ON TRAVERSAL SEQUENCES, EXPLORATION SEQUENCES AND COMPLETENESS OF KOLMOGOROV RANDOM STRINGS

BY MICHAL KOUCKÝ

A dissertation submitted to the

Graduate School—New Brunswick

Rutgers, The State University of New Jersey

in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

Graduate Program in Computer Science

Written under the direction of

Dr. Eric Allender

and approved by

New Brunswick, New Jersey

May, 2003

ABSTRACT OF THE DISSERTATION

On Traversal Sequences, Exploration Sequences and Completeness of Kolmogorov Random Strings

by Michal Koucký

Dissertation Director:
Dr. Eric Allender

Traversal sequences were defined by Aleliunas et al. (1979) as a tool for the study of undirected s-t-connectivity. In the first part of this thesis we study traversal sequences. We improve on previous results and we present a simple construction of polynomial length universal traversal sequences for 2-regular graphs. These universal traversal sequences are log-space (even NC^1) constructible and are of length $O(n^{4.03})$.

Further, we introduce a new notion of traversal sequences that we call *exploration sequences*. Exploration sequences share many properties with the original traversal sequences but they also exhibit some new properties. For instance, they have an ability to backtrack, and their random properties are robust under choice of the probability distribution on labels. We present simple constructions of polynomial length universal exploration sequences for several classes of graphs, e.g., 2-regular graphs, cliques, expanders, trees. These constructions do not obey previously known lower bounds on the length of universal traversal sequences thus, they highlight another difference between exploration and traversal sequences. We also show that universal traversal sequences can be efficiently converted into universal exploration sequences. Further, we show certain self-correcting properties of traversal and exploration sequences and we propose a candidate universal exploration sequence.

In the second part of this thesis we study Kolmogorov complexity, which is a measure of randomness of finite strings, and its resource-bounded variants. We show that sets consisting of strings of high resource-bounded Kolmogorov complexity provide examples of sets that are complete for complexity classes in which they naturally lie under probabilistic and non-uniform reductions whereas they are provably not complete under the usual many-one reductions.

Further, our results imply that under certain intractability assumptions, resource-bounded Kolmogorov complexity of a string as well as circuit size, formula size and branching program size of a Boolean function can not be efficiently approximated.

We also study unbounded Kolmogorov complexity. We show completeness results in that setting and point out certain inconsistencies caused by the usual definition of Kolmogorov complexity. We also consider the problem of deterministically generating a Kolmogorov random string when the set of Kolmogorov random strings is given to us as an oracle.

Acknowledgements

This work does not stand in a vacuum. It builds upon and is influenced by work, knowledge and wisdom of many others. Among those there are several people I have the pleasure to know personally who influenced me and in turn this work. I am grateful to all of them.

Over the course of the past several years Eric Allender was my guide. I am grateful for enlightening discussions with him, for his continuous support of me during my graduate study and for helping me on many occasions going far beyond our work. He is truly my friend. It is always a pleasure to talk to Mike Saks, an awesome teacher, discussions with whom are enlightening, too. Navin Goyal is a friend to me and I greatly benefited from discussions with him as well as with my other friends and fellow students Detlef Ronneburger and Sambuddha Roy. I am pleased to have Dieter van Melkebeek as a friend. I benefited from many stimulating discussions with him. My work directly benefited from discussions and friendship with Yuval Ishai, Venkatesh S., Mario Szegedy, Muthu. Friends like Marcelo Mydlarz, Ofer Melnik, Stefan Langerman and Sachin Lodha helped to keep me going and influenced my work, too. I also benefited from people attending our reading seminars and theory lunches as well as from seminars at DIMACS and IAS in Princeton.

I might have not choose to work in complexity theory without an influence of Antonín Kučera and I would not be writing these acknowledgments at all if there were not all my teachers, from the elementary school till the university, most notably Kristýna Krupková, pí. Kršková, pí. Saksová, Emílie Habětínová, Boženka Lesná, pí. Nejedlá, Jiří Kocourek, Václav Koubek, and also my childhood friends Míla Kořínek and Michal Kubeček.

There would be no education for me without my friends and family.

I already mentioned my friends in my second home, New Jersey. I could not survive there without friendship and encouragement of Blanka Kopřivová. Always inspiring were my emails and discussions with Tom Holan, Jára Zamastil and Ondra Kárný. I draw my inspiration also from Bára Kolářová, Lenka Kebortová, Lucka Pelikánová, Lenka Tahalová, Honza Kotas, Adam Votruba, Honza Vondrák, Franta Veselý and all my other friends. I am glad for Miculka.

The greatest influence on me of all have my parents, Klára and Karel Koucký, my sister Barča and my brother Ondra. I am grateful for encouragements from my wonderful aunt Lída and uncle Tonda, my grandmother as well as my other aunts, my uncle, my cousins and nieces.

Let me now conclude with acknowledgments that specifically pertain to the content of this thesis. The work in Chapter 4 appeared in Koucký (2003) and while working on this paper I benefited from interaction with Eric Allender, Mike Saks, Navin Goyal and Dieter van Melkebeek. The content of Chapter 5, Sections 5.1 through 5.2, comes from Koucký (2002). Eric Allender, Navin Goyal, Jeff Kahn, János Komlós, Sachin Lodha, Dieter van Melkebeek, Mike Saks, and Venkatesh Srinivasan as well as many valuable comments of the anonymous referees were helpful during work on this paper. The results in Section 5.3 would not be possible without an enlightening discussion with Howard Karloff and Omer Reingold. Results in Part II are not by any means results of a single person. The results in Chapter 8 and a part of Chapter 9 appeared in Allender, Buhrman, Koucký, van Melkebeek & Ronneburger (2002) and are result of collaborative research effort with Eric Allender, Harry Buhrman, Dieter van Melkebeek and Detlef Ronneburger. While working on these results we benefited from discussions with Lance Fortnow, Kolya Vereshchagin, Steven Rudich, Valentine Kabanets, and Manindra Agrawal. In Section 9.1 there are results obtained together with Harry Buhrman during my visit at CWI, Amsterdam. I appreciate the hospitality of Harry Buhrman, his family and Troy Lee while at CWI. We benefited from discussions with Kolya Vereshchagin, Troy Lee, Hein Roehrig, John Tromp and Eric Allender. Some of the results in Chapter 10 will appear in Allender, Koucký, Ronneburger & Roy (2003) and are result of collaborative effort with Eric Allender, Detlef Ronneburger and Sam

Roy.

I am grateful to Eric Allender for reading the entire thesis and giving me many comments on it.

During my graduate study I was partially supported by NSF grants CCR-9734918 and CCR-0104823, by DIMACS Graduate Student Fellowship NSF CCR-9906105 and by a travel grant of Nadání Josefa, Marie a Zdeňky Hlávkových.

Dedication

To my parents.

Table of Contents

List of Figures

Chapter 1

Introduction

Complexity theory studies how much of a given resource (such as time and space) is necessary to solve different computational problems. The way to formalize this study is by means of complexity classes. A complexity class is a class of computational problems that can be solved within specific resource bounds in a particular computational model. The fundamental questions concern the relationships among different complexity classes; for instance, which classes are equal and which differ? Some important tools for the study of these relationships are provided by *complete problems*, which are the hardest problems inside their respective classes. If a complete problem of one class belongs also to another class then the former class is a sub-class (not necessarily proper) of the latter one.

In Part I we contribute to the study of space-bounded computation in different computational models. In Part II we contribute to the study of resource-bounded Kolmogorov complexity and to study of hardness and completeness of sets of strings of high resource-bounded Kolmogorov complexity.

Part I focuses on traversal sequences. Traversal sequences are a tool for study of the s-t-connectivity on undirected graphs. Different variants of the s-t-connectivity problem represent complete problems for space-bounded computation in different computational models. The most general version of the s-t-connectivity problem, the s-t-connectivity on directed graphs, is a complete problem for nondeterministic computations running in logarithmic space. On the other hand, s-t-connectivity on forests is a complete problem for deterministic computations running in logarithmic space. In between these two extremes there is the s-t-connectivity on undirected graphs which is complete for so-called *symmetric computations* in logarithmic space. (See Chapter 3 for definitions.)

A commonly held belief is that the classes of problems that are solvable by a symmetric computation running in logarithmic space and a deterministic computation running in logarithmic space coincide. Traversal sequences could be useful in proving this equality. To augment the study of traversal sequences, we also introduce a new notion of traversal sequences that we call *exploration sequences*. We study properties of exploration sequences and we show that they seem to be even better suited for proving the above equality.

In Part II we study Kolmogorov complexity and sets of strings of high Kolmogorov complexity. Kolmogorov complexity is a measure on finite strings that tries to capture the amount of non-computable randomness that is contained in these strings. We study sets of strings of high Kolmogorov complexity, i.e., sets of Kolmogorov random strings. In addition to the usual notion of Kolmogorov complexity (which is "resource-unbounded"), we also study different variants of resource-bounded Kolmogorov complexity and sets of strings of high complexity with respect to these measures. What we show is that – contrary to prior indication – these sets are complete for classes in which they naturally reside. Hence, these sets give a fundamentally new type of complete sets. We also study the question of estimating the resource-bounded Kolmogorov complexity of a string and we show that the resource-bounded Kolmogorov complexity cannot be approximated well unless some unlikely complexity assumptions are true. Our results regarding resource-unbounded Kolmogorov complexity question the commonly-used definition of Kolmogorov complexity.

Part I

Traversal Sequences

Chapter 2

Introduction

Graph s-t-connectivity is a fundamental and widely studied problem. In this problem we are given a graph G and two vertices s and t, and we should decide whether there is a path from s to t. This problem is of great interest not only in theory. It occurs also in daily life for example while walking in unknown city with a help of a map or when two feet of snow cover all the roads, some of the roads get plowed and we have to figure out whether it is possible for us to get to work (New Jersey, February 18, 2003).

In computer science this problem is studied for two main reasons. First, it is the heart of many practical applications and second, its variants capture the computational power of different computational devices. The study of these variants of s-t-connectivity aims at understanding the computational power of those devices.

The most general version of the s-t-connectivity problem, the s-t-connectivity on directed graphs, is a complete problem for nondeterministic computations running in logarithmic space. On the other hand, the s-t-connectivity on forests is a complete problem for deterministic computations running in logarithmic space. In between these two extremes there is the s-t-connectivity on undirected graphs which is complete for so-called *symmetric computations* in logarithmic space. (Symmetric computation was defined by Lewis & Papadimitriou (1982) and it is a nondeterministic computation in which for every move there is also a reverse move, i.e., if it is legal to go from one configuration to another one then it is also legal to go from the latter one to the former one.)

Therefore the study of different variants of s-t-connectivity sheds light on the relationships among deterministic, nondeterministic and symmetric computation as well as randomized computation (as we shall see later).

This work focuses on *traversal sequences* which are one of the tools that is used in study of undirected s-t-connectivity. Traversal sequences were proposed by Cook in the late 1970's to be an analog of pebble-moving automata. The idea is that a traversal sequence will guide a walk of the pebble in a graph. The graph will have all the edges that leave a vertex v labeled by distinct labels from $\{0, \ldots, \deg(v)-1\}$ and the traversal sequence will be a sequence of these labels.

An exceptional place among traversal sequences is possessed by so-called *universal traversal sequences*. A traversal sequence t is called universal for a class of graphs $\mathcal{G}$ if t guides every walk in every graph from $\mathcal{G}$ so that all vertices in the traversed graph are visited. A fundamental result of Aleliunas, Karp, Lipton, Lovász & Rackoff (1979) shows that there are polynomial length traversal sequences that are universal for the class of connected undirected graphs on n vertices. However, the proof of Aleliunas et al. (1979) is probabilistic so it does not yield an explicit construction of such sequences. If there were a universal traversal sequence for 3-regular undirected graphs constructible in logarithmic space then the undirected s-t-connectivity problem would be solvable in deterministic logarithmic space and hence, symmetric and deterministic logarithmic space would coincide. (In deterministic logarithmic space (even in NC^1) it is possible to reduce an s-t-connectivity question for an undirected graph to an s-t-connectivity question for an undirected 3-regular graph.) The current knowledge about these classes suggests that they indeed coincide and it is broadly believed that universal traversal sequences constructible in logarithmic space do exist.

Since the introduction of traversal sequences, many different traversal sequences that are universal for certain classes of graphs were explicitly constructed, and several extensions and generalizations of traversal sequences has been proposed. In this work we survey this development and we present a new construction of universal traversal sequences for 2-regular graphs. Further we propose a new notion of traversal sequences that we call *exploration sequences*. We study this new notion, we present an explicit construction of universal exploration sequences for several classes of graph, e.g., 2-regular graphs, cliques, expanders. We also present an efficient algorithm for converting universal traversal sequences into universal exploration sequences and we propose a candidate for a universal exploration sequence.

The organization of this part is as follows. In Chapter 3 we present necessary definitions and we survey what is currently known about space complexity of undirected s-t-connectivity. In Chapter 4 we review the current knowledge regarding traversal sequences and we present the new construction of a universal traversal sequence for 2-regular graphs. In Chapter 5 we introduce exploration sequences which are generalization of traversal sequences and study their properties. There we also establish the relationship between universal traversal and exploration sequences.

Chapter 3

The s-t-connectivity Problem

In this chapter we survey the development of knowledge concerning s-t-connectivity. We focus mainly on s-t-connectivity in undirected graphs. We also review basic definitions and properties of different complexity classes that are of our interest.

3.1 Definitions and Basic Properties

In this section we review the basic notation and facts concerning complexity classes that are relevant to us. The notation that we use is rather standard and for further discussion we refer a reader to a textbook by Papadimitriou (1994).

Throughout this work, $\mathbf{Z}$ denotes the set of integers, $\mathbf{N}$ denotes the set of non-negative integers, $\log n$ denotes the logarithm based 2 of n and for $a, b \in \mathbf{Z}$, $b > 0$, $a \bmod b$ denotes $r \in \{0, 1, \ldots, b-1\}$ such that $a - r$ is divisible by b. We use the big-O notation in the usual way. By Σ we will denote the input alphabet which will typically include symbols '0', '1' and some other punctuation symbols depending on the particular problem. Σ^* denotes the language containing all words. For $x \in \Sigma^*$, $|x|$ denotes the length of x. For a language A, $\overline{A}$ will denote the complement of A. For a class of languages $\mathcal{C}$, co$-\mathcal{C} = \{\overline{A}; A \in \mathcal{C}\}$. For a function $f : \Sigma^* \to \Sigma^*$ we say that f belongs to a complexity class $\mathcal{C}$ if the language $A_f = \{(x, i, b_i);\ x \in \Sigma^*$ and i-th bit of $f(x)$ is $b_i\}$ belongs to $\mathcal{C}$ and the length of $f(x)$ is polynomial in $|x|$. For a complexity class $\mathcal{C}$, the non-uniform version of $\mathcal{C}$ is defined to be $C/poly = \{A;\ \exists B \in \mathcal{C}$ and $\exists f : \mathbf{N} \to \Sigma^*$ such that $x \in A$ if and only if $(x, f(|x|)) \in B$ and $\exists c > 0;\ \forall n \in \mathbf{N};\ |f(n)| \leq n^c + c\}$; function f is called the *advice*.

Let $\mathcal{R}$ be a complexity class and A and B be languages. We say that A $\mathcal{R}$-*many-one reduces* to B ($A \leq_{\mathrm{m}}^{\mathcal{R}} B$) if there is a function $f \in \mathcal{R}$ such that for any $x \in \Sigma^*$, $x \in A$ if and only if $f(x) \in B$. Further, we say that B is *hard* for a class $\mathcal{C}$ under $\mathcal{R}$-many-one reductions if for any $A \in \mathcal{C}$, $A \leq_{\mathrm{m}}^{\mathcal{R}} B$. If B is hard for $\mathcal{C}$ under $\mathcal{R}$-many-one reductions and $B \in \mathcal{C}$ then we say that B is *complete* for $\mathcal{C}$ under $\mathcal{R}$-many-one reductions.

Let $t(n), s(n) : \mathbf{N} \to \mathbf{N}$ be functions. Then:

- $DSPACE(s(n))$ ($DTIME(t(n))$) denotes the class of languages accepted by deterministic Turing machines in space $O(s(n))$ (time $O(t(n))$).

- $NSPACE(s(n))$ denotes the class of languages accepted by nondeterministic Turing machines in space $O(s(n))$.

- $DTISP(t(n), s(n))$ ($NTISP(t(n), s(n))$) denotes the class of languages accepted by (non)deterministic Turing machines running simultaneously in space $O(s(n))$ and time $O(t(n))$.

The following specific classes are of interest to us.

- $L = DSPACE(\log n)$ and more generally, for $c \geq 0$, $L^c = DSPACE(\log^c n)$.

- SL denotes the class of languages accepted by symmetric Turing machines in logarithmic space. Symmetric Turing machines were defined by Lewis & Papadimitriou (1982) and they are nondeterministic Turing machines with the following property: if there is a legal move from a configuration c to a configuration d then a move from d to c is also legal. The actual definition involves certain technicalities that go beyond the scope of this work. We equivalently define SL to be the class of languages that are logarithmic space reducible to the undirected s-t-connectivity problem (see $USTCONN$ in the next section).

- $NL = NSPACE(\log n)$.

- $P = \bigcup_{c>0} DTIME(n^c)$.

- $ZPLP$ denotes the class of languages accepted by zero-error probabilistic Turing machines running in expected polynomial time and logarithmic space. Zero-error means that whenever the machine produces an output, the output is correct.

- RL denotes the class of languages accepted by one-sided error probabilistic (*randomized*) Turing machines running in polynomial time and logarithmic space. I.e., $A \in RL$ if there is a probabilistic Turing machine running in polynomial time and logarithmic space such that if $x \in A$ then $\Pr[M \text{ accepts } x] \geq 1/2$ and if $x \notin A$ then $\Pr[M \text{ accepts } x] = 0$. For a function $l(n) : \mathbf{N} \rightarrow \mathbf{N}$, $RL(l(n))$ denotes the class of languages accepted by RL machines that use $O(l(n))$ random bits.

- $2{-}RL$ denotes the class of languages accepted by one-sided error probabilistic (*randomized*) Turing machines running in polynomial time and logarithmic space that have two-way access to random bits. A $2{-}RL$ machine may examine any of its random bits multiple times in any order whereas an RL machine which has only one-way access to random bits can examine each of its random bits only once. In the further text when we talk about space-bounded probabilistic computation we refer to the probabilistic computation with one-way access to its random bits, unless otherwise is stated explicitly. Similarly to $RL(l(n))$, $2{-}RL(l(n))$ denotes the class of languages accepted by $2{-}RL$ machines that use only $O(l(n))$ random bits.

- $SC^2 = \bigcup_{c>0} DTISP(n^c, \log^2 n)$.

- $\oplus L$ denotes the class of languages A for which there is a nondeterministic Turing machine M running in logarithmic space such that $x \in A$ if and only if the number of accepting nondeterministic computations of M on input x is odd.

- UL denotes the class of languages accepted by nondeterministic Turing machines that have at most one accepting computation on any input.

Further, we will use the following notation for classes of languages computed by Boolean circuits. We will assume $DLOGTIME$ uniform circuit families. The uniformity of circuit families is discussed in depth in Ruzzo (1981) and Barrington, Immerman & Straubing (1990).

- For an integer $i \geq 0$, NC^i denotes the class of languages computed by uniform families of polynomial size Boolean circuits of depth $\log^i n$ consisting of bounded fan-in AND, OR, and NOT gates. We refer to such circuits as to uniform NC^i circuits.

- AC^0 denotes the class of languages computed by uniform families of polynomial size Boolean circuits of constant depth consisting of unbounded fan-in AND, OR, and NOT gates. We refer to such circuits as to uniform AC^0 circuits.

- TC^0 denotes the class of languages computed by uniform families of polynomial size Boolean circuits of constant depth consisting of unbounded fan-in AND, OR, NOT and MAJ gates, where MAJ computes the majority function. We refer to such circuits as to uniform TC^0 circuits.

Throughout the text we will refer to logarithmic space also as to *log-space* and to polynomial time as to *poly-time*.

The following facts are standard and straightforward.

Proposition 1

- $AC^0 \subseteq TC^0 \subseteq NC^1 \subseteq L \subseteq SL \subseteq NL \subseteq NC^2 \subseteq P.$

- $L \subseteq ZPLP = RL \cap co - RL \subseteq RL \subseteq NL.$

- $L \subseteq UL \subseteq \oplus L \subseteq P.$

None of the inclusions in the proposition is currently known to be proper except for $AC^0 \subsetneq TC^0$, Håstad (1986). Note that integer addition, subtraction, multiplication, division and computing modulus are all in TC^0, Hesse, Allender & Barrington (2002).

3.1.1 Three Variants of s-t-connectivity

In this section we define three variants of s-t-connectivity problem. Let $G(V, E)$ be a graph with the vertex set V and the set of edges $E \subseteq V \times V$. We will need to encode G into a string over a chosen alphabet Σ, where Σ contains at least two distinct symbols. There are several standard ways how to represent graph G: a) by the adjacency matrix, b) by a list of edges. The former representation can be encoded into a string of length

$O(|V|^2)$ and the latter one into a string of length $O(|E| \log |V|)$. Since each of these representations can be transformed into the other one by uniform NC^1 circuits, from our point of view these representations are equivalent. Typically we will think of G as being represented by the list of edges.

- (Directed) s-t-connectivity: $STCONN = \{< G, s, t >;\ G$ is a directed graph and there is a path from vertex s to vertex t in $G\}$.
- Undirected s-t-connectivity: $USTCONN = \{< G, s, t >;\ G$ is an undirected graph and there is a path from vertex s to vertex t in $G\}$.
- Forest s-t-connectivity: $FSTCONN = \{< G, s, t >;\ G$ is an undirected forest and there is a path from vertex s to vertex t in $G\}$.

By $\overline{STCONN}$, $\overline{USTCONN}$ and $\overline{FSTCONN}$ we will denote languages of instances of directed, undirected and forest s-t-connectivity, respectively, where s and t are not connected.

The following facts are well established:

Proposition 2

- *Savitch (1970), $STCONN$ is complete for NL under NC^1 many-one reductions.*
- *Lewis & Papadimitriou (1982), $USTCONN$ is complete for SL under NC^1 many-one reductions.*
- *Cook & McKenzie (1987), $FSTCONN$ complete for L under NC^1 many-one reductions.*

There are several other graph problems that are known to be complete or to lie within some of the log-space classes. Most notably, 2-colorability (Bi-partiteness) of Undirected Graphs is complete for SL (Reif, 1984 and Nisan & Ta-Shma, 1995), Tree Isomorphism is in L (Lindell, 1992), Planarity Testing is in SL and it is hard for L (Allender & Mahajan, 2000). Also Graph Connectivity is known to be in SL, hard for L, but not known to be hard for SL under many-one-reductions (folklore, see Wigderson, 1992). For a survey of problems that are complete for L see Cook & McKenzie (1987) and Jenner, Lange & McKenzie (1997), for SL see Àlvarez & Greenlaw (2000), and for NL see Jones, Lien & Laaser (1976).

Here we would like to point out the following simple observation.

Proposition 3 *If a language A is reducible to $FSTCONN$, $USTCONN$, $STCONN$ via an L, SL, NL many-one reduction, respectively, then A is reducible to $FSTCONN$, $USTCONN$, $STCONN$ via AC^0 many-one reduction, respectively.*

The log-space classes are of major interest because of the following translation theorem.

Proposition 4 *If $L = NL$ then $DSPACE(s(n)) = NSPACE(s(n))$, for any space constructible function $s(n) \geq \log n$.*

3.2 Complexity of Undirected s-t-connectivity

In this section we briefly survey progress on undirected s-t-connectivity from the perspective of space bounds on Turing machines. We do not include references to all the work that has been done on this topic since the volume of the work that has been done in this area is enormous. Rather we try to point out the key developments and the state-of-the-art results in the area. We would like to point the reader also to an excellent survey of Wigderson (1992). Although the survey is out-dated by now because of new results in 1990's, it captures the state of the art regarding s-t-connectivity at the beginning of 1990's and it is a worthwhile reading even today. Complementary reading would also be a survey of Saks (1996) on derandomization of space-bounded computation.

It has been well known since 1970's that s-t-connectivity can be solved on random-access Turing machines by depth-first and breadth-first search simultaneously in space (almost) linear in the number of vertices and time (almost) linear in the number of edges and vertices. The first fundamental result that relates different variants of s-t-connectivity was a result of Savitch in 1970. He made substantial progress toward answering the question of the role of nondeterminism in space-bounded computation. He has shown that $STCONN$ is a complete problem for NL and presented a recursive path-halving algorithm that solves $STCONN$ in deterministic space $O(\log^2 n)$ and time $O(n^{\log n})$.

Theorem 5 (Savitch, 1970) *$NL \subseteq L^2$.*

An immediate corollary is the next statement.

Corollary 6 *$SL \subseteq L^2$.*

Savitch's construction can be viewed as a reduction of an instance of $STCONN$ on an n vertex graph to an instance of $FSTCONN$ on a forest with $O(n^{\log n})$ vertices. There is another result with a similar flavor that is worth of mentioning. Lewis & Papadimitriou (1982) show how to reduce an instance of $STCONN$ on an n vertex graph to an instance of $USTCONN$ on graph with $O(n^{\log n})$ vertices such that if s and t are connected then they are connected by a path of length at most polynomial. In their notation the result can be stated as follows.

Theorem 7 (Lewis & Papadimitriou, 1982) *NL is in symmetric polynomial time and space $O(\log^2 n)$.*

The next major breakthrough regarding s-t-connectivity occurred almost ten years after Savitch's theorem. Aleliunas et al. (1979) present a randomized log-space algorithm for undirected s-t-connectivity. The algorithm is very simple, it performs a random walk on a graph:

On input $< G, s, t >$:

$u \leftarrow s$.

Repeat for $O(n^3)$ steps:

Choose uniformly at random a neighbor v of u.

$u \leftarrow v$.

If $u = t$ then output YES and stop.

Output NO.

End.

It can be shown that if s and t are in the same connected component then the algorithm outputs YES with probability at least 1/2. If s and t are not in the same component then the algorithm clearly outputs NO always. Hence the algorithm makes only a one-sided error:

Theorem 8 (Aleliunas et al., 1979) *$SL \subseteq RL$.*

By performing the random walk for $O(n^5 \log n)$ steps instead of $O(n^3)$ the probability of error reduces to less than $2^{-n^2 \log n}$. That means that there is a particular setting of the random bits for the algorithm that gives a correct answer on every n vertex input graph. This setting can be given to the algorithm as an advice.

Theorem 9 (Aleliunas et al., 1979) *$SL \subseteq L/poly$.*

This randomized algorithm has a very interesting consequence. At every vertex v let the input graph have edges leaving v labeled by $\{0, \ldots, \deg(v) - 1\}$. The random walk performed by the algorithm can be described by a sequence of these labels. Such a sequence is called a *traversal sequence*. What the previous argument indeed shows is that there is a traversal sequence of length $O(n^5 \log n)$ that in any labeled undirected connected graph on n vertices leads a walk in such a way that all vertices are visited regardless of the starting vertex. Such a sequence is called *universal*. Universal traversal sequences are the focus of our study and will be discussed in next chapter.

The algorithm of Aleliunas et al. searches for a witness that s and t are connected. For a long time it was not clear whether in the case that s and t are not connected one can find a witness of that fact, i.e., whether the algorithm can be made zero-error. Borodin, Cook, Dymond, Ruzzo & Tompa (1989) eventually found a log-space zero-error probabilistic algorithm for $USTCONN$. Their algorithm uses the random walk method of Aleliunas et al. to explore the graph together with a then recently introduced inductive counting method to verify that all the vertices of the connected component of s has been visited.

Theorem 10 (Borodin et al., 1989) *$SL \subseteq ZPLP$.*

The inductive counting method was discovered independently by Immerman and Szelepcsényi in 1987. They used it to settle a long standing open problem whether nondeterministic space-bounded computation is closed under complement.

Theorem 11 (Immerman, 1988, Szelepcsényi, 1988) *$NL = \text{co}{-}NL$.*

The proof of Immerman and Szelepcsényi can be viewed as an NC^1 reduction of $STCONN$ to $\overline{STCONN}$. In other words, they present an NC^1 algorithm that on input $< G, s, t >$ outputs $< H, s', t' >$ such that there is a path from s to t in G if and only if there is no path from s' to t' in H.

Except for the aforementioned result of Borodin et al., the inductive counting technique fell short of providing a similar reduction for $USTCONN$. It took several years before Nisan & Ta-Shma (1995) found such a reduction. Using sorting networks of Ajtai, Komlós & Szemerédi (1983) they reduce counting the number of connected components in an undirected graph to $USTCONN$. Comparing the number of connected components in $G(V, E)$ and $G(V, E \cup \{(s, t)\})$ then yields the desired reduction of $USTCONN$ to $\overline{USTCONN}$.

Theorem 12 (Nisan & Ta-Shma, 1995) *$SL = \text{co}{-}SL$.*

The result of Aleliunas et al. gives rise to the following conjecture that is still unresolved. Note, that in non-uniform setting this conjecture is known to be true, i.e., $SL/poly = L/poly$.

Conjecture 13 *$SL = L$.*

Since the randomized algorithm for $USTCONN$ was found, most of the progress toward proving this conjecture is closely connected to progress in derandomization and explicit constructions of universal traversal sequences. Hence, the next section is a detour into derandomization.

3.2.1 Derandomization

In the 1980's randomness has been identified as an important computational resource. The algorithm of Aleliunas et al. from the previous section is an algorithm that uses a polynomial amount of random bits. One of the major questions in complexity theory is whether randomness can be reduced or completely eliminated from computation without paying a much of penalty in other computational resources. In particular, the question is whether a randomized computation can be replaced by a deterministic computation running within the same time and space.

One way to reduce the amount of randomness used by a particular algorithm would be to construct a family of functions $g_n : \{0,1\}^{r(n)} \to \{0,1\}^{l(n)}$, where $\log n \leq r(n) << l(n) \leq \mathrm{poly}(n)$, so that there is only a negligible difference in the acceptance probability of the algorithm when run with a random string chosen at random from the uniform distribution $U_{l(n)}$ and when run with a string chosen according to distribution induced by $g_n(U_{r(n)})$. Functions with these properties are called *pseudo-random generators.* They were originally introduced for cryptographic applications by Blum & Micali (1984) and since then they found many applications also in complexity theory.

Pseudo-random generators can easily be constructed non-uniformly. In fact, for a given randomized algorithm using $l(n) = \mathrm{poly}(n)$ many random bits, for some $r(n) = O(\log n)$, a family of randomly chosen functions will be a pseudo-random generator for that algorithm with high probability. This can be easily seen using Chernoff bound. The major open question is how to construct pseudo-random generators uniformly.

If we had a randomized algorithm running in space $O(r(n))$ that uses $l(n)$ random bits and a pseudo-random generator $g_n : \{0,1\}^{r(n)} \to \{0,1\}^{l(n)}$ uniformly computable in $DSPACE(r(n))$, then we could estimate the acceptance probability of the randomized algorithm by running it on all $2^{r(n)}$ strings generated by g_n. In this way, we could get a $DSPACE(r(n))$ algorithm solving the original problem, i.e., we would successfully remove all the randomness.

In the last fifteen years there were many fundamental results in the area of construction of pseudo-random generators and derandomization. Next we will briefly survey development that is relevant to our study.

The first result on the construction of pseudo-random generators for space-bounded computation was a result of Ajtai, Komlós & Szemerédi (1987). They presented a log-space procedure that takes $r(n) = O(\log n)$ random bits and produces $l(n) = O(\log^2 n/\log\log n)$ bits that look random to any one-sided error randomized log-space algorithm M that uses $l(n)$ random bits. The produced bits look random to M in the following sense: for any input x of size n, if $\Pr[M \text{ accepts } x] \geq 1/n$ then M accepts x on at least one string produced by the procedure of Ajtai et al.. The procedure is based on random walks in expanders.

Theorem 14 (Ajtai et al., 1987) $RL(\log^2 n/\log\log n) \subseteq L$.

Using a different technique Nisan & Zuckerman (1996) substantially improved on the previous result. In log-space, they stretch $O(\log n)$ bits into $\log^{1+\gamma} n$ bits, for any $0 < \gamma < 1$, using extractors. Then, by recursive composition of this stretching they produce $l(n) = \log^c n$ pseudo-random bits out of $O(\log n)$ random bits, for any $c \geq 1$. The produced pseudo-random bits have the following property: for any log-space randomized algorithm M that uses $\text{poly}(\log n)$ random bits and any input x, the probability that M accepts x while using truly random bits differs by at most $1/2^{\log^{1-\gamma} n}$ from the probability that M accepts x while using the pseudo-random bits produced by the generator of Nisan and Zuckerman.

Theorem 15 (Nisan & Zuckerman, 1996) $RL(\text{poly}(\log n)) \subseteq L$.

Building on previous work of Impagliazzo, Nisan & Wigderson (1994), Raz & Reingold (1999) improve the result of Nisan and Zuckerman in certain settings. For a log-space randomized machine M that uses $2^{\sqrt{\log n}}$ random bits, if there is a certain very weak estimator of probabilities of reaching different configurations of M then $O(\log n)$ random bits can be stretched into $2^{\sqrt{\log n}}$ pseudo-random bits that look random to machine M. The main idea of the construction is to extract and reuse randomness that is contained in configurations of M during computation. The technique can be applied in particular to random walks on undirected graphs. Their result implies existence of a log-space algorithm that given an instance $< G, s, t >$ of undirected s-t-connectivity, where G is d-regular, if s and t are not connected then the algorithm outputs NO and if s and t are connected and the probability of reaching t from s by a random walk of length $2^{\sqrt{\log n}}$ is at least 1/2 then the algorithm outputs YES. (The result of Raz & Reingold (1999) depends on an existence of optimal extractors.)

A different line of research extended the work of Ajtai et al. (1987) in the following way. Babai, Nisan & Szegedy (1992) present a pseudo-random generator for log-space randomized machines, that stretches $r(n) = O(2^{\sqrt{\log n}})$ random bits into polynomially many pseudo-random bits. The construction relies on a proven lower bound for a multi-party communication complexity of a particular function and it fits into the general framework of *trading hardness for randomness*. The generator has the following pseudo-random property: the acceptance probability of any log-space randomized algorithm when run with truly random bits differs by at most $1/n$ from the acceptance probability of the algorithm when run with the pseudo-random bits produced by the generator.

Theorem 16 (Babai et al., 1992) $RL \subseteq 2{-}RL(2^{\sqrt{\log n}})$.

Corollary 17 $RL \subseteq DSPACE(2^{\sqrt{\log n}})$.

Note that Savitch's algorithm gives actually a better bound $RL \subseteq L^2$ however, as we shall see further the construction of the pseudo-random generator also implies existence of short universal traversal sequences.

Within a short time, result of Babai et al. was superseded by a fundamental result of Nisan (1992). He constructs a pseudo-random generator that stretches $r(n) = O(\log^2 n)$ random bits into polynomially many pseudo-random bits. This pseudo-random generator has found a huge number of applications throughout all of computer science since its introduction. The generator is based on the 2-universal hash functions of Carter & Wegman (1979). It is a recursive construction where at every level i of the recursion a new hash function is used to double the length of the output $g^{(i)}$ in the following way: $g^{(i+1)}(r) = g^{(i)}(r)g^{(i)}(h_i(r))$, where $g^{(1)}(r) = r$. The $O(\log^2 n)$ bits of the input to the generator is used to specify $O(\log n)$ random hash functions and a random string $r \in \{0,1\}^{O(\log n)}$. The generator has the same pseudo-random properties as the generator of Babai et al.

Theorem 18 (Nisan, 1992) $RL \subseteq 2{-}RL(\log^2 n)$.

Corollary 19 $RL \subseteq L^2$.

There are several results that directly build on Nisan's work. Nisan's pseudo-random generator implies the existence of universal traversal sequence of length $O(n^{\log n})$ for undirected graphs on n-vertices. This sequence has been employed in the algorithm of Nisan, Szemerédi & Wigderson (1992) to obtain a $DSPACE(\log^{3/2} n)$ algorithm for $USTCONN$. The algorithm is recursive and resembles Savitch's algorithm. At every level of recursion it explores neighborhood of size $2^{O(\sqrt{\log n})}$ of every vertex using universal traversal sequences for graphs on $2^{O(\sqrt{\log n})}$ vertices. Then it chooses a small subset of vertices to represent the graph and shrinks the graph by a factor of $2^{O(\sqrt{\log n})}$.

Theorem 20 (Nisan et al., 1992) $SL \subseteq L^{3/2}$.

Motivated by the technique of Nisan et al., Saks & Zhou (1999) improve the result of Nisan (1990) as follows.

Theorem 21 (Saks & Zhou, 1999) $RL \subseteq L^{3/2}$.

Saks and Zhou devised a way how to reuse the same hash functions multiple times while derandomizing a log-space computation using Nisan's pseudo-random generator. Their technique does not yield a new construction of pseudo-random generators, though.

Building upon ideas from the previous two results, Armoni, Ta-Shma, Wigderson & Zhou (2000) obtained the current state-of-the-art algorithm.

Theorem 22 (Armoni et al., 2000) $SL \subseteq L^{4/3}$.

There are several other results concerning $USTCONN$ that are based on derandomization and should also be mentioned.

Nisan (1994) improves his 1992 result by observing that one can deterministically choose the hash functions used in his pseudo-random generator, and he presents an algorithm that finds the right sequence of the hash functions in polynomial time.

Theorem 23 (Nisan, 1994) $RL \subseteq SC^2$.

Here, Nisan improves the bound given to $USTCONN$ by Savitch's algorithm.

Corollary 24 $SL \subseteq SC^2$.

It should be noted that algorithms of Nisan et al. (1992), Saks & Zhou (1999), and Armoni et al. (2000) do not run in polynomial time. Thus, Nisan's algorithm is the most space efficient polynomial time algorithm for undirected s-t-connectivity that is currently known. Improving on this algorithm is an open problem.

Beside the pseudo-random generators that are designed for derandomization of space-bounded computation, there is a large body of work on pseudo-random generators for time-bounded randomized computation. All the known constructions of pseudo-random generators for time-bounded computation rely on unproven complexity assumptions and they fit the framework of trading hardness for randomness. Some of these constructions also translate into the space-bounded setting. Namely, the following result is known:

Theorem 25 (Klivans & van Melkebeek, 1999) *If there is a language A computable in $DSPACE(n)$ such that for some constant $\epsilon > 0$, no family of Boolean circuits of size $2^{\epsilon n}$ can compute A correctly on infinitely many input lengths, then there is a log-space constructible pseudo-random generator $g_n : \{0,1\}^{O(\log n)} \to \{0,1\}^{n^{O(1)}}$ such that for any log-space randomized algorithm M and input x, the probability that M accepts x using truly random bits differs by at most $1/n$ from the probability that M accepts x using pseudo-random bits produced by generator g_n.*

In fact a weaker assumption is sufficient for the existence of the pseudo-random generator namely, an existence of a language $A \in DSPACE(n)$ such that for some $\epsilon > 0$ no branching program of size $2^{\epsilon n}$ computes A correctly on infinitely many input lengths. This pseudo-random generator can be used to derandomize not only log-space randomized computation with one-way access to its input bits but also log-space randomized computation with two-way access to its random bits.

Corollary 26 *If there is a language A in $DSPACE(n)$ such that for some $\epsilon > 0$, no family of Boolean circuits of size $2^{\epsilon n}$ can compute A correctly on infinitely many input lengths, then $SL = L$.*

Recently Goldreich & Wigderson (2002) published a new result regarding undirected s-t-connectivity.

Theorem 27 (Goldreich & Wigderson, 2002) *For any $\epsilon > 0$, there is a deterministic log-space algorithm that solves undirected s-t-connectivity problem correctly on all inputs of size n except for 2^{n^ϵ} instances. On these 2^{n^ϵ} instances the algorithm outputs* I DON'T KNOW.

The idea of the algorithm is to use extractors to extract randomness from the input itself to obtain a universal traversal sequence for graphs on $n^{\epsilon'}$ vertices. This universal traversal sequence is then employed in the algorithm of Nisan, Szemerédi and Wigderson. For all but 2^{n^ϵ} instances of size n, the algorithm can extract enough randomness to produce the universal traversal sequence. Currently it is an open problem to reduce the number 2^{n^ϵ} of bad instances. It seems that $2^{\mathrm{poly}(\log n)}$ should be attainable by current techniques.

All the results that we have surveyed so far can be taken as a strong evidence that $SL = L$. Proving that $SL = L$ is one of the major open problems in complexity theory.

3.2.2 Other Upper Bounds on s-t-connectivity

Beside the results that were mentioned in previous sections there are several other results regarding s-t-connectivity problem that we would like to mention here.

Karchmer & Wigderson (1993) show the following result using span programs.

Theorem 28 (Karchmer & Wigderson, 1993) *$SL \subseteq \oplus L$.*

Gál & Wigderson (1996) extend the result to NL using a different technique. They assign random weights to the edges of the input graph G. If there is a path from s to t then with high probability there is a unique minimum weight path from s to t in G.

Gál & Wigderson take advantage of this to construct a graph H out of weighted graph G so that there is path from s to t in G if and only if there is odd number of path from s to t in H. H consists of many serially interconnected copies of weighted G.

Theorem 29 (Gál & Wigderson, 1996) $NL/poly \subseteq \oplus L/poly$.

Reinhardt & Allender (2000) extend the previous result even further. They use the same idea of assigning random weights to G together with inductive counting to obtain the following result.

Theorem 30 (Reinhardt & Allender, 2000) $NL/poly \subseteq UL/poly$.

The last two theorems can be made uniform under the same assumption as in Theorem 25.

3.2.3 Other Computational Models

There is a large body of work regarding s-t-connectivity in the following areas.

Time-space trade-offs. Several algorithms for s-t-connectivity have been devised that exhibit trade-off between time and space. Contrary to previous believes (Tompa, 1982), Barnes, Buss, Ruzzo & Schieber (1998) constructed $DTISP(\mathrm{poly}(n), n/2^{\sqrt{\log n}})$ algorithm for $STCONN$. For undirected s-t-connectivity, Barnes & Ruzzo (1996) gave $DTISP(n^{1/\epsilon}, n^{\epsilon}\log^2 n)$ algorithm that was almost immediately superseded by $DTISP(\mathrm{poly}(n), \log^2 n)$ algorithm of Nisan (1994). Feige (1997) gives randomized algorithm for $USTCONN$ that achieves certain time-space trade-offs.

Jumping Automaton for Graphs. Jumping automata for graphs (JAG's) were introduced by Cook & Rackoff (1980). JAG's are pebble moving automata. Several upper and lower bounds on space as well as on time-space trade-offs required for solving directed and undirected s-t-connectivity on these machines were obtained. The tight trade-off lower bound of time $\Omega(n^{\log n})$ for space $O(n^{1-\gamma})$, $0 < \gamma < 1$, for solving directed s-t-connectivity on JAG's was given in Edmonds, Poon & Achlioptas (1999). The matching upper bound is from Barnes et al. (1998). In Edmonds et al. there is also tight space lower bound $\Omega(\log^2 n)$ for solving directed s-t-connectivity on JAG's.

Circuits. There is a large number of depth lower bounds for solving s-t-connectivity in different circuit models. Here we assume that graphs are represented by adjacency matrices. Savitch's algorithm can be implemented by monotone NC^2 circuits. (A circuit is *monotone* if it consists only of gates AND and OR.) By result of Furst, Saxe and Sipser (1984) $STCONN \notin AC^0$. Quantitatively better lower bounds were obtained by Håstad (1986), Ajtai (1988), Bellantoni, Pitassi and Urquhart (1992) and Beame, Impagliazzo & Pitassi (1998). In the monotone circuit model Karchmer & Wigderson (1990) prove a tight lower bound that $STCONN$ requires monotone NC^2 circuits.

Traversal sequences. This is the main focus of this work and will be surveyed in the next chapter.

Others There is also an extensive study of random walks on graphs (finite Markov chains), and s-t-connectivity is studied in many other theoretical ares as well, for instance, in logic.

Chapter 4

Traversal Sequences

Traversal sequences were proposed by Cook in the late 1970's to be an analog of pebble-moving automata. The idea is that a traversal sequence will guide a walk of the pebble in a graph. We present here the formal definition. Let $G(V, E)$ be a graph. For a vertex u of G, the degree of vertex u is defined to be $\deg(u) = |\{v;\ (u, v) \in E\}|$. A *labeling* l of G is a map assigning to every edge $(u, v) \in E$ a label $l(u, v) \in \{0, \ldots, \deg(u) - 1\}$, so that for every two distinct edges $(u, v), (u, w)$ incident to u, $l(u, v) \neq l(u, w)$. We are mostly concerned with undirected graphs, i.e., graphs where $(u, v) \in E$ implies $(v, u) \in E$. For simplicity, we will consider traversal sequences only for undirected graphs where all the vertices are of fixed degree d, i.e., d-regular graphs.

Let $d > 1$ be an integer, $G(V, E)$ be a d-regular graph and l be a labeling of G. Let u, v be vertices of G. We say that $i \in \{0, \ldots, d-1\}$ *takes us from vertex* u *to vertex* v, if $(u, v) \in E$ and $l(u, v) = i$. We denote this by $u \to^i v$. For $t \in \{0, \ldots, d-1\}^*$, we extend the notation recursively: if $t = \epsilon$ then $u \to^t u$, and if $t = t'i$, for $t' \in \{0, \ldots, d-1\}^*$ and $i \in \{0, \ldots, d-1\}$, then $u \to^t w$, given that $u \to^{t'} v$ and $v \to^i w$. We call sequence t a *traversal sequence*. We say that t *visits vertex* v *starting at* u if there is a prefix t' of t, such that $u \to^{t'} v$.

Let $\mathcal{G}$ be a class of connected undirected graphs. We say that a traversal sequence t is a *universal traversal sequence for* $\mathcal{G}$, if for every $G \in \mathcal{G}$ and every labeling of G, t visits every vertex in G starting at any vertex u of G.

Cook asked the question whether there are *short* traversal sequences (i.e., of length polynomial in n) that are universal for the class of undirected connected d-regular graphs on n vertices. The motivation for this question comes from the study of the relationship between deterministic and nondeterministic log-space. One way to prove

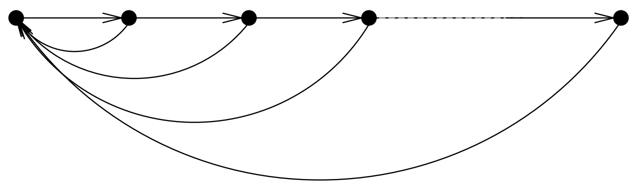

Figure 4.1: The directed graph counter-example.

that L differs from NL would be to exhibit a lower bound on the space that is needed for solving s-t-connectivity. Typically, lower bounds that are obtained are actually lower bounds for the stronger non-uniform setting. A non-uniform algorithm for s-t-connectivity would be a family of two-way finite automata $\{M_n\}_{n\in\mathbf{N}}$ such that M_n would solve s-t-connectivity on instances of size n. If the space needed to store the internal state of M_n were logarithmic, then s-t-connectivity would be solvable non-uniformly in logarithmic space, since automaton M_n could be given as advice to a log-space machine that would simulate it. If there were short universal traversal sequences then there would be a simple non-uniform algorithm for s-t-connectivity and hence, it would be impossible to obtain a non-uniform space lower bound for s-t-connectivity.

Aleliunas et al. (1979) gives a positive answer to Cook's question:

Theorem 31 (Aleliunas et al., 1979) *For any d, there are traversal sequences of length $O(d^2 n^3 \log n)$ that are universal for undirected connected d-regular graphs on n vertices.*

Corollary 32 *$SL/poly = L/poly$.*

It is rather straightforward to show that traversal sequences that are universal for directed graphs on n vertices must be of exponential length. (Consider the graph in Figure 4.1 with all possible labelings.) Thus the previous result indicates that complexity of undirected and directed s-t-connectivity may differ. The conjecture, that $SL = L$ has been formulated. In the previous section we have reviewed the development of the past twenty years that concerns this conjecture. Many of the results are closely related to the development that regards universal traversal sequences for undirected graphs which we would like to survey here.

The result of Aleliunas et al. (1979) is proven by probabilistic method. It actually shows that most of the traversal sequences of length $O(d^2n^3 \log n)$ are universal. However, it does not yield an explicit efficient deterministic construction of short universal traversal sequences. If there is a log-space algorithm that constructs universal traversal sequences for undirected 3-regular graphs then $SL = L$, since in log-space we can reduce the s-t-connectivity problem on arbitrary graphs to the s-t-connectivity problem on graphs of degree 3. It is a widely-held belief that universal traversal sequences are log-space constructible.

Conjecture 33 *There are log-space constructible universal traversal sequences for* 3*-regular graphs of polynomial length.*

This conjecture implies $SL = L$. There is a large body of work that goes toward constructing short universal traversal sequences explicitly. An implication of the result of Aleliunas et al. (1979) is that a pseudo-random generator for randomized logarithmic space yields universal traversal sequences. A universal traversal sequence is obtained by concatenating all the pseudo-random strings produced by the generator in any order, e.g., in the order in which they are produced. The first sub-exponential universal traversal sequence for 3-regular graphs obtained in this way was given by Babai et al. (1992).

Theorem 34 (Babai et al., 1992) *There exists a* $DSPACE(2^{O(\sqrt{\log n})})$ *constructible traversal sequence of length* $2^{2^{O(\sqrt{\log n})}}$ *that is universal for* 3*-regular graphs.*

The current state-of-the-art construction of universal traversal sequences for 3-regular graphs was obtained by Nisan (1992).

Theorem 35 (Nisan, 1992) *There is a* L^2 *constructible traversal sequence of length* $2^{O(\log^2 n)}$ *that is universal for* 3*-regular graphs.*

Some other pseudo-random generators for small space yield similar results, e.g., the generator of Impagliazzo et al. (1994). Conditionally, the following result was stated by Klivans & van Melkebeek (1999).

Theorem 36 (Klivans & van Melkebeek, 1999) *If there is a language A computable in $DSPACE(n)$ such that for some $\epsilon > 0$, no family of Boolean circuits of size $2^{\epsilon n}$ can compute A correctly on infinitely many input lengths, then there is a log-space constructible universal traversal sequence for 3-regular graphs.*

A more direct approach has been taken by several authors. They try to find a combinatorial construction of universal traversal sequences. So far, there have been constructed universal traversal sequences for several classes of graphs. First of these constructions were constructions of Bar-Noy, Borodin, Karchmer, Linial & Werman (1989) and Bridgland (1987). They present two different constructions of universal traversal sequences for connected undirected 2-regular graphs, *cycles*, of length $n^{O(\log n)}$. There is exactly one cycle on n vertices so the construction could be thought to be trivial however, the graph can be labeled in 2^n different ways and the universal traversal sequence has to work for all of these labelings.

Theorem 37 (Bar-Noy et al., 1989, Bridgland, 1987) *There is an explicit construction of a traversal sequence of length $n^{O(\log n)}$ that is universal for cycles on n vertices.*

Istrail (1988) subsequently improves on these results and presents the first log-space constructible[1] universal traversal sequence for cycles.

Theorem 38 (Istrail, 1988) *There is a log-space constructible traversal sequence of length $O(n^{4.76})$ that is universal for cycles on n vertices.*

In this work we present a new simple construction of universal traversal sequences for cycles that improves the result of Istrail.

Theorem 39 *There is a log-space constructible traversal sequence of length $O(n^{4.03})$ that is universal for cycles on n vertices.*

[1]Feigenbaum and Reingold [J. Feigenbaum, N. Reingold, *Universal Traversal Sequences*, American Mathematical Monthly, 101 (1994), pp. 262-265] claim without any further argument that Istrail's construction is not log-space constructible. However, it seems to us to be possible to show the log-space constructibility of Istrail's construction by an argument similar to one in Section 4.3.3.

Our sequence is even constructible by TC^0 circuits. The construction is presented in the subsequent sections.

Beside constructions for cycles, Karloff, Paturi & Simon (1988) present an explicit construction of a universal traversal sequence for $(n-1)$-regular graphs, *cliques*. Again, there is exactly one clique on n vertices so the challenge is to let the traversal sequence work on all $(n-1)!^n$ labelings.

Theorem 40 (Karloff et al., 1988) *There is an explicit construction of a traversal sequence of length $n^{O(\log n)}$ that is universal for the $(n-1)$-regular graph on n vertices.*

Hoory & Wigderson (1993) observed that the construction of Karloff et al. (1988) can be extended to a universal traversal sequence for expander graphs that are labeled *consistently*. A labeling l is *consistent* if for every two distinct edges $(u, v), (u, w)$, $l(v, u) \neq l(w, u)$.

Theorem 41 (Hoory & Wigderson, 1993) *For any integer d, there is an explicit construction of traversal sequence of length $n^{O(d \log d)}$ that is universal for the class of d-regular expander graphs on n vertices.*

The constant in the exponent of $n^{O(d \log d)}$ depends on the expansion factor of the expander.

Beside the explicit constructions several authors improve the non-constructive upper bound on the length of universal traversal sequences that is given in Aleliunas et al. (1979). These improvements are based on better analysis of the cover time of a graph by a random walk. Here is a summary of the current state-of-the-art non-constructive upper bounds on the length of universal traversal sequences for d-regular graphs:

- $d = 2$, the upper bound $O(n^3)$ given by Aleliunas (1978),
- $3 \leq d \leq n/2 - 1$, the upper bound $O(dn^3 \log n)$ given by Kahn, Linial, Nisan & Saks (1989),
- $n/2 - 1 < d$, the upper bound $O(n^3 \log n)$ given by Chandra, Raghavan, Ruzzo, Smolensky & Tiwari (1997).

Motivated by proving time-space trade-offs for undirected s-t-connectivity several authors also studied lower bounds on the length of universal traversal sequences for d-regular graphs. These lower bounds appear in Bar-Noy et al. (1989), Alon, Azar & Ravid (1990), Borodin, Ruzzo & Tompa (1992), Tompa (1992), Buss & Tompa (1995) and Dai (2001). Here is a summary of the state-of-the-art lower bounds of the length of universal traversal sequences for d-regular graphs:

- $d = 2$, the lower bound of $\Omega(n^{1.51})$ given by Dai (2001),
- $3 \leq d \leq n/14 + 1$, the lower bound of $\Omega(d^{0.49}n^{2.51})$ given by Dai (2001),
- $n/14 + 1 < d \leq n/3 - 2$, the lower bound of $\Omega(d^2 n^2)$ given by Borodin et al. (1992),
- $n/3 - 2 < d$, the lower bound of $\Omega(n^2)$ given by Alon et al. (1990).

4.1 Universal Traversal Sequence for Cycles

The rest of the chapter is devoted to the proof of Theorem 39. We present here an explicit construction of universal traversal sequence for cycles. Our construction has two parts:

1. The first part is a construction of an $n^{O(\log n)}$ universal traversal sequence for cycles of length n. This part uses essentially the same idea as the construction of Bridgland (1987). It is a recursive construction in which at every iteration a universal traversal sequence for cycles of length cn, $c \geq 2$, is constructed from a universal traversal sequence for cycles of length n. Every iteration consists of two dual stages. The depth of the recursion is $O(\log n)$ and the factor by which the generated universal traversal sequence is expanded at each stage is $O(n)$.

2. The second part of the construction involves again two dual stages that are aimed at reducing the expansion factor to $O(1)$, and they interleave the stages of the first part.

At both stages of the first part the traversed graph is reduced to a smaller one, whereas at both stages of the second part the graph is expanded a little bit by inserting new vertices. After applying all four stages the graph is smaller by at least a constant factor.

The main contribution of our work lies in the construction of the second part where we show that we can modify the traversed graph by inserting new vertices so that this modification is transparent for traversal sequences. This idea could be possibly useful in construction of universal traversal sequences for 3-regular graphs.

The previous construction by Istrail was also based on Bridgland's construction. Istrail's construction also reduced the expansion factor to a constant as our does but using slightly different (ad hoc) approach. Our construction seems more natural and better motivated hence, much simpler to understand.

The presentation is organized as follows. Section 4.1.1 contains preliminaries. Section 4.2 contains the first part of the construction yielding an $n^{O(\log n)}$ universal traversal sequence. Section 4.3 contains the second part of the construction including the algorithm. Section 4.4 contains the analysis of the length of the universal traversal sequence that is produced.

4.1.1 Preliminaries

Let $G(V, E)$ be an undirected cycle. Let $V = \{v_1, \ldots, v_n\}$ so that $(v_n, v_1), (v_i, v_{i+1}) \in E$, for $1 \leq i < n$. For $1 \leq i < n$, edges (v_i, v_{i+1}) and (v_n, v_1) are called the *right edges*, and (v_{i+1}, v_i) and (v_1, v_n) are called the *left edges*. Similarly, v_{i+1} is called the *right neighbor* of v_i, and v_i is the *left neighbor* of v_{i+1}. Let $l : E \to \{0, 1\}$ be a labeling of G. We label every vertex v_i in G according to the label of its right outgoing edge $(v_i, v_{(i \bmod n)+1})$ (Figure 4.2). Any sequence from $\{0, 1\}^*$ is called a *0-1-sequence.*

A 0-1 sequence $b(v_1), b(v_2), \ldots, b(v_n)$ uniquely describes G, where $b(v_i)$ is the label of vertex v_i. We call such a sequence a *graph sequence* corresponding to G. Two graph sequences are considered to be the same if one of them is a cyclic shift of the other one. Hence, we consider any graph sequence to be wrapped around (cyclic). We consider all the graphs with the same graph sequence to be the same. We identify every graph sequence with its corresponding graph (actually a class of the same graphs). We refer to the digits of a graph sequence as *vertices*. (We use term vertex also for digits of any

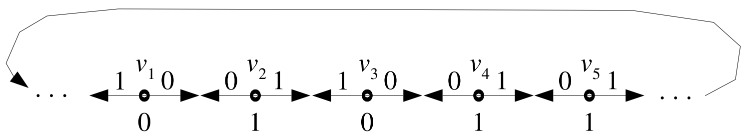

Figure 4.2: A labeled cycle.

0-1 sequence.)

A *traversal sequence* is a 0-1 sequence. "*To run a traversal sequence t on a graph G starting at a vertex v*" means to walk in G starting at v and following edges labeled consistently with t. For example, if we run a sequence 10100 on G starting at v_2 (Figure 4.2), we visit vertices $v_2, v_3, v_4, v_5, v_4, v_3$ in this order. Note, if the label of a vertex in the traversal sequence matches the label of the traversed vertex then we move to the right neighbor, otherwise to the left one. Because we identify a graph with its graph sequence we may also talk about running a traversal sequence on a 0-1 sequence. It should be always clear from the context which sequence is a traversal sequence and which one is a graph sequence.

Note, in traversing a graph sequence by a traversal sequence there are two kinds of symmetry. If we negate both, the traversal sequence and the graph sequence, we will run exactly as before. If we reverse the graph sequence (the first vertex becomes the last, and so on) and then we negate it, any traversal sequence will run exactly in opposite direction than before. (This corresponds to changing left-right orientation of the underlying graph.)

Let s be a 0-1 sequence. A 0-*run* (1-*run*) in s is any maximal part of s consisting only of zeroes (ones). The type of the run, 0-run or 1-run, is called a *color* of that run. The leftmost and rightmost vertices of the run are called *border vertices*. The border vertices of a run of length one coincide. A 0-1 sequence consisting only of runs of length i and j is (i, j)-sequence e.g., 000 111 000 111111 000 is a $(3, 6)$-sequence. A sequence consisting of a single run is called *monochromatic*.

A 0-1 sequence t *completely traverses* s starting at vertex v if during traversal of s by t starting at v we visit all the vertices of s. The 0-1 sequence t *strongly traverses*

s starting at v if it completely traverses s starting at v and for every vertex there is a visit to that vertex which comes from left and continues to the right, or it comes from right and continues to the left, i.e., the vertex is entered using one edge and left using the other one. (In the singular case of a one vertex cycle s, t strongly traverses s if t contains at least two consecutive vertices of the same color.) A 0-1 sequence t is a *(strong) universal traversal sequence* (UTS) for cycles of length n if for any cycle G of length n, t completely (strongly) traverses G starting at any vertex.

Proposition 42 *Let $n \geq 2$ be an integer.*

1. *Any (strong) universal traversal sequence for cycles of length $n+1$ is a (strong) universal traversal sequence for cycles of length n.*

2. *Any universal traversal sequence for cycles of length $n+2$ is a strong universal traversal sequence for cycles of length n.*

Proof. We leave the proof of the first claim as an easy exercise and we prove the second one. Assume that for some UTS t for cycles of length $n+2$ there is a graph G on n vertices and a vertex v in G such that G is not strongly traversed by t starting at v. By the first part of the proposition, G is traversed completely by t starting at v. Thus, there is a vertex u in G which is not strongly traversed i.e., it is always visited in the way that it is left using the same edge by which it was entered. Replace u in graph G by three interconnected vertices labeled identically to u. If $u \neq v$ then clearly the middle inserted vertex is never visited by t starting at v. If $u = v$ then the middle inserted vertex is never visited by t if t is started at either the rightmost or the leftmost inserted vertex, depending whether the first digit of t has the same color as u. Both cases give a contradiction. □

4.2 An $n^{O(\log n)}$ Universal Traversal Sequence

4.2.1 A Parity Contraction

Consider running a 1-run t of length l on a 1-run r_1 of length $n \leq l$ starting at a left border vertex of r_1. Assume that r_1 is followed by another 0-run r_0 (111...111000...000).

It is easy to see that if the parity of l and n are the same then we end up at the left border vertex of r_0, otherwise we end up at the right border vertex of r_1. The first $n-1$ digits of t bring us to the right border vertex of r_1 and the remaining digits alternate us between the right border vertex of r_1 and the first vertex of r_0. By symmetry, we get symmetric behavior for running a 0-run on a 0-run starting at the left border vertex, and for running a 1-run on a 0-run and a 0-run on a 1-run starting at the right border vertices. Hence, if the parity of the length of the traversed run corresponds to the parity of the length of the traversal run we end up at the border vertex of the next run, otherwise we end up at the opposite border vertex of the same run.

This motivates the following definition. Let s be a non-monochromatic graph sequence. The *parity contraction* $s_\oplus$ of s is a (1,2)-sequence obtained from s by replacing every run of even length by a run of length two of the same color, and by replacing every run of odd length by a run of length one of the same color. (Keep in mind that sequence s is cyclic.) For example, $\ldots 111\ 000\ 11\ 0000\ 1111\ 0\ 11\ 0 \ldots$ is parity contracted to $\ldots 1\ 0\ 11\ 00\ 11\ 0\ 11\ 0 \ldots$.

More formally, let s be the same as $r_1 r_2 \cdots r_k$, where every r_i is a monochromatic run and for every $1 \leq i < k$, r_i has an opposite color than r_{i+1}, and r_1 and r_k have different colors. For every $1 \leq i \leq k$, let r'_i be a run of the same color as r_i, and let the length of r'_i be one if r_i is of odd length and two otherwise. Then the parity contraction of s is a sequence $r'_1 r'_2 \cdots r'_k$.

There is a natural mapping of border vertices of original runs to border vertices of reduced runs. In the case of odd length runs, both border vertices are mapped to the same vertex.

Observe that for traversing a parity contraction it suffices to consider (1,2)-sequences.

Proposition 43 *Let s be a non-monochromatic graph sequence of length l, and let $s_\oplus$ be its parity contraction. Let v be a border vertex in s and $v_\oplus$ the corresponding border vertex in $s_\oplus$. If a traversal (1,2)-sequence $t_\oplus$ strongly traverses $s_\oplus$ starting at $v_\oplus$ then t, obtained from $t_\oplus$ by inflating every run by $l + (l \bmod 2)$ vertices, strongly traverses s starting at v.*

Proof. First observe that t visits all runs in s in the same order as $t_\oplus$ in $s_\oplus$. This is because after every run in t and $t_\oplus$, respectively, we are in s and $s_\oplus$, respectively, at the corresponding border vertices. The runs in $s_\oplus$ were obtained by removing even number of vertices from runs in s, and the runs in $t_\oplus$ could be obtained by removing even number of vertices from runs in t. The number of vertices in every run of t guarantees that every run in s is traversed completely starting at a border vertex and going in direction of the run. Because $t_\oplus$ strongly traverses $s_\oplus$, during traversal of s by t we enter at least one border vertex of each run in s and continue to traverse the whole run. During that traversal we strongly traverse all the vertices of that run. □

4.2.2 A Pair Contraction

Let s be a (1,2)-sequence. We call a 0-run (1-run) of length two in s a 00-*pair* (11-*pair*). A 0-run and a 1-run of length one are *singletons* and any maximal part of s consisting of singletons is 01-*run*. (A 01-run does not necessary start by 0 and end by 1.) Let us consider a graph sequence s containing a 01-run followed by a 11-pair and another 01-run: 01010 11 0101010101 ..., and let us consider a traversal sequence $t =$ 01010101010 11 01010101.... Starting t at the first vertex of s, we reach the left vertex v_L of 11-pair after 5 digits of t. The next 1 in t brings us to the right vertex v_R of the 11-pair in s, the following 0 in t takes us back to the vertex v_L, and so on. When 11-pair in t comes we get to the vertex following the 11-pair in s and we continue to run to the right.

If we would use t containing a 00-pair instead of the 11-pair, $t =$ 0101010101 00 101010101..., again we would first reach the 11-pair in s, but then we would bounce on that 11-pair using the 00-pair in t, and we would go back to the left. By symmetry, the two cases of running t from the right have a similar behavior.

Thus, if s is a graph (1,2)-sequence and we traverse s by a (1,2)-sequence t then whenever we reach a pair in s during that traversal we stay at that pair until a pair appears in t and if the color of the pairs in t and s is the same then after that pair in t, the traversal goes to the right neighbor of the pair in s otherwise to the left one. Even if several pairs are next to each other in s not separated by any 01-run, this behavior of the traversal can be observed on each of these pairs.

This motivates the following construction. Let s be a (1,2)-sequence containing at least one pair. The *pair contraction* s_c of s is a 0-1 sequence obtained from s by first removing all singletons, and then replacing every pair by one vertex of the same color e.g., 01010 11 010 11 01 00 10... is contracted to 110.... Note, the length of s_c is at most half of the length of s.

Proposition 44 *Let s_c be a pair contraction of a (1,2)-sequence s of length l. Let t be a traversal sequence obtained from a strong traversal sequence t_c of s_c starting at v_c by replacing every* 1 *by* $011(01)^{\lceil l/2 \rceil}$, *and every* 0 *by* $001(01)^{\lceil l/2 \rceil}$. *Then t is a strong traversal sequence for s starting at the left vertex of a pair corresponding to v_c, if v_c is colored* 0, *and at the right vertex of that pair otherwise.*

Proof. Because any 01-run in s must be shorter than l, the 01-runs between consecutive pairs in t always bring us from one pair in s to another. It is easy to see that t visits pairs in s in the same order as t_c visits corresponding vertices in s_c. Thus, all pairs in s get visited by t started at the vertex specified in the proposition. Because t_c strongly traverses s_c, every 01-run in s must be entirely traversed either from left or from right (strong traversal implies traversal of all edges in s_c). Thus t is a complete traversal sequence of s starting at the appropriate vertex. Observe that it is actually a strong traversal sequence. □

4.2.3 Putting Them Together

Propositions 43 and 44 give us a way to construct a universal traversal sequence of length $n^{O(\log n)}$. Having constructed a strong universal traversal sequence t_n for cycles of length n, we may construct a strong UTS t_{2n} for cycles of length $2n$ using the construction of Proposition 44 and then construction of Proposition 43 as follows.

We first replace every 1 in t_n by $011(01)^n$ and every 0 in t_n by $001(01)^n$ as in Proposition 44 to get t_{2n}^{-3}. Then we prepend $(01)^n$ in front of t_{2n}^{-3} to get t_{2n}^{-2}, which is a (1,2)-sequence. Now, we expand every run in t_{2n}^{-2} by $2n = 2n + (2n \bmod 2)$ vertices as in Proposition 43 to get t_{2n}^{-1}. Finally, we prepend 1^{2n} in front of t_{2n}^{-1} to get t_{2n}. We call the prepended sequences $(01)^n$ and 1^{2n} *loaders*.

We claim that t_{2n} is a strong UTS for cycles of length $2n$. Let s be a graph sequence of length $2n$, let $s_\oplus$ be its parity contraction (assuming s is not monochromatic), and let s_c be the pair contraction of $s_\oplus$ (assuming $s_\oplus$ contains at least one pair). Clearly, $|s_c| \leq n$, so t_n strongly traverses s_c starting at any vertex.

Observe that t_{2n}^{-2} strongly traverses $s_\oplus$ starting at any vertex. That is because during traversal of $s_\oplus$ by t_{2n}^{-2}, loader $(01)^n$ brings us to the pair "closest" to the starting vertex in $s_\oplus$, and then by Proposition 44 applied on $s_\oplus$, s_c, t_{2n}^{-3}, and t_n, sequence $s_\oplus$ is strongly traversed by t_{2n}^{-3}. (If there is no pair in $s_\oplus$ then the loader of t_{2n}^{-2} strongly traverses $s_\oplus$ by itself.)

Similarly, t_{2n} is a strong traversal sequence for s starting at any vertex. During a traversal of s by t_{2n} starting at any vertex loader 1^{2n} brings us to a border vertex "closest" to the starting vertex in s (or strongly traverses s), and then by Proposition 43 applied on s, $s_\oplus$, t_{2n}^{-1} and t_{2n}^{-2}, sequence s is strongly traversed by t_{2n}^{-1}. Hence, t_{2n} is a strong UTS for cycles of length $2n$.

We need a strong UTS for cycles of length (let us say) 5 to start this recursive construction of UTS's. Such a sequence can be obtained from any UTS for cycles of length 7 by Proposition 42.

4.3 An $O(n^c)$ Universal Traversal Sequence

A disadvantage of the previous construction is the super-polynomial length of the produced UTS. The reason for this length is a pessimism of the construction in two directions. The construction is pessimistic about the length of runs at each step, and about the number of rounds the construction has to be repeated.

A possible way how to reduce the overall length of the resulting UTS is to protect us against appearance of long runs. The way how we achieve this is by introducing two new stages in the construction which break down long runs. One of these stages takes place before parity contraction and the other one before pair contraction. The former stage reduces the length of 0-runs and 1-runs to a constant length, the latter one reduces the length of 01-runs to a constant length. A consequence of that run length reduction is an expansion of a traversal sequence in Propositions 43 and 44 just by a constant factor.

Both run breaking stages split long runs by inserting extra vertices in them. This actually increases the length of the graph sequence but in the consecutive contraction stages these increases are eliminated. The insertion is done in a way which ensures that any generated traversal sequence behaves on the new graph sequence and on the old one in the same way, i.e., the insertion seems to be transparent for the traversal sequence.

4.3.1 1-run and 0-run Breaking

This stage is aimed at breaking down long 0-runs and 1-runs.

Let us first explicitly define the property of sequences which we implicitly used in Propositions 43 and 44. Let $\mathcal{T}$ be a class of 0-1 sequences, and let $\sqsubseteq$ be a transitive and reflexive binary relation on $\mathcal{T}$. We say that $\sqsubseteq$ is a *prefix relation* if $\forall x, y \in \mathcal{T}$, $x \sqsubseteq y$ implies x is a prefix of y. E.g., let $\mathcal{T}_{3,6}$ denote the class of all (3,6)-sequences. We define a prefix relation $\sqsubseteq$ on $\mathcal{T}_{3,6}$ in the following way: $\forall x, y \in \mathcal{T}_{3,6}$, $x \sqsubseteq y$ if and only if x is a prefix of y which can be obtained from y by removing several (or possibly none) of the last runs in y. (Hence, $000 \sqsubseteq 000\ 111111$, but $000\ 111 \not\sqsubseteq 000\ 111111$.) For a prefix relation $\sqsubseteq$ we naturally define a relation $\sqsubset$ by $(x \sqsubset y \Leftrightarrow x \sqsubseteq y\ \&\ x \neq y)$.

Definition 45 *Let s and s' be graph sequences. Let sequences r and r' be parts of s and s', respectively. Let v and v' be vertices in r and r', respectively. Let v_{l} and v_{r} (v'_{l} and v'_{r}) be vertices in s (s'). (Vertices v and v' are called input vertices and the other vertices are called output vertices. Think about them as in Figure 4.3.)*

Let $\mathcal{T}$ be a class of traversal sequences, and $\sqsubseteq_{\mathcal{T}}$ be a prefix relation on $\mathcal{T}$. For every $t \in \mathcal{T}$, denote a vertex reached by running t on s (s') starting at v (v') by v_t (v'_t).

We say that $t \in \mathcal{T}$ behaves the same *on r* and *r' with respect to input and output vertices if:*

1. *$\forall t' \sqsubseteq_{\mathcal{T}} t$, $v_{t'}$ is in r and $v'_{t'}$ is in r', or*

2. $\exists t_0 \sqsubseteq_{\mathcal{T}} t$ *such that* $(v_{t_0} = v_{\mathrm{l}}$ *and* $v'_{t_0} = v'_{\mathrm{l}})$ *or* $(v_{t_0} = v_{\mathrm{r}}$ *and* $v'_{t_0} = v'_{\mathrm{r}})$, *and* $\forall t' \sqsubset_{\mathcal{T}} t_0$, $v_{t'}$ *is in* r *and* $v'_{t'}$ *is in* r'.

The rough meaning of this definition is that t behaves the same on r and r' if t keeps us in r as long as it keeps us in r' and then it leaves r and r' at the same time on corresponding vertices. As soon as t brings us outside of r and r' we do not worry about it anymore. In that, we are not looking at positions in s and s' after every single digit of t but rather after a bigger units given by $\sqsubseteq$.

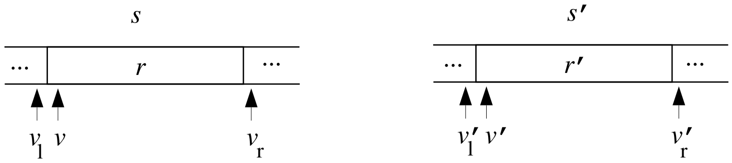

Figure 4.3: Input and output vertices.

The following proposition describes behavior of (3,6)-sequences on short runs. The proof of that proposition is just an easy case by case analysis, and we leave it as an exercise.

Proposition 46 1. *Any (3,6)-sequence started at left border vertices or right border vertices behaves the same on runs* 11, 1111 *and* 111111, *where the output vertices are vertices of the other color that are adjacent to these runs in graph sequences that contain these runs. The same holds for* 00, 0000 *and* 000000.

2. *Any (3,6)-sequence started at left border vertices or right border vertices behaves the same on runs* 1 *and* 111, *where output vertices are as above. The same holds for* 0 *and* 000.

We are going to use the following simple proposition for m equal 3 and 5.

Proposition 47 *Let* $m \geq 3$ *be an odd integer. Then for any* $l > m$ *there is a* k *and even* d *such that* $2 \leq d \leq 2m$ *and* $l = mk + d$.

Let s be a graph sequence. We replace every run of length five and every run of length longer than six in s by another sequence to obtain a graph s' with all runs of short length. The replacement is done in the following way. Let r be a 0-run or 1-run of length $l = 3k+d \in \{5, 7, 8, 9, \ldots\}$ in s, where k and d are from the previous proposition. If r is a 0-run then we replace it by $r' = (000\ 01)^k 0^d$ otherwise r is a 1-run and we replace it by $r' = (111\ 10)^k 1^d$. We call this operation *0-run and 1-run breaking* (or simply *1-run breaking*). We refer to r as to the *original* sequence and to r' as to the *stuffed* sequence.

We want to show that any (3,6)-sequence that (strongly) completely traverses s starting at a border vertex of some run (strongly) completely traverses also s' starting at the corresponding border vertex. To show that we will show that any (3,6)-sequence traverses s and s' in a similar way. In particular, we will show that any (3,6)-sequence enters any run in s at the same time as it enters the corresponding stuffed sequence in s', and also it leaves them at the same time at corresponding locations. We will use the following lemma to show that.

Lemma 48 *Let s' be obtained from a non-monochromatic graph sequence s by 1-run breaking. Let r be a run in s and r' its corresponding (stuffed) sequence in s'. Let v be the leftmost vertex of r, and let v' be the leftmost vertex of r'. Let v_{l} (v'_{l}) be the left neighboring vertex of v (v'), and let v_{r} (v'_{r}) be the vertex to the right of the rightmost vertex of r (r') (Figure 4.4). Then every $t \in \mathcal{T}_{3,6}$ behaves the same on r and r' with respect to vertices $v, v_{\mathrm{l}}, v_{\mathrm{r}}$ and $v', v'_{\mathrm{l}}, v'_{\mathrm{r}}$.*

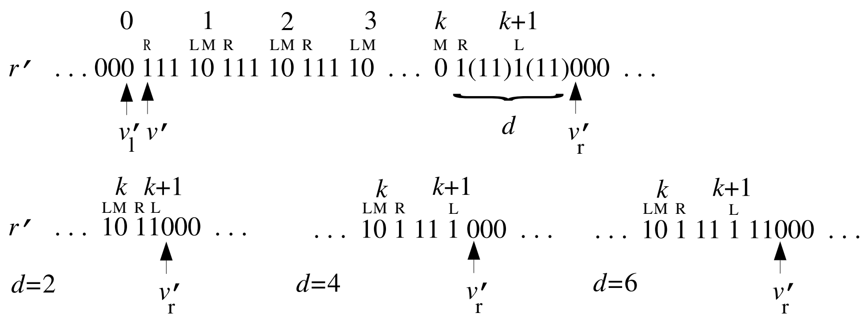

Figure 4.4: 1-run breaking.

Notice that in the case of graph s containing only two runs v_{l} may actually coincide with v_{r}. To establish this lemma we will show a correspondence between traversals of sequences r and r' by $t \in \mathcal{T}_{3,6}$.

Proof. Let the sequences and vertices be as in the statement of the lemma and let $l = |r|$. If $l \leq 4$ or $l = 6$ then $r = r'$ and the lemma is trivial. So let us assume $l = 5$ or $l \geq 7$ and assume r is a 1-run (the case of a 0-run would be similar, just interchange 0's and 1's in the proof). We are going to number vertices in r from left to right by $\{0, \ldots, l-1\}$ and to refer to vertices in r' as in Figure 4.4, i.e., there are vertices labeled $0_{\mathrm{R}}, 1_{\mathrm{L}}, 1_{\mathrm{M}}, 1_{\mathrm{R}}, 2_{\mathrm{L}}, \ldots, k_{\mathrm{M}}, k_{\mathrm{R}}, (k+1)_{\mathrm{L}}$ in r'. (Figure 4.4 explicitly shows the right end of r' for $d = 2, 4, 6$.)

If the first run of t is a 0-run then after running this run on r and r' we either end up at v_{L} and v'_{L}, respectively, or back at v and v', respectively. In the former case the lemma is proven for t. Thus, for the rest of the proof assume that t starts with 1-run (which actually will always be the case in application of this lemma).

For any $\tau \in \mathcal{T}_{3,6}$, define $o_0^\tau, o_1^\tau, e_0^\tau, e_1^\tau$ to be the number of odd 0-runs, odd 1-runs, even 0-runs, and even 1-runs in τ, respectively. Further, define $p(\tau) = (o_1^\tau + 2e_1^\tau) - (o_0^\tau + 2e_0^\tau)$. (Note, $3p(\tau)$ is the difference between the number of ones and zeroes in τ.) Further, let $[\tau]$ denote the last run in τ, let $\tau - 1$ denote a sequence obtained from τ by removing $[\tau]$, and let $l(t)$ denote the number of runs in τ.

Lemma 48 is a consequence of the following lemmas which capture a correspondence between traversal of r and r' by t. These lemmas essentially say that during the traversal of r and r', we are at vertex $3i$ in r if and only if we are at vertex i_{L}, i_{R}, or $(i \pm 1)_{\mathrm{M}}$ in r'.

Lemma 49 *Let $t \in \mathcal{T}_{3,6}$ be such that $t = \epsilon$ or t starts with a 1-run. If $\forall t' \sqsubseteq t;\ 0 \leq p(t') \leq k+1$ then the following hold:*

1. *$\forall t' \sqsubseteq t$, if $p(t') = 0$ then $[t']$ is a 0-run, and if $p(t') = k+1$ then $[t']$ is a 1-run,*
2. *if $d = 4, 6$ then we are at a vertex numbered $3p(t)$ in r after running t on r starting at v,*
3. *if $d = 2$ we are at vertex $3p(t) - \delta(t)$ in r after running t on r starting at v, where*

$\delta(t) \in \{0, 2\}$, *and* $\delta(t) = 2$ *iff* $(\exists t' \sqsubseteq t;\ p(t') = k+1$ *and* $\forall t''$ *such that* $t' \sqsubseteq t'' \sqsubseteq t$, $p(t'') > 0)$.

Proof. Let us prove the first part. Clearly, $\forall \epsilon \neq t' \sqsubseteq t$, $p(t') \neq p(t'-1)$. By assumption, $\forall t' \sqsubseteq t$, $0 \leq p(t') \leq k+1$, so $p(t') = 0$ implies $p(t'-1) > 0$, hence $[t']$ has to be a 0-run. Similarly, $p(t') = k+1$ implies $p(t'-1) < k+1$, so $[t']$ has to be a 1-run.

For the second part, if $d = 4, 6$ then $3(k+1) \leq l-1$, and given that $0 \leq p(t') \leq k+1$, for all $t' \sqsubseteq t$, we clearly are at vertex numbered $3p(t)$ in r after running t starting at v (v is numbered by 0).

Let us consider the last part i.e., $d = 2$. We prove the claim by induction on the number $c(t)$ of $t' \sqsubseteq t$ such that $p(t') = 0$ or $k+1$. For $c(t) = 1$ the claim is obvious because $p(\epsilon) = 0$, so $\forall \epsilon \neq t' \sqsubseteq t$, $0 < p(t') < k+1$ and $\delta(t) = 0$, thus $0 < 3p(t') - \delta(t') < l-1$.

For $m > 1$, assume that the claim holds for any $t \in \mathcal{T}_{3,6}$ with $c(t) < m$, and let us consider $t \in \mathcal{T}_{3,6}$ such that $c(t) = m$. There is a $t' \sqsubseteq t$ such that $c(t'-1) = m-1$ and $c(t') = m$. Clearly, $p(t') = 0$ or $k+1$. Denote $t'-1$ by t''.

Consider the case where $p(t') = k+1$. Note, $\delta(t') = 2$. By the above argument $p(t'') < k+1$, and also $p(t'') \geq k-1$, hence $p(t'')$ is $k-1$ or k, depending on whether $[t']$ are 6 ones or 3 ones. If $\delta(t'') = 2$ then we are at vertex $3(k-1) - 2$ (if $[t'] = 1^6$) or $3k-2$ (if $[t'] = 1^3$) in r after running t'' starting at v, and we are at vertex $3(k+1)-2 = 3p(t') - \delta(t')$ after running the following $[t']$.

If $\delta(t'') = 0$ then we are at vertex $3(k-1)$ or $3k$, resp., in r after running t'' starting at v. We are going to run $[t']$, i.e., 6 or 3 ones, resp., from that vertex. In the case of the last run of t' being 6 ones, after the first five out of them we reach a vertex v_{r}, and by the last one we return to vertex $l-1$ of r. A similar thing happens for $[t']$ being 3 ones. Hence, after running t' on r starting at v we are at vertex numbered $l-1 = 3p(t') - \delta(t')$ in r.

For every $t_i \in T$ such that $t' \sqsubset t_i \sqsubseteq t$, $0 < p(t_i) < k+1$, hence $0 < 3p(t_i) - \delta(t_i) < l-1$, and we are at vertex $3p(t_i) - \delta(t_i)$ after running t_i starting at v. This is true in particular for $t_i = t$.

Now consider $p(t') = 0$. Note, $\delta(t') = 0$. By similar argument as the previous one, after running t'' starting at v we are at vertex $3 \cdot 1 - \delta(t'')$ (if $[t'] = 0^3$) or $3 \cdot 2 - \delta(t'')$ (if $[t'] = 0^6$). If $\delta(t'') = 0$ then after running $[t']$ from the vertex to which we get by t'' we are at vertex numbered 0 in r. If $\delta(t'') = 2$ then after running t'' we are at vertex 1 or 4, resp. By running $[t']$ and bouncing on vertex v_{l}, we get to vertex numbered 0 in r. Note, $3p(t') - \delta(t') = 0$. By the same argument as before, any $t' \sqsubset t_i \sqsubseteq t$ brings us to vertex numbered $3p(t_i) - \delta(t_i)$ in r. In particular, t brings us to vertex $3p(t) - \delta(t)$ starting at v. $\square$

By the definition of to behave the same, to establish Lemma 48 we just need to consider t such that, for all $t' \sqsubset t$, after running t' on r and r', respectively, we are at vertices inside of r and r', respectively.

Lemma 50 *Let $\epsilon \neq t \in \mathcal{T}_{3,6}$ start with a 1-run. The following are equivalent:*

1. *$\forall t' \sqsubset t$, we are at a vertex belonging to r after running t' on r starting at v, and we are at a vertex outside of r after running t on r,*

2. *$\forall t' \sqsubset t$; $0 \leq p(t') \leq k+1$ and $((p(t-1) = 1$ and $[t]$ is an even 0-run), or $(p(t-1) = k$ and $[t]$ is an even 1-run.))*

Proof. By the previous lemma, (2) $\Rightarrow$ (1). Let us prove (1) $\Rightarrow$ (2). If t brings us outside of r then, by the previous lemma, $\exists t_m \sqsubseteq t$ such that $p(t_m) < 0$ or $p(t_m) > k+1$. Take such shortest t_m, i.e., t_m such that $\forall t' \sqsubset t_m$; $0 \leq p(t') \leq k+1$. The previous lemma applies for $t_m - 1$.

If $p(t_m) < 0$ then $p(t_m - 1) \leq 1$ and $[t_m]$ has to be a 0-run, hence $[t_m - 1]$ is a 1-run. Thus $p(t_m - 1) \neq 0$ (by the previous lemma), so $p(t_m - 1) = 1$ and $[t_m]$ is an even 0-run. By the previous lemma, t_m brings us to vertex 3 or $3 - \delta(t_m)$. In both cases t_m brings us to vertex v_{l}.

If $p(t_m) > k+1$ then $[t_m]$ is a 1-run, $[t_m - 1]$ is an 0-run, thus $p(t_m - 1) = k$, so $[t_m]$ is an even 1-run, and t_m brings us to vertex v_{r}. Hence by the opposite direction of this lemma applied on t_m, $t_m = t$, and the lemma is proven. $\square$

Let us now prove similar two lemmas for traversing r'.

Lemma 51 *Let $\epsilon \neq t \in \mathcal{T}_{3,6}$ start with a 1-run. If $\forall t' \sqsubseteq t;\ 0 \leq p(t') \leq k+1$ then by running t on r' starting at v:*

- *we get to vertex $(p(t))_{\mathrm{L}}$ iff $[t]$ is an odd 1-run,*
- *we get to vertex $(p(t))_{\mathrm{R}}$ iff $[t]$ is an odd 0-run,*
- *we get to vertex $(i)_{\mathrm{M}}$ iff ($i = p(t) - 1$ and $[t]$ is an even 1-run) or ($i = p(t) + 1$ and $[t]$ is an even 0-run). Furthermore, $0 < i < k+1$.*

Proof. The proof is by induction on $l(t)$. We can easily verify that claim is true for any t such that $l(t) = 1$. For $m > 1$, let us assume that claim is true for any t, such that $l(t) < m$, and let us prove it for t such that $l(t) = m$. Let $t' = t - 1$. By induction hypothesis the lemma is true for t'.

If after running t' on r' starting at v' we are at $(p(t'))_{\mathrm{L}}$ then $[t]$ has to be a 0-run. If $[t]$ is an odd 0-run then we get to $(p(t') - 1)_{\mathrm{R}}$ by running t starting at v', if $[t]$ is an even 0-run we get to $(p(t') - 1)_{\mathrm{M}}$ by running t starting at v. In the former case $p(t) = p(t') - 1$, in the latter one $p(t) = p(t') - 2$ and $(i)_{\mathrm{M}} = (p(t') - 1)_{\mathrm{M}} = (p(t) + 1)_{\mathrm{M}}$, hence $0 < i < k+1$.

If after running t' we are at $(p(t'))_{\mathrm{R}}$ then $[t]$ has to be a 1-run, and if it is an odd run then we get to $(p(t))_{\mathrm{L}}$ by running t on r', otherwise we get to $(p(t) - 1)_{\mathrm{M}} = (p(t') + 1)_{\mathrm{M}} = (i)_{\mathrm{M}}$ and $0 < i < k+1$.

If after t' we are at $(i)_{\mathrm{M}}$ then either $i = p(t') - 1$ and $[t]$ is a 0-run, or $i = p(t') + 1$ and $[t]$ is a 1-run. If $[t]$ is an even 0-run then $p(t) = p(t') - 2$ and we are at $(i)_{\mathrm{M}} = (p(t) + 1)_{\mathrm{M}}$ after t ($0 < i < k+1$,) if $[t]$ is an odd 0-run then $p(t) = p(t') - 1$ and we are at $(i)_{\mathrm{R}} = (p(t))_{\mathrm{R}}$. If $[t]$ is an even 1-run we are at $(p(t) - 1)_{\mathrm{M}} = (i)_{\mathrm{M}}$ after running t and $0 < i < k+1$, otherwise we are at $(p(t))_{\mathrm{L}}$.

There are no other possibilities. That establishes the lemma. □

Lemma 52 *Let $\epsilon \neq t \in \mathcal{T}_{3,6}$ start with a 1-run. The following are equivalent:*

1. *$\forall t' \sqsubset t$, we are at a vertex belonging to r' after running t' on r' starting at v', and we are at a vertex outside of r' after running t on r',*

2. $\forall t' \sqsubset t;\ 0 \le p(t') \le k+1$ *and* $((p(t-1) = 1$ *and* $[t]$ *is an even 0-run), or* $(p(t-1) = k$ *and* $[t]$ *is an even 1-run.))*

Proof. Again, (2) $\Rightarrow$ (1) follows from the previous lemma. (Note, if $p(t-1) = 1$ and $[t]$ is an 0-run then $[t-1]$ can be neither an even 1-run nor a 0-run. The former case would imply that $p((t-1)-1) < 0$, the latter one that $[t]$ is a 1-run. Hence, after running $t-1$ we are at 1_{L}. Similarly, if $p(t-1) = k$ and $[t]$ is an even 1-run then we are at k_{R}.)

(1) $\Rightarrow$ (2) follows by a similar argument as lemma 50. Let $t_m \sqsubseteq t$ be such that $p(t_m) < 0$ or $p(t_m) > k+1$, and $\forall t' \sqsubset t_m;\ 0 \le p(t') \le k+1$. (Such t_m must exists by the previous lemma.)

If $p(t_m) < 0$ then $p(t_m - 1) \le 1$, $[t_m]$ is a 0-run, $[t_m - 1]$ is a 1-run, hence $p(t_m - 1) = 1$, $[t_m - 1]$ must be an odd 1-run and $[t_m]$ must be an even 0-run. Thus, t_m brings us to vertex v'_{l}.

Similarly, if $p(t_m) > k+1$ then $[t_m]$ is a 1-run, $p(t_m - 1) = k$, $[t_m - 1]$ is an odd 0-run and t_m is an even 1-run. Thus, t_m brings us to vertex v'_{r}. Therefore by the opposite direction of this lemma applied on t_m, $t_m = t$, and the lemma is proven. □

To conclude Lemma 48 notice, that $p(t)$ depends only on t, but neither on r nor on r'. Hence, Lemma 48 follows from Lemmas 50 and 52. □

Further, we can establish an equivalent of Lemma 48 for traversal from right, i.e., claim that any (3,6)-sequence t behaves the same on original sequence r and stuffed sequence r' starting at their rightmost vertices (instead of the leftmost ones), where output vertices are as in Lemma 48.

The proof of such a claim would be entirely symmetric to the proof of Lemma 48 except for the fact that the rightmost run of the stuffed sequence has length 2, 4, or 6 depending on d. (Proposition 46 tells us that this really does not make any difference for traversal by (3,6)-sequences.)

Vertices in r would be numbered by $\{0, \ldots, l-1\}$ from the right, and labels of vertices in r' would be interchanged in the following way: $(k+1)_{\mathrm{L}} \leftrightarrow 0_{\mathrm{R}}$, $(i)_{\mathrm{L}} \leftrightarrow (k+1-i)_{\mathrm{R}}$, $(i)_{\mathrm{R}} \leftrightarrow (k+1-i)_{\mathrm{L}}$, $(i)_{\mathrm{M}} \leftrightarrow (k+1-i)_{\mathrm{M}}$, for $1 \le i \le k$ (Figure 4.5). Then, Lemmas 49, 50, and 52 would be exactly the same. The only difference would be in Lemma 51 for the case of $d = 6$. Because vertex v' is not labeled by 0_{R}, we would need to assume w.l.o.g. that the first run of t is an even run to get to a labeled vertex. (The case of t beginning with an odd run can be reduced to the case of t beginning with an even run.)

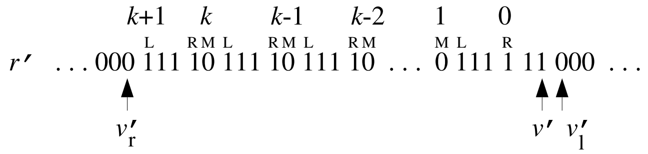

Figure 4.5: The labeling for traversal from right.

We may conclude that any (3,6)-sequence behaves the same on the original and stuffed sequences if started at the first or last vertices and with respect to output vertices being the neighboring ones.

Let us state a lemma which relates traversal of s and s' obtained from s by 1-run breaking.

Lemma 53 *Let s' be obtained from a non-monochromatic graph sequence s by 1-run breaking. Let v be a border vertex of some run in s and let v' be its corresponding vertex in s'. Let (3,6)-sequence t strongly traverse s' starting at v'. Then t, starting at v, completely traverses s, and all vertices in s except possibly v are strongly traversed.*

Proof. Because t behaves the same on original runs in s as on the corresponding stuffed runs in s', t traverses all runs in s in the same order as it traverses the corresponding stuffed sequences in s'. Thus, we only need to argue that for any run r in s and its corresponding stuffed sequence r' in s', if all vertices in r' are strongly traversed during traversal of s' by t, then all vertices in r but v are strongly traversed during traversal of s by t, too.

If $r = r'$ then there is nothing to prove. (Note, bouncing at a vertex during traversal of a neighboring stuffed sequence does not constitute a complete traversal of that vertex, so there must be some run in t which starts its running in r and r', respectively.) Let us assume $r \neq r'$ and let $|r| = l = 3k + d$, for $d \in \{2, 4, 6\}$.

If there is a (3,6)-subsequence[2] t' of t which during traversal of s' by t starts to run on r' at the leftmost vertex of r' and which leaves r' at the right neighboring vertex of r', or which runs on r' starting at the rightmost vertex and which leaves r' at the left neighboring vertex of r', then, by Lemmas 49 and 51, during traversal of s by t, subsequence t' traverses r in exactly the same manner, i.e., it starts in r at one side and leaves r on the other side. Clearly, all the vertices of r are strongly traversed during traversal by t'. (The possible exception is if t' is a prefix of t and t' starts running at v then v may not be strongly traversed by t'.)

If such a (3,6)-subsequence t' does not exist, let t_l be a (3,6)-subsequence of t which during traversal of s' by t starts its traversal of r' at the leftmost vertex of r' and traverses r' furthest to the right among all such subsequences. Similarly, let t_r be a (3,6)-subsequence which traverses r' from right furthest to the left. It is not obvious that t_r and t_l exist. We will show that t_l always exists, and that t_r exists if $d \neq 2$.

Suppose that t_l does not exist. Vertex 0_R in r' has to be strongly traversed, hence it has to be strongly traversed from the right. That means, in t_r there must be a run of length 6 running from 1_L. But that run ends outside of r', thus, t_r has a property of sequence t' which is a contradiction. Therefore t_l always exists. For $d \in \{4, 6\}$, t_r has to exist by similar argument. (If $d = 2$, the argument for existence of t_r fails because the rightmost vertex of r' might be strongly traversed from the left by t_l using a run of length 3.)

Let u_l be the furthest labeled vertex to the right strongly traversed by t_l in r'. If u_l is 0_R or 1_L then t_r has to exist to strongly traverse 1_M. By Lemmas 49 and 51, t_l traverses vertices $0, \ldots, 3$ in r, so it strongly traverses vertices $0, \ldots, 2$ in r. By Lemmas 49 and 51 modified for traversal from right, t_r has to visit vertices $0, \ldots, 3(i+1)$ numbered from right in r, where $i = (k+1) - 1$, in order to traverse vertex 1_M (which is s vertex $(i)_\mathrm{M}$ labeled from right). (Lemmas 49 and 51 are applicable because t_l and t_r have to start with a run of appropriate color.) Hence, t_r strongly traverses $3(k+1)$ vertices from right in r, and t_l along with t_r strongly traverse all vertices.

[2] t' is obtained from t by removing complete runs from the beginning of t and from the end of t.

If u_{l} is neither 0_{R} nor 1_{L}, let i_{l} be the largest i such that t_{l} strongly traverses $(i)_{\mathrm{M}}$ during its traversal in r'. Thus, t_{l} has to strongly traverse vertices $0, \ldots, 3(i_{\mathrm{l}}+1)-1$ in r. If $i_{\mathrm{l}} = k$ and $d = 2$ then all vertices in r are traversed strongly by t_{l}. If $i_{\mathrm{l}} = k$ and $d = 4, 6$, then t_{r} has to exist and must strongly traverse at least three rightmost vertices of r. Hence, t_{l} and t_{r} strongly traverse altogether $3(k+2) \geq l$ vertices in r (for $d = 4$ some of them twice), therefore all vertices in r are strongly traversed.

If $i_{\mathrm{l}} < k$ then t_{r} must exist, and let i_{r} be the largest i such that $(i)_{\mathrm{M}}$ is reached by t_{r} in r' (labeled right-to-left). Clearly, $(k+1) - i_{\mathrm{r}} \leq i_{\mathrm{l}} + 1$. By Lemmas 49 and 51, t_{l} strongly traverses vertices $0, \ldots, 3(i_{\mathrm{l}}+1)-1$ in r (labeled left-to-right) and t_{r} strongly traverses vertices $0, \ldots, 3(i_{\mathrm{r}}+1)-1$ in r (labeled right-to-left). Thus, they strongly traverse at least $3(i_{\mathrm{l}}+1) + 3((k+1) - (i_{\mathrm{l}}+1) + 1) \geq l$ vertices in r.

Note, in all these cases the first and last vertices of r are traversed strongly because t has to run "around" s to get from one end of r to the other one in between t_{l} and t_{r}.

Hence, in all cases, all vertices in s except possibly v are strongly traversed. □

Let s and s' be as in the statement of Lemma 53. Let t be a traversal (3,6)-sequence which strongly traverses s' starting at any vertex. Let sequence t' be obtained from t by prepending a run w (called a "loader") of length $|s|$ of opposite color than the first run of t. Then, t' strongly traverses s starting at any vertex.

Clearly, if we run t' on s starting at any vertex then w brings us to a border vertex v in s, and t starts to run from that border vertex. By the previous lemma, all vertices in s are strongly traversed by t but vertex v. However, v is also traversed strongly because of w followed by t.

We conclude that stuffing long runs of 0's and 1's is transparent for (3,6)-traversal sequences. We presented the analysis explicitly for (3,6)-sequences but essentially identical analysis can be carried out for any $(2i+1, 4i+2)$-sequence, for $i \geq 1$, where we stuff long runs every $(2i+1)$ vertices. The reason for choosing (3,6)-sequence in this step is the optimality of resulting sequence with respect to other possible choices. (See also Section 4.4.2.)

Let s' be a (1,2)-sequence and t' be a strong traversal sequence of s' starting at a vertex v'. We say that t' *extra strongly traverses* s' if every pair in s' is traversed by t' in such a way that it is entered from outside at one border vertex, walked back and forth on it and then left through the other border vertex.

Lemma 54 *Let s be a non-monochromatic graph sequence with every run of length from $\{1, 2, 3, 4, 6\}$. Let $s_\oplus$ be the parity contraction of s. Let v be a border vertex in s, and $v_\oplus$ its corresponding vertex in $s_\oplus$. Let $t_\oplus$ be an extra strong traversal (1,2)-sequence of $s_\oplus$ starting at $v_\oplus$. Then t, obtained from $t_\oplus$ by replacing every run of length 1 by run of length 3, and every run of length 2 by run of length 6, completely traverses s starting at v.*

Proof. First observe that t started on s at v traverses all runs of s in the same order as $t_\oplus$ started on $s_\oplus$ at $v_\oplus$. This is because any (3,6)-sequence has the same behavior on runs of length 2,4 and 6, and on runs of length 1 and 3, and these are the only lengths that may appear in s.

Now notice that t traverses s completely. It is easy to see that because of the strong traversal of $s_\oplus$ by $t_\oplus$, all vertices in s of runs no longer than 4 are traversed completely (and strongly) (use the same argument as in Proposition 43).

Let us consider a run r in s of length 6. Its parity contraction is a pair $r_\oplus$ in $s_\oplus$, vertices of which are extra strongly traversed. Hence, there is a run of length 6 in t which runs starting at one border vertex of r, traverses all vertices of r and leaves on the other end of r. Clearly, all vertices in r are strongly traversed. $\square$

Note, if s' is obtained from a non-monochromatic graph sequence s by 1-run breaking, and $s_\oplus$ is obtained from s' by parity contraction, then the length of $s_\oplus$ is at most the length of s. (During 1-run breaking we insert 10 and 01 after every three 1's and 0's, respectively. Two of these vertices are always removed by parity contraction.)

Let $t_\oplus$ be a (1,2)-sequence which extra strongly traverses any graph (1,2)-sequence of length $|s|$ starting at any vertex. W.l.o.g. the first vertex in $t_\oplus$ has color 0. Let t' be obtained from $t_\oplus$ by the construction from Lemma 54, and let t be obtained from t' by prepending a loader $1^{|s|}$. Then t strongly traverses s starting at any vertex. Moreover, t is a strong UTS for cycles of length $|s|$. (If t is started at any vertex in

s then it first reaches a border vertex in s by the loader (or s is strongly traversed if s is monochromatic.) Call that vertex v. There is a vertex v' in s' corresponding to v, and a vertex $v_{\oplus}$ in $s_{\oplus}$ corresponding to v'. By Lemma 54, s' is strongly traversed by t' starting at v', therefore, s is strongly traversed by t starting at any vertex, using Lemma 53 and the remark following it.)

4.3.2 01-run Breaking

This stage is aimed at breaking down long 01-runs. The method used here for breaking long 01-runs shares the same spirit with the previous method of breaking 0-runs and 1-runs. Because of that the substance of the proof of correctness of this stage is identical to the proof of correctness of the previous stage but the objects are different. Hence, instead of presenting here the full proof, which would be merely a rephrasing of the previous one, we point out the common elements among these two stages and state the appropriate lemmas. Let us first describe this stage.

A (5,10)-01-*sequence* is a (1,2)-sequence in which every two consecutive pairs are separated by a 01-run and every 01-run is of length five or ten. Note, a 01-run between consecutive pairs of the same color always has odd length, and a 01-run between pairs of different colors always has even length. Therefore, in any (5,10)-01-sequence pairs of the same color are separated by 01-runs of length five and pairs of different colors are separated by 01-runs of length ten.

Let s be a graph (1,2)-sequence containing at least one pair. We replace every 01-run in s of length $l \in \{7, 9, 11, 12, 13, 14, \ldots\}$ by a *stuffed* sequence r' according to the following table. Let $l = 5k + d$ as in Proposition 47. Let $r_0 = 01010$, $r_1 = 10101$, $r'_0 = 01010\ 100100$, and $r'_1 = 10101\ 011011$.

k even:

$(r_0 r_1)^{k/2}(01)^{d/2} \rightarrow (r_0' r_1')^{k/2}(01)^{d/2}$

$(r_1 r_0)^{k/2}(10)^{d/2} \rightarrow (r_1' r_0')^{k/2}(10)^{d/2}$

k odd:

$(r_0 r_1)^{(k-1)/2} r_0 (10)^{d/2} \rightarrow (r_0' r_1')^{(k-1)/2} r_0' (10)^{d/2}$

$(r_1 r_0)^{(k-1)/2} r_1 (01)^{d/2} \rightarrow (r_1' r_0')^{(k-1)/2} r_1' (01)^{d/2}$

Hence, we insert 011011 and 100100, alternatively, every five vertices in r. Observe that the first stuffing that is inserted in r is 011011 if and only if the pair ending r on the left is a 00-pair, and the last stuffing that is inserted in r is 011011 if and only if the pair ending r on right is a 00-pair. Thus, all 01-runs in stuffed sequence r' are of even length less or equal to ten.

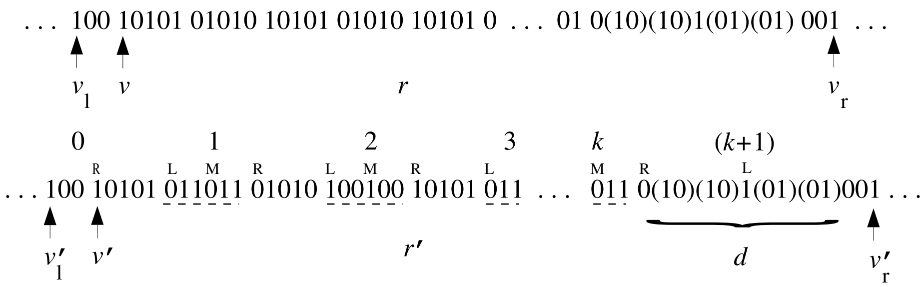

Figure 4.6: 01-run breaking.

Similarly to the previous 1-run breaking stage we want to argue that a (5,10)-01-sequence t behaves the same on the original and on the stuffed sequence with respect to particular input and output vertices.

Let $\mathcal{T}_{5,10}$ be the class of (5,10)-01-sequences starting with a 01-run and ending with a pair. Let $\epsilon \in \mathcal{T}_{5,10}$. Let a prefix relation $\sqsubseteq$ on $\mathcal{T}_{5,10}$ be defined by $x \sqsubseteq y$ if and only if x is a prefix of y.

Lemma 55 (≈48) *Let s' be obtained from a graph (1,2)-sequence s, that contains at least one pair, by 01-run breaking. Let r be a 01-run in s and r' its corresponding (stuffed) sequence in s'. Let v be the leftmost vertex of r, and let v' be the leftmost vertex* *of r'. Let v_{l} (v'_{l}) be the vertex immediately to the left of the pair that is neighboring r (r') on the left, and let v_{r} (v'_{r}) be the vertex just to the right of the pair that is neighboring r (r') on the right (Figure 4.6). Then every $t \in \mathcal{T}_{5,10}$ behaves the same on r and r' with respect to vertices $v, v_{\mathrm{l}}, v_{\mathrm{r}}$ and $v', v'_{\mathrm{l}}, v'_{\mathrm{r}}$.*

A proof of this lemma goes along the same lines as the proof of Lemma 48. Therefore, we will not present the full proof but rather we will present statements of the necessary lemmas which correspond to lemmas in the proof of Lemma 48. Lemma 55 follows from them in a similar way.

*Proof of Lemma 55 (*sketch*).* For the proof, we assume that $r \neq r'$, otherwise the lemma is trivial, we set $l = |r|$, and $l = 5k+d$ as in Proposition 47. We number vertices in r from left to right by $\{0, 1, \ldots, l-1\}$, and we refer to vertices in r' as in Figure 4.6, i.e., there are vertices labeled $0_{\mathrm{R}}, 1_{\mathrm{L}}, 1_{\mathrm{M}}, 1_{\mathrm{R}}, 2_{\mathrm{L}}, \ldots, k_{\mathrm{M}}, k_{\mathrm{R}}, (k+1)_{\mathrm{L}}$ in r'. (The vertex labeled $(k+1)_{\mathrm{L}}$ is the sixth vertex from the left of the last 01-run, for $d \in \{6, 8, 10\}$, and it is the rightmost vertex of that 01-run otherwise.) For the rest of the proof we may assume that t and r start with the vertex of color 1. (This corresponds to assumption that r is a 1-run and t starts with a 1-run in the proof of Lemma 48.)

For any $\tau \in \mathcal{T}_{5,10}$ let us call any 01-run in τ, which is preceded by an odd number of pairs, an *even positioned 01-run* (e.p. 01-run) and let us call a 01-run, which is preceded by an even number of pairs, an *odd positioned 01-run* (o.p. 01-run). (Even positioned 01-runs correspond to 0-runs, and odd positioned 01-runs correspond to 1-runs in t in the proof of Lemma 48.) Let $[\tau]$ denote the last 01-run in τ, let $\tau - 1$ denote a sequence which is obtained from τ by removing $[\tau]$ and the last pair in τ, and let $l(t)$ denote the number of 01-runs in τ.

Define $o_0^\tau, o_1^\tau, e_0^\tau, e_1^\tau$ to be the number of odd e.p. 01-runs, odd o.p. 01-runs, even e.p. 01-runs, and even o.p. 01-runs in τ, respectively. (Notice, $l(\tau) = o_0^\tau + o_1^\tau + e_0^\tau + e_1^\tau$.) Further, define $p(\tau) = (o_1^\tau + 2e_1^\tau) - (o_0^\tau + 2e_0^\tau)$.

Now we state the lemmas which describe traversal of r by t.

Lemma 56 (≈49) *Let $t \in \mathcal{T}_{5,10}$ be such that $t = \epsilon$ or t starts with a vertex of color 1. Let $\forall t' \sqsubseteq t;\ 0 \le p(t') \le k+1$. Then the following hold:*

1. $\forall t' \sqsubseteq t$, *if* $p(t') = 0$ *then* $[t']$ *is an e.p. 01-run, and if* $p(t') = k+1$ *then* $[t']$ *is an o.p. 01-run,*

2. *if* $d = 6, 8, 10$ *then we are at a vertex numbered* $5p(t)$ *in* r *after running* t *on* r *starting at* v*,*

3. *if* $d = 2, 4$ *then we are at vertex* $5p(t) - \delta_d(t)$ *in* r *after running* t *starting at* v*, where* $\delta_d(t) \in \{0, 6-d\}$*, and* $\delta_d(t) \neq 0$ *iff* $(\exists t' \sqsubseteq t;\ p(t') = k+1$ *and* $\forall t''$ *such that* $t' \sqsubseteq t'' \sqsubseteq t,\ p(t'') > 0)$.

The first two parts of this lemma are easy to verify. You can show the third part of the lemma by an induction on the number of $t' \sqsubseteq t$ such that $p(t') = 0$ or $k+1$, similarly to the proof of Lemma 49.

Lemma 57 (≈50) *Let* $\epsilon \neq t \in \mathcal{T}_{5,10}$ *start with a vertex of color 1. The following are equivalent:*

1. $\forall t' \sqsubset t$*, we are at a vertex belonging to* r *after running* t' *on* r *starting at* v*, and we are at a vertex outside of* r *after running* t *on* r*,*

2. $\forall t' \sqsubset t;\ 0 \leq p(t') \leq k+1$ *and* $((p(t-1) = 1$ *and* $[t]$ *is an even e.p. 01-run), or* $(p(t-1) = k$ *and* $[t]$ *is an even o.p. 01-run.))*

The proof of this lemma is based on the previous lemma and is essentially identical to the proof of Lemma 50.

The following two lemmas describe a traversal of r' by t.

Lemma 58 (≈51) *Let* $\epsilon \neq t \in \mathcal{T}_{5,10}$ *start with a vertex of color 1. If* $\forall t' \sqsubseteq t;\ 0 \leq p(t') \leq k+1$ *then by running* t *on* r' *starting at* v*:*

- *we get to vertex* $(p(t))_{\mathrm{L}}$ *iff* $[t]$ *is an odd o.p. 01-run,*

- *we get to vertex* $(p(t))_{\mathrm{R}}$ *iff* $[t]$ *is an odd e.p. 01-run,*

- *we get to vertex* $(i)_{\mathrm{M}}$ *iff* $(i = p(t)-1$ *and* $[t]$ *is an even o.p. 01-run) or* $(i = p(t)+1$ *and* $[t]$ *is an even e.p. 01-run). Furthermore,* $0 < i < k+1$.

This lemma can be proven by an induction on $l(t)$, similarly to the proof of Lemma 51.

Lemma 59 (≈52) *Let $\epsilon \neq t \in \mathcal{T}_{5,10}$ start with a vertex of color 1. The following are equivalent:*

1. *$\forall t' \sqsubset t$, we are at a vertex belonging to r' after running t' on r' starting at v', and we are at a vertex outside of r' after running t on r',*

2. *$\forall t' \sqsubset t$; $0 \leq p(t') \leq k+1$ and $((p(t-1) = 1$ and $[t]$ is an even e.p. 01-run), or $(p(t-1) = k$ and $[t]$ is an even o.p. 01-run.))*

The proof of this lemma is based on the previous lemma and is essentially identical to the proof of Lemma 52. The conclusion of the proof of Lemma 55 is now the same as of the proof of Lemma 48. $\square$

We state the following proposition a proof of which is trivial hence, omitted.

Proposition 60 (≈46)

1. *Any sequence in $\mathcal{T}_{5,10}$ started at the leftmost vertices or rightmost vertices behaves the same on sequences* 10*,* 10 10*,* 10 10 10*,* 10 10 10 10*, and* 10 10 10 10 10*, where the output vertices are vertices that are neighbors of the surrounding pairs in graph sequences containing these sequences (similarly to Lemma 48). The same holds for negated sequences.*

2. *Similarly, any sequence in $\mathcal{T}_{5,10}$ behaves the same on sequences* 1*,* 101 *and* 10101*, where input and output vertices are as above. The same holds for negated sequences.*

Similarly to the case of Lemma 48 we could now state a lemma that would deal with traversal of r and r' by t starting at the rightmost vertices of r and r', respectively. All Lemmas 56 – 59 can be shown to be valid for such a traversal if r and r' are numbered from the right instead of from the left. The only minor difference would be in Lemma 58. For the case of $d = 8$ or 10, it would be necessary to assume (w.l.o.g.) that t begins with a 01-run of length ten. (This requirement is similar to the requirement that t

begin with an even 1-run in the modification of Lemma 51 for traversals starting at the rightmost vertices of r and r', respectively, which is necessary because the traversal does not start at a labeled vertex.)

Based on the previous lemmas it is possible to establish the following statement.

Lemma 61 (≈53) *Let s' be obtained from a graph (1,2)-sequence s containing at least one pair by 01-run breaking. Let t be a traversal (5,10)-01-sequence ending with a 01-run of length ten. Let v be the left vertex of some pair in s if that pair has the same color as the first vertex of t, or let v be the right vertex of that pair otherwise. Let v' be the vertex corresponding to v in s'. If t extra strongly traverses s' starting at v' then t extra strongly traverses s starting at v.*

Proof. First observe that t traverses s in similar way as s', in particular, a pair in s is extra strongly traversed during traversal by t whenever its corresponding pair in s' is extra strongly traversed. The first pair in t brings us in s and s', respectively, outside of the pair containing v and v', respectively. In both sequences we get either to the left neighbor of that pair or to the right neighbor, and that vertex is either a vertex of a pair or a vertex of a 01-run in both sequences. (A 01-run breaking never inserts a pair next to another pair, so such a pair must exist in both sequences.) In the former case we can apply an induction on the length of t to conclude that t traverses s and s' in similar way. In the latter case we can use the fact that a (5,10)-01-sequence starting with a 01-run and ending with a pair, i.e., a sequence from $\mathcal{T}_{5,10}$, behaves the same on a 01-run and its corresponding stuffed sequence, and then again use the induction on the length of t.

Hence, all pairs in s are extra strongly traversed by t starting at v. Using an argument similar to the one in the proof of Lemma 53, because all vertices in s' are strongly traversed by t, all vertices in 01-runs in s are strongly traversed, too. (t starts the traversal on a vertex belonging to a pair which is extra strongly traversed by t, so the exception of the starting vertex not to be strongly traversed does not occur here.) □

We conclude this section with the following lemma.

Lemma 62 *Let s be a (1,2)-sequence containing at least one pair with every 01-run having length in $\{1, 2, 3, 4, 5, 6, 8, 10\}$, and let s_c be the pair contraction of s. Let t_c be a strong traversal sequence of s_c starting at the vertex v_c corresponding to some pair in s. Let v be the left vertex of that pair if the color of that pair is the same as the color of the first vertex in t_c, or let v be the right vertex of that pair otherwise. Let t be obtained from t_c using the following rules:*

vertex	*replace by*
0 followed by 0	*00 10101*
0 followed by 1 (or nothing)	*00 1010101010*
1 followed by 1	*11 01010*
1 followed by 0 (or nothing)	*11 0101010101*

Then t extra strongly traverses s starting at v.

Proof. We want to argue that t traverses pairs in s in a similar order as t_c traverses vertices in s_c, in particular, if $v_{a_0}, v_{a_2}, \ldots, v_{a_l}$ is a sequence of vertices in s_c as traversed by t_c, $l = |t_c|$, then a sequence of pairs in s as traversed by t can be written as $p_{a_{i_0}}, p_{a_{i_2}}, \ldots, p_{a_{i_{l'}}}$, where pair p_i corresponds to vertex v_i in s_c, $i_0 = 0$, for all $1 \leq j \leq l'$, $i_j \in \{i_{j-1}+1, i_{j-1}+2\}$, and $i_{l'} \in \{l, l-1\}$.

Let us consider the following cases occurring during the traversal of s and s_c by t and t_c, respectively. In all four considered cases let us assume that we reached a vertex u_c colored 1 in s_c and its corresponding pair in s during these traversals. Let us further assume that the next digit in t_c is 1 and the digit following it has color c. Hence we assume, in t there is going to be $11(01)^5$ or $11\,01010$ (depending on c). Let u_c^R denote the right neighbor of u_c in s_c.

If u_c^R has color 1 then pairs corresponding to u_c and u_c^R in s are separated by a 01-run of odd length, i.e., of length at most five. Thus the digit 1 in t_c brings us to u_c^R, and the 11-pair followed by a 01-run in t brings us to a pair corresponding to u_c^R in s.

If u_c^R has color 0 then pairs corresponding to u_c and u_c^R in s are separated by a 01-run of even length at most ten (or they are separated by an empty string). If c is 0 then there is $11(01)^5$ followed by a 00-pair in t. In that case, the digit 1 in t_c brings us to vertex u_c^R in s_c, and $11(01)^5$ in t brings us to the pair corresponding to u_c^R in s. If c is 1 then there is $11\,01010\,11(01)^5$ or $11\,01010\,11\,01010$ in t. Digit 1 followed by the other digit 1 in t_c brings us first to u_c^R and then back to u_c in s. Similarly, $11\,01010\,11(01)^5$ and $11\,01010\,11\,01010$ in t bring us to the pair corresponding to u_c in s, where the pair in s corresponding to u_c^R may (but need not) be visited during that.

If u_c and t_c are of different colors than that we have considered the behavior is symmetric.

Hence we may conclude that pairs in s are traversed in the same order as the corresponding vertices in s_c, but some pairs in s may occasionally not be visited by t during the traversal. However, if a vertex in s_c is traversed strongly then the corresponding pair in s is (extra strongly) traversed at the same time. Because all vertices in s_c are strongly traversed, all pairs in s are extra strongly traversed.

The extra strong traversal of pairs in s means that we have to reach every pair from the left or right, respectively, and then leave it to the right or left, respectively. In particular, if a pair in s has neighboring 01-runs, at least five of the neighboring vertices (or fewer if the 01-runs are shorter than five) in each of these runs are traversed during the extra strong traversal of that pair. Actually, these vertices are all traversed strongly because the 01-runs in t, which traverse them, are preceded and followed by pairs. Because every 01-run in s has length at most ten and has neighboring pairs on both sides, all vertices in 01-runs are traversed strongly. Hence, s is extra strongly traversed. $\square$

Note, if s' is obtained from a graph (1,2)-sequence s containing at least one pair by a 01-run breaking, and s_c is obtained from s' by a pair contraction, then $|s| \geq 2|s_c|$. (During the 01-run breaking we insert two pairs at most every five singletons, where all singletons are removed by the pair contraction, and every pair is reduced by half.)

Let t_c be a UTS for cycles of length $\lceil |s|/2 \rceil$. W.l.o.g. the first vertex in t_c has color 1. Let t' be obtained from t_c by the construction from Lemma 62, and let t be obtained from t' by prepending a loader $0(10)^{\lfloor |s|/2 \rfloor}$. Then t extra strongly traverses s starting at any vertex. Moreover, t extra strongly traverses any graph (1,2)-sequence of length

$|s|$. (If t is started at any vertex in s then it first reaches some pair in s by the loader. Let v be a vertex of that pair to which we get by the loader. There is a vertex v' in s' corresponding to v, and a vertex v_c in s_c corresponding to the pair containing v. By Lemma 62, s' is extra strongly traversed by t' starting at v', therefore, by Lemma 61, s is extra strongly traversed by t' starting at v, hence is extra strongly traversed by t.)

We choose here to stuff 01-runs every 5 singletons. Similarly to the 1-run breaking stage, we could choose to stuff 01-runs every $2i+1$ singletons, for $i \geq 1$, and use for traversal a $(2i+1, 4i+2)$-01-sequence.

4.3.3 The Construction

Lemmas 62, 61, 54, and 53 combine to give a construction that converts a strong UTS for cycles of length m into a strong UTS for cycles of length $2m$. This is presented explicitly in step 3 of the algorithm below.

Here is the algorithm constructing a UTS for cycles of length n. Let $k = \lceil \log_2 n \rceil$.

The Algorithm:

1. Let t_0 be a strong UTS for cycles of length 6, that starts with a vertex of color 1.

2. Apply the construction of Proposition 44 to t_0 to obtain a sequence t'_1, and attach in front of t'_1 sequence $0(10)^6$ to get t_1.

3. For $i = 1, \ldots, k-1$, construct a sequence t'_{i+1} from t_i by applying the following rules:

	replace by
a 00-pair	$[00\,10101]^5\,00(10)^5$
singleton 0	$[00\,10101]^2\,00(10)^5$
a 11-pair	$[11\,01010]^5\,11(01)^5$
singleton 1	$[11\,01010]^2\,11(01)^5$

   Prepend loader $0(10)^{6\cdot 2^i}[11\,01010]^{6\cdot 2^i}\,11(01)^5$ in front of t'_{i+1} to get a sequence t_{i+1}. (This step is a composition of the constructions from Lemmas 54 and 62.)

4. Apply the construction of Lemma 54 to t_k to obtain a sequence t', and attach sequence 1^n in front of t' to get a sequence t.

Sequence t produced by this algorithm is a universal traversal sequence for cycles of length n. There is a uniform NC^1 circuit that, given 1^n, will construct the UTS for cycles of length n.

Let us analyze this algorithm. Note that $t_1, \ldots, t_k$ are (1,2)-sequences. Also note that the sequences by which we replace individual pairs and singletons start and end by vertices of the same color as the replaced pairs and singletons. That means that there never appears a new pair consisting of one vertex from one replacement sequence and the other one from the next sequence. Thus, the replacement process is essentially "context free". We can describe the replacement process by context free grammar in which there is a single variable for a whole pair of each color and a single variable for a singleton of each color. Hence, we can associate with the output sequence a derivation tree implicitly created by the algorithm.

That derivation tree has essentially degree bounded by 41, except for the nodes that correspond to loaders and the root corresponding to t_1. Note that loaders may appear only on the left side of the tree and the parse tree leading to the loaders can have degree 2. Hence, we may represent a path from root to any leaf by a vector $< a_1, a_2, \ldots, a_{k+1}, m >$, where $0 \leq a_1 \leq l_{t_1}$, $a_2, \ldots, a_k \in \{0, 1, \ldots, 41\}$, $a_{k+1} \in \{0, \ldots, 6\}$, and $1 \leq m \leq 48 \cdot 2^k + 12$, where l_{t_1} denotes the length of t_1. The $a_1, \ldots, a_{k+1}$ denote the index number of the descendant node among its siblings, where $a_i = 0$ has a special meaning. If $a_1, \ldots, a_j$ are all zeros, for some j, and all the other a_i's are non-zero then that vector corresponds to a path going through m-th descendant of a loader at $(j+1)$-th level.

Observe that there are many vectors which actually do not correspond to any path. Any vector where $a_i \neq 0$ and $a_{i+1} = 0$, for some i, is illegal. Also not every node necessarily has 41 descendants, singletons have only 23 descendants. In any case we can test the legality of a given vector in NC^1 with respect to n (the length of the vector is logarithmic in n, hence a deterministic traversal of the vector is even $DLOGTIME(n)$).

Thus, there is an NC^1 algorithm which computes the path vector for any leaf of the derivation tree given index i of the leaf. (For every vector we can compute by brute force the number of invalid vectors lexicographically preceding it. By comparing this number with i we may decide which vector is the one which we are looking for. The lexicographical order has to appropriately take m into account.) From the path vector we may easily compute the value of i-th bit by $DLOGTIME(n)$ algorithm. Note, the resulting NC^1 circuit can actually be converted into a TC^0 circuit.

4.4 The Analysis

Let $f(i)$ denote the number of nodes at the i-th level in the derivation tree associated with a traversal sequence produced by the algorithm. (Every internal node in that tree corresponds either to a pair or to a singleton, leaves correspond to single vertices.) A rough analysis of the algorithm gives us a recurrence $f(i+1) \leq 41f(i) + 48 \cdot 2^i + 12$. By solving this recurrence, at level $k = \lceil \log_2 n \rceil$ of the tree there are $O(n^{\log_2 41}) = O(n^{5.358})$ nodes. At step 4 of the algorithm every node of the tree is expanded to 3 or 6 vertices. Thus the total length of the universal traversal sequence generated by the algorithm is $O(n^{5.358})$.

By slight modification of the algorithm and more careful analysis we may show that only $k' = 2\lceil \log_5 n \rceil \leq 2 + \log_{\sqrt{5}} n$ iterations of step 3 are necessary, and that the number $f'(i)$ of nodes at the i-th level of the derivation tree associated with the produced sequence satisfies recurrence $f'(i+1) \leq 25.635f'(i) + 48 \cdot 4^i + 12$. By solving the recurrence we get that the length of the produced sequence is $O(n^{\log_{\sqrt{5}} 25.635}) = O(n^{4.031})$.

4.4.1 The Modified Algorithm

We have to slightly modify the algorithm to get the better bound. The modification is to replace 2^i by 4^i in the loader of t_{i+1} in step 3 of the algorithm, so to use the loader $0(10)^{6 \cdot 4^i}[11\,01010]^{6 \cdot 4^i}11(01)^5$ instead. This modification is necessary because we want to argue that a graph sequence shrinks to $1/\sqrt{5}$ of its length by contractions (to

1/5 every two iterations) and not only to 1/2. The loader has to be long enough to traverse the whole graph sequence, thus its length has to correspond to the length of the traversed sequences at the given iteration of the algorithm ($(\sqrt{5})^i \leq 4^i$).[3]

Proposition 63 *For $i = 1, \ldots, k$, let p_i and s_i denote the number of pairs and singletons, respectively, in t_i. Then $\frac{p_i}{p_i+s_i} < \frac{6}{41}$.*

Proof. By construction, t'_1 contains one pair per 7 singletons. The loader attached to t'_1 to get t_1 can only lower the ratio of pairs. Thus, $\frac{p_1}{p_1+s_1} \leq \frac{1}{8} < \frac{6}{41}$.

For $i = 1, \ldots, k-1$, t'_{i+1} is obtained from t_i by replacing singletons and pairs according to rules specified in step 3 of the algorithm. The ratio of pairs is $\frac{6}{41}$ in the replacement for pairs, and $\frac{3}{23}$ in the replacement for singletons. The ratio of pairs is at most $\frac{1}{8}$ in the loader attached to t'_{i+1} to get t_{i+1}. Hence, $\frac{p_{i+1}}{p_{i+1}+s_{i+1}} < \frac{6}{41}$. □

By the proposition, t_i contains fraction of at most $\frac{6}{41}$ of pairs, $i = 1, \ldots, k$. That means that at the i-th level in the derivation tree there is a fraction of at most $\frac{6}{41}$ of nodes having 41 descendants, whereas the remaining nodes have only 23 descendants. Thus on average, there are at most $41 \cdot 6/41 + 23 \cdot 35/41 \doteq 25.635$ descendants per node. Thus we obtain the recurrence $f'(i+1) \leq 25.635 f'(i) + 48 \cdot 4^i + 12$ for the number of nodes at the i-th level.

The Shrinking Factor

We want to show that a graph sequence s_c^2 obtained from an arbitrary graph sequence s by repeating the sequence of operations (1-run breaking, parity contraction, 01-run breaking, pair contraction) twice has length at most one-fifth of the length of s. We are going to use the following lemmas.

Lemma 64 *Let s be a graph sequence and let a (1,2)-sequence $s_\oplus$ be obtained from s by 1-run breaking and parity contraction. Then there is a 1-1 mapping of vertices in $s_\oplus$ into vertices of s, which preserves order of vertices and which maps every pair in $s_\oplus$ to two neighboring vertices of the same color in s.*

[3] We could use 3^i instead of 4^i as well, but for the sake of uniformity it is better to use 4^i.

By preserving the order of vertices we mean that images of any three vertices are in the same left-to-right order as their pre-images. A proof of this lemma follows from Figure 4.7-a),b),c). Notice, any vertex colored 1 in $s_\oplus$, which comes from 01 stuffing of some 0-run, is always surrounded by 00-pairs in $s_\oplus$, hence it cannot form a pair with neighboring vertices. Similarly, any vertex colored 0 coming from 10 stuffing of some 1-run cannot form a pair with its neighbors. Cases a) and b) in Figure 4.7 occur only for runs having length in $\{1, 2, 3, 4, 6\}$.

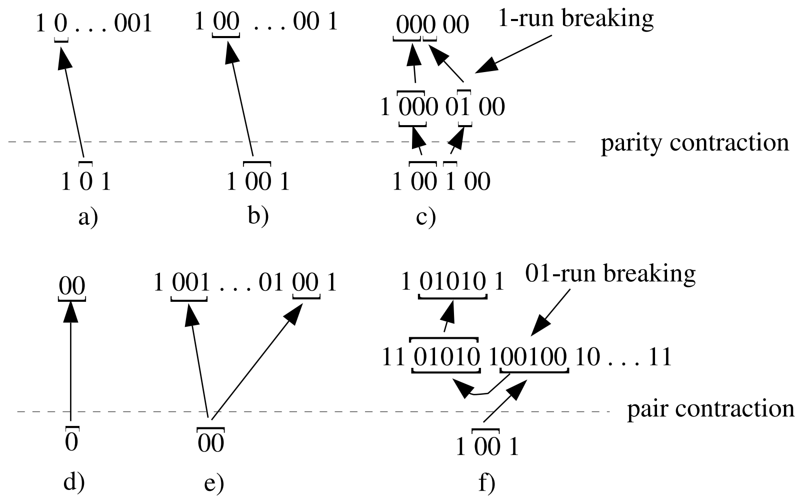

Figure 4.7: The vertex mappings.

Lemma 65 *Let $s_\oplus$ be a (1,2)-graph sequence and let a sequence s_c be obtained from $s_\oplus$ by 01-run breaking and pair contraction. Let some vertices in s_c be grouped into non-overlapping pairs with adjacent vertices of the same color, and let the other vertices in s_c be left single.*

Then there is a map which maps every pair in s_c to five vertices in $s_\oplus$, and every single vertex in s_c to two vertices in $s_\oplus$, such that no vertex in $s_\oplus$ is in an image of two distinct objects from s_c. Moreover, every single vertex in s_c which does not come from a contraction of a pair inserted by the 01-run breaking is mapped to a pair in $s_\oplus$.

Proof. Let us consider a pair of vertices v_1, v_2 in s_c. Each of these two vertices comes from the contraction of some pair. Vertices v_1 and v_2 have the same color so the pairs they come from are of the same color, too. Each of these pairs may either be present in $s_\oplus$ or be inserted by 01-run breaking.

Let us consider the case when both pairs come from $s_\oplus$. Because these pairs are of the same color they must be separated by at least one vertex of different color in $s_\oplus$. Clearly, between these pairs there cannot be any other pair, so they are separated by 01-run of length at least one (but no more than five otherwise 01-run breaking would insert other pairs between them). We may associate pair v_1, v_2 with vertices as in Figure 4.7-e).

Let us consider the case that one of v_1 and v_2 comes from contraction of a pair inserted by 01-run breaking. Then we claim that the other one also comes from a pair inserted by 01-run breaking, moreover they both come from the same 011011 or 100100. W.l.o.g let us assume that v_1 and v_2 have color 0, so the one which comes from 01-run breaking comes from the contraction of a pair from an inserted 100100. We know that we alternate between 011011 and 100100 during 01-run breaking, and that the pair ending the broken 01-run on the left side has a different color than pairs in the first inserted 011011 or 100100, respectively, and also the pair ending the broken 01-run on the right side has a different color than the pairs in the last inserted 011011 or 100100, respectively. Hence, two vertices that come from the contraction of pairs from 100100 are surrounded by vertices of the opposite color, hence the two vertices obtained by the contraction of 100100 are v_1 and v_2. We may associate inserted 011011 and 100100, respectively, with five preceding vertices coming from the broken 01-run in $s_\oplus$ (Figure 4.7-f)). Hence, we have associated every pair in s_c with a five vertices in $s_\oplus$.

Let us consider single vertices in s_c. Any single vertex v in s_c comes from a contraction of some pair. If that pair comes from $s_\oplus$, we associate v with that pair (Figure 4.7-d)). Otherwise the pair comes from 011011 or 100100 inserted by 01-run breaking. We may associate v with the left two or right two vertices of the five vertices preceding the inserted 011011 and 100100, respectively, depending on whether v comes from the left or the right pair of the inserted 011011 and 100100, respectively. $\square$

We use the previous lemmas to prove the following one.

Lemma 66 *Let s be a graph sequence, and let s_c^2 be obtained from s by repeating the sequence of operations (1-run breaking, parity contraction, 01-run breaking, pair contraction) two times. Then the length of s_c^2 is at most one-fifth of length of s.*

Proof. We are going to show that we may assign to every vertex from s_c^2 five vertices from s so that no vertex in s is assigned to more than one vertex in s_c^2. Let us denote by $s_\oplus^1$ a sequence obtained from s by 1-run breaking and parity contraction, let s_c^1 denote a sequence obtained from $s_\oplus^1$ by 01-run breaking and pair contraction, and let $s_\oplus^2$ denote a sequence obtained from s_c^1 by 1-run breaking and parity contraction.

Some vertices in s_c^2 may come from contraction of pairs that were inserted during the last 01-run breaking. Group these vertices into pairs with their neighboring vertices of the same color (which originated from the same insertion), and leave the other vertices single. By Lemma 65 we may assign to every pair from s_c^2 five vertices in $s_\oplus^2$, and to every single vertex from s_c^2 a pair of vertices of the same color $s_\oplus^2$. By Lemma 64 any five vertices from $s_\oplus^2$ may be mapped to five vertices in s_c^1, and any pair from $s_\oplus^2$ to a pair of the same color from s_c^1. Let us keep together these pairs from s_c^1, and leave single the remaining vertices from s_c^1. By Lemma 65 every pair from s_c^1 may be mapped to five vertices from $s_\oplus^1$, and every single vertex from s_c^1 to two vertices from $s_\oplus^1$. Thus, we may associate with every vertex from s_c^2 five vertices from $s_\oplus^1$, and to every pair from s_c^2 ten vertices from $s_\oplus^1$. By one more application of Lemma 64 we get a map which maps single vertices from s_c^2 to five vertices from s, and pairs from s_c^2 to ten vertices from s. Hence, we may assign to every vertex from s_c^2 five vertices from s so that no vertex from s is assigned to more than one vertex from s_c^2. Thus, s has at least five times more vertices than s_c^2. □

Correctness of the Modified Algorithm

The correctness of the modified algorithm follows from the following lemma.

Lemma 67 *Let t be a strong UTS for cycles of length $n \geq 1$. Let t^2 be obtained from t by repeating the sequence of operations (construction of Lemma 62, attaching a loader $0(10)^{6\cdot 4^{j-1}}$, construction of Lemma 54, attaching a loader $1^{6\cdot 4^j}$) two times, where*

$j = 2\lceil \log_5 n \rceil + 1$ during the first repetition, and $j = 2\lceil \log_5 n \rceil + 2$ during the second one. Then t^2 is a strong UTS for cycles of length $5n$.

Proof. Let s be a graph sequence of length $5n$, let s_c^1 be obtained by application of 1-run breaking, parity contraction, 01-run breaking and pair contraction on s, and let s_c^2 be obtained by application of that sequence of operations on s_c^1. Let t^1 denote a traversal sequence obtained from t after the first repetition of operations in the statement of the lemma.

By Lemma 66 sequence s_c^2 has the length of at most n, so that it is strongly traversed by t starting at any vertex. By Lemmas 62 and 54, sequence s_c^1 is strongly traversed by t^1 starting at any vertex, given that attached loaders are longer than s_c^1. The length of s_c^1 is at most $5n/2$, the length of the first loader is $12 \cdot 4^{2\lceil \log_5 n \rceil} + 1 \geq 12 \cdot 16^{\log_5 n} \geq 5n/2$, and the length of the second loader is $6 \cdot 4^{2\lceil \log_5 n \rceil + 1} \geq 6 \cdot 16^{\log_5 n} \geq 5n/2$. Because t^1 is a strong traversal sequence for s_c^1 starting at any vertex, t^2 is a strong traversal sequence for s_c^2 starting at any vertex by similar argument. Thus, t^2 is a strong traversal sequence for cycles of length $5n$. $\square$

4.4.2 Further Improvements

The previous analysis of the shrinking factor is tight. It is easy to see that a long 01-run shrinks to almost exactly one-fifth of its size every two rounds. 01-runs are really a bottleneck of the shrinking. If we would choose to stuff 01-runs every 7 singletons instead of five during 01-run breaking, and we would use (7,14)-01-sequences for traversal, then 01-run breaking would not be a bottleneck, anymore. The 01-runs would shrink at rate 7 per two rounds, i.e., $\sqrt{7} \doteq 2.645$ per round on average.

In that setting it could be possible to show by iterating the previous analysis over several rounds using modifications of Lemma 64 and 65 that the shrinking factor converges to $1 + \sqrt{2}$ per round. In particular, for any $\epsilon > 0$, there is a $k \geq 1$ such that any graph sequence shrinks to $(1 + \sqrt{2} - \epsilon)^{-k}$ of its original length during k rounds of contractions.

The recurrence describing the algorithm using (7,14)-01-sequences would be $f(i +$

$1) \leq 33.618 f(i) + 60 \cdot 4^i + 16$, where $55 \cdot 6/55 + 31 \cdot 49/55 \doteq 33.618$. For ϵ small enough, the solution of the recurrence is $O(n^{3.989})$.

The details of the analysis are sufficiently tedious that we do not claim this as a theorem, but merely give an indication of how this further improvement can be obtained.

Chapter 5

Exploration Sequences

In Chapter 4 we have surveyed traversal sequences and presented a new construction of a universal traversal sequence for cycles. As one can notice, it seems to be rather difficult to explicitly construct universal traversal sequences even for such simple graphs as cycles. In fact, the existing lower bounds on the length of universal traversal sequences suggest that this ought to be the case. In this chapter we propose a new notion of traversal sequences that we call *exploration sequences*.

Exploration sequences share many useful properties with the formerly defined traversal sequences, but they also exhibit some new properties. In particular, they have an ability to backtrack, and their random properties are robust under choice of the probability distribution on labels. We also present simple constructions of polynomial length universal exploration sequences for cycles, cliques, expanders and trees. These universal exploration sequences are shorter than the corresponding lower bounds on the length of universal traversal sequences for cycles and cliques.

Exploration sequences are very similar to traversal sequences. The difference between them is how we interpret the edge labels given by the sequences. In the case of traversal sequences the label is interpreted absolutely as an edge label; in the case of exploration sequences there is a cyclic ordering of the edges at every vertex and the label is interpreted as a relative shift with respect to the edge by which we entered a given vertex.

Similarly to traversal sequences, it can be shown that with high probability a random exploration sequence of length $O(d^2n^3 \log n)$ is a universal exploration sequence for d-regular graphs. Thus, constructions of universal traversal sequences of length $n^{O(\log n)}$

for 3-regular graphs that are based on pseudo-random generators apply to universal exploration sequences, too. Also the results of Nisan et al. (1992), and Armoni et al. (2000) can be formulated in terms of exploration sequences, as well as other results on random traversal sequences. In particular, the strengthening of the upper bound on the length of non-uniform universal traversal sequences of Kahn et al. (1989), and Chandra et al. (1997) apply to universal exploration sequences, too. Beside that, random exploration sequences exhibit robustness under the choice of probability distribution on labels.

Our definition is motivated by the following. If we are told that we reach a vertex v in a graph following a traversal sequence t, it is impossible in principle to determine where the walk started (or even to tell what vertex was visited immediately previously to v). This ambiguity does not occur if the labeling of the graph is consistent as required for the construction of Hoory & Wigderson (1993). However, in neither of these cases is there a traversal sequence that depends only on t and that brings us back from v to the starting vertex. In contrast, with exploration sequences we can have such a sequence, we can *backtrack*. That could eventually be useful in construction of a universal exploration sequence for 3-regular graphs.

There is yet another work that is related to ours. Istrail (1990) defines even more powerful notion of traversal sequences. In his work the traversal sequence is used to guide a finite state automaton that traverses the graph. The automaton makes its moves depending on the traversal sequence and what he observes in the graph. Istrail uses this model to extend his previous construction of polynomial length universal traversal sequence for cycles to universal traversal sequences for graphs of small diameter. His construction is complicated and it appeared only as an extended abstract in conference proceedings.

5.1 Definitions, Basic Properties, and Explicit Constructions

In our new notion, during a walk on a graph G guided by an exploration sequence t, digits of t will be interpreted as labels *relative* to the edge by which we got to the vertex.

In contrast, in the former definition of traversal sequences, digits of t are interpreted directly as edge labels. Formally we define the exploration sequences as follows. We do not limit our definition to d-regular graphs.

Let $G(V, E)$ be a graph and l be a labeling of G. Let (u, v) be an edge of G. We say that $i \in \mathbf{Z}$ *takes us from edge* (u, v) *to an edge* (v, w), if $l(v, w) = l(v, u) + i \bmod \deg(v)$. We denote this by $(u, v) \rightarrow^i (v, w)$. (Intuitively, being at an edge (u, v) corresponds to being at v's end of that edge.) The symbol Λ denotes the empty move, i.e., $(u, v) \rightarrow^\Lambda (u, v)$. For $t \in (\mathbf{Z} \cup \{\Lambda\})^*$, we extend the notation recursively: for edges e, f, g, if $t = \epsilon$ then $e \rightarrow^t e$, and if $t = t'i$, for $t' \in (\mathbf{Z} \cup \{\Lambda\})^*$ and $i \in \mathbf{Z} \cup \{\Lambda\}$, then $e \rightarrow^t g$, provided that $e \rightarrow^{t'} f$ and $f \rightarrow^i g$. We call sequence t an *exploration sequence*. (If t contains symbol Λ then it is a *general exploration sequence*, otherwise it is *simple*.) We say that *t visits a vertex w starting at* (u, v) if there is a prefix t' of t and an edge (w', w) incident to w, such that $(u, v) \rightarrow^{t'} (w', w)$.

Let $\mathcal{G}$ be a class of connected undirected graphs. We say that an exploration sequence t is a *universal exploration sequence for* $\mathcal{G}$, if for every $G \in \mathcal{G}$ and every labeling of G, t visits every vertex in G starting at any edge e of G. Some of our constructions deal with d-regular graphs but note that many of our theorems and constructions do not require regularity. Observe than when we consider traversal of a d-regular graph by an exploration sequence we may assume without loss of generality that the exploration sequence is from $\{0, 1, \ldots, d-1\}^*$.

For comparison, recall that traversal sequences require $l(v, w) = i$ rather than $l(v, w) = l(v, u) + i \bmod \deg(v)$, when choosing where to go next according to i. There is also a minor technical difference between the former definition and this definition since we go from one (directed) edge to another (directed) edge whereas with traversal sequences we go from vertex to vertex.

We state the following simple theorem.

Theorem 68 (Backtracking Theorem) *1. Let $t \in \mathbf{Z}^*$ be an exploration sequence, where $t = t_1 t_2 \ldots t_k$. Then there is an exploration sequence $t' = \overline{t_k t_{k-1}} \cdots \overline{t_1}$, where $\overline{t_i} = -t_i$, such that for any graph G, and any labeling of G the following holds. If*

$(u, v) \in E(G)$, then $(u, v) \to^{t0t'0} (u, v)$.

2. *Let $d > 1$ be an integer. Let $t \in \{0, \ldots, d-1\}^*$ be an exploration sequence, where $t = t_1 t_2 \ldots t_k$. Then there is an exploration sequence $t' = \overline{t_k t_{k-1}} \cdots \overline{t_1} \in \{0, \ldots, d-1\}^k$, where $\overline{t_i} = -t_i \bmod d$, such that for any d-regular graph G, and any labeling of G the following holds. If $(u, v) \in E(G)$, then $(u, v) \to^{t0t'0} (u, v)$.*

The proof by induction on the length of t is rather simple, so we leave it as an exercise. We defined backtracking only for simple exploration sequences but it can be easily extended to general exploration sequences.

The following theorem says that (analogous to the situation with random traversal sequences) a random exploration sequence of sufficient length is a universal exploration sequence.

Theorem 69 *Let $d > 1$ be an integer, and let $n > 1$ be an integer. With high probability, a sequence chosen uniformly at random from $\{0, \ldots, d-1\}^{O(d^2 n^3 \log n)}$ is a universal exploration sequence for d-regular graphs on n vertices.*

This theorem follows from the fact that with high probability, a general exploration sequence of length $O(d^2 n^3 \log n)$ chosen uniformly at random is a universal exploration sequence for d-regular graphs on n vertices. This fact can be proven in exactly the same way as the equivalent theorem for traversal sequences (Aleliunas et al., 1979). If we consider a random walk according to a random traversal sequence then the next edge is chosen uniformly at random among the incident edges; the same is true if we consider a random walk according to a random exploration sequence. Since the proof of Aleliunas et al. (1979) is based on analysis of random walks in graphs and the random walks are the same in both settings, the proof remains valid for exploration sequences as well. Moreover, in the case of exploration sequences the probability distribution on the edge labels does not have to be uniform. This is discussed in section 5.3 where we prove a generalization of Theorem 69 for exploration sequences. Similarly to traversal sequences, results of Kahn et al. (1989) and Chandra et al. (1997) can be used to improve the upper bound on the length of universal exploration sequences to $O(dn^3 \log n)$ for $3 \le d \le n/2 - 1$, and $O(n^3 \log n)$ for $n/2 \le d$, respectively.

If we could construct a universal exploration sequence of polynomial length for 3-regular graphs in log-space, we could conclude $SL = L$ since the universal exploration sequence could be used for an s-t-connectivity testing. We do not know how to construct a universal exploration sequence for 3-regular graphs in log-space. However, we present several log-space constructions of universal exploration sequences for some other classes of graphs. We state here also two results concerning universal exploration sequences for 3-regular graphs. They are equivalent to similar results for traversal sequences. Using Nisan's pseudo-random generator we obtain the following claim.

Theorem 70 *There is an L^2 constructible exploration sequence of length $n^{O(\log n)}$ that is universal for* 3*-regular graphs.*

Proof. We give a sketch of the proof following Babai et al. (1992). Consider the randomized s-t-connectivity algorithm of Aleliunas et al. (1979). Fix an n-vertex connected undirected 3-regular graph G. Given an edge e_s and a vertex t in G, if we walk randomly for $O(n^3 \log n)$ steps in G starting from e_s, then the probability that we do not visit t is at most $1/12n^2$. Consider Nisan's pseudo-random generator $g_n : \{0,1,2\}^{O(\log^2 n)} \to \{0,1,2\}^{O(n^3 \log n)}$ such that the difference between acceptance probability of the algorithm when using a truly random walk and when using a random walk induced by g_n is at most $1/12n^2$. Since there are at most $3n^2$ different pairs of e_s and t, at least a half of the walks guided by an exploration sequence generated by g_n have the property that all vertices of G are visited regardless of the starting edge. This is true for any fixed graph G. Hence, by concatenating all possible outputs of g_n we obtain a universal exploration sequence of length $3^{O(\log^2 n)} = n^{O(\log n)}$. □

Using Theorem 25 we obtain the following conditional result. The proof is similar to that of Theorem 70; we omit the proof, here.

Theorem 71 *If there is a language A in $DSPACE(n)$ such that for some $\epsilon > 0$, no family of Boolean circuits of size $2^{\epsilon n}$ can compute A correctly on infinitely many* *input lengths, then there is a log-space constructible universal exploration sequence for* 3*-regular graphs.*

Let us now focus on other classes of graphs. The constructions of universal traversal sequences for cliques and expanders of Karloff et al. (1988) and Hoory & Wigderson (1993), respectively, can be "lifted-up", and reformulated in terms of exploration sequences using the backtracking ability. This reformulation is possible, although technically complicated, so we do not present it here and we rather present different constructions. These constructions are very simple.

Theorem 72 1^{n-1} *is a universal exploration sequence for cycles of length* n*.*

The following construction for cliques corresponds to visiting all neighbors of the starting vertex in the order given by a labeling of the clique.

Theorem 73 $(10)^{n-1}$ *is a universal exploration sequence for cliques of size* n*.*

Next, we give a construction of a universal exploration sequence for constant degree expander graphs. Indeed, the constructed sequence is a universal exploration sequence for any class of d-regular graphs of bounded maximum distance of any two vertices (the *diameter*). The construction employs a full backtracking up to the diameter of the graph.

We follow Hoory & Wigderson (1993) in the definition of expanders. Let $d \geq 3$ be an integer. A d-regular undirected graph $G = (V, E)$ on n vertices is a c*-expander* if for every $U \subset V$, the number of edges between U and $V - U$ is at least $c|U|(n - |U|)/n$. It is called an *expander* if it is c-expander for some $c > 0$. It is straightforward to verify that any d-regular c-expander has diameter at most $c' \log n$, for some constant c' independent of n. The constant c' depends however on the expansion factor c.

Define recursively: $s_d^0 = 0$, and for $h \geq 1$, define $s_d^h = 1(s_d^{h-1}1)^{d-1}$, and $t_d^h = (1s_d^{h-1})^d$.

Theorem 74 t_d^h *is a universal exploration sequence for undirected* d*-regular graphs with maximum distance (diameter) at most* h*.*

Corollary 75 $t_d^{O(\log n)}$ *is a universal exploration sequence for d-regular expanders.*

As in Hoory & Wigderson (1993), we hide the expansion factor of the expander in $O(\log n)$. The length of t_d^k is at most $2d^k$, hence, the universal exploration sequence for d-regular expanders is of length $n^{O(\log d)}$. The UTS for expanders, that is presented in Hoory & Wigderson (1993), is of length $n^{O(d \log d)}$.

We prove Theorem 74 for the special case of $d = 3$. For $d > 3$, the proof is analogous. The proof uses the following two lemmas. The first lemma says that s_3^h brings us always back to the starting edge but in the opposite direction. The second lemma says that s_3^h visits all vertices in a graph G that are reachable from the starting point by a path of length at most h without crossing the starting edge. Then, the proof of Theorem 74 combines these two lemmas.

Lemma 76 *Let G be an undirected 3-regular graph. For any integer $h \geq 0$, and any edge (u, v) of G, $(u, v) \to^{s_3^h} (v, u)$.*

Proof. The proof is by induction on h. For $h = 0$, the lemma is trivial. Assume $h > 0$. $s_3^h = 1s_3^{h-1}1s_3^{h-1}1$ and G is 3-regular, so by the induction hypothesis we get $(u, v) \to^1 (v, u_1) \to^{s_3^{h-1}} (u_1, v) \to^1 (v, u_2) \to^{s_3^{h-1}} (u_2, v) \to^1 (v, u)$, for some vertices u_1 and u_2. $\square$

Lemma 77 *Let G be an undirected 3-regular graph. Let (u, v) be an edge of G and $h_0 \geq 0$ be an integer. Let $B^{h_0}(u, v)$ be the set of vertices of G that are reachable from v by a path of length at most h_0 without using edge (v, u). Then for any $h \geq h_0$, s_3^h visits all vertices of $B^{h_0}(u, v)$ starting at (u, v).*

Proof. We prove the lemma by induction on h_0. For $h_0 = 0$, the lemma is trivial, so consider $h_0 > 0$. Let $h \geq h_0$ be arbitrary. Let u_1, u_2 be vertices so that $(u, v) \to^1 (v, u_1)$ and $(u_1, v) \to^1 (v, u_2)$. Clearly, $(u_2, v) \to^1 (v, u)$. By the previous lemma, $(u, v) \to^1 (v, u_1) \to^{s_3^{h-1}} (u_1, v) \to^1 (v, u_2) \to^{s_3^{h-1}} (u_2, v) \to^1 (v, u)$. By the induction hypothesis, the first s_3^{h-1} visits all vertices in $B^{h_0-1}(v, u_1)$, and the other s_3^{h-1} visits all vertices in

$B^{h_0-1}(v,u_2)$. Since $B^{h_0}(u,v) \subseteq B^{h_0-1}(v,u_1) \cup B^{h_0-1}(v,u_2) \cup \{v\}$, the lemma follows. $\square$

Proof of Theorem 74. Let G be any 3-regular graph with diameter at most h, and let $(u,v) \in E(G)$. Further, let u_1, u_2 be vertices, so that $(u,v) \to^1 (v,u_1)$ and $(u_1,v) \to^1 (v,u_2)$. Similarly to the previous lemma, the traversal by t_3^h goes as follows, $(u,v) \to^1 (v,u_1) \to^{s_3^{h-1}} (u_1,v) \to^1 (v,u_2) \to^{s_3^{h-1}} (u_2,v) \to^1 (v,u) \to^{s_3^{h-1}} (u,v)$. Since every vertex in G is at distance at most h from v, the claim follows by the previous lemma. $\square$

Further, we present a polynomial length universal exploration sequence for trees. Although trees are not d-regular graphs we can still talk about universal exploration sequences on trees. There is no known explicit polynomial length universal traversal sequence for trees under any reasonable definition of traversal sequences for non-regular graphs.

The universal exploration sequence for trees presented in the next theorem employs depth first search of trees (as in the standard Euler tour technique for graph algorithms).

Theorem 78 1^{2n-3} *is a universal exploration sequence for trees of size n.*

Theorems 72 and 73 in conjunction with known super-linear lower bounds on the length of universal traversal sequences for cycles and cliques suggest that exploration sequences are in some sense more powerful than traversal sequences. As we shall see in Section 5.3 universal traversal sequences can be efficiently converted into universal exploration sequences.

The universal exploration sequences constructed in this section are sequences from $\{0,1\}^*$. This may not be just a coincidence. Theorems of the following section show that we can restrict the alphabet of exploration sequences for traversal of any graph.

5.2 Unbalanced Probability Distributions on Labels

In this section we show that in contrast to traversal sequences, properties of random exploration sequences do not change dramatically with a change of the probability distribution on labels. In particular, for the randomized s-t-connectivity algorithm of Aleliunas et al. (1979) one can use a source of randomness that does not give a uniform probability distribution on labels at every step.

Let us first show that traversal sequences *are* sensitive to the choice of the probability distribution on labels. Consider a labeled graph G in Figure 5.1. It is a 3-regular undirected graph on $2n$ vertices.

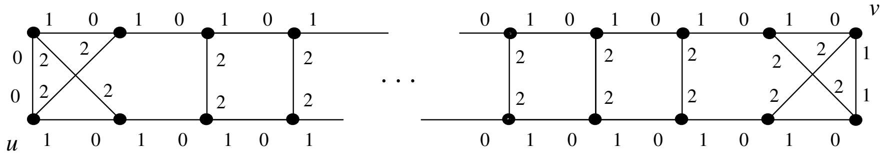

Figure 5.1: A good example.

Consider a random walk on G, where at every step of the walk the edge labeled by 0 is chosen with probability 1/2, the edge labeled by 1 is chosen with probability 1/4, and the edge labeled by 2 is chosen with probability 1/4. (This is the distribution one obtains via a naïve scheme to use a two-sided coin to generate a random number from $\{0, 1, 2\}$. I.e., using two independent uniformly distributed random bits r_i and r_{i+1} to obtain a number from $\{0, 1, 2\}$ by formula $(2r_i + r_{i+1}) \bmod 3$.) It is straightforward to verify that the stationary distribution of the random walk is not uniform. Moreover, the probability of being at vertex v is exponentially small. (If p_i denotes the probability of being at the top i-th vertex (or the bottom one, the graph is symmetric) then $p_2 = 2p_1/3$, $p_n = 2p_{n-1}/3$, and for $1 < i < n$, $p_i = p_{i-1}/2$.) Hence, the expected recurrence time of v is exponential in n. This implies that a random walk of polynomial length starting from u that uses the above defined probability distribution visits v with exponentially small probability.

Thus, the randomized log-space algorithm for s-t-connectivity, that performs a polynomial length random walk on a graph, will produce incorrect answers using the biased distribution on labels.

The following theorem asserts that this is not the case if we perform the random walk using relative labeling as in an exploration sequence.

Theorem 79 *Let $d \geq 2$ be an integer. Let $\mathcal{D}$ be a probability distribution on $\{0,\ldots,d-1\}$, such that there exist $0 \leq r < q < d$, for which $G.C.D.(q-r,d) = 1$, and r and q occur with positive probabilities under $\mathcal{D}$. Then there is a constant $c > 0$ such that for any integer $n > 1$, for every connected d-regular graph G on n vertices, and every labeling of G, the following holds.*

If $(u,v) \in E(G)$, then with probability at least $1/2$, a random walk of length cd^2n^3 starting at (u,v), where at every step of the walk the next relative label is chosen according to $\mathcal{D}$, visits all vertices of G.

Constant c depends on $\mathcal{D}$ but it does not depend directly on d. In particular, we can set $c = 3/\min\{Pr_{x\in\mathcal{D}}[x=r], Pr_{x\in\mathcal{D}}[x=q]\}$.

To establish the theorem we need the following lemma which ensures that all vertices are reachable under $\mathcal{D}$.

Lemma 80 *Let $d \geq 2$, and let integers $r < q < d$ satisfy that $G.C.D.(q-r,d) = 1$. Let G be any connected d-regular graph, and let l be its labeling. Then $\forall (u,v),(u',v') \in E(G)$, there exists $t \in \{r,q\}^{\leq |E(G)|}$ such that $(u,v) \to^t (u',v')$.*[1]

Proof. Let r, q and G be given. Fix an edge (u,v). We want to show that any other edge (u',v') is reachable by a sequence consisting of only r's and q's. (Note, if it is reachable then it is also reachable by such an exploration sequence of length at most $|E(G)|$.) For $w \in V(G)$, define the in-degree and out-degree of w as follows:

$$d_+(w) = |\{(z,w) \in E(G);\ \exists t \in \{r,q\}^*;\ (u,v) \to^t (z,w)\}|,$$

and

$$d_-(w) = |\{(w,z) \in E(G);\ \exists t \in \{r,q\}^*;\ (u,v) \to^t (w,z)\}|.$$

We claim that for any $w \in V(G)$, $d_+(w) \leq d_-(w)$. Moreover, $d_+(w) < d_-(w)$, unless $d_+(w) = d_-(w) = d$. We prove this claim. Fix $w \in V(G)$, and define the sets of incoming and outgoing edge labels:

$$L_+(w) = \{l(w,z);\ (z,w) \in E(G);\ \exists t \in \{r,q\}^*;\ (u,v) \to^t (z,w)\},$$

[1] *We consider $E(G)$ to contain directed edges, i.e., $E(G) \subseteq V(G) \times V(G)$ and $|E(G)| = d|V(G)|$.*

and

$$L_-(w) = \{l(w,z);\ (w,z) \in E(G);\ \exists t \in \{r,q\}^*;\ (u,v) \to^t (w,z)\}.$$

Observe, $d_+(w) = |L_+(w)|$, and $d_-(w) = |L_-(w)|$. Clearly, $\forall x \in L_+(w)$, $x + r \in L_-(w)$, where the arithmetic is done modulo d. Hence, $|L_+(w)| \le |L_-(w)|$, which implies that $d_+(w) \le d_-(w)$. For the second part of the claim: $|L_+(w)| = |L_-(w)| \Rightarrow \forall x \in L_+(w),\ \exists y \in L_+(w)$ such that $x + q = y + r \Rightarrow \forall x \in L_+(w),\ \exists y \in L_+(w)$ such that $y = x + q - r \Rightarrow \exists a \in \{0, \ldots, d-1\}$ such that $\{a + (q-r)b;\ b \in \{0, \ldots, d-1\}\} \subseteq L_+(w)$, where all the arithmetic is done modulo d. If $G.C.D.(q-r, d) = 1$, then $\{a + (q-r)b;\ b \in \{0, \ldots, d-1\}\} = \{0, \ldots, d-1\}$. Hence, $d_+(w) = d_-(w)$ implies that $L_+(w) = \{0, \ldots, d-1\}$ and $d_+(w) = d_-(w) = d$, which establishes the claim.

Clearly, $\sum_{w \in V} d_+(w) = \sum_{w \in V} d_-(w)$. Hence, if there were a vertex w such that $d_+(w) < d$, then there would have to be a vertex w' such that $d_+(w') > d_-(w')$, which would be a contradiction. Thus, for all $v' \in V(G)$, $d_+(v') = d$, which means that all pairs (u', v') are reachable from (u, v). $\square$

Note, the key property in the previous lemma is that for every $w \in V(G)$, $G.C.D.(q - r, \deg(w)) = 1$. Hence, if $q = r + 1$ then the lemma holds for any graph, not only for d-regular graphs.

Proof of Theorem 79. Let $\mathcal{D}, n, G, u, v$ be given. Instead of considering a traversal of G by a random simple exploration sequence, we consider a traversal of G by a random general exploration sequence, i.e., we allow also Λ to be chosen. We define a probability distribution $\mathcal{D}_\epsilon$ on $\{\Lambda, 0, \ldots, d-1\}$ as follows: $Pr_{x \in \mathcal{D}_\epsilon}[x = \Lambda] = \epsilon$, and for every $i \in \{0, \ldots, d-1\}$, $Pr_{x \in \mathcal{D}_\epsilon}[x = i] = (1-\epsilon) Pr_{x \in \mathcal{D}}[x = i]$, where we set $\epsilon = 1/3$.

Let us consider a random walk on G during which distribution $\mathcal{D}_\epsilon$ is used for choosing a label at every step of the walk. This random walk corresponds to a Markov chain with the set of states $S = E(G)$. (We assume general knowledge of Markov chains. For background see Isaacson & Madsen (1976).) The previous lemma implies that this Markov chain is irreducible, and the non-zero probability of Λ implies that the chain is aperiodic. Hence, it has a stationary distribution. It is easy to verify that the uniform distribution is stationary. Thus for every $a \in S$, the expected recurrence time

$T_{a,a} = |S| = |E(G)|$.

The previous lemma implies that there is $t \in \{r,q\}^*$ of length m that is less than or equal to $|E(G)| \cdot |V(G)|$, such that t visits all vertices in G starting at (u,v). Let $t = t_1 t_2 \cdots t_m$, where $t_1, t_2, \ldots, t_m \in \{q,r\}$. Let $a_0, a_1, \ldots, a_m \in S$ be such that $(u,v) = a_0 \to^{t_1} a_1 \to^{t_2} a_2 \cdots \to^{t_m} a_m$. We want to upper-bound the expected time T of a walk that visits $a_1, a_2, \ldots, a_m$ in this order starting from a_0 and ending at a_m. (Some a_i's may be the same.) We use the following lemma.

Lemma 81 *Let a and b be states of a finite state, irreducible, aperiodic Markov chain M. Let $p_{a,b} > 0$ be the transition probability of going from a to b. Then the expected time $T_{a,b}$ of the first visit to b starting from a satisfies: $T_{a,b} \le T_{a,a}/p_{a,b}$, where $T_{a,a}$ is the recurrence time of a.*

We give a proof of this lemma after the end of this proof.

Let $T_{a_i,a_{i+1}}$ denote the expected time of the first visit to state a_{i+1} starting from a_i. Then by linearity of expectation $T = \sum_{i=0}^{m-1} T_{a_i,a_{i+1}}$. By Lemma 81, for all $i \in \{0,\ldots,m-1\}$, $T_{a_i,a_{i+1}} \le T_{a_i,a_i}/p_{a_i,a_{i+1}}$, where $T_{a_i,a_i} = |E(G)|$. Let $\gamma_\epsilon = \min\{Pr_{x\in\mathcal{D}_\epsilon}[x = r], Pr_{x\in\mathcal{D}_\epsilon}[x = q]\}$. Clearly, $p_{a_i,a_{i+1}} \ge \gamma_\epsilon$. Hence, $T \le \gamma_\epsilon^{-1}|E(G)|m \le \gamma_\epsilon^{-1} d^2 n^3$.

Set $c = 2\gamma_\epsilon^{-1}$. By the Markov inequality, with probability at least $1/2$, a random walk starting at (u,v) of length $l = cd^2n^3$ visits all vertices of G. This is a random walk according to $\mathcal{D}_\epsilon$. Clearly, with at least the same probability, a random walk of length l according to $\mathcal{D}$ starting at (u,v) visits all vertices of G. (By removing all empty moves the event that all vertices of G are visited happens only earlier. We can also argue formally: $1/2 \le Pr_{t\in\mathcal{D}_\epsilon^l}[t$ visits all vertices of $G] = \sum_{i=0}^{l} \binom{l}{i}\epsilon^i(1-\epsilon)^{l-i} Pr_{t\in\mathcal{D}^{l-i}}[t$ visits all vertices of $G]$. Since $\sum_{i=0}^{l}\binom{l}{i}\epsilon^i(1-\epsilon)^{l-i} = 1$, the claim follows by noting that for any $j \le l$, $Pr_{t\in\mathcal{D}^j}[t$ visits all vertices of $G] \le Pr_{t\in\mathcal{D}^l}[t$ visits all vertices of $G]$.) □

Proof of Lemma 81. Let S be the set of states of Markov chain M. Let $p_{s,t}$ denote the transition probability from s to t in M, and let $T_{s,t}$ denote the expected time of the first visit to t starting from s.

Let us modify Markov chain M by adding an extra state b', i.e., $S' = S \cup \{b'\}$, and defining the transition probabilities $p'_{s,t}$ to be the same as in M but $p'_{a,b'} = p_{a,b}$,

$p'_{a,b} = 0$, $p'_{s,b'} = 0$, for all $s \in S \cup \{b'\} - \{a\}$, and $p'_{b',s} = p_{b,s}$, for all $s \in S$. Denote this new Markov chain by M'.

It is easy to verify that M' is irreducible and aperiodic, too. (We assume here that $p_{s,b} > 0$, for some $s \neq a$. If that is not the case, we use similar argument as below for $M' = M$. In our setting, it is always true that $p_{s,b} > 0$ for some $s \neq a$.)

Let π be the stationary distribution of M, and let π' be the stationary distribution of M'. It is straightforward to verify, that for all $s \in S - \{b\}$, $\pi'(s) = \pi(s)$, $\pi'(b') = p_{a,b}\pi(a)$, and $\pi'(b) = \pi(b) - p_{a,b}\pi(a)$.

Hence, the recurrence time of b' in M' satisfies $T'_{b',b'} = 1/p_{a,b}\pi(a) = T_{a,a}/p_{a,b}$. It is easy to verify that $T'_{b',b'} = T'_{a,b'} + T'_{b',a}$ thus, $T_{a,a}/p_{a,b}$ is an upper bound on $T'_{a,b'}$.

$T'_{a,b'}$ exactly corresponds to the expected time of first reaching b using the transition from a to b starting the walk from a in M. Hence, $T'_{a,b'}$ is an upper bound on $T_{a,b}$. Thus, $T_{a,b} \leq T'_{a,b'} \leq T_{a,a}/p_{a,b}$. □

Now we can use an argument of Aleliunas et al. (1979) to prove the following statement.

Corollary 82 *Let $d \geq 2$ be an integer. Let $\mathcal{D}$ be a probability distribution on $\{0, \ldots, d-1\}$ that satisfies the same conditions as in statement of Theorem 79. Then there is a constant $c > 0$ such that for any integer $n > 1$ and $l = 2cd^3n^4 \log n$, a sequence $t \in \{0, \ldots, d-1\}^l$ chosen at random according to distribution $\mathcal{D}^l$ is a universal exploration sequence for d-regular graphs on n vertices with probability at least $1 - 2^{-n}$.*

Proof. Theorem 79 implies that there is a constant $c > 0$ such that if G is an n-vertex labeled connected d-regular graph and e is an edge of G, then a random walk starting from e that is of length $(cd^2n^3) \times (2dn \log n)$ and where the next relative label is chosen according to $\mathcal{D}$, fails to visit all vertices of G with probability at most $2^{-2dn \log n}$. There are less than $n^{dn} = 2^{dn \log n}$ different d-regular graphs on n vertices including all possible labelings and the starting edge. Thus the probability that an exploration sequence of length l chosen at random according to $\mathcal{D}^l$ fails to visit all vertices in some of the graphs is at most $2^{-dn \log n} < 2^{-n}$. □

The next theorem is an extension of Theorem 79 to all graphs. The proof of this theorem is similar to the proof of Theorem 79; the proof is omitted here. (This theorem could be further generalized for r and $r+1$ to have different probabilities.)

Theorem 83 *Let r be an integer. Then there is a constant $c > 0$ such that for any integer $n > 1$, for every connected graph G on n vertices, and every labeling of G, the following holds.*

If $e \in E(G)$ and $v \in e$, then with probability at least $1/2$, a random walk of length cn^5 starting at (e, v), where at every step of the walk the next relative label is chosen uniformly at random from $\{r, r+1\}$, visits all vertices of G.

As a consequence we obtain the next claim that says that in search for a universal exploration sequence we can restrict our attention to exploration sequences consisting of only two labels for instance, labels 1 and 2.

Corollary 84 *Let r and $n > 1$ be integers. With high probability, a sequence chosen uniformly at random from $\{r, r+1\}^{O(n^7 \log n)}$ is a universal exploration sequence for graphs on n vertices.*

5.3 Traversal versus Exploration

The preceding sections suggest that universal exploration sequences might be easier to construct than universal traversal sequences. Is this really the case? In this section we indicate that this is the case. We prove here that given a universal traversal sequence, one can efficiently construct a universal exploration sequence. Hence, constructing a universal exploration sequence is at most as hard as constructing a universal traversal sequence.

This section contains several theorems that regard conversion of universal traversal sequences into universal exploration sequences and also theorems that regard conversion of universal exploration sequences for d'-regular graphs into universal exploration sequences for d-regular graphs. We start with the simple theorem.

Theorem 85 *Let $d, n \geq 3$ be integers. For any traversal sequence $t' \in \{0,1,2\}^*$ that is universal for 3-regular graphs on dn vertices there is an exploration sequence t that is universal for d-regular graphs on n vertices and the length of t is at most the length of t'. Furthermore, t can be computed from t' by TC^0 circuits.*

Proof. In order to prove the theorem we introduce the following graph transformation that will be used also in later proofs. Let G be an undirected graph and l be its labeling. We define a map F_3 that maps (G, l) to (G', l'), where G' is a 3-regular graph and l' is its labeling. Graph G' is obtained from G by replacing every vertex of degree d by a cycle of length d. More precisely, for every vertex v of G, graph G' contains vertices $v_0, v_1, \ldots, v_{\deg(v)-1}$ and for $i = 0, \ldots, \deg(v) - 1$, vertices v_i and $v_{i+1 \bmod \deg(v)}$ are connected in G'(Figure 5.2). Further, for every edge (u, v) of G, there is an edge $(v_{l(v,u)}, u_{l(u,v)})$ in G'. Labeling l' of G' is defined so that $l'(v_i, v_{i+1 \bmod \deg(v)}) = 1$, $l'(v_{i+1 \bmod \deg(v)}, v_i) = 0$ and $l'(v_{l(v,u)}, u_{l(u,v)}) = 2$.

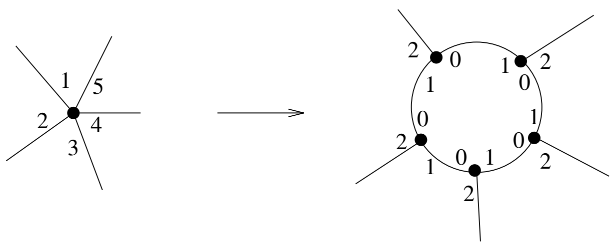

Figure 5.2: The degree reduction procedure.

We define a transformation $\sigma_d : \{0,1,2\}^* \to \{0,1,\ldots,d-1\}^*$ that transforms traversal sequences into exploration sequences. For $r \in \{0,1\}^*$ define $\mathrm{eval}_d(r) = (|\{j;\ r_j = 1\}| - |\{j;\ r_j = 0\}|) \bmod d$. Let $t' \in \{0,1,2\}^*$ be a traversal sequence and let $t'_1, t'_2, \ldots, t'_{m'} \in \{0,1\}^*$ be such that $t' = t'_1 2 t'_2 2 \cdots 2 t'_{m'}$. Define $\sigma_d(t') = \mathrm{eval}_d(t'_1)\,\mathrm{eval}_d(t'_2) \cdots \mathrm{eval}_d(t'_{m'-1})$. Notice, that $\sigma_d(t')$ can be computed from t' by TC^0 circuits.

Let t' be a universal traversal sequence for d-regular graphs on dn vertices. Set $t = \sigma_d(t')$. We claim that t is a universal exploration sequences for d-regular graphs on n vertices. Let G be any d-regular graph on n vertices and l be its labeling. Let $(G', l') = F_3(G, l)$. Consider the traversal of G by exploration sequence t starting from an edge (u,v) and the traversal of G' by t' starting from $v_{l(v,u)}$. Let $t'_1, t'_2, \ldots, t'_{m'} \in \{0,1\}^*$ be as in the definition of σ_d, i.e., $t' = t'_1 2 t'_2 2 \cdots 2 t'_{m'}$. It is straightforward to prove by induction on k that for $1 \le k < m'$, $(u,v) \to^{\mathrm{eval}_d(t'_1)\cdots \mathrm{eval}_d(t'_k)} (z,w)$ if and only if $v_{l(v,u)} \to^{t'_1 2 \cdots t'_k 2} w_{l(w,z)}$. Observe that if t' visits all vertices of G' then t visits all vertices of G. Since G' is a 3-regular graph on dn-vertices, t' visits all vertices of G' starting at any vertex. Hence, t visits all vertices of G starting at any edge. The claim follows. $\square$

We can obtain a more general result.

Theorem 86 *Let $d, n \ge 3$ and $d' \ge 4$ be integers. For any traversal sequence $t' \in \{0, 1, \ldots, d'-1\}^*$ that is universal for d'-regular graphs on $(d'+1)dn$ vertices there is an exploration sequence t that is universal for d-regular graphs on n vertices and the length of t is at most the length of t'. Furthermore, t can be computed from t' by a log-space algorithm.*

Proof. The proof is very similar to the previous one, we just need to use a different map $F_{d'}$. For a d-regular graph G and its labeling l define $F_{d'}(G,l) = (G', l')$ as follows. First construct $(G_3, l_3) = F_3(G, l)$, where F_3 is the map from the proof of Theorem 85. For every (undirected) edge (u,v) of G, replace the edge $(u_{l(u,v)}, v_{l(v,u)})$ in G_3 by the graph gadget in Figure 5.3. By doing so for all the edges of G we obtain a d-regular graph G' on $(d'+1)dn$ vertices.

The gadget in Figure 5.3 consists of vertices $u_{l(u,v)}, v_{l(v,u)}, a_0, \ldots, a_{d'-1}, b_0, \ldots, b_{d'-1}$. For $i = 2, \ldots, d'-1$, a_i and $u_{l(u,v)}$ are connected by an edge as well as b_i and $v_{l(v,u)}$, and $l'(a_i, u_{l(u,v)}) = l'(b_i, v_{l(v,u)}) = l'(u_{l(u,v)}, a_i) = l'(v_{l(v,u)}, b_i) = i$. Further, there are edges $(a_0, a_1), (b_0, b_1)$ labeled so that $l'(a_0, a_1) = l'(b_0, b_1) = 0$ and $l'(a_1, a_0) = l'(b_1, b_0) = 1$. Finally, for all $i, j \in \{0, \ldots, d'-1\}$, $i \ne j$, vertices a_i and b_j are connected by an edge and $l'(a_i, b_j) = j$ and $l'(b_j, a_i) = i$. For every vertex v of G and for every $i \in \{0, \ldots, d-1\}$, let $l'(v_i, v_{i+1 \bmod d}) = 1$ and $l'(v_{i+1 \bmod d}, v_i) = 0$.

For a sequence $r \in \{0, \ldots, d'-1\}^*$, let $\mathrm{eval}'_{d'}(r)$ be defined in terms of behavior of r in the gadget of Figure 5.3, as follows:

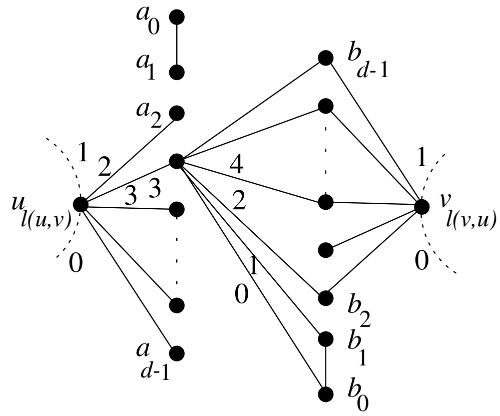

Figure 5.3: The edge replacement gadget.

$$
\begin{aligned}
\mathrm{eval}'_{d'}(r) &= 0 && \text{if } r = 0,\\
\mathrm{eval}'_{d'}(r) &= 1 && \text{if } r = 1,\\
\mathrm{eval}'_{d'}(r) &= 2 && \text{if } u_{l(u,v)} \to^r v_{l(v,u)} \text{ without leaving the gadget,}\\
\mathrm{eval}'_{d'}(r) &= \epsilon && \text{if } u_{l(u,v)} \to^r u_{l(u,v)} \text{ without leaving the gadget,}\\
\mathrm{eval}'_{d'}(r) &= \uparrow && \text{otherwise.}
\end{aligned}
$$

Let traversal sequence t' be $t'_1 t'_2 \cdots t'_{m'}$, where $t'_i \in \{0, \ldots, d'-1\}$. Define $0 = j_0 < j_1 < j_2 < \cdots \leq m'$ inductively: $j_0 = 0$ and for $i \geq 0$, let j_{i+1} be the last $j \leq m'$ such that $\mathrm{eval}'_{d'}(t'_{j_i+1} t'_{j_i+2} \cdots t'_j) \neq \uparrow$. If there is no such j let j_{i+1} be undefined. Let m be the largest i so that j_i is defined. Define a traversal sequence t_3 to be the concatenation of $\mathrm{eval}'_{d'}(t'_{j_i+1} t'_{j_i+2} \cdots t'_{j_{i+1}})$, for $i = 0, 1, \ldots, m-1$. Clearly, t_3 can be obtained from t' by a log-space algorithm. (In fact, because of the high uniformity of the graph gadget, t_3 can be obtained form t' even by TC^0 circuits.)

Sequence t_3 is in $\{0,1,2\}^*$. Let exploration sequence t be $\sigma_d(t_3)$, where the function σ_d was defined in the proof of Theorem 85. Let (u, v) be an edge of G. Similarly to the proof of Theorem 85, it can be easily verified that if traversal sequence t' visits all vertices in G starting from $v_{l(v,u)}$, then traversal sequence t_3 visits all vertices in G_3 starting from $v_{l(v,u)}$ and further, exploration sequence t visits all vertices of G starting from (u, v). Since, t is a universal traversal sequence for d'-regular graphs on $(d'+1)dn$ vertices, sequence t is a universal exploration sequence for d-regular graphs on n vertices.

□

We can generalize Theorem 85 also in the following direction.

Theorem 87 *Let $m \geq 1$ be an integer. For any traversal sequence $t' \in \{0,1,2\}^*$ that is universal for 3-regular graphs on $3m$ vertices there is an exploration sequence $t \in \{-1,0,1\}^*$ that is universal for connected graphs containing m (undirected) edges. The length of t is at most twice the length of t'. Furthermore, t can be computed from t' by AC^0 circuits.*

Proof. The proof is very similar to the proof of Theorem 85. We use a map F_3' that is similar to map F_3 in the proof of Theorem 85 and that accounts for vertices of degree one and two. Let G be a (possibly irregular) connected graph G on at least two vertices and l be its labeling. F_3' maps (G,l) to (G',l'), where G' is a 3-regular graph and l' is its labeling. Graph G' is obtained from G by replacing every vertex v of degree $d = \deg(v)$ by a cycle of length $3d$ that consists of vertices $v_0, v_1, \ldots, v_{3d-1}$. For $i = 0, \ldots, 3d-1$, vertices v_i and $v_{i+1 \bmod 3d}$ are connected in G' and $l'(v_i, v_{i+1 \bmod 3d}) = 1$, $l'(v_{i+1 \bmod 3d}, v_i) = 0$. Further, for every edge (u,v) of G, there are edges $(v_{l(v,u)}, u_{l(u,v)})$, $(v_{d+l(v,u)}, u_{\deg(u)+l(u,v)})$, $(v_{2d+l(v,u)}, u_{2\deg(u)+l(u,v)})$, in G', and $l'(v_{k\deg(v)+l(v,u)}, u_{k\deg(u)+l(u,v)}) = 2$, for $k = 0,1,2$.

Define $\mathrm{eval}(0) = -1\,0$, $\mathrm{eval}(1) = 1\,0$, $\mathrm{eval}(2) = 0$. Let $t' = t_1' t_2' \cdots t_{m'}'$, where $t_i' \in \{0,1,2\}$. Set $t = \mathrm{eval}(t_1')\,\mathrm{eval}(t_2') \cdots \mathrm{eval}(t_{m'}')$.

As in the previous proofs one can show that t is a universal exploration sequence for graphs on m (undirected) edges, given that t' is a universal traversal sequence for 3-regular graphs on $3m$ vertices. We leave details of the argument to the reader. □

One can observe that the universal exploration sequence that was obtained in the last theorem visits all the edges of any connected graph on m edges.

The ideas that are contained in the proofs of the previous theorems can be used not only for conversion of universal traversal sequences to universal exploration sequences but also for example for converting universal exploration sequences for d'-regular graphs into universal exploration sequences for d-regular graphs. In particular, the proof of Theorem 85 can be extended to a proof of the following statement.

Theorem 88 *Let $d, n \geq 3$ be integers. For any exploration sequence t' that is universal for 3-regular graphs on dn vertices there is an exploration sequence t that is universal for d-regular graphs on n vertices and the length of t is at most the length of t'. Furthermore, t can be computed from t' by a log-space algorithm.*

Similarly, the proof of Theorem 86 can be used after minor technical modifications to prove the following claim. Since the modifications are rather straightforward and technical we leave details to the reader.

Theorem 89 *Let $d, n \geq 3$ and $d' \geq 4$ be integers. For any exploration sequence t' that is universal for d'-regular graphs on $(d'+1)dn$ vertices there is an exploration sequence t that is universal for d-regular graphs on n vertices and the length of t is at most the length of t'. Furthermore, t can be computed from t' by a log-space algorithm.*

Theorem 87 could be restated in similar way.

To conclude, we would like to note that we do not know how to convert exploration sequences to traversal sequences although the proofs contained in this section have some flavor of doing so. This connection could be made formal in the following way. Let there be a black-box procedure A that on input (G, l, u, t), where G is a graph, l is its labeling, u is a vertex of G and t is a traversal sequence, outputs name of the vertex v that is reached by t starting from u. Then this black-box procedure can be used to answer similar questions for exploration sequences as follows.

Let G be a graph, l be its labeling, (u, v) be an edge of G and $t = t_1 \cdots t_m \in \mathbf{Z}^m$ be an exploration sequence. For $i \in \{1, \ldots, m\}$, define $t'_i = 1^{t_i}$ if $t_i \geq 0$ and $t'_i = 0^{-t_i}$ otherwise. Let $(G', l') = F_3(G, l)$, where F_3 is the map from the proof of Theorem 85. Then t brings us to an edge (z, w) in G starting from (u, v) if and only if the black-box procedure A outputs $w_{l(w,z)}$ on input $(G', l', v_{l(v,u)}, t'_1 2 t'_2 2 \ldots t'_m 2)$.

5.4 Self-correcting Properties

In this section we propose a candidate for a universal exploration sequence, we state some conjectures and we also show certain self-correcting properties of universal traversal and exploration sequences.

Consider the following experiment. Fix an infinite exploration sequence ω over alphabet $\mathbf{Z}$. Fix a graph G and a starting edge (u, v) in G. Pick a random labeling of graph G. A question: What is the probability p_i that by following ω starting from (u, v) we visit all vertices of G after exactly i steps?

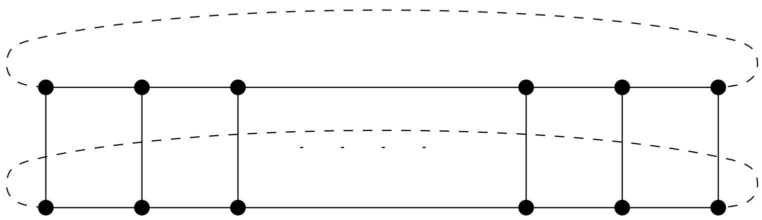

Figure 5.4: *Ladder* 34, a graph on 2×17 vertices.

Clearly, the probability p_i depends on the sequence ω and also on the graph G. Let ω be the concatenation of all strings in $\{1, 2\}^*$ in lexicographical order. Let us ask the above question about this particular sequence ω and some fixed graph. We do not know how to analyze the induced probability distribution analytically so we performed some experiments. In Figure 5.5 we plot the probability distribution $\{p_i\}_{i \in \mathbf{N}}$ for the graph in Figure 5.4.

We performed this experiment also for other graphs (e.g., the two-dimensional grid) and we estimated the probability distribution also for some large graphs by random sampling (Figure 5.6) and we consistently obtained a similar probability distribution. The distribution for a graph on n vertices is always tightly concentrated around its mean and it decays exponentially with growing i. The mean of the distribution seems to be approximately $O(n \log n)$ and the maximum non-zero p_i we estimate to be at $i \in O(n^2 \log n)$. The graphs that we have chosen for our experiments were chosen for their large diameter. Clearly, graphs of small diameter (say logarithmic) are easy to traverse.

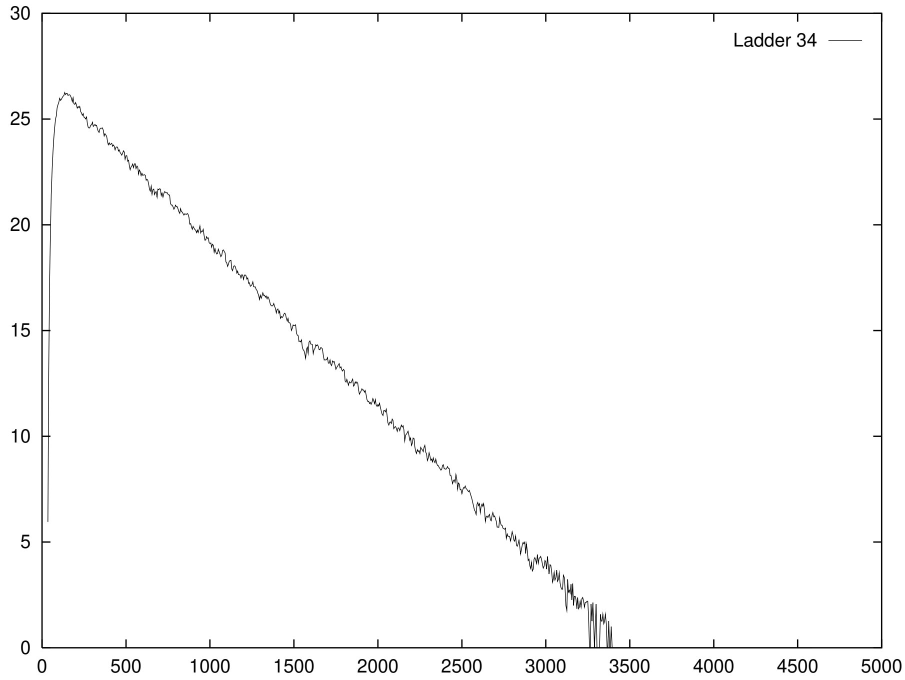

Figure 5.5: The distribution of p_i for Ladder 34. Horizontal axis: the number of steps i; vertical axis: $\log 2^{34} p_i$. The maximum i such that $p_i > 0$ is 4583 which may not be visible on the graph. (For every vertex there are two possible cyclic orderings of the incident edges hence, in total there are 2^{34} different orderings that were examined.)

We believe that our particular exploration sequence ω gives a universal exploration sequence hence, we state the following conjecture.

Conjecture 90 *Let ω be the concatenation of all the strings in $\{1,2\}^*$ in lexicographical order. There exists a polynomial $p(n)$ such that the sequence $t_n = \omega_1\omega_2\cdots\omega_{p(n)}$ is a universal exploration sequence for graphs on n vertices.*

The property of sequence ω that we see as relevant is that it does not contain any long subsequence multiple times. No sequence of length $4\log p(n)$ can appear twice in t_n.

Our experimental results suggest a tight concentration of probability distribution $\{p_i\}_{i\in\mathbf{N}}$ around its mean. However, that may not give enough insight about the behavior of the tail of the distribution and in particular, about the largest i such that $p_i > 0$ which is of our primary interest. Fortunately, the following theorem postulates that

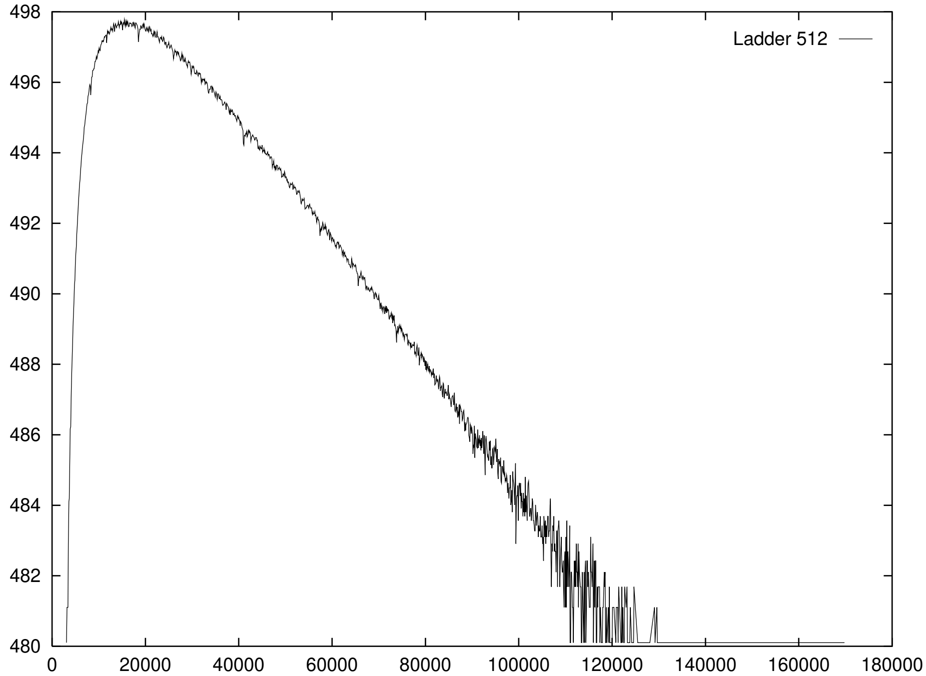

Figure 5.6: An estimate of the upper part of the distribution p_i for Ladder 512, a ladder graph on 2×256 vertices. The estimate was obtained by sampling $4 \cdot 10^7$ labelings at random. Horizontal axis: the number of steps i; vertical axis: $\log 2^{512} p_i$.

the knowledge of the tail of the distribution is not necessary. In fact it gives certain restrictions on the tail of any possible distribution. The theorem could be useful in construction of universal exploration or traversal sequences.

Theorem 91 (Self-correction) *Let ω be an infinite exploration (traversal) sequence. Let $p : \mathbf{N} \to \mathbf{N}$ be a function and $0 < \epsilon \leq 1$ be a constant. If for any $n \geq 4$, any 3-regular connected graph G on n vertices, all labelings of G except for 6^{n-n^ϵ} labelings, and any edge (u,v) of G, the sequence $\omega_1\omega_2\cdots\omega_{p(n)}$ visits all vertices of G starting from (u,v), then $\omega_1\omega_2\cdots\omega_{p(n^{1/\epsilon})}$ is a universal exploration (traversal) sequence for 3-regular graphs on n vertices.*

Proof. The proof is by contradiction. Assume that the hypothesis of the theorem is satisfied but for some n, $\omega_1\omega_2\cdots\omega_{p(n^{1/\epsilon})}$ is not a universal exploration (traversal) sequence for 3-regular connected graphs on n vertices. That means that there is a graph G on n vertices, a labeling l of G and a starting edge (u,v) so that $\omega_1\omega_2\cdots\omega_{p(n^{1/\epsilon})}$

does not visit all vertices of G starting from (u, v). Let w be a vertex of G that is not visited and such that G is still connected when w is removed. Such a vertex clearly exists. Construct a graph G' on $n^{1/\epsilon}$ vertices as follows.

Pick an arbitrary 3-regular connected graph G^f on $n^{1/\epsilon} - n + 2$ vertices. Pick any vertex w^f of G^f. Let (v_1, w), (v_2, w),(v_3, w) be edges incident to w in G and (v_1^f, w^f), (v_2^f, w^f),(v_3^f, w^f) be edges incident to w^f in G^f. Graph G' will have the set of vertices $V(G') = V(G) \cup V(G^f) - \{w, w^f\}$ and the set of edges $E(G') = E(G) \cup E(G^f) \cup \{(v_i, v_i^f), (v_i^f, v_i);\ i = 1, 2, 3\} - \{(v_i, w), (w, v_i), (v_i^f, w^f), (w^f, v_i^f);\ i = 1, 2, 3\}$. Clearly, G' is a connected 3-regular graph on $n^{1/\epsilon}$ vertices. Consider traversal of G' by $\omega_1\omega_2 \cdots \omega_{p(n^{1/\epsilon})}$ starting from (u, v) (or (v_i^f, v) if $v_i = u$). There are $6^{n^{1/\epsilon}-n+1}$ labelings of G' that are consistent with l on vertices of G. For none of these labelings does $\omega_1\omega_2 \cdots \omega_{p(n^{1/\epsilon})}$ visit all vertices of G', in contradiction to the hypothesis. $\square$

Notice, in the proof we did not put any particular restriction on the graph G^f and hence we could weaken the assumption of the theorem by excluding also a small fraction of graphs from it.

The theorem regards infinite traversal (exploration) sequences. In the usual setting we have a sequence $t_1, t_2, t_3, \ldots$ of traversal sequences, where sequence t_n is used for traversal of graphs on n vertices. For the theorem to be applicable in that setting, we can concatenate all the sequences t_n to obtain $\omega = t_1t_2t_3\cdots$. If the length of t_n is upper-bounded by some non-decreasing function $p'(n)$ then we can apply the theorem to ω and $p(n) = np'(n)$. This way we might be losing just a polynomial factor in the length of the universal traversal (exploration) sequence.

We would like to conclude this section with a lesson that we learned from our experiments. The lesson can be put in the following thesis:

The worst case examples are always simple.

As an example we can give the worst case labeling of the graph in Figure 5.4: the labeling can be concisely described by 11 0 1 00 1 0000000 1 0 1-0 111 0 1111 0 11 0 1 0 11, where each digit corresponds to the orientation of edges at a particular vertex. (From the perspective of exploration sequences there are just two possible labelings of edges at every vertex, up to an isomorphism.) This phenomenon and the thesis has a natural explanation by means of Kolmogorov complexity: the worst case examples have a very simple algorithmic description. To be precise, the lexicographically first worst case example has a small Kolmogorov complexity, given that the problem for which the example is worst case has a (simple) algorithmic description. Kolmogorov complexity is the focus of the second part of this work.

5.5 Conclusions

We have introduced a new type of traversal sequences which we call exploration sequences. We have shown that exploration sequences have more computational power than the original traversal sequences; in particular, exploration sequences can backtrack. We have also demonstrated that explicit universal exploration sequences are often easier to construct than explicit universal traversal sequences although we did not improve on the known constructions of explicit universal traversal sequences for 3-regular graphs. We have also shown an efficient conversion of universal traversal sequences into universal exploration sequences.

There are several interesting open questions we would like to point out. Any explicit construction of a universal exploration sequence gives rise to an algorithm for undirected s-t-connectivity. Is it possible to go also the other way, i.e., to derive an explicit universal exploration sequence from an algorithm for undirected s-t-connectivity? Our constructions of universal exploration sequences for cycles, expanders and trees in fact go in this direction. So would it be possible to obtain a universal exploration sequences from, for instance, the algorithm of Nisan et al. (1992)? Such a result would lead to improvements in algorithms for undirected s-t-connectivity. (Any explicit universal exploration sequence can be employed in the algorithm of Nisan et al. (1992). The shorter the sequence the less space the algorithm needs.)

In Section 5.4, Conjecture 90 we have proposed a candidate universal exploration sequence. It would be interesting to obtain some analytical results regarding the probability distribution p_i for that sequence ω. At this point we do not know how to analyze it. Also, if one cannot prove Conjecture 90, would it be possible to prove it for some larger but sub-exponential function $p(n)$? Such a result would be highly interesting. (For exponential $p(n)$ it holds trivially.) Beside these questions, the major open question is of course the $L = SL$ question.

Part II

Kolmogorov complexity

Chapter 6

Introduction

Complexity theory tries to understand the relationships among different complexity classes such as $P, NP, PSPACE, EXP$. A central role in this study is played by the notions of hardness and completeness. A complete set for a given class captures the computational power of that class. The holy grail of complexity theory is to separate complexity classes such as P and NP. In order to separate them it suffices to show that a complete set for one class does not belong to the other class.

However as Buhrman & Torenvliet (2001) note, Kolmogorov suggested not to focus the separation effort on complete sets but rather on sets of time-bounded Kolmogorov random strings that seem to lack any particular structure that is otherwise omnipresent in known complete sets. His intuition was that the structure contained in the complete sets may hinder the lower bound proof.

In our work we continue the study of sets of resource-bounded Kolmogorov random strings. We show that these sets are indeed complete for the classes in which they naturally lie under certain reductions whereas they are provably not complete under other reductions. This way we obtain natural examples of sets that exhibit this phenomenon.

Further, our results imply that under certain intractability assumptions, resource-bounded Kolmogorov complexity of a string as well as circuit size, formula size and branching program size of a Boolean function can not be efficiently approximated.

We also study unbounded Kolmogorov complexity. We show completeness results in that setting and point out certain inconsistencies caused by the usual definition of Kolmogorov complexity. We also study the problem of deterministically generating a Kolmogorov random string when the set of Kolmogorov random strings is given to us as an oracle.

Kolmogorov complexity measures the amount of randomness in a string x as the length of the shortest program that prints out string x. The tools that we use to establish some of our results are tools that were developed in the area of derandomization, where a string x is random if it is drawn at random from a particular probability distribution. Several connections between these two notions of randomness have been established previously. Here we exploit yet another one.

6.1 Focus of Our Study

We focus our study on the set $R_{\mathrm{K}} = \{x : \mathrm{K}(x) \geq |x|\}$ of strings with high traditional Kolmogorov complexity (denoted by function symbol K), as well as various resource-bounded variants R_μ for resource-bounded Kolmogorov measures $\mu = \mathrm{Kt}, \mathrm{KS}, \mathrm{KT}, \mathrm{KB}, \mathrm{KF}$. The measure Kt is Levin's time-bounded Kolmogorov complexity, KS is a variant of space-bounded Kolmogorov complexity, KT is a variant of time-bounded Kolmogorov complexity that is defined in Allender (2001) and that roughly corresponds to the minimum size of a circuit that computes the Boolean function represented by the truth-table corresponding to string x, and similarly KB and KF are resource-bounded variants of Kolmogorov complexity that correspond to the minimum size of a branching program and a formula, respectively, computing the Boolean function represented by string x. We give precise definition in Section 7.3. The choice of $|x|$ as a quantification of "high complexity" is rather arbitrary. Almost all of our results hold for any reasonable bound ranging from $|x|^\gamma$ to $|x|$, for $0 < \gamma < 1$.

There is a large body of literature on the sets of time- and space-bounded versions of Kolmogorov random strings. The majority of these results goes in the direction of proving that these sets are not complete for the class in which they lie. Buhrman & Mayordomo (1997) studied the K^t-random strings, for time bound $t(n) = 2^{n^2}$, and showed that this set is in $EXP - P$, but is *not* complete under polynomial time Turing reducibility. Allender (2001) argues that R_{Kt}, a set in EXP, is not complete under polynomial time many-one or truth-table reductions. A similar result carries over to completeness of R_{KS} for $PSPACE$.

As another example of this phenomenon, Kabanets & Cai (2000) studied the Minimum Circuit Size Problem ($MSCP$), a set in NP which is closely related to R_{KT}. They present evidence that $MCSP$ is not in P, but that it will be hard to prove that $MCSP$ is NP-complete under $\leq_m^P$ reductions. Also, Ko (1991) showed for a variant of this set that there are relativized worlds where it is neither in P nor co$-NP$-complete with respect to polynomial time Turing reductions.

These results suggest that the sets of resource-bounded random strings are not complete for the complexity class they naturally live in. Buhrman & Torenvliet (2001) gave some evidence that this is not the complete picture. They showed that for the *conditional* version of space-bounded Kolmogorov complexity the set of random strings is hard for $PSPACE$ under NP reductions. However, their result had the major drawback that it needed conditional Kolmogorov complexity, used NP reductions, and, moreover, their proof technique could not be used beyond $PSPACE$.

Using very different techniques, we provide stronger results in the same direction. For instance, we show that the set R_{Kt} of strings with high complexity using Levin's time-bounded Kolmogorov notion Kt is complete for EXP under truth-table reductions computable by polynomial size circuits. Similarly we show that the set that was shown not to be complete for EXP under poly-time Turing reduction by Buhrman & Mayordomo (1997), is complete for EXP under truth-table reductions computable by polynomial size circuits. Thus we obtain natural examples that witness the difference in power of various reducibilities.

In some instances, we are also able to provide completeness results under uniform reductions. By making use of multiple-prover interactive proofs for EXP presented by Babai, Fortnow & Lund (1991) we show that R_{Kt} is complete for EXP under NP-Turing reductions. The same result translates also to the set of Buhrman & Mayordomo thus giving a relativized world where P differs from NP.

Of even greater interest is the fact that the set R_{KS} of strings with high space-bounded Kolmogorov complexity is complete for $PSPACE$ under ZPP-Turing reductions. Our proofs rely on the existence of complete sets for $PSPACE$ that are both downward self-reducible and random self-reducible (Trevisan & Vadhan, 2002), and hence our proofs do not relativize. It remains unknown if the results themselves hold relative to all oracles.

For the unbounded case we even show that $PSPACE$ is reducible to R_{K} under *deterministic* polynomial time Turing reductions. The main tool here in addition to the previous results, is the fact that we can construct, in polynomial time, a string of high Kolmogorov complexity using R_{K} as an oracle.

When no resource bounds are present, the set of Kolmogorov random strings R_{K} is easily seen to be co$-r.e.$ and not decidable, but the complement of the halting problem is *not* reducible to R_{K} via many-one or bounded truth-table reductions. It was shown only recently by Kummer (1996) that a truth-table reduction is sufficient, although it had long been known that Turing reductions can be used (see Martin, 1966). It should be emphasized that Kummer's reduction, in fact a conjunctive truth-table reduction, is *not* feasible and asks *many* queries. We show that R_{K} is complete for $r.e.$ under truth-table reductions computable by polynomial size circuits. In contrast to Kummer's result, the size of our reduction is independent of the choice of universal Turing machine with respect to which the Kolmogorov complexity is measured.

Motivated by the result that $PSPACE$ is contained in $P^{R_{\mathrm{K}}}$ we investigate how much recursive information one can extract from R_{K} by a polynomial time reduction. We show that any set that is reducible to R_{K} via a polynomial time bounded or monotone truth-table reduction is in P or $P/poly$, respectively. On the other hand we show that any sparse enough recursive set is polynomial time disjunctive truth-table reducible to R_{K} if one picks appropriately the optimal Turing machine with respect to which Kolmogorov complexity is measured. To complement this result we also show that for any recursive time bound t there is a (sparse) recursive set and an optimal Turing machine U such that the sparse set is not reducible to R_{K} relative to U by any disjunctive truth-table reduction running in time t. These results point out a certain weakness in the usual definition of Kolmogorov complexity where we disregard the optimal Turing machine with respect to which the complexity is measured, and they raise a question about a proper universal definition of Kolmogorov complexity.

The technique we use to obtain our hardness results allow us to obtain $n^{1-\epsilon}$ non-approximability results for the Minimum Circuit Size Problem, Minimum Branching Program Size Problem and Minimum Formula Size Problem. These results hold under certain complexity intractability assumptions.

6.2 Techniques

There is an obvious connection between Kolmogorov complexity and derandomization. For any randomized algorithm $\mathcal{A}$ with a small probability of error and any input x, the coin flip sequences r that result in a wrong answer are atypical, and therefore have short descriptions of some kind.

By considering different notions of Kolmogorov complexity, less-obvious (and more-useful) connections can be exposed. Assume that the coin flip sequences that result in wrong answers have small μ-complexity, for some Kolmogorov measure μ. If we can generate a coin flip sequence r belonging to the set R_μ of strings with high μ-complexity, then we obtain an upper bound on the complexity of $\mathcal{A}$, by running the randomized algorithm on (x, r). If we can at least check membership in R_μ, we can reduce the error probability to zero by picking a coin flip sequence r at random until we get one in R_μ (of which there are many), and then running $\mathcal{A}$ on (x, r).

[The first interesting application of this approach is due to Sipser (1983). His proof that BPP lies in the polynomial time hierarchy uses $\mu = \mathrm{KD}^{poly}$, the polynomial time bounded distinguishing complexity. The corresponding set R_μ lies in co$-NP$. The hardness versus randomness trade-offs of Babai, Fortnow, Nisan & Wigderson (1993) and of Impagliazzo & Wigderson (1997) can be cast as an application with $\mu = \mathrm{KT}$, a time-bounded Kolmogorov measure introduced in Allender (2001), which essentially measures the circuit complexity of the Boolean function defined by the string described. The observation from Klivans & van Melkebeek (1999) that the construction of Impagliazzo & Wigderson (1997) relativizes with respect to any oracle A can be viewed in terms of a Kolmogorov measure which we denote as KT^A. This interpretation plays a crucial role in Trevisan's recent construction of extractors out of pseudo-random generators, Trevisan (2001). Other connections are surveyed in Allender (2001).]

Our main technique is to use relativizing hardness versus randomness trade-offs in the contrapositive.

Such results state that if there exists a computational problem in a certain complexity class $\mathcal{C}$ that is hard when given oracle access to A, then there exists a pseudo-random generator secure against A that is computable within $\mathcal{C}$. However, we argue that no pseudo-random generator computable in $\mathcal{C}$ can be secure against R_μ. Thus we conclude that every problem in $\mathcal{C}$ is easy given oracle access to R_μ, i.e., $\mathcal{C}$ reduces to R_μ. For our results, we exploit the non-uniform hardness versus randomness tradeoffs in Babai et al. (1993) and Impagliazzo & Wigderson (1997), as well as the uniform ones in Håstad, Impagliazzo, Levin & Luby (1999) and Impagliazzo & Wigderson (1998).

The organization of the rest of this part is as follows. In Chapter 8 we investigate the computational power of sets R_{Kt} and R_{KS} and in Chapter 9 we study the computational power of R_{K}. Chapter 10 contains our non-approximability results.

Chapter 7

Resource-bounded Kolmogorov Complexity

In this chapter we review necessary definitions and we define several notions of Kolmogorov complexity that we will use.

7.1 Definitions

Throughout this chapter we will refer to notions and complexity classes that were already defined in Section 3.1. In this section we recall several other classes that we did not define so far.

- ZPP denotes the class of languages accepted by zero-error probabilistic Turing machines running in expected polynomial time. Zero-error means that whenever the machine produces an output, the output is correct.

- BPP denotes the class of languages accepted by bounded-error probabilistic Turing machines running in polynomial time. I.e., $A \in BPP$ if there is a probabilistic Turing machine running in polynomial time such that if $x \in A$ then $\Pr[M \text{ accepts } x] \geq 2/3$ and if $x \notin A$ then $\Pr[M \text{ accepts } x] < 1/3$. The choice of 2/3 and 1/3 is rather arbitrary, since it is well known that the error can be made exponentially small.

- MA denotes the class of languages for which there is a one-round Merlin-Arthur protocol. Arthur-Merlin protocols are defined in Babai & Moran (1988). The class MA can be equivalently defined as follows. $A \in MA$ if there is a probabilistic Turing machine M running in polynomial time such that if $x \in A$, $|x| = n$ then there exists $y \in \{0,1\}^{n^{O(1)}}$ so that $\Pr[M(x,y) \text{ accepts}] \geq 2/3$ and if $x \notin A$, $|x| = n$

then for any $y \in \{0,1\}^{n^{O(1)}}$, $\Pr[M(x,y) \text{ accepts}] < 1/3$. Again, the error can be made exponentially small.

- $PSPACE = \bigcup_{c>0} DSPACE(n^c)$.
- $E = \bigcup_{c>0} DTIME(2^{cn})$.
- $EXP = \bigcup_{c>0} DTIME(2^{n^c})$.
- $EXPSPACE = \bigcup_{c>0} DSPACE(2^{n^c})$, $EE = \bigcup_{c>0} DTIME(2^{2^{cn}})$, $EEXP = \bigcup_{c>0} DTIME(2^{2^{n^c}})$.
- *rec* denotes the class of recursive languages.
- *r.e.* denotes the class of recursively enumerable languages.

We also recall several other definitions. We identify every string $x \in \{0,1\}^*$ with the integer i that has the binary representation $1x$. By $<.,.>$ we denote a standard pairing function that maps $\{0,1\}^* \times \{0,1\}^*$ to $\{0,1\}^*$ so that for any $x, y \in \{0,1\}^*$, $< x, y >= 0^{|x|}1xy$. For a language $A \subseteq \Sigma^*$ and any integer $n \geq 0$, $A^{=n} = A \cap \Sigma^n$ and $A^{<n} = \bigcup_{i<n} A^{=i}$. Further, we say that A *is of polynomial density* if there is a polynomial $p(n) > 0$ such that $|A^{=n}|/|\Sigma^n| \geq 1/p(n)$ for all n. If A is a set and x is an element of the universe, then $A(x) = 1$ if $x \in A$ and $A(x) = 0$ if $x \notin A$. A function $s : \mathbf{N} \rightarrow \mathbf{N}$ is *space-constructible* if there is a Turing machine that on input 1^n outputs $1^{s(n)}$ and runs in space $O(s(n))$.

Let $x \in \{0,1\}^*$ be the truth-table of a Boolean function. (Whenever we will view a string x as the truth-table of a Boolean function we will assume that the size of x is a power of two.) For a set A we define $\text{SIZE}^A(x)$ to be the smallest number of wires in any Boolean circuit consisting of bounded fan-in AND, OR, and NOT gates and unbounded fan-in oracle gates for the set A that computes the function represented by x. $\text{SIZE}(x)$ denotes $\text{SIZE}^{\emptyset}(x)$. (We choose to count the number of wires because of large oracle gates. For circuits with no oracle gates, $\text{SIZE}(x)$ corresponds (up to a constant factor) to the minimum number of gates in a circuit computing x.) For truth-table x, we also define $\text{FSIZE}(x)$ to be the minimum number of connectives in a Boolean formula that has truth-table x and that is build of binary connectives AND, OR and unary NOT. Similarly, BSIZE(x) denotes the size of the smallest branching program computing x.

For a non-negative integer i, M_i will denote the i-th Turing machine in a standard enumeration of Turing machines. In particular, we know that there is a machine M_j, such that for all i and x, $M_j(< i, x >) = M_i(x)$.

7.2 Reductions

Reductions play a central role in our investigation. There are two main types of reductions: *adaptive* and *non-adaptive.* We already defined the simplest non-adaptive reductions — many-one reductions. For completeness, here we will give the definition again.

Let $\mathcal{R}$ be a complexity class and A and B be languages.

- Non-adaptive reductions:
  - *Many-one reductions.* We say that A *$\mathcal{R}$-many-one reduces* to B ($A \leq_{\mathrm{m}}^{\mathcal{R}} B$) if there is a function $f \in \mathcal{R}$ such that for any $x \in \Sigma^*$, $x \in A$ if and only if $f(x) \in B$.
  - *Truth-table reductions.* We say that A *$\mathcal{R}$-truth-table reduces* to B ($A \leq_{\mathrm{tt}}^{\mathcal{R}} B$) if there is a pair of functions $(q, r) \in \mathcal{R}^2$ such that on an input $x \in \Sigma^*$, function q produces a list of queries $q_1, q_2, \ldots, q_m$ so that for $a_1, a_2, \ldots, a_m \in \{0, 1\}$ where $a_i = B(q_i)$, r accepts $< x, (q_1, a_1), (q_2, a_2), \cdots, (q_m, a_m) >$ if and only if $x \in A$.

    If the function r computes the conjunction of $a_1, a_2, \ldots, a_m$, then the reduction is called a *conjunctive truth-table reduction* ($\leq_{\mathrm{ctt}}^{\mathcal{R}}$). If the function r computes the disjunction of $a_1, a_2, \ldots, a_m$, then the reduction is called a *disjunctive truth-table reduction* ($\leq_{\mathrm{dtt}}^{\mathcal{R}}$). If the function r computes the parity of $a_1, a_2, \ldots, a_m$, then the reduction is called a *parity truth-table reduction* ($\leq_{\oplus\mathrm{tt}}^{\mathcal{R}}$). If the function r is monotone with respect to $a_1, a_2, \ldots, a_m$ then the reduction is called a *monotone truth-table reduction* ($\leq_{\mathrm{mtt}}^{\mathcal{R}}$). (A function r is monotone with respect to $a_1, \ldots, a_m$, if for any input x, any set of queries $q_1, \ldots, q_m$, and $a_1, \ldots, a_m, a'_1, \ldots, a'_m \in \{0,1\}$, where for all i, $a_i \leq a'_i$, if r accepts $< x, (q_1, a_1), (q_2, a_2), \cdots, (q_m, a_m) >$ then r accepts $< x, (q_1, a'_1), (q_2, a'_2), \cdots, (q_m, a'_m) >$, too.) If the number of queries m is bounded by a constant, then the reduction is called a *bounded truth-table reduction* ($\leq^{\mathcal{R}}_{\mathrm{btt}}$).

- Adaptive reductions:
  - *Turing reductions.* We say that A $\mathcal{R}$ *Turing reduces* to B ($A \leq^{\mathcal{R}}_{\mathrm{T}} B$) if there is an oracle Turing machine in class $\mathcal{R}$ that when equipped with oracle for language B accepts language A.

Observe, that any many-one reduction is a truth-table reduction, and any truth-table reduction is a Turing reduction. An example of a monotone truth-table reduction is a disjunctive or a conjunctive truth-table reduction.

Further, we say that B is *hard* for a class $\mathcal{C}$ under particular reductions if for any $A \in \mathcal{C}$, A reduces to B using that particular reductions, and if further $B \in \mathcal{C}$, then B is *complete* for $\mathcal{C}$ under that particular reductions.

7.3 Variants of Kolmogorov Complexity

In this section we review the notion of Kolmogorov complexity. The computational model that we will use throughout this part of the thesis is the multi-tape Turing machine with random-access to its tapes. Our techniques and ideas work in any general model of computation. Except for some statements in *this* section (e.g., Theorems 94 and 95), all our results hold verbatim for models such as sequential-access Turing machines.

The history of Kolmogorov complexity is rather complicated and we refer reader to the excellent book of Li & Vitanyi (1993) for a nice treatment. For a fixed Turing machine U, the *Kolmogorov complexity of a string* $x \in \{0,1\}^*$ is defined $\mathrm{K}_U(x) = \min\{|d|;\ U(d) = x\}$. A Turing machine U is said to be *optimal* (*universal*) if for any Turing machine U', there is a constant $c \geq 0$ such that for any $x \in \{0,1\}^*$, $\mathrm{K}_U(x) \leq \mathrm{K}_{U'}(x) + c$. It is well known that there are many optimal (universal) Turing machines.

For most of the applications of Kolmogorov complexity including theorems presented in this work, it is insignificant which optimal universal Turing machine is fixed. A notable exception is to be found in Section 9.1, where we show that certain properties of Kolmogorov complexity do significantly depend on the choice of the machine U. When the result is independent of the choice of the optimal Turing machine we will drop the subscript U in K_U and assume that U is fixed. For further background on Kolmogorov complexity we refer reader to the book by Li & Vitanyi (1993) (but note that the function $\mathrm{K}(x)$ is called $C(x)$ there).

Several notions of time-bounded Kolmogorov complexity have been previously introduced and studied (Sipser, 1983, Hartmanis, 1983, Ko, 1986, Buhrman, Fortnow & Laplante, 2002). We will mainly use the definition of Levin (1984) that seems to be best suited for our investigation.

Definition 92 (Levin, 1984) *Let U be a fixed Turing machine. The Kt-complexity is defined to be* $\mathrm{Kt}_U(x) = \min\{|d| + t;\ U(d) = x \text{ in at most } 2^t \text{ steps}\}$.

Time-bounded Kolmogorov complexity is more sensitive to the choice of the fixed Turing machine U because of the slow-down in simulation of one Turing machine by another one. We say that a Turing machine U is Kt-*optimal* (*universal*) if for any Turing machine U' there is a constant $c \geq 0$ so that for any $x \in \{0,1\}^*$, $\mathrm{Kt}_U(x) \leq c\mathrm{Kt}_{U'}(x) + c$. It is easy to see that there are Kt-optimal universal Turing machines. Since all of our results are independent of the particular choice of Kt-optimal universal Turing machine, we will assume that one of them is fixed and we will drop the subscript U in Kt_U.

One of the motivations for this particular definition of Kt-complexity is the following. One can define the conditional Kt-complexity by $\mathrm{Kt}(x|y) = \min\{|d| + t;\ U(d, y) = x$ in at most 2^t steps$\}$. For a string $y \in \{0,1\}^*$, the elements of $\{0,1\}^*$ can be enumerated in order of increasing $\mathrm{Kt}(x|y)$, and Levin observed that this ordering gives an essentially optimal strategy to search for an accepting computation of any nondeterministic Turing machine on input y.

Allender (2001) defines the following time-bounded Kolmogorov complexity.

Definition 93 (Allender, 2001) *Let* U *be a Turing machine. Define* $\mathrm{KT}_U(x) = \min\{|d| + t;$ *for all* $i \le |x|$, $U(d, i) = x_i$, *and* $U(d, |x| + 1) = *$, *in at most* t *steps*$\}$.

We say that a Turing machine U is KT-*optimal* (*universal*) if for any Turing machine U', there exists a constant $c \ge 0$ such that for any $x \in \{0,1\}^*$, $\mathrm{KT}_U(x) \le c\mathrm{KT}_{U'}(x) \log \mathrm{KT}_{U'}(x) + c$ and $\log\log|x| \le \mathrm{KT}(x) \le |x| + O(\log|x|)$. It is easy to see that there are KT-optimal universal Turing machines. Since our results are independent of a particular choice of the KT-optimal Turing machine U, we will assume that one of them is fixed and we will drop the subscript U in KT_U.

The difference between the KT and Kt measures lies in the weight that is given to the time bound in the two measures; the fact that the description d in KT describes the string x bit-wise is necessary in order to allow sublinear running times. The properties of the measure Kt would be essentially unchanged if the bit-wise convention were employed in that definition as well. Also observe that KT can be generalized to obtain KT^A for oracles A, by giving U access to the oracle.

When x is viewed as the truth-table of a Boolean function, $\mathrm{KT}^A(x)$ essentially measures the circuit size of x relative to A. (We assume that $|x|$ is a power of two.) Allender (2001) shows the following relationship.

Theorem 94 $\mathrm{SIZE}^A(x) \in O(\mathrm{KT}^A(x)^3)$, *and* $\mathrm{KT}^A(x) \in O((\mathrm{SIZE}^A(x))^2 \log^2|x|)$.

We are using a slightly different machine model than in Allender (2001), and thus the precise statement of the relationship is slightly different. For the first inequality, t steps of a computation of a random-access machine can be simulated by a circuit with $O(t^3)$ wires. For the second inequality, a circuit consisting of $\mathrm{SIZE}^A(x)$ wires can be described by $O(\mathrm{SIZE}^A(x) \cdot (\log \mathrm{SIZE}^A(x) + \log|x|))$ bits and there is a random-access Turing machine that evaluates it in time $O(\mathrm{SIZE}^A(x) \cdot (\log \mathrm{SIZE}^A(x) + \log|x|))$ hence, any KT-optimal Turing machine can evaluate the circuit in time $O((\mathrm{SIZE}^A(x))^2 \log^2|x|)$.

Allender (2001) shows that KT-complexity is related to the "easy witness" method of Kabanets (2001) and Impagliazzo, Kabanets & Wigderson (2002), and the "natural proofs" framework of Razborov & Rudich (1997).

The significance of the KT measure lies in the following theorem which is credited to Ronneburger in Allender (2001). It establishes a close relation between Kt-complexity and KT-complexity.

Theorem 95 (Ronneburger) *Let A be a complete set for E under linear-time reductions. Then* $\mathrm{Kt}(x) \in O(\mathrm{KT}^A(x) + \log|x|)$ *and* $\mathrm{KT}^A(x) \in O(\mathrm{Kt}(x)\log \mathrm{Kt}(x))$.

For technical reasons, our statement slightly differs from the original one. The theorem holds for any choice of Kt- and KT-optimal Turing machines; for natural choices of Kt- and KT-optimal Turing machines U and U', respectively, an even stronger statement is true: $\mathrm{Kt}_U(x) = \Theta(\mathrm{KT}^A_{U'}(x) + \log|x|)$. Thus, one can view Kt-complexity as KT-complexity relative to E.

There are several possible definitions of space-bounded Kolmogorov complexity. We define the space-bounded analog of KT-complexity as follows:

Definition 96 *Let U be a Turing machine. Define* $\mathrm{KS}_U(x) = \min\{|d| + s : U(d) = x$ *in space at most* $s\}$.

Since there are space efficient simulations among different space-bounded machines, we can define Turing machine U to be KS-*optimal* if for any Turing machine U', there is a constant $c \geq 0$ such that for any $x \in \{0,1\}^*$, $\mathrm{KS}_U(x) \leq c\mathrm{KS}_{U'}(x) + c$. Since our results do not depend on a particular choice of the KS-optimal Turing machine we will assume that one is fixed and we will drop the subscript U in KS_U.

We can state a statement similar to Theorem 95 that relates KT-complexity and KS-complexity:

Theorem 97 *Let A be a complete set for $DSPACE(n)$ under linear-time reductions. Then* $\mathrm{KS}(x) \in O(\mathrm{KT}^A(x) + \log|x|)$ *and* $\mathrm{KT}^A(x) \in O(\mathrm{KS}(x)\log \mathrm{KS}(x))$

The definition of KT-complexity is motivated in large part by the fact that $\mathrm{KT}(x)$ is a good estimate of the circuit size required to compute the function f that has x as its truth-table. But circuit size is only one of many possible interesting measures of the "complexity" of f. There is also great interest in knowing the size of the smallest branching programs computing f, as well as the size of the smallest Boolean formula representing f. We present here two more notions of resource-bounded Kolmogorov complexity that relate to these natural measures.

Definition 98 *Let U_1 be a fixed Turing machine. Define* $\mathrm{KB}_{U_1}(x) = \min\{|d| + 2^s$*; for all* $i \leq |x|$, $U_1(d,i) = x_i$, *and* $U_1(d, |x|+1) = *$, *in space at most* $s\}$.

Let U_2 be a fixed alternating Turing machine. Define $\mathrm{KF}_{U_2}(x) = \min\{|d| + 2^t;\ \forall b \in \{0,1,*\}, \forall i \leq |x|+1, U_2(d,i,b)$ *runs in time t and accepts if and only if* $x_i = b\}$. *Here, we say that $x_i = *$ if $i > |x|$.*

We say that a Turing machine U_1 is KB-*optimal* if for any Turing machine U_1', there exists a constant $c \geq 0$ such that for any $x \in \{0,1\}^*$, $\mathrm{KB}_{U_1}(x) \leq (\mathrm{KB}_{U_1'}(x))^c + c$. Further also, an alternating Turing machine U_2 is KF-*optimal* if for any alternating Turing machine U_2', there exists a constant $c \geq 0$ such that for any $x \in \{0,1\}^*$, $\mathrm{KF}_{U_2}(x) \leq (\mathrm{KF}_{U_2'}(x))^c + c$. Since there are space and time efficient simulations among deterministic and alternating Turing machines, respectively, one can easily see that KB- and KF-optimal Turing machines exist. Since our results are independent of particular choices of the KB- and KF-optimal Turing machines, we will assume that some are fixed and we will drop the subscripts U_i in KB_{U_1} and KF_{U_2}.

The following claim relates KB- and KF-complexity to branching program and formula size, respectively.

Theorem 99 *There is a constant $c > 0$ such that:*

1. $\mathrm{KB}(x) \leq (\mathrm{BSIZE}(x) + \log |x|)^c$ *and* $\mathrm{BSIZE}(x) \leq (\mathrm{KB}(x) + \log |x|)^c$,

2. $\mathrm{KF}(x) \leq (\mathrm{FSIZE}(x) + \log |x|)^c$ *and* $\mathrm{FSIZE}(x) \leq (\mathrm{KF}(x) + \log |x|)^c$.

The relation between KB and BSIZE is easy to see. The fact that $\mathrm{FSIZE}(x)$

upper-bounds $\mathrm{KF}(x)$ follows from Ruzzo (1981) and the fact that $\mathrm{KF}(x)$ upper-bounds $\mathrm{FSIZE}(x)$ follows from Buss (1993).

Our study focuses on sets of high Kolmogorov complexity. We introduce the following definition:

Definition 100 *Let $p : \mathbf{N} \to \mathbf{N}$ be a function such that for any n, $p(n) \leq n$. For any Kolmogorov complexity measure μ (such as* $\mathrm{K}, \mathrm{Kt}, \mathrm{KS}, \mathrm{KT}$*), define $R^p_\mu = \{x \in \{0,1\}^*;\ \mu(x) \geq p(|x|)\}$.*

Typically we will assume that $p(n) = n$; in such a case we will drop the subscript p in R^p_μ. For resource-bounded Kolmogorov complexity, we will restrict $p(n)$ to be polynomially bounded from below, i.e., we will assume that there exists a constant $0 < \gamma < 1$ such that $n^\gamma \leq p(n)$, for all n. As we will see in Section 8.2 (see Proposition 108), this is a rather natural requirement.

Our techniques will often require that R^p_μ be of polynomial density. For $\mu = K$, we know by a standard counting argument that for any $n > 0$ there exists $x \in \{0,1\}^n$ such that $K(x) \geq n$. Since the number of strings of length n that have unbounded Kolmogorov complexity at least n has to be Kolmogorov random by itself and of length $n - O(1)$, a constant fraction of strings of length n has Kolmogorov complexity at least n. We can state the following proposition:

Proposition 101 *Let $p : \mathbf{N} \to \mathbf{N}$ be a function such that for any n, $p(n) \leq n$. Let U be any optimal Turing machine. Then there exists a constant $c > 0$ such that for all n, $\left(R^p_{\mathrm{K}_U}\right)^{=n} \geq 2^{n-c}$.*

Similar propositions also hold for all of our resource-bounded Kolmogorov measures by simple counting arguments since in all of our resource-bounded measures we always add a non-zero quantity (resource bound) to the size of a description. Hence, sets R^p_μ are of polynomial density.

Buhrman & Mayordomo (1997) use the following notion of time-bounded Kolmogorov complexity: for a function $t : \mathbf{N} \to \mathbf{N}$ they define $\mathrm{K}^t(x) = \min\{|d|;\ U(d) = x$ in $t(|x|)$ steps $\}$ and they study the set $R^{2^{n^2}}_{\mathrm{K}} = \{x \in \{0,1\}^*;\ \mathrm{K}^{2^{n^2}}(x) \geq |x|\}$.

The sets R_μ^p are of our main interest. In Chapter 8 we investigate the computational power of sets R_{Kt} and R_{KS} and in Chapter 9 we study the computational power of R_{K}. Chapter 10 contains results on non-approximability of resource-bounded Kolmogorov complexity.

Chapter 8

Complexity of R_{Kt} and R_{KS}

In this chapter we study the hardness and completeness of sets R_{Kt} and R_{KS}. It is obvious that $R^p_{\mathrm{Kt}} \in E$, and $R^p_{\mathrm{KS}} \in DSPACE(n)$, for any space-constructible p. Allender (2001) observes that neither of these sets lie in AC^0. No other upper or lower bounds are known for these sets. As stated in the introduction, earlier results had indicated that these sets would not be complete for the complexity classes in which they reside. In particular, the following result was known.

Theorem 102 (Allender, 2001) *For any space-constructible $p(n)$ such that $n^\gamma \leq p(n) \leq n$, for some $0 < \gamma < 1$, R^p_{Kt} is not hard for EXP under $\leq^P_{\mathrm{tt}}$ reductions.*

The ideas used to prove Theorem 102 can be used to prove a similar result for R^p_{KS}. We present the argument here.

Theorem 103 *For any space-constructible $p(n)$ such that $n^\gamma \leq p(n) \leq n$, for some $0 < \gamma < 1$, R^p_{KS} is not hard for $PSPACE$ under $\leq^L_{\mathrm{T}}$ reductions.*

Proof. Note first of all that $\leq^L_{\mathrm{T}}$ reducibility coincides with $\leq^L_{\mathrm{tt}}$ reducibility, Ladner & Lynch (1976).

Let A be a subset of 0^* that is in $PSPACE$ but not in L. Suppose $A \leq^L_{\mathrm{tt}} R^p_{\mathrm{KS}}$, and let q be the function that computes the set of queries for such a reduction. Note that each of the strings y that the reduction q produces on input 0^n has $\mathrm{KS}(y) = O(\log n)$. Hence, the only strings y for which the query can receive a "yes" answer are of length $O(\log n)$, and for such queries the answer is computable directly in space $O(\log n)$. Hence all of the queries can be answered in space $O(\log n)$ and it follows that $A \in L$, contrary to our choice of A. $\square$

Improving substantially on the results of Buhrman & Torenvliet (2001), we show that R^p_{Kt} and R^p_{KS} are useful as oracles. Our main technique is to use relativizing trade-offs between hardness and randomness in the contrapositive. In particular, we will argue that an appropriate set R^p_μ of Kolmogorov random strings can be used to distinguish the output of a pseudo-random generator G_f (based on a function f) from truly random strings. This in turn will enable us to efficiently reduce f to R^p_μ.

We first describe the tools from derandomization that we will use. Then we present hardness results for R^p_{Kt} and R^p_{KS}. In the case of non-uniform reductions our technique can also be applied to many complexity classes beyond $PSPACE$ and EXP.

8.1 Tools in Derandomization

First, we recall the definition of *$PSPACE$-robustness* from Babai et al. (1993). A language A is $PSPACE$-robust if $PSPACE^A = P^A$. Clearly, the complete sets for many large complexity classes such as $PSPACE$, EXP, $EXPSPACE$, $EEXP$, etc., have this property. In general, our technique will be applicable to any complexity class possessing a $PSPACE$-robust complete set.

We will use several related constructions that build a pseudo-random generator G_f out of a function f. All these constructions are based on the paradigm of Nisan & Wigderson (1994).

Babai et al. (1993) construct, for any $\epsilon > 0$, a function (pseudo-random generator) $G_f^{\mathrm{BFNW}} : \{0,1\}^{n^\epsilon} \to \{0,1\}^n$ such that for any x of size n^ϵ, the function $G_f^{\mathrm{BFNW}}(x)$ is computable in space $O(n^\epsilon)$ given access to the Boolean function f on inputs of size at most n^ϵ.[1] Moreover, if f is $PSPACE$-robust, then there is a constant c independent of ϵ, such that each bit of $G_f^{\mathrm{BFNW}}(x)$ is computable in time $n^{\epsilon c}$ with oracle access to f. (Computing G_f^{BFNW} involves taking a multi-linear extension f^{MLE} of f (Babai et al., 1993), computing the Goldreich-Levin predicate f^{GL} of f^{MLE} (Goldreich & Levin, 1989), and xor-ing several copies of f^{GL} to get a function f^{Y} (Yao's XOR

[1] Here as well as in many other places we abuse notation in order to avoid distracting technicalities. Often we refer to quantities such as n^ϵ where integral quantity or even power of two should stand. In such cases we assume that the quantity which is stated is rounded to a near integer or power of two as appropriate.

Lemma, Yao, 1982, see also Levin, 1987 and Goldreich, 1995), that is then employed in Nisan-Wigderson generator (Nisan & Wigderson, 1994).) The following hardness versus randomness trade-off holds and is implicit in Babai et al. (1993) and Klivans & van Melkebeek (1999).

Theorem 104 (Babai et al., 1993, Klivans & van Melkebeek, 1999) *Let $\epsilon > 0$ be any constant. There exists a polynomial $s(n)$ such that for any Boolean function f, any set B and any polynomial $q(n)$, if the pseudo-random generator $G_f^{\mathrm{BFNW}} : \{0,1\}^{n^\epsilon} \to \{0,1\}^n$, that is described above, satisfies $|Pr_{r \in U_n}[r \in B] - Pr_{x \in U_{n^\epsilon}}[G_f^{\mathrm{BFNW}}(x) \in B]| \geq 1/q(n)$ for all large n, then there exists an oracle circuit family $\{C_n\}_{n \in \mathbf{N}}$, with oracle B, that computes f and queries B non-adaptively, and for all large enough n circuit C_n is of size $s(n)$.*

The oracle circuit family that computes f can be constructed in $PSPACE$ given oracle access to f on inputs of length n and to B on strings of size n^c, where the constant c depends only on ϵ.

Impagliazzo & Wigderson (1998) reexamine the approach of Babai et al. (1993). For any $\epsilon > 0$, their pseudo-random generator $G_f^{\mathrm{IW98}} : \{0,1\}^{n^\epsilon} \to \{0,1\}^n$ is also computable in space $O(n^\epsilon)$ with oracle access to f on inputs of size at most n^ϵ. It satisfies the following uniform version of Theorem 104.

Theorem 105 (Impagliazzo & Wigderson, 1998) *Let $f : \{0,1\}^* \to \{0,1\}^*$ be a random and downward self-reducible function, $\epsilon > 0$, and $G_f^{\mathrm{IW98}} : \{0,1\}^{n^\epsilon} \to \{0,1\}^n$ be the pseudo-random generator described above. Let B be a set and $q(n)$ be a polynomial. If $|\Pr_{r \in U_n}[r \in B] - \Pr_{x \in U_{n^\epsilon}}[G_f^{\mathrm{IW98}}(x) \in B]| \geq 1/q(n)$, for all n large enough, then there exists a probabilistic, polynomial time Turing machine with oracle B that on input x outputs $f(x)$ with probability at least $2/3$.*

The preceding two theorems provide the key derandomization techniques that are needed to prove our completeness results. They are stated in the contrapositive of their original formulations since that is the way we will use them.

8.2 Hardness Results

The following theorem is one of our main results.

Theorem 106 *Let p be a space-constructible function such that $n^\gamma \leq p(n) \leq n$, for some $0 < \gamma < 1$ and all $n > 0$. Then:*

- *R^p_{Kt} is complete for EXP under $\leq^{P/poly}_{\mathrm{tt}}$ reductions,*
- *R^p_{KS} is complete for $PSPACE$ under $\leq^{P/poly}_{\mathrm{tt}}$ reductions.*

In order to prove the theorem we will need the following lemma which illustrates our proof technique.

Lemma 107 *Let A be any $PSPACE$-robust set. Let B be a set of polynomial density such that for every $x \in B$, $\mathrm{KT}^A(x) > |x|^\gamma$, for some constant $0 < \gamma < 1$. Then $A \leq^{P/poly}_{\mathrm{tt}} B$.*

Proof. Let f be the characteristic function of A. Consider $G^{\mathrm{BFNW}}_f : \{0,1\}^{n^\epsilon} \to \{0,1\}^n$, where we choose ϵ as follows. We know that for a string x of length n^ϵ, every bit of $G^{\mathrm{BFNW}}_f(x)$ is computable with access to A in time $n^{\epsilon c}$ for some constant c that is independent of ϵ. We may assume that $c > 1$, so set $\epsilon = \gamma/6c$. We claim that any string in the range of G^{BFNW}_f has small KT^A-complexity. Let $y = G^{\mathrm{BFNW}}_f(x)$, for some $x \in \{0,1\}^{n^\epsilon}$. On input x, every bit of $G^{\mathrm{BFNW}}_f(x)$ is computable in time $n^{\gamma/6}$ with access to oracle A. Hence, $\mathrm{KT}^A(y) \leq 2|x| + O(n^{\gamma/2}) + O(1) \leq O(n^{\gamma/2})$. It follows that B distinguishes the output of G^{BFNW}_f from truly random strings, so by Theorem 104, f is $\leq^{P/poly}_{\mathrm{tt}}$ reducible to B. □

Proof of Theorem 106. We already stated that $R^p_{\mathrm{Kt}} \in EXP$. For the hardness result, pick a set A that is complete for E under linear-time reductions. Since there exists γ, such that $\mathrm{Kt}(x) \geq |x|^\gamma$ for all $x \in R^p_{\mathrm{Kt}}$, by Theorem 95 we have that $\mathrm{KT}^A(x) \geq \Omega(|x|^\gamma)$. Lemma 107 implies that $A \leq^{P/poly}_{\mathrm{tt}} R^p_{\mathrm{Kt}}$, so $EXP \leq^{P/poly}_{\mathrm{tt}} R^p_{\mathrm{Kt}}$. The proof for $PSPACE$ is similar. □

The polynomial size circuits that compute the reduction of EXP to R^p_{Kt} can be constructed in EXP. Similarly, the circuits that compute the reduction of $PSPACE$

to R^p_{KS} can be constructed in $PSPACE$. The hardness results apply not only to R^p_{Kt} and R^p_{KS}, but to any polynomially dense set containing no strings of low resource-bounded Kolmogorov complexity (including the Buhrman-Mayordomo set and R^p_{K}).

We restrict $p(n)$ in Theorem 106 to be at least polynomial. We do not know if the theorem holds for sub-polynomial $p(n)$. The following proposition implies, though, that it is rather unlikely that the theorem is true for sub-polynomial $p(n)$.

Proposition 108 *Let $2\log n \leq p(n) \leq n$ be a non-decreasing function. Let μ be a Kolmogorov measure and A be a set. If A is computable by oracle circuits of size $O(n^c)$ with oracle R^p_μ, then A is computable by circuits of size $2^{O(p(n^c))}$ without any oracle.*

To see this proposition, observe that $|(\overline{R^p_\mu})^{=n}| < 2^{p(n)}$. Hence, $(\overline{R^p_\mu})^{<O(n^c)}$ can be encoded into a table of size $2^{O(p(n^c))}$ and the claim immediately follows. A similar observation can be made for other resource-bounded reductions. Using this proposition, if $EXP \leq_{\mathrm{tt}}^{P/poly} R^p_{\mathrm{Kt}}$, where $p(n)$ is sub-polynomial, then EXP is computable by sub-exponential size circuits. A commonly held belief is that EXP cannot be computed by sub-exponential size circuits so proving Theorem 106 for sub-polynomial $p(n)$ would be rather surprising.

We have already obtained hardness results for R^p_{Kt} and R^p_{KS} under non-uniform reductions. We also know that for certain uniform reductions R^p_{Kt} and R^p_{KS} are not hard for EXP and $PSPACE$, respectively. Despite that, next we will focus on obtaining certain uniform hardness results. In particular, we will show the following:

Theorem 109 *Let p be a function such that $n^\gamma \leq p(n) \leq n$, for some $0 < \gamma < 1$ and all $n > 0$. Then:*

- $EXP \leq_{\mathrm{T}}^{NP} R^p_{\mathrm{Kt}}$.
- $PSPACE \leq_{\mathrm{T}}^{ZPP} R^p_{\mathrm{KS}}$.

As an immediate corollary we will get:

Corollary 110 *Let p be a space-constructible function such that $n^\gamma \leq p(n) \leq n$, for some $0 < \gamma < 1$ and all $n > 0$. Then:*

- $EXP = NP^{R^p_{\mathrm{Kt}}}$.
- $PSPACE = ZPP^{R^p_{\mathrm{KS}}}$.

In the case of unbounded Kolmogorov complexity, we can state the following claim.

Theorem 111 *Let p be a computable function such that $p(n+1) \leq p(n) + O(\log n)$ and $n^\gamma \leq p(n) \leq n$, for some $0 < \gamma < 1$ and all $n > 0$. Then $PSPACE \subseteq P^{R^p_{\mathrm{K}}}$.*

Before going into proofs of these claims let us recall a result of Buhrman and Mayordomo:

Theorem 112 (Buhrman & Mayordomo, 1997) *$R_{\mathrm{K}}^{2^{n^2}}$ is in $EXP - P$ and it is not hard for EXP under polynomial time Turing reductions.*

As a corollary to the proof of Theorem 109 and to Lemma 107 we get:

Theorem 113 *$R_{\mathrm{K}}^{2^{n^2}}$ is complete for EXP under NP-Turing reductions and under $P/poly$ truth-table reductions.*

Last two theorems give us a relativized world where $P \neq NP$. It is already known that there are oracles relative to which P differs from NP. Such an oracle was first constructed by Baker, Gill & Solovay (1975). In fact, Bennett & Gill (1981) show that relative to a random oracle, P differs from NP with probability 1. Explicit constructions of oracles that separate P from NP typically proceed by diagonalization. Here we obtain a natural explicit oracle separating P from NP:

Corollary 114 $P^{R_{\mathrm{K}}^{2^{n^2}}} \neq NP^{R_{\mathrm{K}}^{2^{n^2}}}$.

We will present proofs of the above claims now. In order to prove Theorem 109, we will show somewhat weaker results first that will be derandomized later.

Lemma 115 *Let B be a set of polynomial density such that for every $x \in B$, $\mathrm{Kt}(x) > |x|^\gamma$, for some constant $0 < \gamma < 1$. Then $EXP \subseteq MA^B$.*

Proof. Babai et al. (1993) show that if EXP has polynomial size circuits then $EXP \subseteq MA$. This result relativizes as follows: if EXP has polynomial size oracle circuits with oracle B then $EXP \subseteq MA^B$. To see this, let $A \in EXP$ and $EXP \subseteq P^B/poly$. By Babai et al. (1991), A is accepted by a 2-prover interactive proof system with provers computable in EXP. (See Babai et al. (1991) for the definition of multi-prover interactive proof systems.) The relativized Merlin-Arthur protocol for A is as follows. Merlin sends Arthur the poly-size circuits that, when given access to the oracle B, compute the answers given by the two provers for protocol of A. Arthur then executes the multi-prover interactive protocol, simulating the provers' answers by executing the circuits and querying the oracle B. By Theorem 106, $EXP \subseteq P^B/poly$ hence, $EXP \subseteq MA^B$. $\square$

To prove a uniform hardness result for $PSPACE$, as stated in Theorem 109, we will use Theorem 105 and the following theorem of Trevisan & Vadhan (2002).

Theorem 116 (Trevisan & Vadhan, 2002) *There is a problem in $DSPACE(n)$ that is hard for $PSPACE$, random self-reducible, and downward self-reducible.*

We will need the following result:

Lemma 117 *Let B be a set of polynomial density such that for every $x \in B$, $\mathrm{KS}(x) > |x|^\gamma$, for some constant $0 < \gamma < 1$. Then $PSPACE \subseteq BPP^B$.*

Proof. Let $f \in DSPACE(n)$ be a function that is downward and random self-reducible and that is hard for $PSPACE$, as guaranteed by Theorem 116. Consider $G_f^{\mathrm{IW98}} : \{0,1\}^{n^{\gamma/2}} \to \{0,1\}^n$. From the properties of G_f^{IW98} it follows that all the strings in the range of G_f^{IW98} have small space-bounded Kolmogorov complexity, in particular, $\mathrm{KS}(y) \leq O(|y|^{\gamma/2})$, for any $y = G_f^{\mathrm{IW98}}(z)$, for some $z \in \{0,1\}^{n^{\gamma/2}}$. Hence, B distinguishes the output of G_f^{IW98} from random strings and by Theorem 105, there is a probabilistic procedure with oracle B that on input x outputs $f(x)$ with probability at least 2/3. $\square$

In the next section we will build necessary derandomization tools to finish proofs of Theorems 109 and 111.

8.3 Derandomization

It was observed many times before (e.g., Sipser, 1983) that full derandomization is possible if one has access to a Kolmogorov random string. If one has access to an oracle that distinguishes Kolmogorov random strings from non-random then one can achieve at least limited derandomization. In this section we will show several results in these directions. We will show that having oracle access to R^p_μ allows us to derandomize MA to NP and BPP to ZPP. In the case of access to oracle R^p_{K} of strings that are random with respect to unbounded Kolmogorov complexity one can fully derandomize BPP computation. The crucial ingredient in this latter result is the ability to efficiently construct a Kolmogorov random string using access to R^p_{K}.

We will use the following derandomization result of Impagliazzo & Wigderson (1997) (see also Sudan, Trevisan & Vadhan, 2001) which is a strengthening of the results of Babai et al. (1993). Given access to a Boolean function f, Impagliazzo & Wigderson (1997) construct a pseudo-random generator $G_f^{\mathrm{IW97}} : \{0,1\}^{O(\log n)} \to \{0,1\}^n$ with the following property (see also Klivans & van Melkebeek, 1999):

Theorem 118 (Klivans & van Melkebeek, 1999) *For any $\epsilon > 0$, there exist constants $c, c' > 0$ such that the following holds. Let A be a set and $n > 1$ be an integer. Let $f : \{0,1\}^{c \log n} \to \{0,1\}$ be a Boolean function that cannot be computed by oracle circuits of size $n^{c\epsilon}$ with oracle A. Then $G_f^{\mathrm{IW97}} : \{0,1\}^{c' \log n} \to \{0,1\}^n$ satisfies: $|\Pr_{r \in U_n}[C^A(r) = 1] - \Pr_{x \in U_{c' \log n}}[C^A(G_f^{\mathrm{IW97}}(x)) = 1]| < 1/n$, for any oracle circuit C^A of size at most n.*

For x of size $c' \log n$, $G_f^{\mathrm{IW97}}(x)$ is computable in time polynomial in n given access to f on inputs of length $c \log n$. Using this tool we will prove the following lemma.

Lemma 119 *Let A be any oracle and B be a set such that $B \in P^A/poly$ and for every $x \in B$, $\mathrm{KT}^A(x) > |x|^\gamma$, for some constant $0 < \gamma < 1$.*

1. *If B contains a string of every length then $MA^B \subseteq NP^B$.*

2. *If B is of polynomial density then $BPP^B \subseteq ZPP^B$.*

Proof. 1. Let J be a language in MA^B accepted by an MA oracle machine N_0^B that on input x of length n, nondeterministically guesses a string $y \in \{0,1\}^{n^{O(1)}}$ and then runs on (x, y) in probabilistic time n^{c_0}. We will construct a nondeterministic oracle machine N_1^B accepting J.

First, N_1^B will nondeterministically construct a pseudo-random generator G_f^{IW97} that allows us to deterministically estimate the acceptance probability of any oracle circuit of size n^{2c_0}. Then, N_1^B will guess a string $y \in \{0,1\}^{n^{O(1)}}$ and using the pseudo-random generator, it will deterministically estimate the acceptance probability of N_0^B on input x with nondeterministic guess y. Since $N_0^B(x, y)$ runs in time n^{c_0}, it can be simulated by a Boolean circuit of size n^{2c_0} and thus the constructed pseudo-random generator is applicable.

Machine N_1^B constructs the pseudo-random generator as follows. Let there be an oracle circuit family $\{C_n^A\}_n$ of size n^k computing B. Let $\epsilon = \gamma/6k$ and let c, c' be the corresponding constants from Theorem 118. N_1^B guesses a string $z \in B$ of length m, for $m = (n^{2c_0})^c$, and interprets it as the truth-table of a Boolean function f on inputs of size $\log m$. Since $z \in B$, $\mathrm{KT}^A(z) \geq m^\gamma$. Thus by Theorem 94, f requires circuits of size at least $\Omega(m^{\gamma/3})$ even when the circuit has access to the oracle A. If, instead of A, the circuit has access to B, the size required to compute f is at least $\Omega(m^{\gamma/6k}) \geq \Omega((n^{2c_0})^{c\epsilon})$.

By Theorem 118, the function f can be used to construct the generator G_f^{IW97} that stretches $O(\log n)$ random bits into n^{2c_0} pseudo-random bits that are indistinguishable from random by any oracle circuit of size n^{2c_0} with access to B.

2. We use a similar argument as above. Here we will be randomly choosing candidates for z, until we find one which is in B. Since B is of polynomial density we will find z that belongs to B in expected polynomial time. The string z will be then used to construct a pseudo-random generator for derandomizing the BPP^B computation. □

Now we have sufficient tools to prove Theorem 109.

Proof of Theorem 109. By Lemma 115, $EXP \subseteq MA^{R^p_{\mathrm{Kt}}}$. Let A be a set complete for E under linear-time reductions. Then $R^p_{\mathrm{Kt}} \in P^A/poly$. By properties of set R^p_{Kt} and by

Theorem 95, there exists $0 < \gamma < 1$ such that for any $x \in R^p_{\mathrm{Kt}}$, $\mathrm{KT}^A(x) > n^\gamma$. Thus we can apply Lemma 119 to obtain $EXP \subseteq NP^{R^p_{\mathrm{Kt}}}$. The result for $PSPACE$ is proved in similar way using Lemma 117 instead of Lemma 115. $\Box$

We would obtain a *deterministic* polynomial time reduction of $PSPACE$ to R^p_{KS} in Theorem 109, if there was a deterministic method to construct a string of high KS-complexity using R^p_{KS} as an oracle. We do not know if this is possible. On the other hand, if one considers Kolmogorov complexity without resource bounds, it is possible to achieve this. Buhrman, Fortnow, Newman & Vereshchagin (2003) present a construction that incrementally builds a Kolmogorov random string bit-by-bit using short walks on expanders and access to oracle R^p_{K}. They prove the following theorem.

Theorem 120 (Buhrman et al., 2003) *Let $2\log n \leq p(n) \leq n$ be a function such that $p(n+1) \leq p(n) + O(\log n)$. Then there is a deterministic polynomial time algorithm that with oracle access to R^p_{K} on input 0^n produces string $x \in R^p_{\mathrm{K}}$ of length n.*

Using the idea of incremental build up we can prove a weaker statement without use of expanders.

Theorem 121 *Let $2\log n \leq p(n) \leq n$ be a function such that for some $0 < \delta < 1$, $p(n + \lceil \log n \rceil) \leq p(n) + \delta \log n$, for all n large enough. Then there is a deterministic polynomial time algorithm that with oracle access to R^p_{K} on input 0^n produces string $x \in R^p_{\mathrm{K}}$ of length between n and $n + O(\log n)$.*

Proof. First, one can easily prove by induction on k that for any integer $k \geq 0$ and any n large enough, $p(n + k\lceil \log n \rceil) \leq p(n) + k\frac{1+\delta}{2}\log n$. We will use the following property referred to as symmetry of information in Section 2.8 of the book by Li & Vitanyi (1993): There exists a constant c_1 such that for any strings x and y, $\mathrm{K}(xy) \geq \mathrm{K}(x) + \mathrm{K}(y|x) - c_1 \log |xy|$. We may assume without loss of generality that c_1 is such that $k = 4c_1/(1-\delta)$ is an integer. Assume that n is large enough.

We build x inductively. We start by finding $x \in R^p_{\mathrm{K}}$ of length $\log n$ by exhaustive search. Now, assume (inductively) that $\mathrm{K}(x) \geq p(|x|)$. Try all strings y of length $k\lceil \log |x| \rceil$, and use the oracle R^p_{K} to see if $\mathrm{K}(xy) \geq p(|xy|)$. We are guaranteed to find such a y, since $\mathrm{K}(y|x) \geq |y|$ holds for most y, and for any such y, $\mathrm{K}(xy) \geq \mathrm{K}(x)+\mathrm{K}(y|x)-c_1 \log|xy| \geq p(|x|)+|y|-2c_1\log|x| \geq p(|x|)+\frac{4c_1}{1-\delta}\log|x|-2c_1\log|x| = p(|x|)+2c_1\frac{1+\delta}{1-\delta}\log x \geq p(|x|+k\lceil\log x\rceil) = p(|xy|)$. Set x to be xy and continue this process until x is of length between n and $n+k\lceil\log n\rceil$. Clearly, this requires at most polynomial time in n. $\square$

Our lemma is applicable, for instance, for any non-decreasing polynomial $p(n)$ that has its first derivative bounded away from one by a constant at every positive (real) point n and that satisfies for some $0<\gamma<1$ that $n^\gamma \leq p(n) \leq \gamma n$. Our theorem does not guarantee that we will obtain a Kolmogorov random string of arbitrary length. However, in typical setting if we need a string of length n of high Kolmogorov complexity, we can use our construction to obtain a string x' of length between n and $n+O(\log n)$ and cut the last $|x'|-n$ bits from x'. This way we obtain a string x such that $K(x) \geq K(x')-O(\log n)$, which will be sufficient in at least our applications.

Theorem 122 *Let $p(n)$ be a computable function such that for some $0<\gamma<1$, $n^\gamma \leq p(n) \leq n$ and $p(n+1) \leq p(n)+O(\log n)$. Then $BPP^{R^p_{\mathrm{K}}} = P^{R^p_{\mathrm{K}}}$.*

Proof. We will derandomize $BPP^{R^p_{\mathrm{K}}}$ computation using a pseudo-random generator G_f^{IW97} that can be used to derandomize oracle circuits of size $m=n^{O(1)}$ with oracle R^p_{K}.

Let c be the constant from Theorem 118 for $\epsilon=\gamma/2$. We claim that a string z of length m^c such that $K(z) \geq m^{c\gamma}$ represents the truth-table of a Boolean function f that cannot be computed by oracle circuits of size $m^{c\gamma/2}$ with oracle R^p_{K}.

Any function $f:\{0,1\}^{c\log m} \to \{0,1\}$ that is computable by a circuit C of size $m^{c\gamma/2}$ with oracle R^p_{K} has Kolmogorov complexity at most $O(m^{c\gamma/2}\log m)$. (We define the Kolmogorov complexity of f to be the Kolmogorov complexity of its truth-table.) This is because a circuit of size at most $m^{c\gamma/2}$ makes oracle queries of size at most $m^{c\gamma/2}$. Since R^p_{K} is a co-recursively enumerable set, to compute the characteristic function of R^p_{K} for all strings of size at most $m^{c\gamma/2}$, we only need to know how many of these strings are not in R^p_{K} (since then we can carry out the enumeration of $\overline{R^p_{\mathrm{K}}}$ until all have appeared). This can be specified by $m^{c\gamma/2}+1$ bits. Hence, f can be fully described by these $m^{c\gamma/2}+1$ bits plus the description of the circuit C plus some constant. Thus, $\mathrm{K}(f) \leq O(m^{c\gamma/2} \log m)$.

By Theorem 120 or 121 in polynomial time we can find a string z of length m^c such that $K(z) \geq O(m^{c\gamma})$. The function f represented by z can be used to construct a pseudo-random generator G_f^{IW97} that in turn can be used to deterministically estimate the acceptance probability of the $BPP^{R_{\mathrm{K}}^p}$ computation. Hence, $BPP^{R_{\mathrm{K}}^p} = P^{R_{\mathrm{K}}^p}$. $\square$

Using Lemma 117 (for $B = R_{\mathrm{K}}^p$) and Theorem 122, we obtain Theorem 111.

Chapter 9

Hardness of R_{K}

In this chapter we study the set R_{K}^{p} of Kolmogorov random strings. We will always assume that p is a computable function such that $2\log n \leq p(n) \leq n$. (So there are random and non-random strings of every length.) Clearly, R_{K}^{p} is a co$-r.e.$ set. Martin (1966) presented a proof that any *simple* set (an infinite co$-r.e.$ set that has no infinite $r.e.$ subset) is Turing complete for the class co$-r.e.$. Since R_{K}^{p} is easily seen to be simple, Martin's result implies that R_{K}^{p} is Turing complete for co$-r.e.$. For completeness, we give here a simple proof of completeness of R_{K}^{p}. Let the Halting problem be $H = \{x;\ M_x(x) \text{ halts}\}$. It is well known that H is complete for $r.e.$ under $\leq_m^P$ reductions.

Theorem 123 (Martin, 1966) *Let $2\log n \leq p(n) \leq n$ be any computable function. Then $H \leq_T^{rec} R_{\mathrm{K}}^{p}$.*

Proof. (*Folklore*) Let $x \in \{0,1\}^*$ be a string for which we want to decide membership in H. Let m be the smallest integer such that $2|x| \leq p(m)$. Since $p(m)$ is unbounded and computable such an integer exists and can be easily found. Using oracle access to R_{K}^{p} we can also find for all $y \in (\overline{R_{\mathrm{K}}^{p}})^{=m}$, a string y such that $|d_y| \leq p(m)$ and $U(d_y) = y$. Let t_y be the running time of $U(d_y)$ and let $t = \max\{t_y;\ y \in (\overline{R_{\mathrm{K}}^{p}})^{=m}\}$. We claim that if $M_x(x)$ halts then it halts in less than t steps. That will imply the theorem. If $M_x(x)$ were to halt in more than t steps, then the running time of $M_x(x)$ could be used as an upper bound on the running time of all descriptions of length at most $p(m)$ to find all strings in $(\overline{R_{\mathrm{K}}^{p}})^{=m}$. Hence the lexicographically first string of length m that does not belong to $(\overline{R_{\mathrm{K}}^{p}})^{=m}$ would have description of length at most $|x| + O(1)$ and that would be a contradiction. $\square$

The procedure described in the above proof may not halt if a set different than R^p_{K} is used as an oracle. The same is true for Martin's original reduction. Kummer (1996) gives a conjunctive truth-table reduction of the Halting problem to R^n_{K}. On input x, Kummer's recursive reduction produces a set of strings S_x with the property that $S_x \subseteq \overline{R^n_{\mathrm{K}}}$ if and only if $x \in H$.

Theorem 124 (Kummer, 1996) $H \leq^{rec}_{ctt} \overline{R^n_{\mathrm{K}}}$.

Kummer's proof strategy gives a reduction of H to R^p_{K}, for any computable $2\log n \leq p(n) \leq n$. The running time of Kummer's reduction depends on the fixed optimal Turing machine with respect to which Kolmogorov complexity is measured. In fact, our Corollary 130 shows that for any recursive time bound $t(n)$ there is an optimal Turing machine U so that Kummer's reduction of H to $\overline{R^n_{\mathrm{K}_U}}$ must run in time more than $t(n)$.

In contrast, we present the following theorem.

Theorem 125 *Let p be a function such that $n^\gamma \leq p(n) \leq n$, for some γ. Let U be any optimal Turing machine. Then $H \leq^{P/poly}_{tt} \overline{R^p_{\mathrm{K}_U}}$. Moreover the size of the reduction does not depend on U (at least for all large enough inputs).*

Given γ and U, the reduction can be constructed in P^H. (Since $\overline{R^p_{\mathrm{K}_U}}$ is polynomial time many-one reducible to H, the construction of the circuits that compute the reduction can be done in $PSPACE^H$, where the $PSPACE$ computation asks only polynomial size queries to H. Such a computation can be simulated in P^H.)

Our result indeed holds for any subset of $R^p_{\mathrm{K}_U}$ of polynomial density. Since our result regards resource-bounded reductions we limit $p(n)$ to be lower-bounded by a polynomial in the theorem. By Proposition 108, the theorem is false for sub-polynomial $p(n)$. If there were a $P/poly$ reduction of H to $R^p_{\mathrm{K}_U}$, where $p(n)$ is sub-polynomial, then H would be computable by sub-exponential size circuits. It is well known (and one can show by a simple diagonalization) that H cannot be computed by sub-exponential circuits.

Proof of Theorem 125. It suffices to show that the characteristic function h of the Halting problem is computable in $P^{R^p_{\mathrm{K}_U}}/poly$ by circuits asking non-adaptive queries.

Consider $G_h^{\mathrm{BFNW}} : \{0,1\}^{n^{\gamma/2}} \rightarrow \{0,1\}^n$. We claim that any string in the range of G_h^{BFNW} has small Kolmogorov complexity. Let $y = G_h^{\mathrm{BFNW}}(x)$, for some $x \in \{0,1\}^{n^{\gamma/2}}$. To compute y given x, we need to query h on inputs of size at most $n^{\gamma/2}$. If we know the *number* l of strings z of length at most $n^{\gamma/2}$ on which $h(z) = 1$, then we can determine exactly which strings z these are (by carrying out the enumeration of H until all have appeared). This number l is less than $2^{n^{\gamma/2}+1}$ and hence can be specified with $O(n^{\gamma/2})$ bits. String y can be fully described by giving x, l, and programs for h and G_h^{BFNW}, thus $\mathrm{K}_U(y) \leq O(n^{\gamma/2})$. It follows that $R^p_{\mathrm{K}_U}$ distinguishes the output of G_h^{BFNW} from random strings. By Theorem 104, h is $\leq_{\mathrm{tt}}^{P/poly}$ reducible to $R^p_{\mathrm{K}_U}$ and the size of the reduction depends only on γ, for all large enough inputs. $\square$

In contrast to these completeness results it is known that R^p_{K} is not complete for co$-r.e.$ under $\leq^{rec}_{btt}$ and $\leq^{rec}_{dtt}$ reductions, since R^p_{K} is a simple set (Post, 1944, see also Kummer, 1996). This in particular implies that it is not complete under recursive many-one reductions.

9.1 Inside $P^{R_{\mathrm{K}}}$

In Section 8.2 we have shown that $PSPACE \subseteq P^{R^p_{\mathrm{K}}}$. The question we want to study now is how much recursive information one can extract from R_{K} by a polynomial time procedure. This question can be cast as what is $P^{R^p_{\mathrm{K}}} \cap rec$. Is it $PSPACE$ or is it some larger class? For instance if H were in $P^{R_{\mathrm{K}}}$ then all recursive sets would be there, too. We do not know answer to this question but we indicate several results that suggest that the answer may not be straightforward.

Theorem 126 *Let p be a non-decreasing function such that $n^\gamma \leq p(n) \leq n$, for some $0 < \gamma < 1$.*

1. $\{A \in rec;\ A \leq^P_{\mathrm{ctt}} R^p_{\mathrm{K}}\} \subseteq P$.

2. $\{A \in rec;\ A \leq^P_{\mathrm{btt}} R^p_{\mathrm{K}}\} \subseteq P$.

3. *If $p(n) \geq \epsilon n$ for some $\epsilon > 0$, then* $\{A \in rec;\ A \leq^P_{\mathrm{mtt}} R^p_{\mathrm{K}}\} \subseteq P/poly$, *else* $\{A \in rec;\ A \leq^P_{\mathrm{mtt}} R^p_{\mathrm{K}}\} \subseteq P/n^{\mathrm{poly}\log n}$.

Proof. In all three arguments we will have a recursive set A which is $\leq_{tt}^{<q,r>}$ reducible to R_{K}^{p}, where $< q, r >$ is the pair of polynomially computable functions computing the $\leq_{\mathrm{ctt}}^{P}$, $\leq_{\mathrm{btt}}^{P}$ and $\leq_{\mathrm{mtt}}^{P}$ reductions, respectively. For $x \in \{0,1\}^*$, Q_x will denote the set of queries produced by q on input x.

1. $< q, r >$ computes a $\leq_{\mathrm{ctt}}^{P}$ reduction. For any $x \in A$, $Q_x \subseteq R_{\mathrm{K}}^{p}$. Hence, $Q = \bigcup_{x \in A} Q_x$ is an *r.e.* subset of R_{K}^{p}. As R_{K}^{p} is simple, Q has to be finite. Hence we can hard-wire Q into a table and conclude that $A \in P$.

2. We will prove the claim by induction on the number of queries. If the reduction does not ask any query, the claim is trivial. Assume that the claim is true for reductions asking less than k queries. We will prove the claim for reductions asking at most k queries. Take $< q, r >$ that computes a $\leq_{\mathrm{btt}}^{P}$ reduction and such that $|Q_x| \leq k$, for all x. For any string x, let $m_x = \min\{|q|;\ q \in Q_x\}$. We claim that there exists an integer l such that for any x, if $m_x > l$ then $r(x, < q_1, 0 >, < q_2, 0 >, \ldots, < q_m, 0 >) = A(x)$. For contradiction assume that for any integer l, there exists x such that $m_x > l$ and $r(x, < q_1, 0 >, < q_2, 0 >, \ldots, < q_m, 0 >) \neq A(x)$. Since A is recursive, for any l, we can find the lexicographically first x_l having such a property. All the queries in Q_{x_l} are longer than l and at least one of them should be in R_{K}^{p}. However, each of the queries can be described by $O(\log l)$ bits, which is the contradiction. Hence, there exists an integer l such that for any x, if $m_x > l$ then $r(x, < q_1, 0 >, < q_2, 0 >, \ldots, < q_m, 0 >) = A(x)$. Thus we can encode possible answers to the smallest query into a table and reduce the number of queries in our reduction by one. Then we can apply the induction hypothesis.

3. $< q, r >$ computes a $\leq_{\mathrm{mtt}}^{P}$ reduction. q is computable in time n^c, for some $c > 1$. Let $p(n) \geq \epsilon n$, for some $\epsilon > 0$. We claim that r does not depend on any query of length more than $2c\epsilon^{-1} \log n$. Assume that for infinitely many x, r does depend on queries of length more than $2c\epsilon^{-1} \log |x|$, i.e., if $Q_x = \{q_1, q_2, \ldots, q_m\}$ and $a_1', a_2', \ldots, a_m' \in \{0,1\}$ are such that $a_i' = 1$ if $q_i \in R_{\mathrm{K}}^{p}$ and $|q_i| \leq 2c\epsilon^{-1} \log |x|$, and $a_i' = 0$ otherwise, then $r(x, < q_1, a_1' >, < q_2, a_2' >, \ldots, < q_m, a_m' >) \neq A(x)$. Since r is monotone, this may happen only for x that belongs to A. R_{K}^{p} is a co$-r.e.$ set so for all x we can be approximating the correct value of $a_1, \ldots, a_m$ from above in *r.e.* fashion until we find x that depends on a long query. Indeed, we can recursively enumerate all such x. Hence

for given l, we can find the first x of length more than l in this enumeration. One of the queries in Q_x is of length more than $2c\epsilon^{-1}\log l$ and it belongs to R^p_{K}. But we can describe every query in Q_x by $c\log l + \log l + O(1)$ bits which is less than $p(2c\epsilon^{-1}\log l)$. That is the contradiction. Since we have established, that r depends only on queries of length at most $2c\epsilon^{-1}\log n$, we can encode information about all strings of this size that belong to R^p_K into a polynomially large table. Thus A is in $P/poly$.

The case of $p(n) \geq n^\gamma$ is proven similarly. Using an argument similar to the one above, one can show that a monotone truth-table reduction running in time n^c cannot depend on queries of length more than $(2c\log n)^{1/\gamma}$. We can encode $(R^p_{\mathrm{K}})^{<(2c\log n)^{1/\gamma}}$ into a table of size $n^{\mathrm{poly}\log n}$ and the claim follows. $\square$

The following development is motivated by a question that one can naturally ask: what is the size of $(R^p_{\mathrm{K}})^{=n}$? It is a part of folklore that the number of strings in R^p_{K} of length n is Kolmogorov random. But is it odd or even? One would be tempted to answer that since $|(R^p_{\mathrm{K}})^{=n}|$ is Kolmogorov random, the parity of it must also be random. The following optimal Turing machine U_{even} shows that this is not the case.

Let U_{st} be a fixed optimal Turing machine. Consider the optimal Turing machine U_{even} defined by: for any $d \in \{0,1\}^*$, $U_{\mathrm{even}}(0d) = U_{\mathrm{st}}(d)$ and $U_{\mathrm{even}}(1d) =$ the bit-wise complement of $U_{\mathrm{st}}(d)$. It is immediate that the size of $(R^p_{\mathrm{K}_{U_{\mathrm{even}}}})^{=n}$ is even for all n. To construct an optimal Turing machine U_{odd} for which the size of $(R^p_{\mathrm{K}_{U_{\mathrm{odd}}}})^{=n}$ is odd for all n (large enough), is a little bit more complicated.

We will need the following definition. For any Turing machine U we can construct an *enumerator* (Turing machine) E that enumerates all pairs (d, x) such that $U(d) = x$, for $d, x \in \{0,1\}^*$. (The running time of E is possibly infinite.) Conversely, given an enumerator E that enumerates pairs (d, x) so that if (d, x) and (d, x') are enumerated then $x = x'$, we can construct a Turing machine U such that for any $x, d \in \{0,1\}^*$, $U(d) = x$ if and only if E ever enumerates the pair (d, x). In the next, we will often define a Turing machine in terms of its enumerator.

We define U_{odd} in terms of its enumerator E_{odd} that works as it is described below. Let p be a computable function such that $2\log n \leq p(n) \leq n$. E_{odd} will maintain sets of non-random strings $\{N_i\}_{i\in\mathbf{N}}$ during its operation. At any point of time, set N_i will contain non-random strings of length i that were enumerated by E_{odd} so far. E_{odd} will try to maintain the size of sets N_i odd (except while they are empty.)

Initialize all $\{N_i\}_{i\in\mathbf{N}}$ to the empty set.[1]

For all $d \in \{0,1\}^*$, run $U_{\mathrm{st}}(d)$ in parallel.

Whenever $U_{\mathrm{st}}(d)$ halts for some d and produces a string x do:

> Output $(0d, x)$.
>
> If $|0d| < p(|x|)$ and $N_{|x|} = \emptyset$ then set $N_{|x|} := \{x\}$.
>
> Else if $|0d| < p(|x|)$ and $x \notin N_{|x|}$ then:
>
> > Pick the lexicographically first string y in $\{0,1\}^{|x|} - (N_{|x|} \cup \{x\})$.
> >
> > Set $N_{|x|} := N_{|x|} \cup \{x, y\}$ and output $(1d, y)$.
>
> Continue.

End.

It is easy to see that the Turing machine U_{odd} defined by the enumerator E_{odd} is optimal. Also it is clear that for all n large enough, $(R^p_{\mathrm{K}_{U_{\mathrm{odd}}}})^{=n}$ is of odd size.

The ability to influence the parity of $(R^p_{\mathrm{K}_U})^{=n}$ allows us to (sparsely) encode any recursively enumerable information into $R^p_{\mathrm{K}_U}$. For a set $A \in \{0,1\}^*$, we say that a set B is a *sparse encoding of* A if $B = \{0^x; x \in A\}$. Further, we say that C is a *super-sparse encoding of* A if C is a sparse encoding of B. We can state the following theorem.

Theorem 127 *Let $p : \mathbf{N} \to \mathbf{N}$ be a computable function such that for all n, $2\log n < p(n) \leq n$. For any recursively enumerable set A, there is an optimal Turing machine U such that if C is a super-sparse encoding of A, then $C \leq^P_{\oplus\mathrm{tt}} R^p_{\mathrm{K}_U}$. Furthermore, $A \leq^{EE}_{\oplus\mathrm{tt}} R^p_{\mathrm{K}_U}$.*

Proof. Observe, $C \subseteq \{0^{2^i};\ i \in \mathbf{N}\}$. We will construct the optimal Turing machine U so that for any integer $i > 16$, $0^{2^i} \in C$ if and only if $(R^p_{\mathrm{K}_U})^{=i}$ is of odd size. Then, the polynomial time parity reduction of C to $R^p_{\mathrm{K}_U}$ can be constructed trivially as well as the double-exponential parity reduction of A to $R^p_{\mathrm{K}_U}$.

[1] We assume in usual way that E_{odd} works in steps and at step s it initializes the s-th set of $\{N_i\}_{i\in\mathbf{N}}$ to the empty set. Our statements regarding actions that involve infinite computation should be interpreted in similar way.

Let M be the Turing machine accepting the recursively enumerable set C. We will construct an enumerator E for U. It will work as follows. E will maintain sets $\{N_i\}_{i\in\mathbf{N}}$ and $\{P_i\}_{i\in\mathbf{N}}$ during its computations. For every $i > 0$, N_i will be a set of non-random strings of length i and P_i will be a set of unavailable strings (descriptions) of length i ($i+2$, respectively). At any point of time, set N_i will contain non-random strings of length i that were enumerated by E so far and E will try to maintain the parity of $|N_i|$ unchanged during most of the computation. E will also run M on all strings $z = 0^{2^i}$ in parallel and whenever some new string z will be accepted by M, E will change parity of $N_{\log|z|}$ by making some new string of length $\log|z|$ non-random. For that it will use some description in $11(\{0,1\}^{p(\log|z|)-3} - P_{p(\log|z|)-3})$. The algorithm for E is the following.

Initialize all $\{N_i\}_{i\in\mathbf{N}}$ and $\{P_i\}_{i\in\mathbf{N}}$ to the empty set.

For all $d \in \{0,1\}^*$ and $z \in \{0\}^*$, run $U_{\rm st}(d)$ and $M(z)$ in parallel.

Whenever $U_{\rm st}(d)$ or $N(z)$ halts for some d or z do:

> If $U_{\rm st}(d)$ halts and produces output x then:
>
> > Output $(00d, x)$.
> >
> > If $|00d| < p(|x|)$ and $x \notin N_{|x|}$ then:
> >
> > > Pick the lexicographically first string y in $\{0,1\}^{|x|} - (N_{|x|} \cup \{x\})$.
> > >
> > > Set $N_{|x|} := N_{|x|} \cup \{x, y\}$ and output $(01d, y)$.
> >
> > Continue.
>
> If $M(z)$ halts then:
>
> > Let $l = \log|z|$.
> >
> > Pick the lexicographically first description w in $\{0,1\}^{p(l)-3} - P_{p(l)-3}$.
> >
> > (If no w was found, skip the rest of this "if".)
> >
> > Pick the lexicographically first string y in $\{0,1\}^l - N_l$.
> >
> > Set $N_l := N_l \cup \{y\}$, $P_{p(l)-3} := P_{p(l)-3} \cup \{w\}$ and output $(11w, y)$.
> >
> > Continue.

End.

Clearly, enumerator E defines an optimal Turing machine. The only non-obvious fact that one has to verify regarding the algorithm is that in all but finitely many cases there is available a string $w \in \{0,1\}^{p(l)-3} - P_{p(l)-3}$ when it is needed. One can easily check that for any $k > 0$, if the length of string z is more than $2^{2^{k/2+2}}$ then $p(\log|z|) - 3 > k$. Since C is super-sparse, there are at most $2^{k/2+2}$ strings of length at most $2^{2^{k/2+2}}$ in C. Hence, there are 2^k strings w of length k available and there may be at most $2^{k/2+2}$ strings w needed. Thus, for any z longer than 2^{2^4}, we can always find a description w to make some string of length $\log|z|$ non-random. Hence, for any integer $i > 16$, $0^{2^i} \in C$ if and only if $(R^p_{\mathrm{K}_U})^{=i}$ is of odd size. □

The parity is not the only way how to encode information into R^p_{K}. The following theorem illustrates that we can encode the information so that one can use $\leq^P_{\mathrm{dtt}}$ reductions to extract it.

Theorem 128 *For any recursively enumerable set A, there is an optimal Turing machine U such that if C is a super-sparse encoding of A, then $\overline{C} \leq^P_{\mathrm{dtt}} R^n_{\mathrm{K}_U}$. Furthermore, $\overline{A} \leq^{EE}_{\mathrm{dtt}} R^n_{\mathrm{K}_U}$.*

Proof. First, define an optimal Turing machine U_{opt} as follows: $U_{\mathrm{opt}}(0d) = U_{\mathrm{st}}(d)$ and $U_{\mathrm{opt}}(1d) = d$. Clearly, for any $x \in \{0,1\}^*$, $\mathrm{K}_{U_{\mathrm{opt}}}(x) \leq |x| + 1$. For any $d \in \{0,1\}^*$ and any $s \in \{0,1\}^5$, U is defined as follows:

On input $0ds$, run $U_{\mathrm{opt}}(d)$ and if $U_{\mathrm{opt}}(d)$ halts then output $U_{\mathrm{opt}}(d)s$.

On input $1d$ do:

Run $U_{\mathrm{opt}}(d)$, until it halts.

Let y be the output of $U_{\mathrm{opt}}(d)$.

Check if $0^{2^{|y|}} \in C$.

If $0^{2^{|y|}} \in C$ then output $y0^5$.

End.

It is clear that for any $x \in \{0,1\}^*$, $\mathrm{K}_U(x) \leq |x| + 2$. Further, for any $s, s' \in \{0,1\}^5 - \{0^5\}$, $K_U(xs) = K_U(xs')$. Finally, for any $y \in \{0,1\}^*$, $0^{2^{|y|}} \in C$ if and only if $K_U(y0^5) < K_U(y1^5) - 4$. Hence, if $0^{2^{|y|}} \in C$ then $y0^5 \notin R^n_{\mathrm{K}}$. The $\leq^P_{\mathrm{dtt}}$ reduction of $\overline{C}$ to R^n_{K} works as follows: on input 0^{2^n}, for all $y \in \{0,1\}^n$ ask queries $y0^5$. Output 0 if none of the queries lies in R^n_{K} and 1 otherwise. Similarly for $\overline{A}$. □

The previous theorem can be proven also for any R^p_{K} instead of R^n_{K}, where $p(n)$ is a non-decreasing polynomially computable function such that for some $0 < \gamma < 1$, $n^\gamma \leq p(n) \leq n$ and $n^\gamma \leq |\{p(i);\ i \leq n\}|$. (Whether a string 0^{2^j} is in C or not is encoded in $R^p_{\mathrm{K}_U}$ on strings of length n_j, where n_j is the smallest integer such that $2j = |\{p(i);\ i \leq n_j \ \&\ p(i) > 5\}|$. For any string $d \in \{0,1\}^*$ and any $s \in \{0,1\}^5$, U maps $00ds$ to $U_{\mathrm{opt}}(d)s$. Further, for any n_j that is assigned to some j, any $d \in \{0,1\}^{p(n_j)-5}$ such that $p(n_j) - 5 \leq |U_{\mathrm{opt}}(d)| \leq p(n_j) - 4$ and any $s \in \{0,1\}^5$, U maps $10ds$ to $U_{\mathrm{opt}}(d)s0^{n_j - p(n_j)}$ and if $0^{2^j} \in C$ then U maps $11d$ to $U_{\mathrm{opt}}(d)0^{n_j+5-p(n_j)}$. We leave remaining details of this extension to the interested reader.)

One could start to suspect that maybe all recursive functions are reducible to R^p_{K} in, say, double exponential time. We do not know if that is true but the following theorem shows that certainly disjunctive truth-table reductions are not sufficient.

Theorem 129 *Let $2\log n \leq p(n) \leq n$ be a computable function. For any recursive time-bound $t(n) \geq n$, there exist a recursive set T and an optimal Turing machine U such that T does not reduce to $R^p_{K_U}$ via any disjunctive truth-table reduction that runs in time $t(n)$ for all n large enough.*

This theorem has also a consequence on the running time of Kummer's conjunctive truth-table reduction of H to $\overline{R^p_{\mathrm{K}}}$. It implies that there is no recursive bound on the running time of Kummer's reduction that would be independent of the fixed optimal Turing machine with respect to which the Kolmogorov complexity is measured. That is in contrast to our non-uniform *P/poly* truth-table reductions. We can state the following corollary.

Corollary 130 *Let $2\log n \leq p(n) \leq n$ be a computable function and $t(n) \geq n$ be a recursive function. Then there exists an optimal Turing machine U such that H does not reduce to $\overline{R^p_{K_U}}$ via any conjunctive truth-table reduction that runs in time $t(n)$ for all n large enough.*

Proof. Consider the recursive set T and the optimal Turing machine U that are guaranteed to exist by Theorem 129 for time bound $t(2^n)$. For contradiction assume that

$H \leq_{\text{ctt}}^{rec} \overline{R^p_{K_U}}$ via a conjunctive truth-table reduction that runs in time $t(n)$ for all n large enough. Clearly, $\overline{H} \leq_{\text{dtt}}^{rec} R^p_{K_U}$ via the same reduction and since $\overline{H}$ is poly-time many-one hard for rec, $T \leq_{\text{dtt}}^{rec} R^p_{K_U}$ via a reduction running in time $t(n^c + c)$, for some constant c and all n large enough. But this reduction also runs in time $t(2^n)$ for all n large enough. That contradicts properties of T and U. □

Proof of Theorem 129. We will construct T and U simultaneously. We will describe U in terms of its enumerator E. Let $q_1, q_2, \ldots$ be an enumeration of all Turing machines (*query generators*) that work in time at most $t(n)$. Without loss of generality we assume that every Turing machine running in time at most $t(n)$ appears in the sequence $q_1, q_2, \ldots$ infinitely often. During the construction of T and E we will diagonalize against all q_i's.

E plays several strategies simultaneously. Only the first two strategies of E enumerate any output. The third strategy defines T. We partition $\{0,1\}^*$ according to $p(n)$: for an integer $j \geq 0$, we define $L_j = \bigcup_{i;\, p(i)=j} \{0,1\}^i$. Since $2 \log n \leq p(n)$, every L_j is finite and there is a recursive procedure than on input j outputs L_j.

The first strategy. For all $d \in \{0,1\}^*$, E runs $U_{\text{st}}(d)$ in parallel. Whenever some computation $U_{\text{st}}(d)$ halts and produces output y, E outputs $(00d, y)$.

This strategy assures that E represents an optimal Turing machine. Also, for any $j \geq 2$, there are at most $2^{j-2} - 1$ strings in L_j that can get outside of R^p_{K} (become non-random) as a result of the first strategy.

The second strategy. From time to time, the third strategy will produce sets $S_j \subseteq L_j$, for different integers j. Whenever a new set S_j is produced, the second strategy does the following:

If $|S_j| > 2^j/4$ or $S_j = \emptyset$ then do nothing with S_j.

If $|S_j| \leq 2^j/4$ and $S_j \neq \emptyset$ then for all $m \in \{1, \ldots, |S_j|\}$ do:

Let $s_{j,m}$ be the m-th element of S_j and $d_{j,m}$ be the m-th element of $\{0,1\}^{j-2}$.

Output $(1d_{j,m}, s_{j,m})$.

The consequence of the second strategy is that if $|S_j| \leq 2^j/4$ then $S_j \subseteq \overline{R^p_{\text{K}_U}}$, else $S_j \cap R^p_{\text{K}_U} \neq \emptyset$.

The third strategy. E proceeds according to the algorithm described below. The algorithm proceeds in stages. At stage k, it will diagonalize against reduction q_k by defining $T(x_i)$ for several strings x_i. It will also define an integer l_{k+1} such that after stage k, there is not going to appear any new set S_j, for $j < l_{k+1}$.

Set l_1 to be the smallest $j \geq 3$ such that $L_{j-1} \neq \emptyset$.

For successive $k := 1, 2, 3, \ldots$, do the following:

Set $\mathcal{L} := \bigcup_{j<l_k} L_j$.

Choose $m = 2^{|\mathcal{L}|}$ new strings $x_1, \ldots, x_m$ not used before.

We will define $T(x_1), \ldots, T(x_m)$ so to diagonalize q_k on at least one of x_i's.

Associate with every x_i a different map $a_i \in \{a : \mathcal{L} \to \{0, 1\}\}$.

Each x_i "guesses" that a_i represents the truth value of $R^p_{\mathrm{K}_U}$ on $\mathcal{L}$.

For $i \in \{1, \ldots, m\}$, set the i-th query set to be $Q_i := q_k(x_i)$.

Set $I := \{i;\ \exists q \in Q_i \cap \mathcal{L}$ such that $a_i(q) = 1\}$.

For $i \in I$, define $T(x_i) = 0$.

Now we have diagonalized on every x_i that guesses that some of its queries $q \in \mathcal{L}$ belongs to $R^p_{\mathrm{K}_U}$.

Set $Q := \bigcup_{i \notin I} Q_i$.

Set $l_{k+1} := 1 + \max\{l_k, p(|z|);\ z \in Q\}$.

Observe that $Q - \mathcal{L} \subseteq \bigcup_{l_k \leq j < l_{k+1}} L_j$.

For $j \in \{l_k, \ldots, l_{k+1} - 1\}$, set $S_j := Q \cap L_j$.

If there is $j \in \{l_k, ..., l_{k+1} - 1\}$, such that $|S_j| > 2^j/4$ then:

For every $i \notin I$, define $T(x_i) = 0$,

Else for every $i \notin I$, define $T(x_i) = 1$.

At this moment the second strategy helps to diagonalize for the remaining x_i's.

Continue with the next k.

End.

The algorithm can use an arbitrary strategy for choosing $x_1, \ldots, x_m$ on which it diagonalizes. For example they may be chosen consecutively from $\{0^{2^i};\ i \in \mathbf{N}\}$ so to obtain a super-sparse set T. The elements of $\{0,1\}^*$ on which the algorithm does not diagonalize, i.e., that are not set by the algorithm to be in T or outside of T, may be set arbitrarily, for example, they all may be set to be outside of T. Given a recursive strategy for choosing x_i's, enumerator E is recursively enumerable.

If the x_i's are chosen consecutively from some recursive set C and all elements not belonging to C are put outside of T, then the constructed set T is recursive: On input x, if $x \notin C$ then $x \notin T$. If $x \in C$, then T simulates the algorithm of E until x is chosen for diagonalization and it is decided whether it is in T or not.

We constructed T so that it is not reducible to R_{K}^p via any disjunctive truth-table reduction running in time at most $t(n)$. Since we diagonalize against any such reduction infinitely often, T is not reducible to R_{K}^p via any disjunctive truth-table reduction running in time $t(n)$ for all n large enough. $\square$

Chapter 10

Hardness of Approximation

Many computational problems that complexity theory studies are decision problems for which an answer is always either "yes" or "no". Examples of such problems are the s-t-connectivity problem — is there a path from s to t in a graph G — and the Clique problem — is there a clique of size k in G. Other problems that are of interest in computational complexity are optimization problems. Examples of optimization problems are the Shortest s-t-path — what is the length of the shortest path from s to t in G — and the Maximum Clique — what is the size of the largest clique in G.

For some of these optimization problems (e.g., the Shortest s-t-path problem) we know efficient (polynomial time) algorithms. For others, we do not know any efficient algorithm. Moreover, we know that these optimization problems are hard for NP. Given that the exact solution of such an optimization problem may be hard to find one can try to find at least an approximation to the solution. There are known many optimization problems for which even finding an approximation cannot be done efficiently, unless something unlikely is true, such as $P = NP$. For example, Håstad (1999) shows that the Maximum Clique cannot be approximated up to factor $n^{1-\epsilon}$ in polynomial time, unless $P = NP$.

In this section we study the following optimization problems — given a truth-table of a function f, what is the smallest size of a circuit, a branching program or a formula, respectively, that computes f. We show that under certain plausible complexity assumptions these optimization problems are hard to approximate.

A related problem was already studied by Czort (1999). In his Master's Thesis he gives for some $\epsilon > 0$ an n^ϵ non-approximability result for a variant of the Minimum DNF Formula problem, under the assumption that $P \neq NP$. His result builds upon probabilistically checkable proofs. We obtain our results using completely different technique. Tools for our non-approximability results are related to hardness results for the sets R^p_{KT}, R^p_{KB} and R^p_{KF}. By similar technique one can prove non-approximability results also for resource-bounded Kolmogorov complexity such as Kt and KS.

For a minimization problem $f : \Sigma^* \to \mathbf{N}$ we say that $g : \Sigma^* \to \mathbf{N}$ *approximates* f *up to factor* $r : \mathbf{N} \to \mathbf{N}$ if for all $x \in \Sigma^*$, $1 \le g(x)/f(x) \le r(|x|)$. For a complexity class $\mathcal{C}$ we say that f *cannot be approximated up to factor* r *in* $\mathcal{C}$ if no $g \in \mathcal{C}$ approximates f up to factor r.

We will define now several problems that are believed to be computationally difficult.

- *Integer Factorization*: Given a composite integer $N \in \mathbf{N}$, find two integers P and Q such that $1 < P \le Q$ and $N = PQ$.
- *Blum Integer Factorization*: An $2n$-bit integer N is called *Blum Integer* if $N = PQ$, where P and Q are two primes such that $P \equiv Q \equiv 3 \bmod 4$. Given a Blum Integer $N \in \mathbf{N}$, find the primes P and Q such that $1 < P \le Q$ and $N = PQ$.
- *Discrete Logarithm*: Given three integers x, z, N, $1 \le x, z < N$, find an i such that $x = z^i \bmod N$ if such i exists.

The following result is implicit in Allender et al. (2002):

Theorem 131 *Let $0 < \gamma < 1$ be a constant and B be a set of polynomial density such that for any $x \in B$, $\mathrm{SIZE}(x) > |x|^\gamma$. Then Integer Factorization and Discrete Logarithm are in BPP^B.*

This theorem implies the non-approximability of circuit size.

Theorem 132 *For any $0 < \epsilon < 1$, $\mathrm{SIZE}(x)$ cannot be approximated up to factor $n^{1-\epsilon}$ in BPP, unless Integer Factorization and Discrete Logarithm is in BPP.*

Proof. Assume that for some $0 < \epsilon < 1$, there is a function $g \in BPP$ that approximates $\mathrm{SIZE}(x)$ up to factor $n^{1-\epsilon}$. We will show that this implies that Integer Factorization and Discrete Logarithm are in BPP.

Consider the set $B = \{x \in \{0,1\}^*;\ g(x) > |x|^{1-\epsilon/2}\}$. Clearly, $B \in BPP$. Since for all $x \in \{0,1\}^*$, $1 \leq g(x)/\text{SIZE}(x) \leq n^{1-\epsilon}$, we have that for all $x \in B$, $\text{SIZE}(x) > |x|^{\epsilon/2}$ and also for all $x \in \{0,1\}^*$, if $\text{SIZE}(x) \geq |x|^{1-\epsilon/2}$ then $x \in B$. By Lupanov (1959), almost all truth-tables $x \in \{0,1\}^*$ require circuits of size at least $O(n/\log n)$. Hence, B is of polynomial density. By Theorem 131, Integer Factorization and Discrete Logarithm are in $BPP^{BPP} \subseteq BPP$. (In the case of Integer Factorization we can actually verify correctness of the result so to get ZPP computation instead of BPP.) □

Similar non-approximability results can be obtained for formula and branching program sizes.

Theorem 133 *For any $0 < \epsilon < 1$, $\text{FSIZE}(x)$ and $\text{BSIZE}(x)$ cannot be approximated up to factor $n^{1-\epsilon}$ in BPP, unless Blum Integer Factorization is in ZPP.*

To prove this theorem we need the following statement.

Theorem 134 *Let $0 < \gamma < 1$ be a constant and B be a set of polynomial density such that for any $x \in B$, $\text{FSIZE}(x) > |x|^{\gamma}$. Then there is a ZPP^B procedure that on input N that is a Blum Integer produces factors P and Q of N.*

Proof. Naor & Reingold (1997) construct a pseudo-random function ensemble $\{f_{N,r}(x) : \{0,1\}^n \to \{0,1\}\}_{N,r}$ with the following two properties (Construction 5.2 and Corollary 5.6 of Naor & Reingold, 1997):

1. There is a TC^0 circuit computing $f_{N,r}(x)$, given $2n$-bit integer N, $4n^2 + 2n$-bit string r and n-bit string x.

2. For every polynomial time probabilistic oracle Turing machine M, that on its $2n$-bit input asks queries of length only n, and any constant $\alpha > 0$, there is a polynomial time probabilistic Turing machine A, such that for any $2n$-bit Blum Integer $N = PQ$, if

$$|\Pr[M^{f_{N,r}(x)}(N) = 1] - \Pr[M^{R_n}(N) = 1]| > 1/n^{\alpha}$$

where $R_n = \{g : \{0,1\}^n \to \{0,1\}\}_n$ is a uniformly distributed random function ensemble and the probability is taken over the random string r and the random bits of M, then $\Pr[A(N) \in \{P,Q\}] > 1/n$.

Their factorization construction relativizes, i.e., the properties of $\{f_{N,r}(x)\}_{N,r}$ hold even if M and A have an access to the same auxiliary oracle.

Let $f_{N,r}(x)$ be computable by a TC^0 circuit of size n^c and hence, by a formula of size $n^{c'}$, for some constants $c, c' > 1$. Let $d = 2c'/\gamma$. Let $x_1, x_2, \ldots, x_{2^n}$ denote the strings in $\{0,1\}^n$ under lexicographical ordering. For all large enough n, all $2n$-bit integers N and all $4n^2 + 2n$-bit strings r, the string y obtained by concatenating $f_{N,r}(x_1), f_{N,r}(x_2), \ldots, f_{N,r}(x_{n^d})$ has formula size at most $n^{c'} < (n^d)^\gamma$. Consider the following oracle TM M with oracles B and a function g:

- On $2n$-bit input N, M asks oracle g queries $x_1, x_2, \ldots, x_{n^d}$ to get answers y_1, y_2, $\ldots$, y_{n^d}. Then, M accepts if $y_1 y_2 \cdots y_{n^d} \in B$ and rejects otherwise.

It is easy to see that if $g \in \{f_{N,r}(x)\}_{N,r}$ then M always rejects, for n large enough. On the other hand, if g is taken uniformly at random from R_n, then $y_1 y_2 \cdots y_{n^d}$ is a random string and the probability that M accepts is at least $1 - 2^{-n^d + n^{d\gamma}}$. Hence, $|Pr[M^{f_{N,r}(x)}(N) = 1] - Pr[M^{R_n}(N) = 1]| > 1/2$, for n large enough. By the properties of $f_{N,r}(x)$ we can conclude that there is a polynomial time probabilistic Turing machine A with oracle B that factors N with non-negligible probability. We can reduce the error to zero by verifying the output of A. $\square$

Since any function in TC^0 is computable by a polynomial size branching program, we can prove Theorem 134 for BSIZE(x) instead of FSIZE(x) by an essentially identical proof. Theorem 133 follows by a proof similar to the one of Theorem 132.

In Theorems 132 and 133, a function f is computable in BPP if there is a polynomial time probabilistic machine M such that for any x, $\Pr[M(x) = f(x)] \geq 2/3$. However, the results hold for an even stronger notion of non-approximability: For any $0 < \epsilon < 1$, if there is a polynomial time probabilistic machine M such that for all x, $\Pr[1 \leq M(x)/\text{BSIZE}(x) \leq n^{1-\epsilon}] \geq 2/3$ or $\Pr[1 \leq M(x)/\text{FSIZE}(x) \leq n^{1-\epsilon}] \geq 2/3$ then Blum Integer Factorization is in ZPP. Similarly, if there is a polynomial time probabilistic machine M such that for all x, $\Pr[1 \leq M(x)/\mathrm{SIZE}(x) \leq n^{1-\epsilon}] \geq 2/3$ then Integer Factorization and Dicrete Logarithm are in BPP. These results follow by exactly the same proofs as the weaker one where one has to observe that the derandomization results that we use hold not only relative to oracles that distinguish between random and pseudo-random strings but also relative to probabilistic procedures that distinguish between random and pseudo-random strings with non-negligible probability.

Theorem 134 together with Theorem 99 also gives the following hardness results.

Theorem 135 *Let p be a function such that $n^\gamma \leq p(n) \leq n$, for some $0 < \gamma < 1$ and all $n > 0$. Then Blum Integer Factorization is in $ZPP^{R^p_{\mathrm{KF}}}$ and $ZPP^{R^p_{\mathrm{KB}}}$.*

We conclude this section with the following non-approximability results for Kt and KS that are implied by our Theorems 106 and 109.

Theorem 136 *For any $0 < \epsilon < 1$, $\mathrm{Kt}(x)$ cannot be approximated up to factor $n^{1-\epsilon}$ in P (P/poly), unless $EXP \subseteq NP$ ($EXP \subseteq P$/poly). Further, $\mathrm{KS}(x)$ cannot be approximated up to factor $n^{1-\epsilon}$ in BPP (P/poly), unless $PSPACE \subseteq BPP$ ($PSPACE \subseteq P$/poly)*

References

Ajtai, M., Komlós, J. & Szemerédi, E. (1983), 'Sorting in $c \log n$ parallel steps', *Combinatorica* **3**, 1–19.

Ajtai, M., Komlós, J. & Szemerédi, E. (1987), Deterministic simulation in LOGSPACE, *in* 'Proceedings of the 19th annual ACM Symposium on Theory of Computing', pp. 132–140.

Aleliunas, R. (1978), A simple graph traversing problem, Master's thesis, University of Toronto.

Aleliunas, R., Karp, R. M., Lipton, R. J., Lovász, L. & Rackoff, C. (1979), Random walks, universal traversal sequences, and the complexity of maze problems, *in* 'Proceedings of the 20th Annual Symposium on Foundations of Computer Science', pp. 218–223.

Allender, E. (2001), When worlds collide: Derandomization, lower bounds, and kolmogorov complexity, *in* 'Conference on Foundations of Software Technology and Theoretical Computer Science (FST&TCS)', Vol. 2245 of *Lecture Notes in Computer Science*, Springer, pp. 1–15.

Allender, E., Buhrman, H., Koucký, M., van Melkebeek, D. & Ronneburger, D. (2002), Power from random strings, *in* 'Proceedings of the 43th Annual Symposium on Foundations of Computer Science', pp. 669–678.

Allender, E., Koucký, M., Ronneburger, D. & Roy, S. (2003), Derandomization and distinguishing complexity, *in* 'Proceedings of the 18th IEEE Conference on Computational Complexity'. to appear.

Allender, E. & Mahajan, M. (2000), 'The complexity of planarity testing', *Lecture Notes in Computer Science* **1770**, 87–98.

Alon, N., Azar, Y. & Ravid, Y. (1990), 'Universal sequences for complete graphs', *DAMATH: Discrete Applied Mathematics and Combinatorial Operations Research and Computer Science* **27**, 25–28.

Àlvarez, C. & Greenlaw, R. (2000), 'A compendium of problems complete for symmetric logarithmic space', *Computational Complexity* **9**, 73–95.

Armoni, R., Ta-Shma, A., Wigderson, A. & Zhou, S. (2000), 'A $(\log n)^{4/3}$ space algorithm for (s, t) connectivity in undirected graphs', *Journal of the ACM* **47**(2), 294–311.

Babai, L., Fortnow, L. & Lund, C. (1991), 'Non-deterministic exponential time has two-prover interactive protocols', *Computational Complexity* **1**, 3–40.

Babai, L., Fortnow, L., Nisan, N. & Wigderson, A. (1993), 'BPP has subexponential time simulations unless $EXPTIME$ has publishable proofs', *Computational Complexity* **3**, 307–318.

Babai, L. & Moran, S. (1988), 'Arthur-Merlin games: A randomized proof system, and a hierarchy of complexity classes', *Journal of Computer and System Sciences* **36**, 254–276.

Babai, L., Nisan, N. & Szegedy, M. (1992), 'Multiparty protocols, pseudorandom generators for logspace, and time-space trade-offs', *Journal of Computer and System Sciences* **45**(2), 204–232.

Baker, T., Gill, J. & Solovay, R. (1975), 'Relativizations of the $\mathcal{P} =? \mathcal{NP}$ question', *SIAM Journal on Computing* **4**(4), 431–442.

Bar-Noy, A., Borodin, A., Karchmer, M., Linial, N. & Werman, M. (1989), 'Bounds on universal sequences', *SIAM Journal on Computing* **18**(2), 268–277.

Barnes, G., Buss, J. F., Ruzzo, W. L. & Schieber, B. (1998), 'A sublinear space, polynomial time algorithm for directed s-t connectivity', *SIAM Journal on Computing* **27**(5), 1273–1282.

Barnes, G. & Ruzzo, W. L. (1996), 'Undirected s-t connectivity in polynomial time and sublinear space', *Computational Complexity* **6**(1), 1–28.

Barrington, D. A. M., Immerman, N. & Straubing, H. (1990), 'On uniformity within NC^1', *Journal of Computer and System Sciences* **41**(3), 274–306.

Beame, P., Impagliazzo, R. & Pitassi, T. (1998), 'Improved depth lower bounds for small distance connectivity', *Computational Complexity* **7**.

Bennett, C. H. & Gill, J. (1981), 'Relative to a random oracle A, $P^A \neq NP^A \neq$ Co-NP^A with probability 1', *SIAM Journal on Computing* **10**(1), 96–113.

Blum, M. & Micali, S. (1984), 'How to generate cryptographically strong sequences of pseudo-random bits', *SIAM Journal on Computing* **13**(4), 850–864.

Borodin, A., Cook, S. A., Dymond, P. W., Ruzzo, W. L. & Tompa, M. (1989), 'Two applications of inductive counting for complementation problems', *SIAM Journal on Computing* **18**(3), 559–578.

Borodin, A., Ruzzo, W. L. & Tompa, M. (1992), 'Lower bounds on the length of universal traversal sequences', *Journal of Computer and System Sciences* **45**(2), 180–203.

Bridgland, M. F. (1987), 'Universal traversal sequences for paths and cycles', *Journal of Algorithms* **8**(3), 395–404.

Buhrman, H., Fortnow, L. & Laplante, S. (2002), 'Resource-bounded Kolmogorov complexity revisited', *SIAM Journal on Computing* **31**(3), 887–905.

Buhrman, H., Fortnow, L., Newman, I. & Vereshchagin, N. (2003), Increasing complexity. Unpublished manuscript.

Buhrman, H. & Mayordomo, E. (1997), 'An excursion to the Kolmogorov random strings', *Journal of Computer and System Sciences* **54**, 393–399.

Buhrman, H. & Torenvliet, L. (2001), 'Randomness is hard', *SIAM Journal on Computing* **30**, 1485–1501.

Buss, J. & Tompa, M. (1995), 'Lower bounds on universal traversal sequences based on chains of length five', *Information and Computation* **120**(2), 326–329.

Buss, S. R. (1993), Algorithms for boolean formula evaluation and for tree-contraction, *in* 'In Proof Theory, Complexity, and Arithmetic, P. Clote and J. Krajíček (eds), Oxford University Press', pp. 95–115.

Carter, J. L. & Wegman, M. N. (1979), 'Universal classes of hash functions', *Journal of Computer and System Sciences* **18**(2), 143–154.

Chandra, A. K., Raghavan, P., Ruzzo, W. L., Smolensky, R. & Tiwari, P. (1997), 'The electrical resistance of a graph captures its commute and cover times', *Computational Complexity* **6**, 312–340.

Cook, S. A. & McKenzie, P. (1987), 'Problems complete for deterministic logarithmic space', *Journal of Algorithms* **8**(3), 385–394.

Cook, S. A. & Rackoff, C. W. (1980), 'Space lower bounds for maze threadability on restricted machines', *SIAM Journal on Computing* **9**(3), 636–652.

Czort, S. L. A. (1999), The complexity of minimizing disjunctive normal form fomulas, Master's thesis, University of Aarhus.

Dai, H. K. (2001), 'Optimizing a computational method for length lower bounds for reflecting sequences', *Lecture Notes in Computer Science* **2108**, 228–236.

Edmonds, J., Poon, C. K. & Achlioptas, D. (1999), 'Tight lower bounds for *st*-connectivity on the NNJAG model', *SIAM Journal on Computing* **28**(6), 2257–2284.

Feige, U. (1997), 'A spectrum of time–space trade-offs for undirected *s*-*t* connectivity', *Journal of Computer and System Sciences* **54**(2), 305–316.

Gál, A. & Wigderson, A. (1996), 'Boolean complexity classes vs. their arithmetic analogs', *Random Structures and Algorithms* **9**, 1–13.

Goldreich, O. (1995), Three XOR-lemmas - an exposition, Technical Report TR95-056, Electronic Colloquium on Computational Complexity.

Goldreich, O. & Levin, L. (1989), A hard-core predicate for all one-way functions, *in* 'Proceedings of the 21st annual ACM Symposium on Theory of Computing', pp. 25–32.

Goldreich, O. & Wigderson, A. (2002), 'Derandomization that is rarely wrong from short advice that is typically good', *Lecture Notes in Computer Science* **2483**, 209–223.

Håstad, J. (1986), Almost optimal lower bounds for small depth circuits, *in* 'Proceedings of the 18th Annual ACM Symposium on Theory of Computing', pp. 6–20.

Hartmanis, J. (1983), Generalized Kolmogorov complexity and the structure of feasible computations (preliminary report), *in* 'Proceedings of the 24th Annual Symposium on Foundations of Computer Science', pp. 439–445.

Håstad, J. (1999), 'Clique is hard to approximate within $n^{1-\epsilon}$', *Acta Mathematica* **182**, 105–142.

Håstad, J., Impagliazzo, R., Levin, L. & Luby, M. (1999), 'A pseudorandom generator from any one-way function', *SIAM Journal on Computing* **28**, 1364–1396.

Hesse, W., Allender, E. & Barrington, D. A. M. (2002), 'Uniform constant-depth threshold circuits for division and iterated multiplication', *Journal of Computer and System Sciences* **65**, 695–716.

Hoory, S. & Wigderson, A. (1993), 'Universal traversal sequences for expander graphs', *Information Processing Letters* **46**(2), 67–69.

Immerman, N. (1988), 'Nondeterministic space is closed under complementation', *SIAM Journal on Computing* **17**(5), 935–938.

Impagliazzo, R., Kabanets, V. & Wigderson, A. (2002), 'In search of an easy witness: Exponential time vs. probabilistic polynomial time', *"Journal of Computer and System Sciences* **65**, 672–694.

Impagliazzo, R., Nisan, N. & Wigderson, A. (1994), Pseudorandomness for network algorithms, *in* 'Proceedings of the 26th Annual ACM Symposium on the Theory of Computing', pp. 356–364.

Impagliazzo, R. & Wigderson, A. (1997), $P = BPP$ if E requires exponential circuits: Derandomizing the XOR lemma, *in* 'Proceedings of the 29th annual ACM Symposium on Theory of Computing', pp. 220–229.

Impagliazzo, R. & Wigderson, A. (1998), Randomness vs. time: de-randomization under a uniform assumption, *in* 'IEEE Symposium on Foundations of Computer Science (FOCS)', pp. 734–743.

Isaacson, D. L. & Madsen, R. W. (1976), *Markov chains, theory and applications*, Wiley.

Istrail, S. (1988), Polynomial universal traversing sequences for cycles are constructible (extended abstract), *in* 'Proceedings of the 20th Annual ACM Symposium on Theory of Computing', pp. 491–503.

Istrail, S. (1990), Constructing generalized universal traversing sequences of polynomial size for graphs with small diameter (extended abstract), *in* 'Proceedings of the 31st Annual Symposium on Foundations of Computer Science', Vol. I, pp. 439–448.

Jenner, B., Lange, K.-J. & McKenzie, P. (1997), Tree isomorphism and some other complete problems for deterministic logspace, Technical Report 1059, Université de Montréal.

Jones, N. D., Lien, Y. E. & Laaser, W. T. (1976), 'New problems complete for nondeterministic loc space', *Mathematical Systems Theory* **10**, 1–17.

Kabanets, V. (2001), 'Easiness assumptions and hardness tests: Trading time for zero error', *Journal of Computer and System Sciences* **63**(2), 236–252.

Kabanets, V. & Cai, J.-Y. (2000), Circuit minimization problem, *in* 'Proceedings of the 32nd annual ACM Symposium on Theory of Computing', pp. 73–79.

Kahn, J. D., Linial, N., Nisan, N. & Saks, M. E. (1989), 'On the cover time of random walks on graphs', *Journal of Theoretical Probability* **2**(1), 121–128.

Karchmer, M. & Wigderson, A. (1990), 'Monotone circuits for connectivity require super-logarithmic depth', *SIAM Journal on Discrete Mathematics* **3**(2), 255–265.

Karchmer, M. & Wigderson, A. (1993), On span programs, *in* 'Proceedings of the 8th Annual Conference on Structure in Complexity Theory', pp. 102–111.

Karloff, H. J., Paturi, R. & Simon, J. (1988), 'Universal traversal sequences of length $n^{O(\log n)}$ for cliques', *Information Processing Letters* **28**(5), 241–243.

Klivans, A. R. & van Melkebeek, D. (1999), Graph nonisomorphism has subexponential size proofs unless the polynomial-time hierarchy collapses, *in* 'Proceedings of the 31st annual ACM Symposium on Theory of Computing', pp. 659–667.

Ko, K.-I. (1986), 'On the notion of infinite pseudorandom sequences', *Theoretical Computer Science* **48**(1), 9–33.

Ko, K.-I. (1991), 'On the complexity of learning minimum time-bounded turing machines', *SIAM Journal on Computing* **20**, 962–986.

Koucký, M. (2002), 'Universal traversal sequences with backtracking', *Journal of Computer and System Sciences* **65**, 717–726.

Koucký, M. (2003), 'Log-space constructible universal traversal sequences for cycles of length $O(n^{4.03})$', *Theoretical Computer Science* **296**(1), 117–144.

Kummer, M. (1996), On the complexity of random strings, *in* 'Symposium on Theoretical Aspects of Computer Science (STACS)', Vol. 1046 of *Lecture Notes in Computer Science*, Springer, pp. 25–36.

Ladner, R. & Lynch, N. (1976), 'Relativization of questions about log space reducibility', *Mathematical Systems Theory* **10**, 19–32.

Levin, L. A. (1984), 'Randomness conservation inequalities; information and independence in mathematical theories', *Information and Control* **61**, 15–37.

Levin, L. A. (1987), 'One-way functions and pseudorandom generators', *Combinatorica* **7**(4), 357–363.

Lewis, H. R. & Papadimitriou, C. H. (1982), 'Symmetric space-bounded computation', *Theoretical Computer Science* **19**(2), 161–187.

Li, M. & Vitanyi, P. (1993), *Introduction to Kolmogorov Complexity and its Applications.*, Springer-Verlag.

Lindell, S. (1992), A logspace algorithm for tree canonization (extended abstract), *in* 'Proceedings of the 24th Annual ACM Symposium on the Theory of Computing', pp. 400–404.

Lupanov, O. B. (1959), 'A method of circuit synthesis', *Izvestiya VUZ, Radiofizika* **1**(1), 120–140.

Martin, D. (1966), 'Completeness, the recursion theorem and effectively simple sets', *Proceedings of the American Mathematical Society* **17**, 838–842.

Naor, M. & Reingold, O. (1997), Number-theoretic constructions of efficient pseudo-random functions, *in* 'Proceedings of the 38th IEEE Symp. on Foundations of Computer Science', pp. 458–467.

Nisan, N. (1992), 'Psuedorandom generators for space-bounded computation', *Combinatorica* **12**(4), 449–461.

Nisan, N. (1994), '$RL \subseteq SC$', *Computational Complexity* **4**(1), 1–11.

Nisan, N., Szemerédi, E. & Wigderson, A. (1992), Undirected connectivity in $O(\log^{1.5} n)$ space, *in* 'Proceedings of the 33rd Annual Symposium on Foundations of Computer Science', pp. 24–29.

Nisan, N. & Ta-Shma, A. (1995), Symmetric Logspace is closed under complement, *in* 'Proceedings of the 27th Annual ACM Symposium on the Theory of Computing', pp. 140–146.

Nisan, N. & Wigderson, A. (1994), 'Hardness vs randomness', *Journal of Computer and System Sciences* **49**(2), 149–167.

Nisan, N. & Zuckerman, D. (1996), 'Randomness is linear in space', *Journal of Computer and System Sciences* **52**(1), 43–52.

Papadimitriou, C. (1994), *Computational Complexity*, Addison-Wesley, New York.

Post, E. L. (1944), 'Recursively enumerable sets of positive integers and their decision problems', *Bulletin of American Mathematical Society* **50**, 284–316.

Raz, R. & Reingold, O. (1999), On recycling the randomness of states in space bounded computation, *in* 'Proceedings of the 31st annual ACM Symposium on Theory of Computing', pp. 159–168.

Razborov, A. A. & Rudich, S. (1997), 'Natural proofs', *Journal of Computer and System Sciences* **55**, 24–35.

Reif, J. H. (1984), 'Symmetric complementation', *Journal of the ACM* **31**(2), 401–421.

Reinhardt, K. & Allender, E. (2000), 'Making nondeterminism unambiguous', *SIAM Journal on Computing* **29**(4), 1118–1131.

Ruzzo, W. L. (1981), 'On uniform circuit complexity', *Journal of Computer and System Sciences* **22**(3), 365–383.

Saks, M. (1996), Randomization and derandomization in space-bounded computation, *in* 'Proceedings of the 11th IEEE Conference on Computational Complexity', pp. 128–149.

Saks & Zhou (1999), '$BP_HSPACE(S) \subseteq DSPACE(S^{3/2})$', *Journal of Computer and System Sciences* **58**(2), 376–403.

Savitch, W. J. (1970), 'Relationships between nondeterministic and deterministic tape complexities', *Journal of Computer and System Sciences* **4**(2), 177–192.

Sipser, M. (1983), A complexity theoretic approach to randomness, *in* 'Proceedings of the 15th Annual ACM Symposium on the Theory of Computing', pp. 330–335.

Sudan, M., Trevisan, L. & Vadhan, S. (2001), 'Pseudorandom generators without the XOR lemma', *Journal of Computer and System Sciences* **62**, 236–266.

Szelepcsényi, R. (1988), 'The method of forced enumeration for nondeterministic automata', *Acta Informatica* **26**(3), 279–284.

Tompa, M. (1982), 'Two familiar transitive closure algorithms which admit no polynomial time, sublinear space implementations', *SIAM Journal on Computing* **11**(1), 130–137.

Tompa, M. (1992), 'Lower bounds on universal traversal sequences for cycles and other low degree graphs', *SIAM Journal on Computing* **21**(6), 1153–1160.

Trevisan, L. (2001), 'Construction of extractors using pseudo-random generators', *Journal of the ACM* **48**(4), 860–879.

Trevisan, L. & Vadhan, S. (2002), Pseudorandomness and average-case complexity via uniform reductions, *in* 'Proceedings of the 17th Computational Complexity Conference', pp. 129–138.

Wigderson, A. (1992), The complexity of graph connectivity, *in* 'Proceedings of MFCS '92. Mathematical Foundations of Computer Science', Vol. 629 of *Lecture Notes in Computer Science*, pp. 112–132.

Yao, A. (1982), Theory and application of trapdoor functions, *in* 'Proceedings of the 23rd Annual Symposium on Foundations of Computer Science', pp. 80–91.

Vita

Michal Koucký

1992 Graduated with the Highest honors from Gymnázium Korunní, Prague 2, Czech Republic.

1992-98 Undergraduate study, Charles University, Prague, Czech Republic.

1998 M. Koucký. Marking one-way multihead finite automata. Master's thesis, Charles University, Prague.

1998 Magister degree (Summa cum laude) in Computer Science, Charles University, Prague, Czech Republic.

1998-03 Graduate study, Department of Computer Science, Rutgers, The State University of New Jersey, New Brunswick, New Jersey.

1998-01 Teaching Assistant, Department of Computer Science, Rutgers, The State University of New Jersey, New Brunswick, New Jersey.

2000-03 Research Assistant, Department of Computer Science, Rutgers, The State University of New Jersey, New Brunswick, New Jersey.

2001 M. Koucký. Universal traversal sequences with backtracking. In *Proceedings of the 16th Annual IEEE Conference on Computational Complexity*, pp. 21–27.

2001 E. Allender, M. Koucký, D. Ronneburger, S. Roy, V Vinay. Space-time tradeoffs in counting hierarchy. In *Proceedings of the 16th IEEE Conference on Computational Complexity*, pp. 295–302.

2001 M. Koucký. Universal traversal sequences for cycles of length $O(n^{4.03})$. In *Proceedings of the 7th Annual International Computing and Combinatorics Conference*, LNCS, Vol. 2108, Springer, Berlin, pp. 11–20.

2002 E. Allender, H. Buhrman, M. Koucký, D. van Melkebeek, D. Ronneburger. Power from random strings. In *Proceedings of the 43rd Annual Symposium on Foundations of Computer Science*, pp. 669–678.

2002 M. Koucký. Universal traversal sequences with backtracking. In *Journal of Computer and System Sciences*, 65, pp. 717–726. Special issue of selected papers from the 16th Annual IEEE Conference on Computational Complexity.

2003 M. Koucký. Universal traversal sequences for cycles of length $O(n^{4.03})$. In *Theoretical Computer Science*, 296(1), pp. 117–144. Special issue of selected papers from the 7th Annual International Computing and Combinatorics Conference.

2003 E. Allender, M. Koucký, D. Ronneburger, S. Roy. Derandomization and distinguishing complexity. In *Proceedings of the 18th IEEE Conference on Computational Complexity*, to appear.

2003 expected Ph.D. in Computer Science, Rutgers, The State University of New Jersey, New Brunswick, New Jersey.