FINITE AND SMALL-SPACE AUTOMATA WITH ADVICE

by

Uğur Küçük

B.S., Computer Engineering, Boğaziçi University, 2003

M.S., Computer Engineering, Boğaziçi University, 2005

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy

Graduate Program in Computer Engineering

Boğaziçi University

2018

FINITE AND SMALL-SPACE AUTOMATA WITH ADVICE

APPROVED BY:

Prof. A. C. Cem Say (Thesis Supervisor)	.....
Assist. Prof. Ahmet Çevik	.....
Prof. Tunga Güngör	.....
Prof. Sema Fatma Oktuğ	.....
Assoc. Prof. Alper Şen	.....

DATE OF APPROVAL: 17.09.2018

ACKNOWLEDGEMENTS

I am grateful to my supervisor Prof. A. C. Cem Say for his patience, guidance and support which made this work possible. I owe special thanks also to Abuzer Yakaryılmaz whose contribution to my research was of great significance.

I wish to thank professors Sema Fatma Oktuğ, Tunga Güngör, Alper Şen and Ahmet Çevik for kindly accepting to participate in my defense jury and for their generosity in spending their valuable time reading my work and commenting on it.

I must express my gratitude to professors Alper Şen and Atilla Yılmaz for their valuable comments and advice which regularly helped shaping this work over the years. During the course of these years, I also had chance to exchange ideas on my research with Özlem Salehi Köken, Gökalp Demirci, Flavio D'Alessandro, Ryan O'Donnell, and Öznur Yaşar Diner. I thank them as well, for the valuable feedback and ideas they shared.

Finally, I wish to thank my family for their endless love and patience. My beloved wife Deniz, and my little son Özgür sponsored this work not only by providing the inspiration I needed but also by tolerating long hours of my absence from home. Without their support, I would not even dare to start what I have finished today.

ABSTRACT

FINITE AND SMALL-SPACE AUTOMATA WITH ADVICE

Advice is a piece of trusted supplemental information that is provided to a computing device, in advance of its execution in order to extend its power beyond its limits and hence to assist it in its task. The content of this assistance, which is not restricted to be computable, typically depends only on the length, and not the full content of the actual input to the device. Advised computation has been studied on various computational models and in relation with concepts as diverse as complexity, nonuniform computation, formal languages and pseudorandomness. Several models for providing such external assistance to finite automata have also been studied by various groups.

In this research, we introduce two novel models of advised finite automata: finite automata with advice tapes and finite automata with advice inkdots. In the former model advice is provided in the form of a string which is placed on a separate tape accessible independently from the input. In the latter one, we model advice as a set of uniform marks placed on the input tape which are called inkdots. We examine the power and limits of each of these models in a variety of settings where the underlying model of computation is deterministic, probabilistic or quantum and the advice is deterministically or randomly chosen. The roles of increasing amounts of advice as a computational resource are taken into consideration in various forms. The variants of each model are compared with each other and with the previously studied models of advised finite automata in terms of language recognition power. The main results of this analysis are demonstrated by establishing various separations, collapses and infinite hierarchies of the language classes that can be recognized with different models in varying settings.

ÖZET

ÖĞÜT ALAN SONLU DURUMLU VE KÜÇÜK BELLEKLİ MAKİNELER

Öğüt, bir hesaplama aygıtına bu aygıtın gücünü kendi sınırlarının ötesinde genişleterek hesaplamasına yardım etmek için sağlanan dış kaynaklı güvenilir bir bilgi parçasıdır. Hesaplanabilir olma kısıtlaması olmayan bu yardımın içeriği tipik olarak yalnızca aygıtın gerçek girdisinin boyutuna bağlıdır ve girdinin esas içeriğinden bağımsızdır. Öğüt alan hesaplamanın özellikleri çeşitli hesaplama modelleri baz alınarak ve karmaşıklık, çok biçimli hesaplama, formel diller ve sözde rastgelelik gibi farklı kavramlar ile bağlantılı biçimde çalışılagelmiştir. Sonlu durumlu makinalara bu türden harici yardım sağlamak için geliştirilen birkaç model de çeşitli gruplar tarafından çalışılmıştır.

Bu araştırma kapsamında iki yeni öğüt alan sonlu durumlu makine modeli tanımlandı: öğüt şeritli sonlu durumlu makineler ve işaretle öğüt alan sonlu durumlu makineler. İlk modelde öğüt, bir dizi şeklinde ve girdi şeridinden bağımsız olarak erişilebilen ayrı bir şerit üzerinde sağlanır. İkinci modelde ise öğüt, girdi şeridi üzerine konulan ve iz adı verilen tek biçimli işaretler aracılığı ile sağlanmaktadır. Bu modellerin her birinin hesaplama gücü ve sınırları, temel hesaplama modelinin belirlenimci, olasılıksal ya da kuantum olmasına ve öğütün belirlenimci ya da rastgele biçimde seçilmesine bağlı olarak değişen çeşitli durumlarda incelendi. Artan öğüt miktarının bir hesaplama kaynağı olarak etkileri çeşitli biçimlerde değerlendirmeye dahil edildi. Her bir modelin versiyonları dil tanıma güçleri açısından, kendi aralarında ve daha önceden çalışılmış benzer makine modelleri ile karşılaştırıldı. Bu incelemenin temel sonuçları olarak söz konusu modeller tarafından değişik durumlarda tanınabilen dil sınıfları arasındaki çeşitli ayrışma, örtüşme ve sonsuz sıradüzen ilişkilerinin varlığı gösterildi.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

$\mathsf{C}(k)$	The k-fold conjunctive closure of a class of languages, C defined as the class of languages that can be obtained by intersecting k member languages from C
$d(.)$	Functional upper bound of nonconstructivity in terms of input length
$\|G\|$	The cardinality of the production set of a grammar G
h	An advice function
$\overline{L}$	The complement of a language, L
$L(G)$	The language produced by a grammar G
n	Input length
$\mathbb{N}$	The set of natural numbers
$\mathbb{N}^+$	The set of positive natural numbers
$o(.)$	Asymptotic notation, Little-O
$O(.)$	Asymptotic notation, Big-O
q	A state of an automaton
q_0	The initial state of an automaton
q_{accept}	The accepting state of an automaton
q_{reject}	The rejecting state of an automaton
Q	The set of states of an automaton
$S : h$	The language of words x, where $h(\|x\|)x$ is a member in the language S
V/F	The collection of languages that can be placed in a collection of languages, V, with the help of some advice function which produces such advice strings that their length is governed by a function from a collection of functions, F
$\|w\|$	The length of a word w
$\|w\|_a$	The number of occurrences of a symbol a within a word w
w_i	The i'th symbol of a word w

$w_{i:j}$	The subword of a word w, which starts with w_i and ends with w_j
w^R	The reverse of a word w
$\mathbb{Z}$	The set of integers
$\mathbb{Z}^+$	The set of positive integers
$[i,j]_{\mathbb{R}}$	The set of real numbers between a pair of values i and j
$[i,j]_{\mathbb{Z}}$	The set of integers between a pair of values i and j
Γ	An advice alphabet
Δ	A function which determines the projective measurements to be applied on the quantum state of a quantum finite automaton, as part of its transitions
δ	The transition function of a finite automaton
ϵ	An error bound
Θ	A function which determines the unitary operations to be applied on the quantum state of a qfat, as part of its transitions
$\Theta(.)$	Asymptotic notation, Big-Theta
λ	Empty string
μ	A probability ensemble
$\prod_s^t$	The class of languages recognized by alternating Turing machines which start in universal state, alternate s times and terminate in $O(t)$ number of steps
Σ	A probability threshold
Σ	An alphabet
$\|\Sigma\|$	The cardinality of an alphabet Σ
Σ^n	The set of words of length n over an alphabet Σ
Σ^*	The set of all words over an alphabet Σ
Σ^+	The set of all non-empty words over an alphabet Σ
$\sum_s^t$	The class of languages recognized by a alternating Turing machines which start in existential state, alternate s times and terminate in $O(t)$ number of steps

χ_L	The characteristic function of a language L which is defined as $\chi_L(x) = 1$ if $x \in L$ and $\chi_L(x) = 0$ otherwise
$\omega(.)$	Asymptotic notation, Little-Omega
$\Omega(.)$	Asymptotic notation, Big-Omega
$\mid$	... conditioned on ...
$\lfloor . \rfloor$	Floor function
$\lceil . \rceil$	Ceiling function
$\odot$	Inkdot
$\emptyset$	Empty set
$\equiv$	An equivalence relation
$\cong$	A closeness relation
$\leq$	A partial order
$\ll$	Much smaller than
$\setminus$	Set difference operator
$\vdash$	Left end-marker
$\dashv$	Right end-marker
∞	Infinity
$^\diamond$	A modifier which indicates rewritable advice track

LIST OF ACRONYMS/ABBREVIATIONS

1-BPFA	The class of languages that can be recognized with bounded error by one-way probabilistic finite automata
1-BPLIN	The class of languages that can be recognized with bounded error by linear-time one tape probabilistic Turing machines
1-$C_=$LIN	The class of languages that can be recognized with exactly 1/2 error probability by linear-time one tape probabilistic Turing machines
$1dfa$	One-way deterministic finite automaton
1-DFA	The class of languages that can be recognized by one-way deterministic finite automata
1-DLIN	The class of languages that can be recognized by linear-time one tape deterministic Turing machines
1-DSPACE($s(n)$)	The class of languages that can be recognized by one-way deterministic Turing machines with space bound, $s(n)$
1-PLIN	The class of languages that can be recognized with unbounded error by linear-time one tape probabilistic Turing machines
$1qfa$	One-way measure-many quantum finite automaton
1-QFA	The class of languages that can be recognized with bounded error by one-way measure-many quantum finite automata
1-$QFA_{(a(n),b(n))}$	The class of languages that can be recognized by one-way measure-many quantum finite automata which accept member strings of length n with probability at least $a(n)$ and rejects the nonmember strings of length n with probability at least $b(n)$
$1rfa$	One-way deterministic reversible finite automaton
1-RFA	The class of languages that can be recognized by one-way deterministic reversible finite automata
$1tm$	One-way Turing machine
$1w$	One-way
2-DFA	The class of languages that can be recognized by two-way deterministic finite automata

$2w$	Two-way
ALL	The class of all languages
be	Bounded error
BPFA	The class of languages that can be recognized with bounded error by real-time probabilistic finite automata
BQFA	The class of languages that can be recognized with bounded error by real-time quantum finite automata
cf	Context-free
cf_L	The context-free cost of a language L
CFL	The class of context-free languages
CFLMV	The multiple-valued partial CFL-function class
co	Complement
$const$	Constant
$Const_{cf}$	The class of languages with constant context-free cost
$Const_{cs}$	The class of languages with constant context-sensitive cost
$Const_{reg}$	The class of languages with constant regular cost
$Const_{re}$	The class of languages with constant recursively enumerable cost
cs	Context-sensitive
CS	The class of context-sensitive languages
cs_L	The context-sensitive cost of a language L
DAG	Directed acyclic graph problem
dfa	Deterministic finite automaton
$dfat$	Deterministic finite automaton with advice tape
DSPACE(f)	The class of languages that can be recognized by deterministic Turing machines which use $O(f(n))$ space
exp	Exponential
EXPTIME	The class of languages that can be recognized by deterministic Turing machines in exponential time
F	Set of final (accepting) states of an automaton
FL	The logarithmic space function class
G	A grammar

$\mathcal{H}(Q)$	The Hilbert space spanned by a finite set of quantum states, Q
$K(w)$	The Kolmogorov complexity of a word, w
KWqfa	Kondacs-Watrous quantum finite automaton
$\mathcal{L}$	Left (Tape head direction)
L	A language
lin	Linear
log	Logarithmic
mod	Modulo operation
$\mathcal{O}(.)$	The set of observation operators (projective measurements) over a Hilbert space
NP	The class of languages that can be recognized by nondeterministic Turing machines in polynomial time
NSPACE(f)	The class of languages that can be recognized by nondeterministic Turing machines which use $O(f(n))$ space
p	Probability
P	The class of languages that can be recognized by deterministic Turing machines in polynomial time
$\mathcal{P}(S)$	Power set of a set S
PAL	The language of even-length palindromes
pfa	Probabilistic finite automaton
PFA	The class of languages that can be recognized with unbounded error by real-time probabilistic finite automata
$pfat$	Probabilistic finite automaton with advice tape
$poly$	Polynomial
$Poly_{cf}$	The class of languages with polynomial context-free cost
$Poly_{cs}$	The class of languages with polynomial context-sensitive cost
$Poly_{re}$	The class of languages with polynomial recursively enumerable cost
$Poly_{reg}$	The class of languages with polynomial regular cost
$polylog$	polynomial of logarithms
PSPACE	The class of languages that can be recognized by deterministic Turing machines which use polynomial amount of space

qfa	Quantum finite automaton
QFA	The class of languages that can be recognized with unbounded error by real-time quantum finite automata
$qfat$	Quantum finite automaton with advice tape
$\mathcal{R}$	Right (Tape head direction)
R	Random
RE	The class of recursively enumerable languages
re_L	The recursively enumerable cost of a language L
REG	The class of regular languages
reg_L	The regular cost of a language L
rfa	Reversible finite automaton
$Rlin$	Randomly picked, linearly sized
Rn	Randomly picked, size n
re	Recursively enumerable
reg	Regular
rt	Real-time
$\mathcal{S}$	Stay (Tape head direction)
SAT	Boolean satisfiability problem
T_I	The set of allowed head movements for the input tape of a finite automaton with advice tape
T_A	The set of allowed head movements for the advice tape of a finite automaton with advice tape
$\mathcal{U}(.)$	The set of unitary operators over a Hilbert space
ue	Unbounded error

1. INTRODUCTION

The idea of advised computation is based on providing supplemental external assistance to computational devices that would enhance their capabilities beyond their limits. Analyzing the effects of such external assistance is a well known method with a long history, proven many times to be fruitful for gaining better insights on the strengths and weaknesses of different computational models, for exploring the nature of complexity inherent within computational problems and for discovering their relations to each other. In this respect, advice has a fundamental similarity with other concepts such as oracle machines and interactive proof systems (see *e.g.* [1]) that might make it convenient to label such approaches altogether as "assisted computation" due to the external assistance in the core of these concepts. Advice is different from the assistance inherent in interactive proofs in the sense that it is trusted and it differs from the assistance from an oracle as the content of the assistance is determined only by the size of the input and it is provided in advance of the execution of the device.

The following set of characteristics can be said to distinguish the concept of advice among other models of assisted computation:

- Advice is provided in advance of the execution of the device in the form of a piece of supplementary information beyond the original input.
- The content of the advice depends only on the length of the actual input to the device and not on the full content of it.
- The supplemental information provided as advice is trusted.
- Advice is assumed to be provided by a computationally unlimited source and hence does not need to be computable.

Since its introduction in the early 1980's, advised computation has been studied in various contexts and in relation with various computational phenomena. Researchers who are in search of better insights for various computational problems or who look for a better understanding of the roles of advice applied this concept with different settings

and on various models of computational devices. As many conventional techniques of reasoning turn out to be useless due to the difficulty of analyzing the underlying steps of computation in the presence of advice, demonstrating the exact nature of advised computation many times required adopting or inventing novel methods or analyzing new models. This lead to a distinctive literature on advised computation. A part of this literature is dedicated to advised computation by finite automata which is particularly interesting due to the relative weakness of the finite automata as the underlying model of computation which leaves wider space for analyzing the roles of advice. A set of advised finite automata models are introduced and examined by several groups of researchers forming a basis of information in this domain, which we aim to extend with this research.

New models of advised finite automata, the classes of languages that can be recognized by these models and the relations among these classes are in the primary focus of this thesis. Through these abstractions, we aim to examine the power and limitations of advice as a computational resource in the context of finite automata. For this purpose, we introduce two novel models of advised finite automata -finite automata with advice tape and finite automata with inkdot advice- and analyze the roles of advice in these models in comparison with each other and with the previously studied models of advised finite automata. The relations among the classes of languages that can be recognized in these models are examined in detail. The results indicating the separations, collapses and hierarchies among these classes are the major contribution of this research, most of which appeared previously on [2, 3] and [4].

The remainder of this work is structured as follows: Chapter 2 begins with the basics of advised computation. Then, in chronological order, previously studied models of advised finite automata will be introduced together with a brief discussion of what is already known on each of them.

Chapter 3 will introduce an alternative model of advised finite automata in which the supplemental information is placed on a separate tape. We will consider several variants of this model where the advice is deterministic or randomized, the input and

advice tape heads are allowed real-time, one-way, or two-way access, and the automaton is deterministic, probabilistic or quantum. We will compare the power of this model with the previously studied models and set several separations and collapses among the classes of languages that can be recognized in each model. We will show that allowing one-way access to either of the input or advice tapes, so that the automaton can stay put on this tape during a transition, makes a deterministic finite automaton with advice tape stronger over those with real-time access. We will present results that demonstrate infinite hierarchies of language classes that can be recognized with increasing amounts of advice in this model both when the advice is restricted to be constant in length and when an increasing function of the input length is set as the limit on the amount of the advice. We will introduce randomness into the model by separately allowing probabilistically selected advice strings and probabilistic transitions of the automata and show that both of these variations add up to the power of the model in bounded-error setting. We will finally consider the quantum finite automata with an advice tape and show it is computationally more capable than its probabilistic counterpart in certain settings.

Chapter 4 will introduce inkdots placed on the input string as an alternative way of providing advice to finite automata. The power of this model will be compared to the previously studied models of advised finite automata and several results indicating the separations and collapses among the classes of languages that can be recognized in each setting will be presented. It will be shown that both when the number of advice inkdots is restricted to be a constant and when it grows with the length of the input, the class of languages that can be recognized with finite automata grows as the amount of allowed inkdots increases, hence forming infinite hierarchies of language classes. We will present results showing that finite automata with inkdots as advice can be more succinct than the other advised automata models and the pure unadvised automata in terms of the number of states required for recognizing certain languages. We will discuss randomly placed inkdots as advice to probabilistic finite automata, and demonstrate the superiority of this model in bounded-error language recognition over its deterministic version. Finally, we will extend the model with access to infinite secondary memory and present a result which shows that even very small and slowly

growing amounts of space can bring in more power for language recognition when it is provided to automata together with an inkdot as advice.

Chapter 5 will be a conclusion which summarizes the results we obtained in the context of advised finite automata and it will also provide a list of open questions which may point to future directions in this line of research.

2. A SHORT SURVEY ON ADVISED FINITE AUTOMATA

The idea of advised computation is put forward in the early 1980's by Karp and Lipton, in their seminal paper [5], "Turing machines that take advice". At the time, a range of models for assisted computation that depend on supplemental external assistance - whether trusted or not- had already been studied by groups of researchers in connection with various concepts. The novelty of Karp and Lipton's approach was due to their underlying motivation which they described as reaching conclusions on uniform complexity of computational problems as a consequence of nonuniform complexity bounds. As a consequence of this motivation, their model of external assistance took the form of a string called advice, which depends only on the length of the actual input. It is provided as trusted external assistance for a computational device in parallel with its actual input in advance of its execution.

This notion of advised computation which was originally applied for Turing machines has been applied to other types of computational devices as well. Among such examples, we are particularly interested in the models of advised finite automata as they precede the models we study in the scope of this research.

The first advised finite automaton model was proposed by Damm and Holzer in [6]. In this setup, prior to the computation of the automaton, advice is placed on the input tape as a prefix of the original input so that the automaton starts by scanning the advice. Yamakami and his coauthors, in a series of papers [7–13], studied another model of advised finite automaton in which the advice is placed on a separate track of the input tape and scanned by the automaton in parallel with the actual input. Finally, the model of advised finite automata proposed by Freivalds and his coauthors in [14–16], incorporates one or more separate tapes for the advice and the automaton is granted two-way access to both the input and the advice tapes. Unlike the other models, it is required in this model that the advice string for inputs of any length must

be helpful not only for the inputs of that length but also for all inputs which are shorter as well.

Below, in this chapter, we will first provide a short overview of the notion of advised computation as introduced by Karp and Lipton and list their findings in Section 2.1. Then in the following parts, we will be presenting the advised finite automata models which precede the models we examine in this research. Finite automata with advice prefix as introduced by Damm and Holzer will be presented in Section 2.2. Finite automata with advice track as introduced and examined by Yamakami and his coauthors will be presented in Section 2.3. Finally, finite automata with multiple two-way advice tapes as introduced by Freivalds and his coauthors will be presented in Section 2.4.

2.1. Turing Machines that Take Advice

Karp and Lipton introduced the notion of advice in order to provide a framework for studying nonuniform measures of complexity and to examine the cases where it is possible to reach conclusions about uniform complexity of problems as a result of nonuniform complexity results. In this context, they modeled advice as a piece of supplementary information coded into a string depending only on the length of the actual input which is provided as trusted external assistance for a computational device in parallel with its actual input in advance of the execution. (The term "nonuniform computation" refers to those models of computation like Boolean circuits or decision trees where a different circuit or structure is needed for solving different instances of a problem which vary in size; whereas "uniform" models of computation are those like Turing machines where a single device is expected to solve every instance of a problem regardless of its size. The reader may refer to [17, 18] for a brief discussion on the complexity of Boolean circuits which is a fundamental example for the concept of nonuniform computation.)

Below, in Definition 2.1, we rephrase the original definition of the notion of advice by Karp and Lipton [5].

Definition 2.1. *Let S be a language defined on the alphabet $\{0,1\}$ and let $h : \mathbb{N} \to \{0,1\}^*$ be a function that maps natural numbers to strings in $\{0,1\}^*$ which are called advice. Define the language $S : h$ so that any member x of $S : h$ prefixed by the appropriate advice string, $h(|x|)$ is a member of S.*

$$S : h = \{x \mid h(|x|)x \in S\}. \tag{2.1}$$

Now let V be any collection of languages defined on the alphabet $\{0,1\}$ and let F be any collection of functions from $\mathbb{N}$ to $\mathbb{N}$. Then, the collection of languages that can be placed in V with the help of some advice the amount of which is bounded by F is symbolized by V/F and defined formally as

$$V/F = \{S : h \mid S \in V \text{ and } h \in F\}. \tag{2.2}$$

P/log, for example, denotes the class of languages that can be recognized in polynomial time by a Turing machine that takes advice strings, the size of which is bounded by some logarithmic function of the input length. Similarly, $\mathsf{DSPACE}(\log n)/poly$ denotes the class of languages that can be recognized by a deterministic Turing machine with logarithmic space bound that takes advice strings the size of which is bounded by some polynomial function of the input length. Fact 2.2 lists some of the direct results of this definition.

Fact 2.2. *([5]) Some preliminary facts are:*

(i) for all V, $V/0 = V$;
(ii) any subset of $\{0,1\}^*$ is in $\mathsf{P}/2^n$;
(iii) if f is infinitely often nonzero, then P/f contains nonrecursive sets;
(iv) if $g(n) < f(n) \leq 2^n$ infinitely many times, then $\mathsf{P}/g \subseteq \mathsf{P}/f$

The first statement in Fact 2.2, formulates the trivial fact that a zero-length advice would be equivalent in power to the no advice case. Second statement can be

justified by noting that a Turing machine would be capable of performing a search for checking if its input occurs on its advice and then noting that for any language, an exponentially-long advice string can list all the members of this language which are of the same length as the input. The third statement can best be exemplified by considering an undecidable unary language (which exists since every language has a unary version). Note that a unary language can have at most one member of any given length. Hence given the length of the input to a machine one bit of advice which specify whether the language has a member of that length or not would be sufficient to help the machine decide whether the input is a member or not. Recall that the advice function needs not to be computable and note that an undecidable unary language (and also its complement) would have infinitely many members hence the advice function defined above need to provide that nontrivial membership information for infinitely many values of the input length. The last statement which states that longer advice strings are at least as powerful as the shorter ones as advice is rather trivial since it is always possible to pack the information in a string into a longer one.

The classes of languages that attracted major initial interest are P/log and $\mathsf{P}/poly$, the classes of languages that can be recognized by a Turing machine within polynomial time and with the assistance of advice strings of logarithmic or polynomial length respectively. Of these two, $\mathsf{P}/poly$ has close ties with classical circuit complexity: $\mathsf{P}/poly$ is equivalent to the class of languages that has circuits which are small in the sense that the number of gates in these circuits are bounded by some polynomial function of the input size.

Below, in Fact 2.3, this relation is expressed in the way Karp and Lipton cited it from an earlier work by Pippenger [19].

Fact 2.3. *([5]) Let S be a subset of $\{0,1\}^*$. Then the following are equivalent.*

(i) S has small circuits.

(ii) S is in $\mathsf{P}/poly$.

Based on this notion of advised computation, Karp and Lipton obtained a set of results most of which are of the form

$$L \subseteq V/F \implies L \subseteq S' \tag{2.3}$$

where the idea of advised computation is used for gaining insights on the nonuniform complexity class V/F from which one can then draw implications on the relation of the uniform complexity classes L and S'. The proofs for these results also share a common pattern: First it is shown that there exists a language K which is complete in L with respect to some reducibility function that is appropriate for use in reaching the conclusion required for proving the particular statement in consideration. Then assuming $K \in V/F$, it is stated that K must be of the form $S : h$ for some language $S \in V$ and where $|h(x)|$ has a known bound. The remainder is to show that $K \in S'$ by providing an appropriate uniform algorithm for recognizing K, hence proving

$$K \in V/F \implies K \in S', \tag{2.4}$$

which is then used for reaching the desired conclusion.

The method for constructing such a uniform algorithm for recognizing K varies depending on the structure of K. As this algorithm will lack access to the advice function h, it is suggested that the method needs to consider all strings that could be the advice for the input string in consideration by exploiting the fact that the advice string provided by h is consistent for all strings of the same length. This core idea is used by Karp and Lipton with appropriate enrichments in cases where K is a game, where K is self reducible and where K has a simple recursive definition and shown to be fruitful in each case. (Informally, a set is said to be self reducible if the problem of deciding whether a string is a member of this set can be reduced in polynomial time to the same problem for smaller strings. For a more formal definition of self reducibility, the reader may refer to the original text or to [20].)

The problems that are complete for the classes of languages that can be recognized by an alternating Turing machine with use of certain amounts of time and space can often be represented in the form of a game. Karp and Lipton introduced a technique which they called round robin tournament method in order to relate the nonuniform complexity of a game to its uniform complexity and hence to obtain a better understanding of various complexity classes. Below is a list of relations, established in this context.

$$\mathsf{PSPACE} \subseteq \mathsf{P}/poly \implies \mathsf{PSPACE} = \textstyle\sum_2^p \cap \prod_2^p \tag{2.5}$$

$$\mathsf{PSPACE} \subseteq \mathsf{P}/log \iff \mathsf{PSPACE} = \mathsf{P} \tag{2.6}$$

$$\mathsf{EXPTIME} \subseteq \mathsf{PSPACE}/poly \iff \mathsf{EXPTIME} = \mathsf{PSPACE} \tag{2.7}$$

$$\mathsf{P} \subseteq \mathsf{DSPACE}((\log n)^l)/log \iff \mathsf{P} \subseteq \mathsf{DSPACE}((\log n)^l) \tag{2.8}$$

For each of these results, the existence of problems which can be represented in the form of a game and which are complete for the uniform complexity class in consideration is expressed with reference to earlier results in [21] and [22]. If some amount of nonuniformity suffices to solve such a game which is complete for this class and if it is possible to play a round robin tournament of the game in consideration, among all potential strings that could be the advice for any input in consideration with use of appropriate amounts of time and space associated with this class, then Karp and Lipton suggested that it is possible to provide a uniform algorithm for the same game within the same time and space bounds as well. The idea is simply to turn the nonuniform algorithm into a uniform one by just playing the round robin tournament among all potential advice strings and then picking the undefeated champion of this tournament to guide the rest of the computation as in the nonuniform case.

Karp and Lipton, next, showed that the uniform complexity of a self reducible set K can be related to its nonuniform complexity by reducing instances of the problem to smaller instances with use of its self reducibility structure. The complexity of the uniform algorithm obtained in this manner would depend on the cost of testing membership in the minimal set specified in the self reducibility structure and its intersection with K in addition to the number of reduction steps to reach a minimal member and the number of strings that can be valid advice strings. The statements below are shown to be true with use of this line of reasoning.

$$\mathsf{NP} \subseteq \mathsf{P}/log \iff \mathsf{P} = \mathsf{NP} \tag{2.9}$$

$$\mathsf{NSPACE}(\log n) \subseteq \mathsf{DSPACE}(\log n)/log \iff \mathsf{NSPACE}(\log n) = \mathsf{DSPACE}(\log n) \tag{2.10}$$

The self reducibility of SAT (satisfiability problem) is used for the proof of the former statement and the self reducibility of DAG (a problem specified on directed acyclic graphs that questions the existence of a directed path between two pre-specified vertices) is used for the latter, for which the contribution of Ravindran Kannan is also noted.

The famous result known as Karp-Lipton theorem (or more conveniently as Karp-Lipton-Sipser theorem as the final form of the theorem is attributed by Karp and Lipton to Sipser) is obtained in this context with use of the idea of using self referential description of a language (the language of satisfiable quantified Boolean formulas in this case) in order to derive conclusions about its uniform complexity based on its nonuniform complexity.

Karp and Lipton first showed as a lemma that

$$\mathsf{NP} \subseteq \mathsf{P}/poly \implies \bigcup_{i=1}^{\infty} \sum_i^p \subseteq \mathsf{P}/poly \tag{2.11}$$

by use of the self reducibility structure inherent within the languages E_i and A_i of satisfiable quantified Boolean formulas with a maximum of i alternating quantifiers that begin with an existential quantifier in the case of E_i and with a universal quantifier in the case of A_i. It is then pointed out that the test of membership of a string x in A_i (or E_i) can be described with reference to the test of membership of strings x' and x'' in A_i (or E_i) where x' and x'' are strings obtained by substituting the first variable within x by values 0 and 1 and hence eliminating the first one of the quantifiers within x.

If one assumes $\mathsf{NP} \subseteq \mathsf{P}/poly$ then by Eq. 2.11 one can show that $A_3 \in \mathsf{P}/poly$. Hence there exists a polynomial-length advice function, access to which enables a Turing machine to perform the test of membership in A_3 within polynomial time. Combining this information with the self referential description of the membership test provides sufficient means to conclude the main statement of the Karp-Lipton theorem:

$$\mathsf{NP} \subseteq \mathsf{P}/poly \implies \bigcup_{i=1}^{\infty} \sum_i^p = \sum_2^p. \tag{2.12}$$

In other words, if NP is contained in $\mathsf{P}/poly$ (or equivalently if there exists small circuits for any problem in NP) then the polynomial hierarchy of Meyer and Stockmeyer ([23]) would collapse at its second level.

Karp and Lipton report that their original statement which pointed to a collapse at the third level of the polynomial hierarchy was later improved by Sipser to show a collapse at the second level. Such a collapse is widely believed to be unlikely and thus many experts believe that $\mathsf{NP} \not\subseteq \mathsf{P}/poly$ which would then imply $\mathsf{NP} \not\subseteq \mathsf{P}$.

Finally, by noting the contribution of Albert Meyer, Karp and Lipton showed that

$$\mathsf{EXPTIME} \subseteq \mathsf{P}/poly \implies \mathsf{EXPTIME} = \sum_2^p \implies \mathsf{P} \neq \mathsf{NP} \tag{2.13}$$

with a similar reasoning on the recursive description of the languages associated with the winning configurations of the special type of games mentioned earlier in this section.

Karp and Lipton's notion of advice, the methods they introduced for analyzing advised computation and the results they obtained have since then affected a wide range of domains in computer science and related fields. Obviously nonuniform complexity and circuit complexity are in the core of all these discussions. On the other hand, their results affected many other fields including pseudorandomness, one-way functions and zero knowledge proofs.

2.2. Advice as a Prefix of the Input

Karp and Lipton's formalization of advised computation as a mean for studying the effects of nonuniformity was later adopted by Damm and Holzer in [6], into the field of automata and formal language theory in the mid 1990s. They examined the classes of Chomsky hierarchy: regular, context-free, context-sensitive and recursively enumerable languages relative to constant and polynomial-length advice and obtained various separation results among these sets.

Following the same notation used by Karp and Lipton, the classes of languages in consideration are denoted by expressions of the form V/F. V is either of the abbreviations REG, CFL, CS and RE denoting respectively the unadvised classes of regular, context-free, context-sensitive and recursively enumerable languages where F takes either of the values *const* or *poly* for representing the settings in which bound on the length of the allowed amount of advice is a constant or a polynomial function of the input length.

Damm and Holzer modeled advice as a supplementary piece of information that depends only on the length of the actual input and which, prior to the computation of the automaton, is placed on the input tape as a string that precedes the original input. The function that maps the length of the input to the advice string is called advice function and is conventionally signified by the letter h.

The advised automata models in consideration are simply identical to the original unadvised models other than the obvious difference that requires them to start their execution while scanning the first symbol of the advice rather than the first symbol of the actual input. Hence, the automata models with prefix advice are required to consume the whole advice string before they can read the symbols from the input string.

The first advised finite automata model, which we will call "finite automata with prefix advice" throughout this work, was introduced in this context together with advised versions of other automata models that correspond to the higher levels of Chomsky Hierarchy. (See Figure 2.1.)

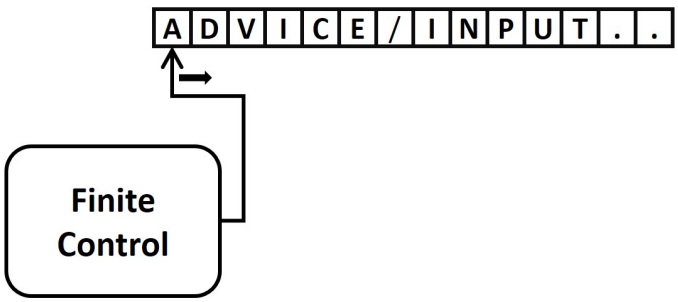

Figure 2.1. Schematic description of a finite automaton with prefix advice.

Damm and Holzer first considered the effects of varying amounts of advice provided to finite automata in this model. It is asserted that allowing advice strings which grow as the input grows would bring in no additional computational power over the advice strings of constant-length. Hence the classes of languages that can be recognized by finite automata with the help of constant-length prefix advice is equal to the class of languages that can be recognized with the help of advice strings whose length is limited by a polynomial function of the input length. On the other hand, the class of languages, REG/k, that can be recognized by finite automata with prefix advice, is

shown to grow as the amount of the allowed advice increases:

$$\mathsf{REG}/const = \mathsf{REG}/poly \tag{2.14}$$

$$\mathsf{REG}/k \subsetneq \mathsf{REG}/(k+1), \text{ for all } k \geq 0 \tag{2.15}$$

The first one of these propositions depends on the observation that, at any point of its computation, the amount of information that a finite automaton can memorize, is constant and this information has to be coded into its current control state. Hence, the operation performed by a finite automaton with prefix advice, while scanning the advice string, has no effect on the rest of the computation other than picking one of its control states as the current state before it consumes the first symbol of the actual input. As the number of control states of a finite automaton is constant, a finite set of constant-length advice strings would be sufficient to guide a slightly modified version of such an automaton to exactly the same set of states before its computation on its actual input.

The second proposition is shown to be true by use of an argument that utilizes Kolmogorov complexity terms, referring to the incompressibility of a random binary sequence in addition to a characterization theorem, that indicates piecewise regularity of the members of REG/k.

This theorem is cited as Fact 2.4 below. (A similar technique will be employed in Chapter 4 of this work for showing the truth of the Theorem 4.5.)

Fact 2.4. *(Piecewise characterization theorem [6]). Let $L \subseteq \Sigma^*$. The following statements are equivalent:*

(i) $L \in \mathsf{REG}/k$.

(ii) There is a mapping $c : \mathbb{N} \Rightarrow \{0, ..., 2^k - 1\}$ *and there are regular languages* $A_0, ..., A_{2^k-1} \in \Sigma^*$ *such that* $L \cap \Sigma^n = A_{c(n)} \cap \Sigma^n$ *for all* $n \in \mathbb{N}$.

Note that an immediate implication of (2.15) is that finite automata with a single bit of prefix advice have strictly more language recognition power than the ordinary finite automata:

$$\mathsf{REG} = \mathsf{REG}/0 \subsetneq \mathsf{REG}/1. \tag{2.16}$$

$REG/1$ is actually big enough to contain all tally languages (*i.e.* the languages over a single letter alphabet) including the nonrecursive ones.

Based on the characterization of REG/k cited in Fact 2.4, Damm and Holzer also proposed a method for showing that certain languages can not be recognized by finite automata with the help of constant-length advice while actually showing that

$$\{a^n b^n \mid n \in \mathbb{N}\} \notin \mathsf{REG}/const, \tag{2.17}$$

which require reference to van der Waerden's theorem which guarantees the existence of monochromatic arithmetic progression under finite coloring of natural numbers in addition to the pumping lemma for regular languages. (For a detailed proof of van der Waerden's theorem, see *e.g.* [24] or [25].)

A similar line of reasoning can be applied for showing

$$\{a^n b^n c^n \mid n \in \mathbb{N}\} \notin \mathsf{CFL}/const. \tag{2.18}$$

Based on these two statements, it is possible to separate the first three levels of Chomsky hierarchy relative to constant-length advice:

$$\mathsf{REG}/const \subset \mathsf{CFL}/const \tag{2.19}$$

$$\mathsf{CFL}/const \subset \mathsf{CS}/const. \tag{2.20}$$

In order to separate the context-sensitive and recursively enumerable languages relative to constant-length advice, Damm and Holzer formed a chain of inclusions:

$$\mathsf{CS}/const \subset \mathsf{DSPACE}(n^2)/const \subset \mathsf{DSPACE}(2^n)/const \subset \mathsf{RE}/const \tag{2.21}$$

The leftmost inclusion is due to the definition of a linear bounded automaton (see *e.g.* [26]) and Savitch's famous theorem (See *e.g.* [18] or [1]) that relates nondeterministic and deterministic space complexity classes. The next inclusion is an application of a result obtained by Mundhenk and Schuler which shows the existence of a nonuniform space hierarchy for space constructible functions. (See [27] for details.) Finally the last inclusion is based on the fact that a Turing machine has no space limit. Hence, this concludes Chomsky Hierarchy relative to constant-length advice:

$$\mathsf{REG}/const \subset \mathsf{CFL}/const \subset \mathsf{CS}/const \subset \mathsf{RE}/const. \tag{2.22}$$

In order to be used in their analysis of Chomsky Hierarchy relative to polynomial-length advice, Damm and Holzer introduced regular, context-free, context-sensitive and recursively enumerable cost of a language, as nonuniform measures of complexity that depends on the nonuniform definition of minimal size of the generating grammars of each type. Below we import their definition for regular cost and related classes of languages.

Definition 2.5. *([6]) Given a language $L \subseteq \Sigma^*$, we define the regular cost of L as the function $reg_L : \mathbb{N} \to \mathbb{N}$ given by*

$$reg_L(n) = min\{|G| \mid G \text{ is a regular grammar such that } L(G) \cap \Sigma^n = L \cap \Sigma^n\}$$

where $|G|$ denotes the size of G and is defined as the cardinality of the production set of G.

The classes of languages with constant and polynomial regular costs are then defined as:

(i) $Const_{reg} = \{L \mid reg_L(n) = O(1)\}$,

(ii) $Poly_{reg} = \{L \mid \exists k : reg_L(n) = O(n^k)\}$.

The context-free, context-sensitive and recursively enumerable cost functions ($cf_L(n)$, $cs_L(n)$, $re_L(n)$) of a language L are defined analogously as well as the related classes of languages, $Const_{cf}$, $Poly_{cf}$, $Const_{cs}$, $Poly_{cs}$, $Const_{re}$, $Poly_{re}$. As steps of proving the separation of the levels of Chomsky hierarchy relative to polynomial-length advice, Damm and Holzer examined the relations among these classes in addition to the classes of languages that each type of automata can recognize with the help of polynomial-length advice.

All of the relations obtained in this while are visible in their inclusion diagram that we cite as Figure 2.2. We will not list all the relations separately and just note some interesting ones without going into details about their proofs. For further details, the reader may refer to the original text in [6].

For each level of Chomsky Hierarchy, it turns out that the classes associated with constant nonuniform cost in the sense defined above, are equivalent to the classes of languages that can be recognized with the help of constant-length advice by the appropriate type of automaton associated with that level of the hierarchy.

$$\mathsf{REG}/const = Const_{reg} \tag{2.23}$$

$$\mathsf{CFL}/const = Const_{cf} \tag{2.24}$$

$$\mathsf{CS}/const = Const_{cs} \tag{2.25}$$

$$\mathsf{RE}/const = Const_{re} \tag{2.26}$$

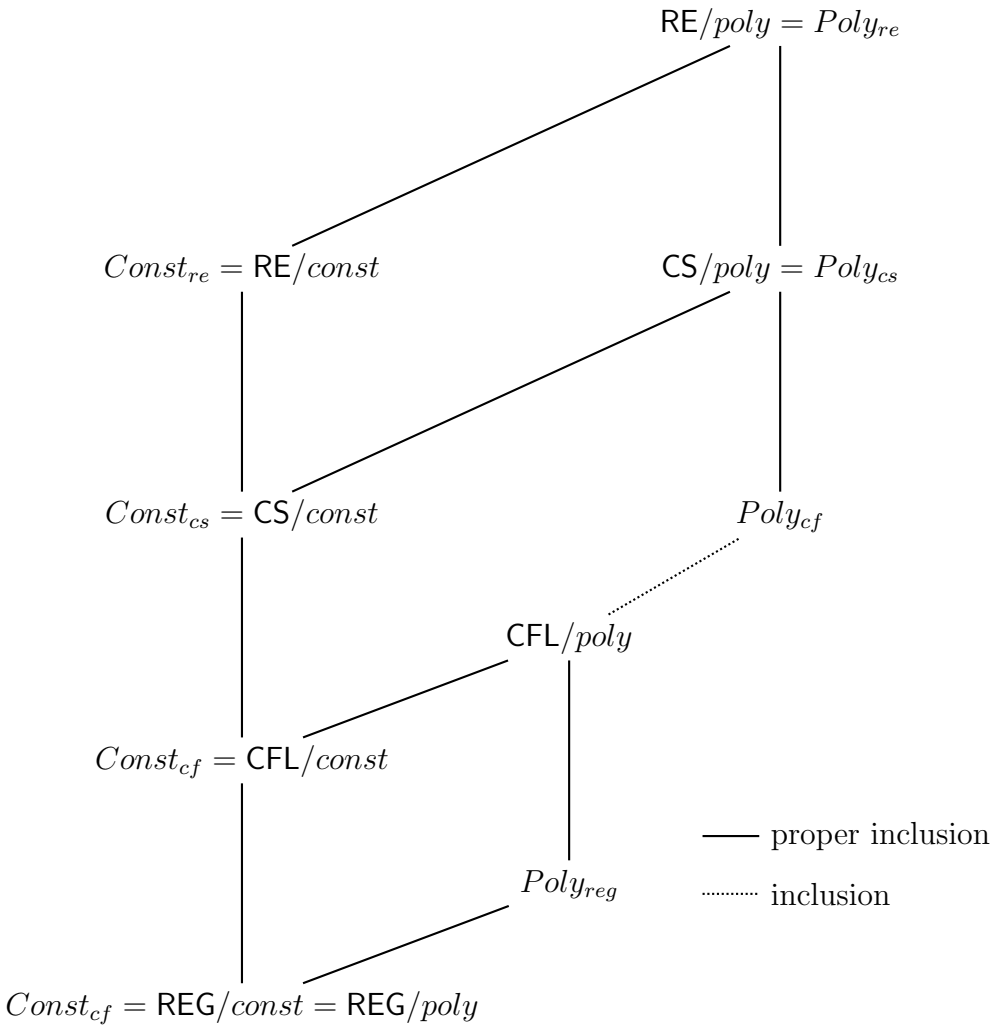

Figure 2.2. The relations among the classes of languages recognized with prefix advice.

These statements can be justified by the piecewise characterization of each class and the fact that there can be only a finite number of devices of each type that would meet the requirement of a constant bound on the associated measure of cost. The same is not always true for the classes associated with polynomial cost and polynomial-length advice:

$$\mathsf{REG}/poly \subset Poly_{reg} \tag{2.27}$$

$$\mathsf{CFL}/poly \subseteq Poly_{cf} \tag{2.28}$$

$$\mathsf{CS}/poly = Poly_{cs} \tag{2.29}$$

$$\mathsf{RE}/poly = Poly_{re} \tag{2.30}$$

The first one of these statements is justified by the linear regular cost of the language $\{a^n b^n \mid n \in \mathbb{N}\}$ which does not belong to $\mathsf{REG}/poly$. The inclusion pointed out in the second statement is obvious. Although not shown to be so, Damm and Holzer conjectured this inclusion to be a proper one. The latter two statements are based on the fact that a Turing machine (or a linear bounded automaton) can read the advice on its input tape several times and that it can simulate the derivation of an unrestricted (or context-sensitive) grammar within a given amount of space.

The classes of languages associated with polynomial regular and context-free costs are shown to be included in the classes of languages which are respectively context-free and context-sensitive relative to polynomial-length advice:

$$Poly_{reg} \subset \mathsf{CFL}/poly \tag{2.31}$$

$$Poly_{cf} \subset \mathsf{CS}/poly \tag{2.32}$$

In both cases, the inclusion is shown to be proper by providing a language which is a member of the difference set. In the former case, this language is $\{ww^R \mid w \in \{a,b\}^*\}$ which is known to have exponential regular cost and in the latter, it is the language $\{ww \mid w \in \{a,b\}^*\}$ which is known to have exponential context-free cost. (See [28] for both.)

The classes of regular languages relative to constant and polynomial-length advice were shown to be equal to each other. The separation results below shows that the advantages of polynomial-length advice are better utilized in the higher levels of the hierarchy.

$$\mathsf{CFL}/const \subset \mathsf{CFL}/poly \tag{2.33}$$

$$\mathsf{CS}/const \subset \mathsf{CS}/poly \tag{2.34}$$

$$\mathsf{RE}/const \subset \mathsf{RE}/poly \tag{2.35}$$

Finally, based on the relations mentioned above and on the nonuniform space hierarchy shown by Mundhenk and Schuler ([27]), the separation of the Chomsky hierarchy relative to polynomial-length advice is concluded as follows:

$$\mathsf{REG}/poly \subset \mathsf{CFL}/poly \subset \mathsf{CS}/poly \subset \mathsf{RE}/poly. \tag{2.36}$$

2.3. Advice on a Separate Track

The advised finite automata model which we call "finite automata with advice track" is first defined in [7] where Tadaki, Yamakami and Lin introduced a novel way of providing advice for computing devices in which instead of coding the supplementary information into an advice string and placing it on the input tape as a prefix of the original input, they suggested splitting the input tape into two tracks, one for holding the original input and the other for the advice string. (See Figure 2.3.)

Other than this obvious difference, the model shares common characteristics of advised computation in the sense defined by Karp and Lipton: advice depends only on the length of the actual input and it is provided in advance of the execution of the device. This model of advised finite automata, together with its probabilistic and quantum variants are later examined in a series of papers ([8–10, 12, 13]) by Yamakami and many relations have been established among the classes of languages that can be recognized in each of these settings.

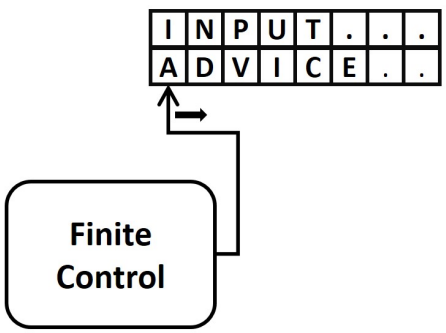

Figure 2.3. Schematic description of a finite automaton with advice track.

2.3.1. Deterministic Automata with Advice Track

It must be noted that unlike the advice prefix model, in this setup the advice string can be scanned in parallel with the actual input by a single tape head and hence it becomes possible to meaningfully utilize advice strings whose lengths grow linearly

in terms of the input length. As a reflection, the class of languages recognizable by deterministic finite automata with an advice track is denoted by REG/n in contrast to the class of languages recognizable by deterministic finite automata with prefix advice which is denoted by $\mathsf{REG}/const$ or by REG/k depending on the context. (In the sequel, we will sometimes need to specify the size of the alphabet supporting the strings on the advice track. In such cases, the number of elements in the advice alphabet will also appear in the class name, *e.g.* $\mathsf{REG}/n(2)$ is the subset of REG/n where advice strings are restricted to be on a binary alphabet.)

Placing advice on a separate track and scanning it in parallel with the actual input brings in an advantage over the prefix advice model in utilizing the additional information within the advice string. The language $\{a^m b^m \mid m > 0\}$, for instance, was stated in (2.17) not to be in REG/k for any constant k, whereas it can easily be recognized by a finite automaton with an advice track. On the other hand, an automaton taking constant-length advice in the prefix format can be converted easily to one that reads it from a separate track. (Simply run in parallel copies of the original automaton each with different initial states and when the end of the constant-length advice is seen on the advice track pick that copy pointed by the constant-length advice as the one to rule the rest of the computation.) Hence, finite automata with advice tracks are strictly more powerful than finite automata with prefix advice which in turn are more powerful than the ordinary finite automata.

$$\mathsf{REG} \subset \mathsf{REG}/k \subset \mathsf{REG}/n. \tag{2.37}$$

On the other hand, as shown in [7] by Tadaki et al., recognizing all context-free languages is beyond the limits of this power:

$$\mathsf{CFL} \not\subseteq \mathsf{REG}/n. \tag{2.38}$$

This is justified by showing that the language $\texttt{EQUAL} = \{w|\ w \in \{a,b\}^* \text{ and } |w|_a = |w|_b\}$, which is well known to be context-free, can not be recognized by a finite au-

tomaton with advice track. To show this fact Tadaki et al. employed an argument that exploits the inability of a finite automaton with advice track to distinguish long enough strings with respect to the language EQUAL in a similar sense of distinguishability that is defined (see *e.g.* [17]) for ordinary finite automata. (This argument was later generalized in [8] with the name "swapping lemma for regular languages". We will cite it as Fact 2.6 below.)

Also in [7], Tadaki et al showed that REG/n is equivalent to $\mathsf{1\text{-}DLIN}/lin$, the set of languages that can be recognized by linear time one tape deterministic Turing machines which are assisted by linear-length advice strings placed on a separate track of the input tape:

$$\mathsf{REG}/n = \mathsf{1\text{-}DLIN}/lin. \tag{2.39}$$

For this result, it is first shown that given a linear-time one tape deterministic Turing machine, it is possible to construct an equivalent machine which uses a technique called "folding" in order to reorganize the original machine's tape content, (including the advice track) during its computation into its input area in a way that resembles a multi track tape. Then the desired conclusion is reached by citing an earlier result by Hennie ([29]) which states $\mathsf{REG} = \mathsf{1\text{-}DLIN}$.

In [8] Yamakami pointed out the fact that pumping lemmata for regular and context-free languages (see *e.g.* [17]) are of no help for showing that certain languages can not be recognized by finite and pushdown automata in the presence of advice. This is because "pumping" brings in a change in the length of the words in consideration which should be matched with a change in the advice string for the pumped words and this violates the original flow of the pumping argument. As substitutes of pumping lemmata for use in showing such nonmembership results in the presence of advice, Yamakami instead introduced a set of useful lemmata which are called swapping lemmata for regular and context-free languages. (It must be noted here that these lemmata can be used as an alternative tool for unadvised cases as well.)

The simple form of swapping lemma for regular languages is quoted from [8] below, in Fact 2.6. The proof of this statement is an application of pigeonhole principle showing that a finite automaton with constantly many states can not distinguish every pair of sufficiently large strings.

Fact 2.6. *(Swapping Lemma for Regular Languages [8]) Let L be any infinite regular language over an alphabet Σ with $|\Sigma| \geq 2$. There exists a positive integer m (called a swapping-lemma constant) such that, for any integer $n \geq 1$ and any subset S of $L \cap \Sigma^n$ of cardinality more than m, the following condition holds: for any integer $i \in [0, n]_{\mathbb{Z}}$, there exist two strings $x = x_1x_2$ and $y = y_1y_2$ in S with $|x_1| = |y_1| = i$ and $|x_2| = |y_2|$ satisfying that*

(i) $x \neq y$,
(ii) $y_1x_2 \in L$, *and*
(iii) $x_1y_2 \in L$.

This idea is further generalized in [8] to obtain a more general form of the swapping lemma where the words in consideration are split into any fixed number of blocks (instead of two) and one of them is used for swapping. This form of the lemma is given as Fact 2.7 below.

Fact 2.7. *(Swapping Lemma for Regular Languages [8]) Let L be any infinite regular language over an alphabet Σ with $|\Sigma| \geq 2$. There is a positive integer m (called a swapping-lemma constant) such that, for any number $n \geq 1$, any set $S \subseteq L \cap \Sigma^n$ and any series $(i_1, i_2, ..., i_k) \in ([1, n]_{\mathbb{Z}})^k$ with $\sum_{j=1}^{k} i_j \leq n$ for a certain number $k \in [1, n]_{\mathbb{Z}}$, the following condition holds: If $|S| > m$ then there exist two strings $x = x_1x_2 \cdots x_{k+1}$ and $y = y_1y_2 \cdots y_{k+1}$ in S with $|x_{k+1}| = |y_{k+1}|$ and $|x_{j'}| = |y_{j'}| = i_{j'}$ for each index $j' \in [1, k]_{\mathbb{Z}}$ such that, for every index $j \in [1, k]_{\mathbb{Z}}$*

(i) $x \neq y$,
(ii) $x = x_1 \cdots x_{j-1}y_jx_{j+1} \cdots x_{k+1} \in L$, *and*
(iii) $x = y_1 \cdots y_{j-1}x_jy_{j+1} \cdots y_{k+1} \in L$.

In order to prove that a language L is not in REG/n, one needs first to pick the swapping lemma of the appropriate form. Then in order to show that assuming $L \in \mathsf{REG}/n$ would lead to a contradiction, a sufficiently large number should be picked as the length of the member words in consideration so that one can define a sufficiently large subset (as described in the lemma) of member words with this length. Then it suffices to show that no matter how the words in this subset are put into blocks, swapping a pair of these blocks would produce a word not in L.

In the same context, Yamakami also introduced a swapping lemma for the context-free languages, which we cite below as Fact 2.8 where the term $S_{i,u}$, for $S \subseteq \Sigma^*$, $i \in \mathbb{N}$ and $u \in \Sigma^*$ for an alphabet Σ, denotes the set of words $w \in S$ such that the subword, $w_{i+1}, \cdots, w_{i+|u|}$ of w, is equal to the word u.

Fact 2.8. *(Swapping Lemma for Context-Free Languages [8]) Let L be any infinite context-free language over an alphabet Σ with $|\Sigma| \geq 2$. There is a positive number m that satisfies the following. Let n be any positive number at least 2, let S be any subset of $L \cap \Sigma^n$, and let $j_0, k \in [2,n]_{\mathbb{Z}}$ be any two indices satisfying that $k \geq 2j_0$ and $|S_{i,u}| < |S|/m(k - j_0 + 1)(n - j_0 + 1)$ for any index $i \in [1, n - j_0]_{\mathbb{Z}}$ and any string $u \in \Sigma^{j_0}$. There exist two indices $i \in [1,n]_{\mathbb{Z}}$ and $j \in [j - 0, k]_{\mathbb{Z}}$ with $i + j \leq n$ and two strings $x = x_1x_2x_3$ and $y = y_1y_2y_3$ in S with $|x_1| = |y_1| = i$, $|x_2| = |y_2| = j$ and $|x_3| = |y_3|$ such that*

(i) $x \neq y$,
(ii) $x = x_1y_2x_3 \in L$, *and*
(iii) $x = y_1x_2y_3 \in L$.

The proof of this lemma requires an analysis of a restricted form of a nondeterministic pushdown automaton and its stack's behavior. Nonmembership results for CFL/n can be obtained with reference to it, as a substitute for the pumping lemma for context-free languages, and with a similar reasoning that was described for REG/n above.

The reader may refer to [8] for details of the proofs for these lemmata and detailed examples where this set of tools are employed for showing that certain languages are not in REG/n or CFL/n.

2.3.2. Probabilistic Automata with Advice Track

Yamakami further extended his analysis of advised computation with track model in [9], where in addition to the deterministically chosen advice, he also studied randomly chosen advice, allowing the underlying machine models to recognize languages with some probability of error. He showed that such randomized advice significantly extends the language recognition power of the underlying machines such as one-tape linear-time Turing machines and one-way finite automata.

The language families associated with linear-time unadvised Turing machines which are taken into account in [9] are 1-DLIN (deterministic), 1-BPLIN (bounded-error probabilistic), 1-PLIN (unbounded-error probabilistic), and 1-$\mathsf{C_{=}LIN}$ (error probability exactly 1/2). In the presence of deterministically chosen linearly sized advice, the language families associated with the same model of underlying machines are named respectively as 1-DLIN/lin, 1-BPLIN/lin, 1-PLIN/lin and 1-$\mathsf{C_{=}LIN}/lin$.

For cases where the advice is chosen randomly according to a certain probability distribution, the notation is extended by use of the mark R (standing for random) preceding the amount of advice so that the language families associated with the same set of machine models are named respectively as 1-$\mathsf{DLIN}/Rlin$, 1-$\mathsf{BPLIN}/Rlin$, 1-$\mathsf{PLIN}/Rlin$ and 1-$\mathsf{C_{=}LIN}/Rlin$. The language families associated with finite automata and push-down automata equipped with randomly chosen advice are named by following the same pattern as REG/Rn or CFL/Rn.

Yamakami examined the exact nature of the relations among these families of languages and obtained various results indicating separations and collapses. Below, in Figure 2.4 we import a drawing from [9] which provides a summary of these results.

In addition to the set of relations shown in Figure 2.4, most of which demonstrate the power of randomized advice, Yamakami also established an immediate limit to this power with the following result which states that even with randomized advice finite automata can not recognize all context-free languages.

$$\mathsf{CFL} \not\subseteq \mathsf{REG}/Rn. \tag{2.40}$$

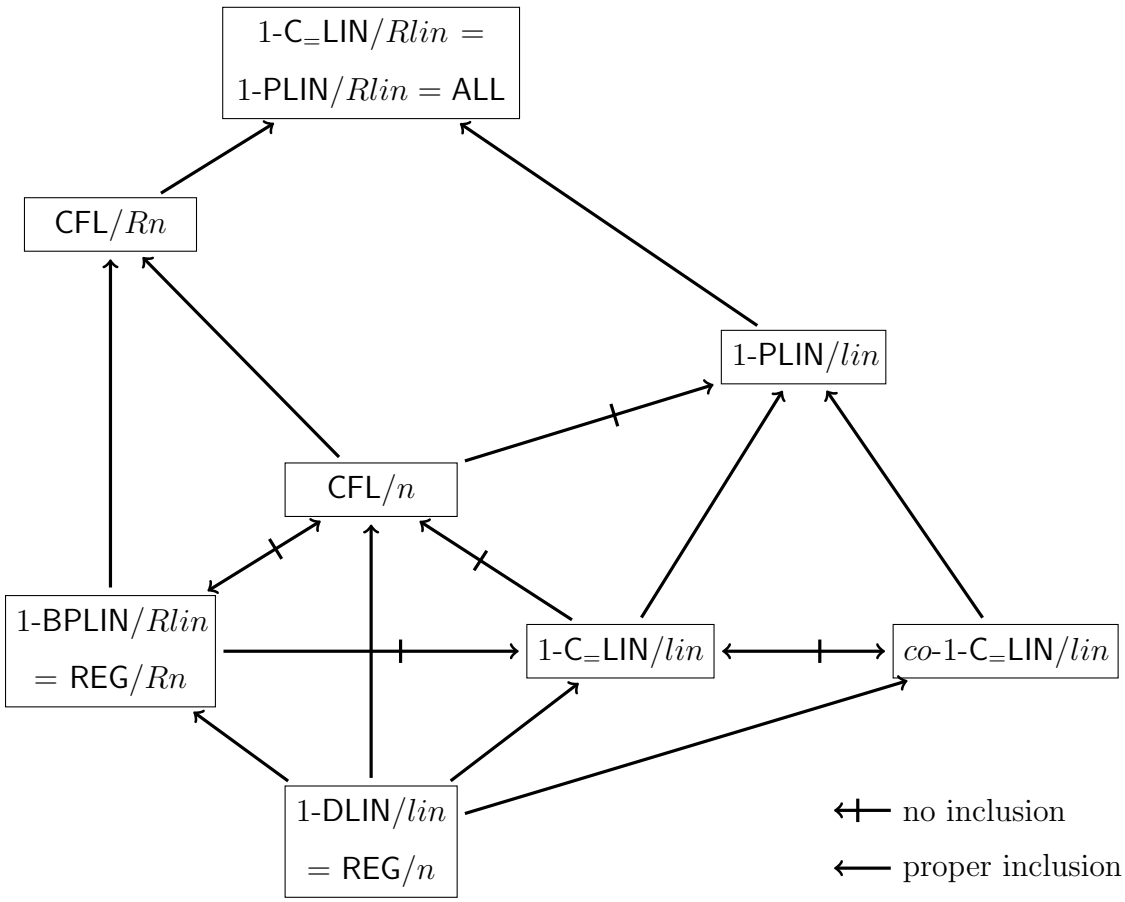

Figure 2.4. The relations among the classes of languages recognized with deterministic and randomized advice provided on a separate track.

In his analysis of the power and weakness of the randomized advice, Yamakami introduced a pair of lemmas which provide simple yet powerful characterization for the classes REG/n and REG/Rn. We quote them as Fact 2.9 and Fact 2.10 below for later reference.

Fact 2.9. *([9]) For any language S over an alphabet Σ, the following two statements are equivalent. Let $\Delta = \{(x, n) \in \Sigma^* \times \mathbb{N} \mid |x| \leq n\}$.*

(i) *S is in REG/n.*

(ii) *There is an equivalence relation $\equiv_S$ over Δ such that*

  (a) *the total number of equivalence classes in $\Delta/\equiv_S$ is finite, and*

  (b) *for any length $n \in \mathbb{N}$ and any two strings $x, y \in \Sigma^*$ with $|x| = |y| \leq n$, the following holds: $(x, n) \equiv_S (y, n)$ iff, for all z with $|xz| = n$, $xz \in S \Leftrightarrow yz \in S$.*

Fact 2.10. *([9]) Let A be any language over an alphabet Σ. The following two statements are equivalent.*

(i) *A is in REG/Rn.*

(ii) *There exist a 1dfa M, an advice alphabet Γ, and an error bound $\epsilon \in [0, 1/2)$ that satisfy the following condition: for every probability ensemble $\{\mu_n\}_{n\in\mathbb{N}}$ over Σ^* there exists an advice function $h : \mathbb{N} \to \Gamma^*$ such that M, when provided with $h(n)$ as advice, correctly decides whether x is a member of A or not with probability more than $1 - \epsilon$ for every length $n \in \mathbb{N}$.*

2.3.3. Reversible and Quantum Automata with Advice Track

In [12], Yamakami extended his analysis of advised finite automata to the cases of reversible and quantum finite automata with advice. In this context, one-way deterministic reversible finite automata (1rfa) and one-way measure-many quantum finite automata (1qfa) are chosen as base models of computation as the relative simplicity of these models are expected to leave a larger room for examining the power and limits of advice. (See [30–32], for formal definitions and fundamental characteristics of these models.)

The language families associated with 1rfa's and 1qfa's augmented with deterministic advice are named respectively as 1-RFA/n and 1-QFA/n. In order to capture

the essential characteristics of these language families, Yamakami proved two main theorems, one for each of the two models in consideration, in which he showed machine-independent, algebraic necessary conditions for languages to be recognized in these settings with the help of deterministic advice. We import them below as Fact 2.11 and Fact 2.12 where the term "closeness relation" is used to denote a reflexive, symmetric, binary relation and given any closeness relation $\cong_S$, an $\cong_S$-discrepancy set denotes a set S satisfying that, for any two elements $x, y \in S$, if x and y are "different" elements, then $x \ncong_S y$.

Fact 2.11. *(A necessary condition for* 1-QFA/n *[12]) Let S be any language over alphabet Σ and let $\Delta = \{(x,n) \in \Sigma^* \times \mathbb{N} \mid |x| \leq n\}$. If S belongs to* 1-QFA/n*, then there exist two constants $c, d \in \mathbb{N}^+$, an equivalence relation $\equiv_S$ over Δ, a partial order $\leq_S$ over Δ, and a closeness relation $\cong_S$ over Δ that satisfy the seven conditions listed below. In the list, we assume that $(x,n),(y,n) \in \Delta, z \in \Sigma^*$, and $\sigma \in \Sigma$ with $|x| = |y|$.*

(i) *The cardinality of the set $\Delta/\equiv_S$ of equivalence classes is at most d.*
(ii) *If $(x,n) \cong_S (y,n)$, then $(x,n) \equiv_S (y,n)$.*
(iii) *If $|x\sigma| \leq n$, then $(x\sigma,n) \leq_S (x,n)$ and, if $|x| = n > 0$, then $(x,n) <_S (\lambda,n)$.*
(iv) *When $(x,n) =_S (xz,n)$ and $(y,n) =_S (yz,n)$ with $|xz| \leq n$, $(xz,n) \cong_S (yz,n)$ implies $(x,n) \equiv_S (y,n)$.*
(v) *$(x,n) \equiv_S (y,n)$ iff $S(xz) = S(yz)$ for all strings $z \in \Sigma^*$ with $|xz| = n$.*
(vi) *Any strictly descending chain (with respect to $<_S$) in Δ has length at most c.*
(vii) *Any $\cong_S$-discrepancy subset of Δ has cardinality at most d.*

Fact 2.12. *(A necessary and sufficient condition for* 1-RFA/n *[12]) Let S be any language over alphabet Σ and define $\Delta = \{(x,n) \mid x \in \Sigma^*, n \in \mathbb{N}, |x| \leq n\}$. The following two statements are logically equivalent.*

(i) *S is in* 1-RFA/n.
(ii) *There are a total order $\leq_S$ over Δ and two equivalence relations $\simeq_S$ and $\equiv_S$ over Δ such that*
  (a) *two sets $\Delta/\simeq_S$ and $\Delta/\equiv_S$ are both finite,*
  (b) *any strictly descending chain (with respect to $<_S$) in Δ has length at most*

2, and

(c) *for any length parameter* $n \in \mathbb{N}$*, any two symbols* $\sigma, \xi \in \Sigma$*, and any three elements* $(x,n),(y,n),(z,n) \in \Delta$ *with* $|x| = |y|$*, the following seven conditions hold.*

i. *If* $|x\sigma| \leq n$*, then* $(x\sigma,n) \leq_S (x,n)$ *and, if* $|x| = n > 0$*, then* $(x,n) <_S (\lambda,n)$*.*

ii. *Whenever* $|x\sigma| \leq n$*,* $(x\sigma,n) \simeq_S (y\sigma,n)$ *iff* $(x,n) \simeq_S (y,n)$*.*

iii. *If* $(x\sigma,n) <_S (x,n) =_S (z,n)$ *with* $|x\sigma| \leq n$*, then* $(x\sigma,n) \not\simeq_S (z,n)$*.*

iv. *In the case where* $(\lambda,n) =_S (x,n) =_S (z,n)$*,* $(x,n) \equiv_S (z,n)$ *iff* $(x,n) \simeq_S (z,n)$*.*

v. *If* $(x\sigma,n) <_S (x,n)$ *and* $(y\xi,n) <_S (y,n)$ *with* $|x\sigma| \leq n$ *and* $|y\xi| \leq n$*, then* $(x\sigma,n) \equiv_S (y\xi,n)$ *iff* $(x\sigma,n) \simeq_S (y\xi,n)$*.*

vi. *If* $(xz,n) =_S (x,n)$ *with* $|xz| = n$*, then* $(xz,n) \equiv_S (x,n)$*.*

vii. *If* $(x,n) \equiv_S (y,n)$*, then* $S(xz) = S(yz)$ *holds for all strings* $z \in \Sigma^*$ *satisfying* $|xz| = n$*.*

Also in [12], Yamakami introduced 1-RFA/Rn and 1-QFA/Rn as the language families associated with 1rfa's and bounded-error 1qfa's augmented with randomized advice. This notation is extended as in 1-QFA$^\diamond$/Qn and 1-QFA$^\diamond$/Rn in order to indicate that the underlying quantum finite automata model uses a rewritable advice track. Finally, for use in cases where unbounded-error language recognition by 1qfa's is in consideration, the notation is extended as in 1-QFA$_{(\mathsf{a(n)},\mathsf{b(n)})}$, in order to indicate the families of languages that can be recognized by 1qfa's which are expected to accept member strings of length n with probability at least $a(n)$ and reject the nonmember strings of length n with probability at least $b(n)$.

In the light of the characterizations of 1-RFA/n and 1-QFA/n cited above, many separations and collapses are obtained by Yamakami, among the above mentioned language families and some of the other language families associated with advised finite automata, pushdown automata and linear-time Turing machines. A summary of these relations are provided in the Figure 2.5 which we cite from [12].

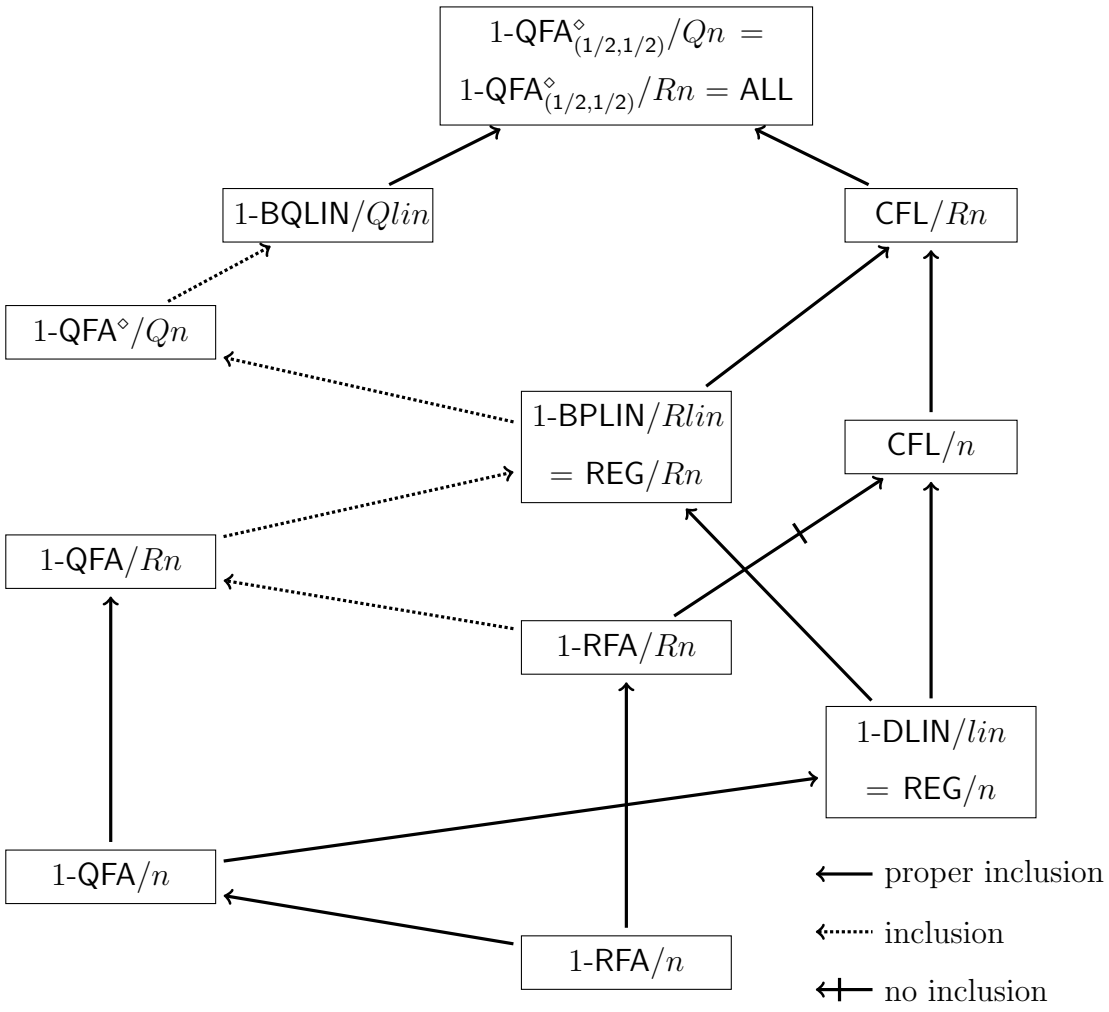

Figure 2.5. The relations among the classes of languages recognized by reversible and quantum finite automata with advice track.

2.3.4. Structural Complexity of Advised Language Families

In [10] and [13], Yamakami examined the structural complexity of the advised language families REG/n and CFL/n in the framework of structural properties such as immunity and pseudorandomness.

Below, we refer [20] for formal definitions of the terms such as immunity, co-immunity, simplicity and bi-immunity which are widely used in this context. (For further discussion on these concepts the reader may refer also to [33].)

Definition 2.13. *([20]) Let C be a class of sets,*

- *A set L is C-immune if and only if it is infinite and no infinite subset of L belongs to C.*
- *A set L is C-co-immune if and only if its complement $\overline{L}$ is C-immune.*
- *A set is C-simple if and only if it is in C and is C-co-immune.*
- *A set L is C-bi-immune if and only if both L and $\overline{L}$ are C-immune.*

In [10], Yamakami introduced primeimmunity as a weaker form of immunity where not all but only polynomially dense subsets of the language in consideration are taken into account. In close relation to immunity, he also examined computational randomness in the framework of pseudorandomness and pseudorandom generators.

The notion of pseudorandomness is introduced as a non-asymptotic variant of the notions of randomness introduced in [34,35]. We cite this definition below in Definition 2.14 where the characteristic function χ_L of a language L is defined as $\chi_L(x) = 1$ if $x \in L$ and $\chi_L(x) = 0$ otherwise, For every input string x.

Definition 2.14. *([10]) A language L is said to be C-pseudorandom if, for every language A in C, the characteristic function χ_A of A agrees with the characteristic function χ_L of L on nearly (with a negligible margin of error) half of the strings of each length.*

Yamakami also applied Yao's formulation ([36]) of pseudorandom generators to the framework of formal language and automata theory where the adversaries to be fooled by the generated sequences are represented in the form of languages.

Definition 2.15. *([13]) Given an arbitrary alphabet Σ, a (single-valued total) function $G : \Sigma^* \to \Sigma^*$, which stretches n-symbol seeds to $s(n)$-symbol strings, is said to fool a language A over Σ if the characteristic function χ_A of A cannot distinguish between the output distribution of $\{G(x)\}_{x\in\Sigma^n}$ and a truly random distribution of $\{y\}_{y\in\Sigma^{s(n)}}$ with non-negligible success probability. [We call G a pseudorandom generator against language family C if G fools every language A over Σ in C. A generator G with the stretch factor $s(n) = n + 1$ is called almost one-to-one if it is one-to-one for all but a negligible fraction of its domain instances.]*

The idea of pseudorandom generators and pseudorandomness of a language are shown to be linked in a way that any generator which has a small stretch factor and which is almost one to one is a pseudorandom generator if and only if its range is pseudorandom.

Below we highlight some of the results Yamakami obtained in [10] and [13] which are interesting in the context of advised automata.

Fact 2.16. *([10, 13])*

(i) *There exists a* REG*-immune language in* $\mathsf{CFL} \setminus \mathsf{REG}/n$.

(ii) *There exists a* REG*-bi-immune language that can be computed deterministically using logarithmic space.*

(iii) *There exists a* REG/n*-bi-primeimmune language in* CFL.

(iv) *There exists a* REG/n*-pseudorandom language in* CFL.

(v) *There exists an almost one to one pseudorandom generator against* REG/n*, computable by nondeterministic pushdown automata equipped with a write only output tape.*

(vi) *There exists an almost one to one pseudorandom generator with stretch factor* $n + 1$ *against* CFL/n *in the intersection of* FL*, the logarithmic space function*

class and CFLMV(2)/n*, a functional analogue of* CFL(2)/n*, the 2-conjunctive closure of* CFL/n*, or in other words, the set of languages that can be obtained as the intersection of two languages in* CFL/n*.*

(vii) There is no almost one to one pseudorandom generator with stretch factor $n+1$ *against* CFL *in* CFLMV*, multiple-valued partial* CFL*-function class.*

2.4. Advice on Additional Two-way Tapes

As part of their research on the notion of nonconstructive computation (see *e.g.* [37,38] or Definition 2.17) Freivalds introduced and examined another model of advised finite automata in [14], which provides means for quantitatively measuring the amount of nonconstructivity in a proof. In this model, which incorporates one or more separate tapes for the advice, the automaton is granted two-way access to both the input and the advice tapes and unlike the advice prefix and advice track models, Freivalds' model requires the advice string for inputs of length n to be helpful for all shorter inputs as well.

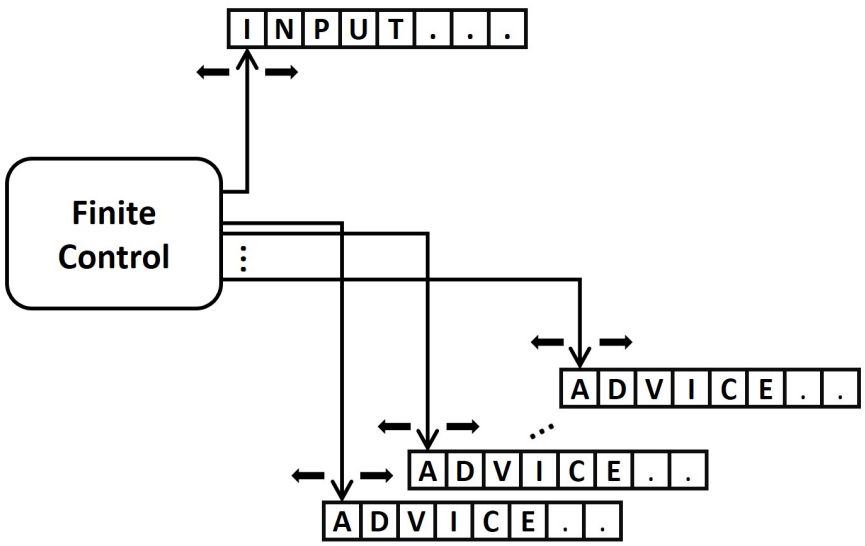

Figure 2.6. Schematic description of a finite automaton with multiple two-way advice tapes.

This model was later used by Agadzanyan and Freivalds in [15], for an analysis of the cases where the given advice contains zero information about the input word and the language to be recognized. In this context, infinite random sequences are considered as advice to finite automata and the term "finite state automata with intuition" is used to denote the resulting model of the advised automaton. Later, in [16], Freivalds revisited this idea and further extended the analysis where the term "finite state automata with written random bits" is preferred for denoting a slightly modified version of finite automata which takes infinite random sequences as advice.

2.4.1. Nonconstructive Language Recognition with Finite Automata

Language recognition with a certain amount of nonconstructivity is defined by Freivalds in [14] as follows.

Definition 2.17. *([14]) We say that an automaton A recognizes the language L nonconstructively with nonconstructivity $d(n)$ if the automaton A has an input tape where a word x is read and an additional input tape for nonconstructive help y with the following property. For arbitrary natural numbers n there is a word y of the length not exceeding $d(n)$ such that for all words x whose length does not exceed n the automaton A on the pair (x, y) produces the result 1 if $x \in L$, and A produces the result 0 if $x \notin L$. Technically, the word y can be a tuple of several words and may be placed on separate additional input tapes. In this case, $d(n)$ is the upper bound for the total of the lengths of these words.*

Based on this notion of nonconstructivity, and referring many times to the properties of Martin-Löf (see *e.g.* [39]) and De Brujin (see *e.g.* [40]) sequences, Freivalds showed the statements listed in Fact 2.18 to be true in [14].

Fact 2.18. *([14])*

(i) There exists a nonregular (and even a nonrecursive) language L such that it can be nonconstructively recognized with nonconstructivity n.

(ii) *There exist nonregular languages that can be nonconstructively recognized by a two-way finite automaton with nonconstructivity n but not with nonconstructivity $n - h(n)$, where $h(n)$ is a total function such that $\log_2 n = o(h(n))$.*

(iii) *For arbitrary natural numbers k, there exist nonregular languages that can be nonconstructively recognized with nonconstructivity not exceeding $n^{1/k}$.*

(iv) *There exist a nonregular language L and a function $g(n)$ such that L can be nonconstructively recognized with nonconstructivity $g(n)$ where $\log n \leq g(n) \leq (\log n)^2$.*

(v) *If a language L can be nonconstructively recognized with a nonconstructivity bounded by a function $d(n) = o(\log n)$, then L is regular.*

(vi) *There exist a nonrecursive language L and a function $g(n)$ such that L can be nonconstructively recognized by a finite automaton with a nonconstructivity $g(n) \in polylog(n)$.*

(vii) *Every language L over the c-ary alphabet $\{0, 1, \cdots, c-1\}$ can be recognized nonconstructively with nonconstructivity $O(c^n)$.*

(viii) *There exists a language L such that it cannot be nonconstructively recognized with nonconstructivity less than 2^n.*

(ix) *There exists a language L in a binary alphabet such that it cannot be nonconstructively recognized by a finite automaton with a nonconstructivity less than $\Omega(2^n)$.*

2.4.2. Infinite Random Sequences as Advice for Finite Automata

The use of arbitrary infinite random sequences as advice to finite automata was first examined by Agadzanyan and Freivalds in [15] in order to set up a case where a finite automaton is given an advice that contains no information about the actual input. In [16], Freivalds revisited this idea, using a slightly modified model and terminology and extended the analysis in [15].[1]

[1]We will mostly cite [16] below, for the definitions and the results obtained in this context. The reader, however, may also refer to [15] for similar content in some cases, where a slightly different terminology is used.

In this setup, the advice is required to be an arbitrary infinite random sequence but the finite automaton can use only a finite initial fragment of this sequence the size of which depends on the automaton. In [15] the infinite random advice is specified to be an arbitrary infinite Martin-Löf random sequence, however the term "primitive Martin-Löf random sequence" is preferred instead in [16], as only that property of a Martin-Löf sequence mentioned in the Definition 2.19 is required for the results obtained in this context.

Definition 2.19. *([16]) An infinite sequence S of bits is a primitive Martin-Löf random sequence if for arbitrary finite binary string w, it is true that S contains infinitely many occurrences of the string w.*

The random sequence used as advice is also required to be infinite to both ends hence to avoid the unintended additional power the model would gain with the ability of simulating a counter on its two-way advice tapes.

Definition 2.20. *([16]) A 2-infinite sequence of bits is a sequence $\{a_i\}$ where $i \in (-\infty, \infty)$ and all $a_i \in \{0, 1\}$.*

Definition 2.21. *([16]) We say that a 2-infinite sequence of bits -$\{a_i\}$- is primitive Martin-Löf random if for arbitrary $i \in (-\infty, \infty)$ the sequence $\{b_n\}$ where $b_n = a_{i+n}$ for all $i \in \mathbb{N}$ is primitive Martin-Löf random, and the sequence $\{c_n\}$ where $c_n = a_{i-n}$ for all $i \in \mathbb{N}$ is primitive Martin-Löf random.*

Finally, "a deterministic finite automaton with written random bits" is defined as follows.

Definition 2.22. *([16]) A deterministic finite automaton with written random bits is a deterministic non-writing 2-tape finite automaton one tape of which contains the input word , and the other tape contains a 2-infinite primitive Martin-Löf random sequence, the automaton is 2-way on every tape, and it stops producing the correct result in a finite number of steps for arbitrary input word. Additionally it is demanded that the head of the automaton never goes beyond the markers showing the beginning and the end of the input word.*

In this context, a language L is said to be recognizable by a deterministic finite automaton A with written random bits if A for arbitrary 2-infinite primitive Martin-Löf random sequence accepts every member of L and rejects every nonmember. Similarly, a language L is said to be enumerable by a deterministic finite automaton A with written random bits if A for arbitrary 2-infinite primitive Martin-Löf random sequence, accepts every member of L and does not accept any nonmember.

The term, "finite automaton with written random bits on unbounded input", is used to denote a slightly different model where the automaton is allowed to go beyond the markers showing the beginning and the end of the input word. In cases where more than one help tapes are in consideration this is made explicit as in "finite automaton with written random bits with 2 help tapes". Both of these variations over the original definition are shown to bring in excessive power of language recognition, hence they are ruled out of the main analysis. This is justified by the following results which are based on the observation that with these variations it would become possible to use the infinite random sequence on the two-way advice tapes for simulating a set of counters which in turn is known (see *e.g.* [41]) to provide a two-way deterministic finite automata sufficient means to simulate a Turing machine.

- A language L is enumerable by a deterministic finite automaton with written random bits on unbounded input if and only if it is recursively enumerable.
- A language L is enumerable by a deterministic finite automaton with written random bits with 2 help tapes if and only if it is recursively enumerable.

Based on these definitions and observations, Freivalds obtained a set of results, which are listed below in Fact 2.23, on the power of deterministic and nondeterministic finite automata with written random bits.

Fact 2.23. *([16])*

(i) There exists a language (e.g. $\mathtt{L} = \{x2x \mid x \in \{0,1\}^*\}$*) that cannot be recognized with a bounded error by a probabilistic 2-way finite automaton while it can be*

recognized by a deterministic finite automaton with written random bits.

(ii) *The unary languages* $\texttt{PERFECT SQUARES} = \{1^n \mid (\exists m)(n = m^2)\}$, $\texttt{PERFECT CUBES} = \{1^n \mid (\exists m)(n = m^3)\}$, *and* $\texttt{PRIMES} = \{1^n \mid n \text{ is prime }\}$ *can be recognized by a deterministic finite automaton with written random bits.*

(iii) *Every* $L \in \mathsf{NP}$ *is reducible by a deterministic log-space bounded Turing machine to a language* L *such that* L *is enumerable by a deterministic finite automaton with written random bits.*

(iv) *If a language* L *is enumerable by a nondeterministic finite automaton with written random bits then* $L \in \mathsf{NP}$.

(v) *If a language* L *is recognizable by a nondeterministic finite automaton with written random bits then* $L \in \mathsf{NP} \cap co\text{-}\mathsf{NP}$.

(vi) *Every language enumerable by a deterministic finite automaton with written random bits is also recognizable by a nondeterministic finite automaton with written random bits if and only if* $\mathsf{P} = \mathsf{NP}$.

(vii) *If a language* L *is enumerable by a nondeterministic finite automaton with written random bits then* L *is also enumerable by a deterministic finite automaton with written random bits.*

(viii) *If a language* L *is recognizable by a nondeterministic finite automaton with written random bits then* L *is also recognizable by a deterministic finite automaton with written random bits.*

3. FINITE AUTOMATA WITH ADVICE TAPES

In this chapter, we propose a new architecture for advised finite-state computation where we place the advice string on a separate one-way tape and thereby enabling the machine to pause on the input tape while processing the advice, or vice versa. (Examples of finite-state machines with such a separate tape for *untrusted* help can be seen in [42].) This model provides a more flexible access to the advice than the advice prefix and advice track setups mentioned above, in Chapter 1 and it differs from the alternative proposal of Freivalds for advised finite-state automata [14] in the number of allowed advice tapes, and the way in which the advice can be accessed.

We consider many variants of our machines, where the advised automaton is classical or quantum, the tapes can be accessed in various alternative modes, and the advice is deterministic or randomized. The language recognition power of these variants are compared among themselves, and also with the corresponding instances of the advice prefix and the advice track models. Freivalds' model with multiple two-way advice tapes is left out in these comparisons, as it would not be fair to compare the power of this model with the others due to the way language recognition is defined in this model, which requires advice strings provided for inputs of a specific length to be helpful for all shorter inputs as well.

Below in Section 3.1 we will introduce the advice tape model in detail and set the notation for naming the classes of languages that can be recognized by variants of this model. Section 3.2 will present some basic observations before proceeding to the main results. Section 3.3 will focus on the power and weaknesses of variants of deterministic finite automata with advice tape. Having separate tape heads for scanning the input and the advice strings is shown to be advantageous when either of these tape heads is allowed to move bidirectionally or at least stay put. For such settings it is shown that more and more languages can be recognized by allowing longer and longer advice strings so that one can form an infinite hierarchy of language classes in this manner. Section 3.4 will introduce randomness into the model by allowing either of probabilistic

transition functions or randomly picked advice strings. The classes of languages that can be recognized with bounded error in both of these settings are shown to be strictly larger than the class of languages that can be recognized with equal resources in fully deterministic settings. Section 3.5 will provide a brief discussion on quantum finite automata with advice tape which is also shown to recognize more languages (with bounded error) than the deterministic automata with similar resources.

3.1. Basic Notions, Formal Definitions and Notation

We model advice as a string provided on a separate read-only tape. As usual, the content of the advice depends only on the length of the input. Formally, the advice to the automaton is determined by an advice function h, which is a mapping from $\mathbb{N}$ to strings in Γ^*, where Γ is the advice alphabet. This function may or may not be computable.

Our advised machine model is then simply a finite automaton with two tapes. The transition function of a (two-way) *deterministic finite automaton with advice tape* (dfat) determines the next move of the machine based on the current internal state, and the symbols scanned by the input and advice tape heads. Each move specifies the next state, and a head movement direction (right($\mathcal{R}$), left($\mathcal{L}$), or stay-put($\mathcal{S}$)) for each tape. A tape head that is allowed to move in all these directions is called *two-way*. A head that is not allowed to move left is called *one-way*. We may also require a head to be *real-time*, forcing it to move to the right at every step. As will be shown, playing with these settings changes the computational power of the resulting model. We assume that both the input and the advice strings are delimited by special end-marker symbols ($\vdash, \dashv$), beyond which the automaton is not allowed to move its heads. The machine halts and announces the corresponding decision when it enters one of the two special states q_{accept} and q_{reject}.

Unlike Freivalds [14], we do not allow two-way motion of the advice tape head, as permitting this head to make leftward moves would cause "unfair" accounting of the

space complexity of the advised machine.[2]

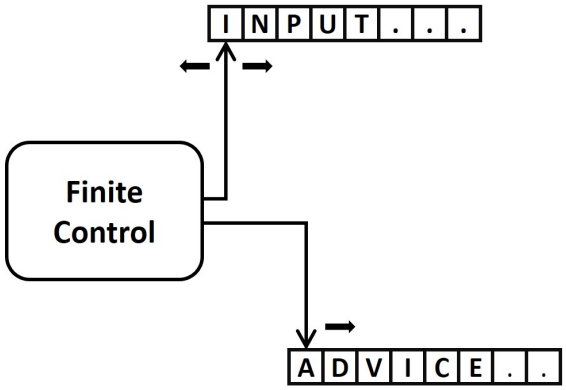

Figure 3.1. Schematic description of a finite automaton with advice tape.

We define the probabilistic and quantum versions of our advised automata by generalizing the definition for deterministic automata in the standard way, see, for instance, [44]. The transition function of a *probabilistic finite automaton with advice tape* (pfat) specifies not necessarily one, but possibly many choices, associated with selection probabilities, for the next move at every step, with the well-formedness condition that the probabilities of these choices always add up to 1. In the case of *quantum finite automata with advice tapes* (qfat's), each such choice is associated not with a probability, but with an amplitude (a real number in the interval [-1,1]).

The presentation of our results on qfat's will not require knowledge of technical details of their definitions such as well-formedness conditions, and we will therefore omit these, referring the reader to [44]. We should stress that there are many mutually inequivalent quantum finite automaton definitions in the literature, and we use the most powerful one [44, 45]. The quantum machines with advice tracks defined in [11] are based on an older model [30], and this difference will be significant in our discussion in Section 3.5.

[2]See Section 5.3.1 of [43] for a discussion of this issue in the context of certificate tape heads.

3.1.1. Deterministic Finite Automata with Advice Tape

Below is a formal definition for a deterministic finite automaton with advice tape.

Definition 3.1. *A deterministic finite automaton with an advice tape is a 9-tuple* $(Q, \Sigma, \Gamma, T_I, T_A, \delta, q_0, q_{accept}, q_{reject})$*, where*

(i) Q *is a finite set of internal states,*

(ii) Σ *is a finite set of symbols called the input alphabet that does not contain the endmarker symbols,* $\vdash$ *and* $\dashv$*, such that* $\Sigma \cap \{\vdash, \dashv\} = \emptyset$ *and* $\Sigma' = \Sigma \cup \{\vdash, \dashv\}$*,*

(iii) Γ *is a finite set of symbols called the advice alphabet that does not contain the endmarker symbols,* $\vdash$ *and* $\dashv$*, such that* $\Gamma \cap \{\vdash, \dashv\} = \emptyset$ *and* $\Gamma' = \Gamma \cup \{\vdash, \dashv\}$*,*

(iv) $T_I \in \{\{\mathcal{L}, \mathcal{S}, \mathcal{R}\}, \{\mathcal{S}, \mathcal{R}\}, \{\mathcal{R}\}\}$ *represents the set of allowed head movements for the input tape,*

(v) $T_A \in \{\{\mathcal{S}, \mathcal{R}\}, \{\mathcal{R}\}\}$ *represents the set of allowed head movements for the advice tape,*

(vi) $\delta : Q \times \Sigma' \times \Gamma' \to Q \times T_I \times T_A$ *is the transition function such that,* $\delta(q_1, \sigma, \gamma) = (q_2, t_I, t_A)$ *implies that when the automaton is in state* $q_1 \in Q$ *and it scans* $\sigma \in \Sigma'$ *on its input tape and* $\gamma \in \Gamma'$ *on its advice tape, a transition occurs which changes the state of the automaton to* $q_2 \in Q$*, meanwhile moving the input and advice tape heads in the directions specified respectively by* $t_I \in T_I$ *and* $t_A \in T_A$*,*

(vii) $q_0 \in Q$ *is the initial state,*

(viii) $q_{accept} \in Q$ *is the accepting state upon entering which the automaton halts and announces that it accepts the input, and*

(ix) $q_{reject} \in Q$ *is the rejecting state upon entering which the automaton halts and announces that it rejects the input.*

A dfat $M = (Q, \Sigma, \Gamma, T_I, T_A, \delta, q_0, q_{accept}, q_{reject})$ is said to accept (reject) a string $x \in \Sigma^*$ with the help of an advice string $a \in \Gamma^*$ if and only if M, when started at its initial state q_0 with $\vdash x \dashv$ on the input tape and $\vdash a \dashv$ on the advice tape and while the tape heads scan the first symbols to the right of the left endmarkers, reaches the accepting state, q_{accept} (q_{reject}), by changing states and moving the input and advice

tape heads as specified by its transition function, δ.

A language L defined on the alphabet Σ, is said to be recognized by such a dfat M with the help of an advice function $h : \mathbb{N} \to \Gamma^*$ if and only if

- $L = \{x \mid M \text{ accepts } x \text{ with the help of } h(|x|)\}$, and
- $\overline{L} = \{x \mid M \text{ rejects } x \text{ with the help of } h(|x|)\}$.

A language L is said to be recognized by such a dfat, M, using $O(f(n))$-length advice if there exists an advice function h with the following properties:

- $|h(n)| \in O(f(n))$ for all $n \in \mathbb{N}$, and
- M recognizes L with the help of $h(n)$.

3.1.2. Probabilistic Finite Automata with Advice Tape

A probabilistic finite automaton with advice tape is defined as follows.

Definition 3.2. *A probabilistic finite automaton with an advice tape is a 9-tuple* $(Q, \Sigma, \Gamma, T_I, T_A, \delta, q_0, q_{accept}, q_{reject})$*, where*

(i) Q *is a finite set of internal states,*

(ii) Σ *is a finite set of symbols called the input alphabet that does not contain the endmarker symbols,* $\vdash$ *and* $\dashv$*, such that* $\Sigma \cap \{\vdash, \dashv\} = \emptyset$ *and* $\Sigma' = \Sigma \cup \{\vdash, \dashv\}$*,*

(iii) Γ *is a finite set of symbols called the advice alphabet that does not contain the endmarker symbols,* $\vdash$ *and* $\dashv$*, such that* $\Gamma \cap \{\vdash, \dashv\} = \emptyset$ *and* $\Gamma' = \Gamma \cup \{\vdash, \dashv\}$*,*

(iv) $T_I \in \{\{\mathcal{L}, \mathcal{S}, \mathcal{R}\}, \{\mathcal{S}, \mathcal{R}\}, \{\mathcal{R}\}\}$ *represents the set of allowed head movements for the input tape,*

(v) $T_A \in \{\{\mathcal{S}, \mathcal{R}\}, \{\mathcal{R}\}\}$ *represents the set of allowed head movements for the advice tape,*

(vi) $\delta : Q \times \Sigma' \times \Gamma' \times Q \times T_I \times T_A \to [0, 1]_{\mathbb{R}}$ *is the transition function such that,* $\delta(q_1, \sigma, \gamma, q_2, t_I, t_A) = p$ *implies that when the automaton is in state* $q_1 \in Q$ *and*

it scans $\sigma \in \Sigma'$ on its input tape and $\gamma \in \Gamma'$ on its advice tape, a transition occurs with probability, $p \in [0,1]_{\mathbb{R}}$ which changes the state of the automaton to $q_2 \in Q$, meanwhile moving the input and advice tape heads in the directions specified respectively by $t_I \in T_I$ and $t_A \in T_A$,

(vii) *$q_0 \in Q$ is the initial state,*

(viii) *$q_{accept} \in Q$ is the accepting state upon entering which the automaton halts and announces that it accepts the input, and*

(ix) *$q_{reject} \in Q$ is the rejecting state upon entering which the automaton halts and announces that it rejects the input.*

The transition function δ must satisfy the following well-formedness condition.

$$\forall (q_1, \sigma, \gamma) \in Q \times \Sigma' \times \Gamma' \sum_{(q_2, t_I, t_A) \in Q \times T_I \times T_A} \delta(q_1, \sigma, \gamma, q_2, t_I, t_A) = 1.$$

A pfat $M = (Q, \Sigma, \Gamma, T_I, T_A, \delta, q_0, q_{accept}, q_{reject})$ is said to accept (reject) a string $x \in \Sigma^*$ with probability $p \in [0,1]_{\mathbb{R}}$ with the help of an advice string $a \in \Gamma^*$ if and only if M, when started at its initial state q_0 with $\vdash x \dashv$ on the input tape and $\vdash a \dashv$ on the advice tape and while the tape heads scan the first symbols to the right of the left endmarkers, reaches, q_{accept}, (q_{reject}) with probability p, by changing states and moving the input and advice tape heads according to the probabilities specified by its transition function, δ.

A language L defined on the alphabet Σ, is said to be recognized by such a pfat M with bounded error, with the help of an advice function $h : \mathbb{N} \to \Gamma^*$ if and only if there exists a constant $\rho \in (1/2, 1]_{\mathbb{R}}$ such that

- $L = \{x \mid M \text{ accepts } x \text{ with probability } p \geq \rho, \text{ with the help of } h(|x|)\}$, and
- $\overline{L} = \{x \mid M \text{ rejects } x \text{ with probability } p \geq \rho, \text{ with the help of } h(|x|)\}$.

A language L is said to be recognized by such a pfat, M with bounded error, using $O(f(n))$-length advice if there exists an advice function h with the following

properties:

- $|h(n)| \in O(f(n))$ for all $n \in \mathbb{N}$, and
- M recognizes L with bounded error, with the help of $h(n)$.

3.1.3. Quantum Finite Automata with Advice Tape

There exists a set of alternative definitions for a quantum finite automaton and they are not mutually equivalent in terms of computational power. Among these alternatives, the base model, on which we will build our definition of a qfat will be the finite automaton with quantum and classical states which was first introduced in [46] in two-way input access setting. The real-time variations of this model were also examined later in [47] and separately in [44], [48] etc. For a brief discussion on the alternative definitions of a qfa with their respective computational power and for further references in this subject, the reader may refer to *i.e.* [49] where a finite automaton with quantum and classical states is shown to be able to simulate not only other qfa models but also its deterministic and probabilistic counterparts which share similar settings such as input access type and the amount of allowed error.

Our definition of a qfat will make frequent use of terms from quantum computation domain such as "quantum states", "Hilbert spaces", "unitary operators" and "projective measurements" with which we assume the reader is familiar. (The reader may however refer to [50] or [51] for the definitions of these terms and for an effective introduction of basic quantum computation concepts.) Given a finite set of quantum states Q, the Hilbert space spanned by Q will be denoted by $\mathcal{H}(Q)$ in the remainder and the sets of unitary operators and projective measurements over $\mathcal{H}(Q)$ will be denoted by $\mathcal{U}(\mathcal{H}(Q))$ and $\mathcal{O}(\mathcal{H}(Q))$ respectively. A finite automaton with quantum and classical states augmented with an advice tape is then defined as follows.

Definition 3.3. *A quantum finite automaton with an advice tape is a 14-tuple* $(Q, S, \Sigma, \Gamma, T_I, T_A, |q_0\rangle, s_0, s_{accept}, s_{reject}, \Theta, \Delta, C, \delta)$*, where*

(i) Q *is a finite set of quantum states,*

(ii) S *is a finite set of classical states,*

(iii) Σ *is a finite set of symbols called the input alphabet that does not contain the endmarker symbols,* $\vdash$ *and* $\dashv$*, such that* $\Sigma \cap \{\vdash, \dashv\} = \emptyset$ *and* $\Sigma' = \Sigma \cup \{\vdash, \dashv\}$*,*

(iv) Γ *is a finite set of symbols called the advice alphabet that does not contain the endmarker symbols,* $\vdash$ *and* $\dashv$*, such that* $\Gamma \cap \{\vdash, \dashv\} = \emptyset$ *and* $\Gamma' = \Gamma \cup \{\vdash, \dashv\}$*,*

(v) $T_I \in \{\{\mathcal{L}, \mathcal{S}, \mathcal{R}\}, \{\mathcal{S}, \mathcal{R}\}, \{\mathcal{R}\}\}$ *represents the set of allowed head movements for the input tape,*

(vi) $T_A \in \{\{\mathcal{S}, \mathcal{R}\}, \{\mathcal{R}\}\}$ *represents the set of allowed head movements for the advice tape,*

(vii) $|q_0\rangle \in Q$ *is the initial quantum state,*

(viii) $s_0 \in S$ *is the initial classical state,*

(ix) $s_{accept} \in S$ *is the classical accepting state upon entering which the automaton halts and announces that it accepts the input,*

(x) $s_{reject} \in S$ *is the classical rejecting state upon entering which the automaton halts and announces that it rejects the input,*

(xi) $\Theta : S \times \Sigma' \times \Gamma' \to \mathcal{U}(\mathcal{H}(Q))$ *is a mapping, which rules the evolution of the quantum state of the automaton by assigning a unitary transformation to be applied on the quantum state, at each transition of the automaton, based on the current classical state of the automaton and the symbols its heads scan on the input and the advice tape,*

(xii) $\Delta : S \times \Sigma' \times \Gamma' \to \mathcal{O}(\mathcal{H}(Q))$ *is a mapping which, following the application of the unitary operation, selects the projective measurement (if any) to be applied on the quantum state,*

(xiii) $C = \{c_1, c_2, \cdots, c_s\}$ *is the set of potential outcomes of a projective measurement over* $\mathcal{H}(Q)$*,*

(xiv) $\delta : S \times \Sigma' \times \Gamma' \times C \to S \times T_I \times T_A$ *is a mapping which at each transition, rules the changes in the classical state and the tape head positions such that,* $\delta(s_1, \sigma, \gamma, c) = (s_2, t_I, t_A)$ *indicates that when the automaton is in state* $s_1 \in S$ *and it scans* $\sigma \in \Sigma'$ *on its input tape and* $\gamma \in \Gamma'$ *on its advice tape and when the outcome of the observation performed on the quantum state during that transition*

is $c \in C$, a transition occurs which changes the classical state of the automaton to $s_2 \in S$, meanwhile moving the input and advice tape heads in the directions specified respectively by $t_I \in T_I$ and $t_A \in T_A$,

The computation starts in the classical state s_0 and the quantum state $|q_0\rangle$. The transition that takes place when the automaton is in quantum state $|q\rangle \in Q$ and the classical state $s \in S$, and when the tape heads scan $\sigma \in \Sigma'$ on the input tape and $\gamma \in \Gamma'$ on the advice tape, consists of the following steps in the specified order:

- The unitary transformation specified by $\Theta(s, \sigma, \gamma)$ is applied on the quantum state $|q\rangle$ to produce the new quantum state $|q'\rangle$.
- If a projective measurement is specified by $\Delta(s, \sigma, \gamma)$, this measurement is applied on the new quantum state $|q'\rangle$ producing the outcome $c \in C$. If no measurement is specified by $\Delta(s, \sigma, \gamma)$, the outcome, c, of this step is assumed to be $c_\varepsilon \in C$.
- The classical state transition and the tape head movements specified by $\delta(s, \sigma, \gamma, c)$ are applied on the classical state and the tape heads producing a new classical state s' and new tape head positions.

The automaton halts when it enters either of the classical states s_{accept} and s_{reject}. We assume that δ is well defined so that every input gets accepted or rejected by the automaton in this way.

A qfat $(Q, S, \Sigma, \Gamma, T_I, T_A, |q_0\rangle, s_0, s_{accept}, s_{reject}, \Theta, \Delta, C, \delta)$ is said to accept (reject) a string $x \in \Sigma^*$ with probability $p \in [0, 1]_{\mathbb{R}}$ with the help of an advice string $a \in \Gamma^*$ if and only if M, when started at its initial quantum state, $|q_0\rangle$ and initial classical state s_0 with $\vdash x \dashv$ on the input tape and $\vdash a \dashv$ on the advice tape and while the tape heads scan the first symbols to the right of the left endmarkers, reaches, s_{accept}, (s_{reject}) with probability p, by changing states and moving input and advice tape heads as specified by the set, $\{\Theta, \Delta, \delta\}$ of transition functions.

A language L defined on the alphabet Σ, is said to be recognized by such a qfat

M with bounded error, with the help of an advice function $h : \mathbb{N} \to \Gamma^*$ if and only if there exists a constant $\rho \in (1/2, 1]_{\mathbb{R}}$ such that

- $L = \{x \mid M \text{ accepts } x \text{ with probability } p \geq \rho, \text{ with the help of } h(|x|)\}$, and
- $\overline{L} = \{x \mid M \text{ rejects } x \text{ with probability } p \geq \rho, \text{ with the help of } h(|x|)\}$.

A language L is said to be recognized by such a qfat, M with bounded error, using $O(f(n))$-length advice if there exists an advice function h with the following properties:

- $|h(n)| \in O(f(n))$ for all $n \in \mathbb{N}$, and
- M recognizes L with bounded error, with the help of $h(n)$.

3.1.4. The Classes of Languages Recognized with Advice Tape

For naming the language families corresponding to different settings of finite automata with advice tapes we will use an extension of the basic template, V/F, introduced by Karp and Lipton. The principal change will be the use of square brackets to cover the amount of advice as in $V/[F]$ which will indicate the fact that the advice in consideration is provided on a separate tape with a real-time tape head. Further specifications (if any) about the advice function will also take place within the square brackets and be separated by a comma. In order to indicate one-way access to the advice tape, the notation will be extended by use of the phrase 1- preceding the advice amount as in $V/1\text{-}[F]$. The function description $f(n)$ used in place of F will denote that the machine uses advice strings of length $O(f(n))$ for inputs of length n. (General descriptors like *poly* and *exp*, for polynomial and exponential bounds, respectively, can also be used in place of F.)

Following the notational conventions set by the past research, the name of the class of languages corresponding to the unadvised version of the automaton in question will be used in place of the V item in the template, thereby making the type of the input tape head explicit. Use of 1-DFA, for instance, will indicate that the underlying

automaton is a deterministic finite automaton with one-way access to its input tape while we prefer to keep the notation, REG, which was used in the naming of the older models, untouched for marking a deterministic finite automaton with real-time input tape access. The resulting notation for naming the classes of languages corresponding to the deterministic finite automata with advice tapes is summarized in Table 3.1 with sample class names.

Table 3.1. Naming of the classes of languages corresponding to the deterministic finite automata with advice tapes.

	Automaton		Tape Access		Advice	
Class Name	Model	Error	Input	Advice	Type	Amount
REG/$[n]$	dfa	none	real-time	real-time	deterministic	$O(n)$
REG/1-$[f(n)]$	dfa	none	real-time	one-way	deterministic	$O(f(n))$
1-DFA/$[n]$	dfa	none	one-way	real-time	deterministic	$O(n)$
1-DFA/1-$[poly]$	dfa	none	one-way	one-way	deterministic	*polynomial*
2-DFA/$[exp]$	dfa	none	two-way	real-time	deterministic	*exponential*
2-DFA/1-$[k]$	dfa	none	two-way	one-way	deterministic	k

The notational convention introduced above is flexible enough to represent the language classes corresponding to the probabilistic and quantum advised machines as well. Note that computation in probabilistic and quantum settings comes with a probability of error in language recognition and the classes of languages recognized in each setting are categorized also with respect to the type of the error -bounded or unbounded- allowed in each setting. PFA and QFA conventionally denote the class of languages that can be recognized with unbounded error(ue) by real-time probabilistic and quantum finite automata. The corresponding classes that can be recognized with bounded error(be) in these settings are named respectively as BPFA and BQFA. The use of 1- and 2- which appear as a prefix of the class names denote the classes of languages corresponding to the one-way and two-way versions of probabilistic and

quantum automata as well. The samples from the class names that can be generated with this rule set are listed in Table 3.2 and Table 3.3.

Table 3.2. Naming of the classes of languages corresponding to the probabilistic finite automata with advice tapes.

	Automaton		Tape Access		Advice	
Class Name	Model	Error	Input	Advice	Type	Amount
PFA/1-[n]	pfa	ue	real-time	one-way	deterministic	$O(n)$
BPFA/1-[n]	pfa	be	real-time	one-way	deterministic	$O(n)$
1-BPFA/1-[n]	pfa	be	one-way	one-way	deterministic	$O(n)$

Table 3.3. Naming of the classes of languages corresponding to the quantum finite automata with advice tapes.

	Automaton		Tape Access		Advice	
Class Name	Model	Error	Input	Advice	Type	Amount
QFA/1-[n]	qfa	ue	real-time	one-way	deterministic	$O(n)$
BQFA/[n]	qfa	be	real-time	real-time	deterministic	$O(n)$
BQFA/1-[n]	qfa	be	real-time	one-way	deterministic	$O(n)$

We will also be examining randomly picked advice, as defined by Yamakami [9]. In this scenario, the advice string is randomly selected from a set of alternatives according to a prespecified probability distribution. (Deterministic finite automata which use randomized advice was shown to be able to perform tasks which are impossible with deterministic advice in [9].) The use of randomized advice will be indicated by the letter R appearing before the advice length in our class names as in $V/[R\text{-}F]$. In our class names corresponding to this model of computation, the items, *be* and *ue* will be used in superscript form next to the original class name in order to indicate whether

bounded or unbounded-error language recognition is intended when this is not clear from the core class name. The samples from the resulting class names are listed in Table 3.4.

Table 3.4. Naming of the classes of languages corresponding to the finite automata with random advice placed on advice tape.

	Automaton		Tape Access		Advice	
Class Name	Model	Error	Input	Advice	Type	Amount
$\mathsf{REG}^{be}/[R\text{-}f(n)]$	dfa	be	real-time	real-time	random	$O(f(n))$
$1\text{-}\mathsf{DFA}^{be}/1\text{-}[R\text{-}n]$	dfa	be	one-way	one-way	random	$O(n)$

3.2. Preliminary Notes

As we set the notation for talking about the corresponding language classes to our models of advised finite automata we can list some basic observations on them, before proceeding to our main results.

The model of real-time finite automata with advice tracks [7] is equivalent to our model with a separate advice tape when we set both the input and advice tape heads to be real-time. Therefore, all the results shown for the advice track model are inherited for this setting of our machines. We have, for instance,

$$\mathsf{REG}/[n] = \mathsf{REG}/n \tag{3.1}$$

where REG/n is defined in [7]. On the other hand, the quantum class $1\text{-}\mathsf{QFA}/n$ of [11] does *not* equal $\mathsf{BQFA}/[n]$, as we will show later in Section 3.5.

Note that we allow only one advice tape in our model. This is justified by the following observation about the great power of one-way finite automata with multiple advice tapes.

Theorem 3.4. *Every language can be recognized by a finite automaton with a one-way input tape and two one-way advice tapes.*

Proof. Let L be any language on the alphabet Σ. We construct a finite automaton M that recognizes L using a one-way input tape and two one-way advice tapes as follows.

Let $\Gamma = \Sigma \cup \{c_a, c_r\}$ be the advice alphabet, where $\Sigma \cap \{c_a, c_r\} = \emptyset$. For an input of length n, the advice on the first advice tape lists every string in Σ^n in alphabetical order, where every member of L is followed by a c_a, and every nonmember is followed by a c_r. So the content of the first advice tape looks like

$$w_1 c_1 w_2 c_2 \cdots w_{|\Sigma|^n} c_{|\Sigma|^n},$$

where $w_i \in \Sigma^n$, and $c_i \in \{c_a, c_r\}$ for $i \in \{1, \ldots, |\Sigma|^n\}$.

The second advice tape content looks like

$$c_a c_r^n c_a c_r^n \cdots c_a c_r^n c_a,$$

with $|\Sigma|^n$ repetitions, and will be used by the machine for counting up to $n+1$ by moving between two consecutive c_a symbols on this tape.

M starts its computation while scanning the first symbols of the input string and w_1 on the first advice tape. It attempts to match the symbols it reads from the input tape and the first advice tape, moving synchronously on both tapes. If the ith input symbol does not match the ith symbol of w_j, M pauses on the input tape, while moving the two advice heads simultaneously until the second advice head reaches the next c_a, thereby placing the first advice tape head on the ith position of w_{j+1}, where $1 \leq i \leq n$, and $1 \leq j < |\Sigma|^n$. As the words on the first advice tape are ordered lexicographically, it is guaranteed that M will eventually locate the word on the first advice tape that

matches the input in this manner. M halts when it sees the endmarker on the input tape, accepting if the symbol read at that point from the first advice tape is c_a, and rejecting otherwise. □

3.3. Deterministic Finite Automata with Advice Tapes

It is clear that a machine with advice tape is at least as powerful as a machine of the same type with advice track, which in turn is superior to a corresponding machine with advice prefix, as mentioned in Section 2.2. We will now show that allowing either one of the input and advice head to pause on their tapes does enlarge the class of recognized languages.

Theorem 3.5. $\mathsf{REG}/n \subsetneq \mathsf{REG}/1\text{-}[n]$.

Proof. It follows trivially from the definitions of the classes that

$$\mathsf{REG}/n = \mathsf{REG}/[n] \subseteq \mathsf{REG}/1\text{-}[n]. \tag{3.2}$$

Let $|w|_\sigma$ denote the number of occurrences of symbol σ in string w. To show that the above subset relation is proper, we will consider the language

$$\texttt{EQUAL}_2 = \{w|\ w \in \{a,b\}^* \text{ and } |w|_a = |w|_b\},$$

which is known [7] to lie outside REG/n.

One can construct a finite automaton that recognizes $\texttt{EQUAL}_2$ with real-time input and one-way access to linear-length advice as follows. For inputs of odd length, the advice to the automaton is 0. Upon scanning 0 on the advice tape, the automaton moves to a rejecting state and never leaves it, hence rejecting the input. For inputs of even length, n, the advice function is $1^{n/2}$. Hence, the advice function h is given by

$$h(n) = \begin{cases} 0, & \text{if } n \text{ is odd} \\ 1^{n/2}, & \text{if } n \text{ is even} \end{cases}$$

The automaton moves its advice tape head one position to the right for each a that it reads on the input tape. The input word is accepted if the number of a's on the input tape and the number of 1's on the advice tape do match and it is rejected otherwise.

State diagram of the dfat described above is shown in Figure 3.2 where an arrow from node q_i to node q_j labeled as "$(S_I, S_A)/(D_I, D_A)$ " marks a transition that occurs when the automaton is in state q_i and scans the symbol S_I on the input tape and S_A on the advice tape. This transition brings the automaton to state q_j while moving the input and advice tape heads in the directions indicated by $D_I \in T_I$ and $D_A \in T_A$ respectively. The symbol $*$ is used as a shortcut for "any symbol" from the appropriate set of symbols.

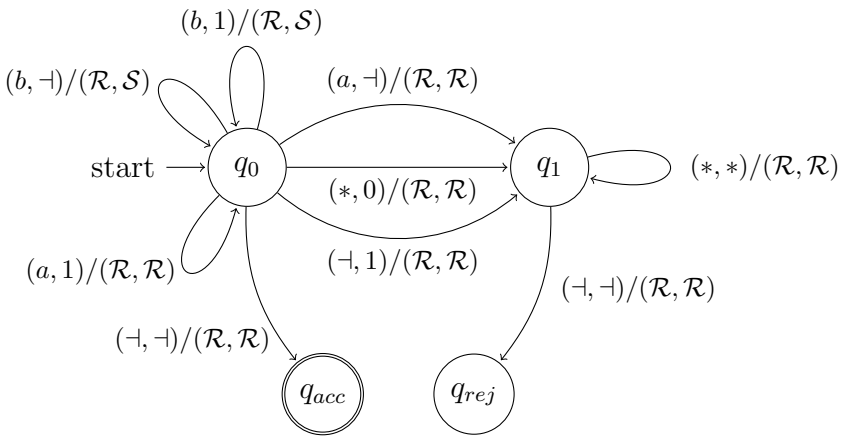

Figure 3.2. State diagram for a finite automaton with advice tape that recognizes the language $\texttt{EQUAL}_2 = \{w|\ w \in \{a,b\}^* \text{ and } |w|_a = |w|_b\}$.

□

Theorem 3.6. $\mathsf{REG}/n \subsetneq \text{1-}\mathsf{DFA}/[n]$.

Proof. Consider the language

$$\texttt{EQUAL} = \{w | w \in \{a, b, c\}^* \text{ where } |w|_a = |w|_b\},$$

which is similar to $\texttt{EQUAL}_2$, but with a bigger alphabet. $\texttt{EQUAL} \notin \mathsf{REG}/n$, as can be shown easily by Yamakami's characterization theorem for this class. We will describe a dfat M with one-way input, and real-time access to an advice string that is just 1^{2n}, where n is the input length. (State diagram for M is given in Figure 3.3.)

M moves the advice head one step to the right for each a that it scans in the input. When it scans a b, it advances the advice head by three steps and for each c, scanned on the input tape, the advice head is moved two steps. If the advice head attempts to move beyond the advice string, M rejects. When the input tape head reaches the end of the tape, M waits to see if the advice tape head will also have arrived at the end of the advice string after completing the moves indicated by the last input symbol. If this occurs, M accepts, otherwise, it rejects.

Note that the advice head is required to move exactly $|w|_a + 3|w|_b + 2(n - |w|_a - |w|_b)$ steps, which equals $2n$ if and only if the input is a member of $\texttt{EQUAL}$. Therefore, we have

$$\texttt{EQUAL} \in \text{1-}\mathsf{DFA}/[n]. \tag{3.3}$$

Then we can safely state that

$$\mathsf{REG}/n \subsetneq \text{1-}\mathsf{DFA}/[n]. \tag{3.4}$$

□

Tadaki et al. [7] studied one-tape linear-time Turing machines with an advice track, and showed that the class of languages that they can recognize coincides with

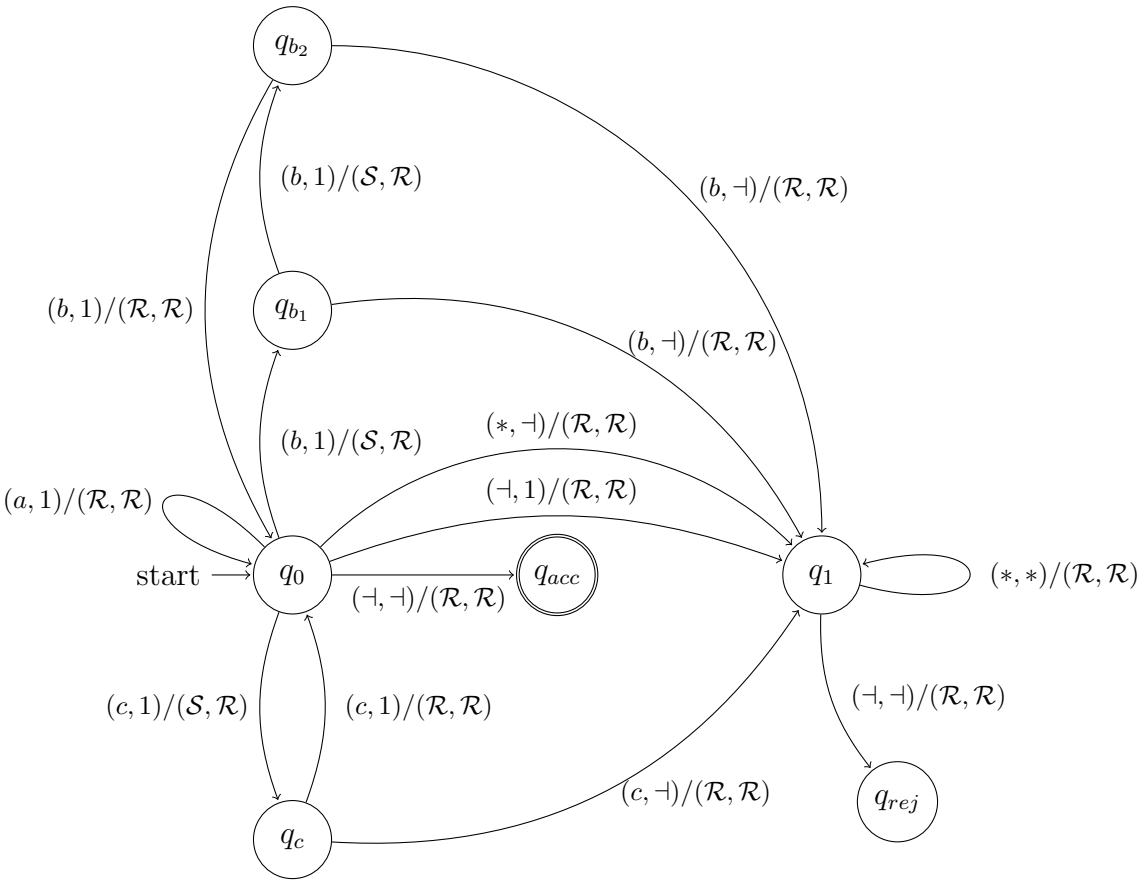

Figure 3.3. State diagram for a finite automaton with advice tape that recognizes the language $\texttt{EQUAL} = \{w | w \in \{a,b,c\}^* \text{ where } |w|_a = |w|_b\}$.

REG/n. Theorem 3.5 above lets us conclude that simply having a separate head for advice increases the computational power of a real-time dfa, whereas the incorporation of a single two-way head for accessing both advice and a linear amount of read/write memory simultaneously does not.

As noted earlier, advice lengths which are increasing functions of the input length are not useful in the advice prefix model. Only linear-sized advice has been studied in the context of the advice track model [7, 9]. Theorem 3.7 demonstrates a family of languages for which very small increasing advice length functions are useful in the advice tape model, but not in the advice track model.

Theorem 3.7. *For every function* $f : \mathbb{N} \to \mathbb{N}$ *such that* $f(n) \in \omega(1) \cap O(\sqrt{n})$, 1-$\mathsf{DFA}$/1-$[f^2(n)] \not\subseteq \mathsf{REG}/n$.

Proof. Consider the language

$$\mathtt{LABC}_f = \{a^k b^m c^k | k \leq f(n), n = k + m + k\},$$

of words made up of ordered sequences of a's, b's and c's the lengths of which is determined by any function f satisfying the properties in the theorem statement. Noting $f(n) \notin O(1)$, one may use Theorem 2 of [9] (which was cited as Fact 2.10 in Section 2.2) to show that

$$\mathtt{LABC}_f \notin \mathsf{REG}/n.$$

One can construct a dfat with one-way access to input and advice that recognizes $\mathtt{LABC}_f$ as follows. For inputs of length n, the advice string is of the form

$$\#\#a\#aa\#aaa\#\cdots\#a^{f(n)}\#,$$

with length $O(f^2(n))$. During any step, if the automaton detects that the input is not of the form $a^*b^*c^*$, it rejects the input. For each a that it reads from the input tape, the automaton moves the advice tape head to the next # on the advice tape. (If the advice ends when looking for a #, the input is rejected.) When the input tape head scans the b's, the advice tape head remains idle. Finally, when the input head starts to scan the c's, the automaton compares the number of c's on the input tape with the number of a's that it can scan until the next # on the advice tape. If these match, the input is accepted; otherwise it is rejected. □

When restricted to constant size advice, the parallelism and the two-way input access inherent in our model does not make it more powerful than the advice prefix model. As we show now, one can always read the entire advice before starting to read the input tape without loss of computational power in the constant-length advice case:

Theorem 3.8. *For every* $k \in \mathbb{N}$, $2\text{-}\mathsf{DFA}/1\text{-}[k] = \mathsf{REG}/k$.

Proof. The relation $\mathsf{REG}/k \subseteq 2\text{-}\mathsf{DFA}/1\text{-}[k]$ is trivial, since an automaton taking constant-length advice in the prefix or track formats can be converted easily to one that reads it from a separate tape. For the other direction, note that a dfat M with two-way input that uses k bits of advice corresponds to a set S of 2^k real-time dfa's *without* advice, each of which can be obtained by hard-wiring a different advice string to the program of M, and converting the resulting two-way dfa to the equivalent real-time machine, which exists by [52]. The advice string's job is just to specify which of these machines will run on the input string. It is then easy to build a dfa with advice prefix which uses the advice to select the appropriate program to run on the input. □

Since our model is equivalent to the advice prefix model for constant-length advice, we inherit the results like Theorem 5 of [6], which states that the longer advice strings one allows, the larger the class of languages we can recognize will be, as long as one makes sure that the advice and input alphabets are identical.

For any language L on an alphabet Σ, and for natural numbers n and k such that $k \leq n$, we define the relation $\equiv_{L,n,k}$ on the set Σ^k as follows:

$$x \equiv_{L,n,k} y \iff \text{for all strings } z \text{ of length } n-k, xz \in L \text{ if and only if } yz \in L.$$

In the remainder of this chapter, we will make frequent use the following lemma, which is reminiscent of Yamakami's characterization theorem for REG/n [9], to demonstrate languages which are unrecognizable with certain amounts of advice by automata with one-way input.

Lemma 3.9. *For any function f, if $L \in$ 1-DFA/1-$[f(n)]$, then for all n and all $k \leq n$, $\equiv_{L,n,k}$ has $O(f(n))$ equivalence classes.*

Proof. Let M be the dfat which is supposed to recognize L with an advice string of length $O(f(n))$. If we fix the position of the input head, there are just $O(f(n))$ combinations of internal state and advice head position pairs that are potentially reachable for M. Assume that the number of equivalence classes of $\equiv_{L,n,k}$ is not $O(f(n))$. Then for some sufficiently large n, there exists two strings x and y of length k in two different equivalence classes of $\equiv_{L,n,k}$ which cause M to reach precisely the same head positions and internal state after being processed if they are presented as the prefixes of two n-symbol input strings in two separate executions of M. But M will then have to give the same response to the two input strings xz and yz for any $z \in \Sigma^{n-k}$, meaning that $x \equiv_{L,n,k} y$. □

We can now establish the existence of an infinite hierarchy of language classes that can be recognized by dfat's with increasing amounts of advice.

Theorem 3.10. *For $k \in \mathbb{Z}^+$,* 1-DFA/1-$[n^k] \subsetneq$ 1-DFA/1-$[n^{k+1}]$.

Proof. In order to prove the theorem statement, we will first define a family $\mathtt{LSPOS}_k$ of languages for $k \in \mathbb{Z}^+$. Each language in this family will consist of words made up

of symmetrically placed ordered sequences of symbols in the corresponding alphabet. Then, we will show that advice strings of length $\Theta(n^i)$ are necessary (Lemma 3.12) and sufficient (Lemma 3.13) to recognize any particular member $\texttt{LSPOS}_i$ of this family. □

Definition 3.11. *For $k \in \mathbb{Z}^+$, we define the language $\texttt{LSPOS}_k$ on the $k+1$-symbol alphabet $\{c_0, c_1, \ldots, c_k\}$ as*

$$\texttt{LSPOS}_k = \{c_k^{n_k} c_{k-1}^{n_{k-1}} \cdots c_1^{n_1} c_0^{n_0} c_1^{n_1} \cdots c_{k-1}^{n_{k-1}} c_k^{n_k} | n_0 > 0 \text{ and } n_j \geq 0 \text{ for } j \in \{1, \ldots, k\}\}.$$

Lemma 3.12. *For $i \in \mathbb{Z}^+$, $\texttt{LSPOS}_i \notin$ 1-DFA/1-$[n^{i-1}]$*

Proof. For a positive integer n, consider the set S of strings of length $k = \lfloor n/2 \rfloor + 1$ and of the form $c_i^* c_{i-1}^* \cdots c_1^* c_0^+$. Note that each member of S is the first half of a different member of $\texttt{LSPOS}_i$, no two distinct members x and y of S satisfy $x \equiv_{\texttt{LSPOS}_i, n, k} y$, and that there are $\Theta(n^i)$ members of S. We conclude using Lemma 3.9 that

$$\texttt{LSPOS}_i \notin \text{1-DFA/1-}[n^{i-1}]. \tag{3.5}$$

□

Lemma 3.13. *For $i \in \mathbb{Z}^+$, $\texttt{LSPOS}_i \in$ 1-DFA/1-$[n^i]$.*

Proof. An inductive argument will be employed to show the truth of the statement, so let us first consider the language $\texttt{LSPOS}_1$. To see that

$$\texttt{LSPOS}_1 \in \text{1-DFA/1-}[n^1], \tag{3.6}$$

we construct an advice function $h_1(n)$ and an automaton M_1 as follows. For inputs of length n, let $h_1(n) = 1^n$ be given as advice. The automaton M_1 checks if the input is of the form $c_1^i c_0^j c_1^k$ for $i, k \geq 0$ and $j > 0$. If not, it rejects. In parallel, M_1 moves the advice tape head while scanning the input as follows:

(i) For each c_1 that comes before the first c_0 in the input, the advice tape head stays put.

(ii) For each c_0 in the input, the advice tape head moves one step to the right.

(iii) Finally, for each c_1 that comes after the last c_0 in the input, the advice tape head moves two steps to the right.

(iv) The input is accepted if the endmarkers are scanned simultaneously on both tapes.

Since the advice head moves exactly $j + 2k$ steps, which equals $n = i + j + k$ if and only if $i = k$, we conclude that M_1 recognizes $\texttt{LSPOS}_1$ when provided with $h_1(n)$, a linear-length advice function.

Now let us prove that

$$\texttt{LSPOS}_i \in \text{1-DFA/1-}[n^i] \implies \texttt{LSPOS}_{i+1} \in \text{1-DFA/1-}[n^{i+1}]. \tag{3.7}$$

Assume $\texttt{LSPOS}_i \in \text{1-DFA/1-}[n^i]$. Then there should be a dfat M_i which recognizes $\texttt{LSPOS}_i$ when it has access to the advice function $h_i(n)$ of length $O(n^i)$. Below, we construct a dfat M_{i+1} and an advice function $h_{i+1}(n)$ of length $O(n^{i+1})$ such that M_{i+1} recognizes $\texttt{LSPOS}_{i+1}$ when it has access to advice given by function $h_{i+1}(n)$.

Note that the members of $\texttt{LSPOS}_{i+1}$ are members of $\texttt{LSPOS}_i$ sandwiched between equal numbers of c_{i+1}'s on each end. Therefore, the method for checking membership in $\texttt{LSPOS}_i$ can be used in the test for membership in $\texttt{LSPOS}_{i+1}$ if one can check whether the c_{i+1} sequences at each end are of the same length separately. Hence, we define the advice function $h_{i+1}(n)$ for $\texttt{LSPOS}_{i+1}$ in terms of the advice function $h_i(n)$ for $\texttt{LSPOS}_i$ as

$$h_{i+1}(n) = h_i(n)\#_{i+1}h_i(n-2)c_{i+1}\#_{i+1}\cdots\#_{i+1}h_i(n-2\lfloor\tfrac{n}{2}\rfloor)c_{i+1}^{\lfloor\frac{n}{2}\rfloor}\#_{i+1},$$

that is, one concatenates all the strings $h_i(n-2j)c_{i+1}^j\#_{i+1}$ for $j \in \{0,\ldots,\lfloor\frac{n}{2}\rfloor\}$ in increasing order, where $\#_{i+1}$ is a new symbol in M_{i+1}'s advice alphabet. As $h_i(n)$ is of length $O(n^i)$, the length of $h_{i+1}(n)$ can be verified to be $O(n^{i+1})$.

When provided access to the advice function $h_{i+1}(n)$, the automaton M_{i+1} performs the tasks below in parallel in order to recognize the language $\texttt{LSPOS}_{i+1}$.

(i) The input is checked to be of the form $c_{i+1}^*c_i^*\cdots c_1^*\ c_0^+\ c_1^*\cdots c_i^*c_{i+1}^*$. If not, it is rejected.

(ii) For each c_{i+1} on the input tape, that comes before any other symbol, the advice head is moved to the next $\#_{i+1}$ on the advice tape. If the endmarker is scanned on the advice tape at this step, the input is rejected. When the first non-c_{i+1} symbol is scanned on the input, the control passes to the automaton M_i for language $\texttt{LSPOS}_i$, which runs on the input tape content until the first c_{i+1} or the endmarker, and uses as advice the content until the first c_{i+1} or $\#_{i+1}$ on the advice tape. If M_i rejects its input, so does M_{i+1}. If M_i accepts its input, M_{i+1} accepts its input only if the number of c_{i+1}'s on the remainder of the input tape matches the number of c_{i+1}'s on the advice tape until the first $\#_{i+1}$.

□

We now show that $\texttt{PAL} = \{ww^R |\ w \in \{a,b\}^* \textit{and } w^R \textit{ is the reverse of } w\}$, the language of even-length palindromes on the alphabet $\{a,b\}$, is unrecognizable by dfat's with one-way input and polynomial-length advice:

Theorem 3.14. $\texttt{PAL} \notin 1\text{-}\mathsf{DFA}/1\text{-}[poly]$.

Proof. Similarly to the proof of Lemma 3.12, we consider the set S of all strings on $\{a,b\}$ of length $k = n/2$ for an even positive number n. No two distinct members x and y of S satisfy $x \equiv_{\texttt{PAL},n,k} y$, and there are $2^{\Theta(n)}$ members of S. We conclude using

Lemma 3.9 that

$$\texttt{PAL} \notin \text{1-DFA}/\text{1-}[poly]. \tag{3.8}$$

□

Note that, this result can be extended to include any sub-exponential advice length function instead of polynomial, as the same line of reasoning can be applied in these cases too. Moreover, since a machine with real-time input does not have time to consume more than a linear amount of advice, we easily have

Corollary 3.15. *For every function* $f : \mathbb{N} \to \mathbb{N}$, $\texttt{PAL} \notin \text{REG}/\text{1-}[f(n)]$.

A natural question that arises during the study of advised computation is whether the model under consideration is strong enough to recognize *every* desired language. The combination of two-way input tape head and exponentially long advice can be shown to give this power to finite automata. Let ALL denote the class of all languages defined over the input alphabet Σ.

Theorem 3.16. $\text{2-DFA}/[exp] = \text{ALL}$.

Proof. The advice string for input length n contains all members of the considered language of length n, separated by substrings consisting of $n + 2$ blank symbols. The automaton compares the input with each of the strings listed on the advice tape in the order of appearance. If it is able to match the input to a word on the advice tape, it accepts the input. If a mismatch occurs, the machine rewinds to the start of the input while consuming blank symbols until the next member string on the advice tape. If the advice ends without a match, the input is rejected. The advice length is $2^{O(n)}$. □

Whether $\text{1-DFA}/\text{1-}[exp] = \text{ALL}$ is an open question. We do not even know if $\texttt{PAL} \in \text{1-DFA}/\text{1-}[exp]$. But we are able to prove a separation between classes corresponding to

machines with one-way versus two-way input that are confined to polynomial-length advice, as the following theorem shows.

Theorem 3.17. 1-DFA/1-[$poly$] $\subsetneq$ 2-DFA/1-[$poly$].

Proof. We already showed in Theorem 3.14 that polynomial-length advice is no help for dfat's with one-way input for recognizing PAL. To prove the present theorem, we shall describe how a two-way dfa with real-time access to a quadratic-length advice string can recognize PAL. On an input of length n, the advice tells the automaton to reject if n is odd. For even n, the advice assists the automaton by simply marking the $n/2$ pairs $(i, n-i+1)$ of positions that should be holding matching symbols on the input string. Consider, for example

$$h(8) = \#10000001\#01000010\#00100100\#00011000\#.$$

The automaton should just traverse the input from the first symbol to the last while also traversing the part of the advice that lies between two separator symbols (#), and then do the same while going from the last symbol to the first, and so on. At each pass, the automaton should check whether the input symbols whose positions match those of the two 1's on the advice are identical. If this check fails at any pass, the automaton rejects the input, otherwise, it accepts.

The method described above requires a two-way automaton with real-time access to an advice of length $n^2/2$. (The separator symbols are for ease of presentation, and are not actually needed for the construction.) □

3.4. Randomized Advice for Deterministic Machines and vice versa

We now turn to randomly selected probabilistic advice given to deterministic machines. Yamakami [9] proved that this setup yields an improvement in language

recognition power over REG/n, by demonstrating a deterministic automaton with advice track recognizing the center-marked palindrome language with randomized advice. Considering the amount of randomness involved in the selection of the advice string as a resource, Yamakami's example requires $O(n)$ random bits, since it requires picking a string from a set with $2^{O(n)}$ elements with uniform probability.

Furthermore, reducing the error bound of Yamakami's automaton to smaller and smaller values requires extending the advice alphabet to bigger and bigger sizes. In the construction we will present in Theorem 3.18, the number of random bits does not depend on the input length, and any desired error bound can be achieved without modifying the advice alphabet.

Theorem 3.18. $1\text{-}\mathsf{DFA}/1\text{-}[n] \subsetneq 1\text{-}\mathsf{DFA}^{be}/1\text{-}[R\text{-}n]$.

Proof. We will use the language

$$\mathtt{EQUAL}_3 = \{w|\ w \in \{a,b,c\}^*, |w|_a = |w|_b = |w|_c\}$$

to separate the language classes in the theorem statement.

Let k be any positive integer, $n = 3k$, and consider the set S of all strings of length k and of the form $a^*b^*c^*$. Note that S has $\binom{k+2}{2} = \omega(n)$ members, and that no two distinct members x and y of S satisfy $x \equiv_{\mathtt{EQUAL}_3,n,k} y$. We conclude using Lemma 3.9 that

$$\mathtt{EQUAL}_3 \notin 1\text{-}\mathsf{DFA}/1\text{-}[n]. \tag{3.9}$$

To show that

$$\mathtt{EQUAL}_3 \in 1\text{-}\mathsf{DFA}^{be}/1\text{-}[R\text{-}n], \tag{3.10}$$

we will describe a set of advice strings, and show how a randomly selected member of this set can assist a one-way dfat N to recognize $\texttt{EQUAL}_3$ with overall bounded error. We shall be adapting a technique used by Freivalds in [53].

If the input length n is not divisible by 3, N rejects. If $n = 3k$ for some integer k, the advice is selected with equal probability from a collection of linear-size advice strings

$$A_i = 1^i \# 1^{ki^2+ki+k} \text{ for } i \in \{1, \ldots, s\},$$

where s is a constant.

N starts by reading the 1's in the advice string that precede the separator character #, thereby learning the number i. N then starts to scan the input symbols, and moves the advice head 1, i, or i^2 steps to the right for each a, b or c that it reads on the input tape, respectively. The input is accepted if the automaton reaches the ends of the input and advice strings simultaneously, as in the proof of Theorem 3.4. Otherwise, the input is rejected. Note that the automaton accepts the input string w if and only if the number of symbols in the advice string that comes after the separator symbol is equal to the total number of moves made by the advice tape head while the input head scans w. N accepts w if and only if

$$|w|_a + |w|_b i + |w|_c i^2 = k + ki + ki^2, \tag{3.11}$$

which trivially holds for $w \in \texttt{EQUAL}_3$ no matter which advice string is selected, since $|w|_a = |w|_b = |w|_c = k$ in that case.

If $w \notin \texttt{EQUAL}_3$, the probability of acceptance is equal to the probability of selecting one of the roots of the quadratic equation

$$(|w|_c - k)i^2 + (|w|_b - k)i + (|w|_a - k) = 0, \tag{3.12}$$

as the value of i. This probability is bounded by $\frac{2}{s}$, and can be pulled down to any desired level by picking a bigger value for s, and reorganizing the automaton accordingly. □

Another way of integrating randomness to the original model is to employ probabilistic computation with access to deterministic advice. We show below that probabilistic automata with advice can recognize more languages with bounded error than their deterministic counterparts.

Theorem 3.19. 1-DFA/1-$[n] \subsetneq$ 1-BPFA/1-$[n]$.

Proof. The following inclusion is by definition:

$$\text{1-DFA/1-}[n] \subseteq \text{1-BPFA/1-}[n]. \tag{3.13}$$

So it remains to show that there is a language which can not be recognized by a one-way dfat with one-way access to linear-size advice but can be recognized with bounded error by a pfat with one-way input with the help of same amount of advice. We claim that $\texttt{EQUAL}_3$, which was introduced and was shown to lie outside 1-DFA/1-$[n]$ in the proof of Theorem 3.18, is one such language.

We now describe how to construct a one-way pfat P and an associated linear-length advice function to recognize $\texttt{EQUAL}_3$ for any specified nonzero error bound $\varepsilon < \frac{1}{2}$. The idea is reminiscent of that used for the proof of Theorem 3.18. However we now specify a deterministic advice function which contains all the alternatives and let the probabilistic automaton randomly pick and use one.

Let n denote the length of the input, and let $s = \lceil \frac{2}{\varepsilon} \rceil$. If n is not divisible by 3, the automaton rejects with probability 1. If n is divisible by 3, the advice is the string which is obtained by concatenating all the strings $\#1^{\frac{n}{3}i^2+\frac{n}{3}i+\frac{n}{3}}$ for $i \in \{1, \ldots, s\}$ in increasing order as seen below:

$$\#1^n\#1^{\frac{7n}{3}}\#\ldots\#1^{\frac{n}{3}s^2+\frac{n}{3}s+\frac{n}{3}},$$

P starts by randomly picking an integer i between 1 and s, and moving its advice head to the i'th #. It then starts scanning the input, moving the advice head by 1, i, or i^2 steps for each a, b or c, just as we had in the proof of Theorem 3.18. It accepts if and only if the advice head reaches the next # (or the end of the advice string) simultaneously with the arrival at the end of the input. The correctness of the algorithm follows from the argument in the proof of Theorem 3.18. Hence we have shown that

$$\texttt{EQUAL}_3 \in \text{1-BPFA}/\text{1-}[n] \tag{3.14}$$

and therefore we can conclude

$$\text{1-DFA}/\text{1-}[n] \subsetneq \text{1-BPFA}/\text{1-}[n]. \tag{3.15}$$

□

3.5. Quantum Finite Automata with Advice Tapes

Yamakami [11, 12] defined the class 1-QFA/n as the collection of languages which can be recognized with bounded error by real-time Kondacs-Watrous quantum finite automata (KWqfa's) with advice tracks. The KWqfa is one of many inequivalent models of quantum finite-state computation that were proposed in the 1990's, and is known to be strictly weaker than classical finite automata in the context of bounded-error language recognition [30]. This weakness carries over to the advised model of [11, 12], with the result that there exist some regular languages that are not members of 1-QFA/n. We use a state-of-the-art model of quantum automaton that can simulate its classical counterparts trivially, [44, 45] so we have:

Theorem 3.20. 1-QFA/$n \subsetneq$ BQFA/$[n]$.

Whether this strong version of qfa can outperform its classical counterparts with advice tapes is an open question. We are able to show a case in which a quantum automaton has an advantage over a classical one in the following restricted setup, which may seem trivial at first sight: Call an advice tape *empty* if it contains the standard blank tape symbol in all its squares. We say that a machine M receives *empty advice* of length $f(n)$, if it is just allowed to move its advice head on the first $f(n)$ squares of an empty advice tape, where n is the input length. This restriction will be represented by the presence of the designation `empty` in the specification lists of the relevant complexity classes.

Theorem 3.21. $\mathsf{BPFA}/1\text{-}[n, \texttt{empty}] \subsetneq \mathsf{BQFA}/1\text{-}[n, \texttt{empty}]$.

Proof. An empty advice tape can be seen as an increment-only counter, where each move of the advice tape head corresponds to an incrementation on the counter, with no mechanism for decrementation or zero-testing provided in the programming language. In [48], Yakaryılmaz et al. studied precisely this model. It is obvious that classical automata augmented with such a counter do not gain any additional computational power, so $\mathsf{BPFA}/1\text{-}[n, \texttt{empty}]$ equals the class of regular languages, just like the corresponding class without advice. On the other hand, real-time qfa's augmented with such an increment-only counter were shown to recognize some nonregular languages like $\texttt{EQUAL}_2$ with bounded error in [48]. □

Since increment-only counters are known to increase the computational power of real-time qfa's in the unbounded-error setting as well, [48], we can also state that

Theorem 3.22. $\mathsf{PFA}/1\text{-}[n, \texttt{empty}] \subsetneq \mathsf{QFA}/1\text{-}[n, \texttt{empty}]$.

4. INKDOTS AS ADVICE TO FINITE AUTOMATA

In this chapter, we introduce a novel way of providing advice to finite automata, namely, by marking some positions on the input with inkdots. Deterministic and probabilistic versions of the model are taken into account and the strengths and the limits of these settings are analyzed in comparison with the previously studied models of advised finite automata. The results of this analysis are presented in the form of separations, collapses and hierarchies among the classes of languages that can be recognized in different settings with varying amounts of advice. The discussion is also extended to issues such as succinctness in terms of the number of states of a minimal finite automaton and to coexistence of advice and access to very small secondary memory.

The remainder of this chapter is structured as follows. The advice inkdot model and the corresponding language classes will be presented in detail in Section 4.1. This new model will be compared with the previously studied models of advised finite automata in Sections 4.2 and 4.3, where it will be shown to be intermediate in power between the prefix and the track models of advice. Section 4.4 will demonstrate the existence of an infinite hierarchy among the classes of languages that can be recognized with the help of different numbers of inkdots as advice; both when those numbers are restricted to be constants, and when bounded by increasing functions of the input length. In Section 4.5, we will show that inkdot advice can cause significant savings in the number of states of the advised automaton when compared with the prefix advice model, which in turn is superior in this sense to pure unadvised automata. In Section 4.6, we will demonstrate that the strength of the model increases if one employs a probabilistic automaton instead of a deterministic one, and assists it with inkdots placed randomly according to an advice distribution. Section 4.7 will extend the advised machine model by allowing it access to a work tape. It is interesting to note that arbitrarily small space turns out to be a fruitful computational resource along with advice, while it is well known that one-way sublogarithmic space Turing machines (TM's) cannot recognize more languages than their constant-space counterparts.

4.1. Basic Notions, Formal Definitions and Notation

In this section, we introduce a new model of advised finite automata in which the advice will be provided as inkdots on the input string. An inkdot is a supplementary marker that can be placed on any symbol of the input of a computing machine. The presence of that mark can be sensed by the automaton only when the input tape head visits that position. This mark can not be erased or moved, and no more than one inkdot is allowed on one cell. (Inkdots are different from pebbles, which are more widely used in the literature (see *e.g.* [54]), only in their inability to be moved around on the input tape.)

It is known (see [55]) that a deterministic Turing machine would not gain any additional computational power if it is also provided with the ability of marking one input tape cell with an inkdot.

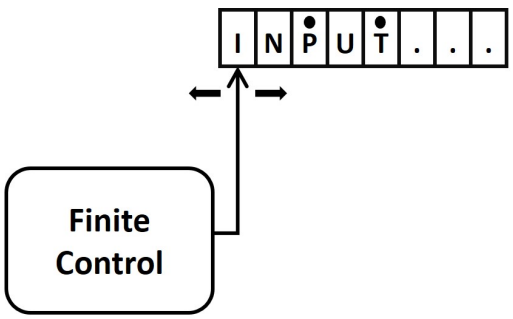

Figure 4.1. Schematic description of a finite automaton with inkdot advice.

A finite automaton that takes inkdots as advice does not have the power to mark cells on the input tape with inkdots, however, it can sense these marks if they are present on the currently scanned input cell. The inkdots are assumed to be placed prior to the execution of the machine in accordance with an advice function, $h : \mathbb{N} \to \mathcal{P}(\mathbb{N})$, which maps the length of the input string to a set of positions on the input string where the inkdots are to be placed. ($p \in h(n)$ implies $p \leq n$.)

A deterministic finite automaton (dfa) with inkdot advice can then be defined in a similar way to a standard unadvised dfa (see *e.g.* [17]), with an additional reference to the set $I = \{0, 1\}$ of values for expressing the presence (1) or absence (0) of an inkdot on a cell. (As we do not allow more than one inkdot an a cell $i \in I = \{0, 1\}$ can be taken as the number of inkdots on a cell as well.)

4.1.1. Deterministic Finite Automata with Advice Inkdots

Below is a formal definition for a deterministic finite automaton with advice inkdots.

Definition 4.1. *A deterministic finite automaton with advice inkdots is a 5-tuple* $(Q, \Sigma, \delta, q_0, F)$*, where*

(i) Q *is a finite set of internal states,*

(ii) Σ *is a finite set of symbols called the input alphabet,*

(iii) $\delta : Q \times \Sigma \times I \to Q$ *is the transition function, such that,* $\delta(q_1, \sigma, i) = q_2$ *implies that when the automaton is in state* $q_1 \in Q$ *and it scans* $\sigma \in \Sigma$ *on its input tape and it senses* $i \in I = \{0, 1\}$ *inkdots on the currently scanned input cell, a transition occurs which changes the state of the automaton to* $q_2 \in Q$*, meanwhile moving the input head to the next cell to the right of the current one,*

(iv) $q_0 \in Q$ *is the initial state,*

(v) $F \subseteq Q$ *is a set of accepting states.*

A dfa with inkdot advice $M = (Q, \Sigma, \delta, q_0, F)$ is said to accept a string $x \in \Sigma^*$ with the help of inkdots as advice if and only if M, when started at its initial state, q_0, reaches an accepting state, $q_f \in F$, after it scans the last symbol of the input by changing states as specified by its transition function, δ. x is said to be rejected by M in these settings if it is not accepted.

A language L defined on the alphabet Σ, is said to be recognized by such a dfa M with the help of an advice function $h : \mathbb{N} \to \mathcal{P}(\mathbb{N})$ iff

- $L = \{x \mid M$ accepts x with the help of inkdots, placed on the input tape, at the positions specified by $h(|x|)\}$, and
- $\overline{L} = \{x \mid M$ rejects x with the help of inkdots, placed on the input tape, at the positions specified by $h(|x|)\}$.

A language L is said to be recognized by such a dfa M using $f(n)$ advice inkdots if there exists an advice function $h : \mathbb{N} \to \mathcal{P}(\mathbb{N})$ with the following properties:

- $|h(n)| \leq f(n)$ for all $n \in \mathbb{N}$, and,
- M recognizes L with the help of $h(n)$.

4.1.2. Probabilistic Finite Automata with Advice Inkdots

Below is a formal definition for a probabilistic finite automaton with advice inkdots.

Definition 4.2. *A probabilistic finite automaton with advice inkdots is a 5-tuple* $(Q, \Sigma, \delta, q_0, F)$*, where*

(i) Q *is a finite set of internal states,*
(ii) Σ *is a finite set of symbols called the input alphabet,*
(iii) $\delta : Q \times \Sigma \times I \times Q \to [0,1]_{\mathbb{R}}$ *is the transition function such that,* $\delta(q_1, \sigma, i, q_2) = p$ *implies that when the automaton is in state* $q_1 \in Q$ *and it scans* $\sigma \in \Sigma$ *on its input tape and it senses* $i \in I = \{0, 1\}$ *inkdots on the currently scanned input cell, a transition occurs with probability* $p \in [0,1]_{\mathbb{R}}$ *which changes the state of the automaton to* $q_2 \in Q$*, meanwhile moving input head to the next cell to the right,*
(iv) $q_0 \in Q$ *is the initial state,*
(v) $F \subseteq Q$ *is a set of accepting states.*

The transition function δ *must satisfy the following well-formedness condition.*

$$\forall (q_1, \sigma, i) \in Q \times \Sigma \times I \sum_{(q_2) \in Q} \delta(q_1, \sigma, i, q_2) = 1.$$

A pfa with advice inkdots $M = (Q, \Sigma, \delta, q_0, F)$ is said to accept a string $x \in \Sigma^*$ with probability $p \in [0,1]_{\mathbb{R}}$ with the help of inkdots as advice if and only if M, when started at its initial state, q_0, reaches an accepting state, $q_f \in F$ with probability p, after it scans the last symbol of the input by changing states according to the probabilities specified by its transition function, δ. x is said to be rejected by M, in these settings with probability $1 - p$ if it is accepted with probability p.

A language L defined on the alphabet Σ, is said to be recognized with bounded error by such a pfa M with the help of an advice function $h : \mathbb{N} \to \mathcal{P}(\mathbb{N})$ if and only if there exists a constant $\rho \in (1/2, 1]_{\mathbb{R}}$ such that

- $L = \{x \mid M$ accepts x with probability $p \geq \rho$, with the help of inkdots, placed on the input tape, at the positions specified by $h(|x|)\}$, and
- $\overline{L} = \{x \mid M$ rejects x with probability $p \geq \rho$, with the help of inkdots, placed on the input tape, at the positions specified by $h(|x|)\}$.

A language L is said to be recognized with bounded error by such a pfa M with the help of $f(n)$ advice inkdots if there exists an advice function $h : \mathbb{N} \to \mathcal{P}(\mathbb{N})$ with the following properties:

- $|h(n)| \leq f(n)$ for all $n \in \mathbb{N}$, and,
- M recognizes L with bounded error, with the help of $h(n)$.

4.1.3. The Classes of Languages Recognized with Inkdot Advice

The language families associated with advice inkdots will be named according to an adaptation of the template V/F, introduced by Karp and Lipton where the fact that the advice in consideration is provided in the form of inkdots will be made explicit by use of the symbol $\odot$ in parentheses, as in $V/F(\odot)$. In that template F may either be a natural number, describing cases where the advised machine is supposed to use at most that constant number of inkdots regardless of the length of the input, or a function $f(n)$, indicating that at most $O(f(n))$ inkdots can appear in the advice for

inputs of length n.

The name of the class of languages corresponding to the unadvised version of the automaton in question will appear in our advised class names in place of the V item in the template. In most of the cases, deterministic finite automata will be considered as the underlying model of computation which will be reflected by use of REG in the corresponding class names. One-way Turing machines which use space bounded by an amount s is the only other model for which inkdots will be considered as advice as part of our discussion and in this case 1-DSPACE(s) will appear in the corresponding class names in place of V in the template.

We will also examine randomly placed inkdots as advice to the finite automata where the positions of the advice inkdots are randomly selected from a set of alternatives according to a prespecified probability distribution. In our class names, the use of such randomized advice will be indicated by the letter R appearing before the amount of the advice inkdots and we will also make it explicit that bounded-error language recognition is in consideration by adding the item *be* in superscript form, next to the original unadvised class name as in $\mathsf{REG}^{be}/R\text{-}a(\odot)$.

Table 4.1 provides a summary of the notation described above for the classes of languages that will be mentioned in the subsequent sections.

Table 4.1. Naming of the classes of languages corresponding to the finite and small space automata with inkdot advice.

	Automaton				Advice	
Class Name	Type	Space	Error	Input	Type	Amount
$\mathsf{REG}/n(\odot)$	dfa	const	none	real-time	deterministic	$O(n)$
1-DSPACE$(\log n)/1(\odot)$	1tm	$\log n$	none	one-way	deterministic	1
$\mathsf{REG}^{be}/R\text{-}n(\odot)$	dfa	const	be	real-time	random	$O(n)$

4.2. Inkdots vs. Prefix Strings as Advice

We start by establishing that the inkdot advice model is stronger than the prefix model, even when it is restricted to a single inkdot per advice.

Theorem 4.3. *For every* $k \in \mathbb{N}$, $\mathsf{REG}/k \subsetneq \mathsf{REG}/1(\odot)$.

Proof. We first show that $\mathsf{REG}/k \subseteq \mathsf{REG}/1(\odot)$ for all $k \in \mathbb{N}$. Let L be a language that is recognized by a dfa M using k bits of binary prefix advice. Without loss of generality, assume that L is defined on a binary alphabet. One can construct a finite automaton N that recognizes L with the help of a single inkdot as advice as follows.

N will use a lookup table to treat inputs shorter than 2^k bits on its own, without utilizing advice. For longer inputs, view each advice string given to M as a natural number b written with k bits. This information will be conveyed by placing the inkdot on the $(b+1)$'st symbol of the input to N.

Note that reading the prefix advice may bring M into one of at most of 2^k different states when it is positioned at the beginning of the actual input. N simulates at most 2^k different instances of M starting from each of those different initial states in parallel on the actual input. When it scans the inkdot, N uses the information encoded in the inkdot's position to pick one of the simulated machines, run it to its completion, and report its outcome as the result of its computation.

Having proven the subset relation, it remains to exhibit a language recognized by a finite automaton that takes an inkdot as advice but not by any dfa that takes prefix advice. Consider the language $\{a^m b^m \mid m \in \mathbb{N}\}$, which can not be recognized by any finite automaton with advice prefix, by Propositions 1 and 7 of [6]. An inkdot marking the $(n/2)+1$'st symbol for inputs of even length n is sufficient to help a dfa recognize this language, since the machine need only check that the string is of the form a^*b^*, and that the first b appears precisely at the marked position. □

4.3. Inkdots vs. the Advice Track

It is evident that inkdot advice is a special case of the advice track model, when the advice alphabet is restricted to be binary. Recall that $\mathsf{REG}/n(t)$ denotes the class of languages recognized by dfa's with the aid of advice written on a track using a t-ary alphabet.

Theorem 4.4. $\mathsf{REG}/n(\odot) = \mathsf{REG}/n(2)$.

Proof. Any inkdot pattern on the n-symbol-long input string corresponds to a unique n-bit advice string, with (say) 1's for the marked positions, and 0's for the rest. The way the advice track is accessed simultaneously with the input track makes the two simulations required for proving the equality trivial. □

The reader is thus referred to Section 2.3 (and to [7] and [9]) for what is already known about the capabilities and limitations of advice read from tracks with binary alphabets. In particular, Theorem 2 of [9] , which is cited as Fact 2.9 in Section 2.3, provides a straightforward characterization of REG/n, and can thus be used to show that certain languages are not in REG/n, and therefore neither in $\mathsf{REG}/n(\odot)$.

In order to show the difference between the inkdot model and the general advice track approach, we consider bigger track alphabets.

Theorem 4.5. $\mathsf{REG}/n(\odot) \subsetneq \mathsf{REG}/n(k)$ *for all* $k \in \mathbb{Z}$ *with* $k > 2$.

Proof. Since we have $\mathsf{REG}/n(\odot) = \mathsf{REG}/n(2)$ by Theorem 4.4 and since it is trivially true that $\mathsf{REG}/n(2) \subseteq \mathsf{REG}/n(k)$ for all $k \in \mathbb{Z}$ with $k > 2$, we just need to show the existence of languages that can be recognized by finite automata with the help of k-ary advice but not with the help of binary advice supplied on the advice track. Below, we will adopt a method used by Damm and Holzer in [6] which is based on the incompressibility of a random binary sequence.

Let $w = w_1 w_2 \cdots$ be an infinite (Martin-Löf) random binary sequence. Let w_i denote i'th symbol of w and let $w_{i:j}$ denote the segment of w which starts with w_i and ends with w_j, where $i \leq j$. As w is random, none of its prefixes can be compressed by more than an additive constant, *i.e.* there is a constant c such that $K(w_{1:n}) \geq n - c$ for all n where $K(w_{1:n})$ stands for the Kolmogorov complexity or equivalently, size of the minimum description of $w_{1:n}$. (For a precise definition of Kolmogorov complexity the reader may refer to [56] or to [57].) With reference to w, we will define a k-ary (for arbitrary $k > 2$) language, $\texttt{LRS}_w$ of words obtained by a special re-encoding of subwords of w as a result of which $\texttt{LRS}_w$ will have exactly one member, l_i, of each length $i \in \mathbb{Z}^+$. For $i \in \mathbb{Z}^+$, we first divide w into consecutive subwords s_i the lengths, $|s_i|$ of which will be specified further below.

We obtain each member l_i of $\texttt{LRS}_w$ from the corresponding subword s_i of w by first reading s_i as a binary number, then converting it to a k-ary number, and then padding with zeros to the left if necessary to make the result i symbols long. This relation can be expressed by a function f as $l_i = f(s_i, k, i)$. We are yet to specify the lengths of each subword s_i in w. We need to be able to encode the content of s_i into i k-ary symbols. This entails $|s_i| \leq \lfloor \log_2(k^i) \rfloor$. So we set $|s_i| = \lfloor \log_2(k^i) \rfloor$ for maximum compression.

Then we can formally define $\texttt{LRS}_w$ as

$$\texttt{LRS}_w = \{f(w_{a:b}, k, i) \mid a = \sum_{u=1}^{i-1} (\lfloor \log_2(k^u) \rfloor) + 1,\ b = \sum_{u=1}^{i} (\lfloor \log_2(k^u) \rfloor) \text{ for } i \in \mathbb{Z}^+\}.$$

Since $\texttt{LRS}_w$ has exactly one member of each length, it is obvious that providing this member as advice and letting the automaton check the equality of the advice and the input would suffice to recognize $\texttt{LRS}_w$ when a k-ary alphabet is used on the advice track. Therefore, we have $\texttt{LRS}_w \in \mathsf{REG}/n(k)$.

Now let us assume $\texttt{LRS}_w \in \mathsf{REG}/n(2)$, which would mean that there is a dfa M which recognizes $\texttt{LRS}_w$ with the help of binary advice on the advice track. Let

$a_1, a_2, \ldots, a_n$ be the binary advice strings to M for inputs of length $1, 2, \ldots, n$ respectively. Then Algorithm 4.2 working on the input $a_1, a_2, \ldots, a_n$ would compute and print the concatenation of the first n subwords $s_1, \ldots, s_n$ of w.

```
1: on input a_1, a_2, ..., a_n
2: result = blank
3: for i = 1 to n do
4:     for every k-ary string s of length i do
5:         Simulate M on input s with advice a_i
6:         if M accepts s then
7:             s_i = TranslateToBinary(s, ⌊log_2(k^i)⌋)
8:             result = concat(result, s_i)
9:             break
10:        end if
11:    end for
12: end for
13: print result
14: end
15: procedure TranslateToBinary(word, length)
16:     binary-number = transform word from k-ary to binary
17:     binary-word = pad binary-number with zeros to the left until it fits length
18:     return binary-word
19: end procedure
```

Figure 4.2. A short program for printing a prefix of the infinite random binary sequence w.

So Algorithm 4.2 (which is, say, c bits long) and the first n advice strings for M form a description of the prefix of w formed by concatenating the first n subwords $s_1, \ldots, s_n$. But this violates the incompressibility of w, since for large values of n, the total length of $s_1, \ldots, s_n$ is given by $\sum_{u=1}^{n}(\lfloor \log_2 k^u \rfloor)$, which will be approximately

$\log_2 k$ times the length of its description, $c + \sum_{u=1}^{n}(u)$. Therefore, we have

$$\mathtt{LRS}_w \notin \mathsf{REG}/n(2), \tag{4.1}$$

and hence

$$\mathtt{LRS}_w \notin \mathsf{REG}/n(\odot), \tag{4.2}$$

which suffices to show the truth of the theorem statement. □

4.4. Language Recognition with Varying Amounts of Inkdots

Having shown that a single inkdot is a more powerful kind of advice to dfa's than arbitrarily long prefix strings, and that no combination of inkdots could match the full power of the track advice model, we now examine the finer structure of the classes of languages recognized by dfa's advised by inkdots. A natural question to ask is: How does the recognition power increase when one allows more and more inkdots as advice? Or equivalently: How stronger does a finite automaton get when one allows more and more 1's in the binary advice string written on its advice track?

We start by establishing the existence of an infinite hierarchy in the class of languages recognized by dfa's aided with a constant number of inkdots, independent of the input length. It will be shown that $m + 1$ inkdots are stronger than m inkdots as advice to dfa's for any value of m. The following family of languages, defined for $m \in \mathbb{Z}^+$, on the binary alphabet $\{0, 1\}$, will be used in our proof:

$$\mathtt{LAELS}_m = \{(0^i 1^i)^{\lceil m/2 \rceil}(0^i)^{m+1-2\lceil m/2 \rceil} \mid i > 0\}.$$

Note that, $\mathtt{LAELS}_m$ is the set of strings that consist of $m + 1$ alternating equal-length sequences of 0's and 1's. $\mathtt{LAELS}_1$ and $\mathtt{LAELS}_2$, for example, coincide with the well known

languages shown below:

$$\texttt{LAELS}_1 = \{0^n 1^n \mid n > 0\}, \tag{4.3}$$

$$\texttt{LAELS}_2 = \{0^n 1^n 0^n \mid n > 0\}. \tag{4.4}$$

Theorem 4.6. *For every* $m \in \mathbb{Z}^+$, $\mathsf{REG}/(m-1)(\odot) \subsetneq \mathsf{REG}/m(\odot)$.

Proof. We start by showing that m inkdots of advice is sufficient to recognize language $\texttt{LAELS}_m$ for any m.

Observe that $\texttt{LAELS}_m$ has no member of length n if n is not divisible by $m+1$, and it has exactly one member of length n if n is divisible by $m+1$. A member string is made up of $m+1$ segments of equal length, each of which contains only one symbol. Let us call the positions $(\frac{n}{m+1}+1), (\frac{2n}{m+1}+1), \ldots, (\frac{mn}{m+1}+1)$ of a member of $\texttt{LAELS}_m$, where a new segment starts after the end of the previous one, the "border points". If m inkdots marking these m border points are provided as advice, a finite automaton can recognize $\texttt{LAELS}_m$ by simply accepting a string whose length is a multiple of $m+1$ if and only if the input consists of alternating runs of 0's and 1's, and that all and only the marked symbols are different than their predecessors in the string.

We have thus proven that

$$\texttt{LAELS}_m \in \mathsf{REG}/m(\odot) \text{ for } m \in \mathbb{Z}^+. \tag{4.5}$$

To show that $\texttt{LAELS}_m \notin \mathsf{REG}/(m-1)(\odot)$ for any m, suppose that there is a finite automaton M which recognizes the language $\texttt{LAELS}_m$ with the help of $m-1$ inkdots for some $m \in \mathbb{Z}^+$. Let q be the number of states of M, and let u be an integer greater than $4q^2$.

Note that the string $s = (0^u 1^u)^{\lceil m/2 \rceil}(0^u)^{m+1-2\lceil m/2 \rceil}$ is in $\texttt{LAELS}_m$, and so it should be accepted by M. We will use s to construct another string of the same length which would necessarily be accepted by M with the same advice, although it would not be a member of $\texttt{LAELS}_m$.

Note that there are m border points in s. Let us call the $(4q^2 + 1)$-symbol-long substring of s centered at some border point the *neighborhood* of that border point. Since M takes at most $m-1$ inkdots as advice, there should be at least one border point, say, b, whose neighborhood contains no inkdot. Without loss of generality, assume that position b of the string s contains a 1, so the neighborhood is of the form $0^{2q^2}1^{2q^2+1}$. Since M has only q states, and there is no inkdot around, M must "loop" (*i.e.* enter the same state repeatedly) both while scanning the 0's, and also while scanning the 1's of this neighborhood. Let d denote the least common multiple of the periods of the two loops (say, p_1 and p_2) described above. Note that as $p_1 \leq q$ and $p_2 \leq q$, d can not be greater than q^2.

Now consider the new string s' that is constructed by replacing the aforementioned neighborhood $0^{2q^2}1^{2q^2+1}$ in s with the string $0^{2q^2-d}1^{2q^2+d+1}$. s' is of the same length as s, and it is clearly not a member of $\texttt{LAELS}_m$, since it contains segments of different lengths. But M would nonetheless accept s' with the advice for this input length, since M's computation on s' would be almost the same as that on s, with the exception that it loops d/p_2 more times on the segment containing position b, and d/p_1 fewer times on the preceding segment, which is still long enough (*i.e.* at least q^2 symbols long) to keep M looping. M therefore enters the same states when it starts scanning each segment of new symbols during its executions on both s and s', and necessarily ends up in the same state at the end of both computations. This means that M accepts a nonmember of $\texttt{LAELS}_m$, which contradicts the assumption that M recognizes $\texttt{LAELS}_m$ with $m - 1$ inkdots as advice. □

It has been shown that every additional inkdot increases the language recognition power of dfa's that operate with constant amounts of advice. Extending the same

argument, we now prove that more and more languages can be recognized if one allows the amount of inkdots given as advice to be faster and faster increasing functions of the length of the input.

Theorem 4.7. *For every pair of functions $f, g : \mathbb{N} \to \mathbb{N}$ such that $f(n) \in \omega(1) \cap o(n)$, if there exists $n_0 \in \mathbb{Z}$ such that $g(n) < f(n) - 2$ for all $n > n_0$, then* $\mathsf{REG}/g(n)(\odot) \subsetneq \mathsf{REG}/f(n)(\odot)$.

Proof. Let f be a function on $\mathbb{N}$ such that $f(n) \in \omega(1) \cap o(n)$, and let $f'(n) = \lceil n/f(n) \rceil$.

Note that $f'(n) \in \omega(1) \cap o(n)$, and $f(n) \geq n/f'(n)$ for all n. Consider the language

$$\texttt{LBSES1}_f = \{w \mid w = w_1 \cdots w_i \cdots w_n \text{ where } w_i \in \{0,1\} \text{ and } w_i = 1 \text{ iff } i = kf'(n) \text{ for some } k \in \mathbb{Z}^+\}.$$

Each member w of $\texttt{LBSES1}_f$ is simply a binary string with equally spaced 1's. It contains 1's at the $f'(|w|)$'th, $2f'(|w|)$'th, etc. positions, and 0's everywhere else. Since the gaps between the 1's is $f'(|w|)$, *i.e.* an increasing function of the input length, sufficiently long members of $\texttt{LBSES1}_f$ can be "pumped" to obtain nonmembers, meaning that $\texttt{LBSES1}_f$ can not be recognized by a finite automaton without advice. However a finite automaton which takes inkdots as advice can recognize $\texttt{LBSES1}_f$ easily: The advice for length n consists simply of inkdots placed on the exact positions where 1's are supposed to appear in a member string of this length. As a member with length n contains at most $n/f'(n)$ 1's, $f(n)$ inkdots are sufficient. We conclude that

$$\texttt{LBSES1}_f \in \mathsf{REG}/f(n)(\odot). \tag{4.6}$$

Now, suppose that $\texttt{LBSES1}_f \in \mathsf{REG}/g(n)(\odot)$ for some function $g : \mathbb{N} \to \mathbb{N}$ such that $g(n) < f(n) - 2$. Then there should be a finite automaton M which would recognize $\texttt{LBSES1}_f$ with the help of at most $g(n)$ inkdots.

Note that the precise number of 1's in a member of $\texttt{LBSES1}_f$ of length n is given by

$$f''(n) = \lfloor n/f'(n) \rfloor = \lfloor n/\lceil n/f(n) \rceil \rfloor. \tag{4.7}$$

For large values of n, $f''(n)$ takes values in the set $\{f(n)-1, f(n)\}$, since $f(n) \in o(n)$. Therefore $g(n) < f(n) - 2$ for sufficiently long inputs implies that $g(n)$ inkdots will not be enough to mark all input positions where a member string of that length should contain a 1. In fact, recalling that the distance between 1's in the members of $\texttt{LBSES1}_f$ of length n is given by $f'(n)$, we see that sufficiently long members of $\texttt{LBSES1}_f$ must contain at least one 1 which has an inkdot-free neighborhood (in the sense of the term used in the proof of Theorem 4.6), where the all-0 segments to the left and the right of the 1 are long enough to cause M to loop. We can then use an argument identical to the one in that proof to conclude that a nonmember of $\texttt{LBSES1}_f$ also has to be accepted by M, reaching a contradiction. Hence we can conclude that

$$\texttt{LBSES1}_f \notin \mathsf{REG}/g(n)(\odot) \tag{4.8}$$

and therefore

$$\mathsf{REG}/g(n)(\odot) \subsetneq \mathsf{REG}/f(n)(\odot). \tag{4.9}$$

□

4.5. Succinctness Issues

In this section, we investigate the effects of advice on the succinctness of finite automata. In particular, we will demonstrate a family of languages where the sizes of the associated minimal automata depend on whether advice is available, and if so, in which format.

For a given integer $k > 1$, we define

$$\texttt{LMOD}_k = \{w \in \{0,1\}^* \mid |w| < k \text{ or } w_{i+1} = 1 \text{ where } i = |w| \mod k\}.$$

Theorem 4.8. *A dfa with no advice would need to have at least 2^k states in order to recognize $\texttt{LMOD}_k$.*

Proof. Let x, y be any pair of distinct strings, $x, y \in \{0,1\}^k$. There exists a position i such that $x_i \neq y_i$. Without loss of generality, assume that $x_i = 0$ and $y_i = 1$. Then for any string $z \in \{0,1\}^*$ with $i - 1 = |z| \mod k$, we have $xz \notin \texttt{LMOD}_k$ and $yz \in \texttt{LMOD}_k$. In other words, the index of $\texttt{LMOD}_k$ (in the sense of the Myhill-Nerode theorem (see *e.g.* [58]) is at least as big as the number of distinct strings in $\{0,1\}^k$, namely, 2^k. Therefore, a dfa would need at least that many states in order to recognize $\texttt{LMOD}_k$. □

Note that testing membership in $\texttt{LMOD}_k$ is as easy as checking whether the $i+1$'st symbol of the input is a 1 or a 0, where the length $|w|$ of the input word satisfies $|w| = mk + i$. The problem is that in the setting without the advice, the value i is learned only after the machine scans the last symbol. i, however is a function of the length of the input, and advice can be used to convey this information to the automaton at an earlier stage of its computation.

Theorem 4.9. $\texttt{LMOD}_k$ *can be recognized by a $(k+3)$-state dfa with the help of prefix advice. However, no dfa with fewer than k states can recognize* $\texttt{LMOD}_k$ *with prefix advice.*

Proof. We describe a $(k+3)$-state machine M_1, which takes binary prefix advice of length k to recognize $\texttt{LMOD}_k$. For inputs of length less than k, the advice is 0^k. For longer inputs of length $i \pmod k$, the advice is the string $0^i 10^{k-i-1}$.

M_1 is depicted in Figure 4.3, where double circles are used to indicate accepting states. M_1 remains at its initial state, which is an accepting state, as long as it scans 0's on the combined advice-input string. If it scans a 1, it attempts to verify if the

k'th symbol after that 1 is also a 1. If the input ends before that position is reached, M_1 accepts. If the input is sufficiently long to allow that position to be reached, M_1 accepts if it scans a 1 there, and rejects otherwise. It is easy to see that $k+3$ states are sufficient to implement M_1.

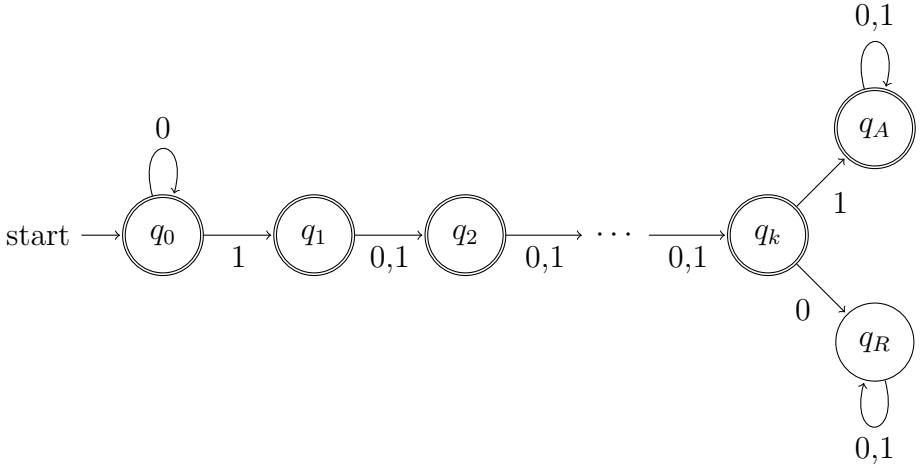

Figure 4.3. State diagram for a finite automaton with $k+3$ states that recognizes $\texttt{LMOD}_k = \{w \in \{0,1\}^* \mid |w| < k \text{ or } w_{i+1} = 1 \text{ where } i = |w| \mod k\}$ with prefix advice.

Let us now assume that a $(k-1)$-state finite automaton M_2 recognizes $\texttt{LMOD}_k$ with the help of prefix advice. Then M_2 must accept the string $s = 0^{k-1}10^{k^2+k-1}$, which is a member of $\texttt{LMOD}_k$, utilizing the advice for (k^2+2k-1)-symbol-long inputs. Since the 0 sequences to the left and right of the single 1 in s are of length at least $k-1$, M_2 must "loop" during its traversals of these sequences while scanning s. Let $p < k$ and $q < k$ be the periods of these loops to the left and right of 1, respectively. Now consider the string $s' = 0^{k-1+pq}10^{k^2+k-1-pq}$. Note that s' is of the same length as s, and therefore it is assigned the same advice as s. Also note that M_2 must enter the same states at the ends of the 0 sequences on both s' and s, since q additional iterations of the loop to the left of the 1 and p fewer iterations of the loop to the right of the 1 does not change the final states reached at the end of these 0 sequences. (The "pumped down" version of the zero sequence to the right of 1 is $k^2+k-1-pq$ symbols long and since we have $pq < k^2$ this means it is still sufficiently long (*i.e.* longer than k symbols) to ensure that it causes at least one iteration of the loop.) This implies that M_2 should accept s' as well, which is a contradiction, since s' is not a member of $\texttt{LMOD}_k$. □

Theorem 4.10. *There exists a dfa with just two states that can recognize* $\texttt{LMOD}_k$ *with the help of inkdot advice for any* k*.*

Proof. For inputs of length less than k, no inkdots are provided as advice. For longer inputs whose length is i mod k, the advice is an inkdot on the $i+1$'st symbol of the input. A dfa that accepts if and only if either senses no inkdots or if the single marked position contains a 1 can be constructed using two states, as seen in Figure 4.4. □

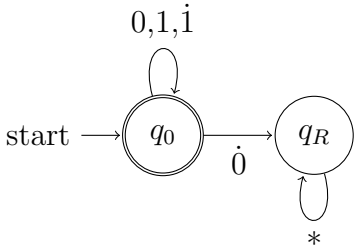

Figure 4.4. State diagram for a finite automaton with 2 states that recognizes $\texttt{LMOD}_k = \{w \in \{0,1\}^* \mid |w| < k \text{ or } w_{i+1} = 1 \text{ where } i = |w| \mod k\}$ with inkdot advice.

4.6. Randomized Inkdot Advice to Finite Automata

It is known that "randomized" advice, selected for each input from a set of alternatives according to a particular distribution, is even more powerful than its deterministic counterpart in several models (see [9] and [3]). In this section, we will show that a similar increase in power is also the case for the inkdot advice model, even with a restriction to a constant number of inkdots.

For each input length n, the advice is a set of pairs of the form $\langle I, p\rangle$, where I is an inkdot pattern to be painted on a string of length n, and p is the probability with which pattern I is to be used, with the condition that the probabilities add up to 1. Whenever the advised dfa is presented with an input, an inkdot pattern is selected from the advice with the corresponding probability. A language is said to be *recognized with bounded error* by such a machine if the automaton responds to each input string with

the correct accept/reject response with probability at least $\frac{2}{3}$. (Recall that the class of languages recognized with bounded error by a dfa aided with randomized advice in the advice track model is called REG/Rn in [9]. We call the corresponding classes for k inkdots $\mathsf{REG}^{be}/R\text{-}k(\odot)$ for $k \in \mathbb{Z}^+$, and $\mathsf{REG}^{be}/R\text{-}n(\odot)$ when there are no restrictions on the number of inkdots.)

By importing Proposition 16 in [9] about the advice track model directly to the inkdot advice case, one sees immediately that

$$\mathsf{REG}/n(\odot) \subsetneq \mathsf{REG}^{be}/R\text{-}n(\odot). \tag{4.10}$$

We will present a new result, showing that randomization adds power even when the number of inkdots is restricted to be a constant.

Theorem 4.11. $\mathsf{REG}/2(\odot) \subsetneq \mathsf{REG}^{be}/R\text{-}2(\odot)$.

Proof. As deterministic advice is a special case of the probabilistic version, the inclusion is trivial. In order to show the separation, consider the language

$$\mathtt{LAELS}_3 = \{0^m1^m0^m1^m \mid m > 0\},$$

defined as part of a family in Section 4.4. By the proof of Theorem 4.6, we have

$$\mathtt{LAELS}_3 \notin \mathsf{REG}/2(\odot). \tag{4.11}$$

So it remains to show that $\mathtt{LAELS}_3 \in \mathsf{REG}^{be}/R\text{-}2(\odot)$.

Recall from the proof of Theorem 4.6 that we call the $m+1$'st, $2m+1$'st, and $3m+1$'st positions of the string $0^m1^m0^m1^m$ the "border points". We build a dfa that will take two inkdots as advice to recognize $\mathtt{LAELS}_3$. In the randomized advice that will be given for inputs of length $4m$, the three pairs of border points $(m+1, 2m+1)$,

$(m+1, 3m+1)$, and $(2m+1, 3m+1)$ will be marked with probability $\frac{1}{3}$ each. The dfa simply checks whether its input is of the form $0^+1^+0^+1^+$, the input length is a multiple of 4, and also whether each inkdot that it sees is indeed on a symbol unequal to the previously scanned symbol. The machine accepts if and only if all these checks are successful.

If the input string is a member of $\texttt{LAELS}_3$, all checks will be successful, and the machine will accept with probability 1. For a nonmember string s of length $4m$ to be erroneously accepted, s must be of the form $0^+1^+0^+1^+$, and contain only one "false border point," where a new segment of 1's (or 0's) starts at an unmarked position after a segment of 0's (or 1's). But this can happen with probability at most $\frac{1}{3}$, since two of the three possible inkdot patterns for this input length must mark a position which contains a symbol that equals the symbol to its left. Hence we have shown that

$$\texttt{LAELS}_3 \in \mathsf{REG}^{be}/R\text{-}2(\odot) \tag{4.12}$$

and therefore we can conclude that

$$\mathsf{REG}/2(\odot) \subsetneq \mathsf{REG}^{be}/R\text{-}2(\odot) \tag{4.13}$$

□

4.7. Advised Computation with Arbitrarily Small Space

In this section, we will consider one-way Turing machines, instead of finite automata, as the underlying advised machine model in order to explore the effects of combining inkdot advice with non-constant, but still very small amounts of memory.

It is known that unadvised deterministic Turing machines with sublogarithmic space which scan their input once from left to right can only recognize regular languages (see [59]). Therefore such small amounts of additional workspace does not increase the

computational power of dfa's. On the other hand, sublogarithmic space can be fruitful for nonuniform computation. In [55] for example, $\log\log n$ space was proven to be necessary and sufficient for a demon machine, which is defined as a Turing machine whose work tape is limited with endmarkers to a prespecified size determined by the length of the input, to recognize the nonregular language $\{a^n b^n \mid n > 1\}$.

Deterministic machines, ($\log\log n$ space Turing machines in particular) was shown in [55] not to gain any additional computational power if they are also provided with the ability of marking one input tape cell with an inkdot. Below, we will show that a one-way Turing machine which has simultaneous access to arbitrarily slowly increasing amounts of space and one inkdot provided as advice can effectively use both of these resources in order to extend its power of language recognition. Note that the head on the single work tape of the TM is allowed to move in both directions. (Recall that the class of languages recognized by one-way input deterministic Turing machines which use $s(n)$ cells in their work tape when presented inputs of length n are denoted by 1-$\mathsf{DSPACE}(s(n))$. With an advice of k inkdots, the corresponding class is called 1-$\mathsf{DSPACE}(s(n))/k(\odot)$.)

Theorem 4.12. *For any slowly increasing function $g(n) : \mathbb{Z}^+ \to \mathbb{Z}^+$, where $g(n) \in \omega(1) \cap o(n)$,*

- 1-$\mathsf{DSPACE}(\log g(n)) \subsetneq$ 1-$\mathsf{DSPACE}(\log g(n))/1(\odot)$,
- $\mathsf{REG}/1(\odot) \subsetneq$ 1-$\mathsf{DSPACE}(\log g(n))/1(\odot)$.

In other words, automata with access to arbitrarily slowly increasing amounts of space and a single inkdot as advice are more powerful than those which lack either of these resources.

Proof. Both inclusions are straightforward, and we proceed to show the separation results. First, note that

$$1\text{-}\mathsf{DSPACE}(\log g(n)) \subseteq \mathsf{REG}/k(\odot) \tag{4.14}$$

for any such g and any k, since one-way deterministic Turing machines can not utilize sublogarithmic space in order to recognize nonregular languages (see [59]), and are therefore computationally equivalent to dfa's without advice. It will therefore be sufficient to demonstrate a language which is in 1-$\mathsf{DSPACE}(\log g(n))/1(\odot)$ but not in $\mathsf{REG}/1(\odot)$.

For this purpose, we define the language $\texttt{LIELS}_g$ over the alphabet $\{\#, 0, 1\}$ to be the collection of words made up of interrelated, equal-length sequences as in $s_1\#s_2\#\cdots\#s_m\#^+$, where

- for a member of length n and for $i = 1, 2, \ldots, m$; s_i is a subword of the form 0^*10^*, the length of which is given by $\lfloor g(n) \rfloor$, (meaning that m is at most $\lfloor n/(\lfloor g(n) \rfloor + 1) \rfloor$), and,
- For all s_i, p_i denotes the position of the symbol 1 within that subword, and $p_i \in \{p_{i-1} - 1, p_{i-1}, p_{i-1} + 1\}$ for $i = 2, 3, \ldots, m$.

As mentioned earlier, Fact 2.9 provides a useful tool for showing that certain languages are not in $\mathsf{REG}/n(\odot)$. We see that $\texttt{LIELS}_g$ is in REG/n if and only if the number of equivalence classes of the equivalence relation $\equiv_{\texttt{LIELS}_g}$ associated with $\texttt{LIELS}_g$ in the way described in Fact 2.9 is finite.

Considering pairs of distinct strings x and y of the form 0^*10^* such that $|x| = |y| = \lfloor g(n) \rfloor$, for values of n which are big enough that $g(n) \ll n$, one sees that $\equiv_{\texttt{LIELS}_g}$ must have at least $\lfloor g(n) \rfloor$ equivalence classes. This is because one has to distinguish $\lfloor g(n) \rfloor$ different subword patterns to decide whether the relationship dictated between subwords s_1 and s_2 holds or not. Noting that the number of equivalence classes of $\equiv_{\texttt{LIELS}_g}$ is not finite, we conclude that $\texttt{LIELS}_g$ is not even in REG/n, let alone in $\mathsf{REG}/n(\odot)$.

To prove $\texttt{LIELS}_g \in$ 1-$\mathsf{DSPACE}(\log g(n))/1(\odot)$, we describe a one-way Turing machine T that uses one inkdot and $O(\log g(n))$ space to recognize $\texttt{LIELS}_g$. The advice for strings of length n is a single inkdot on the $\lfloor g(n) \rfloor$'th position, *i.e.* where the last symbol of the first subword s_1 should appear in a member of the language. During its

left-to-right scan of the input, T performs the following tasks in parallel:

- T checks if its input is of the form $(0^*10^*\#)^+\#^*$, rejecting if this check fails.
- Using a counter on its work tape, T counts the rightward moves of the input head up to the point where the inkdot is scanned. Having thus "learned" the value $\lfloor g(n) \rfloor$, T notes the number of bits required to write this value on the work tape, and marks the tape so as to never use more than a fixed multiple of this number of cells. It compares the lengths of all subsequent subwords with this value, and rejects if it sees any subword of a different length. T also rejects if the cell with the inkdot does not contain the last symbol of the first subword.
- T checks the condition $p_i \in \{p_{i-1} - 1, p_{i-1}, p_{i-1} + 1\}$ for $i = 2, 3, \ldots, m$, by using another pair of counters to record the position of the symbol 1 in each subword, rejecting if it detects a violation.
- T accepts if it finishes scanning the input without a rejection by the threads described above.

Clearly, the amount of memory used by T is just a fixed multiple of $\log g(n)$.Therefore we have

$$\texttt{LIELS}_g \in 1\text{-}\mathsf{DSPACE}(\log g(n))/1(\odot), \tag{4.15}$$

which suffices to conclude that

$$1\text{-}\mathsf{DSPACE}(\log g(n)) \subsetneq 1\text{-}\mathsf{DSPACE}(\log g(n))/1(\odot), \tag{4.16}$$

and that

$$\mathsf{REG}/1(\odot) \subsetneq 1\text{-}\mathsf{DSPACE}(\log g(n))/1(\odot). \tag{4.17}$$

□

Let us also note that the argument of the proof above can be applied to the general advice track and advice tape models, showing that such small amounts of space are useful for advised automata of those types as well.

5. CONCLUSION

We introduced two novel models of advised finite automata -finite automata with advice tapes and finite automata with inkdot advice- and analyzed the power and weaknesses of advice in these models. We examined variations of these models including probabilistic and quantum versions and compared them among themselves and with the previously studied models of advised finite automata. In addition to many separations and collapses among the classes of languages that can be recognized in each settings we also obtained several results that show infinite hierarchies of language classes that can be recognized with increasing amounts of advice.

We first introduced a model of advised computation by finite automata where the advice is provided on a separate tape which enables the automata to use the advice more flexibly than the previously studied models of advised finite automata by enabling it to pause on the input tape while processing the advice, or vice versa. Many variants of this model were taken into account, where the advised automaton is classical or quantum, the tapes can be accessed in various alternative modes, and the advice is deterministic or randomized. The power of these variants are compared among themselves, and also with the corresponding instances of the alternative models in the literature. Below is a list of results which summarizes our contribution in this context.

- A finite automaton with an advice tape when it is restricted to have real-time access on both of its input and advice tapes is equivalent in power to a finite automaton with an advice track. We showed that allowing one-way access to either of the input or advice tapes, so that the automaton can stay put on this tape during a transition, brings in strictly more language recognition power to the advice tape model.
- We showed that arbitrarily small and slowly increasing amounts of advice is a fruitful resource for finite automata with advice tape with one-way input access. We further showed that there exist languages that can be recognized in this

setting and not by a finite automaton with an advice track that uses linearly sized advice.

- When restricted to constant-length advice, finite automata with advice tape is shown to be equivalent in power to finite automata with prefix advice even in two-way input access mode. However we proved that allowing increasing amounts of advice brings in more and more power so that it is possible to show an infinite hierarchy of language classes that can be recognized by finite automata with one-way input access which is assisted by increasing amounts of advice provided on a separate one-way advice tape.
- It is shown that exponentially long advice strings would be sufficient for finite automata with a two-way input and a real-time advice tape to recognize any language. A finite automaton with two-way input head is also shown to be strictly more powerful than a finite automaton with one-way input head when both of them are assisted with polynomially long advice strings on a one-way advice tape.
- We examined probabilistic advice function together with deterministic transitions and deterministic advice function together with probabilistic transition functions as alternative ways of integrating randomness into the model of finite automata with advice tapes. In both cases, it is shown that allowing bounded error would strictly increase the language recognition power of the model over the fully deterministic settings.
- We also considered quantum finite automata with advice tapes, noting first that this model is by definition more powerful than a previously introduced model of quantum finite automata with advice track. Then we showed that one-way access to an advice tape, even when it is empty, would bring in additional power both in bounded and unbounded-error language recognition to a quantum finite automata with real-time input head, over its probabilistic counterpart with the same set of settings.

Next, we introduced another novel method for providing advice to finite automata, which depends on marking some cells on the input tape with advice inkdots.

We examined the power of this model in various settings and compared the language recognition power of this model with the previously studied models. We established many relations including separations, collapses and hierarchies of language classes that can be recognized with different models of advised finite automata and varying amounts of advice as a computational resource. This analysis also covers issues such as the randomly placed inkdots as advice and the succinctness of the advised finite automata. We also discussed an extension of this model with access to secondary memory and obtained nontrivial results in this context as well. The list of items below provides a summary of these results.

- We showed that a single inkdot as advice to finite automata is strictly more powerful than the prefix advice no matter how long an advice string is being used in the prefix model. On the other hand, inkdot advice model is shown to be equivalent in power to advice track model restricted to a binary alphabet which is separately shown to be strictly less powerful than the advice track model with no restriction on the alphabet size.
- We showed that in case where the number of advice inkdots is restricted to be a constant, the class of languages that can be recognized in this setting, grows with every additional inkdot, hence forming an infinite hierarchy of language classes. Another infinite hierarchy is shown to exist among the classes of languages that can be recognized with the help of inkdots as advice in case where the amount of advice inkdots is allowed to grow with the input size rather than being a constant.
- We showed that finite automata with inkdots as advice can be more succinct than the other advised automata models and the pure unadvised automata in terms of the number of states required for recognizing certain languages.
- The fact that more languages can be recognized by finite automata which takes randomly placed inkdots as advice and which is allowed to make errors with a bounded probability can be inherited from similar results previously shown for advice track model. We showed that this is still true when the amount of randomly placed inkdots is restricted to be a constant rather than being bounded by the size of the input.

- We extended the model with access to infinite secondary memory and showed that automata with access to arbitrarily slowly increasing amounts of space and a single inkdot as advice are more powerful than those automata which lack either of these resources. This result is particularly interesting because such small amounts of space is known to be not fruitful for many well-known models of computation.

The advice tape model, we introduced in [2, 3] has recently been the subject of further investigation by Duris et al. in [60]. A question that we had listed among open questions in [2, 3] has been answered in this context, as it is shown that there exist languages which cannot be recognized with any amount of advice by a deterministic finite automaton with an advice tape which has one-way input access. `PAL`, the language of even length palindromes, in particular, is shown to be one such language. In [60] it is also shown that finite automata with advice tape, can not utilize more than exponential advice if its access to the input tape is restricted to be one way.

Nondeterministic version of advice tape model is also considered by Duris et al. in [60] and it is shown in this setting, that exponential advice suffices to recognize all languages even with one-way input access whereas there exist languages, such as `PAL`, which can not be recognized with the help of polynomial amounts of advice even in nondeterministic settings. When constant length advice is taken into consideration, nondeterministic finite automata are shown not to be more powerful than their deterministic counterparts.

Our analysis on advised computation can be extended in multiple ways such as considering further alternatives as means of providing advice to the underlying automata or by taking other models of computation as the base model of computation in the analysis. Applying computational limits on the advice functions to be used in our models in a similar way to that was discussed for advised Turing machines in [61], is another direction that has the potential to reveal interesting relations. Below is a list of open questions that could be the subject of further analysis in this line of research.

- Real-time probabilistic automata can be simulated by deterministic automata which receive coin tosses within a randomly selected advice string. It would be interesting to explore the relationship between deterministic automata working with randomized advice, and probabilistic automata working with deterministic advice further.
- Are there languages which cannot be recognized with bounded error with any amount of advice by a probabilistic or quantum finite automata with advice tapes?
- Can a quantum finite automaton assisted with a non-empty advice on its advice tape recognize any language which is impossible for probabilistic finite automata in the same settings?
- Are there hierarchies like those in Theorems 4.6 and 4.7 for the case of randomly placed advice inkdots as well?
- We considered only deterministic transition models for our machines that takes inkdot advice. How would things change if we allowed probabilistic or quantum models?
- Can the inkdot advice model be applied to other machine models, like pushdown or counter automata?
- In what other ways we can provide advice to "weaker" models of computation so that we can establish nontrivial relations?
- What would be the effects of applying limits on the computational power for the source of the advice?

To conclude, it can be stated that the analysis of advised computation is the focus of a rich past literature which has many strong relations with the core problems in the closely related domains such as computability, computational complexity, automata theory and formal language theory. Research in this domain uncovered many relations of advice also with seemingly unrelated problems in a wide range of other fields which can be as diverse as cryptography (see *e.g.* [62, 63]), neural networks (see *e.g.* [64–66]) and pseudorandomness (see *e.g.* [67, 68]). This literature keeps growing with the contribution of recent research activities including ours: new ways of reasoning on

the roles of advice are being investigated and these techniques are being examined in relation with a growing set of computational phenomena. As a result, the field of advised computation is rich in terms of questions which are yet to be answered and research directions which are open for further investigation.

REFERENCES

1. Moore, C. and S. Mertens, *The Nature of Computation*, Oxford University Press, Inc., New York, NY, USA, 2011.

2. Küçük, U., A. C. C. Say and A. Yakaryılmaz, "Finite Automata with Advice Tapes", M.-P. Béal and O. Carton (Editors), *Developments in Language Theory*, pp. 301–312, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

3. Küçük, U., A. C. C. Say and A. Yakaryılmaz, "Finite Automata with Advice Tapes", *Int. J. Found. Comput. Sci.*, Vol. 25, No. 8, pp. 987–1000, 2014.

4. Küçük, U., A. C. C. Say and A. Yakaryilmaz, "Inkdots as advice for finite automata", *Discrete Mathematics & Theoretical Computer Science*, Vol. 19, No. 3, 2017.

5. Karp, R. and R. Lipton, "Turing machines that take advice", *L'Enseignement Mathematique*, Vol. 28, pp. 191–209, 1982.

6. Damm, C. and M. Holzer, "Automata That Take Advice", *Mathematical Foundations of Computer Science 1995, 20th International Symposium, Proceedings*, Vol. 969 of *Lecture Notes in Computer Science*, pp. 149–158, Springer, 1995.

7. Tadaki, K., T. Yamakami and J. C. Lin, "Theory of one-tape linear-time Turing machines", *Theoretical Computer Science*, Vol. 411, No. 1, pp. 22 – 43, 2010.

8. Yamakami, T., "Swapping Lemmas for Regular and Context-Free Languages with Advice", *CoRR*, Vol. abs/0808.4122, 2008, `http://arxiv.org/abs/0808.4122`.

9. Yamakami, T., "The Roles of Advice to One-Tape Linear-Time Turing Machines and Finite Automata", *Int. J. Found. Comput. Sci.*, Vol. 21, No. 6, pp. 941–962, 2010.

10. Yamakami, T., "Immunity and pseudorandomness of context-free languages", *Theoretical Computer Science*, Vol. 412, No. 45, pp. 6432 – 6450, 2011.

11. Yamakami, T., "One-Way reversible and quantum finite automata with advice", *Proceedings of the 6th international conference on Language and Automata Theory and Applications*, LATA'12, pp. 526–537, Springer-Verlag, Berlin, Heidelberg, 2012.

12. Yamakami, T., "One-way reversible and quantum finite automata with advice", *Information and Computation*, Vol. 239, pp. 122 – 148, 2014.

13. Yamakami, T., "Pseudorandom generators against advised context-free languages", *Theoretical Computer Science*, Vol. 613, pp. 1 – 27, 2016.

14. Freivalds, R., "Amount of nonconstructivity in deterministic finite automata", *Theoretical Computer Science*, Vol. 411, No. 38, pp. 3436 – 3443, 2010, implementation and Application of Automata (CIAA 2009).

15. Agadzanyan, R. and R. Freivalds, "Finite State Transducers with Intuition", C. S. Calude, M. Hagiya, K. Morita, G. Rozenberg and J. Timmis (Editors), *Unconventional Computation*, Vol. 6079 of *Lecture Notes in Computer Science*, pp. 11–20, Springer Berlin Heidelberg, 2010.

16. Freivalds, R., "Multiple Usage of Random Bits in Finite Automata", M. Agrawal, S. B. Cooper and A. Li (Editors), *Theory and Applications of Models of Computation*, pp. 537–547, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

17. Sipser, M., *Introduction to the Theory of Computation*, Thomson Course Technology, second edn., 2006.

18. Arora, S. and B. Barak, *Computational Complexity: A Modern Approach*, Cambridge University Press, New York, NY, USA, 1st edn., 2009.

19. Pippenger, N., "On Simultaneous Resource Bounds", *Proceedings of the 20th Annual Symposium on Foundations of Computer Science*, SFCS '79, pp. 307–311, IEEE Computer Society, Washington, DC, USA, 1979.

20. Balcazar, J. L., J. Diaz and J. Gabarro, *Structural Complexity 1*, Springer-Verlag, Berlin, Heidelberg, 1988.

21. Chandra, A. K. and L. J. Stockmeyer, "Alternation", *17th Annual Symposium on Foundations of Computer Science (sfcs 1976)*, pp. 98–108, Oct 1976.

22. Schaefer, T. J., "On the complexity of some two-person perfect-information games", *Journal of Computer and System Sciences*, Vol. 16, No. 2, pp. 185 – 225, 1978.

23. Stockmeyer, L. J., "The polynomial-time hierarchy", *Theoretical Computer Science*, Vol. 3, No. 1, pp. 1 – 22, 1976.

24. Graham, R., B. Rothschild and J. Spencer, *Ramsey Theory*, A Wiley-Interscience publication, Wiley, 1990.

25. Landman, B. and A. Robertson, *Ramsey Theory on the Integers: Second Edition*, Student Mathematical Library, American Mathematical Society, 2014.

26. Hopcroft, J. E. and J. D. Ullman, *Formal Languages and Their Relation to Automata*, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1969.

27. Mundhenk, M. and R. Schuler, "Random languages for nonuniform complexity classes", *Journal of Complexity*, Vol. 7, No. 3, pp. 296 – 310, 1991.

28. Balcázar, J. L., J. Díaz and J. Gabarró, "Uniform characterizations of non-uniform complexity measures", *Information and Control*, Vol. 67, No. 1, pp. 53 – 69, 1985.

29. Hennie, F., "One-tape, off-line Turing machine computations", *Information and Control*, Vol. 8, No. 6, pp. 553 – 578, 1965.

30. Kondacs, A. and J. Watrous, "On the power of quantum finite state automata", *Proceedings 38th Annual Symposium on Foundations of Computer Science*, pp. 66–75, Oct 1997.

31. Brodsky, A. and N. Pippenger, "Characterizations of 1-Way Quantum Finite Automata", *SIAM Journal on Computing*, Vol. 31, No. 5, pp. 1456–1478, 2002.

32. Ambainis, A. and R. Freivalds, "1-way quantum finite automata: strengths, weaknesses and generalizations", *Proceedings 39th Annual Symposium on Foundations of Computer Science (Cat. No.98CB36280)*, pp. 332–341, Nov 1998.

33. Balcázar, J. L., J. Díaz and J. Gabarró, *Structural Complexity 2*, Springer-Verlag, Berlin, Heidelberg, 1990.

34. Meyer, A. and E. McCreight, "Computationally complex and pseudo-random zero-one valued functions", Z. Kohavi and A. Paz (Editors), *Theory of Machines and Computations*, pp. 19 – 42, Academic Press, 12 1971.

35. Wilber, R. E., "Randomness and the density of hard problems", *24th Annual Symposium on Foundations of Computer Science (sfcs 1983)*, pp. 335–342, Nov 1983.

36. Yao, A. C., "Theory and application of trapdoor functions", *23rd Annual Symposium on Foundations of Computer Science (sfcs 1982)*, pp. 80–91, Nov 1982.

37. Freivalds, R., "Rūsiņš Freivalds: Complexity of Probabilistic Versus Deterministic Automata. Baltic Computer Science 1991: 565-613", Vol. 502, pp. 565–613, 01 1991.

38. Freivalds, R., "Non-Constructive Methods for Finite Probabilistic Automata", *Int. J. Found. Comput. Sci.*, Vol. 19, No. 3, pp. 565–580, 2008.

39. Martin-Löf, P., "The definition of random sequences", *Information and Control*,

Vol. 9, No. 6, pp. 602 – 619, 1966.

40. Aardenne-Ehrenfest, van, T. and N. Bruijn, de, "Circuits and trees in oriented linear graphs", *Simon Stevin : Wis- en Natuurkundig Tijdschrift*, Vol. 28, pp. 203–217, 1951.

41. Kozen, D. C., *Automata and Computability*, Springer-Verlag, Berlin, Heidelberg, 1st edn., 1997.

42. Dwork, C. and L. Stockmeyer, "Finite state verifiers I: The power of interaction", *Journal of the ACM*, Vol. 39, No. 4, pp. 800–828, 1992.

43. Goldreich, O., *Computational Complexity: A Conceptual Perspective*, Cambridge University Press, 2008.

44. Yakaryılmaz, A. and A. C. Say, "Unbounded-error quantum computation with small space bounds", *Information and Computation*, Vol. 209, No. 6, pp. 873 – 892, 2011.

45. Hirvensalo, M., "Quantum Automata with Open Time Evolution", *International Journal of Natural Computing Research (IJNCR)*, Vol. 1, No. 1, pp. 70–85, 2010.

46. Ambainis, A. and J. Watrous, "Two-way finite automata with quantum and classical states", *Theoretical Computer Science*, Vol. 287, No. 1, pp. 299 – 311, 2002, natural Computing.

47. Zheng, S., D. Qiu, L. Li and J. Gruska, *One-Way Finite Automata with Quantum and Classical States*, pp. 273–290, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

48. Yakaryilmaz, A., R. Freivalds, A. C. C. Say and R. Agadzanyan, "Quantum computation with write-only memory", *Natural Computing*, Vol. 11, No. 1, pp. 81–94, 2012.

49. Say, A. C. and A. Yakaryılmaz, *Quantum Finite Automata: A Modern Introduction*, pp. 208–222, Springer International Publishing, Cham, 2014.

50. Nielsen, M. A. and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition*, Cambridge University Press, New York, NY, USA, 10th edn., 2011.

51. Gruska, J., *Quantum Computing*, Advanced topics in computer science series, McGraw-Hill, 1999.

52. Shepherdson, J. C., "The reduction of two–way automata to one-way automata", *IBM Journal of Research and Development*, Vol. 3, pp. 198–200, 1959.

53. Freivalds, R., "Fast Probabilistic Algorithms", *Mathematical Foundations of Computer Science 1979*, Vol. 74 of *Lecture Notes in Computer Science*, pp. 57–69, 1979.

54. Szepietowski, A., *Turing Machines with Sublogarithmic Space*, Ernst Schering Research Foundation Workshops, Springer, 1994.

55. Ranjan, D., R. Chang and J. Hartmanis, "Space bounded computations: review and new separation results", *Theoretical Computer Science*, Vol. 80, No. 2, pp. 289 – 302, 1991.

56. Li, M. and P. Vitányi, *An Introduction to Kolmogorov Complexity and Its Applications (2Nd Ed.)*, Springer-Verlag, Berlin, Heidelberg, 1997.

57. Downey, R. and D. Hirschfeldt, *Algorithmic Randomness and Complexity*, Springer-Verlag, Berlin, Heidelberg, 2010.

58. Hopcroft, J. E. and J. D. Ullman, *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley Publishing Company, 1979.

59. Stearns, R. E., J. Hartmanis and P. M. L. II, "Hierarchies of memory limited

computations", *6th Annual Symposium on Switching Circuit Theory and Logical Design, Ann Arbor, Michigan, USA, October 6-8, 1965*, pp. 179–190, 1965.

60. Duris, P., R. Korbas, R. Královic and R. Královic, "Determinism and Nondeterminism in Finite Automata with Advice", *Adventures Between Lower Bounds and Higher Altitudes - Essays Dedicated to Juraj Hromkovič on the Occasion of His 60th Birthday*, pp. 3–16, 2018.

61. Köbler, J. and T. Thierauf, "Complexity-Restricted Advice Functions", *SIAM J. Comput.*, Vol. 23, No. 2, pp. 261–275, 1994.

62. Blum, M., A. D. Santis, S. Micali and G. Persiano, "Noninteractive Zero-Knowledge", *SIAM J. Comput.*, Vol. 20, No. 6, pp. 1084–1118, 1991.

63. Zheng, Y., T. Matsumoto and H. Imai, "Structural Properties of One-Way Hash Functions", A. J. Menezes and S. A. Vanstone (Editors), *Advances in Cryptology-CRYPT0' 90*, pp. 285–302, Springer Berlin Heidelberg, Berlin, Heidelberg, 1991.

64. Balcazar, J. L., R. Gavalda and H. T. Siegelmann, "Computational Power of Neural Networks: A Characterization in Terms of Kolmogorov Complexity", *IEEE Trans. Inf. Theor.*, Vol. 43, No. 4, pp. 1175–1183, Sep. 2006.

65. Síma, J. and P. Orponen, "General-Purpose Computation with Neural Networks: A Survey of Complexity Theoretic Results", *Neural Computation*, Vol. 15, No. 12, pp. 2727–2778, 2003.

66. Cabessa, J. and H. T. Siegelmann, "The Computational Power of Interactive Recurrent Neural Networks", *Neural Computation*, Vol. 24, No. 4, pp. 996–1019, 2012.

67. Trevisan, L. and S. Vadhan, "Pseudorandomness and average-case complexity via uniform reductions", *Proceedings 17th IEEE Annual Conference on Computational Complexity*, pp. 103–112, 2002.

68. Vadhan, S. P., "Pseudorandomness", *Foundations and Trends in Theoretical Computer Science*, Vol. 7, No. 1-3, pp. 1–336, 2012.