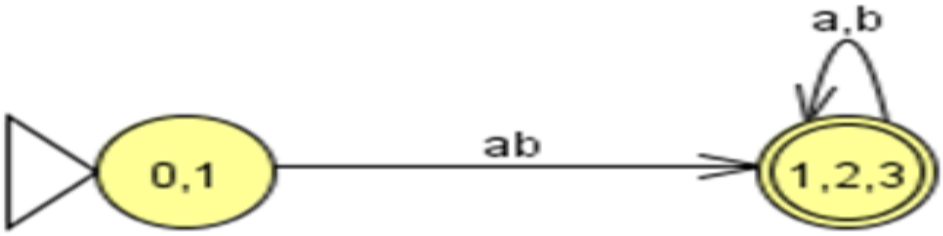


Fig. 5.20: After removal of $\in$ moves the language accepted becomes

$$L(M) = (a,b)^*(ab)(a,b)^*$$

5.10. Orthogonal Concatenation

Orthogonal Concatenation is worth studying in formal languages, and here we demonstrate the results of orthogonal concatenation in jumping finite automata [32]. We define the closure properties of GJFA and JFA under this operation. Orthogonal concatenation operation practical use is in the networking field. CDMA (Code division multiple access) is the multiplexing scheme that is practically used in radio communication, it allows simultaneous reception by assigning every sender a different waveform.

If, in case, it is superimposed with another sender waveform then it can generate a signal that allows unique decomposition of the superimposed signal back to the original waveform. This is practically possible due to the presence of additive nature of the radio waves that are interfering and due to careful and handy choice of mutually orthogonal waveforms.

Therefore, this form of study needs to generalize the idea of applying orthogonal concatenation to the arbitrary word for facilitating the code division multiplexing investigation in abstract coding theory [32]. In binary signaling case, waveforms can be represented as binary vector forms of finite size, and this leads to a choice of containing that waveform that can be represented in the sense of traditional linear algebra by orthogonal vectors.

5.10.1 Definition of orthogonal concatenation

We first start by introducing the idea of Orthogonal operation. It is a binary word operation on Σ^* it denotes a function such that $o:(\Sigma)^* \rightarrow 2\,\Sigma^*$. The operation o can be extended to languages in the form such that $L_1 L_2 \subseteq \Sigma^*$ as

$$L_1 o L_2 \equiv \bigcup_{y \in L2}^{x \in L1} xoy$$

Let there be three languages $L1, L2 \text{ and } L$ over some $\sum$.We can then say that L is $orthogonal\ \ compositon$ of the languages $L1, L2$ denoted as [32]

$$L = L_1 o \perp L_2$$

If one of these following conditions holds:

(OR 1) $L = L_1 \circ L_2$

(OR2) $(u_i, v_i \in Li, i = 1,2) \quad if \quad (u_1, u_2) \neq (v_1, v_2) \quad then (u_1 \circ u_2 \cap v_1 \circ v_2 = \phi)$

Orthogonal concatenation is represented by the symbol $\bullet\perp$. It is binary operation between two languages L_1 and L_2 shown [24] as $L = L_1 \bullet\perp L_2$ .It states that for every word present in L ;if that word can be uniquely formed by the concatenation of words from L_1 *and* L_2 then it is said to be closed otherwise it remains undefined.

5.10.2 JFA is not closed for orthogonal concatenation

Let $M_1 = (Q_1, \sum_1 R_1, s_1, F_1)$ and $M_2 = (Q_2, \sum_2 R_2, s_2, F_2)$ be two JFA. Let $L(M_1) = \{w \in \Sigma^* \| w|_a = |w|_b = |w|_c\}$ and $L(M_2) = (a+b)^*$ .Without loss of generality when orthogonal concatenation operation is applied $L = L_1 \bullet\perp L_2$ then string *ababab* can be formed in two ways. If L_1 contain *abab* and L_2 contain *ab* or vice – versa. Therefore, every word is not unique hence JFA is not closed under orthogonal concatenation.

5.10.3GJFA is not closed orthogonal concatenation

General jumping finite automata can also be proved in similar way as given above for jumping finite automata

5.11 Init Operator

Prefix operation is also known as initoperation [25, 27].In automata init operation is checked by making all states as final state so that all prefixes can be accepted including $\in$.

5.11.1 GJFA is not closed for init operation

Let $M = (Q1, \Sigma, \delta, s1, F1)$ be GJFA. Without loss of generality assumption is GJFA reading of string is discontinuous. Now consider GJFA reading prefixes, other non-prefix acceptance makes it contradictory. For e.g. consider language $L = (p)^*(pqr)(rr)(p)^*$contain string as *prrpqrp* which have prefix as *prrpqr* but it is also showing affirmation to *rrpqrp*

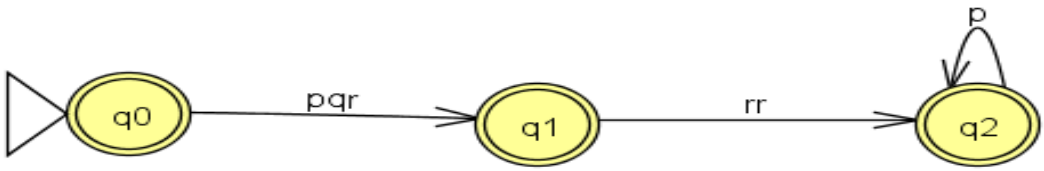


Fig. 5.21: GJFA init operation

5.11.2 JFA is closed for init operation

Let $M = (Q1, \Sigma, \delta, s1, F1)$ be JFA.As all permutation of string are accepted and jumping finite automata can jump in either direction to read an input symbol; it show positive affirmation for jumping finite automata.

5.12 Substitution

5.12.1Both GJFA and JFA are not closed

Considering the language $K' = (ab, ba)$ accepted [26, 27] by JFA as explained above. Now we explicate σ from $(a,b)*$ to $2^{(a',b')*}$ as $\sigma(a) = (a)*$ $\sigma(b) = (b)*$.Therefore, it demonstrate $\sigma(a)$ and $\sigma(b)$ are showing positive affirmation to JFA. $\sigma(K)$ Shows negative affirmation to GJFA.

5.12.2 GJFA is closed under finite substitution.

Let $M = (Q1, \Sigma, \delta, s1, F1)$ be a GJFA. Moreover, let there be alphabet (ϑ) and σ be finite substitution from Σ^* *to* $2^{\vartheta*}$.GJFA, therefore, accepts language as $\sigma(L(M))$

$$M' = (Q1, \vartheta, \delta', s1, F1)$$

Where rules changed to

$$\delta' = \{py' \to q \mid y' \in \sigma(y) \mid py \to q \in \delta\}$$

5.12.3 Regular substitution of JFA and GJFA not closed.

Considering the language $K' = (ab, ba)$ accepted by JFA as explained above [26, 27]. Now we explicate σ from $(a,b)*$ to $2^{\{a',b'\}*}$ as $\sigma(a) = (a)*$ $\sigma(b) = (b)*$.Therefore, it

demonstrate and $\sigma(b)$ are showing positive affirmation to JFA. $\sigma(K)$ Shows negative affirmation to GJFA.

5.13Homomorphism

5.13.1 JFA is closed under $\in -free$ homomorphism

The $\in -free$ homomorphism φ from $(a)to(a,b)^{+}$ as φ $(a) = ab$ and [25] considering $(a)*$ language, which shows positive affirmation by the JFA.

$$M = (\{s\},\{a\},\{sa \rightarrow s\},\{s\})$$

Now $(L(M)) = (ab)*$ cannot be accepted by any JFA. Hence, it is not closed for JFA.

5.13.2 Inverse homomorphism is closed for GJFA and JFA

Let $M = (Q1, \Sigma, \delta, s1, F1)$ be GJFA, the alphabet is $\sum$ and φ be homomorphism [27] from $\sum *to\, \vartheta *$.Therefore, the construct of JFA can be M' as $L(M') =$ φ_{inverse} $L(M)$ defined by [26]

$$M' = (Q1, \vartheta, \delta', s1, F1)$$

Where rules modified as

$$\delta' = \{pa \rightarrow q \mid a \in \ , p \ \ (a) \mapsto q \quad in\ \Delta(M)\}$$

Observing that $w1s1w2 \rightarrow^{*} q$ in M if $w1's1w2' \mapsto *q$ in M' where $w1w2 = \varphi(w1'w2')$ such that $L(M') =$ φ_{inverse} $L(M)$

5.14 Shuffle

Shuffle operation is used in different context, and it appears to be one of the fundamental operations in the field of theory of automation [31]. There exist many variations of shuffle like that of insertion; literal shuffle, etc.

- **Shuffle:** Let $w \in L_1$ and $v \in L_2$ be two words. If $w = x_1x_2..x_n$ and $v = y_1y_2..y_n$ .The word formed by shuffling these two words is $shuffle(W_1,V_1) = \{x_1y_1x_2y_2....x_ny_n)$

Various different aspects are being studied of shuffling in the field of formal languages [30].It is worth noting that shuffle operation is used for verification and modeling, in a database system, etc. Let's understand these different aspects of shuffle in detail:

- In the verification and modeling of systems with the help of interleaving of processes. Transition based model have an advantage over the algebraic system because they are graphical in nature whereas other just promote verification and hierarchical description. For this reason, it is beneficial and easier sometimes to use transition based system and other time use the algebraic system. However, according to Garg and Ragunath [18]. formal description technique should have the capability of accepting both styles of description. Algebraic model known as concurrent regular expressions is used to model concurrent systems [18]. The concurrent regular expression can be transformed automatically into corresponding Petri nets and, therefore, all techniques used for Petri nets analysis can be used.
- The shuffle operator is also used in XML database systems for the purpose of schema definition [20].
- In plan recognition, where the task is to infer the goals and plans of the agent, and this is based on actions performed by an agent or the effects generated by those actions. Different field that refers to plan recognition is intent, activity, behavior, goal recognition. It is also a mean for automatic threat detection and is used by military intelligence. This plan recognition problem involves steps for deducing the complete work based on a set of observations [21]. The problem is further classified as keyhole or intended or obstructed. In intended, agents deliberately try to make their action whereas this is not the case for a keyhole. In case of obstructed plan recognition agents try to confuse potential observers with their erroneous plans.
- The interest is growing in the field of linguistic models for the languages with free ordering of words. While scanning a sentence parser starts building a tree

by going from left to right through all the words of the sentence. Graph based parser start with the completely connected graph in which edges are weighted according to their statistical model [22].

They then emphasize to find spanning tree that covers all nodes of the graph (words) and at the same moment of time maximizes the sum of spanning tree. Transition based parsers are frequently used with a degree of polynomial kernels. On the other side vector machine with linear support provide faster training and faster classification.

Two types of shuffling operations are studied; they are ordinary shuffle and perfect shuffle. For given two words $x = a_1a_2a_3.....a_n$ $y = b_1b_2b_3.....b_n$ that are of the same length, therefore, perfect shuffle defined as $x\ shuffle\ y = y = a_1b_1a_2b_2a_3b_3.....a_nb_n$ .For example $terms\ shuffle\ hoes = theorems$ .Also, note that perfect shuffle is not equivalent to reversed words means $x\,shuffle\,y \neq y\,shuffle\,x$ .This definition can be extended to languages, and it is as follows:

$$L_1\ shuffle\ L_2 \equiv \bigcup_{x\in L1, y\in L2, |x|=|y|} x\ shuffle\ y$$

If x_r denote reverse of the word x , therefore, note the fact that $(x\ shuffle\ y)_r \equiv y_r\ shuffle\ x_r$ .However, sometimes perfect shuffle also allow the fact $|y|=|x|+1$ .In this case $x\,shuffle\,y = y = a_1b_1a_2b_2a_3b_3.....a_nb_{n+1}$. The second type of shuffle called ordinary shuffle. It is a finite set, where each member of the set is formed by two words.It contains the set of words, which are obtained by merging the words in the direction from left to right, but, in this case, the next symbol is chosen arbitrarily from $x\,or\,y$ .Hence the mathematical form of ordinary shuffle: $x\,shuffle\,y \equiv \{z : z = x_1y_1x_2y_2..x_ny_n\ \ for\ some\ n > 1\ and\ for\ words\ x = x_1x_2..x_n, y = y_1y_2..y_n\}$

Un-Shuffling of a finite word $w = a_1a_2....a_n$ can be decimated in two ways:

- Odd(w)= $a_1a_3....a_{n(n+1) \bmod 2}$

  Even (w) = $a_2a_4....a_{n \bmod 2}$

- Similarly decimation can be applied to extract first and second half

  First half (w) = $a_1a_2...a_{n/2}$

Last half (w) = $a_{n/2+1}.....a_n$

Four variations of un-shuffling exist and they are binary decimation, binary decimation with the reverse, inverse decimation and inverse decimation with reverse.

5.14.1 JFA is closed under shuffle

Consider $M_1 = (Q_1, \sum_1 R_1, s_1, F_1)$ and $M_2 = (Q_2, \sum_2 R_2, s_2, F_2)$ be two JFA. Now without loss of generality, the assumption can be made that $Q_1 \cap Q_2 \equiv \phi$ .Therefore, JFA can be defined as

$$H = (Q_1 \bigcup Q2, \sum_1 \cup \sum_2, R_1 \bigcup R_2 \bigcup \{ f \to s_2 \mid f \in F_1 \}, s_1, F_2)$$

Analyze the fact that $L(H) = shuffle(L(M_1), L(M_2))$ we need to observe the working of H. On string such that $w \in (\sum_1 \bigcup \sum_2)*$, H first runs over M_1 on w and when it reaches final state then it runs M_2 on remaining input. If M_2 ends in final state while reading input, then it accept w. Otherwise, it rejects w.

The above operation can be proved by $L(Mi) = perm(L(Mi))$. Let $M = (Q1, \sum, \delta, s1, F)$ is jumping finite automata. By analogy let us consider M is DJFA. If $w' \in L(M)$ with no loss of generality $perm(w') \subseteq L(M)$ .For e.g. $perm(\in) = \in$ and it belongs $L(M)$ so assume $w \neq \in$.

Without generality loss considering $w = p_1, p_2, p_3..p_n$ where $pi \in \sum$ for all $i = 1,2,3...n$ for $n \geq 1$ Therefore $w \in L(M)$

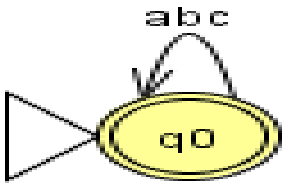
$$s_1 p_1 \longrightarrow s_2, s_2 p_2 \longrightarrow s_3.....s(i \quad 1) p_n \longrightarrow s_n \in \delta$$

5.14.2 GJFA is not closed under shuffle

Let $M_1 = (Q_1, \sum_1 R_1, s_1, F_1)$ and $M_2 = (Q_2, \sum_2 R_2, s_2, F_2)$ be two GJFA. Consider $L(M_1)$ as $(ab)*$ and $L(M_2)$ as $(a+b)*$. If we apply shuffle on the words of both the language, the language formed of shuffled words is $(a+b)*$ .Moreover, this language is accepted by GJFA.

Similarly consider another example $L(M_1)$ as $(a+b)*$ $L(M_2)$ $(a+b)*ab(a+b)*$ .If we apply shuffle on the words of both the language, then the language formed of

shuffled words is $(a+b)*$.Moreover, this language is accepted by GJFA.Thirdly consider another example $L(M_1)$ $(ab)*$ $L(M_2)$ $(a+b)*ab(a+b)*$.If we apply shuffle on the words of both the language then language of shuffled words becomes $(a+b)*ab(a+b)*$.Moreover, this language is accepted by GJFA.



.

Fig. 5.22: GJFA for $(abc)*$

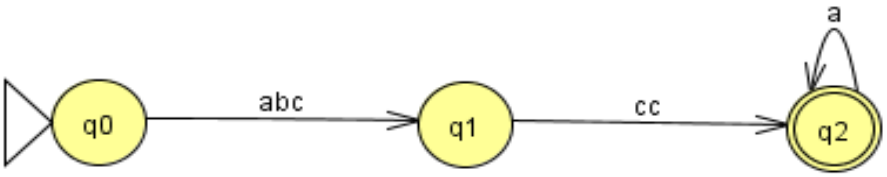


Fig. 5.23: GJFA for $(a)*(abc)(cc)(a)*$

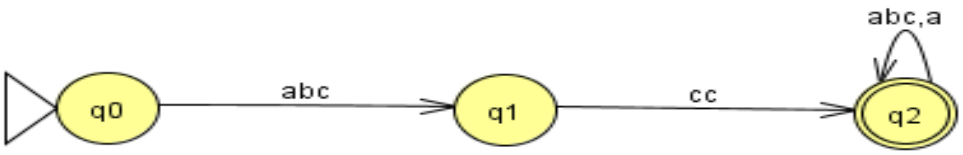


Fig. 5.24: GJFA for $(a)*(abc)*(abc)(cc)(abc)*(a)*$

However, there exists a contradictory case. Let us consider $M_1 = (Q_1, \sum_1 R_1, s_1, F_1)$ and $M_2 = (Q_2, \sum_2 R_2, s_2, F_2)$ be two GJFA. Let $L(M_1)$ as $(abc)*$ $L(M_2)$ $(a)*(abc)(cc)(a)*$. Now apply shuffle operation on both languages words. Therefore, the final language of shuffled words becomes $(a)*(abc)*(abc)(cc)(abc)*(a)*$ .This language is accepted by GJFA. Now consider a word $ccabcabc$, which is

accepted by above language. However, this should be rejected because in shuffle operation cc cannot come before *abc* whereas in general jumping finite automata direction does not matter, only the occurrence of essential part matters.Hence, it can be said that general jumping finite automata are not closed under shuffle operation because it is giving contradictory results

Chapter 6

Testing and Results

This chapter provides the description of a tool developed for checking the acceptance power of general jumping finite automata and jumping finite automata with heuristics proposed by Alexander and Peter.

6.1 Software Description

The tool is designed in Java language to demonstrate the working of jumping finite automata. It has the ability of discontinuous reading of input symbol. It takes the pattern and input string as input and specifies whether GJFA or JFA accept it. This tool has the limitation of accepting the only regular expression as a pattern.

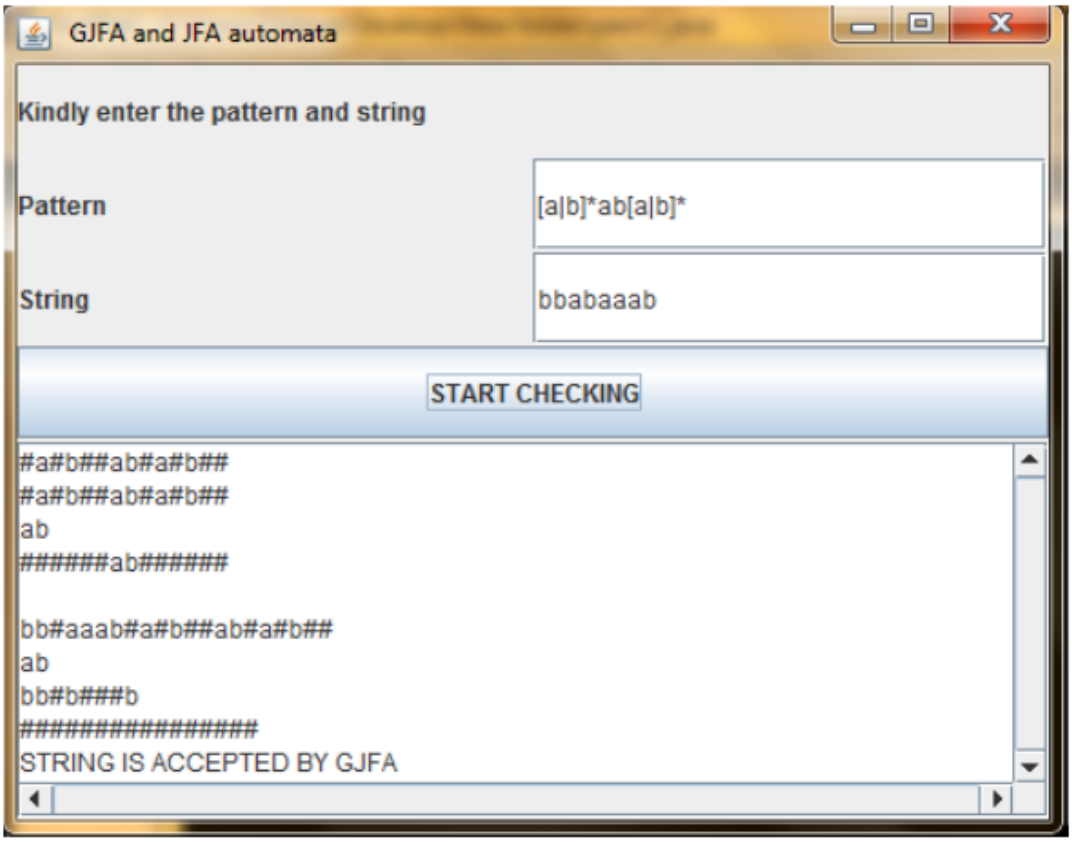


Fig. 6.1: String acceptance for GJFA $(a+b)^* ab(a+b)^*$

Firstly it checks whether given string is made from input alphabet. Then it checks whether given pattern represents general jumping finite automata or jumping finite automata.

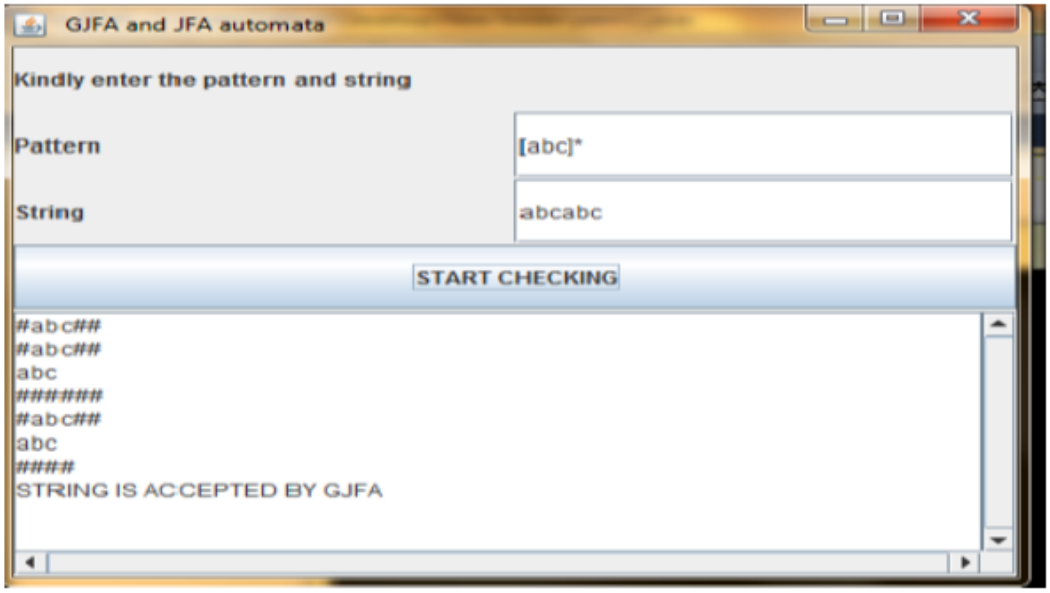


Fig. 6.2: String acceptance for GJFA $(abc)^*$.

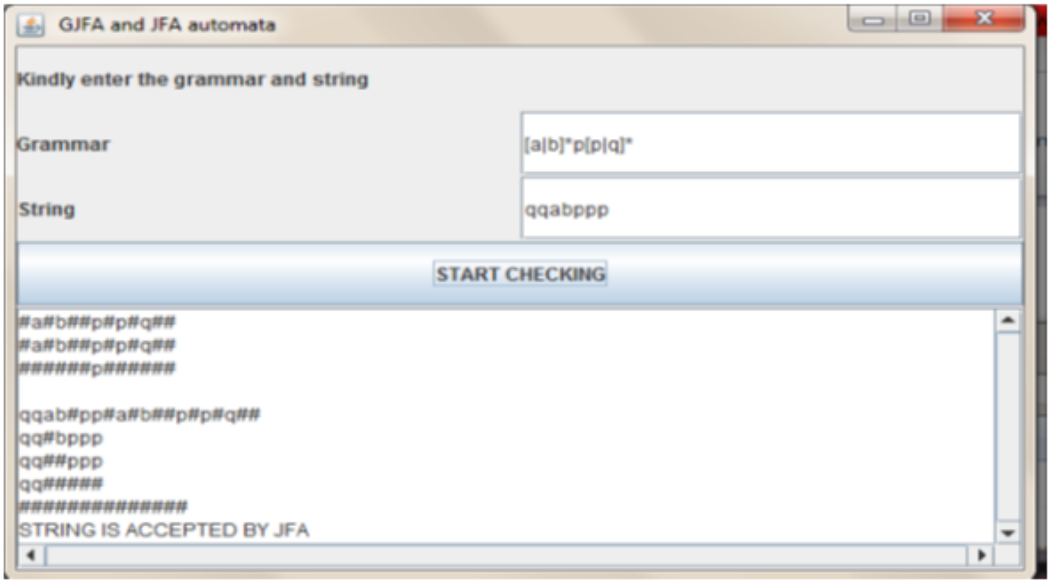


Fig. 6.3: String acceptance for JFA $(a+b)^* p(p+q)^*$

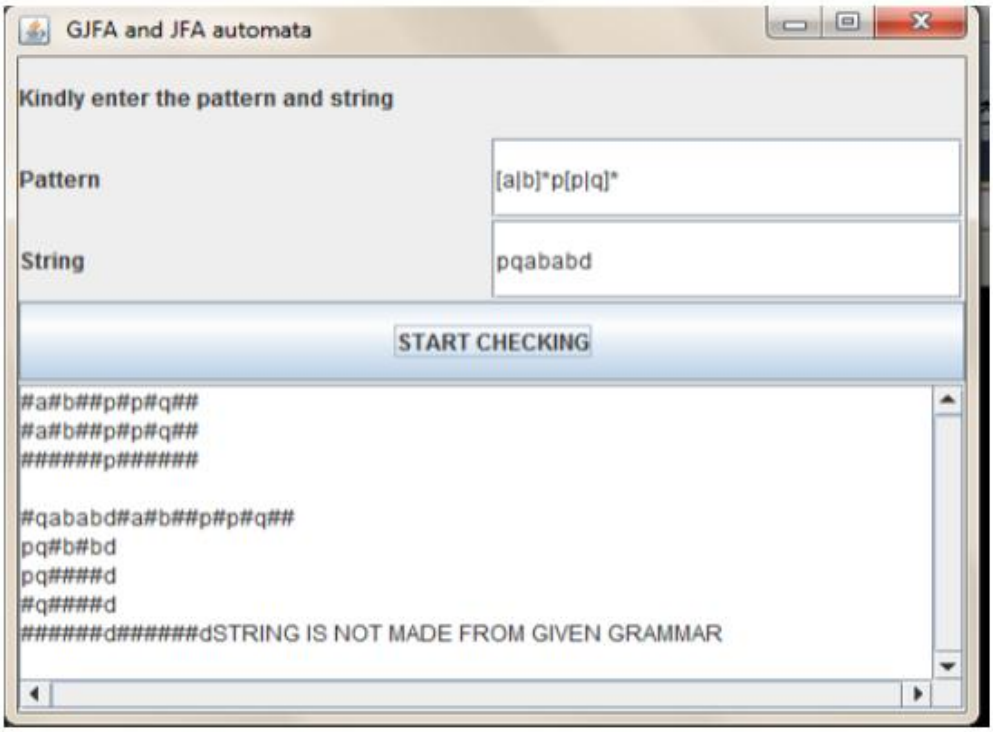


Fig. 6.4: String not made of given pattern $(a+b)^* p(p+q)^*$

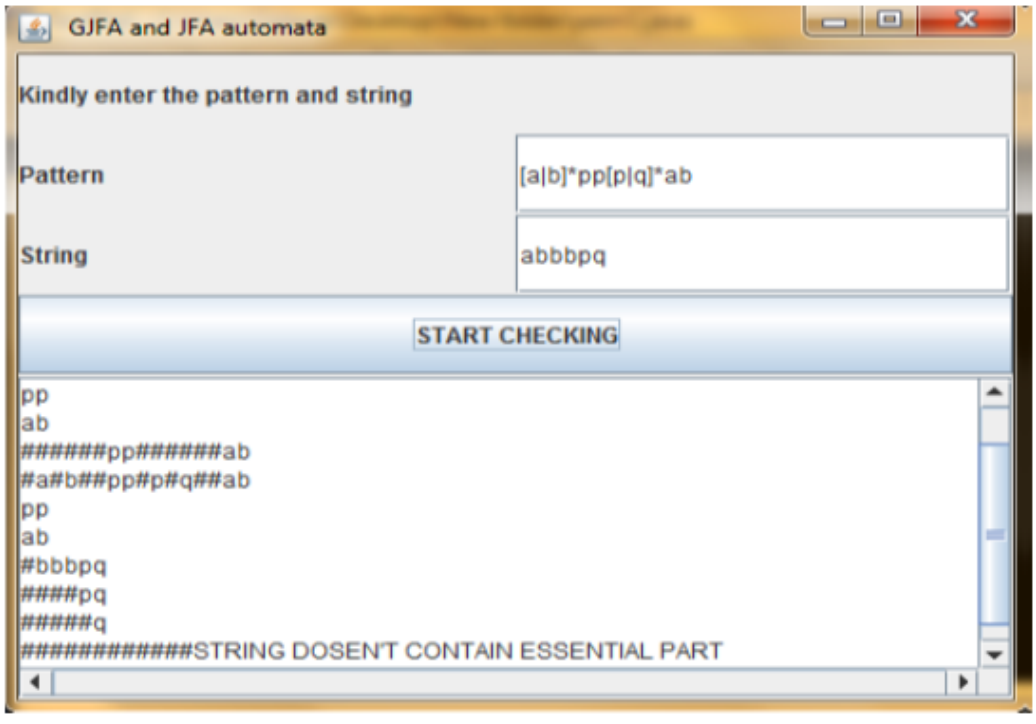


Fig. 6.5: String does not contain essential part *pp*

Chapter 7

Conclusion and Future scope

The thesis work provides beneficial insight for a newly investigated topic in the field of automation called jumping finite automata. The discontinuous reading of input string makes this automaton more powerful than the formal models.

7.1 Conclusion

The following conclusion can be made from the presented thesis work:

- GJFA is closed under cut,cyclic shift, reverse, mirror image, Kleene star, Kleene plus, finite substitution, inverse homomorphism operator .
- JFA is closed under cut ,cyclic shift, difference with regular, symmetric difference, revere, shuffle, init, inverse homomorphism.
- GJFA is not closed under difference, difference with regular, symmetric difference, shuffle, orthogonal concatenation, init, substitution, regular substitution.
- JFA is not closed under difference, orthogonal concatenation, substitution, and regular substitution.
- Universality, inclusion, and equivalence are decidable for GJFA and JFA.
- A tool is designed in Java for implementing the properties of jumping finite automata and general jumping finite automata.

7.2 Future scope

Following are the future directions on which work can be carried out:

- Acceptance power of GJFA and JFA under right jumps have been presently considered by Meduna [5].However how left jump effects acceptance power is today also on an open problem.
- In jumping finite automata, present determinism requirement does not guarantee deterministic behavior for discontinuous automata. So how requirements can be modified to meet the same is presently open challenge.
- To enhance the acceptance power of the pattern in the designed tool.

References

[1] R. Alur and P. Madhusudan, “Adding nesting structure to words,” *Journal of the ACM*, vol. 56, no. 3, pp. 1-43, 2009

[2] S. Bensch, H. Bordihn, M. Holzer and M. Kutrib, “On input-revolving deterministic and nondeterministic finite automata,” *Information and Computing*, vol. 207, no. 11, pp. 1140-1155, 2009.

[3] S. Buettecher, C. L. A. Clarke and G. V. Cormack, Information retrieval:Implementing and evaluating Search Engines., Cambridge: The MIT Press, 2010.

[4] D. A. Grossman and O. Frieder, Information Retrieval: Algorithms and Heuristics, Berlin: Springer, 2004.

[5] A. Meduna and P. Zemek, “Jumping Finite Automata,”*International Journal of Foundation of Computer Science*, vol. 23, no. 7, pp. 1555-1578, 2012.

[6] P. Leupold and A. Meduna, “Finitely expandable deep PDAs,” in *Automata, Formal Languages and Algebric Systems: Proceedings of AFLAS*, Hong kong, 2008.

[7] R. Baeza-Yates and B. Ribeiro-Neto, Modern Information Retrieval: The Concepts and Technology behind Search, boston: Addison-Wesley Professional, 2011.

[8] R. Alur, M. Arenas, P. Barcel'o, K. Etessami, N. Immerman and L. Libkin, “Firstorder and temporal logics for nested words,”*Logical Methods in Computer Science*, vol. 4, no. 4, 2008.

[9] C. Moore and J. Crutchfield, “Quantum automata and quantum grammars,”*Theoretical Computer Science* , vol. 237, no. (1-2), pp. 275-306, 2000.

[10] D. Kolar and A. Meduna, "Regulated pushdown automata," *ActaCybernetica*, vol. 2000, no. 4, pp. 653-664, 2000.

[11] G. Vaszil, E. Csuhaj-varj, T. Masopust, "Blackhole state-controlled regulated pushdown automata," in *Second workshop on Non-Classical Models for Automated Applications*, 2010.

[12] C. D. Manning, P. Raghavan and H. Schutze, Introduction to Information Retrieval, New York: Cambridge University Press, 2008.

[13] M. Cermak and A. Meduna, "N-accepting restricted pushdown automata systems,"in *13th Internatrional Conference on Automata and Formal Language*, pp. 168-183, 2011.

[14] M. Daley, M. Eramian and I. McQuillan, "Bag automata and stochastic retrieval of biomolecules in solution," in *Implementation and Application of Automata 8thInternational Conference CIAA* , 2003.

[15] B. Bouchon-Meunier, G. Coletti and R. R. Yager, Information Processing: From Theory to Applications, New York: Elsevier Science, 2006.

[16] A. Cherubini, L. Breveglieri, C. Citrini, and S.C. Reghizzi, "Multipushdown languages and grammars,"*International Journal of Foundations of Computer Science* , vol. 7, no. 3, pp. 253-292, 1996.

[17] R. Alur and P. Madhusudan, "Visibly pushdown languages," in *Annual ACM Symposium on Theory of Computing*, Chicago, 2004.

[18] V. Garg and M. Ragunath, "Concurrent regular expressions and their relationship to petrinets,"*Theoretical Computer Science,* vol. 96, no. 2, pp. 285-304, 1992.

[19] M. Berglund, H. Bjourklund, F. Drewes, B. Nerwe and B. Watson, "Cuts in regular expressions,"*Development in Language Theory in Computer Science,* vol. 79, pp. 70-81, 2013.

[20] W. Gelade, W. Martens and F. Neven, "Optimizing Schema languages for XML:Numerical constraints and interleaving,"in *SIAM Journal on Computing*, vol. 38, no. 5, pp. 1484-1530, 2009.

[21] S. Carberry, "Techniques for plan recognition,"*User Modeling and User-Adapted Interaction* , vol. 11, no. 1-2, pp. 31-48, 2001.

[22] J.Nivre, "Non-projective dependency parsing in expected linear time," in *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL*, vol. 1, pp 351-359,2009.

[23] N. Nisan and S.Schoken, The elements of computing system building a modren computer from first principles, Cambridge: The MIT press, 2005.

[24] W. Gelade, "Succinctness of regular expressions with interleaving, intersection and counting,"*Theoretical computer science*, vol. 411, no. 31, pp. 2987-2998, 2010.

[25] Ullman, J. E. Hopcroft and R. Motwani, "Introduction to Automata Theory, Languages, and Computation," Pearson Education Inc., ISBN 0-201-44124-1. Addison Wesley, 2001.

[26] D. Wood, Theory of computation: A primer, Boston: Addison- Wesley, 1987.

[27] C. Campeanu, K. Culikand and K.Salomaa, "State complexity of basic operations on finite languages," in *Automata Implementation*, pp. 60-70,Springer, Heidelberg, 2001.

[28] W. Martens, F. Neven and T. Schwentick, "Complexity of decision problems for simple regular expressions," *Mathematical Foundations of Computer Science* , pp. 889-900, Springer Berlin Heidelberg, 2004.

[29] JFLAP, http://www.jflap.org/,[Online]

[30] M. Latteux and Y. Roos, "Synchronized shuffle and regular languages," in *Jewels are Forever*, pp. 35-44, Springer Berlin Heidelberg, 1999.

[31] M. Berglund, H. Björklund and J. Högberg. "Recognizing shuffled languages," in *Language and Automata Theory and Applications*, vol. 6638, pp. 142-154, Springer Berlin Heidelberg, 2011.

[32] M. Daley, M. Domaratzki and K. Salomaa, "Orthogonal Concatenation: Language Equations and State Complexity," *Journal of Universal Computer Science*, vol.16, no. 5, pp. 653-675, 2010.

List of Publications

Communicated

- Shweta Sharma and Ajay Kumar, "TGJFA: Java Tool for General Jumping Finite Automata", Paper ID 199, 6th International Conference on Computer and Communication Technology, Allahabad, Uttar Pradesh, India.
- Shweta Sharma and Ajay Kumar, "Operators of Jumping Finite Automata", Paper ID 456, Next Generation Computing Technologies, Dehradun, Uttarakhand , India.

Video Presentation

Link to video: https://www.youtube.com/watch?v=n85txcRTbbQ