On Sequences, Languages, and Automata*

[Preliminary Version]

Henry N. Adorna †
Department of Computer Science
(Algorithms and Complexity)
Velasquez St., UPDCampus, Diliman 1101
Quezon City
henri@csp.org.ph

ABSTRACT

Automata theory is one of the oldest and most simplest computing model investigated in theoretical computer science. Words and languages are the basic building blocks in studying automata theory. For the last ten years or so, efforts to put automata theory in the main stream of computer science research has been the goal of most formal languages and automata theorists. There are conferences organized every year as venue to new results in automata and formal languages theory. These, indeed contribute significantly in placing automata theory approxiamately near where it was some half a century ago.

In this paper, we report on some results based on the work of the author and his collaborators on properties of some new finite automaton accepting regular languages. In particular, we consider questions about characterization and state complexities of this finite automaton. We also report on a new regular language preserving operation. A deterministic finite automaton with output (dfao) considered as another way of looking at cosets coloring of n-dimensional objects or patterns. Finally, we present a reasonable generalization of a so-called k-automatic sequence, which we call k-context-free sequence.

Keywords

Sequences, Languages, Automata, Cosets, Coloring

*This is supported in part by a Faculty Research Incentive Award (FRIA) from the Department of Computer Science, College of Engineering, UPDiliman and by an NSRI grant no. MAT 06-2-02 of UPDiliman.

†Henry N. Adorna, Ph.D. is an Associate Professor of Computer Science at UPDiliman, Q.C.

1. INTRODUCTION

Automata is one of the oldest and the simplest model of computation that is extensively investigated in the area of theoretical computer science. Words and languages are the basic building blocks in studying automata theory [11]. After more than five decades, automata theory significantly lost its prime position in the computer science research main stream. It was once argued that most efforts should be excerted to find (further) applications of the theory [25]. While it is indeed interesting to look for further applications of automata theory, there are still problems that need our attention [12, 1]. Most of the problems mentioned in [12, 1] are some of the remaining hard ones in the area. Hromkovič [12] and Adorna [1] mentioned that solving or giving insight(s) to the possible solutions to these problems will certainly put automata theory again in the main stream of computer science research.

Most of the issues mentioned in [12, 1] were on descriptional complexity. By descriptional complexity, we mean the study of measures of complexity of language and operations. It is a measure required to describe a language. Since it is well known that we can describe a language in many ways (e.g. via regular expression, finite automata, formal grammars), then there are many different measures of descriptional complexity. A descriptional complexity measure which considers the minimun number of states of a (deterministic) finite automaton (dfa) that describes a regular language is called *(deterministic) state complexity.* Other descriptional complexity for regular languages are the so-called *regular expression size* and *radius.* The *regular expression size* is the number of alphabetic symbols in a regular expressions, while *radius* is the distance from the start stae to the farthest state. Among these complexity measures, state complexity is the most popular for regular languages. A survey in different descriptional complexities is provided by Ellul in [7]. In this paper, we considered problems related to state complexity.

Automata do not only describe regular languages but also (infinite) sequences. As a computing model, it computes the nth term of an infinite sequence. Such automaton is called *deterministic finite automaton with output* (dfao). An infinite sequence that can be described by a dfao is called an *automatic sequence* [4]. An alternative characterization of an automatic sequence would be based on its so-called *fibers.*

An infinite sequence is automatic if and only if its fibers are regular languages [4]. An excellent source of automatic sequences and its applications is the book by Jean-Paul Allouche and Jeffrey Shallit [4]. A natural generalization of automatic sequence, which is so-called *context-free sequence* will be also considered.

The paper is organized as follows: In Section 2, we discuss some known results in the hierarchy of ambiguity of the class of nondeterministic finite automaton. We provided new class of automata in the class of nondeterministic finite automata **NFA**, which we call *Strict Tree Unambiguous Nondeterministic Finite Automata* **StUFA**. This class satisfies a certain property that must be satisfied by the autmaton that belongs in the class of unambiguous nondeterministic finite automata (**UFA**). We call such property Strict Tree Property of the computation tree. Section 3 provides discussion on a proposed regular operation on set of strings, which we called *commutative complement.* The regularity of the language will be preserved if and only if the commutative complement is *maximal.* We introduce a possible interplay of coloring symmetry groups with automata in Section 4. It is observed that the number of states for a so-called *coloring automaton* is equal to the number of cosets of G with respect to a subgroup S of G. Finally, in Section 5, we defined a so-called *k-context-free sequence,* which is basically base on the characterization of an automatic sequence via a so-called *fibers.*

In most of the discussions below, we provide some questions at the end of the section.

2. AMBIGUOUS COMPUTATION OF NFA

Nondeterministic finite automaton $\mathcal{M}$ computes or recognizes words in L in at least one way, where L is the language described by the nfa $\mathcal{M}$. We call this property *ambiguity.* Ambiguity in nfa's provides information on the number of ways the computing model accepts an input word. Degree of ambiguity is also an intensively investigated concept in automata theory. Measuring the degree of nondeterminism in finite automata had been considered in [15, 18, 19, 21, 24]. To understand the influence of the degree of ambiguity on the size of nfa's is the central question in this sub-area.

If an nfa $\mathcal{M}$ accepts every word in $L(\mathcal{M})$ in one and only one way or computation, then such an nfa is called *unambiguous nfa.* If it accepts words in k computation for some constant k, it is called *constantly ambiguous nfa.* If it accepts the words polynomially many times, then it is a *polynomially ambiguous nfa.* Bounding its asymtotic behavior provides us the following classes of nondeterministic finite automata. We denote by **NFA** the class of all nfa's and **UNFA** the class of unambiguous nfa, **CAFA** the class of constantly ambiguous nfa, **PAFA** the class of polynomially ambiguous nfa, and **DFA** the class of deterministic finite automata. The following is the ambiguity hierarchy of finite automata:

$$\mathbf{DFA} \subset \mathbf{UNFA} \subset \mathbf{CAFA} \subset \mathbf{PAFA} \subset \mathbf{NFA}.$$

For a given sequence of automata $(\mathcal{M}_i)_{i\geq 0}$, we define $|Q_{\mathcal{M}}|(i)$ as the number of states of $\mathcal{M}$. For a function f on the set of natural numbers with $f \notin O(n)$ and two classes X_1 and X_2 of finite automata,we will say that there is a *$f(n)$-size gap* or $f(n)$ *separation* between X_1 and X_2, denoted by

$$X_1 \prec_{f(n)} X_2,$$

if there is a sequence of regular languages $L_i \in (2^{\Sigma^*})^N$ for $i \geq 0$, such that there exists a sequence $(\mathcal{M}_i)_{i\geq 0} \in X_2^N$ with $L(\mathcal{M}_i) = L_i$ and $|Q_{\mathcal{M}}| \in O(i)$ and all sequences $(\mathcal{N}_i)_{i\geq 0} \in X_1^N$ with $L(\mathcal{N}_i) = L_i$ satisfy $|Q_{\mathcal{N}}| \in \Omega(f(i))$.

The following gaps between ambiguity classes have been established:

$$\mathbf{DFA} \prec_{2^n} \mathbf{UNFA} \prec_{2^n} \mathbf{CAFA}$$

and

$$\mathbf{PAFA} \prec_{2^n} \mathbf{NFA}.$$

The exponential gap between $s(L)$ and the size of the minimal unambiguous finite automata for some languages L is known since 1978 [22, 21, 24]. The exponential gap between **UNFA** and **CAFA** had been established in [21, 24]. Leung in [18] proved the exponential gap between **PAFA** and **NFA.**

It is still not known whether or not there exists an exponential gap between **CAFA** and **PAFA.** One may regard this as a central open question in the context of automata ambiguity. A partial solution to this problem has been provided in [15], where an exponential gap between the sizes of k-ambiguous nfa's and polynomially ambiguous nfa's was established for every k.

2.1 Computation Tree of UNFA

In [15], the following hypothesis on the computation tree of any unambiguous nondeterministic finite automata was given:

> **Strict Tree Property**
> *The computation tree of any minimal $\mathcal{M} \in$ **UNFA** on an input w has exactly one path P from root to a leaf with several nondeterministic guesses and all paths having only one vertex in common with P do not contain any nondeterministic branching.*

However, [3] gave counter example by showing a minimal unambiguous finite automaton that accepts a word wherein the computation tree disobeys the Strict Tree Property. In particular, the automaton $\mathcal{M}$ that would accept the following language,

$$L(\mathcal{M}) = \{0,1\}^*101 \cup \{0,1\}^*010 \cup \{0,1\}^*100.$$

where

$\mathcal{M} = (Q, \Sigma, \{q_0\}, \Delta, F) \in$ **UNFA** s.t.

$$\begin{array}{ll} Q = & \{q_0, q_1, q_2\} \\ \Sigma = & \{0,1\} \\ F = & \{q_5\} \\ \Delta = & \{(q_0,1,q_0),(q_0,1,q_1),(q_2,1,q_5),(q_3,1,q_4) \\ & (q_0,0,q_0),(q_0,0,q_3),(q_1,0,q_2),(q_1,0,q_4),(q_4,0,q_5)\}. \end{array}$$

This (strict tree) property on the computation tree of nondeterministic finite automata is found to be satisfied by some

automata which are different from finite automata in **DFA.** In [2], we introduced a class of nfa's between **DFA** and **UNFA**, and we call it **StUFA**[1] **StUFA** contains properly **DFA** and is properly contained in **UNFA.** Thus we have

$$\textbf{DFA} \subset \textbf{StUFA} \subset \textbf{UNFA}$$

Realized that **StUFA** is the class of automata that satisfy the strict tree property and as far as its asymptotic behavior, they belong to **UNFA.** The class **StUFA** is characterized in [3] by considering the computation paths of the automaton.

It is known that if $w \in L(\mathcal{M})$, for some automaton $\mathcal{M}$, then we can construct labelled paths in the transition diagram of $\mathcal{M}$ from the start state to an accepting state, where the label of these paths spells-out the word w. If $\mathcal{M}$ is in **UNFA,** then such a path must be unique. Note that we can also obtain computation paths that either end in a non-accepting state or never finished at all. If we take all these path together, we will produce a so-called computation tree of $\mathcal{M}$ on w.

DEFINITION 1. *Let $\mathcal{M} \in$ **UFA**. For all $w \in L(\mathcal{M})$.*

$$^1C_w \in T_{w;\mathcal{M}}$$

$$\textit{whenever } \exists\, v \in \mathit{Path}(\mathcal{M}) \cap I \cdot (\Sigma \cdot Q)^*, \chi_\Sigma(v) = w$$

$$\textit{and } \chi_Q(v) = {}^1C_w \textit{ is an accepting path.}$$

All other paths which leads to either non-accepting or unfinished computation of $\mathcal{M}$ on w will be denoted by ${}^iC_w$, for $i \geq 2$.

DEFINITION 2. *For all $w \in (\Sigma \cup Q)^*$, we define the set of all proper leading symbols of w as its set of proper prefixes and is denoted by* **Pref**(w). *Note that $\forall\, v \in$* **Pref**(w), $|v| \leq |w| - 1$.

The proof of following is provided in [3]:

THEOREM 1. *[3] Let $\mathcal{M} \in$ **UFA**. For all $w \in L(\mathcal{M})$,*

*$\mathcal{M} \in$ **StUFA** if and only if $\forall\, i, j \in N$, $i, j \geq 1, i \neq j$,*

$$\exists\, S \subset \textbf{Pref}\,({}^1C_w) \textit{ s.t. } \quad \textbf{Pref}\,({}^iC_w) \cap \textbf{Pref}\,({}^jC_w) - S = \emptyset.$$

2.2 State Complexity

As far as the state complexity results on these ambiguity hierarchy of finite automata, the exponential gaps between these classes seems inevitable although we showed that automata in **StUFA** are *almost deterministic* [2]. This can be attributed to the subset construction used in [20].

LEMMA 1. *[2, 3] There exists a regular language L_n accepted by some $\mathcal{M} \in$ **StUFA** such that no* **DFA** *can accept in less than $2^{|Q_\mathcal{M}|-1}$ states.*

[1]In [1, 2], we called this **SUFA.**

The language L_n, over the alphabet $\{0, 1\}$, which is the set of strings whose n^{th} term from the end is a 1 witnessed this gap. We simply use the subset construction to convert $\mathcal{N} \in$ **StUFA** to an $\mathcal{M} \in$ **DFA,** such that $L(\mathcal{N}) = L(\mathcal{M})$.

THEOREM 2. *[2, 3]*

$$\textbf{DFA} \prec_{2^n-1} \textbf{StUFA}.$$

We still do not know whether or not there is an exponential gap between **StUFA** and **UNFA.**

The example provided in [3] donot require any additional state to convert the automaton to follow the strict tree property.

3. REGULAR LANGUAGE OPERATION

In this section, we will be considering regularity preserving operation. In particular, we will define an operation introduced in [5].

Languages are sets of words. Thus set theoretic operations can be apply to these languages to obtain another language. The natural question to ask is whether the resulting language L after applying the operation $*$ to L_1 and L_2 belongs to the same class of languages $\mathcal{R}$, where L_1 and L_2 belong. That is, we ask: if $L_1, L_2 \in \mathcal{R}$, then is $L_1 * L_2 \in \mathcal{R}$?

It is well-known that the operations union, intersection, catenation and complementation are regular language preserving operations. That is, whenever any of these operations is applied to regular languages, the resulting language is again a regular language. Below we define new operation on languages.

3.1 Commutative Complement

DEFINITION 3. *Let L and K be languages over an alphabet Σ. We call K a commutative complement of L, denoted by L^{CC} if for all $u, w \in K$, $uw, wu \notin L$.*

If there is no other language $H \subseteq \Sigma^*$, such that H is a commutative complement of L that contains K, then K is called maximal commutative complement of L.

Note that any language can have at least one commutative complement language, that is the trivial language, $\emptyset$.

EXAMPLE 1. *Let $L = \Sigma^*$. Then the only commutative complement of this language is the trivial language, $\emptyset$. If $L = \emptyset$, then its commutative complement would be all $L \subseteq \Sigma^*$. But Σ^* is the maximal commutative complement.*

EXAMPLE 2. *Let L be a set of words over $\Sigma = \{a, b\}$ that ends in a. In particular, define $L = L((a+b)^*a)$. If we take the complement of L, that is $L^C = L((a+b)^*b + \epsilon)$, then we can verify that L^C is a commutative complement of L. And this commutative complement is maximal.*

Let $K' \subseteq K$, where K is commutative complement of L. It is trivial to see that K' is again a commutative complement of L. Because any catenation of any two elements of K' cannot be found in L.

REMARK 1. *Note here that regular language may not have a regular commutative complement. Take the instance when our language is the empty language, i.e. $\emptyset$. Its commutative complements are any $L \subseteq \Sigma^*$. Therefore may not be regular always, but Σ^*, the maximal commutative complement is certainly a regular language.*

In [5], we proved the following:

THEOREM 3. *[5] Every maximal commutative complement of regular language is regular.*

We do not have yet results on applying the operation commutative complement to a non-regular language.

3.2 State Complexity

In [5], a method of constructing the commutative complement of a regular language is provided. It is based on monoid transition. In particular, we state

THEOREM 4. *[5] If the minimun DFA accepting a regular language L has n states, then the maximal commutative complement K of L will be accepted by a DFA with at most n^n number of states.*

This result which follows from series of results in [5], provides the above naive upper bound. Note that this upper bound is the number of transformations in the transition monoid, which is $|Q|^{|Q|}$.

Although the construction in [5] characterizes some properties of maximal commutative complements, it is however inadequate in estimating the state complexity of a commutative complement. It is just too large an estimate. Hence we ask, is there a much tighter bound than n^n for the state complexity of maximal commutative complements? Or is n^n the best possible? If yes, is there a language L whose maximal commutative complement requires at least n^n states?

4. COLORING AUTOMATA

Since color symmetry uses certain procedures in order to color an n-dimensional object or pattern, it is possible to create an algorithm which defines the coloring. In this section, we report on the dfa's constructed for coloring the simplest type of colorings, namely coloring a regular n-gon using cosets[8]. Algebra terms used in this section follows the standard book in algebra such as [16].

DEFINITION 4. *A group is a nonempty set G together with a binary operation on G such that,*

• *the binary operation on G is associative, i.e. for all $a, b, c \in G$, $(a * b) * c = a * (b * c)$.*

• *there is an identity element e in G with respect to the binary operation, such that for all $a \in G$, $a * e = e * a = a$*
• *for every element $a \in G$, there is a correponding inverse element $a^{-1} \in G$, such that $a * a^{-1} = a^{-1} * a = e$.*

Given an n-dimensional object or pattern, the set of isometries which send the object to itself turns out to be a group, which is called *symmetry group* of the object or the pattern. The symmetry group of a regular n-gon is isomorphic to the dihedral group

$$D_{2n} = \langle a, b \,|\, a^n = e, b^2 = e, (ba)^2 = e \,\rangle.$$

Geometrically, the symmetry group is generated by a rotation of order n and one of the reflections.

A regular n-gon can be divided into $2n$ congruent tiles. One tile can be assigned to be the identity, denoted by e. The rest of the tile can be assigned to one of the other elements x of the symmetry group. The particular tile that e goes to upon application of x is labelled x.

We provide below the general procedure of coloring by cosets.

DEFINITION 5. *Let $S \subseteq G$. If H is again a group under the binary operation of G, then S is called a subgroup of G.*

DEFINITION 6. *Let G be a group. Let S be a subgroup of G. Then for any $a \in G$,*

$$Sa = \{\, ha \,|\, h \in S \,\}$$

is a right coset of G.

DEFINITION 7. *Let G be a group. Let S be a subgroup of G. Then for any $a \in G$,*

$$aS = \{\, ah \,|\, h \in S \,\}$$

is a left coset of G.

The procedure for coloring left and right cosets is as follows:

1. assign S, a subgroup of the symmetry group G.
2. Get all the left/right cosets of S.
3. Assign one color to each cosets of S.

Upon looking at a paricular coloring using cosets, one can easily identify the subgroup S used for coloring. The elements of S would be the labels of the tiles having the same color as the tile labelled e.

Given the procedure for coloring using cosets, the number of colors will be equal to the number of cosets. Consequently, the number of colors is equal to the index of S in G for coloring using left and right cosets.

It is important to note that while isometries are normally applied on objects from right to left, words are read from left to right. Thus, the coloring procedure for left cosets described in this section is actually the conventional procedure for right cosets in color symmetry and vice versa.

4.1 Automaton for Coset Coloring

Let G be our symmetry group. Choose a subgroup S of G, and form (left) right cosets of G. Let $(L_L)\, L_R$ be the set of all (left) right coset representatives.

Given a coloring using (left) right cosets. Let $\Sigma = \{a, b\}$, where a represents the action of the rotation of order n on S (or the color of S), and b the action of the reflection on S.

Note that $(L_L)\, L_R$ contains all possible colors, since the cosets are colored differently. In particular, we call $(L_L)\, L_R$ *color language(s).* Then we construct a machine $\mathcal{M} \in \mathbf{DFA}$, such that $L_R = L(\mathcal{M})$.

We define

$$\mathcal{M} = (Q, \Sigma, q_0, \delta, F),$$

such that

Q is finite set of right cosets or colors.

$q_0 = q_S$, which is the color associated to coset represented by the identity element e of G.

$F = Q$

As for the transition function δ, we have, the following:

$\delta(\, q_{Sy}, x\,) = q_S$, for all $yx \in S$.
$\delta(\, q_{Sy}, x\,) = q_{Syx}$, for all $y, x \in G$ and $Syx \neq S$.

Application of a or b on S will send it to one of the cosets. In fact, application of a or b on any of the cosets will send that coset to another coset. Thus, applying a or b on any of the state will bring it to one of the final states. Thus, we have an automaton the accepts L_R. Therefore, we have

THEOREM 5. *[8] L_R is a regular language.*

Note that the dfa we constructed for L_R accepts all elements of G.

The proof of the next theorem is similar to that of Theorem 5, instead of right cosets, we use left cosets.

THEOREM 6. *[8] L_L is a regular language.*

It should be noted that in [8] coloring via double cosets is also shown to be regular, that is we can also construct an automaton that colors using double cosets of a symmetry group.

4.2 State Complexity

Since the automaton we created for our color language uses the cosets of a symmetry group G with respect to a subgroup S, then we have the following;

THEOREM 7. *The deterministic finite automaton that accepts color language with respect to S of G is the cardinal number of the set of distinct (left) right cosets of S in G.*

4.3 Remark

We believed that this opens many possibilities in integrating automata theory and color symmetry. One can examine the ohers of colorings and construct an automaton for each of them. It would be intersting to try to develop a corresponding automaton for a general framework for coloring.

Automaton might be used to determine others propeties of colorings. In particular, the subgroup which permutes the colors and the subgroup the fixes the colors are very important in color symmetry. It might be possible to use automata theory in determining these subgroups for especially complex types of colorings.

5. AUTOMATA FOR SEQUENCES

We are aware of the standard finite automaton models which either accept or reject any given input string [11]. We will be interested in to more general models of function computation by finite automata.

Let w be an input string. The automaton moves from state to state according to its transition function δ, while reading the symbols in w. After reading the whole string w, the automaton halts in a state, say q. Then the automaton outputs $\tau(q)$, where τ is the output mapping. This automaton is called an automaton with output. In particular, we define a *deterministic finite automaton with output* (DFAO) as 6-tuple $M = (Q, \Sigma, \delta, q_0, \Delta, \tau)$ where the sets Q, Σ, δ and q_0 are define classically as in a DFA in [11], Δ is the output alphabet and $\tau : Q \longrightarrow \Delta$ is the output function. Machine M defines a function from Σ^* to Δ, which we denote as $f_M(w)$, as follows:

$$f_M(w) = \tau(\delta(q_0, w)).$$

In this Section, we will be particularly interested in to the case where the input represents a number in base k, for some positive integer $k \geq 2$. If this is the case our DFAO will be called k-DFAO.

5.1 Automatic Sequences

DEFINITION 8. *The sequence $(a_n)_{n\geq 0}$ over a finite alphabet Δ is called k-automatic if there exists a k-DFAO $M = (Q, \Sigma, \delta, q_0, \Delta, \tau)$ such that $a_n = \tau(\delta(q_0, w))$ for all $n \geq 0$ and all w with $[w]_k = n$.*

Alternatively, we can characterize automatic sequences as follows. Let $(a_n)_{n\geq 0}$ be a sequence over Δ, let $k \geq 2$ be an integer and let $d \in \Delta$. Define the set $I_k((a_n)_{n\geq 0}, d) = \{[n]_k \mid a_n = d\}$ as *k-fiber.*

THEOREM 8. *[4] The sequence $(a_n)_{n\geq 0}$ is k-automatic iff each of the fibers $I_k((a_n)_{n\geq 0}, d)$ is a regular language for all $d \in \Delta$.*

We give a reasonable generalization of the automatic sequences. We call such generalization as *k-context-free sequences.* The definition below is suggested in [4]

DEFINITION 9. *A sequence* $\mathbf{a} = (a_n)_{n\geq 0}$ *over an alphabet* Δ *is called* k**-context-free sequences,** *if for all* $d \in \Delta$, *the* k*-fiber* $I_k(\mathbf{a}, d)$ *is a context-free language.*

Note that from Definition 9, it is immediate that k-automatic sequences are k-context-free sequences.

EXAMPLE 3. *Consider the sequence* $\mathbf{a} = (a_n)_{n\geq 0}$ *where* $a_n = 1$, *if* $n = (2^i - 1)2^i$, $i \geq 0$. *The firstfew terms of this sequence is given below.*

$n =$	0	1	2	3	4	5	6	7	8	9	10
$a_n =$	1	0	1	0	0	0	0	0	0	0	0

$n =$	11	12	13	14	15	16	17	18	19	20	21
$a_n =$	0	1	0	0	0	0	0	0	0	0	0

Since $[1^i 0^i]_2 = (2^i - 1)2^i$, *the* 2*-fibers of* $\mathbf{a}$ *are* $I_2(\mathbf{a}, 1) = \{\, 1^i 0^i \mid i \geq 0 \,\}$ *and* $I_2(\mathbf{a}, 0) = \{\, 0, 1 \,\} - I_2 = (\mathbf{a}, 1)$. *Each of this is a context-free language, hence,* $\mathbf{a}$ *is a* 2*-context-free sequence.*

Note that the 2-context-free sequence above is not 2-automatic.

From the definition of k-context-free sequences and the fact that the union of a finite language and a context-free language is context-free, the following are observed[5]:

OBSERVATION 1. *If a sequence* $(b_n)_{n\geq 0}$ *differs only in a finitely many terms from a* k*-context-free sequences* $(a_n)_{n\geq 0}$, *then it is a* k*-context-free sequence.*

OBSERVATION 2. *If* $(a_n)_{n\geq 0}$ *is ultimately periodic sequence, then it is* k*-context-free for all* $k \geq 2$.

Let $k \geq 2$. Cobham showed that a sequence is k-automatic if and only if it is the image, under coding, of a fixed point of k-uniform morphism. This is not however true for k-context-free sequences. Example 3 is a counter example. It can be shown that there is no uniform which has the sequence in Example 3 as its fixed point. But if we relaxed the codition a bit, say we remove the condition of uniformity of morphism, will we obtain a positive answer?

Acknowledgements

The author would like to thank the partial support provided by a Faculty Research Incentive Award (FRIA) from the Department of Computer Science, College of Engineering, UPDiliman and by an NSRI grant no. MAT 06-2-02 of UPDiliman.

6. REFERENCES

[1] H.N. Adorna,: Some Problems in Descriptional Complexity of Finite Automata. *Proc. 5th Phil. Computing Science Congress* 2005, pp 27-32.

[2] H.N. Adorna: On the Class of Automata Between DFA and UFA, (preprint) 2004.

[3] H.N. Adorna, N.H.S. Hernandez, R.D. Lorenzo: On UFA and the Strict Tree Property. *Proc. 8th Phil. Computing Science Congress* 2008.

[4] J-P Allouche, J. Shallit: *Automatic Sequences, Theory, Applications, Generalizations.* Cambridge Uni. Press, 2003.

[5] P.A. Cabral, H.N. Adorna: On the Commutative Complement of Regular Languages. (unpublished manuscript). UPDilman, 2007.

[6] P.A. Cabral, H.N. Adorna: k-Context-Free Sequences and Palindromic Complexity. (unpublished manuscript). UPDiliman, 2006.

[7] K. Ellul: Descriptional Complexity Measures of Regular Languages. Masters Thesis, University of Waterloo, 2004.

[8] I.F. Evidente, H.N. Adorna: A Note on Coloring via Automaton. (unpublished manuscript). UPDilman, 2007

[9] M.R. Garey, D.S. Johnson: *Computers and Intractability: A Guide to the Theory of NP-Completeness.* Freeman, San Francisco, 1979.

[10] J.E. Hopcroft: An $n \log n$ algorithm for minimizing states in finite automata. Technical Report STAN-CS-71-190. Stanford University, Computer Science Department, 1971.

[11] J.E. Hopcroft, J.D. Ullman: *Introduction to Automata Theory, Languages and Computation.* Addison-Wesley, Reading, MA, 1979

[12] J. Hromkovič: Descriptional Complexity of Finite Automata: Concepts and Open Problems, *Journal of Automata, Languages and Combinatorics* Vol. 7, No. 4, 2002, pp 519-531.

[13] J. Hromkovič: *Communication Complexity and Parallel Computing.* Springer 1997.

[14] J.Hromkorič, G. Schnitger: Nondeterminism versus determinism for two-way finite automata: Generalizations of Sipser's separation. *Lecture Notes in Computer Science* 2719, 2003, pp. 439-451.

[15] J. Hromkovič, J. Karhumäki, H. Klauck, G. Schnitger, S. Seibert: Communication complexity method for measuring nondeterminism in finite automata. *Information and Computation* 172, 2002, pp 202-217.

[16] T. Hungerford: *Algebra.* GTM, Springer Verlag (1974).

[17] B. Krawetz: Monoids and the State Complexity of the Operation root(L). Master's Thesis, University of Waterloo (2003)

[18] H. Leung: Separating exponentially ambiguous finite automata and polynomially ambiguous finite automata. *SIAM J. Compt.* 27, 1998, pp 1073-1082.

[19] F. Moore: On the bounds for state-set size in the proofs of equivalence deterministic, nondeterministic and two-way finite automata. *IEEE Trans. Comput.* 10, 1971, pp 1211-1214.

[20] M.O. Rabin, D. Scott: Finite automata and their decision problems. *IBM J. Research and Development* 3, No. 2, 1959, pp 115-125.

[21] B. Ravikumar, O. Ibarra: Relating the type of ambiguity of finite automata to the succinctness of their representation. *SIAM J. Comput.* 19, 1989, pp 1263-1282.

[22] E. Schmidt: Succinctness of description of context-free, regularand finite languages. Ph.D. thesis, Cornell University, Ithaca,NY. 1978.

[23] C.E. Shannon, J McCarthy: *Automata Studies.* Princeton University Press, 1956.

[24] R. Stern, H. Hunnt: On the equivalence and containment problems for unambiguous regular expressions, regular grammars, and finite automata. *SIAM J. Compt.* 14, 1985, pp 598-611.

[25] S. Yu: A Renaissance of Automata Theory? Bulletin of the EATCS, No. 72, 2000, pp 270-272.