SWITCHOVER AUTOMATA

A Project Report Submitted

in Partial Fulfilment of the Requirements

for the Degree of

BACHELOR OF TECHNOLOGY

in

Mathematics and Computing

by

Kavya Basuti

Roll No. 140123009

Prakhaar Bhargava

Roll No. 140123051

to the

DEPARTMENT OF MATHEMATICS

INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI

GUWAHATI - 781039, INDIA

April 2018

CERTIFICATE

This is to certify that the work contained in this project report entitled "**Switchover Automata**" submitted by **Kavya Basuti** (**Roll No.: 140123009**) and **Prakhaar Bhargava** (**Roll No.: 140123051**) to Department of Mathematics, Indian Institute of Technology Guwahati towards the requirement of **Bachelor of Technology** in Mathematics and Computing has been carried out by them under my supervision.

It is also certified that, along with the literature survey, a few new results are established. Further, empirical analysis has also been carried out by the students under the project.

Turnitin Similarity: _____%

Guwahati - 781 039 (Prof. K.V.Krishna)

April 2018 Project Supervisor

ABSTRACT

Given an automaton, a Switchover of the automaton is obtained by changing some transitions in it. Utilizing this concept we investigate the problem of whether binary circular automata is synchronizing or not. In this connection, using Černý automata we systematically develop a technique to study the problem.

Contents

List of Figures

Introduction

In this report, we briefly survey synchronizing automata and the Černý Conjecture from [9]. We then introduce the concept of Switchover Automata and investigate the problem of whether binary circular automata is synchronizing or not. The report has been organized into 3 chapters as described below.

In chapter 1, we first look what a synchronizing automata is and look at several algorithms used to detect whether an automata is synchronizing or not. Further, we see that computing the shortest reset word of a synchronizing automata is an NP-complete problem.

In chapter 2, we look at Cerny conjecture and a greedy algorithm which finds a reset word of synchronizing automata, not necessarily the shortest one.

In chapter 3, we introduce the concept of Switchover Automata. In order to investigate the synchronization problem of binary circular automata, utilizing the Cerny automata, we study 1-Switchover Automata of $\mathscr{C}_n$ and we identify various synchronizing automata, non-synchronizing automata and certain empirical results.

Chapter 1

Synchronizing Automata

1.1 Introduction

The very first synchronizing automaton that we were able to trace back in the literature appeared in Ashbys classic book ([2], pp. 6061) where he presented a puzzle dealing with taming two ghostly noises, Singing and Laughter, in a haunted mansion. Each of the noises can be either on or off, and their behaviour depends on combinations of two possible actions, playing the organ or burning incense.
In figure 1.1, 00 encodes the state when both Singing and Laughter are silent, 01 stands for the state when Singing is off but Laughter is on, etc.
Similarly, a stands for the transition that happens when neither the organ is played nor incense is burned, b encodes the transition caused by organ-playing in the absence of incense-burning, etc. The problem is to ensure silence, in other words, to bring the automaton in figure above to the state 00.

Ashby only solves the problem under the assumption that the automaton

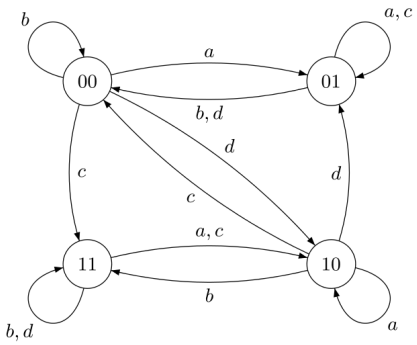


Figure 1.1: Ashby ghost taming automaton

is in the state 11 and his suggested solution is encoded by the word *acb* . However, it is easy to check that *acb* is in fact a reset word for the automaton so applying the corresponding sequence of actions will get the house quiet from any initial configuration.

Let us consider another application in assembling parts of certain device. Parts come in four orientation as shown:

We need to assemble the pieces as the last orientation. We will put obstacles

Fig. 4. Four possible orientations

such that the pieces orient themselves in one orientation. there are two types of obstacles HIGH and LOW. Consider the following automata:

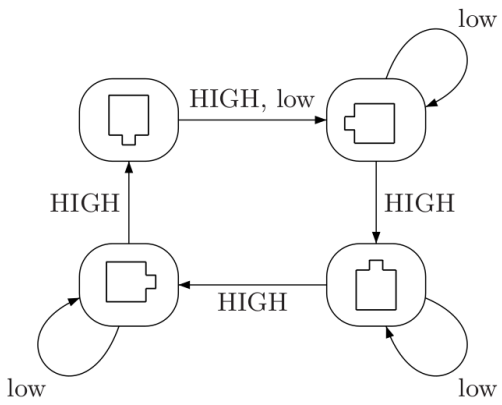


Figure 1.2: The action of the obstacles

Now, this automata is much similar to Cerny automata of four states (discussed in chapter 2) and hence gives us the desired sensor-free orienter. Further possible applications of synchronizing automata, one can think of bio-computing. Experiments have been carried out in which DNA molecules have been used as both hardware and software for finite automata of nanoscaling size.

1.2 Synchronizing automata at a glance

1.2.1 Definitions

Definition 1.2.1. Deterministc Finite Automata: $\mathscr{A} = (Q, \Sigma, \delta)$ is a deterministic finite automaton (DFA), where:

- Q denotes the state set
- Σ stands for the input alphabet.
- $\delta : Q \times \Sigma \to Q$ is the transition function.

δ is the transition function defining an action of the letters in Σ on Q . This is then extended in a unique way to $\delta : Q \times \Sigma^* \to Q$ of the free monoid Σ^* over Σ.

Definition 1.2.2. Reset word: A word $w \in \Sigma^*$ such that $\delta(q, w) = \delta(q', w)$, $\forall q, q' \in Q$ for a DFA $\mathscr{A} = (Q, \Sigma, \delta)$ is called a reset word.

Definition 1.2.3. Power Automaton of a DFA: Given, DFA $\mathscr{A} = (Q, \Sigma, \delta)$, $\mathscr{P}(\mathscr{A}) = (Q', \Sigma', \delta')$ is power automaton of $\mathscr{A}$ iff:

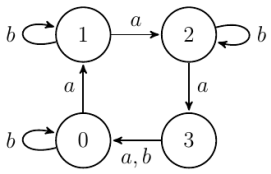
- $Q' = \mathscr{P}(Q) \setminus \emptyset$ where $\mathscr{P}(Q)$ is power set of Q.
- $\Sigma' = \Sigma$
- $\delta' : Q' \times \Sigma \to Q'$ such that $\delta'(S, a) = \bigcup_{q \in S} \delta(q, a)\ \forall S \in Q''$

Definition 1.2.4. Pair Automaton of a DFA: Given, DFA $\mathscr{A} = (Q, \Sigma, \delta)$, $\mathscr{P}^{[2]}(\mathscr{A}) = (Q'', \Sigma'', \delta'')$ is pair automaton of $\mathscr{A}$ iff:

- Q'' is the set of 2-element and 1-element subsets of Q.
- $\Sigma'' = \Sigma$
- $\delta'' : Q'' \times \Sigma'' \to Q''$ such that $\delta'(S, a) = \bigcup_{q \in S} \delta(q, a)\ \forall S \in Q''$

Definition 1.2.5. Synchronizing Automata: A DFA $\mathscr{A} = (Q, \Sigma, \delta)$ is called synchronizing if $\exists w \in \Sigma^*$ such that w is a reset word.

Example:



(Cerny Automata of 4 states.)

Reset word: ab^3ab^3a

1.2.2 Algorithms and complexity

Basic question is how to find an automata is synchronizing. We present two algorithms to detect whether an automata is synchronizing or not.

- **Algorithm 1**:

  1. Construct the power automaton from the given DFA.
  2. Run BFS from Q as a source .
  3. If we reach to a singleton set in this process then automata is synchronizing otherwise after completion of BFS all singleton sets are unvisited then it is not synchronizing.

Claim: The above algorithm takes DFA as an input and find whether its synchronizing or not. Moreover, it also gives the shortest reset word.

Correctness and Proof:

Proof. Note that in the constructed automaton with Q' states, each element of Q' is the node in the graph of power automaton.
State X(or node X) and Y(or node Y) have an edge with label 'a' iff
$\forall q \in X, \delta(q, a) \in Y$. Denote $X \to Y$ as edge from X to Y.
So,

$$X \to Y \iff \forall q \in X, \delta(q, a) \in Y \tag{1.1}$$

As, $|Q| < \infty \Rightarrow |Q'| = 2^{|Q|} - 1 < \infty$

So, there is path from X to Y iff $\exists U = \{u_1, u_2, u_3 \dots u_r\}$ set of edges each labelled with some letter $x_i \in \Sigma$. Now ,consider the corresponding word $w = x_1 x_2 \dots x_r \in \Sigma^*$.
Clearly, $\delta'(X, w) = Y$.
As,the previous algorithm finds whether Q is connected to any singleton set

or not.

Case1: Suppose, there exist such path and corresponding word w and q_{reset} be such state in singleton set.

$\therefore \delta'(Q, w) = q_{reset} \Rightarrow \forall q \in Q, \delta(q, w) = q_{reset}$ (from 1.1)

$\Rightarrow$ w is a reset word for automaton $\mathscr{A}$. $\Rightarrow \mathscr{A}$ is a Synchronizing automata.

Case2: Suppose there is no such path from Q to any singleton set.

$\Rightarrow \nexists$w$\in \Sigma^*$,such that $\delta'(Q, w) = q$

$\Rightarrow \mathscr{A}$ is not synchronizing. □

Before, heading towards next algorithm lets look at an important result.(from [9])

Theorem 1.2.6. *A DFA $\mathscr{A} = (Q, \Sigma, \delta)$ is synchronizing iff $\forall q, q' \in Q, \exists w \in \Sigma^*$ such that $\delta(q, w) = \delta(q', w)$.*

Proof. '$\Leftarrow$' part follows from the definition.

Let T be the maximal subset of Q such that $\exists w \in \Sigma^*, \forall t, t' \in T, \delta(t, w) = \delta(t', w)$.

If $T = Q$ then done.

Suppose $T \neq Q$,

$\Rightarrow \exists s \in Q \setminus T$ such that $\delta(s, w) \neq \delta(t, w)$ for some $t \in T$.

However,$\exists w' \in \Sigma^*$ such that $\delta(s, ww') = \delta(t, ww')$

(follows from the fact that $\forall t, t' \in Q, \exists w \in \Sigma^*$ such that $\delta(t, w) = \delta(t', w)$)

So,ww' is a word for states s,t such that $\delta(s, ww') = \delta(t, ww')$ which contradicts the fact that T is maximal.

$\Rightarrow Q = T$

$\Rightarrow \mathscr{A}$ is synchronizing.

□

- **Algorithm 2:**

    1. Construct the pair automaton from the given DFA.
    2. Pick a 2-element unvisited node and run BFS from it until we get a singleton node. Mark the visited ones.
    3. From the set of 2-element unvisited nodes, pick another unvisited node and run BFS till we reach singleton node and continue till all 2-element nodes have been visited.
    4. At the end of each run if we reach to a singleton node then automata is synchronizing otherwise it is not synchronizing.

Claim: The above algorithm takes DFA as an input and find whether its synchronizing or not.

Correctness and Proof:

Proof. Considering the pair automaton $\mathscr{P}^{[2]}$ from definition 1.2.4 for DFA $\mathscr{A}$, we apply BFS from the nodes consisting of 2-element(say q, q') states, we actually find a path to a singleton state node. If we find such a path,
$\Rightarrow \delta(q, w) = \delta(q', w)$ where w is generated using argument in previous proof. From 1.2.6, clearly $\mathscr{A}$ will be synchronizing. If there is no such path,then it won't be synchronizing. □

Analysis of above algorithms:

Algorithm 1 involves generation of power automaton which itself takes exponential time. $T(Q) = O(2^{|Q|}.|\Sigma|)$.
Algorithm 2 on the other hand takes polynomial time as it deals with $\binom{|Q|}{2}$ $+|Q| = \frac{|Q|(|Q|+1)}{2} = O(|Q|^2)$ nunmber of states. Therefore time for BFS required $O(|Q|^2.|\Sigma|)$.
The previous algorithm however calculates the shortest reset word but the

latter one does not give us the reset word. If one requires that, whenever $\mathscr{A}$ turns out to be synchronizing, a reset word is produced, then the best of the known algorithms (which is due to ([4]Eppstein, 1990, Theorem 6),([9]) see also (Sandberg, 2005, Theorem 1.15 [7])) has an implementation that consumes $O(|Q|^3 + |Q|^2|\Sigma|)$ time and $O(|Q|^2 + |Q||\Sigma|)$ working space, not counting the space for the output which is $O(|Q|^3)$.

1.3 Complexity of computing reset words

Let us look at the two basic problems arise here:

Short Reset Word: Given a synchronizing automaton $\mathscr{A}$ and a positive integer l, is it true that $\mathscr{A}$ has a reset word of length l?

Shortest Reset Word: Given a synchronizing automaton $\mathscr{A}$ and a positive integer l, is it true that the minimum length of a reset word for $\mathscr{A}$ is equal to l?

Note: Here `Short Reset Word` belongs to complexity class NP as one can non-deterministically take a word $w \in \Sigma^*$ of length l and check if w is a reset word of $\mathscr{A}$ in time $l|Q|$ as we start from each state $q \in Q$ and apply this certificate of length l and check whether we attain the same reset state.

Moreover, this problem was proved NP-complete by Eppstein (1990)[4] as this problem is obtained by reduction from 3-SAT problem.

Sketch of the proof:

Consider a satisfiability problem as a Boolean formula in conjunctive normal form, with m variables $x_1, x_2 \dots x_m$, and with n clauses. The automaton we construct will be such that we need only two transition functions a and b (The ones we consider for the synchronizing automata) and any word of length $m + 1$ is reset word, but any reset sequence of length m or less will

correspond to a satisfying assignment.

The assignment is constructed as follows:

$x_j = true \iff j^{th}$ input symbol $= a$

$x_j = false \iff j^{th}$ input symbol $= b$

Hence, for a satisfiable expression we also get a reset word and vice-versa. The automaton will have one special state r, and $mn + m$ other states:

$s_{i,j}$ for $1 \leq i \leq n$ and $1 \leq j \leq m + 1$.

$\delta(s_{i,m+1}, a) = r$, $\forall i$

$\delta(s_{i,m+1}, b) = r$,

$\delta(r, x) = r$ for $x \in a, b$.

If the i^{th} clause of the formula contains x_j , a will take state $s_{i,j}$ to r; if that clause contains $\overline{x_j}$, b will take $s_{i,j}$ to r. We call these transitions to r from states other than $s_{i,m+1}$ shortcuts. All remaining transitions take $s_{i,j}$ to $s_{i,j+1}$.

It can be seen that, as we stated above, any input sequence will take all states to r in at most $m + 1$ steps. Further, all states except $s_{i,1}$ will always be taken to r in m steps, so we need only concern ourselves with the former states. If a reset sequence of length m or less exists, then each of these initial states $s_{i,1}$ must be taken by that sequence using shortcut transition, because otherwise an initial state $s_{i,1}$ would progress through all the states $s_{i,j}$ before reaching r, and that would take $m + 1$ steps.

The variable assignment computed from the reset sequence must have a true variable in each clause, corresponding to the shortcut taken by the initial state corresponding to that clause. Thus we see that a satisfying assignment to the formula can be derived from a short reset sequence.

Conversely, if an assignment satisfies the formula, the derived input sequence would cause each initial state $s_{i,1}$ to take a shortcut corresponding to

the first true variable in the corresponding clause, and we would have a reset sequence of length m. Thus we see that the formula will be satisfiable if and only if the derived automaton has a short reset sequence.

Chapter 2

The Černý Conjecture

2.1 Černý conjecture

Černý function (C(n)) gives the maximum length of shortest reset words for synchronizing automata with n states.

Conjecture 2.1.1. *The Černý conjecture (Černý, 1964) states that each n-state synchronizing automata, C(n)* $\leq (n-1)^2$.

Černý further proposed an infinite series of automata, $\mathscr{C}_n$, known as the Černý automata, for which, the shortest reset word meets the Černý bound, $(n-1)^2$. This automata is characterized as shown in Fig.2.1.

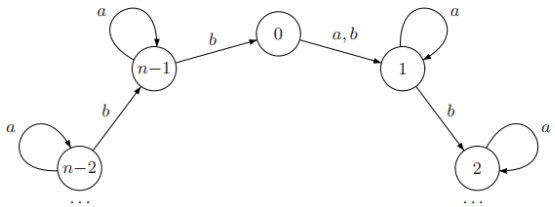


Figure 2.1: The automaton $\mathscr{C}_n$ with the shortest reset word $w = (ab^{n-1})^{n-2}a$

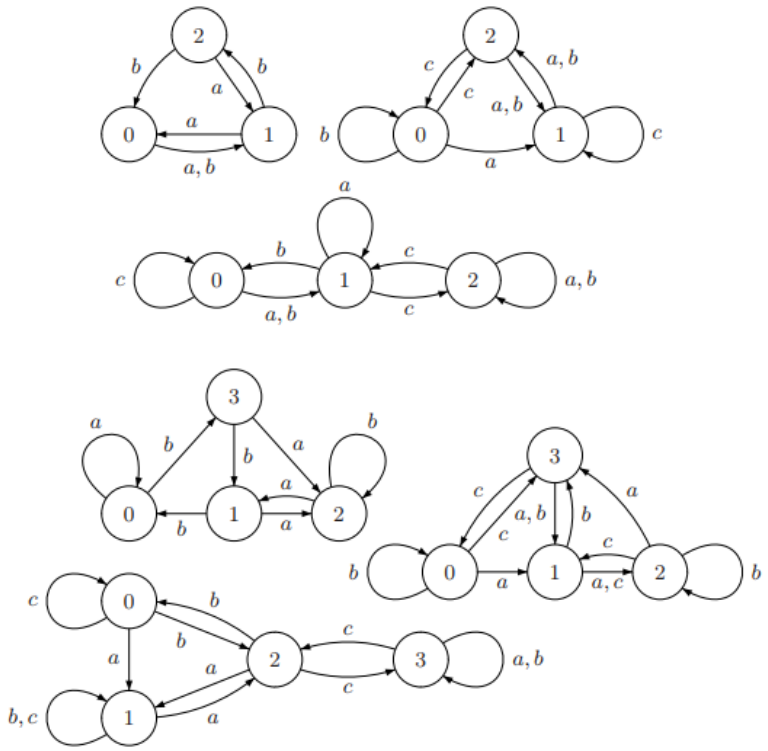


Figure 2.2: Extreme automata with 3 and 4 states

Apart from the $\mathscr{C}_n$ series, there are only 8 other known *extreme* automata. In fact, the $\mathscr{C}_n$ series is the only known infinite extreme automata series. There are three extreme automata with 3 states, three extreme automata with 4 states (see Fig.2.2), one extreme automaton with 5 states recently found by Roman, see Fig.2.3, and one extreme automaton with 6 states found by Kari (2001), see Fig.2.4.

2.2 A greedy algorithm

The best upper bound for the Cerny function $C(n)$ achieved so far guarantees that for every synchronizing automaton with n states there exists a reset

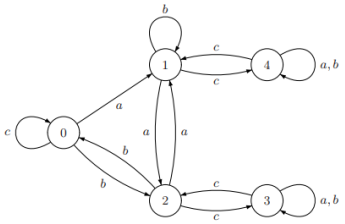


Figure 2.3: Roman automata

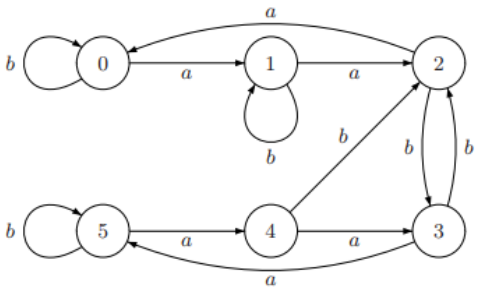


Figure 2.4: Kari automata

word $\frac{n^3-n}{6}$. Let's look at a greedy algorithm to obtain a reset word in this bound.(see [9])

Algorithm :

1. Take DFA $\mathscr{A}$ as input.
2. Let $w = \varepsilon$ i.e an empty word. Assign $P = Q$
3. Find a word $v \in \Sigma^*$ of minimum length until either $|P| \leq 1$ or $|\delta(P, v)| < |P|$ and replace w as wv and P as $\delta(P, v)$. If $\nexists v \in \Sigma^*$ it does not give any reset word.
4. Finally return the word w if obtained.

Chapter 3

Switchover Automata

In this chapter, we introduce the notion of a switchover automaton of a given automaton. Further, we use this concept to study the synchronization of binary automata with one cyclic permutation and one non-permutation.

3.1 Definition

Definition 3.1.1. Given an automata $\mathscr{A}$, a switchover of $\mathscr{A}$ is obtained by changing some transitions in $\mathscr{A}$.

Definition 3.1.2. Given an automata $\mathscr{A}$, a k-switchover of $\mathscr{A}$ is obtained by changing precisely k transitions of $\mathscr{A}$.

Consider an automaton $\mathscr{A} = (Q, \Sigma, \delta)$. Let $|Q| = n$, then there are a total of $n|\Sigma|$ transitions that can be precisely "Switched-over". Therefore, the summation of k-switchover Automata $0 \leq k \leq n|\Sigma|$ of $\mathscr{A}$ covers the complete set of all possible automata.

3.2 Binary circular automaton

An automaton over two-letter alphabet i.e., $|\Sigma| = 2$ further referred to as $\Sigma = \{a, b\}$ and one of the letter is a cyclic permutation i.e., $\delta(k, a) = (k+1) \pmod{n}$ $\forall k \in \{0, 1...(n-1)\}$ is called a binary circular automaton.

We focus on the problem of whether or not given a binary circular automaton is synchronizing.

Some existing results under this class of automata include:

Theorem 3.2.1. *A binary automata with a prime length cyclic permutation and a non-permuation is always synchromizing and follows the Černý Conjecture[6].*

This result was first posed and proven by Pin[6] and a generalization of the result was further proven by Dubac[3].

To address the problem, we use the concept of switchover automaton. We make the switchover transitions to the alphabet 'b' alone.

Consider the notation $\mathscr{A}_{n\_I\_J}$ where,

- $I = (i_1, i_2...i_k)$: k-distinct tuple where $i_m \in Q$ $\forall m \in \{1, 2...k\}$
- $J = (j_1, j_2...j_k)$: k-tuple where $j_m \in Q$ $\forall m \in \{1, 2...k\}$

The sets I and J stand for the changes $\delta(i_m, b) = j_m \forall m \in \{1, 2...k\}$ made in a given automata $\mathscr{A} = (Q, \Sigma, \delta)$ and $Q = \{0, 1, 2, ..., n-1\}$.

Consider an automaton $\mathscr{A}$ with $n = 6$ and $Q = 0, 1, 2, 3, 4, 5$ as shown in figure 3.1

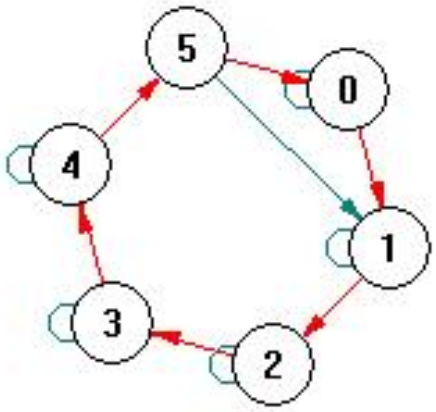


Figure 3.1: Base Automata $\mathscr{A}$

Here, *red* edges stand for the a cyclic transition $\delta(k, a) = (k+1) \pmod{n}$ and the *blue* edges stand for the b transitions.

For the given base automata, b transitions are defined as :

$\delta(0, b) = 0$

$\delta(1, b) = 1$

$\delta(2, b) = 2$

$\delta(3, b) = 3$

$\delta(4, b) = 4$

$\delta(5, b) = 1$

Now consider 1-switchover i.e. $k = 1$, we write this class as $\mathscr{A}_{n\_I\_J} = \mathscr{A}_{n\_(i_1)\_(j_1)}$. To make a "switchover" one has to make exactly one change in the transition table over the alphabet b.

The number of possible ways one can choose i_1 is n and for every chosen i_1 there are exactly $n - 1$ choices for j_1 to make a 1-switchover to the base automata.

For example, consider figure 3.2. In this figure, the 1-switchover made is $\delta(5, b) = 2$ from $\delta(5, b) = 1$, thus giving $i_1 = 5$ and $j_1 = 2$.

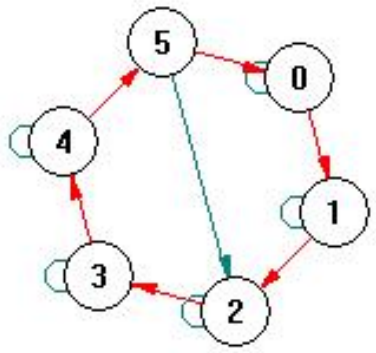


Figure 3.2: $\mathscr{A}_{6\_(5)\_(2)}$

Inductively, for 2-switchover or when $k = 2$, we chose a distinct 2-tuple of transitions to change and for each transition, j_m for $m = 1, 2$, there are exactly $n - 1$ choices.

For example, consider figure 3.3. In this figure, the 2-switchover made is $\delta(5, b) = 2$ from $\delta(5, b) = 1$ and $\delta(5, b) = 2$ from $\delta(5, b) = 1$ thus giving $i_1 = 5$, $j_1 = 2$, $i_2 = 4$, $j_2 = 1$.

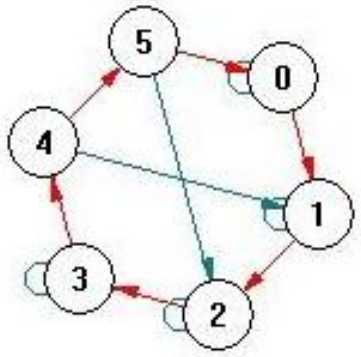


Figure 3.3: $\mathscr{A}_{6\_(5,4)\_(2,1)}$

k-switchover is thus bounded by $1 \leq k \leq n$. For each k, there are nC_k ways to chose I and for each i_m in $m = 1, 2, ...k$ there are $n-1$ choices for the corresponding j_m. Therefore, number of k-switchover are $nC_k * (n-1)^k$.
$\Rightarrow \sum_{k=1}^{n} nC_k * (n-1)^k + 1$
$\Rightarrow (1 + (n-1))^n - 1 + 1$
$\Rightarrow n^n$

Therefore, the summation of k-switchover for $1 \leq k \leq n$ along with the base automata cover an important subclass of automata over 2-alphabet where one letter is a cyclic permutation.

3.3 1-Switchover of $\mathscr{C}_n$

We study 1-Switchover Automata of $\mathscr{C}_n$ and identify various synchronizing automata, non-synchronizing automata and certain empirical results using TESTAS [8].

3.3.1 Synchronizing automata

Based on the experiments, we get $\mathscr{C}_{n\_n-1\_1}$ for n odd, $\mathscr{C}_{n\_1\_0}$ for all n, $\mathscr{C}_{n\_1\_n-1}$ for all n as synchronizing.

Theorem 3.3.1. *The infinite series* $\mathscr{C}_{n\_n-1\_1}$ $\forall n \in \{2k-1 : k \in \mathbb{N}\}$ *is always synchronizing with the the reset word* $w = ((ab^{n-2})^{n-2}a)$ *with* $|w| = n^2 - 3n + 3$*, thus satisfying Cerny conjecture.*

Proof. Let $\mathscr{A} = \mathscr{C}_{n\_n-1\_1} = (Q, \Sigma, \delta)$

Let $w = (ba^{n-2})^{n-2}b$ and $w_1 = ba^{n-2}$.

Therefore,$w = (w_1)^{n-2}b$.

We show that w which is of length $(n-1)(n-2)+1$ length is synchronizing word for $\mathscr{A}$.

Case 1: Let $\{2k\}$ be the starting state where $n = 2k+1$.

$\delta(\{2k\}, w_1) = \{2k\}$

Therefore, $\delta(2k, (w_1)^{n-2}) = \{2k\}$

$\delta(\{2k\}, b) = \{1\}$

$\Rightarrow \delta(2k, w) = \{1\}$

Case 2: Let $\{1\}$ be the starting state.

$w = b(a^{n-2}b)^{n-2}$

$\delta(\{1\}, b) = \{1\}$

$\delta(\{1\}, a^{n-2}b) = \{1\}$

$\Rightarrow \delta(\{1\}, a^{n-2}b) = \{1\}$

Case 3: Let the starting state be $2l+1$ such that $l \leq k-1$

Therefore, $2l + 1 \leq 2k - 1 = n - 2$

$\Rightarrow \leq k - 1$

Here, $k = \frac{n-1}{2}$.

Now, $\delta(q_i, w_1) = q_{(i-2)modn}$ [for $i \in \{3, 5, 7, \ldots 2t+1\, t \in \mathscr{N}\}$]

So, $\delta^l(\{2l+1\}, w_1) = \{1\}$

Also, since$l \leq k-1 \Rightarrow l \leq n-2 (\because l \leq k-1 = \frac{n-3}{2})$

Also,$\delta(\{1\}, b) = \{1\}$

Hence,$\delta(\{2l+1\}, (w_1)^{n-2}b) = \{1\}$

Case 4: Let the starting state be $2l+1$ such that $l \leq k-1$

$\Rightarrow l \leq \frac{n-3}{2}$

Now, $\delta^l(\{2l\}, w_1) = \{0\}$—i)

$\delta(0, w_1) = \{2k-1\}$—ii)

Repeating case 3 henceforth from state $2k-1 \Rightarrow$ in atmost $\frac{n-3}{2}$ steps we reach $\{1\}$ upon application of w_1—iii)

Combining i),ii),iii)

$\Rightarrow \delta^{l+1+\frac{n-3}{2}}(\{2l+1\}, w_1) = \{1\}$ and $\delta(\{1\}, b) = \{1\}$

Now, $l+1+\frac{n-3}{2} \leq \frac{n-3}{2}+1+\frac{n-3}{2} = n-2$

Therefore,$\delta(\{2l\}, (w_1)^{n-2}b) = \{1\} \Rightarrow \delta(\{2l\}, w) = \{1\}$

Hence, from case 1,2,3,4 we get $\mathscr{A}$ as synchronizing.

Hence proved.

□

Theorem 3.3.2. *$\mathscr{C}_{n\_1\_0}$ $\forall n$ is synchronizing with w as a reset word where $w = (ba)^{n-3}b$ satisfying Cerny conjecture.*

Proof. Let $\mathscr{A} = \mathscr{C}_{n\_1\_0}$ and $w = (ba)^{n-3}b$.

Case 1: Let $\{0\}$ be the starting state.

$\delta(\{0\}, b) = \{0\}$.

$\delta(\{0\}, (ab)^{n-3}) = \{0\}$

$\delta(\{0\}, w) = \{0\}$

Case 2: Let $\{1\}$ be the starting state.

$\delta(\{1\}, (ba)^{n-3}) = \{1\}$

$\delta(\{1\}, b) = \{0\}$.

$\delta(\{1\}, w) = \{0\}$

Case 3: Let $\{i\}$ starting state where i is not 0 and not 1.

$\delta(i, (ba)^{n-i}) = \{0\}$.Repeat the case 1 after that.

From case 1,2,3 we get $\mathscr{A}$ as synchronizing.

Hence proved. □

Theorem 3.3.3. *$\mathscr{C}_{n\_1\_n-1}$ $\forall n$ is synchronizing with w as a reset word where $w = b(ab)^{n-3}b$ satisfying Cerny conjecture.*

Proof. Let $\mathscr{A} = \mathscr{C}_{n\_1\_n-1}$ and $w = b(ab)^{n-3}b$.

Case 1: Let $\{n-1\}$ be the starting state.

$\delta(\{n-1\}, b) = \{0\}$.

$\delta(\{n-1\}, (ab)) = \{0\}$

$\delta(\{0\}, ab) = \{n-1\}$

$\delta(\{0\}, b) = \{0\}$

Therefore,$\delta(\{n-1\}, w) = \{0\}$

Case 2: Let $\{0\}$ be the starting state.

$\delta(\{0\}, b) = \{0\}$

$\delta(\{0\}, ab) = \{n-1\}$

$\delta(\{n-1\}, b) = \{0\}$.

$\delta(\{n-1\}, (ab)) = \{0\}$

Therefore, $\delta(\{0\}, w) = \{0\}$

Case 3: Let $\{i\}$ starting state where i is not 0 and not n-1.

Applying, *ab* (n-i) times we end up at $\{0\}$ or $\{n-1\}$.Then, repeat case 1 or 2 .

From case 1,2,3 we get $\mathscr{A}$ as synchronizing.

Hence proved. □

3.3.2 Non-synchronizing automata

Based on the experiments, we get $\mathscr{C}_{n\_n/2-1\_n/2}$ for n even, $\mathscr{C}_{n\_n-1\_k}$ for n even and k odd, $\mathscr{C}_{n\_n-1\_n/2}$ for n even as non-synchronizing.

Theorem 3.3.4. *The infinite series* $\mathscr{C}_{n\_n/2-1\_n/2}$ $\forall n \in \{2k : k \in \mathbb{N}\}$ *is not synchronizing.*

Proof. Let $\mathscr{A}$ be $\mathscr{C}_{n\_n/2-1\_n/2}$ and consider $\mathscr{P}(\mathscr{A}) = (Q, \Sigma, \delta)$ is power automaton of $\mathscr{A}$ with $\Sigma = \{a, b\}$.

Define level function $\mathscr{L}$ of power automaton as follows:
$\mathscr{L}(j) = \{q : |q| = j, j \in \mathbb{N}, j \leq n, q \in Q\}$
Here, $|q|$ gives the size of the state of the power automaton.

Suppose, $\mathscr{P}((A))^{rev}$ is the reverse automaton of $\mathscr{P}(\mathscr{A})$ i.e
$\mathscr{P}((A))^{rev} = (Q, \Sigma, \delta')$ where,
$\delta' : Q \times \Sigma \to Q$,
$\delta'(p, a) = q \Leftrightarrow \delta'(q, a) = p \ \forall a \in \Sigma$

Note that if $\mathscr{A}$ is synchronizing then there exists a path P in $\mathscr{P}((A))^{rev}$ from $\{i\}$ to $\{0, 1, 2 \ldots n-1\}$ for some $i \in \{0, 1, 2, \ldots n-1\}$.

While continuing the proof, we present the following remarks.

Remark 3.3.5. If $\{n-1\}$ or $\{n/2-1\} \in q$, $q \in Q$, then $\delta'(q, b)$ does not exist.

Remark 3.3.6. In order to move from $\mathscr{L}(k)$ to $\mathscr{L}(m)$ such that $k \leq m$, a 'b' reverse transition is necessary.

Remark 3.3.7. The following are the properties of $p, q \in Q$ such that $\delta'(q, b) = p$:

- if $\{0\} \in q \Rightarrow p = q \cup \{n-1\}$
- if $\{n/2\} \in q \Rightarrow p = q \cup \{n/2-1\}$
- if $\{0, n/2\} \in q \Rightarrow p = q \cup \{n-1, n/2-1\}$

- if neither $\{0\}$ nor $\{n/2\} \in q$, $\{n-1\} \notin q$ and $\{n/2-1\} \notin q$, then $p = q$

Remark 3.3.8. For $\delta'(q, a) = p$, Let $q = \{q_1, q_2...q_k\}$, then $p = \{(q_1 + n - 1) \pmod{n}, (q_2 + n - 1) \pmod{n}, ..., (q_k + n - 1) \pmod{n}\}$

Consider the set $\mathscr{L}(1)$. From 3.3.8, we have

$\delta'(\{0\}, a) = \{n - 1\}$

$\delta'(\{1\}, a) = \{0\}$

$\delta'(\{2\}, a) = \{1\}$

...

$\delta'(\{n - 1\}, a) = \{n - 2\}$

Note that a 'n' length cycle is formed here.

Consider the word $w = ba^{n-1}$ applied on the state $\{0\}$ in $\mathscr{L}(1)$.

$\delta'(\{0\}, w) = \{0, 1\}$

Applying w to the resultant state gives

$\delta'(\{0, 1\}, w) = \{0, 1, 2\}$

i.e. : $\delta'(\{0\}, w^2) = \{0, 1, 2\}$

By induction $\delta'(\{0\}, w^{n/2-1}) = \{0, 1, 2..., n/2 - 1\}$

Note that $\forall k \leq n/2$, the state $q = \{0, 1, 2..., k - 1\}$ in $\mathscr{L}(k)$ is reachable from $\mathscr{L}(1)$. Also, in $\mathscr{L}(n/2)$, the state $\{0, 1, 2..., n/2 - 1\}$ no longer satisfies 3.3.5. Therefore a 'b' transition in the reverse power automaton is not possible.

At every Level $\mathscr{L}(k)$ $\forall k \leq n/2$, an 'n' length cycle is formed due to 'a' reverse transitions, i.e,

$\delta'(\{0, 1, 2, ..., k - 1\}, a) = \{n - 1, 0, 1, ..., k - 2\}$

$\delta'(\{n - 1, 0, 1, ..., k - 2\}, a) = \{n - 2, n - 1, 0, ...k - 3\}$

$\delta'(\{n - 2, n - 1, 0, ..., k - 3\}, a) = \{n - 3, n - 2, ..., k - 4\}$

...

$\delta'(\{1,2,...,k\},a) = \{0,1,2,...k-1\}$

For Level $\mathscr{L}(k)$ $\forall k < n/2$, the only states satisfying 3.3.5 and 3.3.7 such that $\delta'(q,b) = p$ and $|p| > |q|$ are of the form:
$\delta'(\{0,1,2,...,k-1\},b) = \{n-1,0,1,2,...,k-1\}$
$\delta'(\{n/2,\ldots,n/2+(k-1)\},b) = \{n/2-1,n/2,\ldots n/2+(k-1)\}$
For all other states, either $\delta'(q,b) = p$ and $p = q$ (i.e., a self-loop occurs) or the state doesn't satisfy 3.3.5. Either way, a transition to a higher level does not occur.

For Level $\mathscr{L}(n/2)$, all the n states of the 'n' length cycle either contain $\{n-1\}$ or $\{n/2-1\}$ and Hence from 3.3.5, $\mathscr{L}(m)$ $\forall m > n/2$ are not reachable from the cycle in $\mathscr{L}(n/2)$. Thus there does not exist a path P from $\mathscr{L}(1)$ to $\mathscr{L}(n/2)$ in $\mathscr{P}((A))^{rev}$ and $\mathscr{C}_{n\_n/2-1\_n/2}$ is not synchronizing. □

Theorem 3.3.9. *The infinite series* $\mathscr{C}_{n\_n-1\_k}$ $\forall n \in \{2t : t \in \mathbb{N}\}$ *and* $\{k|k \in \{2t-1 : t \in \mathbb{N}\}\}$ *is not synchronizing.*

Proof. Let $\mathscr{A}$ be $\mathscr{C}_{n\_n-1\_i}$ where i is odd and consider $\mathscr{P}(\mathscr{A}) = (Q,\Sigma,\delta)$ is power automaton of $\mathscr{A}$.
Define level function $\mathscr{L}$ of power automaton as follows: $\mathscr{L}(j) = \{q : |q| = j, j \in \mathbb{N}, j \leq n, q \in Q\}$
Here, $|q|$ gives the size of the state of the power automaton.
Suppose, $\mathscr{P}((A))^{rev}$ is the reverse automaton of $\mathscr{P}(\mathscr{A})$ i.e $\mathscr{P}((A))^{rev} = (Q,\Sigma,\delta')$ where,
$\delta' : Q \times \Sigma \to Q$,
$\delta'(p,a) = q \Leftrightarrow \delta'(q,a) = p$ $\forall a \in \Sigma$

If $\mathscr{A}$ is synchronizing then there exists a path P from $\{i\}$ to $\{0,1,2\ldots n-1\}$ for some $i \in \{0,1,2,\ldots n-1\}$.
Consider the set $\mathscr{L}(1)$.

Note: a is a cyclic permutation on the set while transition b loops every state to itself except $\{i\}$.

So, from $\mathscr{L}(1)$ only way to go up to $\mathscr{L}(2)$ is from $\{i\}$. $\Rightarrow$ P contains an edge from $\{i\}$ to $\{i, n-1\}$.

Note: If i and $n-1$ are in same state then b transition is not defined. Let $n = 2l$. We show the further steps for $i = 1$ which are similar for any i odd.

Now consider the state $\{1, 3, \ldots 2k-1, n-1\}$ at level $(k+1)$

Note: $\delta'(\{1, 3, \ldots 2k-1, n-1\}, (aa)^{l-1})$ $=1, 3, \ldots 2k+1 \forall k \in \{1, 2 \ldots l-1\}$ and, $\delta'(\{1, 3, \ldots 2k+1\}, aa)$=$\{1, 3, \ldots 2k-1, n-1\}$.

Therefore, $\{1, 3, \ldots 2k+1\}$ and $\{1, 3, \ldots 2k-1, n-1\}$ lie on a cycle $\forall k \in \{1, 2, \ldots l-1\}$.

Similarly, for the same word for others i we will get a cycle.

Hence, $\delta'(\{i\}, (b(aa)^{l-1})^k) = \{1, 3, \ldots 2k+1\}$.

Let us call this path X.

Note: For $k < l$, X is the unique path from i to level k i.e to $\{i, i+2, \ldots\}$

Proof. Suppose, $\exists X'$ path such that $X' \neq X$. Let t be the first state in X' that is not in path X. Let s precedes t in path X' which is also in X. Also, there will be always such an s as to start with we have $\{i\}$ as common to X and X'.

Case 1: If s and t lie on the same levels.

$\Rightarrow \exists y \in \mathbb{N}$ such that $s \in \mathscr{L}(y)$ and $t \in \mathscr{L}(y)$ So, t must lie on a cycle containing $\{1, 3, \ldots 2y-3, n-1\}$ and $\{1, 3, \ldots 2y-3, 2y-1\}$ as the cycle is the connected component among level y states.

Consider the cycle C at level y. Now, t lie on C but not on X

$\Rightarrow t$ lie on sequence $q_0, q_1 \ldots q_n$ where $t \neq q_0$ and $t \neq q_n$ and $\forall i \in \{1, 2, \ldots n-1\}, q_i$ is not in path X and q_n is first state of path X in level y and q_0 is the last state of path X in level y.

But, as path is a sequence of vertices which are all distinct, we cannot get any state on X' after q_0 as there will be repetition. Hence, s and t lie on same path.

Case 2: If s and t lie on different levels. If $s \in \mathscr{L}(y)$ and s lie on a cycle containing $\{1, 3, \ldots 2y-3, n-1\}$ and $\{1, 3 \ldots 2y-1\}$.

since, s precedes t and only way to go out from cycle is from $\{1, 3, \ldots 2y-1\}$.

Therefore, $s = 1, 3, \ldots 2y-1$ and $t = 1, 3, \ldots 2y-1, n-1 \Rightarrow t$ lies on X.

Hence, $X = X'$.

So, the path is unique. □

Now, $\delta'(\{i\}, (b(aa)^{l-1})^{l-1}b) = \{1, 3, \ldots n-1\}$

Also, $\delta'(\{1, 3, \ldots n-1\}, a) = \{0, 2 \ldots n-2\}$ $\delta'(\{0, 2, \ldots n-2\}, a) = \{1, 3, \ldots n-1\}$ and b transition is not defined on $\{1, 3, \ldots n-1\}$

Note that $\{1, 3 \ldots n-1\} \in \mathscr{L}(n/2) = \mathscr{L}(l)$ and we obtain 2 cycle here as shown above. Hence, we cannot go beyond level l.

So, from $\{i\}$ to level l, we got a unique path upwards and we cannot go to level $(l+1)$ or beyond that, as at a time we can go one level up.

since,$\delta'(q, b) = q_1$ where $\{i, n-1\} \nsubseteq q$ and $\{i\} \subset q$ and $q_1 = \{n-1\} \cup q$.

Hence, we cannot go to level $(l+1)$ onward.

$\Rightarrow$ We cannot reach $\{0, 1, \ldots n-1\}$ from $\{i\}$.

$\Rightarrow \mathscr{A}$ isn't synchronizing.

Hence proved. □

Theorem 3.3.10. *The infinite series $\mathscr{C}_{n\_n-1\_n/2-1}$ $\forall n \in \{2k : k \in \mathbb{N}\}$ is not synchronizing.*

Proof. Consider level 1 and 2 of power automaton of $\mathscr{A} = \mathscr{C}_{n_n-1_n/2-1}$with level defined in previous proof. If $\mathscr{A}$ is synchronizing then there is a path from $\{\frac{n}{2} - 1\}$ to $\{0, 1, \ldots n-1\}$. Consider possible paths from $\{\frac{n}{2} - 1\}$

upwards. At level 2 we are reachable to $\{n-1, \frac{n}{2}-1\}$.

But, here 'b' transition is not defined.

Note: We need $\frac{n}{2}-1$ in the state set in order to advance to higher level.

Note: $\delta^l(\{n-1, \frac{n}{2}-1\}, (a)^l) = \{n-1, \frac{n}{2}-1\}$ and $\nexists q$ in this cycle containing $\{\frac{n}{2}-1\}$.

$\Rightarrow$ There is no path to higher level outside this cycle. Also, note that this is the only path from level 1 to level 2 and we end up in this cycle.

$\Rightarrow \mathscr{A}$ is not synchronizing. □

3.3.3 Assertions

Based on experiments following results were observed but are not proved.

Assertion 3.3.11. For $n \in \{2k : k \in \mathbb{N}\}$, except $\mathscr{C}_{n\_0\_n-1}$ and $\mathscr{C}_{n\_n/2-1\_n/2}$ every non-synchronizing automata is of the form $\mathscr{C}_{n\_n-1\_i}$ for $i \in \{0, 1, 2, ..., n-1\}$. Further, if $n = 2^k, k \in \mathbb{N}$, then for all i odd we get non synchronizing automata.

Assertion 3.3.12. When $n \in \{2k-1 : k \in \mathbb{N}\}$, all possible non-synchronizing automata occur when $i = n-1$, if any.

Assertion 3.3.13. Except for the above mentioned non-synchronizing automata with $n \in \mathbb{N}$ states, all other automata obtained are synchronizing.

3.3.4 Conclusion

In this subsection, we summarize all the results on 1-switchover automata presented in this report. Excluding above assertions, the results are tabulated below.

Table 3.1: Observations for 1-switchover with base as Černý Automata

n	i_1	j_1	Comments
	0	...	
	1	...	
	...	...	
Odd	$n/2$	...	
	...	...	
	$n-1$	1	Synchronizing with $\|w\| = n^2 - 3n + 3$
	$n-1$	...	All possible non-synchronizing automata occur here
	0	...	
	1	...	
	...	...	
Even	$n/2-1$	$n/2$	Not Synchronizing
	...	...	
	$n-1$	$\{k \mid k \in Odd\}$	Not Synchronizing
	$n-1$	$n/2-1$	Not Synchronizing

3.4 Future work

We can extend the experiments to k-switchovers, for $k > 1$, in order to systematically approach the set of automata having one cyclic permutation and one non-permutation.

The concept can be further used to systematically study synchronization of different classes of automata by choosing an appropriate base automata.

Bibliography

[1] Dmitry Ananichev, Vladimir Gusev, and Mikhail Volkov. Slowly synchronizing automata and digraphs. In *Mathematical foundations of computer science 2010*, volume 6281 of *Lecture Notes in Comput. Sci.*, pages 55–65. Springer, Berlin, 2010.

[2] W. Ross Ashby. *An introduction to cybernetics*. Chapman and Hall, Ltd., London, 1956.

[3] L. Dubuc. Sur les automates circulaires et la conjecture de černý. *RAIRO Inform. Théor. Appl.*, 32(1-3):21–34, 1998.

[4] David Eppstein. Reset sequences for monotonic automata. *SIAM J. Comput.*, 19(3):500–510, 1990.

[5] Andrzej Kisielewicz, Jakub Kowalski, and Marek Szykuł a. Experiments with synchronizing automata. In *Implementation and application of automata*, volume 9705 of *Lecture Notes in Comput. Sci.*, pages 176–188. Springer, [Cham], 2016.

[6] J.-E. Pin. Sur un cas particulier de la conjecture de Cerny. In *Automata, languages and programming (Fifth Internat. Colloq., Udine, 1978)*, volume 62 of *Lecture Notes in Comput. Sci.*, pages 345–352. Springer, Berlin-New York, 1978.

[7] Sandberg S. Homing and synchronizing sequences. in: Broy m., jonsson b., katoen jp., leucker m., pretschner a. (eds) model-based testing of reactive systems. lecture notes in computer science, vol 3472. springer, berlin, heidelberg. *SIAM J. Comput.*, 19:500–510, 2005.

[8] Trahtman. Testas package. `http://u.cs.biu.ac.il/~trakht/readme.html`.

[9] Mikhail V. Volkov. Synchronizing automata and the černý conjecture. In *Language and automata theory and applications*, volume 5196 of *Lecture Notes in Comput. Sci.*, pages 11–27. Springer, Berlin, 2008.