A COMPLEXITY THEORY BASED ON INFINITELY OFTEN CONDITIONS

Jose Diaulas Palazzo Rolim

September 1986
CSD-860029

UNIVERSITY OF CALIFORNIA

Los Angeles

A Complexity Theory Based On Infinitely Often Conditions

A dissertation submitted in partial satisfaction of the
requirements for the degree Doctor of Philosophy
in Computer Science

by

José Diaulas Palazzo Rolim

1986

The dissertation of José Diaulas Palazzo Rolim is approved.

Herbert Enderton

D. Anthony Martin

Eliezer M. Gafni

Jack W. Carlyle

Sheila A. Greibach, Committee Chair

University of California, Los Angeles

1986

TABLE OF CONTENTS

ACKNOWLEDGEMENTS

A great debt of gratitude is owed to Professor S. Greibach whose guidance, assistance and new ideas made this research possible.

Furthermore, my appreciation is extended to Professor J. Carlyle for his support, especially on the probabilistic part of the work. The fruitful criticism of Professor E. Gafni is greatly appreciated. Also, my thanks to Professor H. Enderton and Professor D. A. Martin for their time in serving the committee and for their judgement of my work.

Finally, I would like to thank the Brazilian Comissão Nacional de Energia Nuclear, which grant made this research practicable.

VITA

December 20, 1956	Born, Cruzeiro, São Paulo, Brazil
1979	Electronics Engineer, University of São Paulo, Brazil
1980-1982	Research Assistant, FDTE, Brazil
1982	Master of Science in Electronics Engineering, University of São Paulo, Brazil

ABSTRACT OF THE DISSERTATION

A Complexity Theory Based On Infinitely Often Conditions

by

José Diaulas Palazzo Rolim

Doctor of Philosophy in Computer Science

University of California, Los Angeles, 1986

Professor Sheila A. Greibach, Chair

In this dissertation, we define a new model for complexity theory. By replacing the almost everywhere conditions of traditional complexity theory by infinitely often conditions, we define the IO-complexity.

We define IO-complexity classes of bound $f(n)$ with density function $d(n)$. We identify the IO-classes with density 1 to the worst-case classes. We establish the foundations of the new complexity theory by extending the results of the worst-case complexity to the IO-complexity.

We study time, space and density hierarchies of languages for deterministic and non-deterministic IO-complexity classes. These results when stated in terms of worst-case complexity are strengthenings of previous hierarchy results; they say that there is a language L computable in time $g(n)$ but every machine for L exceeds time $f(n)$ on every word of length n for infinitely many n. For space bounds, we show the existence of a language L computable in space $g(n)$ such that every machine for L can operate within space $f(n)$ only for a constant number of points.

We show that there exists positive density function $d(n)$ for which $P(d(n)) \neq NP(d(n))$ if and only if $P \neq NP$. On the other hand if there exists a positive density function $d(n)$ for which $P(d(n))=NP(d(n))$ then $E=NE$.

We show that a recursive language L is in a IO-complexity class of bound $f(n)$ with density $d(n)$ if and only if L can be approximated by $f(n)$ bounded machine agreeing with L on input w with probability at least $d(|w|)$.

We also show the relationship between the IO-complexity classes and some non-standard complexity classes. We relate the mean-case, the median-case and the probabilistic complexity classes to IO-complexity classes with density functions. Finally, we point out open questions related to the IO-complexity.

CHAPTER 1

INTRODUCTION

1-1 Motivation and Objectives

In the quotidian life of computing, it is not enough to know that a solution can or cannot be found; the critical question about the solution, if one exists, is: how much does the solution cost? The amount of time or space used may be traded off against the degree of approximation of the particular solution achieved to the ideal solution.

The achievement of a solution depends partly on our skill in computing and the sophistication of our computers, but there is also an additional factor which can be associated with the intrinsic difficulty of the problem itself [Cutl83]. The theory of computational complexity has been based on such aspects of computability theory.

The traditional approach in complexity theory deals with asymptotic definitions of performance measures, for example, the *worst* running time or *maximum* amount of space used among all possible computations. The *worst-case complexity* has its limitations, i.e., it is not a very accurate measure in the sense that some algorithms will be classified in that way as more expensive in theory than they are in practice. Quicksort is a classical example of an algorithm with a poor worst-case performance that is efficient on the average [Horo78]. These drawbacks have stimulated several attempts at developing an *average-case complexity* theory.

The study of average-case or expected-case complexity has been developed from two basic points of view [Yao77]. In the first one, the so called *distributional approach,* the input probability must be known and the theory is developed under these input assumptions [Levi84]. The second one, called the *randomized approach,* allows stochastic moves in the computation [Karp76].

In the analysis of performance of solutions it is customary to distinguish between the worst case and the expected behavior of an algorithm. This independence of approaches does not agree with our intuition, which suggests that both the average and the worst case behavior are so related that they should be governed by the same general laws. Several attempts have been made toward a more unified complexity theory [Yao77].

However, all the complexity theories defined until now have been based on *almost everywhere* conditions [Knut76] ; for example, running time of algorithms upper bounded for all inputs, except, perhaps, for a finite number of inputs in which the algorithm is allowed not to respect the bound. This fact suggests that, perhaps, we should have a more general complexity theory if, for example, we allow the running time of the algorithm to exceed the bound infinitely many times as long as it respects the bound infinitely often.

The main objective of this dissertation is the definition of a new approach to computational complexity theory. By defining notions of functions bounded *infinitely often* instead of *almost everywhere* and by defining *density functions* related to the number of points at which functions are bounded, we will try to enlarge the domain of complexity theory. These notions will lead to a new model of complexity theory, the *IO-complexity,* defined below.

1-2 Background

A *symbol* is an abstract entity. A *word* w is a finite sequence of symbols juxtaposed. The *length* of a word w, denoted $|w|$, is the number of symbols composing the word. The *empty* word, e, is the word consisting of zero symbols.

An *alphabet* Σ is a finite set of symbols. A *language* L is a set of words formed from symbols of an alphabet. The set of all possible words over a fixed alphabet Σ is denoted by Σ^*. The set Σ^+ denotes the set Σ^* minus the empty word. We denote by Σ^n the set of words of length n.

There have been many proposals for a precise mathematical characterization of the intuitive idea of computability. The remarkable result of investigation by many researchers is that each of the these definitions gives rise to the same class of functions. Furthermore, by *Church's thesis,* this class of functions coincides exactly with the notion of computable functions [Cutl83].

Today the Turing machine has become the accepted formalization of an effective procedure in complexity theory and we shall assume it as our formal model of computation. We will analyze this model from two point of views: the class of *languages* it defines and the class of integer *functions* it computes.

We assume standard models of Turing machines: deterministic and non-deterministic multitape off-line Turing machines. An off-line Turing machine is a Turing machine in which the input tape is read-only in two directions and the input tape head is not allowed to move off the input [Hopc79]. We suppose that the input is a word w limited by blank symbols and that the machine starts with all working tapes blank and all heads leftmost.

An instantaneous description (*ID*) of a Turing machine M is a compact notation for the state of M and for the input and current contents of the working tapes and the location of the tape heads of M. A computation $\alpha(w,M)$ is a sequence of *IDs* for a Turing machine M on input w, such that the sequence of moves defined by the sequence of *IDs* is feasible for machine M when its input tape contains the word w. Note that the last *ID* of $\alpha(w,M)$ contains the maximum number of working tape cells used in the computation.

Any Turing machine has a special set of states called *accepting* states. Whenever there exists a computation of a Turing machine M on input w that ends in an accepting state, we say that *M accepts w*. The set of all words w accepted by machine M constitutes the language accepted by M and we denote this set by $L(M)$.

We are concerned with the notion of time and space bounded computations and thus we use the standard definitions of *running time*, $T_M(w)$, of a deterministic Turing machine M on input w as the number of steps M takes before halting and of the *space* spent on input w, $S_M(w)$ as the maximum number of working tape cells used by M for input w in any computation. Obviously, $T_M(w)$ is undefined if M does not stop on input w.

We frequently impose a requirement of constructability on bounds. A function $f(n)$ is *time constructible* if there is a deterministic Turing machine M such that for each input w of length n, M halts in exactly $f(n)$ computation steps. The function $f(n)$ is *space constructible* if M scans exactly a total of $f(n)$ cells for each input w of length n on all working tapes.

When we turn to the non-deterministic Turing machine model there are various ways we could define time or space bounds. There are different senses in which a non-deterministic Turing machine could compute within time $T(w)$ or space $S(w)$. All computations for input w could halt in time $T(w)$. Or there could be some computations for input w which halt in time $T(w)$. Or all halting computations for input w could halt in time $T(w)$.

This variety of definitions is due to the fact that for a non-deterministic Turing machine there are several possible computations for a fixed input w. However the time for a fixed computation $\alpha(w,M)$ is uniquely defined- $time(\alpha(w,M))$ - as the number of steps taken in the sequence of actions defined by computation $\alpha(w,M)$. Analogously, the space- $space(\alpha(w,M))$- is defined as the number of working tape cells in the last *ID* of computation $\alpha(w,M)$. For input w, we select the definition of time and space bounds as follows.

Definition 1-2-1: Given a non-deterministic Turing machine M we define:

(i)The *running time* $T_M(w)$ of M on w as:

$$T_M(w)=\max \{time(\alpha(w,M)) \mid \alpha(w,M) \textit{ is a computation} \text{ for } w \text{ and } M\}$$

(ii)The *space spent* $S_M(w)$ of M on w as:

$$S_M(w)=\max \{space(\alpha(w,M)) \mid \alpha(w,M) \textit{ is a computation} \text{ for } w \text{ and } M\}$$

Definition 1-2-1 is usually called *operating* time or space in contrast to other definitions that are concerned with acceptance conditions. For deterministic Turing machine the various possible definitions are essentially the same, since a deterministic machine M defines a unique computation for any word w. Also if $T(w)$ or $S(w)$ are time or space constructible functions the various possible definitions are equivalent for non-deterministic Turing machines [Grei84].

1-3 The IO-complexity Definitions

Let Σ denote a finite input alphabet, X represent a random variable, w represent any word of Σ^* and n represent the length of word w. We want to relate X, w and n by some probability of the random variable X being word w, given that the length of w is n, i.e. $|w|=n$. This probability will be denoted by $P[X=w/n]$.

We suppose that the probability $P[X=w/n]$ is known and that the probability distribution is positive, i.e. every word w of length n has a non-null probability of occurrence.

The definitions of running time and space are concerned with computations for word w. In complexity theory, it is customary to associate time and space bounds with the *length n* of the word instead of the word itself. There can be several words with the same fixed length n. It is usual to select some particular criterion and choose time and space bounds which fits best the selected criterion. For instance, in the worst-case complexity, time for length n is defined as the maximum among the running times for words of that length. In the average case complexity, it is usual to take some kind of average over the running times for all words of a particular length.

In the above cases, we map all words of fixed length into a single value. The price for doing that is always some loss of information. We instead decide to define our complexity measure not as a single-valued function but as a mapping of words of fixed length into possible values. In other words, we use an auxiliary probabilistic quantity for length n that can assume all individual values for words of that length according to the occurrence probability of the word.

We say that M respects the bound $f(|w|)$ on input w if *all computations* of M on input w halt within $f(|w|)$ steps for time complexity or if *no computation* of M on w visits more than $f(|w|)$ working tape cells for space complexity. Thus, for example, we would like to say that a Turing machine is of IO-time complexity $f(n)$ with density function $d(n)$ and probability distribution $P[X=w/n]$ if the sum of the probabilities of all words of length n that respect the bound $f(n)$ is at least $d(n)$. More formally, we define the IO-complexity measure as follows.

Definition 1-3-1: Let $d(n)$ be a function such that $0 \leq d(n) \leq 1$ for all n and $d(n)>0$ infinitely often. Let M be a non-deterministic Turing machine.

(1) We say that M is a $f(n)$ *IO-time bounded* Turing machine (or of *IO-time complexity* $f(n)$) with *density function* $d(n)$ and probability distribution $P[X=w/n]$ if for all n

$$d(n) \leq \sum_{|w|=n:\, T_M(w) \leq f(n)} P[X=w/n]$$

(2) M is a $f(n)$ *IO-space bounded* Turing machine (or of *IO-space complexity* $f(n)$) with *density function* $d(n)$ and probability distribution $P[X=w/n]$ if for all n

$$d(n) \leq \sum_{|w|=n:\, S_M(w) \leq f(n)} P[X=w/n]$$

The sums $\sum_{|w|=n:\, T_M(w) \leq f(n)} P[X=w/n]$ and $\sum_{|w|=n:\, S_M(w) \leq f(n)} P[X=w/n]$ represent the probability distribution of time and space bounds for all words of length n for a Turing machine M. If some particular word w_p of length n is "important" in the sense that $P[X=w_p/n]$ is "large", then this will be reflected in the complexity measure by making the above sums weigh the values $T_M(w_p)$ and $S_M(w_p)$ with a large probability, namely $P[X=w_p/n]$. Thus, Definition 1-3-1 is a more precise measure of the complexity of the computation than other measures, like worst running time or space, that take in account just one particular value, which value can have a very small occurrence probability. Also expected values of complexity are limited when compared to Definition 1-3-1, because they do not give the full range of possible values for the bounds, only being an approximation for the complexity of the computation. We shall expect, therefore, that a *new complexity theory* based on the IO-complexity measures will not only include aspects of the worst-case and expected complexity, but must also be a more general complexity theory than theories based on the traditional complexity measures.

Based on the above definition, we can join languages into families of languages. Definition 1-3-1 is based on a particular probability distribution on the words of the input alphabet of machine M. However, when we consider languages over different alphabets we must consider several possible probability distributions. Thus, in the definition below, we consider a functor Φ that assigns to each possible alphabet a convenient probability distribution. We give some formal notation for new complexity classes as follows.

Definition 1-3-2: Let Φ be a functor assigning to each alphabet Σ a positive probability distribution $P[X=w/n]$ over Σ^*. Let $d(n)$ and $f(n)$ be functions in N such that $0 \le d(n) \le 1$ for all n. Then:

(i) $DSPACE(f(n),d(n),\Phi)$ is the class of languages recognized by deterministic Turing machines of IO-space complexity $f(n)$ with density function $d(n)$ and probability distribution $\Phi(\Sigma)$ for each alphabet Σ.

(ii) $NSPACE(f(n),d(n),\Phi)$ is the class of languages recognized by non-deterministic Turing machines of IO-space complexity $f(n)$ with density function $d(n)$ and probability distribution $\Phi(\Sigma)$ for each alphabet Σ.

(iii) $DTIME(f(n),d(n),\Phi)$ is the class of languages recognized by deterministic Turing machines of IO-time complexity $f(n)$ with density function $d(n)$ and probability distribution $\Phi(\Sigma)$ for each alphabet Σ.

(iv) $NTIME(f(n),d(n),\Phi)$ is the class of languages recognized by non-deterministic Turing machines of IO-time complexity $f(n)$ with density function $d(n)$ and probability distribution $\Phi(\Sigma)$ for each alphabet Σ.

1-4 Notation

Notice that the pattern of the complexity classes is the same and meant to be mnemonic:

$$XBOUND$$

for the classes of languages accepted by deterministic, if $X=D$, or by non-deterministic, if $X=N$, Turing machines with bounded space, if $BOUND=SPACE$, or with bounded time, if $BOUND=TIME$.

We are going to use $XBOUND(f(n),d(n))$ to denote the union of the complexity classes $XBOUND(f(n),d(n),\Phi)$ for all functors Φ that assign to each input alphabet Σ a positive probability distribution. In particular, we denote the uniform distribution, i.e. $P[X=w/n]=\frac{1}{|\Sigma|^n}$ by U, and thus, for example, $DTIME(f(n),d(n),U)$ is the class of languages $L(M)$ accepted by deterministic Turing machine M of IO-time complexity $f(n)$ with density $d(n)$ and uniform distribution for the input alphabet of M.

We use the symbol M to denote Turing machines and L to denote languages. The language accepted by Turing machine M is denoted by $L(M)$. We use the letters T and S to denote running time and space of the Turing machine under consideration. If confusion can occur, then we use the symbols T_M and S_M to denote the running time and space of the particular machine M.

We reserve the symbols $f(n)$ and $g(n)$ to functions from N to N, with N the set of natural numbers. We also imply that $d(n)$ denotes density functions, that is, $0 \le d(n) \le 1$ for all n, $d(n)>0$ infinitely often. We say that $d(n)$ is *positive* almost everywhere if $d(n)>0$ almost everywhere. The density function $d(n)$ is *positive* if $d(n)>0$ for all n.

1-5 Overview of the Dissertation

The complexity theory developed here is called IO-complexity, since it is based on conditions that are met infinitely often. This chapter introduced the IO-complexity model formally. We started by presenting our assumptions about the computer model chosen and its background. We defined time and space bounds for deterministic and non-deterministic Turing machines related to the IO-definitions.

We also defined the IO-complexity classes denoted by $XBOUND(f(n), d(n))$.

In chapter 2, we identify the density function $d(n)=1$ case with the worst-case complexity. Theorem 2-2-1 says that $XBOUND(f(n), 1)=XBOUND(f(n))$, therefore, incorporating the worst-case complexity to the IO-complexity. Furthermore, we show that most results of the worst-case complexity can be extended to the IO-complexity. Theorem 2-2-5 establishes the foundations for a complexity theory based on IO-conditions, since it can be used as fundamental lemma to translate results from the worst-case complexity theory to the IO-complexity theory. As a consequence of this result, we derive properties for the IO-complexity such as speed up, Corollary 2-2-6; tape compression, Corollary 2-2-7; deterministic simulation of space bounded non-deterministic machines, Corollary 2-2-8; tape reductions, Corollaries 2-3-1 to 2-3-6; and other useful relations related to time and space bounds, Corollaries 2-2-9 to 2-2-11. Finally, in chapter 2, we study the effect of tape reductions in machines infinitely often bounded with density function $d(n)$.

In chapter 3, we study the structure of the complexity classes. We develop hierarchies of languages for the IO-complexity. We investigate time and space hierarchies for deterministic and non-deterministic IO-classes of languages. Theorem 3-2-1 shows the existence of languages recognized in time $g(n)$ with density 1 that cannot be recognized in time $f(n)$ with any density function $d(n)$, such that $d(n)$ is positive almost everywhere, and functions $f(n)$ and $g(n)$ related by $\inf_{n \to \infty} \frac{g(n)}{f(n)} = \infty$. For space bounds, Theorem 3-3-4 shows the existence of languages accepted within space bound $g(n)$ that cannot be accepted with IO-space bound $f(n)$ and density function $d(n)$, with $d(n)$ positive infinitely often. Theorems 3-2-1 and 3-3-4 when stated in terms of worst-case complexity are strengthenings of the basic hierarchy results. Theorem 3-2-1 says that for functions $f(n)$ and $g(n)$ related by $\inf_{n\to\infty} \frac{g(n)}{f(n)}=\infty$ there is a language L computable in time $g(n)$, but every machine for L exceeds time $f(n)$ on every word of length n for infinitely many n. Theorem 3-3-4 is even stronger, it asserts the existence of a language L computable in space $g(n)$ such that every machine for L can only respect space bound $f(n)$ for a constant number of points.

We also show that for a fixed bound $f(n)$, a decrease in the density function allows more languages to be recognized; that is there are languages in $XBOUND(f(n), d_1(n))$ that cannot be in $XBOUND(f(n), d_2(n))$ for density functions $d_1(n)$ and $d_2(n)$ such that the difference between $d_1(n)$ and $d_2(n)$ is at least $max\{P[X=w/n]: |w|=n\}$.

In chapter 4, we make conjectures about the deterministic and non-deterministic polynomial time classes, P and NP, and about the deterministic and non-deterministic exponential time classes, E and NE. We enlarge the definitions of these classes to embody density $d(n)$. Theorem 4-2-4 shows that there exists a positive density function $d(n)$ for which $P(d(n)) \neq NP(d(n))$ if and only if $P \neq NP$. On the other hand, Theorem 4-3-1 shows that the existence of any positive density function $d(n)$ for which $P(d(n))=NP(d(n))$ implies $E=NE$. Still in chapter 4, we point out the different nature of the density functions and of the oracle computations. We also enlarge the polynomial space classes by incorporating the concept of density functions; we denote this class by $PSPACE(d(n))$.

We also give formal definitions for the notion of finding an approximate solution for a hard problem. We give an interpretation of the IO-complexity classes in terms of classes of languages that have approximate solutions. We show that the recursive languages of $DBOUND(f(n), d(n))$ are those languages L which can be approximated by $f(n)$ bounded machines agreeing with L on input w with probability at least $d(|w|)$. In section 4-6, we prove that there are problems so hard that they do not admit even such approximate solutions.

In chapter 5, we discuss the extension of the ideas of IO-complexity to different types of complexity classes. We define mean and median complexity classes. We expand the probabilistic polynomial classes R, BPP and PP to include density functions. We show the inclusion relations among these probabilistic classes and the classes $P(d(n))$, $NP(d(n))$ and $PSPACE(d(n))$. We point out that this research area is still being exploited and several questions are unanswered.

We conclude, in chapter 6, by pointing out additional problems deserving further investigation and we give conjectures on open questions that appear in this dissertation.

CHAPTER 2

THE COMPLEXITY MODEL

2-1 Introduction

In this chapter, we study general properties that are valid for the IO-complexity model. We use the basic definitions of section 1-3.

We start by analyzing the relationship between the IO-complexity classes and the classes of the worst-case complexity theory. We identify IO-complexity classes with density function 1 to the worst-case complexity classes.

In section 2-3, we establish the theoretical foundations for the IO-complexity model. We show that containments relations of the worst-case complexity classes translate into equivalent relations among the IO-complexity classes. For example, we show that if $DSPACE(f(n))$ is contained in $DSPACE(g(n))$, then $DSPACE(f(n),d(n))$ is contained in $DSPACE(g(n),d(n))$ for any density function $d(n)$ and any monotonic increasing space constructible function $f(n)$. We show similar results for non-deterministic machines and for time and space bounds. We prove these results using translational techniques [Hopc79].

The last section of this chapter deals with tape reductions results. We show that for IO-time bounds the number of working tapes can be reduced at cost of an increase of the IO-time bound. For IO-space bounds, we show that the number of working tapes can be reduced to only one working tape without increasing the IO-space bound.

2-2 Worst-case Complexity

Consider a Turing machine M and let $T(w)$ denote the running time of M on input w. We can, for the worst-case complexity, define the functions:

$$T_{\max}(n)=\max\ \{T(w):\ |w|=n\}$$

$$S_{\max}(n)=\max\ \{S(w):\ |w|=n\}$$

The family of languages accepted by deterministic (non-deterministic) multitape off-line Turing machines for which $T_{\max}(n)\le f(n)$ is called $DTIME(f(n))$ ($NTIME(f(n))$). The family of languages accepted by deterministic (non-deterministic) multitape off-line Turing machines for which $S_{\max}(n)\le f(n)$ is called $DSPACE(f(n))$ ($NSPACE(f(n))$).

Notice that the pattern of the names of complexity classes is the same and meant to be mnemonic:

$$XBOUND(f(n))$$

for the class of languages accepted by X multitape off-line Turing machines within $BOUND$ f. Also let $R(w)$ denote $T(w)$ or $S(w)$, $R_{\max}(n)$ denote $T_{\max}(n)$ or $S_{\max}(n)$, whether we are talking about time or space, respectively.

We would like to relate the new complexity classes just defined to the traditional worst-case complexity classes $XBOUND(f(n))$. The next theorem does that.

Theorem 2-2-1:

$$L \in XBOUND(f(n)) \text{ if and } \textit{only} \text{ if } L \in XBOUND(f(n), 1)$$

Proof: Suppose $L \in XBOUND(f(n))$. Let Σ be the input alphabet and let $P[X=w/n]$ be any positive probability distribution. Then there is a Turing machine accepting L for which $R_{\max}(n) \le f(n)$ for all n and therefore $R(w) \le R_{\max}(n) \le f(n)$ for all $w \in \Sigma^n$. Thus:

$$\sum_{|w|=n:\, R(w) \le f(n)} P[X=w/n] = \sum_{|w|=n} P[X=w/n] = 1$$

since $P[X=w/n]$ is positive. Therefore $L \in XBOUND(f(n), 1)$.

Conversely, suppose that $L \in XBOUND(f(n), 1)$. Then there is a Turing machine M for which:

$$1 \le \sum_{|w|=n:\, R(w) \le f(n)} P[X=w/n] \le 1 \textit{ for all } n$$

Thus $\sum_{w \in \Sigma^n:\, R(w) \le f(n)} P[X=w/n] = 1$.

Since $P[X=w/n]$ is positive, every input w of length n has $P[X=w/n] \ne 0$. Thus, in order to get the sum equal to 1, all w must met the condition $R(w) \le f(|w|)$. Then $R_{\max}(n) \le f(n)$ for M. Thus $L \in XBOUND(f(n))$ $\square$

Corollary 2-2-2:

$$DTIME(f(n)) = DTIME(f(n), 1)$$

Corollary 2-2-3:

$$NTIME(f(n)) = NTIME(f(n), 1)$$

Corollary 2-2-4:

$$DSPACE(f(n)) = DSPACE(f(n), 1)$$

Corollary 2-2-5:

$$NSPACE(f(n)) = NSPACE(f(n), 1)$$

The above corollaries simply say that the worst-case complexity corresponds to the IO-complexity with density function 1, for any positive probability distribution. Theorem 2-2-1 does not depend on the particular probability distribution selected and it allow us to use all the results of traditional complexity theory for the IO-complexity classes with density 1.

2-3 Complexity Results For Density d(n)

Theorem 2-2-1 relates completely IO-complexity classes with density 1 to the worst-case complexity classes, thus extending all results of the worst-case to the density 1 case. This section deals with the question of which results of traditional complexity theory can be extended to density functions not necessarily 1.

From a language L recognized by a machine M of IO-complexity $f(n)$ with some density function $d(n)$, we want to define another language $LDT(M,f)$ accepted by machine M' *always* bounded by $f(n)$. Furthermore, we also want that from language $LDT(M,f)$ we have the capacity of recognizing in bound $f(n)$ the words w accepted/rejected by machine M that respect the bound $f(|w|)$.

Given any property of the worst-case complexity theory respected by some machine accepting language $LDT(M,f)$, we show that there is a machine accepting language L that respects the same property for those words w for which M respects the bound $f(|w|)$. We are going to use a variant of standard padding arguments, also known as translational techniques [Hopc79]. We start with the case of deterministic time.

Definition 2-3-1: Let $f(n)$ be a monotonic increasing function. Let M be a deterministic Turing machine with input alphabet Σ. Let $T(w)$ be the running time of M on w. For each $a \in \Sigma$ consider a new symbol $new(a) \notin \Sigma$. Also let c be another symbol different from any a or $new(a)$, $a \in \Sigma$. We define:

$$LDT(M,f) = \{wc^i : w \in L(M) \ \& \ T(w) \le f(|w|+i)\}$$

$$\cup$$

$$\{u.new(a) : ua \in \Sigma^*, \ |a|=1 \ \& \ T(ua) > f(|ua|)\}.$$

Notice that there is a correspondence between words w accepted by M and words wc^i in $LDT(M,f)$. If a word w is accepted by M, then w is padded with as many symbols c as necessary to achieve $T(w) \le f(|w|+i)$.

The second part of $LDT(M,f)$ makes it possible to identify the words rejected by M for which M respects the bound $T(w) \le f(|w|)$. This set of words is the set of words ua such that $u.new(a)$ is not in $LDT(M,f)$. Note that if M does not respect the bound for the empty word e and if e is not in $L(M)$, then e can never appear in $LDT(M,f)$ as $u.new(a)$. However, the word e can be treated separately with the answer of the computation of M on e being stored in the finite control of the machines defined below.

The next lemma makes use of $LDT(M,f)$ in order to prove that the results of complexity theory for deterministic time can be extended to the IO-complexity theory.

Lemma 2-3-1: Let $f(n)$ be a monotonic increasing time constructible function such that $\inf_{n\to\infty} \frac{f(n)}{n}=\infty$. Let $g(n)$ be a function such that $\inf_{n\to\infty} \frac{g(n)}{n}=\infty$.

Then:

$$DTIME(f(n)) \subseteq DTIME(g(n))$$

implies:

$$DTIME(f(n),d(n)) \subseteq DTIME(g(n),d(n))$$

Proof: Let M be a k_1-tape deterministic machine of IO-time complexity $f(n)$ with density function $d(n)$. Let $T(w)$ be the running time of w for M. Let $|w|=n$. We build a deterministic machine M_1 that accepts $LDT(M,f)$, the language specified in definition 2-3-1. Let $NEW=\{new(a)\colon a\in\Sigma\}$ and Σ' denote the set $\Sigma\cup\{new(a)\colon a\in\Sigma\}\cup\{c\}$. Let M_1 behave on input $y\in\Sigma'$ as follows.

(i) If $y\notin\Sigma^* c^* \cup \Sigma^* NEW$, then M_1 rejects y.

If $y=wc^i$, then M_1 behaves as follows.

(ii) It counts up to $f(|w|+i)$ using tapes T_j, $j\geq k_1$, which is possible since f is time constructible, and simultaneously:

(iii) Simulates M on input w using tapes T_1 to T_{k_1}, for up to $f(|w|+i)$ steps:

-If M accepts w, then M_1 accepts y.

-If M rejects w or reaches no decision on w within time $f(|w|+i)$, then M_1

rejects y.

If $y = u.new(a)$, $u \in \Sigma^*$, then M_1 behaves as follows.

(iv) It counts up to $f(|y|)$ using tapes T_j and simultaneously:

(v) Simulates M on $u.a$ for up to $f(|y|)$ steps, accepting y if and only if M does not halt within time $f(|y|)$.

The language accepted by M_1 is the set of words accepted by M_1 in step (iii) or step (v). The words accepted in (iii) are the words of the form wc^i, for which M accepts w within time $f(|w|+i)$. The words accepted by M_1 in (v) can be described as the words of the form $u.new(a)$ such that M does not halt within time $f(|ua|)$ on input ua. Thus:

$$L(M_1) = \{wc^i : w \in L(M) \ \& \ T(w) \leq f(|w|+i)\}$$

$$\cup$$

$$\{u.new(a) : T(ua) > f(|ua|)\} = LDT(M,f)$$

Consider any word y of size m. Step (i) takes at most $2m$ steps, since we just need to read the input and check if it is of the form $y = wc^i$ or $u.new(a)$ and then back up to get ready for the next actions. But by hypothesis $\inf_{n \to \infty} \frac{f(n)}{n} = \infty$ and thus step (i) is bounded by $f(m)$ almost everywhere for any such y. Steps (ii) and (iv) are obviously bounded by $f(m)$, since f is time constructible. Also, the presence of the counter guarantees that the simulation, steps (iii) and (iv), will take at most $f(m)$ steps. Then $LDT(M,f) \in DTIME(3f(n))$. But by the linear speed-up result for the worst-case complexity, $LDT(M,f) \in DTIME(f(n))$ [Hopc79].

Thus, $LDT(M,f) \in DTIME(g(n))$ since by hypothesis $DTIME(f(n)) \subseteq DTIME(g(n))$. Applying again the linear speed-up result for the worst-case complexity, we get a deterministic machine M' with k_2 tapes that recognizes $LDT(M,f)$ in time $\frac{g(n)}{3}$. We build a deterministic machine M_2 that acts on input $w=ua$, $w \in \Sigma^*$, $a \in \Sigma$ as follows.

(i) Simulate machine M' on input w.

- If M' accepts w, then M_2 accepts w.

(ii) Otherwise, simulate M' on $u.new(a)$.

- If M' rejects $u.new(a)$, then M_2 rejects w.

(iii) Otherwise, simulate M' on input wc^i, $i=1,2...$ until M' reaches a decision on wc^i. M_2 accepts w if and only if M' accepts wc^i.

The language accepted by M_2 is the set of words accepted in (i) and in (iii). The words w accepted by M_2 in (i) are the words w in Σ^+ that belong to $L(M')=LDT(M,f)$. The set of words w accepted in (iii) are the words w, which are accepted by machine M in more than $f(|w|)$ time steps. More formally:

$$\{w : w \in LDT(M,f),\ w \in \Sigma^*\}$$

$$\cup$$

$$\{w : w \in \Sigma^*,\ \exists i\ wc^i \in LDT(M,f)\}$$

$$= \{w : w \in L(M)\ and\ \exists i\ T(w) \leq f(|w|+i)\}$$

But $T(w) \leq f(|w|+i)$ for some i for any $w \in L(M)$, since f is monotonic increasing. Thus $L(M_2) = L(M)$.

Consider the original machine M and consider any input w for which M respects the bound, i.e. $T(w) \leq f(|w|)$. If $w \in L(M)$ and $T(w) \leq f(|w|)$, then w will be in $LDT(M,f)$. But then machine M_2 will accept w in step (i). This action is bounded by $\frac{g(n)}{3}$ since $L(M') \in DTIME(\frac{g(n)}{3})$.

Similarly if $w = u.a \notin L(M)$ and $T(ua) \leq f(|ua|)$ then $u.new(a) \notin LDT(M,f) = L(M')$. So $u.new(a)$ is rejected by M' in at most $\frac{g(n)}{3}$ steps, since M' is of worst-case complexity $\frac{g(n)}{3}$. Thus w is rejected by M_2 in step (ii).

Therefore M_2 halts for all w for which M respects the bound before (iii). But (i) costs at most $\frac{g(n)}{3}$ computation steps. Step (ii) requires the writing of $u.new(a)$ on some working tape; this action is bounded by $\frac{g(n)}{3} \geq n$, since $\inf_{n \to \infty} \frac{g(n)}{n} = \infty$ implies $\frac{g(n)}{k} \geq n$ almost everywhere. The simulation on (ii) costs at most $\frac{g(n)}{3}$ steps. Thus the sum of steps (i) and (ii) is bounded by $g(n)$ computation steps, i.e. $T_2(w) \leq g(|w|)$ for these words.

Thus all words w that respect the bound $T(w) \leq f(|w|)$ for machine M have $T_2(w) \leq g(|w|)$ for machine M_2. So for all n:

$$\sum_{|w|=n : T_2(w) \leq g(n)} P[X = w/n] \geq \sum_{|w|=n : T(w) \leq f(n)} P[X = w/n] \geq d(n)$$

But then $L(M) \in DTIME(g(n), d(n))$. $\square$

A variant of the argument used in Lemma 2-3-1 shows the analogous result for *NTIME*. It seems impossible to find out if all computations of a non-deterministic Turing machine M on input w are bounded by $f(|w|)$, if machine M is not allowed to spend more than time $f(|w|)$. We avoid this feature, implicit in definition 2-3-1, by defining a language $LTN(M,f)$ which takes into consideration the time spent on each computation on input w by machine M instead of running time of M on input w.

Definition 2-3-2: Let $f(n)$ be a monotonic increasing function. Let M be a non-deterministic Turing machine with input alphabet Σ. Let $T(w)$ be the running time of M on w and $\alpha(w,M)$ denote a computation of M on input w. For each $a \in \Sigma$ consider a new symbol $new(a) \notin \Sigma$. Also let c be another symbol different from any a or $new(a)$, $a \in \Sigma$. We define:

$$LNT(M,f) = \{wc^i : w \in L(M) \ \& \ \exists \ accepting \ \alpha(w,M), time(\alpha(w,M)) \leq f(|w|+i)\}$$

$$\cup$$

$$\{u.new(a) : ua \in \Sigma^*, \ |a|=1 \ \& \ \exists \alpha(ua,M), time(\alpha(ua,M)) > f(|ua|)\}.$$

Notice that for deterministic machine M given an input w there is only one computation α, thus for such a machine $LDT(M,f)$ and $LNT(M,f)$ are equivalent.

Lemma 2-3-2: Let $f(n)$ be a monotonic increasing time constructible function such that $\inf_{n \to \infty} \frac{f(n)}{n} = \infty$. Let $g(n)$ be a function such that $\inf_{n \to \infty} \frac{g(n)}{n} = \infty$.

Then:

$$NTIME(f(n)) \subseteq NTIME(g(n))$$

implies:

$$NTIME(f(n), d(n)) \subseteq NTIME(g(n), d(n))$$

Proof: The proof is similar to the proof of Lemma 2-3-1, so we follow that notation. To go from L to $LDT(M,f)$, we build machine M_1 which behaves as follows.

(i) If $y \notin \Sigma^* c^* \cup \Sigma^* NEW$, then M_1 rejects y.

If $y = wc^i$, then M_1 behaves as follows.

(ii) It counts up to $f(|w|+i)$ using tapes T_j, $j \geq k_1$ and simultaneously:

(iii) It non-deterministically simulates the behavior of M on input w for up to $f(|w|+i)$ steps using tapes T_1 to T_{k_1}.

-If M accepts w, then M_1 accepts y.

-If M rejects w or reaches no decision on w within time $f(|w|+i)$, then M_1 rejects y.

If $y = u.new(a)$, $u \in \Sigma^*$, then M_1 behaves as follows.

(iv) It counts up to $f(|y|)$ in tapes T_j and simultaneously:

(v) It non-deterministically simulates M on $u.a$ for up to $f(|y|)$ steps, accepting y if and only if M does not halt within time $f(|y|)$.

Notice that M_1 is a non-deterministic machine because M is non-deterministic. The language accepted by M_1 is the set of words accepted in step (iii) or in step (v). The words accepted in (iii) are the words wc^i for which M has at least one accepting computation on w within time $f(|w|+i)$. The words accepted in (v) are the words $u.new(a)$ for which M has at least one computation on $u.a$ with more than $f(|ua|)$ steps. Thus:

$$L(M_1) = \{wc^i : w \in L(M) \ \& \ \exists \ accepting \ \alpha(w,M), time(\alpha(w,M)) \leq f(|w|+i)\}$$

$$\cup$$

$$\{u.new(a) : ua \in \Sigma^*,\ |a|=1\ \&\ \exists\alpha(ua,M), time(\alpha(ua,M)) > f(|ua|)\} = LNT(M,f)$$

In the worst case any computation of M_1 is bounded by $3f(n)$, the sum of steps (i) to (iii) for inputs of the form wc^i or the sum of steps (i), (iv) and (v) for inputs of the form $u.new(a)$. Thus $LNT(M,f) \in NTIME(3f(n))$. But by the linear speed-up result of the worst-case complexity $LNT(M,f) \in NTIME(f(n))$ [Hopc79].

Thus, $LNT(M,f) \in NTIME(g(n))$ since by hypothesis $NTIME(f(n)) \subseteq NTIME(g(n))$. Applying again the linear speed-up result, we get a non-deterministic machine M' with k_2 tapes that recognizes $LNT(M,f)$ in time $\frac{g(n)}{3}$. We build a non-deterministic machine M_2 that acts on input w as follows.

(i) Non-deterministically simulate machine M' on input w.

- If M' accepts w, then accept w.

(ii) Otherwise, non-deterministically simulate M' on $u.new(a)$, $w=ua$.

- If M' rejects $u.new(a)$, then reject w.

(iii) If M' accepts $u.new(a)$, then simulate M' on input wc^i, $i=1,2\ldots$ until M' reaches a decision on wc^i. M_2 accepts w if and only if M' accepts wc^i.

The language accepted by M_2 is the set of words accepted in (i) or (iii). That is:

$$\{w : w \in LNT(M,f),\ w \in \Sigma^*\}$$

$$\cup$$

$$\{w : w \in \Sigma^*,\ \exists i\ wc^i \in LNT(M,f)\}$$

$= \{w : w \in L(M) \text{ and } \exists i, \exists \text{ accepting } \alpha(w,M),\ time(\alpha(w,M)) \leq f(|w|+i)\}$

But $time(\alpha(w,M)) \leq f(|w|+i)$ for some i for any $w \in L(M)$, since f is monotonic increasing. Thus $L(M_2) = \{ w : w \in L(M) \} = L(M)$.

Furthermore if w is accepted by M with running time $T(w) \leq f(|w|)$, then all computations of M on w must halt within time $f(|w|)$. Thus $w \in L(M')$. Thus all computations of M' on w must end within time $\frac{g(|w|)}{3}$, with at least one accepting computation, since w is in $L(M')$. Therefore this accepting computation will be simulated by machine M_2 and will make M_2 accept w in step (i). Furthermore the rejecting computations of M' on w will result in the rejection of w in step (ii). Thus M_2 accepts w within time bounded by steps (i) and (ii).

Similarly if $w = u.a \notin L(M)$ and $T(w) \leq f(n)$ then $u.new(a)$ is rejected by M' in all computations within time $\frac{g(n)}{3}$ steps. Thus all computations for $u.new(a)$ are rejecting and within the bound. Then w is rejected by M_2 in step (ii).

Therefore M_2 halts for all w for which M respects the bound in (i) or (ii). But (i) costs at most $\frac{g(n)}{3}$ computation steps. Step (ii) requires the writing of $u.new(a)$ on some working tape; this action is bounded by $\frac{g(n)}{3} \geq n$, since $\inf_{n \to \infty} \frac{g(n)}{n} = \infty$ implies $\frac{g(n)}{k} \geq n$ almost everywhere. The simulation on (ii) costs at most $\frac{g(n)}{3}$ steps. Thus the sum of steps (i) and (ii) is bounded by $g(n)$ computation steps. Therefore all words w for which machine M respects the bound $f(|w|)$ have the bound $g(|w|)$ respected by machine M_2. Thus $L(M_2) \in NTIME(g(n), d(n))$. □

We want to use the same kind of argument for space bounds. Thus we define language $LDS(M,f)$ as follows.

Definition 2-3-3: Let $f(n)$ be a monotonic increasing function. Let M be a deterministic Turing machine with input alphabet Σ. Let $S(w)$ be the working-tape space spent on w by M. For each $a \in \Sigma$ consider a new symbol $new(a) \notin \Sigma$. Also let c be another symbol different from any a or $new(a)$, $a \in \Sigma$. We define:

$$LDS(M,f) = \{wc^i : w \in L(M) \;\&\; S(w) \leq f(|w|+i)\}$$

$$\cup$$

$$\{u.new(a) : ua \in \Sigma^*,\ |a|=1 \;\&\; S(ua) > f(|ua|)\}.$$

There are analogous results to Lemma 2-3-1 and Lemma 2-3-2 for $DSPACE$ and $NSPACE$. The arguments are similar, with the difference that we talk about $LDS(M,f)$ instead of $LDT(M,f)$ and the counters deal with visited cells on the working tapes instead of steps. Also the requirement of $\inf_{n\to\infty} \frac{f(n)}{n} = \infty$ can be dropped, because the tape compression result of the worst-case complexity does not require it [Grei84].

Lemma 2-3-3: Let $f(n)$ be a monotonic increasing space constructible function.

Then:

$$DSPACE(f(n)) \subseteq DSPACE(g(n))$$

implies:

$$DSPACE(f(n), d(n)) \subseteq DSPACE(g(n), d(n))$$

Proof: We follow the general technique of the former lemmas. Now, given a machine M operating in IO-space complexity $f(n)$ with density $d(n)$, we construct a deterministic machine M_1 recognizing $LDS(M,f)$ within space bound f. From $LDS(M,f)$ we build a deterministic machine M_2 to accept $L(M)$ within IO-space bound $f(n)$ with density $d(n)$. We start by defining machine M_1 as follows.

(i) If $y \notin \Sigma^* c^* \cup \Sigma^* NEW$, then M_1 rejects y.

If $y = wc^i$, then M_1 behaves as follows.

(ii) Lay off $f(|w|+i)$ cells in tape T_0. This can be done within space $f(n)$ since f is space constructible.

(iii) Simulate M on input w using tapes T_1 to T_{k_1} and at most $f(|w|+i)$ working tape cells. At each new cell visited in tapes T_1 to T_{k_1}, the head of T_0 moves right. The simulation is interrupted if the head of T_0 reaches the rightmost blank symbol of T_0. This guarantees that the simulation uses at most $f(|w|+i)$ cells in tapes T_j, $j \geq 1$. M_1 accepts y if and only if M accepts w.

If $y = u.new(a)$, $u \in \Sigma^*$, then M_1 behaves as follows.

(iv) Lay off $f(|w|)$ cells in T_0.

(v) Simulate a computation of M on $u.a$, checking that the number of cells used does not exceed $f(|y|)$. Accept y if and only if M tries to use more than $f(y)$ cells.

The language accepted by M_1 is:

$$\{wc^i : w \in L(M) \ \& \ S(w) \leq f(n+i)\}$$

$$\cup$$

$$\{u.new(a) : u \in \Sigma^*,\ a \in \Sigma, S(ua) > f(|ua|)\} = LDS(M,f)$$

Furthermore M_1 uses at most $2f(n)$ cells, sum of steps (ii) and (iii) for inputs of the form wc^i, $n=|wc^i|$, or sum of steps (iv) and (v) for inputs of the form $u.new(a)$, $n=|ua|$. Thus $LDS(M,f) \in DSPACE(2f(n))$ and therefore, due to the tape compression result for the worst-case complexity, $LDS(M,f) \in DSPACE(f(n))$.

Thus, $LDS(M,f) \in DSPACE(g(n))$ since by hypothesis $DSPACE(f(n)) \subseteq DSPACE(g(n))$. Applying the tape compression result, we get a deterministic machine M' with k_2 tapes that accepts $LDS(M,f)$ within space $\frac{g(n)}{2}$. We build a deterministic machine M_2 that acts on input w in Σ^* as follows.

(i) Simulate machine M' on input w.

- If M' accepts, w then accept w.

(ii) Otherwise, simulate M' on $u.new(a)$, $w=ua$, $a \in \Sigma$.

- If M' rejects $u.new(a)$, then reject w.

(iii) Otherwise, simulate M' on input wc^i, $i=1,2...$ until M' reaches a decision on wc^i. M_2 accepts w if and only if M' accepts wc^i.

The language accepted by M_2 is the language:

$$\{w \in \Sigma^* : w \in LDS(M,f)\}$$

$$\cup$$

$$\{w \in \Sigma^* : \exists i,\ wc^i \in LDS(M,f)\}$$

$$= \{w : w \in L(M) \ and\ \exists i,\ S(w) \leq f(|w|+i)\}$$

But $S(w) \leq f(|w|+i)$ for some i for any $w \in L$, since f is monotonic increasing. Thus

$L(M_2)=L(M)$.

Consider the original machine M. Let w be a word of length n that respects the bound f, i.e. $S(w)\leq f(|w|)$. If $w\in L(M)$ and $S(w)\leq f(|w|)$, then w will be in $LDS(M,f)$. But then machine M_2 will accept w in step (i). This action is bounded by $\frac{g(n)}{2}$ since $L(M')\in DSPACE(\frac{g(n)}{2})$. Similarly if $w=u.a\notin L(M)$ and $S(ua)\leq f(|ua|)$, then $u.new(a)$ is rejected by M' using at most $\frac{g(n)}{2}$ steps. Thus w is rejected by M_2 in step (ii). So M_2 simulates M' only through (ii) for all w for which M respects the bound. But (i) costs at most $\frac{g(n)}{2}$ working cells. The simulation on (ii) costs at most $\frac{g(n)}{2}$ new working tape cells. Thus the sum of steps (i) and (ii) is bounded by $g(n)$ working tape cells.

Therefore for all words w for which M respects the bound $S(w)\leq f(|w|)$ have $S_2(w)\leq g(|w|)$ where S_2 is the space function for machine M_2.

$$\sum_{|w|=n:S_2(w)\leq g(n)} P[X=w/n] \geq \sum_{|w|=n:S(w)\leq f(n)} P[X=w/n] \geq d(n)$$

But then $L(M)\in DSPACE(g(n),d(n))$. □

For non-deterministic space bounded machines, we have a more complicated definition of an auxiliary language $LNS(M,f)$.

Definition 2-3-4: Let $f(n)$ be a monotonic increasing function. Let M be a non-deterministic Turing machine with input alphabet Σ. Let $S(w)$ be the space of M on w and $\alpha(w,M)$ denote a computation of M on input w. For each $a\in\Sigma$ consider a new symbol $new(a)\notin\Sigma$. Also let c be another symbol different from any a or $new(a)$, $a\in\Sigma$. We define:

$$LNS(M,f) = \{wc^i : w \in L(M) \ \& \ \exists \ accepting \ \alpha(w,M),\ space(\alpha(w,M)) \leq f(|w|+i)\}$$

$$\cup$$

$$\{u.new(a) : ua \in \Sigma^*, |a|=1 \ \& \ \exists \alpha(ua,M),\ space(\alpha(ua,M)) > f(|ua|)\}.$$

Lemma 2-3-4: Let $f(n)$ be a monotonic increasing space constructible function.

Then:

$$NSPACE(f(n)) \subseteq NSPACE(g(n))$$

implies:

$$NSPACE(f(n),d(n)) \subseteq NSPACE(g(n),d(n))$$

Proof: We follow the notation of former lemmas. Consider M a non-deterministic machine of IO-space complexity $f(n)$ with density $d(n)$. We construct a non-deterministic machine M_1 to accept $LNS(M,f)$ as follows.

(i) If $y \in \Sigma^* c^* \cup \Sigma^* NEW$, then M_1 rejects y.

If $y = wc^i$, then M_1 behaves as follows.

(ii) Deterministically lay off $f(|w|+i)$ cells in tape T_0.

(iii) Non-deterministically simulate the behavior of M on input w using tapes T_1 to T_{k_1} and at most $f(|w|+i)$ cells. At each new cell visited in tapes T_1 to T_{k_1} the head of T_0 moves right. If M_1 tries to use more than $f(|w|+i)$ cells the simulation is interrupted. M_1 accepts y if and only if M accepts w within the space bound.

If $y = u.new(a)$, $u \in \Sigma^*$, then M_1 behaves as follows.

(iv) Deterministically lay off $f(|w|)$ cells in T_0.

(v) Non-deterministically simulate the behavior of M on $u.a$, checking that the number of cells used does not exceed $f(|y|)$. Accept y if and only if M tries to use more than $f(|y|)$ cells.

The language accepted by M_1 is composed of the sets of words accepted in (iii) and in (v). The words accepted in (iii) can be described as the words wc^i for which M has an accepting computation on w within space $f(|w|+i)$. The words accepted in (v) are the words $u.new(a)$ for which M has at least one computation on ua using more than $f(|ua|)$ cells. Thus:

$$L(M_1) = \{wc^i : w \in L(M) \;\&\; \exists \; accepting \; \alpha(w,M),\; space(\alpha(w,M)) \leq f(|w|+i)\}$$

$$\cup$$

$$\{u.new(a) : ua \in \Sigma^*, |a|=1 \;\&\; \exists \alpha(ua,M),\; space(\alpha(ua,M)) > f(|ua|)\} = LNS(M,f)$$

Machine M_1 uses at most $2f(n)$ working tape cells for any word of length n; $f(n)$ cells on tape T_0 and $f(n)$ cells on the other tapes. Thus $LNS(M,f) \in NSPACE(2f(n))$ and therefore, due to the tape compression result for the worst-case, $LNS(M,f) \in NSPACE(f(n))$.

Thus, $LNS(M,f) \in NSPACE(g(n))$ since by hypothesis $NSPACE(f(n)) \subseteq NSPACE(g(n))$. Applying the tape compression result, we get a non-deterministic machine M' with k_2 tapes that accepts $LDS(M,f)$ within space $\frac{g(n)}{2}$. Thus consider the following definition of non-deterministic machine M_2 which behaves as follows on input w.

(i) Simulate a computation of machine M' on input w.

- If M' accepts, w then accept w.

(ii) Otherwise, simulate a computation of M' on $u.new(a)$, $w=ua$.

- If M' rejects $u.new(a)$, then reject w.

(iii) Otherwise, simulates a computation of M' on input wc^i. M_2 accepts w if and only if M' accepts wc^i.

The language accepted by M_2 is the set of words accepted in (i) or (iii). That is:

$$\{w \in \Sigma^* : w \in LNS(M,f)\}$$

$$\cup$$

$$\{w \in \Sigma^* : \exists i \; wc^i \in LNS(M,f)\}$$

$$= \{w : w \in L(M) \; and \; \exists i \; and \; accepting \; \alpha(w,M), \;\; space(\alpha(w,M)) \leq f(|w|+i)\}$$

But $space(\alpha(w,M)) \leq f(|w|+i)$ for some i for any $w \in L(M)$, since f is monotonic increasing. Thus $L(M_2) = L(M)$.

Suppose M respects the bound on input w. Then $S(w) \leq f(|w|)$, so all computations of M on w are within space $f(|w|)$. First suppose w in $L(M)$. Then w is in $LNS(M,f) = L(M')$. Thus all computations of M' on w must use only $\frac{g(|w|)}{2}$ working tape cells, with at least one accepting computation. Therefore this accepting computation will be simulated by machine M_2 and will make M_2 accept w in step (i). Furthermore the rejecting computations of M' on w will result in the rejection of w in step (ii). Thus M_2 accepts w within space bounded by (i) and (ii).

Similarly if $w = u.a \notin L(M)$ and $S(w) \leq f(n)$, then $u.new(a)$ is rejected by M' in all computations using at most $\frac{g(n)}{2}$ working tape cells. Thus all computations for $u.new(a)$ are rejecting and within the bound. Then w is rejected by M_2 in (ii).

Therefore all words w for which M respects the bound $f(|w|)$ have computations by M_2 before (iii). But (i) costs at most $\frac{g(n)}{2}$ cells. The simulation on (ii) costs at most $\frac{g(n)}{2}$ cells. Thus the sum of steps (i) and (ii) is bounded by $g(n)$ working tape cells. Therefore all words for which M respect the bound f have space function for machine M_2 bounded by $g(n)$. Thus $L(M') \in NSPACE(g(n), d(n))$. □

Still more general results can be obtained even for mixed complexity classes. We can state the following.

Theorem 2-3-5: Let $X, Y \in \{D,N\}$, $BOUND1, BOUND2 \in \{TIME, SPACE\}$, Let $f(n)$ be a monotonic increasing $BOUND1$ constructible function such that $\inf_{n\to\infty} \frac{f(n)}{n} = \infty$. Let $g(n)$ be a function such that $\inf_{n\to\infty} \frac{g(n)}{n} = \infty$.

Then

$$XBOUND1(f(n)) \subseteq YBOUND2(g(n))$$

implies

$$XBOUND1(f(n), d(n)) \subseteq YBOUND2(g(n), d(n))$$

Proof: Lemmas 2-3-1 to 2-3-4 prove the result for the cases $X=Y$ and $BOUND1 = BOUND2$. We prove for $X=N$, $BOUND1 = SPACE$, $Y=D$, $BOUND2 = TIME$. The other proofs are analogous.

Consider M a non-deterministic machine of IO-space complexity $f(n)$ with density $d(n)$. We build machine M_1 to accept $LNS(M,f)$ as follows.

(i) If $y \notin \Sigma^* c^* \cup \Sigma^* NEW$, then M_1 rejects y.

If $y = wc^i$, then M_1 behaves as follows.

(ii) Deterministically lay off $f(|w|+i)$ cells in tape T_0.

(iii) Non-deterministically simulate the behavior of M on input w using tapes T_1 to T_{k_1} and at most $f(|w|+i)$ working tape cells. M_1 accepts y if and only if M accepts w within $f(|w|+i)$ space cells.

If $y = u.new(a)$, $u \in \Sigma^*$, then M_1 behaves as follows.

(iv) Deterministically lay off $f(|w|)$ cells in T_0'.

(v) Non-deterministically simulate the behavior of M on $u.a$, checking the if the number of cells used does not exceed $f(|y|)$. Accept y if and only if M tries to use more than $f(|y|)$ cells.

The language accepted by M_1 is composed of the sets of words accepted in (iii) and (v). The words accepted in (iii) can be described as the words wc^i for which M has an accepting computation on w within space $f(|w|+i)$. The words accepted in (v) are the words $u.new(a)$ for which M has at least one computation on ua using more than $f(|ua|)$ cells. Thus:

$$L(M_1) = \{wc^i : w \in L(M) \ \& \ \exists \ accepting \ \alpha(w,M),\ space(\alpha(w,M)) \leq f(|w|+i)\}$$

$$\cup$$

$$\{u.new(a) : ua \in \Sigma^*,\ |a|=1 \ \& \ \exists \alpha(ua,M) : space(\alpha(ua,M)) > f(|ua|)\} = LNS(M,f)$$

Machine M_1 uses at most $2f(n)$ working tape cells for any word of length n; $f(n)$ cells on tape T_0 and $f(n)$ cells on the other tapes. Thus $LNS(M,f) \in NSPACE(2f(n))$ and therefore, due to the tape compression result for the worst-case, $LNS(M,f) \in NSPACE(f(n))$.

Thus, $LNS(M,f) \in DTIME(g(n))$ since by hypothesis $NSPACE(f(n)) \subseteq DTIME(g(n))$. Applying the linear speed-up result, we get a machine M' with k_2 tapes that recognizes $LNS(M,f)$ in time $\frac{g(n)}{3}$. We build deterministic machine M_2 that acts on input w as follows.

(i) Simulate a computation of machine M' on input w.

- If M' accepts w, then accept w.

(ii) Otherwise, simulate a computation of M' on $u.new(a)$, $w = ua$.

- If M' rejects $u.new(a)$, then reject w.

(iii) Otherwise, simulate M' on input wc^i. M_2 accepts w if and only if M' accepts wc^i.

The language accepted by M_2 is the set of words accepted by machine M_2 in (i) or in (iii). We describe the words accepted in (i) and (iii) as the set of words w for which there exists i such that wc^i is accepted by M'. But word wc^i is accepted by M' if and only if w is in $L(M)$ and $S(w) \le f(|w|+i)$, by definition of $LNS(M,f) = L(M')$. Then:

$$L(M_2) = \{w : w \in L(M) \text{ and } \exists i \; S(w) \le f(|w|+i)\}$$

But $S(w) \le f(|w|+i)$ for some i for any $w \in L(M)$, since f is monotonic increasing. Thus $L(M_2) = L(M)$.

Let w be a word of length n for which M respects the bound, i.e. $S(w) \le f(n)$. If this word w is in $L(M)$, then $w \in L(M_1) = LNS(M,f)$, by construction. Thus $w \in L(M') = LNS(M,f)$. But then machine M_2 will accept w in step (i).

Similarly if $w = u.a \notin L(M)$ and $S(w) \le f(n)$, then $u.new(a) \notin L(M_1) = LNS(M,f) = L(M')$. Thus $u.new(a)$ is rejected by M', and then w is rejected by M_2 in step (ii).

Then M_2 halts in (i) or (ii) for all inputs w for which M respects the bound. But (i) costs at most $\frac{g(n)}{3}$ computation steps. Step (ii) requires the writing of $u.new(a)$ on some working tape; this action is bounded by $\frac{g(n)}{3} \ge n$, since $\inf_{n \to \infty} \frac{g(n)}{n} = \infty$ implies $\frac{g(n)}{k} \ge n$ almost everywhere. The simulation on (ii) costs at most $\frac{g(n)}{3}$ steps. Thus the sum of steps (i) and (ii) is bounded by $g(n)$ computation steps. But then $L(M) \in DTIME(g(n), d(n))$. □

Theorem 2-3-5 has a lot of significant and helpful consequences. These are some examples:

SPEED-UP:

This result says that any language accepted in worst-case time complexity $kf(n)$, k constant greater than zero, can be accepted in the worst-case within time $f(n)$.

Corollary 2-3-6: Let $f(n)$ be a monotonic increasing time constructible function such that $\inf_{n \to \infty} \frac{f(n)}{n} = \infty$ and $k > 0$, then:

$$DTIME(f(n),d(n))=DTIME(kf(n),d(n))$$

and

$$NTIME(f(n),d(n))=NTIME(kf(n),d(n)).$$

Proof: By the results for the worst-case complexity we get $XTIME(f(n))=XTIME(kf(n))$. Thus by proposition 2-3-5, we get $XTIME(f(n),d(n))=XTIME(kf(n),d(n))$. □

TAPE COMPRESSION:

This result is the equivalent of the linear speed-up for space bounds.

Corollary 2-3-7: Let $f(n)$ be a monotonic increasing space constructible function. Let k be any constant greater than zero then

$$DSPACE(f(n),d(n))=DSPACE(kf(n),d(n))$$

and

$$NSPACE(f(n),d(n))=NSPACE(kf(n),d(n)).$$

SAVITCH'S RESULT:

This result describes the simulation of a non-deterministic machine space bounded by $f(n)$ in deterministic space $f(n)^2$, for the worst-case complexity.

Corollary 2-3-8: Let $f(n)$ be a monotonic increasing space constructible function such that $f(n)\geq \log n$.

$$NSPACE(f(n),d(n))\subseteq DSPACE(f(n)^2,d(n)).$$

OTHER RELATIONS:

Corollary 2-3-9: Let $f(n)$ be a monotonic increasing time constructible function with $\inf_{n\to\infty} \frac{f(n)}{n}=\infty$. Then

$$DTIME(f(n),d(n)) \subseteq DSPACE(f(n),d(n))$$

and:

$$NTIME(f(n),d(n)) \subseteq NSPACE(f(n),d(n)).$$

Corollary 2-3-10: Let $f(n)$ be a monotonic increasing time constructible function with $\inf_{n\to\infty} \frac{f(n)}{n}=\infty$. If $L \in DSPACE(f(n),d(n))$ then there is constant k such that $L \in DTIME(k^{f(n)},d(n))$

Corollary 2-3-11: Let $f(n)$ be a monotonic increasing time constructible function with $\inf_{n\to\infty} \frac{f(n)}{n}=\infty$. If $L \in NTIME(f(n),d(n))$ then there is constant k such that $L \in DTIME(k^{f(n)},d(n))$

2-4 Tape Reductions

In this section we consider the effect of the number of tapes on the complexity classes. We denote by k*-TAPE-XBOUND* the limitation of the complexity class, IO or worst-case, *XBOUND* to off-line Turing machines with only k working tapes.

Theorem 2-3-5 extends the results of the worst-case complexity classes to other classes. However all the machines involved in that theorem were multitape machines. We want to extend all the tape reduction results of the worst-case complexity to the new complexity. We start by considering the effect on running time of simulating a multitape Turing machine by a machine with only one working tape.

Corollary 2-4-1: Let $f(n)$ be a monotonic increasing time constructible function with $\inf_{n\to\infty} \frac{f(n)}{n} = \infty$. Then

$$DTIME(f(n), d(n)) \subseteq 1\text{-}TAPE\text{-}DTIME(f(n)^2, d(n))$$

Proof: Let M be a deterministic Turing machine of IO-time complexity $f(n)$ with density $d(n)$ and consider machine M_1 of Lemma 2-3-1 which recognizes $LDT(M,f)$ and is of worst-case time complexity $3f(n)$. But then $LDT(M,f)$ belongs to $DTIME(\frac{f(n)}{2})$, due to the linear speed up theorem for the worst-case complexity. Furthermore, we can simulate any deterministic multitape Turing machine of worst-case time complexity T by a one-tape deterministic machine of worst-case time complexity T^2 [Hopc79]. Thus, we get $LDT(M,f)$ in $1\text{-}TAPE\text{-}DTIME(\frac{f(n)^2}{4})$.

Let M' be a one-tape deterministic machine of time complexity $\frac{f(n)^2}{4}$ accepting $LDT(M,f)$. We have to build machine M'_2 that plays the role of machine M_2 of Lemma 2-3-1, but has only one working tape. We suppose the input w limited by blanks is written in tape T_0. Let $w = u.a$, $u \in \Sigma^*$ and $a \in \Sigma$. Then we define the behavior of M'_2 on w as follows.

(i) It first simulates M' using input tape T_0 and working tape T_1. If M' accepts w, then M'_2 accepts w.

(ii) Otherwise, M'_2 places its input tape head at the beginning of w on T_0. Let $w=ua$, $a\in\Sigma$.

(iii) Now M'_2 has to simulate M' on $u.new(a)$ without writing $u.new(a)$ on any additional working tape nor using another track in tape T_1 (because of time constraints). So each time M'_2 reads a symbol a, it has to check if the next symbol to the right is blank. If this is the case, i.e. the next symbol is *blank*, then M'_2 simulates M' on symbol $new(a)$. Otherwise, when the next symbol is not *blank*, it simulates M' on a.

(iv) If M' rejects $u.new(a)$, then M'_2 rejects w.

(v) Otherwise, M'_2 uses tape T_1 as a double track tape. In the second track it simulates M' with input wc^i written on the first track of T_1, for $i=1,2....$ until M' makes a decision, which is then the decision of M'_2.

Notice that M_2 and M'_2 accept the same language, i.e. $L(M)$. Also, if M respects the bound for input w, that is $T(w)\le f(n)$, then M'_2 halts for w in steps (i) to (iii). Step (i) is bounded by $\frac{f(n)^2}{4}$, since that is the time complexity of M'. Step (ii) costs at most n time steps. But $\inf_{n\to\infty}\frac{f(n)}{n}=\infty$ implies $f(n)\ge 4n$ almost everywhere. Thus (ii) is bounded by $\frac{f(n)^2}{4}\ge\frac{f(n)}{4}\ge n$ almost everywhere. Therefore, machine M'_2 can store the answers in its finite control for all inputs w for which $f(|w|)<4|w|$. Steps (iii) and (iv) spend twice the cost of M', since for each action of M', M'_2 has to check the next symbol, this yields $\frac{2f(n)^2}{4}$. So the total of time steps for these words is $f(n)^2$, sum of (i) and (ii). Therefore, $L(M)\in 1\text{-}TAPE\text{-}DTIME(f(n)^2,d(n))$. □

The above simulation allow us to extend other tape reduction results of the worst-case to IO-complexity classes with density $d(n)$. The proofs for the next corollaries are similar to the proof of Corollary 2-4-1 and will be omitted.

Corollary 2-4-2: Let $f(n)$ be a monotonic increasing time constructible function with $\inf_{n \to \infty} \frac{f(n)}{n} = \infty$. Then

$$NTIME(f(n), d(n)) \subseteq 1\text{-}TAPE\text{-}NTIME(f(n)^2, d(n)).$$

Corollary 2-4-3: Let $f(n)$ be a monotonic increasing time constructible function with $\inf_{n \to \infty} \frac{f(n)}{n} = \infty$. Then

$$DTIME(f(n), d(n)) \subseteq 2\text{-}TAPE\text{-}NTIME(f(n)\log n, d(n)).$$

Corollary 2-4-4: [Book70] Let $f(n)$ be a monotonic increasing time constructible function with $\inf_{n \to \infty} \frac{f(n)}{n} = \infty$. Then

$$NTIME(f(n), d(n)) \subseteq 2\text{-}TAPE\text{-}NTIME(f(n), d(n)).$$

For space bounds the one-tape simulation is even easier, since we can use a multiple track tape without concern for time constraints. Thus, we have:

Corollary 2-4-5: Let $f(n)$ be a monotonic increasing space constructible function. Then

$$DSPACE(f(n), d(n)) = 1\text{-}TAPE\text{-}DSPACE(f(n), d(n)).$$

Corollary 2-4-6: Let $f(n)$ be a monotonic increasing space constructible function. Then

$$NSPACE(f(n), d(n)) = 1\text{–}TAPE\text{–}NSPACE(f(n), d(n)).$$

CHAPTER 3

THE STRUCTURE OF XBOUND(f(n),d(n))

3-1 Introduction

It is intuitive that the more you have the more you get- the more resources allotted the more languages can be accepted. In this chapter, we examine how tight these hierarchy results can be- how much time or space or density must be added to guarantee a larger complexity class.

Our object is to demonstrate the existence of a language L in $XBOUND(g(n),d_1(n))$ but not in $XBOUND(f(n),d_2(n))$, using diagonalization techniques with $g(n)$ and $f(n)$ and $d_1(n)$ and $d_2(n)$ as close as possible. The language L is to contain names of Turing machines and be in $XBOUND(g(n),d_1(n))$. To negate membership in $XBOUND(f(n),d_2(n))$ a counterexample must be found for each language L' in $XBOUND(f(n),d_2(n))$, generally by showing that there is a machine M' for L' whose name is in L if and only if it is not in L'. We say that a machine M' is cancelled by witness w in L if $w \in L \leftrightarrow w \notin L'$. In these terms, L must cancel every Turing machine which operates in bound $f(n)$ with density $d_2(n)$. If $d_2(n)>0$ almost everywhere, it suffices to cancel every machine that respects the bound $f(n)$ for at least one word of almost every length. If w is associated to a Turing machine M' which respects the bound for w, then w can be a witness to cancel M'.

A crucial point is that a machine for L must be accepted in IO-bound $g(n)$ with density $d_1(n)$. It must have a finite number of tapes and symbols. This machine must be able to simulate f-bounded machines with an arbitrary number of tapes and symbols. Hence we must be able to code multiple tapes into some finite number of tapes. As far as we know, this always has a cost, as seen in section 2-4. This cost depends on whether the bound is time or space. We start by analyzing the deterministic time hierarchy.

3-2 Deterministic Time Hierarchy

We start by investigating functions $f(n)$ and $g(n)$ such that there are languages recognized deterministically in time bound $g(n)$, that cannot be accepted by any deterministic Turing machine of IO-bound $f(n)$ with density $d(n)$, for *any* $d(n)$ positive almost everywhere.

Using diagonalization techniques for the deterministic time case introduces a slow-down, due to the cost of simulating many tapes in one working tape. For example, in order to show the existence of a language in $DTIME(g(n),1)$ that cannot be in $DTIME(f(n),d(n))$, the next theorem asks the function $g(n)$ to beat the function $f(n)^2$.

Theorem 3-2-1: Let $f(n)$ and $g(n)$ be monotonic increasing time constructible functions such that $\inf_{n\to\infty} \frac{g(n)}{f(n)^2}=\infty$ and $g(n)\geq n$. There exists a language L such that for all density function $d(n)$ that are positive almost everywhere:

$$L \in DTIME(g(n),1) \;\; \mathit{and} \;\; L \notin DTIME(f(n),d(n)).$$

Proof: We are going to prove this theorem using diagonalization arguments over the class of one working tape deterministic Turing machines with input alphabet Σ. Here we must cancel every Turing machine which respects the bound for at least one word of each length n for almost all n. We suppose that Σ has at least two symbols. We assume that we have a naming scheme $w \leftrightarrow M_w$ for giving machines names over an alphabet Σ with two properties [Grei84]. First, the names are arranged so that for some symbol in Σ, say 1, if w names M_w so does $1^j w$ for all j and also y if $w=1^j y$. Thus M_w has names of all lengths above some minimal length. Let $\gamma(w)$ be name w stripped of the initial 1s; this is the portion that carries the information. Second, if we have z on the input tape, w on some other tape and the working tape of M_w encoded on yet another tape, then one step of M_w on z can be simulated in time $k\,|\gamma(w)|$ for some constant k independent of z and M_w; this includes the time for encoding many symbols into some finite number of symbols.

For any word w in Σ^*, let u be the representation of the integer $|w|$ in base $|\Sigma|$. We will show that the following language has the desired properties:

$$L=\{w \in \Sigma^* : M_u \textit{ halts and rejects } w \textit{ within } \frac{g(|w|)}{|\gamma(u)|} \textit{ steps}\}.$$

Notice that the language L consists of words w for which the Turing machine encoded by u, which is the representation of $|w|$ in base $|\Sigma|$, halts and rejects w within $\frac{g(|w|)}{|\gamma(u)|}$ time steps. The membership of w in L for any word of length n depends on the behavior of the machine M_u and thus M_u will be cancelled if M_u respects the bound for any word of length n.

The proof consists of two parts. The first part is the definition of a machine M accepting L within worst-case time $g(n)$. The second part consists in showing that there is at least one word for which L and any language in $DTIME(f(n), d(n))$ disagree. We proceed by defining the deterministic machine M which acts on input w as follows.

(i) It writes u on tape T_1. One way of doing this is as follows.

-It writes the unary representation of $|w|$ on T_2.

-It writes the unary representation of $|\Sigma|$ on T_4.

-It divides the number in T_2 by the number in T_4. It just check how many times the number in T_4 fits the number in T_2. This number is the quotient of the division.

-The quotient will be used again as the next dividend.

-Let r be the rest of each division. At the end of each division the r^{th} symbol of Σ is written on T_3 on the first available position.

-The divisions continue until dividend 0 is reached. The contents of tape T_3 are the code u of integer $|w|$ written in base $|\Sigma|$. Furthermore, $\gamma(u)$ is the string u stripped of the initial 1s.

(ii) Machine M must simulate machine M_u on input w. Thus, the working tape symbols of M_u are encoded in tape T_5 with uniform length at most $|\gamma(u)|$.

(iii) It writes $\gamma(u)$ on tape T_3. This will be used as the program tape of M_u.

(iv) It uses the input tape as the input tape of M_u.

(v) It records the current state on tape T_4 and the working tape on T_5.

(vi) It counts off $\frac{g(|w|)}{|\gamma(u)|}$ on tape T_2 and simultaneously simulates M_u on input w for at most $\frac{g(|w|)}{|\gamma(u)|}$ steps as follows.

- The machine reads the input symbol of w.

- The instruction of M_u is found on T_3.

- The state information is found on T_4.

- Each step of M_u is simulated on tape T_5 and the input and the states are updated for the next cycle. Each step of M_u decreases the number on T_2 by one.

(vii) If M reaches a halting ID of M_u, then it accepts w if and only if M_u rejects w.

Let $|w|=n$; each cycle of step (i) is bounded by some constant multiplied by the length of the dividend. It starts with length n in unary, which is decreased to $\frac{n}{|\Sigma|}$ and then to $\frac{n}{|\Sigma|^2}$ and so on. The sum $\sum_{k=1}^{\infty} \frac{n}{|\Sigma|^k}$ is the sum of a geometric progression with factor $\frac{1}{|\Sigma|} \leq 1$. Thus, this sum is bounded by $\frac{n}{1-|\Sigma|^{-1}}$. Thus, the total cost of (i) is bounded by $k'n$, k' constant, which is no more than $k'g(n)$ by hypothesis, k' constant. From (ii) to (vii) M simulates at most $\frac{g(n)}{|\gamma(u)|}$ steps of M_u. But by assumption each step can be simulated in time $k|\gamma(u)|$. Thus, we spend at most $(k+k')g(n)$ steps. Then, $L(M) \in DTIME((k+k')g(n))$ and by the tape compression result it belongs to $DTIME(g(n))$.

We still must show that L is not in $DTIME(f(n),d(n))$. Suppose that this is not the case. By Corollary 2-4-1, L is accepted by some deterministic Turing machine M' with one working tape within IO-time $f(n)^2$ with density $d(n)$. We procceed by showing the existence of at least one word w for which machine M simulates M', accepting w if and only if M' rejects it. We show that for this word w machine M simulates machine M' until M' halts; and thus, machine M halts on step (vii) for this word w, accepting it if and only if M' rejects it.

Let $x=\gamma(x)$ name M' in our naming scheme. By hypothesis, $\inf_{n\to\infty} \frac{g(n)}{f(n)^2}=\infty$, which implies $g(n)\geq cf(n)^2$ almost everywhere for any constant c. Thus after some constant n_o, $g(n)\geq|\gamma(x)|f(n)^2$, for all $n\geq n_o$.

Furthermore, $d(n)$ is positive almost everywhere. Thus after some $n_{o'}$, the density function $d(n)$ is greater than zero for any $n\geq n_{o'}$. Thus, for all $n\geq n_{o'}$, there is at least one word w of length n for which M' completes its computation within time $f(n)^2$, or otherwise $d(n)\leq \sum_{|w|=n:\, T(w)\leq f(n)^2} P[X=w/n]=0$.

Let $m\geq \max\ \{\ |x|,\ n_o,\ n_{o'}\ \}$ and $y=1^{m-|x|}x$ be a name for machine M'. There is such a y, since M' has infinitely many names. Thus, $\gamma(x)=\gamma(y)$, since x and y name the same machine. But then:

$$g(m)\geq|\gamma(y)|f(m)^2$$

Let y be the base $|\Sigma|$ representation of $n>m$. Note that if $|w|=n$ then membership of w in L depends on the behavior of machine $M_y=M_x=M'$.

But since $n \ge n_{0'}$ there is w, $|w|=n$, such that M' completes its computation on input w within time $f(n)^2$. Therefore $M'=M_y$ completes its computation on w in time $\frac{g(n)}{\gamma(y)}$, since $g(n) \ge |\gamma(y)| f(n)^2$. Thus:

$$w \in L = L(M') \textit{ if and only if } w \notin L(M_y) = L(M')$$

This is a contradiction and so $L \notin DTIME(f(n), d(n))$. □

Notice that the above result is stronger than the standard hierarchy results obtained by diagonalization. It says that there is a language that is accepted with density 1 for function g that cannot be accepted with any positive density $d(n)$ for function f. Here we have two variables: the time bounds f and g and the densities d and 1, instead of fixing one variable and varying the other. In terms of worst-case complexity, it says that there is a language L computable in time $g(n)$ but every machine for L exceeds bound $f(n)$ on every word of length n for infinitely many n.

We recall Corollary 2-4-3, which says that we can simulate any number of tapes with two tapes by going from time f to time $f \log f$. Thus the previous arguments go through if we diagonalize over two tape Turing machines and let g beat $f \log f$ almost everywhere. Therefore, we have next proposition.

Theorem 3-2-2: Let $f(n)$ and $g(n)$ be monotonic increasing time constructible functions such that $\inf_{n \to \infty} \frac{g(n)}{f(n) \log f(n)} = \infty$ and $g(n) \ge n$. There exists a language L such that for all density function $d(n)$ that are positive almost everywhere:

$$L \in DTIME(g(n), 1) \textit{ and } L \notin DTIME(f(n), d(n)).$$

Obviously, since the set $DTIME(f(n),1) \subseteq DTIME(f(n),d(n))$, we get a more symmetric result as follows.

Corollary 3-2-3: Let $f(n)$ and $g(n)$ be monotonic increasing time constructible functions such that $\inf_{n\to\infty} \frac{g(n)}{f(n)\log f(n)} = \infty$ and $g(n) \geq n$. Let $d(n)$ be positive almost everywhere. Then:

$$DTIME(g(n),d(n)) - DTIME(f(n),d(n)) \neq \emptyset$$

3-3 Deterministic Space Hierarchy

In order to obtain a result similar to Theorem 3-2-1 for space bounds, we need to require IO-space bounded machines to halt as well as respect the bound. The next lemma tell us we can do so without loss of generality.

Lemma 3-3-1: Let $f(n)$ be a monotonic increasing space constructible function. Given a k-tape Turing machine M accepting a language L in IO-space bound $f(n)$ with density $d(n)$ there is a k-tape Turing machine M' accepting L for which

$$d(n) \leq \sum_{|w|=n : S'(w) \leq f(n) \& T'(w) < \infty} P[X=w/n] \quad \textit{for all } n,$$

where $S'(w)$ is the space spent on input w by M' and $T'(w)$ is the running time of machine M' on w.

Proof: The basic idea of the proof is that if a machine M is space bounded on some input, then after some number of computation steps, if M does not halt, then M loops. We simulate M by a new machine M' that rejects the word if M repeats *ID*s. Thus M' halts in finite time for all computations of M on words that use limited amount of cells.

Let s and t be the number of states and tape symbols of a $f(n)$ IO-space bounded machine M accepting L with density $d(n)$. If M uses at most $f(n)$ cells for a word w of size n then it uses at most $(n+2)sf(n)t^{f(n)} \leq 4st^{f(n)}$ different IDs. Thus, if after this number of computation steps M does not halt for w and M does not visit more than $f(n)$ cells, then M rejects w using at most $f(n)$ cells and looping on w. We have to avoid this case, by constructing machine M' that acts on w as follows.

(i) Lay off $f(n)$ cells on each working tape.

(ii) Set a counter of length $f(n)$ in base $4st$ using a new track of one of the k working tapes.

(iii) Simulate M on the delineated space until the count ends.

- if M accepts w, then accept w.

- if M rejects w or does not halt, then reject w.

(iv) If M leaves the delineated space, then continue simulating M.

Obviously, $L(M')=L(M)$. Let $S(w)$ denote the space spent on word w by machine M. Then, for all words with $S(w) \leq f(n)$, M' will reach a decision in step (iii). Thus for those words the running time $T'(w)$ of M' on w is finite. Furthermore, until step (iii) machine M' visits the same number of cells as M does. Therefore, $S(w) \leq f(n)$ implies $S'(w) \leq f(n)$ cells and $T'(w) \leq f(n)$.

But machine M is of space complexity $f(n)$ with density $d(n)$. So: $d(n) \leq \sum_{|w|=n : S(w) \leq f(n)} P[X=w/n]$. But $S(w) \leq f(n)$ implies $S'(w) \leq f(n)$ and $T'(w) < \infty$. Thus:

$$d(n) \leq \sum_{|w|=n:S'(w)\leq f(n)\&T'(w)<\infty} P[X=w/n]$$

Finally, notice also that M and M' have the same number of working tapes. □

Notice that Lemma 3-3-1 allow us to replace a machine M that operates in IO-space bound $f(n)$ with density $d(n)$ by another machine M' that not only operates in IO-space bound $f(n)$ with density $d(n)$ but also halts on all words for which it respects the bound. This feature will be useful in proving next result, which states the existence of a language in $DSPACE(g(n),1)$ that cannot be in $DSPACE(f(n),d(n))$ for appropriate f and g.

Theorem 3-3-2: Let $f(n)$ and $g(n)$ be monotonic increasing space constructible function with $g(n)\geq n$ and $\inf_{n\to\infty} \frac{g(n)}{f(n)}=\infty$. There exists a language L such that for all density functions $d(n)$ that are positive almost everywhere:

$$L\in DSPACE(g(n),1) \ and \ L\notin DSPACE(f(n),d(n)).$$

Proof: By Corollary 2-4-5, any language in $DSPACE(f(n),d(n))$ can be recognized by an off-line one working tape Turing machine in IO-space bound $f(n)$ and density $d(n)$. Let the input alphabet Σ have at least two symbols. By Lemma 3-3-1, we can assume that all languages in $DSPACE(f(n),d(n))$ are accepted by one tape deterministic Turing machines that operate within IO-space bound $f(n)$ with density $d(n)$ and halt on all words for which they respect the bound.

We assume that we have a naming scheme $w\leftrightarrow M_w$ for giving machine names over Σ with two properties [Lewi81]. The names of machines are such that from a name w and an ID I the next ID under M_w can be computed in space $w+|I|$. Further, the names are arranged so that for some symbol in Σ, say 1, if w names M_w

so does $1^j w$ for all j, thus M_w has name of all lengths above some minimal length. For each machine M, let $\Gamma(M)$ be the number of working tape symbols; we can assume that is easily computable from the name of M.

Let u be the representation of number $|w|$ in base Σ. We consider the language

$L = \{w$: *M_u halts* and *rejects w without visiting more than* $g(|w|)/\Gamma(M_u)$ *squares* $\}$.

Consider the multitape machine M which acts on input w of size n as follows.

(i) It lays out $g(|w|)$ squares on all working tapes.

(ii) It counts number n in base $|\Sigma|$. The final code is u.

(iii) It divides tape T_2 in $\Gamma(M_u)$ cells.

(iv) It simulates machine M_u acting on input w as follows.

- It uses tape T_2 as the working tape of M_u; the head of T_2 can keep the position of the working tape head of M_u. Each working tape symbol of M_u is encoded in $\Gamma(M_u)$ squares of T_2.

- It records the current state on T_3; we can assume that there are at most $|w|$ states; so the current state can be recorded in space $\log|w|$.

-Each simulation cycle starts by reading the input symbol of w. Then the appropriate instruction of M_u is found on tape T_1 using T_3 for the state information. Next M simulate this step of M_u on T_2 updating the input and the state on tape T_3.

(v) If any part of this simulation would cause M to leave its delineated space, M halts and rejects w.

(vi) If M reaches a halting ID of M_u, it accepts w if and only if M_u rejects it.

First, observe that the computation of step (ii) is bounded by $n \le g(n)$ squares. During all the other steps, M uses at most $g(n)$ cells on each working tape. Thus $L = L(M) \in DSPACE(g(n), 1)$ since M never goes off the marked cells.

We have to prove that L cannot be in $DSPACE(f(n), d(n))$ with $d(n) \ne 0$ almost everywhere. We proceed by contradiction. Suppose L is in $DSPACE(f(n), d(n))$. Then we can assume that $L = L(M')$ for an off-line Turing machine M' with one working tape which is IO-space bounded with density $d(n)$ and always halts within $f(n)$ cells for at least one word w of size n for all n large enough, since $d(n) \ne 0$ almost everywhere. Furthermore, M' has a name in our scheme. By hypothesis $\inf_{n \to \infty} \frac{g(n)}{f(n)} = \infty$, this implies that for each $k > 0$ $g(n) \ge kf(n)$ almost everywhere. Also $d(n) \ne 0$ almost everywhere. Thus, M' has a name $y = 1^{n' - |x|} x$, with x the minimal name for M', such that:

$$g(n) \ge \Gamma(M_y) f(n) \;\;\&\;\; d(n) > 0 \;\; \textit{for all} \;\; n \ge n'.$$

Let $n > n'$ be the integer encoded by y. Then for some input w of this size n, M_y will halt and either accept or reject without visiting more than $f(n) \le \frac{g(n)}{\Gamma(M_y)}$ squares. Hence:

$$w \in L(M') \;\textit{if and only if}\; M' = M_y \;\textit{rejects}\; w \;\textit{if and only if}\; w \notin L(M_y) = L(M')$$

This is a contradiction and so $L \notin DSPACE(f(n), d(n))$. □

Corollary 3-3-3: Let $f(n)$ and $g(n)$ be monotonic increasing space constructible function with $f(n) \geq n$ and $\inf_{n \to \infty} \frac{g(n)}{f(n)} = \infty$. Let $d(n)$ be positive almost everywhere. Then:

$$DSPACE(g(n), d(n)) - DSPACE(f(n), d(n)) \neq \emptyset$$

Notice that for space bounds the function g does not need to beat *flogf* as for time bounds, since the one tape simulation of a space bounded machine does not require it.

Theorem 3-3-2 requires that $d(n)$ be positive almost everywhere. However, better results can be obtained for the deterministic space; the next theorem will require only the condition that $d(n)$ be positive for infinitely many n.

Theorem 3-3-4: Let $f(n)$ and $g(n)$ be monotonic increasing space constructible functions with $g(n) \geq n$ and $\inf_{n \to \infty} \frac{g(n)}{f(n)} = \infty$. There exists a language L such that for all density function $d(n)$ that are positive for infinitely many n:

$$L \in DSPACE(g(n), 1) \textit{ and } L \notin DSPACE(f(n), d(n)).$$

Proof: We suppose that all languages in $DSPACE(f(n), d(n))$ are accepted within the appropriate IO-bounds by one tape off-line Turing machines, due to Corollary 2-4-5. We follow the notation of Theorem 3-3-2 and we denote the number of states of machine M by $s(M)$.

Consider the previous naming scheme for Turing machines. Let $\alpha(0)$ be the empty word. We recursively define $\alpha(i)$, $i \geq 1$, as the first valid name of Turing machine in canonical order after $\alpha(i-1)$. Thus $\alpha(0) < \alpha(1) < \alpha(2) < \cdots < \alpha(k) \ldots$, with < representing the canonical order .

Let $h(w, 0)$ be the empty word. We recursively define $h(w,i)$, $i=1,2...$ as follows. Intuitively $h(w,i)$ gives us candidate inputs for cancelling $\alpha(i) \leftrightarrow M_{\alpha(i)}$.

→ $h(w,i)=z$ if $\exists z$, $h(w,i-1)<z\leq w$, $z\geq\alpha(i)$ such that $M_{\alpha(i)}$ visits no more than $\frac{g(|z|)}{\Gamma(M_{\alpha(i)})}$ cells for input z and $\forall y$, $h(w,i-1)<y<z$, $y\geq\alpha(i)$, $M_{\alpha(i)}$ visits more than $\frac{g(|y|)}{\Gamma(M_{\alpha(i)})}$ cells for input y;

→ $h(w,i)=h(w,i-1)$ otherwise.

The function $h(w,i)$ defines the first word z after $h(w,i-1)$ for which the machine named by $\alpha(i)$ visits no more than $\frac{g(|z|)}{\Gamma(M_{\alpha(i)})}$ cells on input z, if there exists such z. Language L is formed of all such z that are rejected by machine $M_{\alpha(i)}$. Intuitively, if $M_{\alpha(i)}$ does not respect the bound $\frac{g(n)}{\Gamma(\alpha(i))}$ for any word $z\geq\alpha(i)$ after $h(w,i-1)$- the candidate for cancelling $M_{\alpha(i-1)}$-, then $M_{\alpha(i)}$ does not respect the bound $f(n)$ infinitely often and need not be cancelled. To record that fact, we define $h(w,i)=h(w,i-1)$. If the first "good" word beyond $h(w,i-1)$ is beyond w, then w is not a candidate for cancelling $M_{\alpha(i)}$. Otherwise, we let $h(w,i)$ be the first word $z\geq\alpha(i)$ beyond $h(w,i-1)$ (but not beyond w) for which $M_{\alpha(i)}$ respects the bound $\frac{g(n)}{\Gamma(\alpha(i))}$. We use this word to cancel $M_{\alpha(i))}$ in the usual way- accept if and only if $M_{\alpha(i)}$ rejects. We really wish to define $h(i)$=*default* if $M_{\alpha(i)}$ does not respect the bound for any $z\geq\alpha(i)$ beyond the last defined $h(i-k)$ and otherwise the first such z. However, that would not be computable and so instead $h(w,i)$ gives us all the information about $h(i)$ available without exceeding w: $h(w,i-1)=h(w,i)$ if $h(i)$=*default* or $h(i)>w$ and $h(w,i)=h(i)$ if $h(i)\leq w$. Formally, L can be expressed as follows.

$$L = \{w : \exists i \text{ for which } w=h(w,i)\neq h(w,i-1) \text{ and } w\notin L(M_{\alpha(i)})\}$$

Consider a multitape deterministic Turing machine M that acts on input w as follows.

(1) Mark $g(|w|)$ squares on each working tape.

(2) LET $z=h(w,0)$ be the empty word;

LET $i=1$;

LET *flag* =*NO*;

LET *overflow* =*NO*;

LET *accept* =*YES*.

(3) WHILE *flag*=*NO* and M does not visit any unmarked cell DO

BEGIN

(3-1) IF $\alpha(i)>z$ THEN *overflow* =*YES*;

(3-2) WRITE $\alpha(i)$ on tape T_2;

(3-3) WRITE z on tape T_3; """ M will simulate machine $M_{\alpha(i)}$ on input z by at most $4s(M_{\alpha(i)})\Gamma(M_{\alpha(i)})^{f(|z|)}$ time steps using at most $g(|z|)$ cells. """

(3-4) Lay off $g(|z|)$ cells on tape T_4. """ Machine M will use tape T_4 as the working tape of $M_{\alpha(i)}$; the head of T_4 will keep the position of the working tape head of $M_{\alpha(i)}$. Each working tape symbol of $M_{\alpha(i)}$ is encoded in $\Gamma(M_{\alpha(i)})$ squares of T_4. The current state of machine $M_{\alpha(i)}$

will be recorded on tape T_5; we can assume that there are at most $|\alpha(i)| \leq |z|$ states; so the current state can be recorded in space $\log |z|$.

""""

(3-5) LET *count*=0;

(3-6) WHILE *overflow*=*NO* and *count*$\leq$ greatest number of length $f(|z|)$ in base $4s(M_{\alpha(i)})\Gamma(M_{\alpha(i)})$ DO

BEGIN

(3-6-1) READ the input symbol of z on tape T_3;

(3-6-2) Find the appropriate instruction of $M_{\alpha(i)}$ on tape T_2 using T_5 for the state information;

(3-6-3) Simulate the instruction found in (3-6-2) on tape T_4 updating the input and the state on tape T_5. IF M tries to use more than $g(|z|)$ cells on tape T_4 THEN *overflow*=*YES*;

(3-6-4) Increment *count* by one in base $4s(M_{\alpha(i)})\Gamma(M_{\alpha(i)})$;

END;

(3-7) IF *overflow*=*YES* and $z \neq w$ THEN LET the next value of z be the next word after z in canonical order;

(3-8) IF *overflow*=*YES* and $z=w$ or *overflow*=*NO* and $z \neq w$ THEN $z=h(w,i-1)$, $h(w,i)=h(w,i-1)$ and $i=i+1$;

(3-9) IF *overflow*=*NO* and z=w THEN *accept*=*NO* IF AND ONLY IF $M_{\alpha(i)}$ accepts w; LET *flag* =*YES*;

END;

(4) IF *accept*=*YES* THEN reject w;

ELSE accept w.

If M tries to use more than $g(n)$ cells on any working tape for any word w of length n, the variable *flag* is made true and the simulation is aborted. Thus $L(M) \in DSPACE(g(n), 1)$.

In order to prove that $L(M)$ cannot be in $DSPACE(f(n), d(n))$ we need to establish the following claims.

Claim 1: For all j, there exist a word w_j such that $h(w,j)=w_j$ for any word $w \geq w_j$. The proof proceeds by induction on j.

For $j=0$, we have $h(w, 0)$=*empty word*=*e*, for all words w.

Now suppose the inductive hypothesis true for $j-1$, that is $h(w,j-1)=w_{j-1}$, for any word $w \geq w_{j-1}$. For machine named by $\alpha(j) \leftrightarrow M_{\alpha(j)}$ there are only two possibilities:

(i) there are no words $z > w_{j-1}$ with $z \geq \alpha(j)$ such that $M_{\alpha(j)}$ visits no more than $\frac{g(|z|)}{\Gamma(M_{\alpha(j)})}$ cells on input z. Thus by definition of $h(w,j)$, $h(w,j)=h(w,j-1)=w_{j-1}$, for all words $w \geq w_{j-1}$. Thus $w_j=w_{j-1}$.

(ii) otherwise, let x be the first word with $w_{j-1)} < x$ and $x \geq \alpha(j)$ such that $M_{\alpha(j)}$ visits no more than $\frac{g(|x|)}{\Gamma(M_{\alpha(j)})}$ cells on input x. Then for all words $w \geq x$: $w_{j-1}=h(w,j-1)<h(w,j)=x \leq w$, so $h(w,j)=x$. Therefore $w_j=x$.

Thus the claim is valid. The second claim is stated as follows.

Claim 2: For all j: $w_j \geq w_{j-1}$.

As in Claim 1, for the machine named by $\alpha(j)$ there are only two possibilities:

(i) there are no words $z > w_{j-1}$ with $z \geq \alpha(j)$ such that $M_{\alpha(j)}$ visits no more than $\frac{g(|z|)}{\Gamma(M_{\alpha(j)})}$ cells on input z. In this case, we say that $w_j = w_{j-1}$.

(ii) otherwise, let $x > w_{j-1}$ be the first word with $w_{j-1} < x$ and $x \geq \alpha(j)$ such that $M_{\alpha(j)}$ visits no more than $\frac{g(|x|)}{\Gamma(M_{\alpha(j)})}$ cells on input x. Then $h(w,j)=x$, for all $w \geq x$. Therefore $w_j = x > w_{j-1}$.

Therefore the second claim is also valid.

We still have to prove that L is not in $DSPACE(f(n), d(n))$. Suppose that this is not the case. Let $\alpha(i) \leftrightarrow M_{\alpha(i)}$ be the name of a deterministic Turing machine accepting L in IO-space complexity $f(n)$ with density $d(n)$. We proceed by showing that there is at least one word w_i such that $w_i \in L$ *if and only if* $w_i \notin L(M_{\alpha(i)}) = L(M')$; that is, $\alpha(i)$ was cancelled by input w_i.

By Claim 1, there are words w_i and w_{i-1} for which $w_i = h(w_i, i)$ and $w_{i-1} = h(w_{i-1}, i-1)$. Furthermore, $w_{i-1} = h(w_{i-1}, i-1) = h(w_i, i-1)$, since by Claim 1 $h(w, i-1) = w_{i-1}$ for all $w \geq w_{i-1}$ and by Claim 2 $w_i \geq w_{i-1}$. We claim that $w_i \neq h(w_i, i-1) = w_{i-1}$. Suppose not, that is $w_i = w_{i-1}$. This implies that after word w_{i-1} there is no word z such that $z \geq \alpha(i)$ and machine $M_{\alpha(i)}$ visits no more than $\frac{g(|z|)}{\Gamma(M_{\alpha(j)})}$ cells for input z; or otherwise $w_i = z \neq w_{i-1}$.

But machine $M_{\alpha(i)}$ is $f(n)$ IO-space bounded with density $d(n)$, $d(n)$ positive infinitely often. Thus $M_{\alpha(i)}$ operates within space $f(n)$ for infinitely many n. Also $g(n) \geq \Gamma(M_{\alpha(i)}) f(n)$ almost everywhere; since, by hypothesis, $\inf_{n \to \infty} \frac{g(n)}{f(n)} = \infty$. But then, we have infinitely many words z after w_{i-1} for which $M_{\alpha(i)}$ visits no more than $\frac{g(|z|)}{\Gamma(M_{\alpha(i)})}$ cells on input z. This is a contradiction and therefore $w_i = h(w_i, i) \neq h(w_i, i-1)$.

Hence by definition of L,

$$w_i \in L \textit{ if and only if } w_i \notin L(M_{\alpha(i)}) = L.$$

This is a contradiction and so $L \notin DSPACE(f(n), d(n))$. □

Notice that if we require the condition $d(n)$ positive almost everywhere the above theorem will still be true if we relax the condition $\inf_{n \to \infty} \frac{g(n)}{f(n)} = \infty$ to $\inf_{n \to \infty} \frac{f(n)}{g(n)} = 0$. In this case, we just need $g(n) \geq kf(n)$ infinitely often instead of almost everywhere.

Corollary 3-3-7: Let $f(n)$ and $g(n)$ be monotonic increasing space constructible functions with $g(n) \geq n$ and $\inf_{n \to \infty} \frac{f(n)}{g(n)} = 0$. There exists a language L such that for all density function $d(n)$ that are positive almost everywhere:

$$L \in DSPACE(g(n), 1) \textit{ and } L \notin DSPACE(f(n), d(n)).$$

Notice that the simulation of Theorem 3-3-4 is valid for time bounds. However it requires functions $f(n)$ and $g(n)$ to have a exponential gap, since the simulation requires exponential time.

3-4 Non-deterministic Hierarchies

The diagonalization argument breaks down for non-deterministic classes. To determine whether w is in $L(M)$ and contradict the situation, we must simulate all computation paths on w. But it seems to take exponentially more time or space. So we would only get exponential results.

However we can do better for non-deterministic space classes using the extension of Savitch's result, Corollary 2-3-8, which says that non-deterministic space f can be simulated by deterministic space f^2. Thus, we have:

Theorem 3-4-1: Let $f(n)$ and $g(n)$ be monotonic increasing space constructible functions such that $\inf_{n\to\infty} \frac{g(n)}{f(n)^2} = \infty$ and $g(n) \geq n$. There exists a language L such that for all density function $d(n)$ that are positive almost everywhere:

$$L \in DSPACE(g(n), 1) \textit{ and } L \notin NSPACE(f(n), d(n)).$$

Proof: By Corollary 2-3-8 $NSPACE(f(n), d(n)) \subseteq DSPACE(f(n)^2, d(n))$. But by Corollary 3-3-7 there is a language in $DSPACE(g(n), 1)$ not in $DSPACE(f(n)^2, d(n))$ and thus much less in $NSPACE(f(n), d(n))$. □

Corollary 3-4-2: Let $f(n)$ and $g(n)$ be monotonic increasing space constructible functions such that $\inf_{n\to\infty} \frac{g(n)}{f(n)^2} = \infty$ and $f(n) \geq n$. There exists a language L such that for all density function $d(n)$ that are positive almost everywhere:

$$L \in NSPACE(g(n), 1) \textit{ and } L \notin NSPACE(f(n), d(n)).$$

For non-deterministic time, however, the results are exponential, that is g has to beat f by at least an exponential amount for our proofs to work.

Theorem 3-4-3: Let $f(n)$ and $g(n)$ be monotonic increasing time constructible functions such that $\inf_{n\to\infty} \frac{g(n)}{k^{f(n)}}=\infty$ and $g(n)\geq n$. There exists a language L such that for all density function $d(n)$ that are positive almost everywhere:

$$L \in DTIME(g(n),1) \text{ and } L \notin NTIME(f(n),d(n)).$$

Proof: Suppose not. Then any L in $DTIME(g(n),1)$ would be in $NTIME(f(n),d(n))$. Then by Corollary 2-3-11, there exists a constant k for which $L \in DTIME(k^{f(n)},d(n))$. But this is a contradiction for the languages of Theorem 3-2-1. □

Corollary 3-4-4: Let $f(n)$ and $g(n)$ be monotonic increasing time constructible functions such that $\inf_{n\to\infty} \frac{g(n)}{k^{f(n)}}=\infty$ and $g(n)\geq n$. There exists a language L such that for all density function $d(n)$ that are positive almost everywhere:

$$L \in NTIME(g(n),1) \text{ and } L \notin NTIME(f(n),d(n)).$$

3-5 Density Hierarchies

We also want to see the effect of fixing f and varying the density function d. This variation depends on the particular probability distribution assumed. Initially, we assume uniform probability distribution, i.e. $P[X=w/n]=\frac{1}{|\Sigma|^n}$, and we denote it by U. The next proposition says that there are languages L in $DSPACE(f(n),d_1(n),U)$ that cannot be in $DSPACE(f(n),d_2(n),U)$, provided that the difference between $d_1(n)$ and $d_2(n)$ is at least $\frac{1}{|\Sigma|^n}$ with $L \subseteq \Sigma^*$.

Theorem 3-5-1: Let $f(n)$ be a total recursive function. If $d_1(n)$ and $d_2(n)$ are density functions such that for some integer $k \geq 2$:

(i) $\lceil d_1(n) k^n \rceil$ is computable in $DSPACE(f(n))$;

(ii) $d_2(n) > d_1(n) + \frac{1}{k^n}$,

then there exists a language L over any k symbol alphabet such that

$$L \in DSPACE(f(n), d_1(n), U) \text{ and } L \notin DSPACE(f(n), d_2(n), U).$$

Proof: Let Σ be a alphabet with $|\Sigma| \geq 2$. Let $<$ denote the lexicographical ordering over words of same length. We assume a naming scheme in Σ^n for integers such that if integer m is less or equal to integer l and m and l have names in Σ^n, then the name of m is less or equal the name of l in lexicographical order. Let $y(n)$ denote the name in Σ^n of the integer $\lceil |\Sigma|^n d_1(n) \rceil$.

Note that for any total recursive function $f(n)$ there exists function $f'(n)$ such that $f(n) \leq f'(n)$ everywhere, $f'(n)$ monotonic and space constructible. Theorem 3-3-2 asserts the existence of a recursive language H such that for any deterministic machine M for H, for infinitely many n, the space spent on w by M, $S_M(w)$, is strictly greater than $f'(n)$ for all w of length n. So let

$$L = \{w : w > y(|w|) \text{ and } w \in H\}$$

Since "$w \le y(|w|)$" can be tested in space $f(n)$, and any algorithm can be used for H, clearly $L \in DSPACE(f(n), d_1(n), U)$. Assume $L = L(M')$, where M' operates in IO-space $f'(n)$ with density $d_2(n)$. From M' we get deterministic Turing machine M for H which follows M' for $w > y(|w|)$ and elsewhere any algorithm for H. So there exists n for which $S_M(w) > f'(n)$ for all words w of length n. So in particular M' cannot obey the bound for any word w of length n with $w > y(|w|)$. Then for M':

$$\sum_{|w|=n:\, S_{M'}(w) \le f'(n)} P[X = w/n] \le \sum_{|w|=n:\, w \le y(|w|)} \frac{1}{|\Sigma|^n} \le d_1(n) < d_2(n)$$

This is a contradiction and so L cannot be in $DSPACE(f'(n), d_2(n), U)$, much less in $DSPACE(f(n), d_2(n), U)$. □

Analogous to Theorem 3-5-1 we can use the results developed in previous sections to generalize the density hierarchy for any IO-complexity class as follows.

Theorem 3-5-2: Let $X = \{D,N\}$ and $BOUND = \{TIME, SPACE\}$. Let $f(n)$ be a total recursive function. If $d_1(n)$ and $d_2(n)$ are density functions such that for some integer $k \ge 2$:

(i) $\lceil d_1(n) k^n \rceil$ is computable in $DBOUND(f(n))$;

(ii) $d_2(n) > d_1(n) + \frac{1}{k^n}$,

then there exists a language L over any k symbol alphabet such that

$$L \in XBOUND(f(n), d_1(n), U) \text{ and } L \notin XBOUND(f(n), d_2(n), U).$$

We can generalize the result above to any positive probability distribution. The proof of Theorem 3-5-1 is based on the uniform distribution; this dependence was implicit in the definition of $y(n)$. We recall that $y(n)$ is a cutpoint in the sense that all words before $y(n)$ are rejected within time $f(n)$ and all words after $y(n)$ follow an algorithm which cannot have time bound $f(n)$ in a very strong sense. For each particular definition of probability distribution the definition of the value of the cutpoint varies.

The crucial point in the proof above is to decide whether a word must be an *easy* word or a *hard* one. We need enough easy words to make the language recognizable in IO-space bound $f(n)$ with density d_1, but not enough for density d_2. For the uniform distribution this was characterized by the cutpoint $y(n)= \left\lceil |\Sigma|^n d_1(n) \right\rceil$. All words of length n before $y(n)$ were "easy" words and all words after $y(n)$ were "hard" ones. More generally, we define a cutpoint λ with the same function as $y(n)$ that can be used for any positive probability distribution. Then, consider an alphabet Σ and let the words of length n be lexicographically ordered; we denote this ordering by indexing the words, i.e. $w_0 < w_1 < \cdots < w_m$. We define λ on words of length n by

$$\lambda_\Sigma(w_j) = \begin{cases} 0 & \text{if } \sum_{i=0}^{j} P_\Sigma[X = w_i / n] \le d_1(n) \\ 1 & \textit{otherwise} \end{cases}$$

Consider the first word w_j for which $\lambda_\Sigma(w_j)$ is 1. The sum of the probability of all words less or equal this word is greater or equal to $d_1(n)$. We are going to use this word as a cutpoint; all words before it will follow an easy algorithm, which guarantees density d_1, and all words after it will follow a hard algorithm in order to avoid density d_2. Once we have defined the cutpoint, the dependence on the particular distribution is expressed by the difference between d_1 and d_2. In other words, the difference between d_1 and d_2 must be large enough to embody at least one word; otherwise $DSPACE(f(n),d_1(n),P_\Sigma[X=w/n])=DSPACE(f(n),d_2(n),P_\Sigma[X=w/n])$, trivially. Thus, we define $\Delta_\Sigma(n)=max\{P_\Sigma[X=w/n]:|w|=n\}$. Then the arguments of Theorem 3-2-4 go through in order to show that the language:

$$\{w: \lambda(w)=1 \text{ and } w\in H\}$$

belongs to $DSPACE(f(n),d_1(n),P_\Sigma[X=w/n])$ and not to $DSPACE(f(n),d_2(n),P_\Sigma[X=w/n])$ for $d_2(n)\geq d_1(n)+\Delta_\Sigma(n)$. Thus, we generalize Theorem 3-5-1 as follows.

Theorem 3-5-3: Let $f(n)$ be a total recursive function. If Σ is a alphabet with size at least two and $d_1(n)$ and $d_2(n)$ are density functions such that:

(i) Φ assigns a positive probability distribution $P[X=w/n]$ to Σ^*,

(ii) "$d_1(n)\geq \sum_{|w|=n:\, w\leq y} P[X=w/n,\ |y|=n]$?" is decidable in $DSPACE(f(n))$;

(iii) $d_2(n)>d_1(n)+max\{P[X=w/n]:\ |w|=n\}$ infinitely often,

then there exists a language L over alphabet Σ such that

$$L\in DSPACE(f(n),d_1(n),\Phi) \text{ and } L\notin DSPACE(f(n),d_2(n),\Phi).$$

Proof: Let < denote the lexicographical ordering over words of same length. We assume a naming scheme in Σ^n for integers such that if integer m is less or equal to integer l and m and l have names in Σ^n, then the name of m is less or equal the name of l in lexicographical order. Let λ be defined as below.

$$\lambda_\Sigma(w_j) = \begin{cases} 0 & \text{if } \sum_{i=0}^{j} P_\Sigma[X=w_i/n] \le d_1(n) \\ 1 & \textit{otherwise} \end{cases}$$

Let $y(n)$ be the first w_j for which $\lambda_\Sigma(w_j)$ is 1.

Note that for any total recursive function $f(n)$ there exists function $f'(n)$ such that $f(n) \le f'(n)$ everywhere, $f'(n)$ monotonic and space constructible. Theorem 3-3-2 asserts the existence of a recursive language H such that for any deterministic machine M for H, for infinitely many n, the space spent on w by M, $S_M(w)$, is strictly greater than $f'(n)$ for all w of length n. So let

$$L = \{w : w > y(|w|) \textit{ and } w \in H\}$$

Since "$d_1(n) \ge \sum_{|w|=n:\, w \le y} P[X=w/n,\ |y|=n]$?" is decidable in $DSPACE(f(n))$, then "$w \le y(|w|)$" can be tested in space $f(n)$, and any algorithm can be used for H, clearly $L \in DSPACE(f(n), d_1(n), \Phi)$. Assume $L = L(M')$, where M' operates in IO-space $f'(n)$ with density $d_2(n)$. From M' we get deterministic Turing machine M for H which follows M' for $w > y(|w|)$ and elsewhere any algorithm for H. So there exists n for which $S_M(w) > f'(n)$ for all words w of length n. So in particular M' cannot obey the bound for any word w of length n with $w > y(|w|)$. Then for M':

$$\sum_{|w|=n:\, S_{M'}(w) \le f'(n)} P[X=w/n] \le d_1(n) < d_2(n) \quad \textit{infinitely often.}$$

This is a contradiction and so L cannot be in $DSPACE(f'(n), d_2(n), \Phi)$ much less in $DSPACE(f(n), d_2(n), \Phi)$. $\square$

We can generalize Theorem 3-5-3 to embody any IO-complexity class as follows.

Theorem 3-5-4: Let $X=\{D,N\}$ and $BOUND=\{TIME,SPACE\}$. Let $f(n)$ be a total recursive function. If Σ is a alphabet with size at least two and $d_1(n)$ and $d_2(n)$ are density functions such that:

(i) Φ assigns a positive probability distribution $P[X=w/n]$ to Σ^*;

(ii) "$d_1(n) \geq \sum_{|w|=n:\, w \leq y} P[X=w/n,\ |y|=n]$?" is decidable in $DBOUND(f(n))$;

(iii) $d_2(n) > d_1(n) + max\{P[X=w/n]:\ |w|=n\}$ infinitely often,

then there exists a language L over alphabet Σ such that

$$L \in XBOUND(f(n), d_1(n), \Phi) \text{ and } L \notin XBOUND(f(n), d_2(n), \Phi).$$

CHAPTER 4

POLYNOMIAL CLASSES AND HARD PROBLEMS

4-1 Introduction

An important problem- considered by many to be the most important open question in complexity theory or indeed in theoretical computer science- is whether $P = NP$ or not. In other words "can every problem solvable non-deterministically in polynomial time actually be solved in polynomial time by a deterministic machine?".

In this chapter, we extend the concept of $XTIME(f(n), d(n))$ to include polynomial time and we study the structure of deterministic and non-deterministic polynomial time classes. We restrict our attention to the uniform probability distribution.

Definition 4-1-1: Let $0 \le d(n) \le 1$. We define:

(i) $P(d(n)) = \bigcup_{c>0} DTIME(n^c, d(n), U) =$ $\{L : \exists$ deterministic Turing machine M accepting L in IO-time n^c, with density $d(n)$ and uniform probability distribution for the input alphabet of M, for some $c > 0\}$

(ii) $NP(d(n)) = \bigcup_{c>0} NTIME(n^c, d(n), U) =$ $\{L : \exists$ non-deterministic Turing machine M accepting L in IO-time n^c, with density $d(n)$ and uniform probability distribution for the input alphabet of M, for some $c > 0\}$

Notice that $P(1)=P$ and $NP(1)=NP$, since by Theorem 2-2-1 $XTIME(n^c)=XTIME(n^c, 1)$.

We want to know the relationship among the classes P, NP, $P(d(n))$ and $NP(d(n))$. We show that there exists a positive density function $d(n)$ for which $P(d(n)) \neq NP(d(n))$ if and only if $P \neq NP$. On the other hand, we also show that the existence of a positive density function $d(n)$ for which $P(d(n))=NP(d(n))$ implies that $E=NE$, where E is the deterministic exponential class and NE is the non-deterministic exponential class of languages.

Using the concept of density function, we give an alternative proof that $NE \neq E$ implies $NP \neq P$. The implication $NE \neq E$ implies $NP \neq P$ was already known. This was showed using the concept of tally sets by Book [Book74], and using the concept of sparse sets by Hartmanis [Hart83a]. Furthermore, the existence of sparse sets in $NP-P$ and the structure of E and NE has been investigated in [Hart83b] and expanded to the exponential hierarchy in [Sewe83].

There has been some research concerning whether NP-problems can be solved "in practice". Usually, claims about an algorithm's performance "in practice" are supported by extensive tests of the algorithm on real instances from the application in question [Gare79, John84]. However, a really satisfactory theory of "in practice" complexity is not available yet. In this chapter, we want to apply the IO-complexity defined in this thesis to the concept of approximate solutions for hard problems. There are different ways in which hard problems can be dealt with. Minimally, there should be a polynomial time algorithm that for all sufficiently large instances, solves the problem with at least some required probability. To *solve* a problem in this context means to determine the correct answer and provide a proof that it is correct. This kind of solution will be denoted as an *APPROXIMATION* solution. Some examples of hard problems for which there are *APPROXIMATION* solutions are Hamiltonian circuit [Angl77], satisfiability [Gold82], subset sum [Laga83], knapsack [Gold84]. The final sections of this chapter are devoted to another interpretation of the IO-complexity classes in terms of *APPROXIMATION* sets.

4-2 Conjectures on P and NP

In this section we show that $P \neq NP$ if and only if there exists a positive density function $d(n)$, i.e. $d(n)>0$ almost everywhere, for which $P(d(n)) \neq NP(d(n))$.

We start by stating the existence of a language that cannot be accepted in deterministic polynomial time, except, for a finite number of words. The existence of such languages was first shown by Blum in abstract complexity theory.

Lemma 4-2-1: [Blum71] There exists a recursive language $L \subseteq \Sigma^*$ such that for any deterministic Turing machine M accepting L and any $c>0$, the running time of M on input w of length n exceeds n^c for almost all words.

The nest result uses Theorem 2-3-5 in order to show that the existence of a positive density function $d(n)$ for which $P(d(n)) \neq NP(d(n))$ implies the separation of the classes P and NP.

Lemma 4-2-2: If there exists a positive density function $d(n)$ for which $P(d(n)) \neq NP(d(n))$, then $P \neq NP$.

Proof: Suppose that $P=NP$, so $NP\subseteq P$.

Let $d(n)$ be any positive density function. By Theorem 2-3-5, $NTIME(f(n))\subseteq DTIME(f(n))$ implies $NTIME(f(n),d(n))\subseteq DTIME(f(n),d(n))$. Thus $\bigcup_{c>0} NTIME(n^c)\subseteq \bigcup_{c>0} DTIME(n^c)$ implies $\bigcup_{c>0} NTIME(n^c,d(n))\subseteq \bigcup_{c>0} DTIME(n^c,d(n))$. Hence, by definition of $P(d(n))$ and $NP(d(n))$, $NP(d(n))\subseteq P(d(n))$. Then $NP(d(n))=P(d(n))$. □

Therefore, if $P=NP$, then $P(d(n))=NP(d(n))$ for all density function $d(n)$ positive almost everywhere. Conversely, we can prove that if $P\neq NP$, then there exists a density function $d(n)$ other than 1 for which $P(d(n))\neq NP(d(n))$.

Lemma 4-2-3: If $P\neq NP$, then there exists a density function $d(n)$, $0<d(n)<1$ for all n, for which $P(d(n))\neq NP(d(n))$.

Proof: Suppose not, that is for all $d(n)$ for which $0<d(n)<1$ almost everywhere, $P(d(n))=NP(d(n))$. Let $L'\subseteq\Sigma^*$ be the hard language of Lemma 4-2-1 accepted by some deterministic machine M'. Let $L\subseteq\Sigma^*$ be any language in $NP-P$ and let M be a machine accepting L in non-deterministic polynomial time. Let 1 denote some symbol of Σ. We define:

$$L_1=\{w: w\notin 1^* \text{ and } w\in L\ \}\cup\{\ w: w\in 1^* \text{ and } w\in L'\}$$

and:

$$L_2=\{w: w\in 1^* \text{ and } w\in L\}\ \cup\ \{\ w: w\notin 1^* \text{ and } w\in L'\ \}$$

Thus, $L=L_1\cap(\Sigma^*-1^*)\cup(L_2\cap 1^*)$ $=\{w: w\notin 1^*\ \&\ w\in L_1\ \ or\ \ w\in 1^*\ \&\ w\in L_2\}$. Intuitively, the language L_1 is the language L, except on words of the form 1^* which require long time computations, since on 1^*, L is equal to L'. Conversely, accepting L_2 will require long time computations for all words except those of the form 1^* on which L_2 agrees with L. The basic idea of the proof is to contradict the presence of L in $NP-P$ by showing the existence of a deterministic Turing machine accepting L in polynomial time.

Furthermore, we claim that $L_1 \in NP(\frac{|\Sigma|^n-1}{|\Sigma|^n})$ and $L_2 \in NP(\frac{1}{|\Sigma|^n})$. For example, a non-deterministic machine IO-polynomial time bounded with density $\frac{|\Sigma|^n-1}{|\Sigma|^n}$ for L_1 would switch between M' and M, machines for L' and L respectively, depending on whether the input is in 1^* or not. The analogous machine for L_2 would do the reverse switching.

But we assumed that $NP(d(n))=P(d(n))$ for any $d(n)$. Then let M_1 and M_2 be deterministic machines accepting L_1 and L_2 in IO-time n^c with densities $\frac{|\Sigma|^n-1}{|\Sigma|^n}$ and $\frac{1}{|\Sigma|^n}$, respectively.

We claim that M_1 cannot halt on infinitely many words of the form 1^* in time n^c, because, otherwise, we could build a machine for L' that halts for infinitely many words in time n^c, which would contradict the properties of L' derived in Lemma 4-2-1. For example, one such machine for L' would check first whether the input is of the form 1^* or not, and if it is in 1^* then it would simulate M_1. If the input is not in 1^* then it would simulate the regular machine M' for L'. Since such a machine cannot exist, M_1 cannot accept/reject in time n^c words of the form 1^n for $n \geq k_1$, for some k_1. Therefore, since M_1 operates in IO-time n^c with density $\frac{|\Sigma|^n-1}{|\Sigma|^n}$, M_1 must accept/reject words w, $w \notin 1^n$, $|w|>k_1$ in time n^c.

By similar arguments, M_2 accepts/rejects in time n^c words w, $w \in 1^n$, $|w| > k_2$, for some k^2. Now, let $k = \max[k_1, k_2]$ and consider machine M'' that acts on input w as follows.

(i) If $|w| \leq k$, then M'' accepts w if and only if $w \in L$

(ii) If $|w| > k$ and $w \notin 1^*$, then M'' simulates M_1 on w.

(iii) If $|w| > k$ and $w \in 1^*$, then M'' simulates M_2 on w.

We claim that the language accepted by M'' is L. Consider any word w. If $|w| \leq k$, then w is in $L(M'')$ if and only if w is in L, by condition (i). Otherwise, if $|w| > k$, there are two cases:

(1) $w \notin 1^*$. Then, by condition (ii), w is in $L(M'')$ if and only if w is in $L(M_1) = L_1$ if and only if w is in L.

(2) $w \in 1^*$. By condition (iii), w is in $L(M'')$ if and only if w is in $L(M_2) = L_2$ if and only if w is in L.

Therefore, in any case w is in $L(M'')$ if and only if w is in L. Thus, $L(M'') = L$.

Furthermore, we claim that machine M'' operates in polynomial time. The information for step (i) can be recorded in the finite state control of M'', since there are only a constant k of those words. Step (ii) takes at most n^c computation steps, since for words w not in 1^*, such that $|w| > k \geq k_1$, M_1 operates in time n^c. Similarly, step (iii) is time bounded by n^c, since for inputs w in 1^*, such that $|w| > k \geq k_2$, M_2 operates in time n^c. Thus, for any L in $NP-P$ we can build a deterministic machine accepting L in polynomial time. Then $P = NP$. But this is a contradiction and so there must exist a density function $d(n)$, $0<d(n)<1$, for which $P(d(n)) \neq NP(d(n))$. □

Lemmas 4-2-2 and 4-2-3 together imply next result.

Theorem 4-2-4: $P \neq NP$ if and only if there exists a positive $d(n)$ for which $P(d(n)) \neq NP(d(n))$.

We know that there are oracles A and B for which $P^A = NP^A$ and $P^B \neq NP^B$* [Bake75]. These contradictory results involving oracles indicate that the existing complexity methods are probably insufficient to settle whether $P = NP$ or not.

However the results above involving density functions do not have the contradictory aspect of oracles. Any proof of $P(d(n)) \neq NP(d(n))$ does imply that $P \neq NP$. Therefore, there would appear to be no obvious connection between the role of oracles and the role of density functions in complexity theory.

Note that we can generalize the proof of Theorem 4-2-4 to show the following property.

Let $f(n) \leq g(n)$ and $f(n) \geq n$ everywhere. $DBOUND(f(n))$ is properly contained in $NBOUND(g(n))$ if and only if there exists a positive density function $d(n)$ such that $DBOUND(f(n), d(n), U)$ is properly contained in $NBOUND(f(n), d(n), U)$.

The above property would tie the hierarchy problems and the open trade-off problems of standard complexity theory to those of IO-complexity theory.

* P^A (NP^A) is defined as the set of languages accepted in polynomial time by deterministic (non-deterministic) Turing machines with oracle A

Another issue to be considered is under what circumstances, we can reverse the translational lemmas of chapter 2, Lemmas 2-3-1 to 2-3-4, to show that:

$$XBOUND(f(n),d(n),U) \subset XBOUND(g(n),d(n),U)$$

implies

$$XBOUND(f(n)) \subset XBOUND(g(n))$$

We could use similar techniques to the proof of Lemma 4-2-3 to show that this certainly holds for $d(n)=r$ for all n, r some fixed rational number. For example, for $d(n)={}^1/_2$, first note that it suffices to consider L in $XBOUND(f(n))$ over alphabet $\{0,1\}$. Let L_0 be $\{w \textit{ in } L\textit{: } w \textit{ starts with } 0\}$ and L_1 be $\{w \textit{ in } L\textit{: } w \textit{ starts with } 1\}$ and L' the hard language of Lemma 4-2-1. Then from $L_0 \cup \{1w : 1w \in L'\}$ in $XBOUND(f(n),d(n),U) \subseteq XBOUND(g(n),d(n),U)$, we get L_0 in $XBOUND(g(n))$ and similarly L_1 in $XBOUND(g(n))$, hence L in $XBOUND(g(n))$. Note that when such a result holds, any hierarchy for $XBOUND(f(n))$ immediately extends to $XBOUND(f(n),d(n),U)$.

4-3 Conjectures on E and NE

In section 4-3, we showed that the existence of a density function for which the separation of deterministic and the non-deterministic polynomial classes would imply that P is properly contained in NP. In this section, we investigate what happen if there exists $d(n)$ for which $P(d(n))=NP(d(n))$.

Let $E = \bigcup_{c>0} DTIME(2^{cn})$ and $NE = \bigcup_{c>0} NTIME(2^{cn})$. These are the exponential complexity classes of the worst case complexity. The next result relates any collapse of the type $P(d(n))=NP(d(n))$ to the collapse $E=NE$. The implication $NP=P$ implies $NE=E$ was shown the first time by Book [Book74]. By Theorem 4-2-4,

$NP=P$ if and only if for all positive density functions $d(n)$, $P(d(n))=NP(d(n))$. Therefore, if for all density functions $d(n)$, $P(d(n))=NP(d(n))$, then $NE=E$. However, the next result shows that the existence of any positive density function $d(n)$ for which $P(d(n))=NP(d(n))$ suffices to imply the convergence of the classes E and NE.

Theorem 4-3-1: If there exists a positive density function $d(n)$ such that $d(n)$ is computable in polynomial time and $P(d(n))=NP(d(n))$, then $E=NE$.

Proof: Suppose that $E \neq NE$. Let $L' \subsetneq \Sigma^*$ be a language in $NE-E$ and let $1 \in \Sigma$. Let $x(w)$ be the number represented by $1w$ in base $|\Sigma|$ and let

$$T=\{1^{x(w)}: w \in L'\}$$

Let M' accept L' non-deterministically in time 2^{cn}. We claim that the language T is in NP. In T we increase the length of the input exponentially in order to make our Turing machine M' for L' run more quickly relative to the input size on T. Such a machine M_T for T, on $z=1^{x(w)}$, would translate it to w and simulate M' acting on w. The translation of $1^{x(w)}$ to w takes at most $k_1 x(w)$ time steps, for some $k_1>0$, as detailed in Theorem 3-2-1, condition (i). Machine M' runs in time $2^{c|w|}$ for inputs w, machine M_T has as input $z=1^{x(w)}$ of length $x(w) \geq |\Sigma|^{|w|}$, since $x(w)$ represents $1w$ in base $|\Sigma|$. Thus M_T runs in time $k_1 x(w)+2^{c|w|} \leq k_1 x(w)+|\Sigma|^{k_2|w|} \leq k_1 x(w)+x(w)^{k_2} \leq k_4 x(w)^{k_3}$, i.e. time polynomial in $|z|=x(w)$.

Consider any positive density function $d(n)$. Let $m(n)$ denote the least positive integer such that $d(n) \leq \frac{m(n)}{|\Sigma|^n}$ for each n. Let $y(n)$ be the representation of length n in Σ^* of $(m(n)-1)$; since $(m(n)-1)<|\Sigma|^n$, $y(n)$ exists. Let $<$ denote the canonical order. Consider the language L that follows.

$$L=T\cup\{w:w\leq y(|w|)\ \&\ w\notin 1^+\}\cup\{w:w>y(|w|)\ \&\ w\in L''\ \&\ w\notin 1^+\}$$

where L'' is the hard language of Lemma 4-2-1, accepted by some deterministic machine M''. Thus L is composed of three parts. The set T is the first part of L. The second set is composed of the words of length n that occur before $y(n)$ in canonical order. Also L has a final part which needs long computation time; which are the words of length n greater than $y(n)$ that belong to L''. Our aim is to show that if $L\in P(d(n))$, then the set T must be recognized in polynomial time.

Consider a Turing machine M'' for L that acts on input z as follows.

(i) If $z\in 1^+$, then simulate M' on w, where $z=1^{x(w)}$.

(ii) If $z\notin 1^+$, but $z\leq y(|z|)$, then accept z.

(iii) Otherwise, simulate M'' on input z.

We have already seen that step (i) takes at most $k|z|^{k'}$ time steps. Step (ii) is bounded by some fixed polynomial in $|z|$, since by hypothesis $d(n)$ is computable in polynomial time. For each length n, there is one word 1^n accepted/rejected in (i) plus $(m(n)-1)$ words accepted/rejected in (ii). Thus, there is a total of at least $m(n)$ words accepted/rejected by M'' in time $n^{c'}$ for some fixed c'. Hence M'' is of IO-time complexity $n^{c'}$ with density $d(n)\leq\frac{m(n)}{|\Sigma|^n}$. Therefore, $L\in NP(d(n))$.

But, by hypothesis $P(d(n))$=$NP(d(n))$. Thus $L\in P(d(n))$.

Let M be a deterministic Turing machine accepting L in IO-time $n^{c'}$ with density $d(n)$. Then the running time of machine M must exceed $n^{c'}$ time steps on inputs of the type (iii) or Lemma 4-2-1 would not be valid. One machine to contradict Lemma 4-2-1 would check whether the input is not of the form 1^+ or if the input is less or equal $y(n)$ in canonical order over words of length n and then switch between machines M and M'. Therefore, machine M must have running time on the inputs w of type 1^+ and inputs $w \leq y(|w|)$ not exceeding $|w|^{c'}$, in order to have density $d(n)$.

But then the set

$$\{w: 1^{x(w)} \in L\} = \{w: 1^{x(w)} \in T\} = L'$$

can be recognized in deterministic exponential time by an algorithm based on M. For example, one such machine would read the input w, translate it to $x(w)$ and simulate M. Machine M spends $|x(w)|^{c'}$ on inputs $1^{x(w)}$, thus this algorithm spends $|\Sigma|^{c'|w|} \leq 2^{c''|w|}$ on input w, for some $c''>0$. But then $L' \in E$. □

Notice that it is not known whether $E = NE$ would imply the existence of a density $d(n)$ for which $P(d(n)) = NP(d(n))$; it is known that $P = NP$ implies $E = NE$ [Book74] and [Hart83a] but not whether $E = NE$ implies $P = NP$.

4-4 Polynomial Space

We observe that we can expand the methods used here for time bounds to the space complexity. For example, we can define $PSPACE(d(n))$ and $NPSPACE(d(n))$ as follows.

Definition 4-4-1: Let $0 \leq d(n) \leq 1$. We define:

(i) $PSPACE(d(n)) = \bigcup_{c>0} DSPACE(n^c, d(n), U) = \{L : \exists$ deterministic Turing machine M accepting L in IO-space n^c with density $d(n)$ and uniform probability distribution over the input alphabet of M, for some $c > 0\}$

(ii) $NPSPACE(d(n)) = \bigcup_{c>0} NSPACE(n^c, d(n), U) = \{L : \exists$ non-deterministic Turing machine M accepting L in IO-space n^c with density $d(n)$ and uniform probability distribution over the input alphabet of M, for some $c > 0\}$

Since is already known that $PSPACE = NPSPACE$, we can make use of Theorem 2-3-5 to prove that $PSPACE(d(n)) = NPSPACE(d(n))$.

Theorem 4-4-1: Let $d(n)$ be positive. Then $PSPACE(d(n)) = NPSPACE(d(n))$.

Therefore, analogous to the worst-case complexity where non-determinism does not add resources in terms of polynomial space, we can say that every problem solvable non-deterministically within polynomial IO-space with density $d(n)$ can be solved in polynomial IO-space with density $d(n)$ by a deterministic machine for any positive density $d(n)$.

We also can make use of Theorem 2-4-5 to prove that $NP(d(n))$ is contained in $PSPACE(d(n))$, since NP is contained in $PSPACE$.

Theorem 4-4-2: Let $d(n)$ be positive. Then $NP(d(n)) \subseteq PSPACE(d(n))$.

4-5 Approximation Languages

We turn to the question of finding approximate solutions to hard problems. Given a language L, which might require a lot of resource time or space to recognize, maybe we can be satisfied with another language L', which costs less time or space to recognize. Obviously, we are not satisfied with any language L'. We require that L' solves part of the problem that L is supposed to represent. By solve we mean determine the correct answer, i.e. whether a word w belongs to L or not, and provide a proof that it is correct [John84].

We say that languages L and L' agree on word w if w is in L if and only if w is in L'. Given an off-line deterministic Turing machine M', we select from the definition of M' a set of states I. We require that whenever M' halts for word w in some state $s \in I$ then $L'=L(M')$ and the language L agree in word w, that is, word w belongs to L if and only if it belongs to L'.

We want the language L', which is an approximation for L, to agree a "lot" with L and to be recognized in a moderate amount of time or space. More formally, we say:

Definition 4-5-1: Let L be a language over Σ^*. We say that a language L' is in $APPROXIMATION\text{-}DTIME(L, f(n), d(n))$ if there is an off-line multitape deterministic Turing machine M' accepting L' with a special set I of states of M' satisfying the following.

(i) M' is of worst-case time complexity $f(n)$.

(ii) If M' halts for input w in some state $s \in I$, then L and L' agree on w.

(iii) $d(n) \leq P[M' \textit{ halts on } w \textit{ in a state of } I \,/\, |w|=n] = \sum_{|w|=n:\, M' \textit{ halts on } w \textit{ in } s \in I} P[X=w/n]$.

Notice that conditions (ii) and (iii) of Definition 4-5-1 imply that L and L' agree on w with *at least* probability $d(n)$, since $d(n) \leq P[M' \textit{ halts on } w \textit{ in a state of } I \,/\, |w|=n] = \sum_{|w|=n:\, M'(w) \textit{ halts on } w \textit{ in } s \in I} P[X=w/n] \leq \sum_{L \text{ and } L' \textit{ agree on } w} P[X=w/n]$.

Note that for time bounds the requirement of M' being of worst-case time complexity $f(n)$ imply that M' is an always halting machine. However, a machine can be of space complexity $f(n)$ but not halt for all inputs. Thus, for space bounds, we consider only always halting machines.

Definition 4-5-2: Let L be a language over Σ^*. We say that a language L' is in $APPROXIMATION\text{-}DSPACE(L, f(n), d(n))$ if there is an off-line multitape always halting deterministic Turing machine M' accepting L' with a special set I of states of M' satisfying the following.

(i) M' is of worst-case space complexity $f(n)$.

(ii) If M' halts for input w in some state $s \in I$, then L and L' agree on w.

(iii) $d(n) \leq P[M' \textit{ halts on } w \textit{ in a state of } I \,/\, |w|=n] = \sum_{|w|=n:\, M' \textit{ halts on } w \textit{ in } s \in I} P[X=w/n]$.

Notice that the requirement that M' be a always halting machine is not a constraint. For any language L' in $DSPACE(f(n))$ there is an always halting deterministic machine M' accepting L' and operating within space bound $f(n)$, provided that $f(n)$ is space constructible.

We want to relate the sets *APPROXIMATION* and the IO-complexity classes. Suppose that language L is recursive and that we have a language L' in $APPROXIMATION\text{-}DBOUND(L, f(n), d(n))$, then we can find a IO-complexity class to which L belongs as follows.

Lemma 4-5-1: Let $d(n)$ be positive. Let L be recursive and $APPROXIMATION-DTIME(L,f(n),d(n)) \neq \emptyset$. Then $L \in DTIME(f(n),d(n))$.

Proof: Since L is recursive, let M be a deterministic always halting Turing machine accepting L. Let $L'=L(M')$ be in $APPROXIMATION-DTIME(L,f(n),d(n))$ with machine M' operating in time $f(n)$ and selected set of states I and consider a deterministic Turing machine M_1 which behaves on input w as follows.

(i) Simulate M' on w.

(ii) If M' does halt on a state of I, then accept w if and only if M' accepts w.

(iii) Otherwise, simulate M on w, accepting w if and only if M accepts w.

Consider any word w. If M' halts in a state of I, then M_1 accepts w if and only if M' accepts w. But, whenever M' halts in a state of I, machine M' accepts w if and only if w is in L. Furthermore, if M' does not halt in a state of I, then machine M_1 simulates machine M on input w; thus w is in $L(M_1)$ if and only if w is in $L(M)=L$. Therefore, for any word w, w is in $L(M_1)$ if and only if w is in L. Therefore, the language accepted by machine M_1 is L.

Let $T_1(w)$ be the running time of M_1 on input w. Conditions (i) to (ii) take at most $f(|w|)$ computation steps, since M' is of worst-case complexity $f(n)$. Then:

$$\sum_{|w|=n:T_1(w)\leq f(n)} P[X=w/n] \geq \sum_{|w|=n:M_1 \textit{ halts on } w \textit{ in (ii)}} P[X=w/n] =$$

$$\sum_{|w|=n:M' \textit{ halts on } w \textit{ in } s\in I} P[X=w/n] \geq d(n),$$ since $L' \in$

APPROXIMATION−DTIME $(L, f(n), d(n))$.

Therefore, by the definition of IO-complexity classes, $L = L(M_1) \in DTIME(f(n), d(n))$. □

Conversely, suppose that we know that $L \in DTIME(f(n), d(n))$, then we can find an approximation language L' for L as follows.

Lemma 4-5-2: Let $f(n)$ be a monotonic increasing time constructible function with $\inf_{n\to\infty} \frac{f(n)}{n} = \infty$, and let $d(n)$ be positive. Then $L \in DTIME(f(n), d(n))$ implies $APPROXIMATION-DTIME(L, f(n), d(n)) \neq \emptyset$.

Proof: If $L \in DTIME(f(n), d(n))$, then by Corollary 2-3-6 there is machine M that makes $L \in DTIME(\frac{f(n)}{2}, d(n))$. Consider machine M' with $I = \{Y, N\}$ and set of accepting states $F = \{Y\}$ which behaves on input w of size n as follows.

(i) Set a $\frac{f(n)}{2}$ counter on a working tape of M'. Machine M' will simulate machine M for $\frac{f(n)}{2}$ steps.

(ii) Simulate M on input w. Each step of M increases the count by one.

- if M halts and accepts w,then M' accepts w halting in state Y.
- if M halts and rejects w,then M' rejects w halting in state N.

(iii) If M does not halt within $\frac{f(n)}{2}$ steps, then M' rejects w, halting in a state not in I.

We have to prove that $L'=L(M') \in APPROXIMATION\text{-}DTIME(L,f(n),d(n))$. We claim that machine M' halts within $f(n)$ steps. Machine M' spends $\frac{f(n)}{2}$ time steps for the simulation of M on input w, plus the additional step of increasing the counter by one at each cycle. There are $\frac{f(n)}{2}$ cycles and, thus, M' halts within $\frac{2f(n)}{2}$ time steps.

Machines M' and M agree on all words accepted/rejected in step (ii) with M' halting in a state in I. But those are the words accepted or rejected by M within $\frac{f(n)}{2}$ steps. But $L(M) \in DTIME(\frac{f(n)}{2},d(n))$, so we have:

$$P[M' \textit{ halts on } w \textit{ in a state of } I \;/\; |w|=n] = \sum_{|w|=n:M' \textit{ halts in } (ii)} P[X=w/n] =$$

$$\sum_{|w|=n:M \textit{ halts on } w \textit{ within } \frac{f(n)}{2} \textit{ steps}} P[X=w/n] \geq d(n).$$

Thus, $L'=L(M') \in APPROXIMATION\text{-}DTIME(L,f(n),d(n))$.□

Lemmas 4-5-1 and 4-5-2 provide a strong relationship between the IO-complexity classes and the approximation languages as follows.

Theorem 4-5-3: Let $f(n)$ be a monotonic increasing time constructible function with $\inf_{n\to\infty} \frac{f(n)}{n}=\infty$, and let $d(n)$ be positive. Let L be a recursive language. Then $L \in DTIME(f(n),d(n))$ if and only if $APPROXIMATION\text{-}DTIME(L,f(n),d(n)) \neq \emptyset$.

Theorem 4-5-3 provides another interpretation for the classes $DTIME(f(n),d(n))$ in terms of approximation languages; it says that the recursive languages of $DTIME(f(n),d(n))$ are those languages L which can be approximated by $f(n)$ bounded machine agreeing with L on w with probability at least $d(|w|)$. We can apply all the results of the IO-complexity theory to the average complexity defined by the *APPROXIMATION* sets.

There are similar results for space bounds. However we must be careful, since a machine can be space bounded and non-halting.

Theorem 4-5-4: Let $f(n)$ be a monotonic increasing space constructible function and let $d(n)$ be positive. Then $APPROXIMATION\text{-}DSPACE(L,f(n),d(n)) \neq \emptyset$ if and only if $L \in DSPACE(f(n),d(n))$.

Proof: The proof is quite similar to the proof for time bounds, so we follow that notation. Let $L' \in APPROXIMATION\text{-}DSPACE(L,f(n),d(n))$ using machine M' with special set I of states. Consider deterministic machine M_1 that acts on input w as follows.

(i) Mark $f(n)$ cells in a working tape.

(ii) Simulate M' on w using at most $f(n)$ working cells. If M' does halt in state $s \in I$ then accept/reject as M' does.

(iii) Otherwise, it simulates M on input w.

Whenever machine M' halts in a state of I for input w, M_1 accepts input w if and only if M' accepts w if and only if w is in L. Otherwise, that is whenever M' does not halt in a state of I, machine M_1 executes step (iii), since the simulation on (ii) always halts, because M' is an always halting machine. But then, also in step (iii), M_1 accepts input w if and only if w is in L. Therefore, the language accepted by machine M_1 is L.

Furthermore, any word of length n accepted/rejected by M' within less than $f(n)$ cells is accepted/rejected by M_1 using less than $f(n)$ working tape cells in step (ii). Then M_1 is $(f(n))$ IO-space bounded with density $d(n)$. Thus $L \in DSPACE(f(n), d(n))$.

Conversely, suppose that $L \in DSPACE(f(n), d(n))$. Thus, by Lemma 3-3-1 there is machine M accepting L within space $f(n)$ and density $d(n)$ that halts for all words that respects the bound $f(n)$. Then consider machine M_2 with final set $F=\{Y\}$ and set $I=\{Y,N\}$ that acts on input w as follows.

(i) Mark $f(n)$ cells on a working tape.

(ii) Simulate M using the marked cells.

- If M accepts w, then M_2 accepts w on state Y.
- If M rejects w, then M_2 rejects w on state N.

(iii) Otherwise, if M tries to use more than $f(n)$ cells, then M_2 rejects w.

Let $S(w)$ denote the space spent on input w by machine M. By hypothesis, machine M operates in IO-space $f(n)$ with density $d(n)$. Whenever $S(w) \leq f(n)$ for input w of length n, machine M_2 halts in step (ii) and accepts w if and only if M accepts w. Thus it halts on state Y or N in I and agrees with M. Therefore, conditions (ii) and (iii) of Definition 4-5-2 are met. For the other words, machine M_2 halts and rejects them using space less than $f(n)$, too. Thus the language accepted by machine M_2 is in $APPROXIMATION{-}DSPACE(L, f(n), d(n))$.□

4-6 Non-existence of Approximation Languages

As a consequence of the interpretation of the IO-complexity classes as families of approximated languages, we get some results related to the existence of solutions for hard problems. For example there are languages so hard that they do not even have an approximation computable within fixed time and space bounds. As a direct consequence of the IO-complexity hierarchy results applied to the *APPROXIMATION* sets, we get results such as the following.

Corollary 4-6-1: Let $f(n)$ and $g(n)$ be monotonic increasing space constructible functions such that $\inf_{n \to \infty} \frac{g(n)}{f(n)} = \infty$. There is a language L in $DSPACE(g(n),1)$ such that for all density function $d(n)$ that are positive infinitely often, $APPROXIMATION\text{-}DSPACE(L, f(n), d(n)) = \emptyset$.

Proof: Let $f(n)$ and $g(n)$ be monotonic increasing space constructible functions such that $\inf_{n \to \infty} \frac{g(n)}{f(n)} = \infty$, and let $d(n)$ be positive for infinitely many n. Let L be in $DSPACE(g(n),1)$. If $APPROXIMATION\text{-}DSPACE(L, f(n), d(n)) \neq \emptyset$ for some suitable $d(n)$ as above, then by Theorem 4-5-4, L would be in $DSPACE(f(n), d(n))$. By Theorem 3-3-6, there is a language L in $DSPACE(g(n),1)$ and not in $DSPACE(f(n), d(n))$, for any $d(n)$ positive infinitely often. Hence the desired result. □

The deterministic time hierarchy yields similar results, with the density function $d(n)$ different from zero almost everywhere.

Corollary 4-6-2: Let $f(n)$ and $g(n)$ be monotonic increasing time constructible functions such that $\inf_{n\to\infty}\frac{g(n)}{f(n)^2}=\infty$ and $f(n)\geq n$. There is a language L in $DTIME(g(n),1)$ such that for all density function $d(n)$, that are positive almost everywhere, $APPROXIMATION\text{-}DTIME(L,f(n),d(n))=\emptyset$.

Proof: Let $f(n)$ and $g(n)$ be monotonic increasing time constructible functions such that $\inf_{n\to\infty}\frac{g(n)}{f(n)^2}=\infty$ and $f(n)\geq n$, and let $d(n)$ be positive almost everywhere. Let L be in $DTIME(g(n),1)$. If $APPROXIMATION\text{-}DTIME(L,f(n),d(n))\neq\emptyset$, for some suitable $d(n)$ as above, then, by Theorem 4-5-3, L would be in $DTIME(f(n),d(n))$. By Theorem 3-2-1, there is a language L in $DTIME(g(n),1)$ and not in $DTIME(f(n),d(n))$. Hence the desired result. □

CHAPTER 5

FURTHER COMPLEXITY CLASSES

5-1 Introduction

Numerous models and classes of languages have been introduced in the literature. This chapter presents a few classes of languages, not previously analyzed in this dissertation, and their relation to the IO-complexity.

We start by defining average-case complexity classes, in particular complexity measures related to the concept of medians and means of a set of numbers. We study the connection between these average-case complexity classes and the IO-complexity sets. We show an interpretation of the median-case complexity classes in terms of IO-complexity classes with density $^1/_2$.

We follow by defining probabilistic computations. Here we introduce a different model of computation, the probabilistic Turing machine. We define probabilistic time and space for words and for length of computations. We introduce the concept of infinitely often complexity to several probabilistic polynomial bounded classes. We present some open problems related to the probabilistic IO-complexity.

It must be pointed out that the topics analyzed in this chapter are part of a much larger research area still under study and several questions are unanswered and unfinished.

5-2 The Median Case Complexity

Usually when we talk about expected complexity, we require that some kind of average (mean, median) over all words of length n be bounded by a function of n in all points. We start with the concept of median complexity.

We recall informally the concept of median. Suppose we have the values $v_1, v_2, \cdots v_k$ each one with given probability $P[X=v_i]$ for $1 \le i \le k$. The median of these values denoted by m is the least element v_i such that $P[X \le v_i] \ge 1/2$.

Let M be a Turing machine and let w be a word of length n. We define $T_{median}(n, P[X=w/n])$ as the median of the running time on words of length n by M with probability distribution $P[X=w/n]$. We define $S_{median}(n, P[X=w/n])$ as the median of the space spent on words of length n by M with probability distribution $P[X=w/n]$.

Definition 5-2-1: Let $T(w)$ and $S(w)$ be respectively the running time and the space spent on w by machine M. Let $P[X=w/n]$ be positive. We define:

(i) $T_{median}(n, P[X=w/n])$ is the least $T(y)$ such that $|y|=n$ and $\sum_{|w|=n:T(w) \le T(y)} P[X=w/n] \ge 1/2$.

(ii) $S_{median}(n, P[X=w/n])$ is the least $S(y)$ such that $|y|=n$ and $\sum_{|w|=n:S(w) \le S(y)} P[X=w/n] \ge 1/2$.

Once we have defined the median complexity measures as above, we can define the median complexity classes as follows.

Definition 5-2-2: Let Φ be a functor assigning to each alphabet Σ a positive probability distribution $P[X=w/n]$ over Σ^*.

(i) $MEDIAN\text{-}DTIME(f(n),\Phi)$ is the family of languages L for which there is a deterministic Turing machine M accepting L with $T_{median}(n,\Phi(\Sigma))\le f(n)$ for all n and probability distribution $\Phi(\Sigma)$ for the input alphabet Σ of M.

(ii) $MEDIAN\text{-}DSPACE(f(n),\Phi)$ is the family of languages L for which there is a deterministic Turing machine M accepting L with $S_{median}(n,\Phi(\Sigma))\le f(n)$ for all n and probability distribution $\Phi(\Sigma)$ for the input alphabet Σ of M.

We use $MEDIAN\text{-}DBOUND(f(n))$ to denote the union of the complexity classes $MEDIAN\text{-}DBOUND(f(n),\Phi)$ for all functors Φ that assign to each input alphabet Σ a positive probability distribution $P[X=w/n]$. The next theorems relate the median complexity classes to the IO-complexity classes.

Theorem 5-2-1:

$$MEDIAN\text{-}DTIME(f(n))=DTIME(f(n),{}^1/2)$$

Proof: Suppose that L is in $MEDIAN\text{-}DTIME(f(n))$. Thus, there is a deterministic Turing machine M accepting L for which $T_{median}(n,P[X=w/n])\le f(n)$, for some positive $P[X=w/n]$. Let $T(w)$ denote the running time on word w by machine M. Thus:

$$\sum_{|w|=n:T(w)\le f(n)} P[X=w/n]\ge \sum_{|w|=n:T(w)\le T_{median}(n,P[X=w/n])} P[X=w/n],$$

since $T_{median}(n)\le f(n)$.

But $\sum_{|w|=n:T(w)\le T_{median}(n,P[X=w/n])} P[X=w/n]\ge {}^1/2$, since $T_{median}(n,P[X=w/n])$ is the median of the values $T(w)$ for words w of length n. Therefore $\sum_{|w|=n:T(w)\le f(n)} P[X=w/n]\ge {}^1/2$. But then L belongs to $DTIME(f(n),{}^1/2)$.

Conversely, suppose that L is in $DTIME(f(n),{}^1/2)$. Then there exists a deterministic Turing machine M accepting L for which: $\sum_{|w|=n:T(w)\le f(n)} P[X=w/n]\ge {}^1/2$. However, the median $T_{median}(n,P[X=w/n])$ is by definition the *least* element $e(n)$ for which $\sum_{|w|=n:T(w)\le e(n)} P[X=w/n]$ is greater or equal ${}^1/2$. Thus $f(n)\ge T_{median}(n,P[X=w/n])$ and so L is in $MEDIAN-DTIME(f(n))$. □

By techniques similar to those in the proof above, we can relate the IO-complexity to others median complexity classes.

Theorem 5-2-2:

$$MEDIAN-DSPACE(f(n))=DSPACE(f(n),{}^1/2)$$

Notice that the theorems above give a useful interpretation of the median-complexity classes as IO-complexity classes, since the results of traditional complexity theory hold for the IO-complexity as shown in chapter 2. Therefore, we can apply the results of the worst-case complexity theory to the median-case complexity classes.

5-3 The Mean Case Complexity

We turn to the complexity classes related to the concept of mean. Let M be a Turing machine with running time $T(w)$ and space spent $S(w)$ on word w of length n. Let $P[X=w/n]$ be a positive probability distribution. We define $T_{mean}(n,P[X=w/n])$ as the mean of the running time of M on words of length n and

$S_{mean}(n, P[X=w/n])$ as the mean of the space spent by M on words of length n with probability distribution $P[X=w/n]$.

Definition 5-3-1: Let $T(w)$ and $S(w)$ be the running time and the space spent on w by machine M, respectively. Let $P[X=w/n]$ be positive. We define:

(i) $T_{mean}(n, P[X=w/n]) = \sum_{|w|=n} T(w)P[X=w/n]$.

(ii) $S_{mean}(n, P[X=w/n]) = \sum_{|w|=n} S(w)P[X=w/n]$.

The complexity classes for the mean case complexity can be defined as follows.

Definition 5-3-2: Let Φ be a functor assigning to each alphabet Σ a positive probability distribution over Σ^*.

(i) $MEAN\text{-}DTIME(f(n), \Phi)$ is the family of languages L for which there is a deterministic Turing machine accepting L with input alphabet Σ and $T_{mean}(n, \Phi(\Sigma)) \le f(n)$.

(ii) $MEAN\text{-}DSPACE(f(n), \Phi)$ is the family of languages L for which there is a deterministic Turing machine accepting L with input alphabet Σ and $S_{mean}(n, \Phi(\Sigma)) \le f(n)$.

We use $MEAN\text{-}DBOUND(f(n))$ to denote the union of the complexity classes $DBOUND(f(n), \Phi)$ for all functors Φ that assigns to each input alphabet Σ a positive probability distribution $P[X=w/n]$. The next results show that any language in a complexity class of the type $MEAN\text{-}DBOUND(f(n))$ belongs to a corresponding IO-complexity class for some density function.

Theorem 5-3-1: If a language L is in $MEAN-DTIME(f(n))$, then there exists a positive density function $d(n)$ for which L is in $DTIME(f(n),d(n))$.

Proof: Let L be in $MEAN-DTIME(f(n))$. Then there exists a deterministic Turing machine M accepting L for which $T_{mean}(n,P[X=w/n])=\sum_{|w|=n} T(w)P[X=w/n]\le f(n)$.

Let

$$d(n)=\sum_{|w|=n:T(w)\le f(n)} P[X=w/n]\ge \sum_{|w|=n:T(w)\le T_{mean}(n,P[X=w/n])} P[X=w/n],$$

since $f(n)\ge T_{mean}(n,P[X=w/n])$.

But there exists a word w of length n for which $T(w)\le T_{mean}(n,P[X=w/n])\le f(n)$. So $d(n)$ as defined is strictly greater than zero everywhere, since $P[X=w/n]$ is positive. Thus, there exists a density function $d(n)$ positive everywhere for which L is in $DTIME(f(n),d(n))$. □

Theorem 5-3-2: If a language L is in $MEAN-DSPACE(f(n))$, then there exists a positive density function $d(n)$ for which L is in $DSPACE(f(n),d(n))$.

Notice that the propositions above have been shown in only one direction. That is given that L is of mean-complexity $f(n)$ then there exists a IO-complexity class for L. The other way around, i.e. "does L in $DBOUND(f(n),d(n))$ imply that L is in $MEDIAN-DBOUND(f(n))$?" is an open problem. Certainly, it does hold if $d(n)=1$ almost everywhere. Another issue here is whether there is any relationship between $MEAN-DBOUND(f(n))$ and $MEDIAN-DBOUND(f(n))$ or not. Actually, we would not really expect it since there is not necessarily a relationship between mean and median.

5-4 Probabilistic Computations

The probabilistic approach has been shown to be efficient to solve a few problems that cannot be efficiently solved by deterministic methods, for example fast algorithms for primality testing [Rabi76]. These results suggest that probabilistic algorithms may be useful for solving other deterministically intractable problems.

We will study a formal model for probabilistic algorithms: the *probabilistic Turing machine* and its relationship to the IO-complexity. Informally, we can describe a probabilistic Turing machine as a computer with the ability to make random decisions. We recall some basic concepts [Gill77].

A probabilistic Turing machine M is a deterministic multitape Turing machine with distinguished states called *coin-tossing* states. For each coin-tossing state, the finite control of M specifies two possible next states. The computation of M is *deterministic* except that in coin-tossing states M tosses an unbiased coin to decide between the two possible next states. The tosses are independent of the result of previous tosses, thus the probability of a computation path is half of the number of tosses on the path.

The definition of probabilistic Turing machines can be extended by allowing that *unbiased* random decisions are made, that is the probability of getting heads can be different from the probability of getting tails. It can be shown that the resulting model has the same computational power as the unbiased model [Sant69].

The computation of a probabilistic Turing machine M is determined by its input and the outcomes of the coin tosses performed by M. The output of machine M on input w is a random variable representing the possible computations of M on w.

Thus, we define $M(w)$ as follows.

Definition 5-4-1: Let M be a probabilistic Turing machine and let w be an input to M. We define $M(w)$ as a random variable denoting the outputs of possible computations of M on w.

We denote by $Pr[M(w){=}y]$ the probability of the output of the computation of M on w be y. In general, a probabilistic Turing machine computes a random function; for each input w, the machine M produces output y with probability $Pr[M(w){=}y]$. We say that M *converges to* y on input w if $Pr[M(w){=}y]>{}^1/2$. Despite the fact that the output of a probabilistic Turing machine is not in general uniquely determined by the input, we can define the partial function computed by a probabilistic machine in terms of cutpoint ${}^1/2$ as follows.

Definition 5-4-2: The partial function f computed by a probabilistic Turing machine M is defined by:

$$f(w) = \begin{cases} y & \text{if } \textit{there exists } y \text{ for } \textit{which } M \textit{ converges on input } w \\ \textit{undefined} & \text{if } \textit{no such } y \textit{ exists} \end{cases}$$

We are primarily interested in Turing machines computing the partial characteristic functions of languages (i.e. 0, 1-valued functions). A probabilistic Turing machine computing the partial characteristic function of a language L is said to *recognize* L.

Definition 5-4-3: A probabilistic Turing machine M is said to accept language L and we denote this by $L = L^P(M)$ if for all inputs w, $M(w) \in \{0,1\}$ and $Pr[M(w)=1] > {}^1/2$ if and only if $w \in L$.

A probabilistic Turing machine can accept a language L and have a non-zero probability of rejecting words that belong to L, for example. Therefore, we should be capable of expressing these cases by defining error probability as follows.

Definition 5-4-4: The error probability of probabilistic machine M recognizing language L is the function e defined by:

$$e(w) = \begin{cases} Pr[M(w)=0] & \text{if } w \in L \\ Pr[M(w)=1] & \text{if } w \notin L \text{ and } Pr[M(w)=0] > {}^1/2 \\ \textit{undefined} & \text{if } w \notin L \text{ and } Pr[M(w)=0] \leq {}^1/2 \end{cases}$$

An useful probabilistic algorithm should have small probability of error. At the very least, the error probability should be uniformly bounded below ${}^1/2$ for all inputs.

Definition 5-4-5: A probabilistic Turing machine M accepts language L with bounded error probability if there exits a constant $k < {}^1/2$ such that $e(w) \leq k$ for every input w.

5-5 Probabilistic Complexity Classes

It is well known that the ability to make random decisions does not increase the computational power of Turing machines [Gill77]. However, one question that is raised often is whether probabilistic machines can compute more efficiently than deterministic machines, that is using less time or tape. Therefore, it is important to have an agreeable definition of bounded computations for probabilistic Turing machines.

Definition 5-5-1: The *Blum run time* T_B and the *Blum space* S_B of probabilistic Turing machine M on input w are defined by: [Gill72]

$$T_B(w) = \begin{cases} \textit{least } i \textit{ such that } Pr[\,M(w){=}y \textit{ in time } i\,] > {}^1/_2 & \text{if } M \textit{ converges on } w \textit{ to } y \\ \infty & \textit{otherwise} \end{cases}$$

$$S_B(w) = \begin{cases} \textit{least } i \textit{ such that } Pr[\,M(w){=}y \textit{ in space } i\,] > {}^1/_2 & \text{if } M \textit{ converges on } w \textit{ to } y \\ \infty & \textit{otherwise} \end{cases}$$

In terms of acceptance of languages by probabilistic Turing machines, definition 5-5-1 works as follows.

Definition 5-5-2: Let L be accepted by probabilistic Turing machine M. We define:

$$T_{BL}(w) = \begin{cases} \textit{least } i\textit{: } Pr[\,M \textit{ accepts } w \textit{ in time } i\,] > {}^1/_2 & \text{if } w \in L \\ \textit{least } i\textit{: } Pr[\,M \textit{ rejects } w \textit{ in time } i\,] \geq {}^1/_2 & \text{if } w \notin L \;\&\; Pr[M(w){=}0] > {}^1/_2 \\ \infty & \text{if } w \notin L \;\&\; Pr[M(w){=}0] \leq {}^1/_2 \end{cases}$$

$$S_{BL}(w) = \begin{cases} \textit{least } i\textit{: } Pr[\,M \textit{ accepts } w \textit{ in space } i\,] > {}^1/_2 & \text{if } w \in L \\ \textit{least } i\textit{: } Pr[\,M \textit{ rejects } w \textit{ in space } i\,] \geq {}^1/_2 & \text{if } w \notin L \;\&\; Pr[M(w){=}0] > {}^1/_2 \\ \infty & \text{if } w \notin L \;\&\; Pr[M(w){=}0] \leq {}^1/_2 \end{cases}$$

Gill [Gill77] has shown that the definitions in 5-5-1 have the property of being Blum complexity measures [Blum67]. That is, given an arbitrary probabilistic Turing machine M computing the partial function f, $T_B(w)$ ($S_B(w)$) is defined if and only if $f(w)$ is defined. In addition, there exists a recursive predicate of w and i that is true if $Pr[M(w){=}f(w) \textit{ in time (space) } i\,] > {}^1/_2$ and false otherwise.

The complexity classes yielded by languages recognized by probabilistic Turing machines can be defined as follows.

Definition 5-5-3: Let $g(n): N \rightarrow N$ be a recursive function. We define:

(i) $PRTIME(g(n))$ is the class of languages recognized by probabilistic Turing machines that have $T_{BL}(w) \leq g(|w|)$ for all inputs w.

(ii) $PRSPACE(g(n))$ is the class of languages recognized by probabilistic Turing machines that have $S_{BL}(w) \leq g(|w|)$ for all inputs w.

We define *polynomial bounded* probabilistic Turing machine as follows.

Definition 5-5-4: A probabilistic Turing machine M is polynomial time bounded if there exists a constant $c > 0$ such that *every* computation on any input w halts within time $|w|^c$.

Using this definition, we define complexity classes as follows.

Definition 5-5-5: We define:

(i) PP is the class of languages recognized by polynomial bounded probabilistic Turing machines.

(ii) BPP is the class of languages recognized by polynomial bounded probabilistic Turing machines with bounded error probability.

(iii) R is the class of languages recognized by polynomial bounded probabilistic Turing machines which have zero error probability for inputs not in the language.

Notice that the definitions above *do not* use the Blum run time to define polynomial bounded machines. For example, we alternatively could say that a probabilistic Turing machine is polynomial bounded if there exists a polynomial $p(|w|)$ such that $T_{BL}(w) \leq p(|w|)$ for all inputs w. It is an open problem if the definition above and Definition 5-5-4 converge for *every* polynomial bounded complexity class. Obviously, the classes PP are the same under both definitions. It can easily be shown that the classes R converge under both definitions. However, there is no trivial proof whether the class BPP does contain the same languages under both definitions or not.

The polynomial classes mentioned above were shown by Gill to satisfy the following relations: [Gill77]

$$P \subseteq R \subseteq \begin{cases} NP \\ BPP \end{cases} \subseteq PP \subseteq PSPACE$$

There has been some research about space bounded simulation of probabilistic machines by deterministic ones. It has been shown that $PRSPACE(f(n)) \subseteq DSPACE(f(n)^6)$ [Hunt79]. For time bounds the results already known yield only exponential simulations [Ajta85].

5-6 IO-Probabilistic Complexity

We say that a machine M respects the time bound $g(n)$ for word w if there is no computation of M on input w that takes more than $f(n)$ steps. We say that w respects the space bound $g(n)$ for M if there is no computation of M on w using more than $g(n)$ working tape cells. Similarly to the non-probabilistic case, we extend the concept of bounded computation to include sets with density functions as follows.

Definition 5-6-1: Let $P[X=w/n]$ be positive. Let M be a probabilistic Turing machine.

(1) We say that M is a $g(n)$ *IO-time bounded* Turing machine (or of *time IO-complexity* $g(n)$) with *density function* $d(n)$ and probability distribution $P[X=w/n]$ if

$$d(n) \leq \sum_{w:\,M \text{ respects time bound } g(|w|) \text{ for } w} P[X=w/n]$$

(2) We say that M is a $g(n)$ *IO-space-bounded* Turing machine (or of *space IO-complexity* $g(n)$) with *density function* $d(n)$ and probability distribution $P[X=w/n]$ if

$$d(n) \leq \sum_{w:\,M \text{ respects bound } g(|w|) \text{ for } w} P[X=w/n].$$

We say that machine M is of IO-time(space) complexity $g(n)$ with density $d(n)$ if there exists a positive probability distribution $P[X=w/n]$ over the input alphabet of M for which M is of IO-time(space) complexity $g(n)$ with density $d(n)$ and probability distribution $P[X=w/n]$. We can define probabilistic complexity classes as follows.

Definition 5-6-2: Let $0 \leq d(n) \leq 1$.

(i) $PRSPACE(g(n), d(n))$ is the class of languages recognized by $g(n)$ IO-space bounded probabilistic Turing machines with density function $d(n)$.

(ii) $PRTIME(g(n), d(n))$ is the class of languages recognized by $g(n)$ IO-time bounded probabilistic Turing machines with density function $d(n)$.

The definitions above were based on whether a machine M halts or not for *every* possible computation of M on input w. Definition 5-5-3 was based on the Blum run time of machine M on input w. However, the next result says that the two definitions are equivalent for density function 1.

Theorem 5-6-1: Let $g(n)$ be total recursive. Then

$$PRTIME(g(n)) = PRTIME(g(n), 1).$$

Proof: We claim that $PRTIME(g(n),1) \subseteq PRTIME(g(n))$. Let L be in $PRTIME(g(n),1)$. Then there is a probabilistic Turing machine accepting L that halts for every input w in bound $g(|w|)$. Thus the Blum run time of such machine on every input w is bounded by $g(|w|)$. Therefore, L is in $PRTIME(g(n))$.

On the other hand, we also claim that $PRTIME(g(n)) \subseteq PRTIME(g(n),1)$. Let L be in $PRTIME(g(n))$. So consider machine M accepting L with $T_{BL}(w)$ bounded by $g(|w|)$ for any input w and $k>0$. We define a probabilistic machine M' that simulates machine M on input w by at most $g(|w|)$ steps. If the computation of M' on w exceeds $g(|w|)$ steps, then M' rejects w. Otherwise, when M does not exceed $g(|w|)$ steps, M' accepts w if and only if M accepts w.

The language accepted by M' is $L(M)$, since $Pr[M(w)=1 \textit{ in time } g(|w|)] > {}^1/2$ for any word w in L, by the definition of $T_{BL}(w)$. Similarly, if w is not in L, then M rejects w in time $g(|w|)$ with probability greater than ${}^1/2$ and thus, M' rejects w within time $g(|w|)$. Therefore, M and M' accept the same language. Thus, L belongs to $PRTIME(g(n),1)$, since machine M' respects the bound $g(|w|)$ for any input w. □

Similarly, for space bounds we can prove that $PRSPACE(g(n)) = PRSPACE(g(n), 1)$.

Theorem 5-6-2: Let $g(n)$ be total recursive. Then

$$PRSPACE(g(n)) = PRSPACE(g(n), 1).$$

A probabilistic Turing machine M is said to be IO-polynomial bounded with density $d(n)$ if there is a polynomial $p(n)$ such that M is of IO-time complexity $p(n)$ with density $d(n)$. We enlarge the concept of probabilistic classes as follows.

Definition 5-6-3: Let $0 \le d(n) \le 1$. We define:

(i) $PP(d(n)) = \bigcup_{k>0} PRTIME(n^k, d(n)) =$ $\{L$: there exists probabilistic Turing machine M accepting L in IO-time n^k with density $d(n)$, for some $k>0\}$.

(ii) $BPP(d(n)) =$ $\{L$: there exists probabilistic Turing machine M accepting L with bounded error probability such that M operates in IO-time n^k with density $d(n)$, for some $k>0\}$.

(iii) $R(d(n)) =$ $\{L$: there exists probabilistic Turing machine M accepting L with zero error probability for any w, $w \notin L$, such that M operates in IO-time n^k with density $d(n)$, for some $k>0\}$.

Obviously, Definitions 5-5-6 and 5-6-3 converge for the IO-complexity classes with density function 1.

Theorem 5-6-3:

(i) $PP = PP(1)$;

(ii) $BPP = BPP(1)$;

(iii) $R = R(1)$.

The following relations among the classes defined above and the classes $P(d(n))$, $NP(d(n))$ and $PSPACE(d(n))$ are valid.

Theorem 5-6-4: Let $0<d(n)\le 1$. Then:

$$PP(d(n)) \subseteq PSPACE(d(n))$$

Proof: Let L be in $PP(d(n))$. Then L is accepted by some probabilistic Turing machine M with k working tapes, that operates in IO-time n^c with density $d(n)$, for some $c>0$.

Consider a word w that respects the bound n^c for machine M. Each computation path of M on w is deterministic and can be simulated using time n^c. Hence, each path uses at most n^c working tape cells, since it is time bounded by n^c.

Consider machine M' that acts on any input w as follows. M' on tape T_0 records the sum of the probability of accepting paths and on tapes T_i, $1\le i\le k$, simulates all possible paths of M on w. M' simulates each computation path of M, one at a time for at most n^c time steps, using always the same cells. If all the computations paths of M on w halt within n^c time steps, then M' accepts w if and only if the total probability recorded on T_0 is greater than $^1/_2$. If the word w respects the bound n^c for M, then the simulation is over. Otherwise, when M does not respect the bound n^c on w, then machine M' must continue simulating machine M on w until a decision is reached. But now M' simulates one step of each computation at a time, since M may have a non-halting computation path on input w. M' accepts w if and only if M accepts w.

For words w that respect the bound n^c for machine M, the number of cells used on tapes T_i, $1 \le i \le k$, is bounded by n^c. But each computation path has probability at least $\frac{1}{2^{n^c}}$, since n^c bounds the longest computation path for these words. But this number can be recorded using n^c cells on tape T_0. Therefore M' is of space complexity $2n^c$, sum of the cells scanned on tapes T_i, $0 \le i \le k$, with density $d(n)$. Thus L is in $PSPACE(d(n))$. □

Theorem 5-6-5: Let $0 < d(n) \le 1$. Then:

$$BPP(d(n)) \subseteq PP(d(n))$$

Proof: This is a straightforward consequence of the definitions of $BPP(d(n))$ and $PP(d(n))$. □

Theorem 5-6-6: Let $0 < d(n) \le 1$. Then:

$$R(d(n)) \subseteq NP(d(n))$$

Proof: Let L be in $R(d(n))$. Thus there is a probabilistic machine M recognizing L such that if w does not belong to L, then M does not have any computation path accepting w, or otherwise M would have non-zero error probability for some input not in the language. Hence M when viewed as a non-deterministic machine does not accept w either. If w is in L, then M has at least one accepting path, which suffices for the acceptance on the non-deterministic case. Thus, M viewed as a non-deterministic machine accepts L.

Furthermore, all words w for which probabilistic machine M respects the bound n^c, for some $c>0$, have no computation path exceeding n^c time steps. Therefore, for these words the running time of non-deterministic machine M on w is at most n^c. So, L belongs to $NP(d(n))$. □

Theorem 5-6-7: Let $0<d(n)\leq 1$. Then:

$$R(d(n))\subseteq BPP(d(n))$$

Proof: Suppose that L is in $R(d(n))$. Consider a probabilistic Turing machine M accepting L with zero error probability for inputs not in L, that operates within IO-time n^k with density $d(n)$.

Notice that if M has an accepting path for w, then w must be in L. This must happen because machine M does not have accepting computations when w is not in L, by definition of the complexity class $R(d(n))$.

Thus consider machine M' accepting L such that on input w of length n, M' sequentially simulates n times the behavior of machine M on w by at most n^k steps each time. If M has a computation that does not halt within n^k steps, then machine M' simulates the behavior of M on w without any time bound; M' accepts w if and only if M accepts w. If at some point M has an accepting path, then M' halts and accepts w. Otherwise, if all n computations halt and are rejecting computations, then M' rejects w.

If w is not in L, then w is not in $L(M')$ with zero error probability, since M has only rejecting paths for w. If w is in L, then M' must have accepting paths on w with probability greater than $^1/_2$, since machine M has such paths. For w in L, M' can make a mistake only when the n simulations of M on input w yield only rejecting paths. But machine M rejects inputs w in L with at most probability $^1/_2$. Thus M' have a probability of error bounded by $\frac{1}{2^n}$, since it simulates n machines M. Therefore, M' recognizes L with bounded error probability, $e(w) \leq \frac{1}{2^n} \leq \frac{1}{4} < \frac{1}{2}$, for all $n \geq 2$ and $w \in \Sigma^n$.

Furthermore, for words w that respect the bound n^k for machine M, all computations paths of M' on w halt computation in time n^{k+1}, since M always halts in time n^k for these words. Thus L is in $BPP(d(n))$. □

Theorem 5-6-8: Let $0 < d(n) \leq 1$. Then:

$$P(d(n)) \subseteq R(d(n))$$

Proof: A deterministic machine is a special case of a probabilistic Turing machine that makes no use of its randomness capacity and that makes no mistakes for any input. □

Theorem 5-6-9: Let $0 < d(n) \leq 1$. Then:

$$NP(d(n)) \subseteq PP(d(n))$$

Proof: Let L be in $NP(d(n))$. Let M be a non-deterministic machine accepting L within IO-time bound n^k with density $d(n)$, for some $k>0$. First, note that we can assume that M has a binary choice at every step and that all computations paths at least reach the bound n^k. Therefore, the computation tree of M on inputs w that respect the bound n^k has 2^{n^k} leaves.

Consider probabilistic machine M' that proceeds on input w as follows. First, M' tosses enough coins to get three computations paths . The first one is an accepting computation and has probability $\left[\frac{1}{2}-\frac{1}{8^{n^k}}\right]$. The second one is a rejecting computation and has probability $\frac{1}{8^{n^k}}$. The third one has probability $\frac{1}{2}$, and in this path M' simulates M but it also incorporates a time counter for n^k. If M' gets an answer just at n^k time steps, it halts with the answer of M with probability $\frac{1}{2}.\frac{1}{2^{n^k}}=\frac{1}{4^{n^k}}$. If the path does not halt at n^k, M' simulates deterministically the behavior of M on input w. Since M may not halt on w, M' simulates each step of each computation path of M on w one at a time.

Hence in all cases, if w is in L an accepting path will be added to M' with at least probability $\frac{1}{4^{n^k}}$. If w is not in L, then there is no accepting path of M on w and a rejection probability of $^1/2$ is added on this computation path. Thus, M' probabilistic recognizes L with at least probability $\frac{1}{2}+\frac{1}{4^{n^k}}>\frac{1}{2}$, for any input w of length n.

Furthermore, all inputs w for which M respects the bound n^k have the bound n^k respected by probabilistic machine M'. Thus, $L \in PP(d(n))$. □

We can summarize the inclusions above as follows.

$$P(d(n)) \subseteq R(d(n)) \subseteq \begin{cases} NP(d(n)) \\ BPP(d(n)) \end{cases} \subseteq PP(d(n)) \subseteq PSPACE(d(n))$$

It has been conjectured that neither $BPP \subseteq NP$ nor $NP \subseteq BPP$ [Gill77]. Thus much less $BPP(d(n)) \subseteq NP(d(n))$ nor $NP(d(n)) \subseteq BPP(d(n))$.

CHAPTER 6

CONCLUSIONS AND FURTHER RESEARCH

We have made a step toward a more general complexity theory, by establishing the theoretical basis for a complexity theory based on infinitely often conditions. The new complexity theory includes the worst-case complexity as a special case and at the same time has as valid most of the worst-case complexity properties, as shown in chapter 2.

As a direct consequence of the IO-hierarchies herein developed, we demonstrated the existence of very hard languages. We showed the existence of languages accepted with worst-case space bound $g(n)$ that cannot be accepted within IO-space bound $f(n)$ for any density function $d(n)$ that is non-trivial infinitely often, if function $f(n)$ satisfies $\inf_{n \to \infty} \frac{g(n)}{f(n)} = \infty$. For deterministic time, we proved a similar relation with $d(n)$ non-trivial almost everywhere. Thus these languages cannot have an approximated solution within any bound less than or equal to $f(n)$. Additional research could be done toward improving the above requirements; for example, we could ask for $\inf_{n \to \infty} \frac{f(n)}{g(n)} = 0$ and $d(n)$ non-trivial infinitely often, for space and time bounds.

The connection between sparse sets [Hart83a] and density functions was only mentioned but further relationships between them seems worthy of investigation. Closely related to the concept of sparse sets is the concept of tally sets [Book74]. In particular, the association between tally sets and density function $\frac{1}{2^n}$ could be useful. There are indications that this connection could be an auxiliary result for the converse of Theorem 4-3-1, i.e., whether $E=NE$ implies $P(d(n))=NP(d(n))$ or not.

Another interesting point for research is the relationship between oracles and density functions. Proposition 4-2-4 and the existence of oracles A and B for which $P^A=NP^A$ and $P^B \neq NP^B$ [Bake75] indicate the likelihood that there is no connection between density functions and relativized computations but does not rule out the possibility.

We demonstrated the connection between IO-sets and *APPROXIMATION* sets. The definitions of *APPROXIMATION* sets, Definitions 4-5-1 and 4-5-2, require the existence of a special set P of states. However, we could drop this requirement in the definition of *APPROXIMATION* sets. It is not hard to verify that, for example, Lemma 4-5-2 would still be valid with the new definition, that is L in $DTIME(f(n),d(n))$ implies $APPROXIMATION\text{-}DTIME(L,f(n),d(n)) \neq \emptyset$. However, the other way around, i.e. whether $APPROXIMATION\text{-}DTIME(L,f(n),d(n)) \neq \emptyset$ implies that L is in $DTIME(f(n),d(n))$ or not, does not seem to be a trivial problem. It would be interesting either to prove the implication or to find a problem that can be approximately solved within time $f(n)$ with density $d(n)$ which full solution cannot be accepted within time bound $f(n)$ with density $d(n)$.

Another interesting point is the addition of non-determinism to the sets *APPROXIMATION*. At first sight, it seems strange to add the power of non-determinism only to find an approximate solution. However, from a theoretical point of view the association seems to be intellectually challenging.

Several other complexity classes could be analyzed within the scope of the IO-complexity. In particular, the connection between average complexity and IO-complexity is worth of additional research. From the probabilistic part it remains open whether $NP(d(n)) \subseteq PP(d(n))$ or not. We conjecture on an affirmative answer since it is known that $NP \subseteq PP$ [Gill77]. On the other hand, the inclusion relations between the classes $BPP(d(n))$ and $NP(d(n))$ do not seem to have any strong evidence. It has been conjectured that neither $BPP \subseteq NP$ nor $NP \subseteq BPP$.

Finally, it must be pointed out that we could have followed an alternative approach for probabilistic computations. We can avoid artificially defining time and space for a probabilistic machine a deterministic function and consider the time and space for probabilistic Turing machines as stochastic functions. We call this second approach as a stochastic one in contrast with the probabilistic one viewed in chapter 5.

From this point of view, we should have defined M as $g(n)$ IO-time bounded with density $d(n)$ if the sum of the probabilities of every possible computation on words of length n that halts within time $g(n)$ is at least $d(n)$ for all n. More formally, we can define it as follows.

Definition 6-1: Let M be a probabilistic Turing machine and $g(n)$ be a function. We say that M is a $g(n)$ *IO-time bounded* Turing machine with *density function* $d(n)$ if

$$d(n) \leq \sum_{w:\,|w|=n} P[T(w) \leq g(n)]P[X=w/n]$$

We can define a polynomial time complexity class based on the above notions as follows.

Definition 6-2: $PS(d(n))$ is the class of languages L for which there exist a polynomial $p(n)$ and a probabilistic Turing machine M recognizing L such that M $p(n)$ IO-time bounded with density function $d(n)$.

Notice that any recursive language can be in, for example, $PS(\frac{2^n-1}{2^n})$.

Theorem 6-1: Let L be a recursive language. Then $L \in PS(\frac{2^n-1}{2^n})$.

Proof: Let L be accepted by some machine M. Then consider a probabilistic Turing machine M' and any input w of length n. Machine M' tosses n coins in a row. If the outcome is n heads, then M' simulates machine M. Otherwise, it tosses a coin one more time accepting w if the result of the last toss is head and rejecting w if this result is tail.

Machine M' on any input w of length n takes at most $n+1$ steps on all computation paths, except one; i.e. except when machine M' simulates M. But this computation path has probability only $\frac{1}{2^n}$, since there are 2^n equiprobable computation paths when n coins are tossed in a row. Thus for any word w of length n, the computation of M' on w is bounded by $n+1$ with probability at least $1-\frac{1}{2^n} = \frac{2^n-1}{2^n}$.

Furthermore, if we do not take in account the last computation path, the acceptance and the rejection probability of any word w is the same, by the construction of M'. Then, the final decision is left for the last path, which is a simulation of machine M. Thus the languages accepted by M and M' are the same. Therefore, $L \in PS\left(\frac{2^n - 1}{2^n}\right)$. □

Therefore it should be pointed out that the above proposition implies that the IO-complexity has its limitations when associated with stochastic bounds.

[Gold82] Goldberg, A., P. Purdom, and C. Brown, ''Average Time Analysis of Simplified Davis-Putnan Procedures,'' *Information Processing Letters* **15**, pp.72-75 (1982).

[Gold84] Goldberg, A. V. and A. Marchetti, ''On Finding the Exact Solution of a Zero-One Knapsack Problem,'' in *Proceedings 16th Annual ACM Symposium on Theory of Computing*, New York (1984).

[Grei84] Greibach, S. A., ''Computability and Complexity,'' , UCLA (1984). Course Notes for CS281A.

[Hart83a] Hartmanis, J., ''On Sparse Sets in NP-P,'' *Information Processing Letters* **16**, pp.55-60 (1983).

[Hart83b] Hartmanis, J, N. Immerman, and V. Sewelson, ''Sparse Sets in NP-P: EXPTIME versus NEXPTIME,'' pp. 382-391 in *Proceedings 15th ACM STOC* (1983).

[Hopc79] Hopcroft, J. E. and J. D. Ullman, *Introduction to Automata Theory, Languages and Computation,* Addison-Wesley (1979).

[Horo78] Horowitz, E. and S. Sahni, *Fundamentals of Computer Algorithms,* Computer Science Press (1978).

[Hunt79] Hunt, J. W., ''Topics in Probabilistic Complexity,'' , Stanford University (1979). Ph.D. dissertation.

[John84] Johnson, D. S., ''The NP-Completeness Column: An Ongoing Guide,'' *Journal of Algorithms*(5) (1984).

[Karp76] Karp, R. M., ''The Probabilistic Analysis of Some Combinatorial Search Algorithms,'' in *Algorithms and Complexity: New Directions and Recent Results*, ed. J. F. Traub (1976).

[Knut76] Knuth, D. E., ''Big Omicron and Big Omega and Big Theta,'' *SIGACT News* (1976).

[Laga83] Lagarias, J. C. and A. M. Odlyzko, ''Solving Low-Density Subset Sum Problems,'' pp. 1-10 in *Proceedings 24th Annual Symposium on Foundations of Computer Science*, Los Angeles (1983).

[Levi84] Levin, L. A., ''Problems Complete in Average Instance,'' in *Proceedings 16th Annual ACM Symposium on Theory of Computing* (1984).

REFERENCES

[Ajta85] Ajtai, M. and A. Wigderson, "Deterministic Simulation of Probabilistic Constant Depth Circuits," pp. 11-19 in *Proceedings 26th Annual Symposium on Foundations of Computer Science*, Portland, Oregon (1985).

[Angl77] Angluin, D. and L. G. Valiant, "Fast Probabilistic Algorithms for Hamiltonian Circuits," pp. 30-41 in *Proceedings 9th Annual ACM Symposium on Theory of Computing*, New York (1977).

[Bake75] Baker, T., J. Gill, and R. Solovay, "Relativizations of the P=?NP Question," *Siam Journal of Computing* **4**, pp.431-442 (1975).

[Blum67] Blum, M., "A Machine-Independent Theory of the Complexity of Recursive Functions," *Journal of the ACM* **14**, pp.322-336 (1967).

[Blum71] Blum, M., "On Effective Procedures For Speeding Up Algorithms," *J. ACM* **18**(2), pp.322-336 (1971).

[Book74] Book, R., "Tally Languages and Complexity Classes," *Information and Control* **26**, pp.186-193 (1974).

[Book70] Book, R. V., S. A. Greibach, and B. Wegbreit, "Time and Tape Bounded Turing Acceptors and AFL's," *Journal of Computer and System Sciences*(4) (1970).

[Cutl83] Cutland, N. J., *Computability: An Introduction to Recursive Function Theory,* Cambridge University Press (1983).

[Gare79] Garey, Michael and David Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness,* W. H. Freeman and Company (1979).

[Gill77] Gill, J., "Computational Complexity of Probabilistic Turing Machines," *SIAM Journal of Computing*(6) (1977).

[Gill72] Gill, J. T., "Probabilistic Turing Machines and Complexity of Computation," , Dept. of Mathematics, University of California, Berkeley (1972). Ph. D.

[Lewi81] Lewis, H. R. and C. H. Papadimitriou, *Elements of Theory of Computation,* Prentice-Hall (1981).

[Rabi76] Rabin, M. O., "Probabilistic Algorithms," in *Algorithms and Complexity: New Directions and Recent Results*, ed. J. F. Traub (1976).

[Sant69] Santos, E. S., "Probabilistic Turing Machines and Computability," in *Proceedings American Mathematical Society* (1969).

[Sewe83] Sewelson, V., "A Study of the Structure of NP," , Cornell University, Ithaca New York (August 1983). PhD dissertation.

[Yao77] Yao, A. C., "Probabilistic Computation: Toward a Unified Measure of Complexity," in *Proceedings 18th Annual Symposium on Foundations of Computer Science* (1977).