RESOURCE BOUNDED REDUCTIONS

RESOURCE BOUNDED REDUCTIONS

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad van doctor aan de
Universiteit van Amsterdam,
op gezag van de Rector Magnificus
prof. dr. P.W.M. de Meijer
in het openbaar te verdedigen in de
Aula der Universiteit
(Oude Lutherse Kerk, ingang Singel 411, hoek Spui)
op woensdag 2 juni 1993 te 15.00 uur.

door

Harry Matthijs Buhrman
geboren te Amsterdam

Promotores:	prof. dr. S. Homer
	dr. P. van Emde Boas
Co-promotor:	dr. L. Torenvliet

Promotie-commissie:
dr. A. Hoene
prof. dr. C.A.J. Klaassen
prof. dr. A. Schrijver
prof. dr. P.M.B. Vitányi

Faculteit Wiskunde en Informatica
Universiteit van Amsterdam
Plantage Muidergracht 24

Druk: Haveka B.V., Alblasserdam

Acknowledgments

First I would like to thank my promotores and Steven Homer and Peter van Emde Boas. Steve invited me in a very early stage of my research to come to Boston. My visits to Boston were very inspiring and invaluable. A substantial part of my research consists of joint work with him. We had many inspiring conversations, sometimes on a daily basis. I thank Steve for his support during my visits to Boston and via e-mail when I was in Amsterdam. Peter provided in the beginning a pleasant atmosphere that gave me above all the freedom to do things my way. Whenever I got stuck I could ask him for advice, that he either could give me *immediately* or, after consulting his oracle, the next day. In the end stages of writing this thesis he was of great value by reading my rapidly changing versions of the manuscript, debugging the proofs and being patient with me.

I thank my co-promotor Leen Torenvliet. Leen is the person who started it all and made it possible for me to do this research in the first place. I remember vividly his inspiring lectures when I was still a masters student. I thank him also for supporting me through the early stages and the numerous exciting 'brain storm sessions' we had. It would not have been possible to write this thesis without him.

This thesis is by no means the work of a single person. It consists of papers written with others. But also discussions and ideas of other people have contributed to this work. I would like to thank Edith Spaan, for many long and inspiring discussions, Luc Longpré for his insights in many things and his hospitality during my visits to Boston. Special thanks are due to Albrecht Hoene, who was my office mate for a very short period in Boston and inspired a lot of the work done back in Amsterdam. I thank Jim Kadin for working with me in very exciting surroundings. I thank Michiel Smid for the conversations we had in the beginning of my research and the frequent e-mail messages later. I would like to thank Alan Selman for sharing results and suggestions related to the last chapter of this thesis. His insights are invaluable. I would like to thank José Balcázar, Ricard Gavaldà, Judy Goldsmith, Bill Gasarch, Steve Fenner, Dick de Jongh, Klaus Ambos-Spies, Paul Vitányi, John Tromp, Peter van Helden and Sara Mocas for many inspiring conversations.

I thank the people of the Computer Science Department of Boston University for their support and hospitality during my visits. Thanks Chris Lynch, Raj, Himanshu Sinha, Venkie, Sue Nagy, Heather Wallick, Mimmo Parente, Gene Itkis, Bob Carter, Arif Bhatti, Biao Wang, Zhixiang Chen, Peter Gacs, Wayne Snyder and Assaf Kfoury.

I thank the people from the theoretical computer science and logic department of the University of Amsterdam for providing a pleasant atmosphere, in particular Harold Schellinx, Dirk Roorda, Wolfgang Lindner, Rineke Verbrugge, Domenico Zambella, Eric Aarts, Maarten de Rijke, Otto Moerbeek, Angelo welling, Peter de Jong and Theo Janssen. I thank Paul van Ulsen for reading my manuscript thoroughly. Special thanks go to the people who shared the office with with me: Mart Trautwein, Sophie Fischer, Huang Zhisheng, Sieger van Dennenheuvel and Martijn Spaan.

During my stay in Boston I spent some time with the Homers, as well in Boston as in the Berkshires. I want to thank Michelle, Ben, Katie and Sara for their hospitality.

Then there is the group of friends that have no direct contributions to the technical contents of this thesis. However without their presence it would have been impossible to even start writing a thesis. I thank them for listening to me when I needed it and for being there.

Most of all I want to thank my parents for all the support they gave me throughout the years. Of course without them I wouldn't exist, but even taking that fact for granted, they provided the ideal surroundings and were always there when I needed them. I owe more to them then I can say here.

Harry Buhrman
Amsterdam April 1993

Contents

Chapter 1

Introduction

In order to appreciate the history of the development of Structural Complexity Theory during the last two decades, we first have to stand back and look at the basic concepts of computation and complexity theory.

One of the starting points of computation theory can be seen in the challenge thrown at the mathematical community by Hilbert in his address at the International Congress of mathematicians in Paris in 1900:

‘Wer von uns würde nicht gern den Schleier lüften, unter dem die Zukunft verborgen liegt, um einen Blick zu werfen auf die bevorstehenden Fortschritte unserer Wissenschaft und in die Geheimnisse ihrer Entwicklung während der künftigen Jahrhunderte! Welche besonderen Ziele werden es sein, denen die führenden mathematischen Geister der kommenden Geschlechter nachstreben? Welche neuen Methoden und neuen Tatsachen werden die neuen Jahrhunderte entdecken – auf dem weiten und reichen Felde mathematischen Denkens?’

[‘Who of us would not gladly lift the veil, behind which the future lies hidden, to cast a glance at the upcoming advances of our science and the secrets of its development during the future centuries. What will be the particular goals that the leading mathematical minds of the comming generations will strive for? What new methods and new facts will the new centuries reveil – in the wide and rich field of mathematical thought?’]

This is the beginning of Hilbert’s famous speech, in which he poses

23 problems, that turned out to be important and fruitful for mathematics.

We would like to discuss one of them as starting point for the research area one can place this thesis in: Structural Complexity Theory. Let's go back to the speech:

'Unermeßlich ist die Fülle von Problemen in der Mathematik, und sobald ein Problem gelöst ist, tauchen an dessen Stelle zahllose neue Probleme auf. Gestatten Sie mir im Folgenden, gleichsam zur Probe, aus verschiedenen mathematischen Disziplinen einzelne bestimmte Probleme zu nennen, von deren Behandlung eine Förderung der Wissenschaft sich erwarten läßt.'

['The supply of problems in mathematics is inexhaustible, and as soon as one problem is solved numerous others come forth in its place. Permit me in the following, tentatively as it were, to mention particular definite problems, drawn from various branches of mathematics, from the discussion of which an advancement of science may be expected.']

Of the twenty three problems that he stated next, we would like to discuss number ten.

Entscheidung der Lösbarkeit einer diophantischen Gleichung

'Eine diophantische Gleichung mit irgendwelchen Unbekannten und mit ganzen rationalen Zahlenkoeffizienten sei vorgelegt: *Man soll ein Verfahren angeben, nach welchem sich mittels einer endlichen Anzahl von Operationen entscheiden läßt, ob die Gleichung in ganzen rationalen Zahlen lösbar ist.*'

[**Determination of the Solvability of a Diophantine Equation**

'Given a diophantine equation with any number of unknown quantities and with rational integral numerical coefficients: *To devise a process according to which it can be determined by a finite number of operations whether the equation is solvable in rational integers.*']

This problem asks for an effective method for solving diophantine equations. The answer, probably not as Hilbert expected, was negative. In 1970 Matijasevič [Mat70] proved that no such effective procedure exists. But what does it mean for this problem to have a negative solution? In order to be able to do this, the notion "effective method" has to be made precise.

The initial work and definition of such a formalism was done by Post, Church, Turing and many others. They developed a nowadays accepted formal notion of computability. The resulting class of *computable functions* can be characterized by various logical calculi and schemes, but the most popular characterization among Computer Scientists is the one based on the machine model presented by Turing [Tur36], which nowadays is known as a Turing machine. In the following we will give an informal description of such a machine; for a formal description see e.g. [BDG88].

1.1 Turing Machines

A Turing machine is a device, resembling (or rather the other way around, that is resembled by) what we call nowadays a computer. The device possesses an input tape, output tape and work tape. On the input tape a finite word, called the input, is written. The characters written on the input tape (or any other tape) are taken from a finite alphabet denoted by Σ. Usually the alphabet consists of 0 and 1. A tape is a two way infinite band, divided into cells that can contain characters from the alphabet or can be blank. The work tape can be seen as the analogy of memory in a computer and the output tape, on which the output of the computation is written, can be seen as representing the screen. The machine is able to read characters from the input and work tape and write characters on the work and output tape. In order to do this it has pointers, called heads, that can move along the tapes. Finally, the device possesses, to be able to compute something, a finite *fixed* program, that prescribes, depending on the contents of the cells currently scanned by the two heads, to write a character in a cell, move a head left or right, or stop the computation in an accepting or rejecting state. By convention a Turing machine always starts its computation with the head of the input tape on the *leftmost* cell containing a non-blank character. Furthermore the contents of the cells of the work tape and output tape contain only blanks.

Since the only relevant part of a Turing machine is its program, we identify Turing machines with programs. Since a program is nothing more than a piece of text, we identify programs with strings, usually consisting of 0's and 1's. Because a string of 0's and 1's can be seen as a natural number (written in binary), we can interpret every natural number n as a Turing machine program: Either n codes correctly a Turing machine program or it codes garbage, in which case we say that n represents the Turing machine that always halts. Thus we have a list (the natural numbers) of all the Turing machines.

A Turing machine computes a function in the following way: On some input x it either ends its computation (in an accepting or rejecting state) with y on the output tape, or the computation does not stop at all. We say that the Turing machine computes the function whose value is y, in the case that the computation halts in an accepting state and is undefined in the other cases.

The functions that can be computed by a Turing machine are called *effectively computable*. We may distinguish between partial functions, i.e. functions that are sometimes undefined and total functions, functions that are always defined. This distinction for computable functions basically translates to Turing machines that do not stop on some inputs, versus Turing machines that *always*, i.e. on all the possible inputs, stop. We call functions that are computed by Turing machines that always stop, *total computable*.

Since characteristic functions represent sets[1], we are able to talk about sets as well. Since the emphasis in this thesis lies on sets we will focus on them. Every Turing machine can be seen as one that describes (or recognizes) a set $A \subseteq I\!N$ as follows. A string x is in A if the Turing machine ends its computation with x on the input tape in an accepting state, otherwise, i.e. the machine didn't stop at all or stopped in a rejecting state, x is not in A. Again we can make the distinction between Turing machines that always stop and Turing machines that sometimes don't stop. Sets that are described by Turing machines that always stop are called *recursive*. The class of recursive sets is denoted by REC. Sets recognized by arbitrary Turing machines, are called *recursively enumerable* and are denoted as RE. It is not hard to see that the following inclusion holds: $REC \subseteq RE$.

Before we continue, we have to mention that although a Turing machine is a fixed program we can construct a Universal Turing machine, of course by means of a program, which is able to *simulate* the

[1]Let A be any set, the characteristic function belonging to A, denoted χ_A, is defined as: $\chi_A(x) = 1$ if $x \in A$ and $\chi_A(x) = 0$ if $x \notin A$.

code of *any* Turing machine program. This Universal Turing machine resembles our computer even better.

1.2 The Halting Problem & Oracle Machines

Turing continued to prove that this inclusion between REC and RE is strict ($REC \subset RE$), by showing the existence of a set H that is recursively enumerable, but not recursive. This set H is based on the following problem: is it possible to determine whether a certain Turing machine program stops on a given input or whether it loops forever? A first attempt to solve this problem, is simply to simulate the program on the given input and stop if the simulation stops. But what if the program does not stop? Then of course our simulation does not stop and the just described attempt is recursively enumerable instead of recursive. It turns out as Turing proved, that this is the best we can do! In some cases there is no effective procedure to determine whether a given Turing machine stops or loops forever. The set H being the set of all pairs $<p, x>$, where p is a program and x is an input, such that Turing machine with program p stops on input x, is called the Halting set and is an example of a set for which no effective procedure exists which always stops and outputs whether a certain string is a member of it or not. Such sets are called *undecidable*

The proof of the undecidability is not hard. Suppose H is decidable. This means that we have an effective procedure that always halts and is able to tell from every program p and input x whether p halts on x or not. Recall that the natural numbers form a list of *all* the Turing machines. Consider now the Turing machine that on input n, with the use of the assumed effective procedure for the halting set, computes whether *program* n stops on input n and then stops if and only if program n does not stop. If the effective procedure for the halting set exists, then it is not hard to see that the above Turing machine exists. But if it exists then it has a program, say it has natural number 666. What happens if we run program 666 on input 666? Since 666 is the program that performs the above procedure, it simulates program

666 on input 666, and stops if and only if this program does not stop. This means that program 666 stops on input 666 only if it does not stop! Surely something is wrong: the effective procedure for H does not exists!

The last main contribution of Turing to computability theory, and probably the most important with respect to this thesis, is the notion of "oracle Turing machine". An oracle Turing machine is an ordinary Turing machine equipped with two extra tapes: a *query tape* and an *answer tape*. Furthermore it disposes of an *oracle* represented as a set. An oracle Turing machine operates in the same way as a Turing machine does, however it has the following extra possibility. It can write a string q, called query, on the oracle tape. Next it can perform a query, by entering a query state, resulting the oracle, say A, to write a 1 on the answer tape if $q \in A$ and a 0 if $q \notin A$.

The notion of oracle Turing machine makes precise the intuitive notion of B is 'at most as hard to compute as' A, for sets A and B as follows. Let A and B be such that B can be computed recursively by an oracle Turing machine with A as oracle. By computed recursively we mean that the oracle Turing machine stops on all its inputs. The intuitive notion of B 'is at most as hard to compute as' A is captured since a (recursive) procedure for A, yields a (recursive) procedure for B. To see this simply convert the oracle Turing machine into an ordinary Turing machine, by replacing the oracle queries, by (recursive) computations for A. On the other hand if A is not recursive, then B is computed recursively, with computations of A for free. Thus the complexity of B is, modulo some effective procedure, the same as A.

1.3 Reductions

The above discussion yields a (quasi) ordering between sets: $B \leq A$ iff B can be recursively computed by an oracle Turing machine with A as oracle. Such an oracle Turing machine is called a *Turing reduction* and it is said to *Turing reduce* B to A.

The notion of Turing machine together with the concept of a reduction turned out, to be a powerful tool in the study of computability. In fact Hilberts problem was first solved in 1970 when Matijasevič [Mat70] completed the design of a reduction from the Halting set to deciding solvability of Diophantine equations. The reduction had been under construction for more than two decades. This reduction provides a negative answer to Hilberts problem, for proving $H \leq D$ (the set of solvable diophantine equations) shows that D cannot be decidable unless H is, and for the latter Turing showed that it is not.

Reductions are easily seen to be transitive ($A \leq B \leq C \Rightarrow A \leq C$) and reflexive ($A \leq A$). Sets that also have the property that $A \leq B$ and $B \leq A$ thus form an equivalence class and are called a *degree*. The class REC stands out as the unique *minimal* degree under this ordering, since the recursive sets can be computed without any oracle. On the other hand it turned out that the Halting set H is maximal among the sets in RE. This means that for *every* set A in RE it is true that $A \leq H$. Sets that have this property are called *complete*.

1.4 Post's Problem and Program

Inspired by the fact that any r.e. set observed in mathematics for which the question makes sense at all, either turned out to be recursive or to be complete like the Halting set, Emil Post raised the question whether there exist *incomplete* sets, i.e. r.e. sets that are neither complete nor recursive. As a possible approach he proposed the following program:

1. Give structural properties of sets that prevent them from being complete or recursive.

2. Prove that recursively enumerable sets with these properties exist.

His first attempt to giving such structural properties gave birth to the notions of *simplicity* and *immunity* [Pos44]. An infinite set A is immune if it has no infinite subset that is recursively enumerable. Thus immune sets are not only not computable, but even all their infinite subsets aren't. Simple sets are recursively enumerable sets, that have an immune complement. Thus simple sets are sets that have a thin complement from a computational point of view.

Simple sets unfortunately turn out to be complete and therefore do not give a solution to Post's original problem. On the other hand they do form a solution to a variant of the problem, for simple sets are not complete with respect to a much stronger reduction, the many-one reduction. A many-one reduction, is an oracle Turing machine with *restricted* access to the oracle. The restriction is, that the machine can only access the oracle once. In order to give a solution to the problem with respect to many-one reductions, Post showed in [Pos44] that the notion of simple sets is not empty, by constructing a simple set.

Next Post tried to solve his problem for stronger forms of reducibility, a path that would eventually lead to success. He introduced stronger forms of simplicity: *hypersimple* sets, sets that have an even thinner complement than simple sets. To go into the details of the definition of hypersimple sets (See [Odi89, Soa87]) is beyond the scope of this introduction. It suffices to state that they provided correct solutions for Post's problem for an intermediate reduction, called truth-table reduction. A truth-table reduction is again an oracle Turing machine with restricted access to the oracle. The restriction however is not the number of strings that it is allowed to query, but that it is required to write down a list of all the strings that it is going to query, before it queries the first string. The difference from a Turing reduction, an oracle Turing machine with no restrictions, is that a Turing reduction can compute the strings it is going to query from answers to previous queries, whereas a truth-table reduction is not allowed to do this. Post [Pos44] proved that hypersimple sets exist and cannot be truth-table complete. Again hypersimple sets can be Turing complete as was shown in [Ars70], and thus do not form a solution to the original problem.

The first solution to Post's problem was not given along the lines of this program. The solution, 12 years after Post raised the problem, was given by Friedberg [Fri57] and independently by Muchnik [Muc56]. They used a complicated form of diagonalization, akin to the proof technique Turing used to show the undecidability of the Halting set, nowadays known by the name *finite injury priority method.* As a consequence of this result *Degree Theory* as a branch of Recursion Theory was created. It flourished by work of Sacks (showing among others the existence of incomparable degrees strictly below an arbitrary degree), Ladner, Jockusch and many others.

Nevertheless the quest for solving Post's problem via his proposed program continued on. As an obvious next step one should find an intermediate reducibility between Turing and truth-table, and refine the notion of hypersimple sets. However the problem is that there is no nice way to relax truth-table reductions, without falling to Turing reductions. A complementary technique was applied: strengthen Turing reductions instead. A somewhat strange reducibility notion called Q reduction emerged. The notion hypersimple is refined to the notion *hyperhypersimple* [Pos44]. This was the place where Post got stuck. He could neither prove that hyperhypersimple sets did exist nor could he prove that such sets were Turing incomplete. It turned out that the intuition of Post was again correct, in the sense that hyperhypersimple sets exist and cannot be Q complete [Sol74, GM74]. Nevertheless there exist Turing complete sets that are hyperhypersimple [Yat65].

So far it was established that hyperhypersimple sets are not Q complete. People started looking for notions that together with Turing completeness would imply Q completeness, and then coupling them with hyperhypersimple sets, would solve Post's problem. The notion that looked promising is semirecursiveness, introduced by Jockusch [Joc68]. A set A is *semirecursive* if there is a recursive way to determine, of *two* elements say x and y, which one is the 'most likely' of the two to be in A, meaning that there exists a recursive procedure, which on input $<x, y>$, outputs x or y and if one of the two strings is in A, it outputs one that is. Although it looks like semirecursiveness implies recursiveness, there exist nevertheless non-recursive semirecursive sets. In fact there exist semirecursive sets of arbitrary complexity [Joc68]. The notion looks promising for Marchenkov [Mar76] proved that every set A that is semirecursive and Turing complete is in fact Q complete. So semirecursive hyperhypersimple sets cannot be Turing complete! Unfortunately step two of Post's program fails sadly as Martin shows: No hyperhypersimple set can be semirecursive.

The notion of hyperhypersimplicity is too strong to go together with semirecursiveness. The idea is thus to weaken the notion to something that still is incompatible with Q completeness, but tolerates semirecursiveness. The notion that does the trick is called η-maximality [Ers71]. Finally Marchenkov [Mar76], proving that η-maximal sets cannot be Q complete, and Degtev [Deg73] showing that there exist an η-maximal, semirecursive and non-recursive set A, brought Post's program to a positive conclusion.

1.5 Complexity Theory

Complexity theory finds its origin in the fact that after the second world war mankind came face to face with real life versions of the machines Turing had introduced as a mathematical model. It turned out however that the idealized *borderline of computability* was way beyond what can be done in practice, due to lack of time, space and materials.

Researchers solving large scale optimization or other computational problems, by experience became aware of the important differences between problems which are effectively solvable and problems which are solvable in principle, but were found to be intractable for all practical purposes. It was in fact a researcher from the Operations Research area, Edmonds [Edm65], who presented the idea that tractability of a combinatorial problem should equate to having a polynomial time bounded algorithm for its solution. This idea introduces a class of sets (problems), for which it can be decided quickly, whether an element is in the set or not. This class is called polynomial time and will be denoted as P. Sets that belong to P, have the property that there exists a recursive Turing machine that describes the set, and furthermore on input x of length n, the Turing machine makes no more than $p(n)$ steps for $p(n)$, a polynomial.

It turns out that some of the problems one wants to solve in practice, indeed are captured by this class. On the other hand there are numerous problems that are *not known* to be in this class P, but for which a quick algorithm would be desirable. A large group of these problems can be characterized as problems, for which it may be hard to *find* a solution, but once a solution is found, a solution can be *checked* quickly (i.e. in polynomial time). As an example we mention here the Traveling Salesperson Problem (TSP). Given are a set of n cities, with roads between them. Each road has a certain length (in km). The problem to solve is for example: 'Is there a route along all the n cities, that has a length of 100 km?'. It may be hard to *find* indeed a route that has a length of 100 km, but if someone shows you a route and claims that its length is 100 km, it easy to check whether this is true: simply add up all the connecting roads on the route and verify whether this is 100 km.

The class of problems that can be characterized as "hard to find but easy to check" is called *NP*, which stands for non-deterministic polynomial time. For practical purposes it would be desirable to show that $P = NP$, but on the other hand it is widely believed that $P \neq NP$. The situation that arizes is somewhat akin to the situation in the beginning of this century. On one side we have the classes *REC* and *P* standing for effective computable and feasible computable, respectively, and on the other side we have *RE* and *NP*.

Because of the analogy between Complexity Theory and Recursion Theory, a similar line of research is followed in order to attack the *P* versus *NP* problem, leading to the notion of *polynomial time* reducibility. The problems that induced the definition of the class *NP* turn out to be all *complete* for this class with respect to polynomial time bounded many-one reductions [Coo71, Lev73]. The fact that almost all the known problems in *NP* are complete, yields that if a polynomial time algorithm is found for *one* of them, a polynomial time algorithm is found for *all* of them.

1.6 Structural Complexity Theory

Structural Complexity Theory, finds its origin in work done by Juris Hartmanis and his students performed in the late 1970-ies. Inspired by the fact in recursion theory that all problems that are complete for *RE* are all isomorphic, Berman and Hartmanis set out to examine the status of all the *NP* complete problems. It turned out that all the known problems were indeed polynomial time isomorphic, which led them to conjecture that that this was true for *all NP* complete problems. Having observed that this conjecture, if true would entail $P \neq NP$, they tried to disprove it, by constructing an *NP* complete set that would not be isomorphic to TSP, on the basis of its *structural* properties. This research can, as such be seen as a revival in Complexity Theory of Post's program. The structural problem that would prevent a *NP* complete set *A* from being polynomial time isomorphic to TSP, is the density of the set. The density of a set is a function that bounds from above the number of elements in the set of length at most n. A set is called sparse if the density function is a polynomial. The quest for a sparse, *NP* complete set was initiated, which ended in failure, when Mahaney [Mah82], proved that such sets could not exists, unless $P = NP$.

By varying the notion of polynomial time reduction, as in the recursion theoretical setting, from many-one to Turing reductions, gives rise to research on the Turing complete sets in *NP*, and polynomial time degree theory inside *NP*. Due to the failure to prove *P* different from *NP*, all the results along these lines can only be stated under the assumption that $P \neq NP$. The direct analogs of Post's problem and some of the degree structures in *RE* can be shown to exist in *NP* [Lad75], under this assumption. However these results may be meaningless if *P* turns out to be equal to *NP*. Furthermore even under the assumption $P \neq NP$, little of the structure of complete sets has been settled.

This led Berman to examine other complexity classes, in the 'neighborhood' of *NP*. The variation from polynomial to *exponential* functions in the definition of *P* and *NP*, yields the classes *E* and *NE* standing for *exponential time* and *non-deterministic exponential time*. These classes are the exponential brothers of *P* and *NP* and contain sets for which elements of length n can be decided within 2^n steps, or for which a solution can be checked in 2^n steps respectively. Berman [Ber77] was able to prove for *E* and *NE*, that sparse sets cannot be many-one complete.

Attention shifted towards these intractable, but subrecursive classes, in order to get a better understanding of the polynomial time reducibility and in order to develop techniques, that could eventually be used down in *NP*. Unfortunately even for classes as *E* and *NE* many hard problems remain open, for example it is not known whether they can possess sparse Turing complete sets.

1.7 This Thesis

The main goal is to get a better understanding to the problems mentioned last in the previous section: What is the structure of *E* and

NE and can these classes have sparse complete sets? All the theorems that are not new work have a reference to where they came from. The research in this thesis takes the work of Berman [Ber77] and later Watanabe [Wat87a] as a starting point. The first goal is to get a better understanding of the polynomial time reductions and the completeness notions induced by them. We prove in Chapter 3 that for almost all reduction types, the completeness notions differ on *NE*. This extends the work of Berman and Watanabe to non-deterministic complexity classes. One surprising exception is the one-truth-table reduction, for which the completeness notion, even on *NE*, turns out to be the same as the completeness notion induced by the the many-one reduction. The new work described in this chapter is taken from [BHT91] and [BST91].

In Chapter 4 we turn our attention to structural properties and examine the consequences of *NP*, *E* and *NE* having subexponential dense complete sets. We extend the work of Karp and Lipton [KL80] to other than sparse, subexponential densities. We show that there is strong evidence that complete sets, w.r.t. Turing reductions cannot be subexponential dense in the sense that this would imply unlikely and widely believed collapses of complexity classes. The work in this chapter is taken from [BH92].

In order to get a better insight of why it is hard to prove that problems like TSP or sets in *E* do *not* reduce to sparse sets, we examine in Chapter 5 the sets that *do* reduce to sparse sets. We solve two open problems from [Ko89]. This work is taken from [BLS92].

In Chapter 6 we follow the lines of Post's problem more closely and try to examine structural properties of complete sets as was done in the recursion theoretical setting. We examine the polynomial time variants of immunity and splittings. Furthermore we inspect the robustness of complete sets: How little can we demolish a complete set, by taking out or adding elements, in order to render it incomplete? It turns out that sometimes a couple of elements can render a set incomplete. On the other hand we show that if we restrict the complexity of the elements to be taken out, to be polynomial time computable, *all* sets for various kinds of completeness notions remain complete after removal of these elements. It seems useful to pursue this approach since we show that extending these robustness results to Turing complete sets, yields a solution to the original problem: there exist no sparse complete sets for

EXP. The results are taken from [BHT93].

In the last chapter, we explore other structural properties of sets. We examine the notion selfreducibility. We get strong structural characterizations of polynomial time computable sets in terms of selfreducibility and p-selectivity, the latter being a direct translation of the semirecursiveness notion, introduced to solve Post's problem. We show that P can be characterized as those sets that are selfreducible and p-selective. By closely examining this statement we see that a generalization of this statement, saying that P can be characterized as those sets that are selfreducible and Turing reducible to a p-selective set yield a solution to the original problem. Unfortunately we cut off this approach by showing that it is unlikely that such a characterization is true, by constructing, under some hypotheses, a counter example. Furthermore we also explore which sets in NP are selfreducible, and show that the class of p-selective sets is closed under positive Turing reductions. The results are taken from [BvHT] and [BTvEB93].

Chapter 2

Definitions

2.1 Machines and languages

We use a, now a days standard, notation as can be found in [BDG88, BDG90]. Let $\Sigma = \{0,1\}$. Strings are elements of Σ^*, and are denoted by small letters $x, y, u, v, \ldots$. For any string x the length of a string is denoted by $|x|$. Languages are subsets of Σ^*, and are denoted by capital letters $A, B, C, S, \ldots$. The complement of a language A in Σ^*, $\Sigma^* - A$ will be denoted by $\overline{A}$. For any set S the cardinality of S is denoted by $\|S\|$. We fix a pairing function $\lambda xy.{<}x,y{>}$ computable in polynomial time from $\Sigma^* \times \Sigma^*$ to Σ^*. Without loss of generality, we assume for all y, y' and x with $|y|, |y'| \leq |x|$ that $|{<}y,x{>}| = |{<}y',x{>}|$. The characteristic function for a set A, denoted $\chi_A(x)$ or $A(x)$, is a function from $\Sigma^* \to \{0,1\}$: $\chi_A(x) = 1$ ($A(x) = 1$) if $x \in A$ and $\chi_A(x) = 0$ ($A(x) = 0$) if $x \notin A$. We interpret string $x \in \Sigma^*$ as natural number $1x$.

We assume that the reader is familiar with the standard Turing machine model. Let M be a nondeterministic polynomial time Turing machine. $\textsc{accept}(M(x))$ is a function from $\Sigma^* \to I\!N$ and denotes the number of accepting computations of M on input x. Let M be a (nondeterministic) Turing machine, we will write $M(x) = 1$ if M on input x halts in an accepting state and we will write $M(x) = 0$ if M halts on input x in a rejecting state.

Whenever it is obvious that a universal recognizing or transducing machine exists for a class of languages (i.e. the class is recursively presentable), we will assume an enumeration of the acceptors and/or transducers and denote this enumeration by $M_1, M_2, \ldots$. For a Turing machine $M, L(M)$ denotes the set of strings accepted by M.

2.2 Time classes

Let $\mathrm{DTIME}\,(f(n))$ be the class of sets such that $A \in \mathrm{DTIME}\,(f(n))$ iff there exists a deterministic Turing machine M whose running time is bounded by $f(n)$ as $n \to \infty$ (n is the length of the input) and $A = L(M)$. Let $\mathrm{NTIME}\,(f(n))$ be the corresponding nondeterministic class. We define the following classes:

$$\begin{aligned}
\mathit{EE} &= \bigcup_{c=1}^{\infty} \mathrm{DTIME}\left(2^{2^{cn}}\right)\\
\mathit{NEE} &= \bigcup_{c=1}^{\infty} \mathrm{NTIME}\left(2^{2^{cn}}\right)\\
\mathit{NEXP} &= \bigcup_{i=1}^{\infty} \mathrm{NTIME}\left(2^{n^i}\right)\\
\mathit{EXP} &= \bigcup_{i=1}^{\infty} \mathrm{DTIME}\left(2^{n^i}\right)\\
\mathit{NE} &= \bigcup_{c=1}^{\infty} \mathrm{NTIME}\left(2^{cn}\right)\\
E &= \bigcup_{c=1}^{\infty} \mathrm{DTIME}\left(2^{cn}\right)
\end{aligned}$$

2.3 Kolmogorov complexity

The *Kolmogorov complexity* of a string x, $K(x)$, is the size of the smallest index of a Turing machine that generates x and halts. A *Kolmogorov random* string is a string x such that $K(x) \geq |x|$. For a more detailed description see for example [LV90].

2.4 Oracle Turing Machines

An oracle Turing machine is a standard multi-tape Turing machine with two extra tapes :

1. a write only tape called the QUERY-tape.

2. a read only tape called the ANSWER-tape.

These tapes will be called the oracle tapes. Furthermore we add an extra state: the QUERY-state. We use the following convention for access to the oracle tapes: M is allowed to write on the QUERY-tape a string q, called *query*, then at some point it decides to go into the QUERY-state. Subsequently the QUERY-tape is cleared[1], and depending on the oracle, something is written on the ANSWER-tape. Now M is allowed to read the ANSWER-tape, until a next QUERY-state is reached. A Turing machine equipped with the above described extra tapes and state is called an *oracle Turing machine.* In the above discussion it was not clear what the role of the oracle is. An oracle is just a set, say A. When an oracle Turing machine M writes a string q on the QUERY-tape and enters the QUERY-state, the oracle writes down – in one step – on the ANSWER-tape the value of the characteristic function of A on q, that is $\chi_A(q)$. Informally, M asks oracle A whether y is a member of A and finds the answer on its ANSWER-tape. We note here that the role of the oracle can be more complex in the sense that it could write down not only one character but a whole string of characters. Examples of this can be found in [ABJ91, FHOS93]. Let A be a set and M be an oracle Turing machine We say that M accepts x *relative to* A if M has an accepting computation on input x, with A as oracle. We say that $L(M, A)$ is the set of strings accepted by M relative to A. As usual, we can talk about polynomial time oracle machines and computations.

2.5 Adaptive and Non-Adaptive

Essentially there are two ways an oracle machine can compute it's queries:

[1]With clearing a tape we mean that after clearing, the only symbols on the tape are blanks.

1. adaptive: M is allowed to read the ANSWER-tape at *any* time during the computation and may compute the next query depending on the contents of the ANSWER-tape. In this case the queries are *dependent* on the oracle.

2. non-adaptive: M is not allowed to read the ANSWER-tape *before* it enters the last QUERY state. In this case the computation of all the queries depends solely on the input and the program, and is *independent* of the oracle.

Sometimes we want to to talk about the set of (possible) queries that M could ask on input x.

Definition 2.1 *Let M be an oracle Turing machine.*

- *$Q(M, x, A)$ is the set of all queries M wrote on its* QUERY*-tape with x as input and A as oracle.*
- *$Q(M, x) = \bigcup_{A \subseteq \Sigma^*} Q(M, x, A)$. This denotes the set of all possible queries M could ask.*

2.6 Reductions

In this section we want to formalize the notion of "A is at most as hard to compute as B", for sets A and B. One of the things that should follow from this is, that if A is at most as hard to compute as B and we know that B itself has a certain computational complexity, then A cannot have higher computational complexity. Eg. if B is in P, then it should follow that A is also in P. One way of capturing this notion is just by saying that A can be recognized with B as an oracle. More precisely: $A = L(M, B)$.

In order to get more grip on the oracle computation, we add restrictions to the oracle machine M. More formally a *restriction* is a 4 tuple: r = <N, COMP, ACCEPT-RESTR, QUERY-TYPE>. Where,

1. N is a function from $I\!N \times \Sigma^* \rightarrow I\!N$. This function depends on the index of M and the input. The function is the number of queries M is allowed to make during the computation. With number of

queries we mean the number of times that M entered the QUERY-state.

2. COMP can be adaptive or non-adaptive.

3. ACCEPT-RESTR is a function from $\Sigma^* \times \Sigma^* \rightarrow \mathcal{P}(\{0,1\})$[2]. This function depends on the INPUT-tape and the ANSWER-tape. The function describes when the Turing machine has to accept and when to reject.

4. QUERY-TYPE is a set of additional constraints on the type of queries M is allowed to make. Eg. all the queries should start with a 0 or should be smaller in length than the input.

We say that an oracle Turing machine M_i (we assume an effective enumeration of oracle Turing machines) *obeys* restriction r, if for all input strings x:

- N $= \emptyset$ or $M_i(x)$ does not make more than N(i, x) queries, and

- COMP $= \emptyset$ or $M_i(x)$ generates it's queries in a COMP (i.e. either adaptive or non-adaptive) fashion, and

- ACCEPT-RESTR $= \emptyset$ or if $M_i(x)$ halts then
  $M_i(x) \in$ ACCEPT-RESTR(x, y), for y the string on the ANSWER-tape when $M_i(x)$ halts, and

- QUERY-TYPE $= \emptyset$ or $M_i(x)$ only wrote down queries q that satisfy the constraints in QUERY-TYPE.

We say that M is an r-restricted oracle machine, if r is a restriction and M is an oracle Turing machine, that obeys restriction r.

We now are able to give a definition of the intuitive notion "A is at most as hard to compute as B". We will call this notion reduction.

Definition 2.2 *A r reduces to B ($A \leq_r^{REC} B$) iff there exists a recursive r-restricted oracle Turing machine M, such that $A = L(M, B)$.*

[2]$\mathcal{P}(\{0,1\})$ denotes the *power* set of $\{0,1\}$

In this thesis, we will only talk about polynomial time oracle machines. Note that this does not necessarily means that the ACCEPT-RESTR function is computable in polynomial time. This notion will be called polynomial time reduction. The notion of polynomial time reduction was first defined by Ladner, Lynch and Selman. [LLS75] In the following we will not redefine the existing notions of reducibility. We will capture them in a machine based framework. We think that the most natural way to think about a reduction is as an oracle Turing machine with several restrictions on the access it has to the oracle. The most general one, is the Turing reductions which has no restrictions at all. The definitions found in the literature are by no means uniform in this sense. Sometimes they define reductions as functions other times the machine based point of view is used.

The approach we take has also the advantage that it gives a taxanomy of the reductions in four natural groups. Several new reductions emerge from this taxonomy by varying the 4 different aspects of the reductions. Sometimes already existing reductions come out. For example *adaptive* conjunctive reductions are the same as non-adaptive conjunctive reductions, but it is probably not true that adaptive parity (or majority) reductions are the same as the non-adaptive couter parts.

We advice the reader not to consume the following list of reductions at ones, but to use it as a reference to consult after further reading.

Definition 2.3 *A reduces r to B in polynomial time ($A \leq_r^p B$) iff there exists an r-restricted polynomial time oracle machine M such that $A = L(M, B)$.*

We will now show that some of the standard reductions found in the literature are easily captured by our formalism. To start with the most general restriction:

1. T $= <\emptyset, \emptyset, \emptyset, \emptyset>$. ($\leq_T^p$)
   This restriction does not restrict the class of oracle machines. This reduction is called Turing reduction.

2. tt$= <\emptyset, \text{non-adaptive}, \emptyset, \emptyset>$. ($\leq_{tt}^p$)
   The oracle machines are restricted in the way they generate their queries. This reduction is called truth-table reduction.

3. btt = $<n_b, \text{non-adaptive}, \emptyset, \emptyset>$. ($\leq^p_{btt}$)
   $n_b(i,x) = i$.
   This reduction is called bounded truth-table reduction.

4. k-tt = $<n_k, \text{non-adaptive}, \emptyset, \emptyset>$. ($\leq^p_{k\text{-}tt}$)
   $n_k(i,x) = $ k, k a constant.
   This reduction is called k-truth-table reduction. Actually this defines a whole class of reductions one for every constant k.

5. k-T = $<n_k, \emptyset, \emptyset, \emptyset>$. ($\leq^p_{k\text{-}T}$)
   $n_k(i,x) = k, k$ a constant.
   This reduction is called k-Turing. The bounded Turing reduction ($\leq^p_{b\text{-}T}$) is defined by replacing n_k by $n_b(i,x) = i$.

6. ctt = $<\emptyset, \text{non-adaptive}, f_c, \emptyset>$. ($\leq^p_{ctt}$)
   $$f_c(x,y) = \begin{cases} \{1\} & \text{if } \forall i, y_i = 1. \quad (y = y_1 \ldots y_n) \\ \{0\} & \text{otherwise.} \end{cases}$$
   This reduction is called conjunctive truth-table reduction.

7. dtt = $<\emptyset, \text{non-adaptive}, f_d, \emptyset>$. ($\leq^p_{dtt}$)
   $$f_d(x,y) = \begin{cases} \{1\} & \text{if } \exists i, y_i = 1. \quad (y = y_1 \ldots y_n) \\ \{0\} & \text{otherwise.} \end{cases}$$
   This reduction is called disjunctive truth-table reduction.

8. bdtt = $<n_b, \text{non-adaptive}, f_d, \emptyset>$. ($\leq^p_{bdtt}$)
   $n_b(i,x) = i$.
   $$f_d(x,y) = \begin{cases} \{1\} & \text{if } \exists i, y_i = 1. \quad (y = y_1 \ldots y_n) \\ \{0\} & \text{otherwise.} \end{cases}$$
   This reduction is called bounded disjunctive truth-table reduction. The bounded conjunctive truth-table reduction ($\leq^p_{bctt}$) is defined similar: f_d is replaced by f_c, and n remains the same.

9. k-dtt = $<n_k, \text{non-adaptive}, f_d, \emptyset>$. ($\leq^p_{k\text{-}dtt}$)
   $n_k(i,x) = k$.
   $$f_d(x,y) = \begin{cases} \{1\} & \text{if } \exists i, y_i = 1. \quad (y = y_1 \ldots y_n) \\ \{0\} & \text{otherwise.} \end{cases}$$

This reduction is called k-disjunctive truth-table reduction. The k-conjunctive truth-table reduction ($\leq^p_{k\text{-}ctt}$) is again defined similar: f_d is replaced by f_c, and n_k remains the same.

10. $\oplus$ = $<\emptyset, \text{non-adaptive}, f_\oplus, \emptyset>$. ($\leq^p_\oplus$)
$$f_\oplus(x,y) = \begin{cases} \{1\} & \text{if } \sum_i^n y_i = 1 \text{ (mod 2)}. \quad (y = y_1 \dots y_n) \\ \{0\} & \text{otherwise.} \end{cases}$$
This reduction is called the parity reduction.

11. m $= <n_m, \emptyset, f_m, \emptyset>$. ($\leq^p_m$)
$n_m(i,x) = 1$.
$$f_m(x,y) = \begin{cases} \{y\} & \text{if } y = 0 \text{ or } y = 1. \\ \emptyset & \text{otherwise.} \end{cases}$$
This reduction is called a many-one reduction. Sometimes it will be more elegant to use the following equivalent definition: $A \leq^p_m B$ iff there exists a total polynomial time computable function f such that $x \in A$ iff $f(x) \in B$. Obviously f can be constructed from an oracle machine that obeys the restriction m and vice versa.

12. $\hat{m} = <n_{\hat{m}}, \emptyset, f_{\hat{m}}, \emptyset>$. ($\leq^p_{\hat{m}}$)
$n_{\hat{m}}(i,x) = 1$.
$$f_{\hat{m}}(x,y) = \begin{cases} \{y\} & \text{if } y = 0 \text{ or } y = 1. \\ \{0,1\} & \text{otherwise.} \end{cases}$$
This reduction will be called extended many-one.

13. m,li $= <n_m, \emptyset, f_m, \text{LI}>$ ($\leq^p_{m,li}$)
n_m and f_m as above.
LI $= \forall y \in Q(M_i, x) : |y| > |x|$.
This constraint says that the queries have to be bigger (in length) than the input.

This reduction is called many-one length increasing. As in the case of the many-one reduction we sometimes use the equivalent functional definition in terms of total polynomial time computable functions that are length increasing.

14. m,1-1 = $<n_m, \emptyset, f_m,$ ONE-ONE$>$ ($\leq^p_{m,1\text{-}1}$)
    n_m and f_m as above.
    ONE-ONE = $\forall y \in Q(M_i, x) : \forall x' < x : y \notin Q(M_i, x')$. This constraint says that each query is asked once. Clearly this implies injectivity. This reduction is called many-one and one to one reduction. We will use sometimes the existence of a total polynomial time computable function that is one-one.

15. m,1-1,li =$<n_m, \emptyset, f_m,$ ONE-ONE-LI$>$ ($\leq^p_{m,1\text{-}1,li}$)
    n_m and f_m as above.
    ONE-ONE-LI = ONE-ONE and LI. This means that both the constraints (ONE-ONE and LI) have to be satisfied in order to satisfy ONE-ONE-LI. This reduction is called many-one, length increasing and one to one. Again the functional equivalent way is sometimes chosen: there exists a total polynomial time computable function that is one-one and length increasing.

16. m,1-1,eh =$<n_m, \emptyset, f_m,$ ONE-ONE-EH$>$ ($\leq^p_{m,1\text{-}1,eh}$)
    n_m and f_m as above.
    EH = $\forall y \in Q(M_i, x) : 2^{|y|} > |x|$.
    This constraint says that the queries do not decrease more than exponential in length. ONE-ONE-EH = ONE-ONE and EH. This reduction is called many-one, one to one and exponentially honest. The same comment applies here: the functional variant must be exponential honest, i.e. not decrease more than an exponential in the length of the argument.

17. pos = $<\emptyset, \emptyset, f_{pos}, \emptyset>$ ($\leq^p_{pos}$)
    Let *POS* be the class of all positive boolean formulas. These are formulas, that can be represented using only disjunctions and conjunctions as connectives. For x a boolean variable, $x := 1(0)$ means $x := \top(-)$. $\phi = 1(0)$ if it evaluates to true (false).
    $f_{pos}(x, y) = \{\phi(x_1 := y1, \ldots, x_i := y_i)\}$ $(y = y_1 \ldots, y_i)$.
    For $\phi \in POS$.
    This reduction is called positive Turing reduction. The positive truth-table, positive bounded-truth-table and the positive k-truth-table reductions are defined as truth-table, bounded truth-table or k-truth-table reduction with f_{pos} as ACCEPT-RESTR.

Another way of looking at this reduction is as follows: M_i is a positive Turing reduction if for all oracles A and B it holds that if $A \subseteq B$ then $L(M, A) \subseteq L(M, B)$.

Chapter 3

Complete Sets

3.1 Getting started

Reductions, first introduced by Turing, give rise to equivalence relations on sets, such that the equivalence classes (sets which are reducible to each other) are partially ordered. Further refinements on reductions have led to the introduction of the *resource bounded* reduction. The reductions are resource bounded in the sense that the amount of space and/or time that is needed to perform the reduction, is restricted. This notion is particularly useful for, but not restricted to, ordering resource bounded (sub recursive) sets. The sets are resource bounded in the sense that they can be recognized by time or space bounded Turing machines. These resource bounded sets are grouped together and form a complexity class, for example P, NP, $PSPACE$, E, EXP, NE or $NEXP$.

The following two notions form the main ingredients for this thesis: What kind of ordering, is induced by reduction r, on sets in complexity class C?

It turns out, that this point of view has been very fruitful in the past. Results along these line include the notion of NP -completeness [Coo71] and the proof that nondeterministic logspace is closed under complementation.

One of the first questions that comes to mind is: what is the smallest element (set) under reduction r in complexity class $\mathcal{C}$ and what is, if any, the biggest? The first question can be answered quite straightforwardly: It is any set in the class P. This because a polynomial time bounded oracle machine that *never* accesses the oracle tapes, reduces the language that it accepts to *any* set in polynomial time, and since it accepts the language in polynomial time it is in P[1].

This chapter will deal with the other extreme: the biggest element in a certain class. Let us define this more precisely:

Definition 3.1 *Let $\mathcal{C}$ be a complexity class, A a set and $\leq_r^p$ a reduction of type r.*

1. *A is called r -hard for complexity class $\mathcal{C}$ if for all $B \in \mathcal{C}, B \leq_r^p A$.*
2. *A is called r -complete for complexity class $\mathcal{C}$ if it is both r -hard for $\mathcal{C}$ and $A \in \mathcal{C}$.*

Describing, in general, the necessary and sufficient conditions for complexity classes to contain complete problems, is still an open problem, but the complexity classes we will consider in this chapter do contain many-one complete sets.

For the following complexity classes enumerations of Turing machines exist: $P, NP, PSPACE, E, EXP, NE$ and $NEXP$. We will denote the i^{th} machine in such an enumeration by M_i. We will assume the following convention for the running time of these machines:

- P: M_i runs in deterministic time n^i.
- NP: M_i runs in non-deterministic time n^i.
- $PSPACE$: M_i uses no more than n^i space.
- E: M_i runs in deterministic time 2^{in}.
- EXP: M_i runs in deterministic time 2^{n^i}.
- NE: M_i runs in non-deterministic time 2^{in}.
- $NEXP$: M_i runs in non-deterministic time 2^{n^i}.

[1]There is one exception to this, namely the many-one reduction. For this the statement should read: any set in P can be many-one reduced to any other set except $\emptyset$ and Σ^*. The alternative reduction $\leq_{\hat{m}}^p$ solves this problem.

The standard way to show that there are complete sets is the following. Consider sets of the form:

$$K = \{<e, x, \mathrm{pad}(<e, x>)> \mid M_e \text{ accepts } x\ \}$$

The function pad, computable in x, makes sure that the set K is computable *within* the appropriate resource bound. For example for NP the function pad is $0^{|x|^e}$, this to assure that K is in NP. K is complete for NP because the reduction from a set, say A, to K works as follows. A is in NP and this fact is witnessed by machine M_j. The reduction on input x simply queries whether $<j, x, 0^{|x|^j}>$ is in K and accepts x iff this is the case.

Since the set K is many-one complete for the above mentioned complexity classes it follows immediately that K is complete for any of the defined reductions. In this chapter we will show that on E, EXP, NE and $NEXP$ for almost all reduction types the resulting completeness notions differ. This will be done by constructing sets that are complete w.r.t. one kind of reduction and are not complete w.r.t. an other kind of reduction.

Unfortunately it is not known whether these differences are true for NP or $PSPACE$. The difficulty lies in the fact that if one could show these differences, then one proved in fact that $P \neq NP$ or $P \neq PSPACE$. This because on P all the completeness notions are the same.

In Section 3.2 we will explore whether the notions $\leq^p_m$ and $\leq^p_{1-tt}$ can be separated. In Section 3.3 we will examine the reductions that query a constant number of queries and in Section 3.4 the adaptive versus non-adaptive reductions will be separated. The new work in this chapter is taken form [BHT91] and citeBuhrmanSpaanTorenvliet91.

3.2 One Query

In this section we will examine the reductions that obey restrictions with $\mathrm{N} = 1$. It is clear that adaptive and non-adaptive computations are the same if only one query is involved, so $\leq^p_{1-tt}$ and $\leq^p_{1T}$ are one and the same reduction. Furthermore for sets not Σ^* or $\emptyset$, $\leq^p_m$ and $\leq^p_{\hat{m}}$ are equal and since neither Σ^* nor $\emptyset$ can be complete for exponential time classes, the completeness notions w.r.t. these reductions coincide.

The only two candidates that remain are $\leq_m^p$ and $\leq_{1-tt}^p$. It is not hard to build a set $A \in EXP$ such that $A \not\leq_m^p \overline{A}$, and since obviously $A \leq_{1-tt}^p \overline{A}$, it follows that these reductions differ on *EXP*. This however does not imply that the completeness notions w.r.t. these reductions differ, and surprisingly we will see that these completeness notions are in fact the same.

We first will show that for *E* and *EXP* the completeness notions for $\leq_m^p$ and $\leq_{1-tt}^p$ coincide, a result which can be found in [HKR]. Then we will show how to extend this proof in order to obtain the analogous result for *NE* and *NEXP*.

3.2.1 Completeness for EXP

In order to show that the $\leq_m^p$ and $\leq_{1-tt}^p$ -completeness notions coincide on *EXP*, we will show that for every $\leq_{1-tt}^p$ -complete set T and every set $A \in EXP$, $A \leq_m^p T$. Standard padding techniques will then suffice to get the result for E, since every set A, that is $\leq_r^p$ -complete for E is also $\leq_r^p$ -complete for *EXP*. For let B be any set in *EXP*, say in DTIME $\left(2^{n^i}\right)$. Define $B' = \{<x, 0^{n^i}> \mid n = |x| \text{ and } x \in B\}$. B and B' are many-one equivalent and since B' is in E, it follows that $B' \leq_r^p A$, as was needed. B' is called the *padded version* of B.

Let $M_1, M_2, \ldots$ be an enumeration of $\leq_{1-tt}^p$ -reductions.

Theorem 3.2 ([HKR]) *Let A be any set in EXP. If T is* $\leq_{1-tt}^p$ *-complete for EXP then* $A \leq_m^p T$.

Proof: We will construct a set D in *EXP* that will witness the fact that $A \leq_m^p T$ in the following sense. D will be in *EXP* and this will ensure that $D \leq_{1-tt}^p T$, via some one-truth-table reduction, say M_j. It will turn out that the construction of D ensures that M_j on inputs of the form $<j, x>$ will be a $\leq_m^p$ -reduction from A to T.

If we take a close look at the possible computation trees that a $\leq_{1-tt}^p$ -reduction, say $M_{one}(x)$, induces we get the following 4 possibilities:

1. $M_{one}(x)$ writes z on the QUERY-tape and accepts iff a 0 is written on the ANSWER-tape.

2. $M_{one}(x)$ writes z on the QUERY-tape and accepts iff a 1 is written on the ANSWER-tape.

3. $M_{one}(x)$ rejects.

4. $M_{one}(x)$ accepts.

Consider the following algorithm that defines the set D:

```
input $<i, x>$
Simulate $M_i(<i, x>)$. $M_i$ will end up in one of the 4 cases.
    If in case 3, accept.
    If in case 4, reject.
    If in case 2, accept iff $x \in A$.
    If in case 1, accept iff $x \notin A$.
end
```

Since A and $\overline{A}$ are both in *EXP*, it is not hard to see that D is also in *EXP*. Since T happens to be $\leq^p_{1-tt}$ -complete for *EXP*, $D \leq^p_{1-tt} T$, say via M_j. Although somewhat magical it will be the case, that $M_j(<j, x>)$ performs a many-one reduction from A to T. Let z be the query written on the QUERY-tape by M_j on input $<j, x>$:

- case 3 and 4 cannot occur.

- case 2: $x \in A \Leftrightarrow <j, x> \in D \Leftrightarrow M_j(<j, x>)$ accepts $\Leftrightarrow z \in T$.

- case 1: $x \in A \Leftrightarrow <j, x> \notin D \Leftrightarrow M_j(<j, x>)$ rejects $\Leftrightarrow z \in T$.

□

3.2.2 Completeness for NEXP

The above proof relies heavily on the fact that *EXP* is closed under complementation. In the last step of the algorithm that defines D it is required to accept $<i, x>$ iff $x \notin A$. Since it is not known whether *NEXP* is closed under complementation this line will not necessarily be computable in nondeterministic exponential time and hence it cannot be guaranteed that the resulting set D is in *NEXP*.

The idea to get around this problem is to first prove for sets $A \in$ *NEXP* $\cap$ *co-NEXP* that are $\leq^p_{1-tt}$ -reducible to a complete set T, that in fact $A \leq^p_m T$. This can be done using the proof technique for the *EXP* case. Once this is done we are able to reduce the general case to this special case.

Lemma 3.3 *Let T be a $\leq^p_{1-tt}$ -complete set for NEXP. For every set $A \in$ NEXP $\cap$ co-NEXP, $A \leq^p_m T$.*

Proof: As above, we will construct a set $D \in$ *NEXP* with the property that $A \leq^p_{1-tt} D \leq^p_{1-tt} T$ and the $\leq^p_m$ -reduction from A to T can be computed from the $\leq^p_{1-tt}$ -reduction from D to T. Observe that the set D in the proof of Theorem 3.2 is in *NEXP* if A is in *NEXP* $\cap$ *co-NEXP*. So let $D \leq^p_{1-tt} T$ via M_j. Again M_j reduces A many-one to T on the pairs $<j, x>$. $\Box$

Now for all sets in *NEXP* if a set is 1-truth-table reducible to a complete set T via say machine M_j, then there are strings that are accepted if the query is in T. Those strings are already many-one reducible to T. The other strings (i.e. the strings that get accepted by a query in the complement of T) form a set that is in *NEXP* $\cap$ *co-NEXP* and by Lemma 3.3 they are many-one reducible to T via some other reduction. We state:

Theorem 3.4 *Every $\leq^p_{1-tt}$ -complete set for NEXP is also $\leq^p_m$ -complete.*

Proof: Let A be a set in *NEXP*, T a 1-truth-table complete set in *NEXP* and let M_j witness the reduction from A to T. On any input M_j can end up in one of the following four situations:

1. M_j queries z and accepts iff $z \in T$
2. M_j queries z and accepts iff $z \notin T$
3. M_j accepts
4. M_j rejects

We now split set A in two subsets A_1 and A_2.

$$A_1 = \{x \mid x \in A \text{ and machine } M_j \text{ is not in case } 2\}$$
$$A_2 = \{x \mid x \in A \text{ and machine } M_j \text{ is in case } 2\}$$

Claim 3.5 *A_2 is in NEXP $\cap$ co-NEXP.*

Proof: We need to show that there is a *NEXP* predicate for A_2 and for the complement of A_2.

$$x \in A_2 \text{ iff machine } M_j \text{ in case 2 and } x \in A$$
$$x \notin A_2 \text{ iff machine } M_j \text{ not in case 2 or } z \in T$$

It is clear that both predicates are *NEXP*. □

Now we can construct the many-one reduction from A to T: Simulate machine M_j on input x. If M_j is in case 1 then output z. If M_j in case 2 then x is in A iff x is in A_2. Since A_2 is in *NEXP* $\cap$ *co-NEXP* there is by Lemma 3.3 a many-one reduction from A_2 to T say g. Now output $g(x)$. If M_j is in case 3 output a fixed element $t_0 \in T$ and if M_j is in case 4 output a fixed element $t_1 \notin T$. The entire construction can be carried out in polynomial time. □

The construction can be generalized to a recursion theoretic setting. We relax the time bounds and end up with recursive reductions. We now have the following equivalent reductions $\leq_m^{REC}$ for a many-one reduction and $\leq_{1-tt}^{REC}$ for a 1-truth-table reduction in exactly the same way as the above theorem was proven we can prove the following:

Corollary 3.6 *let Σ_k be the k^{th} level of the arithmetic hierarchy as defined in [Soa87]. For all k if A is $\leq_{1-tt}^{REC}$ -complete for Σ_k then A is $\leq_m^{REC}$ -complete for Σ_k.*

It would be interesting to prove the same result for the class *NP*. The problem is that the technique used in Lemma 3.3 is not applicable for sets in *NP*. Under the strong assumption that $P = NP \cap co\text{-}NP$ however, we can prove it.

Corollary 3.7 *If $P = NP \cap co\text{-}NP$ then every $\leq_{1-tt}^{p}$ -complete set for NP is $\leq_m^p$ -complete*

3.3 A constant number of Queries

In this section we will consider not only reductions that query one string but reductions that are allowed to query a constant number of strings. We will examine the resulting completeness notions on E, EXP, NE and $NEXP$.

3.3.1 K-Truth-Table

In 1987, Watanabe [Wat87a] and [Wat87b] building upon earlier work of Berman [Ber76], proved that $(k+1)$-truth-table completeness on E differs from k-truth-table completeness. In this section we will prove that this is also true for NE and $NEXP$. The proof technique used to show this will also work on E and thus will prove the result for E as well.

As we saw in the previous section many-one completeness is the same as 1-truth-table completeness. In this section we will see that one extra query suffices to exhibit a set in $NEXP$ that is $\leq^p_{2-tt}$ -complete but not $\leq^p_m$ -complete. Since we are working on $NEXP$ we have to take care that our construction does not use that $NEXP$ is closed under complementation anywhere. The constructions presented in [Wat87a, Wat87b] do make use of the fact that E is closed under complements, so a quite different strategy is needed here.

The first result (Theorem 3.9) exhibits a difference between complete sets with respect to $\leq^p_m$ and $\leq^p_{dtt}$ -reductions. The new tool that is needed comes from the work of Ganesan and Homer [GH89].

The next result appears in Theorem 3 in [GH89]

Theorem 3.8 *Any $\leq^p_m$ -complete set for NE is $\leq^p_{m,1\text{-}1,eh}$ -complete.*

While the major new contribution of this theorem was the one-one completeness, it is the exponential honesty[2] which will be crucial here.

We can now state the theorem which yields the desired differences between complete sets. A similar theorem can be found in Ganesan and Homer [GH89]. The proof presented here is simpler, more complete and will be generalized to other reducibilities later in this section.

[2]Recall that a reduction is exponential honest if it does not query strings that are exponentially smaller than the size of the input.

Theorem 3.9 *There is a set B which is $\leq^p_{2-dtt}$ -complete for NEXP but not $\leq^p_m$ -complete for NEXP.*

Proof: Let K be the $\leq^p_m$ -complete set for *NE* defined earlier, K is $\leq^p_m$ -complete for *NEXP* as well. The set B will be constructed so that its only elements are of the form $<e, x, l, i>$, $i = 0$ or $i = 1$.

B will be complete via the $\leq^p_{2-dtt}$ -reduction:

$$<e,x,l> \in K \leftrightarrow [<e,x,l,0> \in B] \vee [<e,x,l,1> \in B]$$

To ensure that B is not $\leq^p_m$ -complete, we diagonalize against all possible $\leq^p_m$ -reductions from Σ^* to B. Let f_i be the i^{th} polynomial-time computable function in some fixed enumeration of all such functions. We may assume that f_i runs in DTIME (n^i). We need a set of elements on which to diagonalize. To this end we define a sequence of integers $\{b(n)\}_{n \in I\!N}$ by $b(0) = b(1) = 1$, $b(m) = 2^{(b(m-1))^{m-1}} + 1$, for $m > 1$.

Let $H = \left\{0^{b(k)}\right\}_{k \in I\!N}$. It is easy to verify that $H \in P$. We use the sequence H to diagonalize against $\leq^p_m$ reductions.

We can now describe the construction of B. The set B is constructed in stages. At stage $k = 1, 2, \ldots$ we determine all elements in B of length $\leq (b(k))^k$. At stage 1 we put all strings s, $|s| \leq 1$ into $\overline{B}$. Now assume we have constructed B through stage $n-1$. Stage $n > 1$ is defined as follows.

stage n:

Compute $f_n(0^{b(n)})$. For all strings s of the form $<e, x, l, i>, (i \in \{0, 1\})$ with $(b(n-1))^{n-1} < |s| \leq (b(n))^n$, put $s \in B$ iff $s \neq f_n(0^{b(n)})$ and $<e, x, l> \in K$.

end of stage n

First note that $K \leq^p_{2-dtt} B$ via the reduction defined above. This is true, as for any $<e,x,l>$, if $<e,x,l> \in K$ then at least one of $<e,x,l,0>, <e,x,l,1>$ is put into B (without loss of generality we assume that $\|<e,x,l,0>\| = \|<e,x,l,1>\|$) and if $<e,x,l> \notin K$ then neither of the two strings is in B.

Claim 3.10 $B \in$ *NEXP*

Proof: Given a string s, $s \in B$ iff:

1. $s = <e, x, l, i>$ for some $e, x, l \in \Sigma^*, i \in \{0,1\}$,

2. $<e, x, l> \in K$, and

3. $s \neq f_k(0^{b(k)})$ where $b(k)$ is the least element in the sequence $\{b(n)\}_n$ with
$(b(k))^k \geq |s|$.

Condition 1 can be decided in linear time. Consider 3. By construction, $|s| > (b(k-1))^{k-1} \geq k$ and hence $(b(k))^k \leq 2^{k|s|}$. Now since $f_k \in \text{DTIME}\left(n^k\right)$ and $H \in P$, the $b(k)$ as in 3 can be found and the condition in 3 checked in $(b(k))^k \leq 2^{k|s|} \leq 2^{O\left(|s|^2\right)}$ steps. As $K \in NE$ the claim follows and in fact $B \in \text{NTIME}\left(2^{O(n^2)}\right)$. □

Thus we have $B \leq^p_{2-dtt}$ -complete for *NEXP*

Claim 3.11 *B is not $\leq^p_m$ -complete for NEXP.*

Proof: Assume B were $\leq^p_m$ -complete. Then by Theorem 3.8 there is a polynomial time computable f_n which reduces Σ^* to B and which is exponentially honest.

At stage n of the construction of B we computed $f_n(0^{b(n)})$. By the exponential honesty of f_n, $2^{|f_n(0^{b(n)})|} \geq |0^{b(n)}| = b(n) = 2^{(b(n-1))^{n-1}} + 1$, and so $|f_n(0^{b(n)})| > (b(n-1))^{n-1}$. Hence at stage n we put $f_n(0^{b(n)})$ into $\overline{B}$. This contradicts the assumption that f_n is a reduction of Σ^* to B □

This completes the proof of Theorem 3.9. □

A standard padding argument now yields the same result for *NE*.

Corollary 3.12 *There is a C which is $\leq^p_{2-dtt}$ -complete for NE but not $\leq^p_m$ -complete for NE.*

Proof: Let B be as in the previous theorem. Then, as noted above, $B \in \text{NTIME}\left(2^{O(n^2)}\right)$. Define $C = \{x10^{|x|^2} \mid x \in B\}$. Then

1. $C \in NE$

2. $B \leq^p_m C$ and hence C is $\leq^p_{2-dtt}$ -complete.

3. C is not $\leq_m^p$ -complete for NE. (Since $C \leq_m^p B$)

Hence C has the desired properties. □

The same proof works to yield these same results for any nondeterministic class, with a paddable $\leq_m^p$ -complete set, containing NE, including the class of recursively enumerable sets.

Next we turn to differentiating between complete sets for bounded truth table reductions. We will prove that, for any $k > 1$ there is a set that is $\leq_{k\text{-}tt}^p$ -complete, but not $\leq_{(k-1)\text{-}tt}^p$ -complete for NE.

For simplicity, we present the proof for the case $k = 3$.

The general theorem is a direct extension of the proof given here. The central idea in the proof is again a careful analysis of the honesty of the reductions. However, here we cannot avoid reductions that are not exponentially honest. Rather, we show that in exponential time, we can directly compute the result of dishonest queries made by the reduction as they are so much shorter than the input. Honest queries made by the reduction are handled as in Theorem 3.9. Furthermore, in addition to constructing the requisite set B, we need to explicitly construct a set of witnesses W in EXP that does not $\leq_{2-tt}^p$ -reduce to B. It is not sufficient to simply use Σ^* for the witness set as in Theorem 3.9.

Theorem 3.13 *There is a set B which is $\leq_{3-d}^p$ -complete for NEXP but not $\leq_{2-tt}^p$ -complete for NEXP.*

Note that this theorem separates both $\leq_{2-tt}^p$ and $\leq_{3-tt}^p$ -completeness and $\leq_{2-dtt}^p$ -completeness from $\leq_{3-d}^p$ -completeness.

Proof: The set B is constructed in stages, in a way similar to that of Theorem 3.9. B will be made $\leq_{3-d}^p$ -complete via the reduction $<e, x, l> \in K \leftrightarrow \exists i \in \{0, 1, 2\} : <e, x, l, i> \in B$. In order to ensure that B is not $\leq_{2-tt}^p$ -complete we simultaneously construct a set W in EXP that witnesses the incompleteness of B. We make use of the sequence $\{b(n)\}$ from the previous proof. Recall that $b(k) = 2^{(b(k-1))^{k-1}} + 1$.

Let M_i be the i^{th} 2-tt -reduction in some enumeration of such reductions and let $Q(M_i, x)$ be the set of (at most two) elements queried by $M_i^S(x)$ during its computation. M_i is assumed to run in time $p_i(n) = n^i$

and, as M_i is a truth-table reduction $Q(M_i, x)$ does not depend on S. We can now present the construction of B and W.

Initially $B = W = \emptyset$.

stage n:

At stage n we determine $B(y)$ for all strings y with $(b(n-1))^{n-1} < |y| \leq (b(n))^n$ and we decide whether or not $0^{b(n)} \in W$. (At this point in the construction $B \subseteq \Sigma^{\leq (b(n-1))^{n-1}}$.)

1. For all y with $(b(n-1))^{n-1} < |y| \leq (B(n))^n$, put

$$y \in B \leftrightarrow \begin{cases} \exists i \in \{0,1,2\}(y = <e,x,l,i>) & \text{and} \\ <e,x,l> \in K & \text{and} \\ y \notin Q(M_n, 0^{b(n)}) \end{cases}$$

2. Put $0^{b(n)} \in W \leftrightarrow M_n^B(0^{b(n)}) = 0$

end of stage n

Clearly, as $\|Q(M_n, 0^{b(n)})\| \leq 2$, $<e,x,l> \in K$ iff one of the three $<e,x,l,i> \in B$ $(i = 1,2,3)$. So B is $\leq_{3-d}^{p}$-hard for *NEXP*. Moreover since only elements not in $Q(M_n, 0^{b(n)})$ are added to B at stage n, and only elements of length greater than $(b(n))^n$ at subsequent stages $M_n^B(0^{b(n)}) = M_n^{B^{\leq (b(n-1))^{n-1}}}(0^{b(n)})$.

We proceed via a series of lemmata to complete the proof.

Lemma 3.14 *$W \in$ EXP.*

Proof: By the construction $W \subseteq \{0^{b(n)}\}$. The following algorithm tests if $0^{b(n)} \in W$.

1. compute $Q(M_n, 0^{b(n)})$.
2. For each $y \in Q(M_n, 0^{b(n)})$, compute if $y \in B$ as follows:
   if $|y| > (b(n-1))^{n-1}$ then $y \notin B$ by the construction.
   if $|y| \leq (b(n-1))^{n-1}$ then find the least k such that $|y| \leq (b(k))^k$.
   if $y \in Q(M_k, 0^{b(k)})$ then $y \notin B$.
   else $y \in B \leftrightarrow y = <e,x,l,i>$ for some $i \in \{0,1,2\}$ and $<e,x,l> \in K$.

3. Using the truth table computed by $M_n^B(0^{b(n)})$ and the information from 2, we can compute the value of $M_n^B(0^{b(n)})$. By the construction $0^{b(n)} \in W$ if and only if $M_n^B(0^{b(n)}) = 0$.

Step 1 takes at most $(b(n))^n$ steps. For step 2, given $0^{b(n)}$ as input, we can in $b(n)$ steps determine $(b(n-1))^{n-1}$ since $b(n) = 2^{(b(n-1))^{n-1}} + 1$. If $|y| \leq (b(n-1))^{n-1}$, then finding k least with $(b(k))^k \geq |y|$ can again be done within $b(n)$ steps. (by the definition of the sequence $\{b(n)\}$.). We can then compute $Q(M_k, 0^{b(k)})$ in $(b(k))^k < (b(n))^n$ steps and test if $y \in Q(M_k, 0^{b(k)})$. If so and if $y = <e,x,l,i>$ then computing if $<e,x,l> \in K$ deterministically takes $2^{2^{|<e,x,l>|}} \leq 2^{2^{(b(n-1))^{n-1}}} < 2^{b(n)}$ steps. Hence step 2 can be carried out in time $2^{b(n)} + (b(n))^n \in 2^{O(b(n))}$.

Finally step 3 can be done in $O\left((b(n))^n\right)$ steps. So deciding $0^{b(n)} \in W$ takes $2^{O(b(n))}$ steps and hence $W \in EXP$. □

Lemma 3.15 *B is $\leq^p_{3-d}$ -complete for NEXP.*

Proof: We have already observed that B is $\leq^p_{3-d}$ -hard. So it remains to prove that $B \in NEXP$.

Given y, the following algorithm tests if $y \in B$.

1. Find the least n with $|y| \leq (b(n))^n$.

2. Compute $Q(M_n, 0^{b(n)})$.

3. If $y \in Q(M_n, 0^{b(n)})$ then $y \notin B$
   else

$$y \in B \leftrightarrow \begin{cases} y = <e,x,l,i> & \text{for some } i \in \{0,1,2\} \\ \text{and } <e,x,l> \in K & \end{cases}$$

Now, for n as in 1, $|y| > (b(n-1))^{n-1}$, so by definition of the $\{b(n)\}$ sequence, $(b(n))^n = \left(2^{(b(n-1))^{n-1}} + 1\right)^n < \left(2^{|y|} + 1\right)^n < 2^{n|y|+1}$. Hence, since $n < |y|$, $(b(n))^n < 2^{O(|y|^2)}$, the value of n in step 1 can be found in $< (b(n))^n$ steps and step 2 can be computed in $(b(n))^n$ steps. Clearly then step 3 can nondeterministically be computed in time $2^{|y|}$ as $|<e,x,l>| < |y|$.

Thus steps 1 and 2 can be carried out deterministically in $2^{O(|y|^2)}$ steps and step 3 can be done in *NEXP*, so $B \in NEXP$. □

Lemma 3.16 *B is not $\leq^p_{2-tt}$ -complete for NEXP.*

Proof: Assume B were $\leq^p_{2-tt}$ -complete. Then by Lemma 3.14, W would be 2-tt-reducible to B, say by reduction M_n. But by the construction, we have $0^{b(n)} \in W$ if and only if $M_n^B(0^{b(n)}) = 0$, contradicting the assumption that M_n is the required reduction. □

This ends the proof of the theorem. □

Via the same padding argument as before one can prove,

Corollary 3.17 *There is a set C which is $\leq^p_{3-d}$ -complete for NE, but not $\leq^p_{2-tt}$ -complete for NE.*

Straightforward modifications of the above method yield a number of extensions of these results.

- The above proof can be generalized to give the same results for k-d-reductions instead of 3-d.
- The above proofs work as well for any nondeterministic class with a paddable $\leq^p_m$ -complete set which contains *NE*.

A very similar argument can be used to separate $\leq^p_{btt}$ and $\leq^p_{tt}$ -completeness. The proof is only sketched.

Theorem 3.18 *There is a set B which is $\leq^p_{tt}$ -complete for NEXP, but not $\leq^p_{btt}$ -complete for NEXP.*

Proof: (Sketch) As before we construct B together with a witness set W. The reduction making B $\leq^p_{tt}$ -complete will be:

$$<e, x, l> \in K \leftrightarrow \exists i(i \leq |<e, x, l>| \text{ and } <e, x, l, i> \in B)$$

At stage n of the construction we treat n as a pair $n = <n_1, n_2>$ and we try to diagonalize against the n_1^{st} tt-reduction M_{n_1}, but we only do this if M_{n_1} queries $\leq n_2$ queries to the oracle.

More formally, stage n of the construction is as follows:

stage n:

1. Let $n = <n_1, n_2>$ and compute $Q(M_{n_1}, 0^{b(n)})$.
2. If $\|Q(M_{n_1}, 0^{b(n)})\| > n_2$ then
   for all y with $(b(n_1))^{n-1} < |y| \leq (b(n))^n$, put
   $y \in B \leftrightarrow y = <e, x, l, 0>$ and $<e, x, l,> \in K$.
3. If $\|Q(M_n, 0^{b(n)})\| \leq n_2$ then
4. put $0^{b(n)} \in W \leftrightarrow M^B_{n_1}(0^{b(n)}) = 0$ and
5. for all y with $(b(n-1))^{n-1} < |y| \leq (b(n))^n$ put

$$y \in B \leftrightarrow \begin{cases} y \notin Q(M_n, 0^{b(n)}) & \text{and} \\ \exists i(y = <e, x, l, i> \text{ and } i \leq |<e, x, l>| & \text{and} \\ <e, x, l> \in K) & \end{cases}$$

end of stage n

Now exactly as in Theorem 3.13, we can prove that $W \in EXP$ and that $B \in NEXP$. In step 5 of the construction we have that $(b(n-1))^{n-1} < |y|$ so it follows from the definition of $\{b(n)\}$ that if $y = <e, x, l>$ and $i \leq |<e, x, l>|$ then $|<e, x, l>| > n > n_2$. So in step 5 we have room to code K into B. Hence B is $\leq^p_{tt}$ -complete for $NEXP$.

Now, let M_{n_1} be a $\leq^p_{btt}$ -reduction, say with norm n_2. Then at stage $n = <n_1, n_2>$ we will find that $\|Q(M_{n_1}, 0^{b(n)})\| \leq n_2$ and so in step 3 of the construction will put $0^{b(n)} \in W \leftrightarrow M^B_{n_1}(0^{u(n)}) = 0$, and so $0^{b(n)}$ will witness the fact that M_{n_1} does not btt-reduce W to B and so B is not $\leq^p_{btt}$ -complete for $NEXP$. $\square$

We end this Subsection by noting that all the constructions so far can be carried out in deterministic exponential time, provided the set K is taken to be the standard complete set for E. This yields that all the corresponding completeness notions w.r.t. these reductions on $EandEXP$ are also different.

3.3.2 Disjunctive versus Conjunctive

In this section we will pay attention to two specific types of truth-table reductions: disjunctive and conjunctive truth-table reductions. First we only consider reductions that query a constant number of strings.

We note that if the bound is 1, disjunctive reductions are the same as conjunctive, which are just many-one reductions.

As we have seen in the previous section $\leq^p_{2-dtt}$ -completeness is different from $\leq^p_m$ -completeness (Theorem 3.9). Furthermore $\leq^p_{k\text{-}dtt}$ is different from $\leq^p_{(k-1)\text{-}dtt}$ (Theorem 3.13).

It is not hard to see that the constructions can be altered such that the statement is true for conjunctive reductions instead of disjunctive reductions. This implies that for conjunctive as well as disjunctive reductions one more query makes a difference.

In this section we will prove that even with the same number of queries disjunctive and conjunctive reductions are incomparable.

We will first show that this is true for $\leq^p_{2-dtt}$ and $\leq^p_{2-ctt}$ completeness notions. Again from that construction it will be clear that this proof can be generalized to arbitrary disjunctive and conjunctive reductions.

Theorem 3.19 *There exists a set $A \in NEXP$ such that A is $\leq^p_{2-dtt}$ -complete but not $\leq^p_{2-ctt}$ -complete.*

Proof: Let K be the standard $\leq^p_m$ -complete set for NE as defined above. To achieve the separation we construct a set $W \in E$ and a set $A \in NEXP$ such that $W \not\leq^p_{2-ctt} A$ but $K \leq^p_{2-dtt} A$. We assume an enumeration of polynomial time 2-conjunctive truth-table reductions $M_1, M_2, \ldots$ where M_i runs in time n^i. We need a set of elements on which to diagonalize. To do this, we define a sequence of integers $\{b(n)\}_n$:

$$b(n) = \begin{cases} 1 & \text{if } n \leq 1 \\ 2^{b(n-1)^{n-1}} + 1 & \text{otherwise} \end{cases}$$

We construct A and W in stages. $A \subseteq \{0,1\} \times \Sigma^*$ and $W \subseteq \{0\}^*$. At stage n, we define A_n and decide whether $0^{b(n)} \in W$ or $0^{b(n)} \in \overline{W}$. We let $A_{<n} = \bigcup_{i=0}^{n-1} A_i$ and $A = \bigcup_{n=0}^{\infty} A_n$

stage 0: $A_0 = W = \emptyset$.

stage n:

Let $A_n^{'} = \{<i,z> \mid z \in K \text{ and } b(n-1)^{n-1} < |<i,z>| \leq b(n)^n \text{ and } i \in \{0,1\}\}$
Simulate M_n on input $0^{b(n)}$. M_n will query at most two strings x and y, w.l.o.g. let x be the largest (in lexicographic order) of the two. M_n accepts iff x and y are both in the oracle set.
There are two cases:

1. $|x| \leq b(n-1)^{n-1}$

2. $b(n-1)^{n-1} < |x| \leq b(n)^n$

In case 1, compute the answers relative to $A_{<n}$ of both x and y and put $0^{b(n)} \in W$ iff M_n rejects. Let $A_n = A_n^{'}$
In case 2, put $0^{b(n)} \in W$ and let $A_n = A_n^{'} - \{x\}$. This ensures that M_n^A rejects on input $0^{b(n)}$.

end of stage n

The remainder of the proof consists of four items. We show that $A \in \mathit{NEXP}$, that $W \in E$, that A is not $\leq_{2-ctt}^p$ -complete, and finally that A *is* $\leq_{2-dtt}^p$ -complete.

- We show first that $A \in \mathit{NEXP}$. To decide $<i,z> \in A$ $(i = 0,1)$ compute n such that $b(n)^n \geq |<i,z>| > b(n-1)^{n-1}$, which can be done in linear time. Simulate machine M_n on input $0^{b(n)}$ and compute x and y. If $<i,z> = x$ reject, else accept iff $z \in K$. All this can be done in nondeterministic exponential time, since simulation of the machine M_n on input $0^{b(n)}$ takes time $b(n)^n \leq 2^{n(b(n-1)^{n-1}+1)} \leq 2^{|<i,z>|^2}$, as $b(n) \geq 1$ and $b(n) \geq 2n$ whenever $n \geq 3$.

- Next we show that $W \in E$. On input $0^{b(n)}$ simulate M_n on input $0^{b(n)}$ and compute the queries x and y. Assume again that x is the larger query in lexicographical order. If $|x| > b(n-1)^{n-1}$ we accept, else we will decide membership of x and y to A, and accept iff M_n rejects. To compute if $x \in A$, determine $n' < n$ such that $b(n'-1)^{n'-1} < |x| \leq b(n')^{n'}$. $x \in A$ iff x is not the largest query queried by $M_{n'}$ and $x \in K$. This takes deterministic time $2^{2^{|x|}} < 2^{b(n)}$. Similarly the membership of y in A can be decided.

- Now assume for a contradiction that A is $\leq^p_{2-ctt}$ -complete. Note that $0^{b(n)} \in W$ iff M_n rejects. Then there must be a 2-conjunctive truth-table reduction from W to A. Let M_j be the machine witnessing this reduction. But $0^{b(j)}$ is in W iff M_j on input $0^{b(j)}$ rejects. This contradicts the fact that M_j reduces W to A. This proves that A is not $\leq^p_{2-ctt}$ -complete.

- Finally we give the $\leq^p_{2-dtt}$ -reduction from K to A. Since in every step only one of the pairs $<1, x>$ or $<0, x>$ can be deleted, $x \in K$ iff $<0, x> \in A$ or $<1, x> \in A$. Therefore, the following reduction reduces K to A:

$$g(x) = \{<0, x> \vee <1, x>\}$$

□

The same proof technique yields the following theorem.

Theorem 3.20 *There exists a set $A \in NEXP$ such that A is $\leq^p_{2-ctt}$ -complete but not $\leq^p_{2-dtt}$ -complete.*

Proof: The proof is almost the same as the previous one. It differs only in case 2 of the diagonalization. Here we put $0^{b(n)}$ not in W and *add* x to $A^{'}_n$. In this way we ensure that $0^{b(n)} \notin W$ iff M_n^A accepts. Note that $x \in K$ iff $<0, x> \in A$ and $<1, x> \in A$. The $\leq^p_{2-ctt}$ -reduction from K to A becomes:

$$g(x) = \{<0, x> \wedge <1, x>\}$$

□

It is easy to see that the proofs generalize to $\leq^p_{k\text{-}dtt}$ -complete sets v.s. $\leq^p_{k\text{-}ctt}$ -complete sets (for $k \geq 2$).

Corollary 3.21 *For all $k \geq 2$, there exists a set A that is $\leq^p_{k\text{-}tt}$ -complete for NEXP but not $\leq^p_{k\text{-}dtt}$ ($\leq^p_{k\text{-}ctt}$) -complete for NEXP.*

This corollary can be strengthened. We are now able to construct a set that is $\leq^p_{k\text{-}tt}$ -complete but neither $\leq^p_{k\text{-}dtt}$ -complete nor $\leq^p_{k\text{-}ctt}$ -complete.

Corollary 3.22 *For all $k \geq 2$, there exists a set A that is $\leq^p_{k\text{-}tt}$ -complete for NEXP but neither $\leq^p_{k\text{-}dtt}$ -complete nor $\leq^p_{k\text{-}ctt}$ -complete for NEXP.*

Proof: To prove this we use the constructions of Theorem 3.19 at the even stages and the constructions of Theorem 3.20 at the odd stages. □

Corollary 3.23 *For all $k \geq 2$ there exists a set A that is $\leq^p_{k\text{-}tt}$ -complete for NEXP but neither $\leq^p_{dtt}$ -complete nor $\leq^p_{ctt}$ -complete.*

Using standard padding techniques the results in this section go through for *NE*, and since we do not make use of any special properties of nondeterminism, they also hold for *E* and *EXP*. By complementation the results also hold for co-*NE* and co-*NEXP*.

3.3.3 Bounded Turing versus bounded Truth-Table

So far we have only considered reductions that obeyed restrictions that admitted only non-adaptive computations. In this section we will explore whether adaptive computations gain power over non-adaptive ones.

The general theme with adaptive computations is that the number of queries *queried* is the same as with non-adaptive computations, but the number of *potential* queries that can be queried is exponentially bigger. So if the number of queries for an adaptive computation is constant, the same computation can be performed by a non-adaptive reduction that queries an exponential (but still constant) number of strings. Thus, a k -Turing reduction can be performed by a $(2^k - 1)$ -truth-table reduction.

In this section we prove that k-Turing reductions are more powerful than k-truth-table reductions for $k > 1$, and that for $k < \ell < 2^k - 1$, k-Turing and ℓ-truth table reductions are incomparable.

Theorem 3.24 *For every k there exists a set D in NEXP that is $\leq^p_{k\text{-}T}$ -complete but not $\leq^p_{(2^k-2)-tt}$ -complete.*

As an example of the techniques used, we first prove the degenerate case $k = 2$, i.e. we will construct a set $D \in NEXP$ such that D is $\leq^p_{2-T}$ -complete but not $\leq^p_{2-tt}$ -complete.

Proof: Let $M_1, M_2, \ldots,$ be an enumeration of the 2-truth-table reductions, where M_i runs in time n^i. Let K be the standard $\leq^p_m$ -complete set for NE and let $\{b(n)\}_n$ be the sequence defined in the proof of Theorem 3.19. We will construct sets $D \in NEXP$ and $W \in E$ such that $W \not\leq^p_{2-tt} D$, and $K \leq^p_{2-T} D$. W and D will be constructed in stages. At each stage n we will define a set D_n and we let $D = \bigcup_{n=0}^{\infty} D_n$.

To ensure that $K \leq^p_{2-T} D$, we have to exploit the fact that a 2-Turing reduction can ask 3 queries in its *entire* oracle tree, whereas a 2-truth-table reduction can ask at most 2 queries in its entire oracle tree. We will ensure that $D \subseteq \{0, 1, 2\} \times K$, and use the following 2-Turing reduction M_T to reduce K to D:

On input x, first query $<0, x>$. If a 1 is written on the ANSWER-tape, query $<1, x>$, and accept iff another 1 is written on the ANSWER-tape. If a 0 is written on the ANSWER-tape for query $<0, x>$, query $<2, x>$ and accept iff a 1 is written on the ANSWER-tape.

For every 2-truth-table reduction, and for every x, at least one of the strings $<0, x>, <1, x>, <2, x>$ is not queried on input $0^{b(n)}$, where n is such that $|<0, x>| \leq b(n)^n$. This provides enough freedom to diagonalize against the 2-truth-table reductions, while still keeping $K \leq^p_{2-T} D$ by M_T.

stage 0 : $D_0 = W = \emptyset$

stage n:

Let $D_n = \{<i, x> \mid x \in K$ and $b(n-1)^{n-1} < |<i, x>| \leq b(n)^n$ and $0 \leq i \leq 2\}$. Simulate M_n on input $0^{b(n)}$. If M_n queries strings of length $\leq b(n-1)^{(n-1)}$ compute the answers to those strings. i.e., compute the membership of those strings in $\bigcup_{i=0}^{n-1} D_i$.

Let Q be the set of (at most two) queries $\in \{0, 1, 2\} \times \Sigma^*$ with length $> b(n-1)^{(n-1)}$. Let $i_0 \in \{0, 1, 2\}$ be a number such that Q contains no string of the form $<i_0, x>$. Now we take the following action, depending on the value of i_0:

$i_0 = 0$: For every y occurring as second member in a pair of Q do
$$D_n := (D_n - \{<2, y>\}) \cup \{<1, y>\}$$
$i_0 = 1$: For every y occurring as second member in a pair of Q do
$$D_n := (D_n - \{<2, y>\}) \cup \{<0, y>\}$$
$i_0 = 2$: For every y occurring as second member in a pair of Q do
$$D_n := (D_n - \{<0, y>\}) \cup \{<1, y>\}$$

Now we are able to compute whether M_n accepts or rejects the input $0^{b(n)}$ when given the oracle $\bigcup_{i=0}^{n} D_i$. The fact that no strings of length $\leq b(n)^n$ (i.e. strings that can be queried by M_n on input $0^{b(n)}$) are defined inside or outside D at subsequent stages ensures that the computation on this input is the same as with oracle D.
Put $0^{b(n)}$ in W iff M_n rejects on input $0^{b(n)}$.

end of stage n

We can use an argument similar to the one used in the proof of Theorem 3.19 to prove that $D \in \mathit{NEXP}$, $W \in E$ and W is not $\leq^p_{2-tt} D$. It remains to prove that D is $\leq^p_{2-T}$ -hard for *NEXP*. Our 2-Turing reduction M_T accepts x iff either $(<0, x> \in D \wedge <1, x> \in D)$ or $(<2, x> \in D \wedge <0, x> \notin D)$.

We have the following possibilities for $D \cap \{<0, x>, <1, x>, <2, x>\}$:

$x \in K$: $\{<0, x>, <1, x>, <2, x>\}$ or $\{<0, x>, <1, x>\}$ or $\{<1, x>, <2, x>\}$.

$x \notin K$: $\emptyset$ or $\{<0, x>\}$ or $\{<1, x>\}$.

Thus, M_T accepts x iff $x \in K$ as required. This ends the proof of Theorem 3.24 □

For this proof, it was essential that a 2-Turing reduction can have more queries in its entire oracle tree than a 2-truth-table reduction. Since a k-Turing reduction can have $2^k - 1$ queries in its entire oracle tree, whereas a $2^k - 2$ truth-table reduction can have at most $2^k - 2$ queries in its entire oracle tree, we can use a generalization of the previous construction to obtain a set D that is $\leq^p_{k-T}$ -complete, but not $\leq^p_{(2^k-2)-tt}$ -complete, thus proving Theorem 3.24.

Proof: Let $M_1, M_2, \ldots,$ be an enumeration of the $2^k - 2$-truth-table reductions, where M_i runs in time n^i. Let K be a standard $\leq^p_m$ -complete set for *NE* and let $\{b(n)\}_n$ be the sequence defined in the proof of Theorem 3.19. We construct set D and W in stages; $D = \bigcup_{n=0}^{\infty} D_n$. We will ensure that $D \subseteq \{0, \ldots, 2^k - 2\} \times \Sigma^*$, and use the following k-Turing reduction M_T to reduce K to D.

On input x, first query $<0, x>$. For the j-th query $<i, x>$, where $j < k$ do the following: if a 1 is written on the ANSWER-tape, then query $<2i + 1, x>$, else query $<2i + 2, x>$. Accept iff the k-th bit on the ANSWER-tape is a 1.

stage 0 : $D_0 = W = \emptyset$

stage n:

Let $D_n = \{<i, x> \mid x \in K$ and $b(n-1)^{n-1} < |<i, x>| \leq b(n)^n$ and $0 \leq i \leq 2^k - 2\}$.

Simulate M_n on input $0^{b(n)}$. If M_n queries strings of length $\leq b(n-1)^{(n-1)}$ compute the answers to those strings.

Let Q be the set of queries $\in \{0, \ldots, 2^k - 2\} \times \Sigma^*$ with length $> b(n-1)^{(n-1)}$. Let $i_0 \in \{0, \ldots, 2^k - 2\}$ be such that Q contains no string of the form $<i_0, x>$.

Consider the following tree of depth k, where the nodes are labeled $0, \ldots, 2^k - 2$: The root has label 0, and for each node at depth $< k$ with label i, the left child has label $2i + 1$, and the right child label $2i + 2$. We will identify roots with their labels, and we will visualize this tree as a query tree such that on input x the node labeled i corresponds to query $<i, x>$. If our Turing reduction receives a 1 on its ANSWER-tape to a query represented at some node then the next query computed is represented by its left child and if a 0 is written on the ANSWER-tape then the next query computed is represented by its right child.

For every y that occurs as second member in a pair of Q and and for every $i \in \{0, \ldots 2^k - 2\}, i \neq i_0$, we take the following action:

1. if i occurs on the path from the root to i_0 then
   if i_0 is in the left subtree of i then $D_n := D_n \cup \{<i, y>\}$
   if i_0 is in the right subtree of i then $D_n := D_n - \{<i, y>\}$
2. if i occurs to the left of the path from the root to i_0 then
   $D_n := D_n \cup \{<i, y>\}$

3. if i occurs to the right of the path from the root to i_0 then $D_n := D_n - \{<i, y>\}$
4. if i is in the left subtree of i_0 then $D_n := D_n \cup \{<i, y>\}$
5. if i is in the right subtree of i_0 then $D_n := D_n - \{<i, y>\}$

Now we are able to compute whether M_n accepts or rejects.
Put $0^{b(n)}$ in W iff M_n rejects on input $0^{b(n)}$.

end of stage n

We can use a similar argument as in the proof of Theorem 3.19, to prove that $D \in NEXP$, $W \in E$ and $W \not\leq^p_{(2^k-2)-tt} D$. It remains to prove that D is $\leq^p_{k\text{-}T}$ -hard for $NEXP$.

Recall that our k-Turing reduction M_T works as follows: on input x, first query $<0, x>$. For each query $<i, x>$ at depth $< k$ do the following: if a 1 is written on the ANSWER-tape, then query $<2i+1, x>$, else query $<2i+2, x>$. Accept iff the last bit on the ANSWER-tape is a 1. View this reduction as a tree of depth k, where the nodes are labeled by the queries, and a 1 (resp. 0) written on the ANSWER-tape for some query, corresponds to taking the left (resp. right) branch.

If $x \in K$, M_T on input x takes either the leftmost path in its oracle tree (if $i_0 = 0$), or the leftmost path through the node labeled i_0 which corresponds to query $<i_0, x>$. In either case we accept.

If $x \notin K$, M_T on input x takes either the rightmost path in its oracle tree, or the rightmost path through $<i_0, x>$. In either case we reject.

Thus, M_T is a reduction from K to D. $\Box$

Now we will construct a set D in $NEXP$ that is $\leq^p_{(k+1)-tt}$ -complete but not $\leq^p_{k\text{-}T}$ -complete. A $\leq^p_{k\text{-}T}$ -reduction can be represented as a binary tree of depth k, where every node in the tree represents a query. The reduction starts with the query represented by the root and proceeds after obtaining an answer to each query as follows. If a 1 is written on the ANSWER-tape, we proceed to the left branch otherwise to the right branch. The leaves of the tree are labeled with the quality accept or reject. The idea of the oracle construction is to force the $\leq^p_{k\text{-}T}$ -reduction into one branch by leaving out all the queries (if possible) of that branch. Since there are only k queries on one branch there remains the freedom to code an extra pair of K into D that can be queried by a $\leq^p_{(k+1)-tt}$ -reduction.

Theorem 3.25 *There exists a set D in NEXP that is $\leq^p_{(k+1)-tt}$ -complete but not $\leq^p_{k\text{-}T}$ -complete.*

Proof: We only give the construction for $k = 4$. Let $M_1, M_2, \ldots$ be an enumeration of $\leq^p_{4-T}$ -reductions. Let K be the standard $\leq^p_m$ -complete set for $NEand\{\mathrm{b(n)}\}_n$ the sequence defined in the proof of Theorem 3.19. Again we use a stage construction.

stage n:

$D'_n := \{<i,x> \mid x \in K$ and $b(n-1)^{(n-1)} < |<i,x>| \leq b(n)^n$ and $0 \leq i \leq 4\}$
Simulate M_n on input $0^{b(n)}$, and compute the answers to the queries of length $< b(n-1)^{n-1}$. Now evaluate the branch where all the other queries receive a 0 on the ANSWER-tape. Let Q' be the set of the queries that are large (of length $\geq b(n-1)^{(n-1)}$).
Put $0^{b(n)}$ in W iff M_n rejects
$D_n := D'_n \backslash Q'$

end of stage n

Note that for every x : $x \in K$ iff $<i,x> \in D$ for some i. The 5-truth-table reduction from K to D becomes:

$$g(x) = \{<0,x> \vee \ldots <4,x>\}$$

□

Corollary 3.26 *If $k < \ell < 2^k - 1$, then $\leq^p_{k\text{-}T}$ and $\leq^p_{\ell-tt}$ are incomparable with respect to complete sets for NEXP.*

As before the results go through for NE, E and EXP.

3.3.4 Positive versus Non-Positive

In this section we will consider the completeness notions induced by positive reductions. The reduction originally stems from recursion theory, where Jockusch [Joc68] introduced this reducibility. He used this reduction to show various facts about recursively enumerable sets.

Selman [Sel82] introduced the polynomial time variant of this reduction. He used it to show that $\leq^p_m$ and $\leq^p_{tt}$ -reductions are not the same on NP, unless $E = NE$. See chapter 7 for more details.

A positive reduction is a reduction that can be characterized with the following behavior: if $A \subseteq B$, then $L(M, A) \subseteq L(M, B)$. This condition is stating that if an oracle machine accepts an input x, then it keeps accepting x if more strings are added to the oracle set (or equivalently if it rejects it keeps rejecting when more strings are added to the complement of the oracle set).

So far, we encountered 3 types of positive reductions: disjunctive, conjunctive and many-one reductions. We saw that k -disjunctive (and k -conjunctive) reductions were able to beat $k-1$ -Turing reductions w.r.t. their respective completeness notions. The main question that will be addressed here is the following: Is there a k and a set that is k truth-table complete, but not k positive complete? We have seen that for $k = 1$ all the completeness notions are positive, so we have to look for $k > 1$.

We will show that such a set indeed exist in a very strong way: There exists a set A in EXP that is $\leq^p_{2-tt}$ -complete but not $\leq^p_{pos}$ -complete.

The idea is again to use a diagonalization construction similar to the previous constructions. Note that we are working on EXP, so we may (and will) use the fact that this class is closed under complements. Again we build a set W and a set A such that $W \not\leq^p_{pos} A$ but on the other hand ensure that $K \leq^p_{2-tt} A$.

The main two differences are that the $\leq^p_{2-tt}$ -reduction cannot be positive. Furthermore since we are dealing with a Turing reduction to diagonalize against, we cannot find strings that are not queried and thus code K on that string. We get around this last problem by finding strings that do not matter in the computation of the $\leq^p_{pos}$ -reduction. The details are given below.

Theorem 3.27 *There exists a set A in EXP such that A is* $\leq^p_{2-tt}$ -

complete, but not $\leq_{pos}^{p}$ -complete.

Proof: Let $M_1, M_2, \ldots,$ be an enumeration of positive Turing reductions, where M_i runs in time n^i. Let K be the standard $\leq_m^p$ -complete set for *EXP*. We will construct W and A in *EXP* such that $W \not\leq_{pos}^{p} A$ and $K \leq_{2-tt}^{p} A$. W and A will be constructed in stages. At each stage n we will define a set A_n and we let $A = \bigcup_{n=0}^{\infty} A_n$. To ensure that A is $\leq_{2-tt}^{p}$ -complete, we take care that for all x: $x \in K \Leftrightarrow$ exactly one of the two strings $<0,x>$ or $<1,x>$ is in A. If x is not in K we have the freedom to either put both $<0,x>$, $<1,x>$ in A or leave them both out of A. So the $\leq_{2-tt}^{p}$ -reduction from K to A is a $\leq_{2\text{-}\oplus}^{p}$ -reduction. Let $b(0) = 1$, and $b(n) = b(n-1)^{(n-1)} + 1$.

stage 0: $A_0 = W = \emptyset$.

stage n:

Simulate M_n on input $0^{b(n)}$; for queries of size less than $b(n)$ compute whether they are in $\bigcup_{i=0}^{n-1} A_i$ and write accordingly a 0 or 1 on the ANSWER-tape. for the other queries q, write a 1 on the ANSWER-tape if q is of the form $<1,x>$ and write a 0 otherwise. let FINAL be the accepting or rejecting final state of $M_n(0^{b(n)})$.
Put $0^{b(n)}$ in W iff FINAL = reject.
For $|<i,x>|$ such that $b(n) \leq |<i,x>| \leq b(n)^n$ do the following:

1. $x \in K : <0,x> \notin A$ and $<1,x> \in A$.
2. $x \notin K$:
   - FINAL = reject: $<0,x> \notin A$ and $<1,x> \notin A$.
   - FINAL = accept: $<0,x> \in A$ and $<1,x> \in A$.

end of stage n

From the previous constructions it should be clear that W and A are both in *EXP*, furthermore A is $\leq_{2-tt}^{p}$ -complete since $K \leq_{2\text{-}\oplus}^{p} A$. It remains to be shown that $W \not\leq_{pos}^{p} A$.

Suppose $W \leq_{pos}^{p} A$ via machine M_m. Consider the simulation of $M_m(0^{b(m)})$ performed at stage m. Suppose that FINAL was accept. (for reject a similar argument holds). So $0^{b(m)} \notin W$. Apparently $M_m(0^{b(m)})$ changed it's mind and now rejects $0^{b(m)}$. For queries of the form $<i,x>$ $(i=0,1)$ with $x \in K$, the answers written on the ANSWER-tape in the simulation of $M_m(0^{b(m)})$ are consistent with A. On the other hand, the answers to the queries of the form $<i,x>$ $(i=0,1)$ with $x \notin K$, might not be consistent, but the only difference is that they now might be in A, where a 0 was written on the ANSWER-tape, but since M_m is a positive reduction, this cannot change it's mind from accepting to rejecting. So $0^{b(m)} \notin W$ and $M_m^A(0^{b(m)})$ accepts, which shows the contradiction. $\Box$

Corollary 3.28 *$\leq^p_{(k-1)\text{-}T}$ -completeness and $\leq^p_{k\text{-}pos}$ -completeness notions $(k>1)$ for EXP are incomparable.*

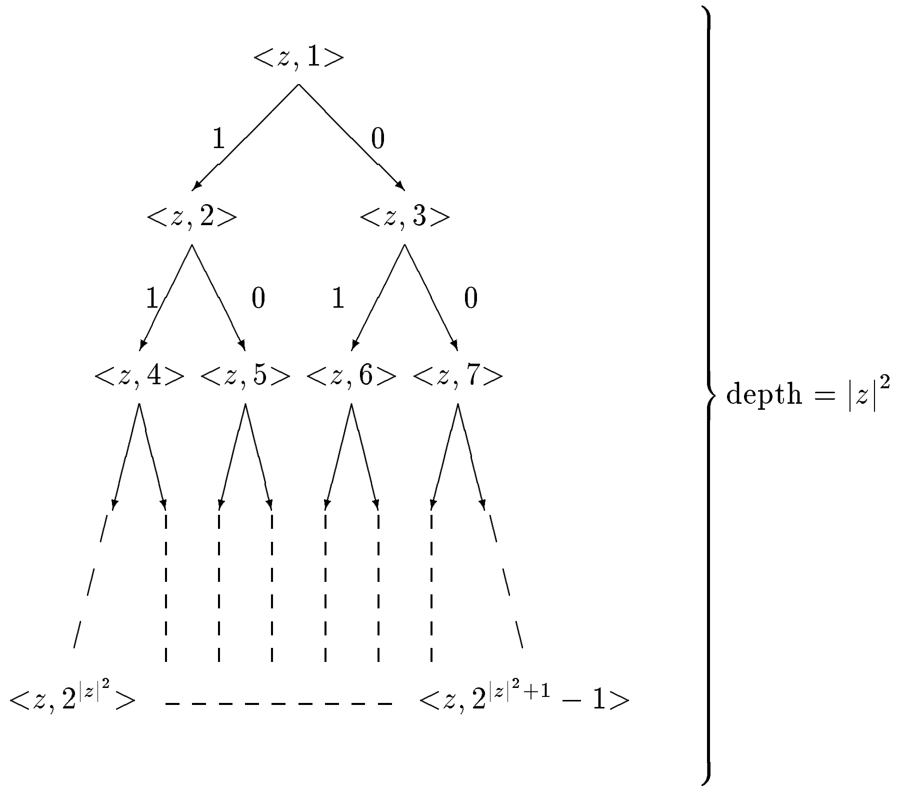
As usual the result goes through for *E*. At the present time we do not know how to extend this result to *NEXP*; again the problem is that we used the fact that *EXP* is closed under complementation.

3.4 The Sky is the Limit

Finally, we want to separate truth-table completeness from Turing completeness on *NE*. While the underlying ideas are the same, the construction is considerably more complex. The key is to define the correct Turing reduction within which there is room to diagonalize against all tt-reductions.

Theorem 3.29 *There is a set B which is $\leq^p_T$ -complete but not $\leq^p_{tt}$ -complete for NEXP.*

Proof: First we define the Turing reduction which we will use to show B $\leq^p_T$ -complete. On any input z, the Turing reduction relative to B queries a series of $|z|^2$ queries. Each query is of the form $<z,i>$, where $i < 2^{|z|^2}$. For convenience we require that all queries have the same length. We do this by padding each i in the pair $<z,i>$ with enough leading 0's so that $|<z,i>| = |<z,2^{|z|^2+1}-1>|$. So the length of each query $<z,i>$ is $|z|+|z|^2$. Which queries are queried depends on the answers to the previous query. The first string queried is $<z, 1>$. If a 1 is written on the ANSWER-tape then $<z, 2>$ is queried next, if on the other hand a 0 is written on the ANSWER-tape then $<z, 3>$ is the next query. More generally, when $<z, i>$ is queried, a 1 is written on the ANSWER-tape, results in $<z, 2i>$ being the next query and a 0 in $<z, 2i + 1>$ being the next query. This process continues on for $|z|^2$ many queries. The reduction then halts and accepts z iff the number of 1's on the ANSWER-tape is odd. A picture of the query tree of this reduction on input z is:



Note that the number of nodes in this tree is $2^{|z|^2+1} - 1$ and the number of leaves is $2^{|z|^2}$. Let $T^B(z)$ be this Turing reduction.

The property we will preserve, and which will guarantee $\leq_T^p$ -completeness of B is that $z \in K$ if and only if $T^B(z)$ accepts.

Some terminology will be helpful here in order to simplify the construction. For any z, let R_z be the query tree for the reduction $T^B(z)$

described and pictured above, and let N_z be the collection of nodes in R_z. Assume we are given a set $S \subseteq N_z$ with $\|S\| < 2^{|z|^2-1}$. Then, as there are $2^{|z|^2-1}$ pairs of leaves in R_z, each of them having the form $<z, 2i>, <z, 2i+1>$, there is some such pair neither of which are in S. Let $\ell(S)$ be the lexicographically least leaf in $N_z - S$ such that $p(S)$, the path from $<z, 1>$ to $\ell(S)$ in R_z, has the property that the number of "1" edges on $p(S)$ is even. (Note: such an $\ell(S)$ and $p(S)$ must exist since both $<z, 2i>$ and $<z, 2i+1>$ are in $\overline{S}$ and one of the paths $<z, 1> \ldots <z, 2i>$ or $<z, 1> \ldots <z, 2i+1>$ must contain an even number of "1" edges.)

The construction will work as follows. For each z we find a particular set $S \subseteq N_z$ as above and corresponding path $p(S)$. We then define B on elements $<z, i>$ so that the reduction $T^B(z)$ follows the path $p(S)$ through R_z. Finally we put $\ell(S) \in B \leftrightarrow z \in K$. This will ensure that

$$\begin{aligned} z \in K &\leftrightarrow \text{ The number of "1" edges on } p(S) \text{ is odd} \\ &\leftrightarrow T^B(z) \text{ accepts} \end{aligned}$$

As in the previous proofs we simultaneously construct B and a witness set W. We again use the sequence $\{b(k)\}$. However, to diagonalize against the nth tt-reduction M_n, we find a suitable point $0^{v_n} \in \{0^{b(k)}\}$ and diagonalize at 0^{v_n}.

It is simplest to define the sequence $\{v_n\}$, and prove it exists before presenting the actual construction. Let $v_0 = 0$ and assume v_{n-1} has been defined. Then v_n is the smallest element $b(t)$ in the sequence $\{b(k)\}$ such that

- $b(t) > v_{n-1}$ and
- $(b(t))^t < 2^{(b(t-1))^{2(t-1)}-1}$

Claim 3.30 *A sequence $\{v_n\}$ exists as defined above.*

Proof: By definition of $\{b(k)\}$, $b(k) = 2^{(b(k-1))^{k-1}} + 1$, so

$$\begin{aligned} (b(k))^k &= \left(2^{(b(k-1))^{k-1}} + 1\right)^k \\ &< \left(2^{(b(k-1))^{k-1}+1}\right)^k \\ &= 2^{k(b(k-1))^{k-1}+k} \end{aligned}$$

Since the sequence $\{b(k)\}$ increases exponentially, $k{<<}b(k-1)$, and so for k sufficiently large $2^{k(b(k-1))^{k-1}+k} < 2^{(b(k-1))^{2(k-1)}-1}$ as needed. $\Box$

Initially $B = W = \emptyset$. We can now define stage n of the construction. At stage n we decide whether to put 0^{v_n} into W and we define B on all y with $(v_{n-1})^{n-1} < |y| \leq (v_n)^n$.

stage n:

At this point $B \subseteq \Sigma^{\leq (v_{n-1})^{n-1}}$.

1. Put $0^{v_n} \in W \leftrightarrow M_n^B(0^{v_n}) = 0$.
2. By definition of v_n, $v_n \in \{b(k)\}$, say $v_n = b(t)$. For all y such that $(v_{n-1})^{n-1} < |y| \leq (b(t-1))^{t-1}$, put

$$y \in B \leftrightarrow \left\{ \begin{array}{l} y = {<}e, x, l, 2^{|<e,x,l>|^2+1} - 1{>} \\ \text{and } {<}e, x, l{>} \in K \end{array} \right.$$

3. For all y such that $(b(t-1))^{t-1} < |y| \leq (v_n)^n$, if $y = {<}e, x, l, i{>}$ with $i < 2^{|<e,x,l>|^2+1}$ then set $S = Q(M_n, 0^{v_n}) \cap N_{<e,x,l>}$. Compute $\ell(S)$ and $p(S)$ as defined above. (Note: we will prove later that $p(S)$ and $\ell(S)$ exist.) Then put $y \in B$ only if either
   (a) $y \in p(S)$ and $y \neq \ell(S)$ and ${<}e, x, l, 2i{>} \in p(S)$ or
   (b) $y \in p(S)$ and $y = \ell(S)$ and ${<}e, x, l{>} \in K$

end of stage n

We first prove that $K \leq_T^p B$ via the reduction procedure T^B defined above. Given ${<}e, x, l{>}$, let $y = {<}e, x, l, 2^{|<e,x,l>|^2+1} - 1{>}$ and let n be such that $(v_{n-1})^{n-1} < |y| \leq (b(n))^n$. There are 2 cases. If $(v_{n-1})^{n-1} < |y| \leq (b(t-1))^{t-1}$ then part 2 applies to y, and to all strings in the tree $R_{<e,x,l>}$. In this case, the only possible element of the computation query tree of $T^B({<}e, x, l{>})$ which is put into B is y itself. So the computation $T^B({<}e, x, l{>})$ writes 0's on the ANSWER-tape for all of its queries until it queries y. By part 2 we have ${<}e, x, l{>} \in K \leftrightarrow y \in B \leftrightarrow$ the computation of $T^B({<}e, x, l{>})$ writes an odd number of 1's (exactly 1) on the ANSWER-tape $\leftrightarrow T^B({<}e, x, l{>})$ accepts.

In the second case $(b(t-1))^{t-1} < |y| \leq (v_n)^n$ and part 3 of the construction applies to y, and to all queries in the tree $R_{<e,x,l>}$. Now note that, since M_n runs in time $p_n(t) = t^n$, $\|S\| = \|Q(M_n, 0^{v_n})\| \leq (v_n)^n$. By choice of v_n, $(v_n)^n < 2^{(b(t-1))^{2(t-1)}-1} \leq 2^{|y|^2}$. Putting this together gives $\|S\| \leq 2^{|y|^2-1}$, so by the discussion at the beginning of the proof, the quantities $\ell(S)$ and $p(S)$ in part 3 are well defined. Then in part 3 we put elements of $p(S)$ into B or $\overline{B}$ in such a way that $T^B(<e,x,l>)$ follows the path $p(S)$ to the leaf $\ell(S)$. And by part 3b we have $<e,x,l> \in K \leftrightarrow \ell(S) \in B \leftrightarrow T^B(<e,x,l>)$ has an odd number 1's written on the ANSWER-tape $\leftrightarrow T^B(<e,x,l>)$ accepts.

Hence in either case the reduction works and we have $K \leq_T^p B$.

We now proceed to finish the proof via a series of three lemmata.

Lemma 3.31 *$W \in EXP$*

Proof: Since $\{0^{v_n}\}$ can easily be seen to be in P, given a string x we can check if $x = 0^{v_n}$ for some n. Then given 0^{v_n} we can determine if $0^{v_n} \in W$ by computing $M_n^B(0^{v_n})$ as follows.

i. Compute $Q(M_n, 0^{v_n})$

ii. For each $y \in Q(M_n, 0^{v_n})$, we determine if $y \in B$ using iii and iv below.

iii. If $|y| \leq (b(t-1))^{t-1}$ and for some e, x, l, i with $i < 2^{|<e,x,l>|^2+1}$, $y = <e,x,l,i>$, then we compute if $y \in B$ directly using parts 2 and 3 of the construction. To do this we compute the relevant $v_{n'}$ and $b(t'-1)$ which bound $|y|$ and, if needed, the corresponding $Q(M_{n'}, 0^{v_{n'}}), S, p(S), \ell(S)$ and deterministically compute if $<e,x,l> \in K$ if part 3b of the construction applies.

iv. If $(b(t-1))^{t-1} < |y| \leq (v_n)^n$ and if for some e, x, l, i with $i < 2^{|<e,x,l>|^2+1}$, $y = <e,x,l,i>$ then we use part 3 of the construction to decide if $y \in B$. Compute $S = Q(M_n, 0^{v_n}) \cap N_{<e,x,l>}$, $\ell(S)$ and $p(S)$. Note that in this case $y \neq \ell(S)$ since $\ell(S) \notin Q(M_n, 0^{v_n})$. So 3a of the construction applies and we have $y \in B \leftrightarrow y \in p(S)$ and $<e,x,l,2i> \in p(S)$.

It is straightforward to check that all of the above can be computed in *EXP* relative to $v_n = |0^{v_n}|$. The key point is that when, in step iii above, $<e,x,l> \in K$ is computed, we have $|<e,x,l>| < |y| \leq$

$(b(t-1))^{t-1}$. Now $<e,x,l> \in K$ can be determined in time $2^{2^{|<e,x,l>|}} \leq 2^{2^{(b(t-1))^{t-1}}} < 2^{v_n}$. Finally, once all of $Q(M_n, 0^{v_n})$ is determined, we have $0^{v_n} \in W \leftrightarrow M_n^B(0^{v_n}) = 0$. □

Lemma 3.32 *$B \in NEXP$*

Proof: The proof is very similar to that of Lemma 3.31. Given y, we want to compute if $y \in B$. We do this just as we determined in Lemma 3.31. The added complexity comes from the fact that here part 3b of the construction may apply. It may be that $y = \ell(S) \in p(S)$ and so $y \in B \leftrightarrow <e,x,l> \in K$ in this case. This puts B into *NEXP*. The other cases for deciding B can all be carried out in *EXP* as in Lemma 3.31 □

Lemma 3.33 *B is not $\leq_{tt}^p$ -complete for NEXP.*

Proof: If B were $\leq_{tt}^p$ -complete, we would have $W \leq_{tt}^p B$, say via reduction procedure M_n. But 0^{v_n} witnesses that this is not the case. □

□

Corollary 3.34 *There is a set C which is $\leq_T^p$ -complete for NE, but not $\leq_{tt}^p$ -complete for NE*

Proof: Again a simple padding argument works. Let B be as in Theorem 3.29. Define $C = \{x10^{|x|^2} \mid x \in B\}$. Since $B \in \mathrm{NTIME}\left(2^{n^2}\right), C \in NE, B \leq_m^p C$, and hence C is $\leq_T^p$ -complete for *NE*. Now assume C were $\leq_{tt}^p$ -complete for *NE*. Then $W \leq_{tt}^p C$, where W is the set defined in the proof of Theorem 3.29. Let M be a $\leq_{tt}^p$ -reduction which reduces W to C. Define a $\leq_{tt}^p$ -reduction M_0 as follows. If $M^C(x)$ generates the *tt*-condition $<<y_1, y_2, \ldots, y_k>, \alpha>$, then $M_0^B(x)$ generates the condition $<<z_1, z_2, \ldots, z_k>, \alpha>$, where $z_i =$ (if $y_i = z10^{|z|^2}$ then z else b) and b is a fixed element of $\overline{B}$. Clearly M_0 is a $\leq_{tt}^p$ -reduction and M_0 reduces W to B, contradicting the proof of Theorem 3.29. □

Watanabe [Wat87a, Wat87b] used Kolmogorov complexity to differentiate between $\leq_{tt}^p$ and $\leq_T^p$ -completeness for E. The proof of Theorem 3.29 would work for E as well and is a more direct construction than the previous one.

Chapter 4

Density of Complete Sets

4.1 Introduction

In this chapter we will examine more closely the density of complete sets. The density of a set A, is a function, f, such that $\|A^{\leq n}\| \leq f(n)$. i.e. the function bounds (from above) the number of strings in A of length n or less. Note that $f = 2^{n^2}$ always bounds the number of strings in *any* set.

The density of NP -$\leq_m^p$ -complete sets has been studied by Mahaney [Mah82]. His motivation though was somewhat different: he wanted to collect evidence for a conjecture in [BH77].

In 1977, Berman and Hartmanis raised a conjecture, nowadays referred to as the Berman-Hartmanis conjecture. They considered the polynomial time version of an old theorem by Cantor, Bernstein and Dedekind, later reproven by Myhill in a constructive setting: If A and B reduce to each other via $\leq_{1-1}^{REC}$ -reduction, then A and B are isomorphic in the sense that there exists a function f such that f is a recursive bijection between A and B. Myhill proved furthermore that all $\leq_m^{REC}$ -complete sets for RE are in fact $\leq_{1-1}^{REC}$ -complete and are thus via the previous theorem recursively isomorphic.

Berman and Hartmanis proved that if A and B are $\leq_m^p$ -reducible to each other by reductions that are length increasing and invertible in polynomial-time, then A and B are polynomial time isomorphic. In imitation of Myhill they continued to prove that all the known many-one complete sets for NP are in fact complete via reductions that are length increasing and polynomial-time invertible. Thus they proved for a whole bunch of problems that they are polynomial-time isomorphic. This "evidence" lead them to state their conjecture.

Note that the truth of the conjecture implies $P \neq NP$, since not all problems in P are polynomial time isomorphic: finite sets cannot be isomorphic to infinite ones, and on the other hand *all* sets in P are $\leq_m^p$ -complete for this class.

Mahaney considered the possibility of polynomial dense sets being NP -complete. These sets whose density is bounded from above by a polynomial are also called sparse. Mahaney then proved that if there would exist sparse $\leq_m^p$ -hard sets then $P = NP$. Thus giving evidence *in favor* of the conjecture.

At about the same time Karp and Lipton[KL80] considered the possibility of sparse $\leq_T^p$ -hard sets for NP. Their motivation was not the isomorphism conjecture, but the following. If SAT is computable in polynomial time, with the help of a small (polynomial size) table, though the table itself may be hard to compute, once computed one can solve instances of SAT quickly. This is equivalent, as we will see, to saying that SAT is $\leq_T^p$ -reducible to a sparse set. Their conclusion was similar to Mahaney's, but slightly weaker, for they proved if this was the case then $PH = \Sigma_2^p$.

They also proved for the complexity classes $PSPACE$ and EXP, that if they would posses sparse $\leq_T^p$ -hard sets then these would be a subclass of Σ_2^p.

Now putting Mahaney's and Karp-Lipton's result together gives us that EXP does not contain $\leq_m^p$ -hard sparse sets. For if it would, Mahaney's result implies $P = NP$ and Karp-Liptons result implies $EXP \subseteq \Sigma_2^p = P$ (this last equality because $P = NP$). This is in contradiction with the hierarchy theorems. Resulting in an absolute result about EXP.

Recent results along the line of Mahaney have been obtained by Ogiwara and Watanabe [OW90]. They proved that even for $\leq_{btt}^p$ -reductions NP cannot have sparse hard sets unless $P = NP$.

In this chapter we will first see how to generalize Mahaney's and Karp-Lipton's result to other densities; Section 4.2, then we will concentrate in Section 4.3 on the question which sets do not reduce to sparse sets. In Section 4.4 we will generalize again to other densities. The new work is taken from [BH92].

4.2 More than Polynomial Density

In order to generalize the Karp-Lipton result to other densities we will need other than polynomial time hierarchies. These hierarchies are most conveniently defined in terms of alternating Turing machine computations.

Definition 4.1 *For a function f, $\Sigma_i^f(\Pi_i^f)$ denotes the class of languages accepted by a $\Sigma_i(\Pi_i)$ alternating Turing machine which runs in time f(n). For a class of functions F, $\Sigma_i^F = \cup_{f\in F}\Sigma_i^f$ and $\Pi_i^F = \cup_{f\in F}\Pi_i^f$.*

Several important examples are,

1. Letting $F = \{p(n) \mid p \text{ is a polynomial}\}$ we obtain the usual levels of the polynomial-time hierarchy.

2. Letting $F = \{2^{p(n)} \mid p \text{ is a polynomial}\}$ we obtain the *EXP*-hierarchy, one of two exponential time hierarchies studied here. In terms of the more modern oracle notation, the first few levels of this hierarchy are $EXP = \Sigma_0^{EXP}$, $NEXP = \Sigma_1^{EXP}$, $NEXP^{NP} = \Sigma_2^{EXP}$, $NEXP^{NP^{NP}} = \Sigma_3^{EXP}, \ldots.$ The whole hierarchy is contained in *EXPSPACE*.

3. Letting $F = \{2^{cn} \mid c \in I\!N\}$ we obtain the E-hierarchy. Its first few levels are $E, NE, NE^{NP}, NE^{NP^{NP}}, \ldots.$

We will have occasion to consider classes between the levels of these two exponential hierarchies. These are commonly denoted by Δ classes defined by, for $F \in \{E, EXP\}$, $\Delta_0^F = F$, $\Delta_1^F = F^{NP}, \Delta_2^F = F^{NP^{NP}}, \ldots.$ So for all i, $\Sigma_i^F \cup \Pi_i^F \subseteq \Delta_i^F \subseteq \Sigma_{i+1}^F \cap \Pi_{i+1}^F$.

Another characterization of sets that $\leq_T^p$ -reduce to sparse sets are as sets that have polynomial advice or polynomial size circuits. We now briefly review the definitions of these non-uniform complexity measures. For more detailed definitions see Balcázar, Díaz and Gabarró [BDG88].

Definition 4.2 *An advice function is a function $f : \mathbb{N} \to \Sigma^*$. Let C be a complexity class and F be a class of advice functions. The class C/F is the collection of all sets A such that for some $B \in C$ and some $f \in F$, $A = \{x \mid <x, f(|x|)> \in B\}$.*

We will consider three nonuniform classes: *P/poly*, *EXP/poly* and *E/lin*. Here *poly* denotes the class of functions whose output length is bounded by some polynomial in the length of the input, and *lin* denotes the class of functions whose output length is bounded by some linear function in the length of the input.

We will be considering efficient reductions of complete sets to sets of different subexponential densities. The following definition specifies our measure of a set's density.

Definition 4.3 *Let f be a nondecreasing function, $f : \mathbb{N} \to \mathbb{N}$. A set $S \subseteq \Sigma^*$ is f-sparse if $\|\{x \in A \mid |x| \leq n\}\| < f(n)$ for all n.*
If F is a class of functions, we say S is F-sparse if S is f-sparse for some $f \in F$.

S is said to be *sparse* if S is $p(n)$-sparse, for some polynomial p. Informally, sets which, for some fixed k, are $n^{log^k n}$-sparse, will be called *almost sparse*. Since $n^{log^k n} = 2^{log^{k+1} n}$, we will use $n^{log^k n}, 2^{log^k n}$, and less formally, $2^{polylog}$ interchangeably. S is said to be *dense* if there is some constant $\epsilon > 0$ such that S is not 2^{n^ϵ}-sparse.

Familiarity with the standard definitions of Boolean circuit complexity is assumed here. A class C is said to have f-size circuits if for every set $A \in C$ there is a sequence $\{D_n\}$ of circuits which recognize A and such that for all n, D_n has size $\leq f(n)$.

First we will show some relations between the non-uniform complexity measures. They will be used in the sequel.

It is well-known that *P/poly* is equal to the class of problems which have polynomial-size circuits, or equivalently to the class of problems which are polynomial-time Turing reducible to some sparse set. There is a similar, though somewhat weaker, relationship concerning almost sparse sets which is captured in the following proposition.

Proposition 4.4 *Let A be a set with $2^{polylog}$-circuits. Then A is reducible to a sparse set in time $2^{polylog}$.*

For clarity and also for later use it is helpful to break the proof into two parts.

Part 1: If A has $2^{polylog}$-circuits then $A \in P/2^{polylog}$.
Proof: Let $\{C_n\}$ be an encoding of a circuit family recognizing A with $\|C_n\| \leq 2^{log^k n}$, for some fixed k. Define the advice function $f(n) = C_n$, so $|f(n)| \leq 2^{log^k n}$. Clearly, for x of length n, $x \in A$ if and only if x is accepted by the circuit coded by $f(n)$. Since the circuit value problem is in P, whether $x \in A$ can be decided using the advice function $f(n)$ and the polynomial-time predicate for the circuit value problem. $\square$

Part 2: If $A \in P/2^{polylog}$ then A is reducible to a sparse set via a $2^{polylog}$-time Turing reduction.
Proof: Assume A is in the class $P/2^{polylog}$ via the advice function f and the set $B \in P$. Define $S = \{<x, 0^n> \mid x \text{ is a prefix of } f(n)\}$. As there are at most k^2 many elements in S of length $\leq k$, S is sparse. Now, given x, a straightforward prefix search can be carried out using S as an oracle to obtain $f(|x|)$ in $n^{log^k n}$ steps, for some fixed k. Having done this, we have $x \in A$ if and only if $<x, f(|x|)> \in B$, and this last can be determined in time $2^{polylog}$ relative to $|x|$. $\square$

Let's now recall the statement of Karp-Lipton: If $SAT \leq_T^p S, S$ sparse, then $PH \subseteq \Sigma_2^p$. Seen in the light of the previous discussion, the theorem can be stated with polynomial size circuits or polynomial advice reaching the same conclusion: $PH \subseteq \Sigma_2^p$.

It is now reasonable to ask whether these results can be extended to higher densities than polynomial, for if we would have a slightly bigger than polynomial size table where we could look up the information needed to solve instances of SAT quickly, the analogy for practical purposes still holds, for we have to do this computation only once.

We will give evidence that even this is not likely to be the case unless the *Exponential* Hierarchy collapses.

We already mentioned the recent developments of Ogiwara and Watanabe stretching the theorem of Mahaney up to $\leq_{btt}^p$ -reductions. Although our main concern in this section will be $\leq_T^p$ -hard sets, we will derive the following proposition using the extended version of the theorem by Homer and Longpré [HL91]

Proposition 4.5 *NP has no $\leq^p_{btt}$ -hard set S with $\|S^{\leq n}\| \leq n^{\log^k n}$ unless*

- $NP \subseteq \bigcup_{t>0} \mathrm{DTIME}\left(n^{\log^t n}\right)$ *and*
- $EXP = NEXP$.

Proof: The main theorem (Theorem 5) of Homer and Longpré, when applied to btt reductions, states that, if $NP \leq^p_{btt} S$ with $\|S^{\leq n}\| \leq n^{log^k n}$ then $NP \subseteq \bigcup_{t>0} \mathrm{DTIME}\left(4^k k! n^{log^{kt} n} n^k\right) = \bigcup_{t>0} \mathrm{DTIME}\left(n^{\log^{kt} n}\right)$. The following padding argument now yields $EXP = NEXP$.

Let $A \in NEXP$, say $A \in NTIME(2^{n^l})$. Define $B = \{x0^{2^{|x|^l}} \mid x \in A\}$. Then $B \in NTIME(n) \Rightarrow B \in DTIME(n^{log^u n})$ for some u. But then $A \in DTIME((2^{n^l})^{log^u 2^{n^l}}) = DTIME((2^{n^l})^{n^{ul}}) = DTIME(2^{n^{l+ul}})$ $\subseteq CEXP$ as needed. □

We are now ready to generalize the Karp-Lipton results to sets with higher density. We show that if an NP -complete set has circuits of size $2^{polylog}$ then the exponential time hierarchy collapses to the second level.

Theorem 4.6 *NP -complete sets do not have circuits of size $2^{polylog}$ unless the exponential hierarchy collapses to $NEXP^{NP}$.*

Proof: Assume that a $2^{polylog}$-size family of circuits exists for SAT. Then, by Proposition 4.4 , SAT is reducible to a P-sparse set in time $n^{log^k n}$, for some fixed k. In fact, we will prove our result starting from the slightly weaker hypothesis that SAT is reducible to a set S of density $n^{(log^k n)}$ in time $n^{log^k n}$, for some k. And from this weaker hypothesis we will obtain the hierarchy collapse.

The proof uses techniques originated by Karp and Lipton [KL80] and Balcázar [Bal90]. The main idea is to use a $NEXP^{NP}$-machine to find a large initial segment of S and thus a large initial segment of SAT. We then show how a $NEXP^{NP}$-machine with a large initial segment of S can simulate a $NEXP^{NP^{NP}}$-computation, collapsing the hierarchy.

Let A be a set in $NEXP^{NP^{NP}}$ and let M_j be a $NEXP$-machine such that $A = L(M_j)$ with a NP^{NP}- oracle. Furthermore assume that M_j runs in time 2^{n^l}. We view queries to a NP^{NP}-oracle as queries to a NP machine M_1 that on its turn is allowed to make queries to SAT. Since the running time of M_j is bounded by 2^{n^l} it can query strings to M_1 of length at most 2^{n^l}. We know that M_1 has a running time bounded by a polynomial it follows that M_1 queries SAT strings of length at most 2^{pn^l} for some p.

In order to be able to simulate M_j with a $NEXP^{NP}$- machine we must compute SAT up to length 2^{pn^l}. We know that SAT is reducible to S in time $n^{log^c n}$, where the density of S is $n^{\log^k n}$, for some fixed k. Let M_c witness the $n^{log^c n}$ reduction from SAT to S and assume that M_c runs in time $n^{log^c n}$. So if we know the elements of S up to length $2^{p^2 n^{l+cl}}$ we're done. There are at most $n^{\log^k n}$ strings in S of length n so there are at most $2^{p^{lk+2} n^{(l+cl)(1+k)}}$ strings of length $2^{p^2 n^{l+cl}}$ in S, which is $\leq 2^{n^i}$ for some i. Now define the string z by,

$$z = x_1 \# x_2 \# \ldots \# x_{2^{n^i}} \text{ where } x_i \in S \ (i = 1 \ldots 2^{n^i})$$

z codes S up to the desired length and the length of z is less than $2^{n^{2i}}$. The idea now is to first find z and than replace machine M_1 that queries SAT by an NP machine that makes no such queries. First we will show that if z is known than there exists a NP machine $M_{1'}$ such that $x \in L(M_1, SAT) \Leftrightarrow <x, z> \in L(M_{1'})$. Machine $M_{1'}$ behaves as follows:

```
input $<x, z>$
nondeterministically simulate $M_1$ on input $x$
if $M_1$ queries a string $y$ to $SAT$ do the following:
  simulate machine $M_c$ on input $y$.
  if $M_c$ queries a string $y'$ to $S$ then
    run through $z$ and find out if $y'$ is in $z$
  if $y'$is in $z$ proceed the simulation of $M_c$
      with a 1 written on the ANSWER-tape
  otherwise proceed the simulation of $M_c$
      with a 0 written on the ANSWER-tape
if $M_c$ accepts continue the simulation of $M_1$
  with a 1 written on the ANSWER-tape
otherwise proceed with a 0 written on the ANSWER-tape
```

If z is correct, i.e. it is a coding of an initial segment of S, then $M_{1'}$ accepts $<x, z>$ iff $x \in L(M_1, SAT)$. Furthermore for $M_{1'}$ to answer the queries y made by M_1 to SAT, $M_{1'}$ only needs to simulate the computations $M_c(y)$ where oracle queries are answered via a search through z which is linear in the length of z. So $M_{1'}$ is an NP-machine.

We now use an EXP^{NP}-computation to find a string z' which, when used by M_c in place of the S-oracle, computes SAT. Of course, the z defined above would work as z', but we may find a different sufficient z' as well. Once a z' is found, we use it to simulate M_j with M_1 replaced by $M_{1'}$. We take advantage of the fact that SAT is self-reducible.

Let M_{self} be an oracle Turing machine that, when equipped with a SAT oracle, witnesses the self-reducibility of SAT. More precisely, M_{self} works as follows. For an input u, $M_{self}(u)$ computes by generating two queries u_1 and u_2, both shorter than u, with the property that $u \in SAT$ iff $u_1 \in SAT$ or $u_2 \in SAT$. In line (*) of the simulation below, we write that $M_{self}(u)$ accepts with the help of $M_c^{z'}$ to mean that $M_{self}(u)$ computes and generates the two oracle queries u_1 and u_2 and then either $M_c^{z'}(u_1)$ or $M_c^{z'}(u_2)$ accepts. Recall that $n^{log^c n}$ bounds the running time of M_c.
The simulation works as follows:

```
input $x$
guess a string $z'$ of length $2^{n^{2i}}$ and pad $z'$ with 0's so that its
length is $2^{n^{4ci}}$.
query the $NP$-oracle: Is there a string $u, |u| \leq 2^{n^i}$ such that
    (*) $M_{self}(u)$ accepts with the help of $M_c^{z'} \Leftrightarrow M_c^{z'}(u)$ rejects?[1]
if a 1 is written on the ANSWER-tape then REJECT (wrong $z'$)
if a 0 is written on the ANSWER-tape then:
simulate $M_j$ on input $x$
if $M_j$ queries a $y$ to its oracle then query $<y, z'>$ to $M_{1'}$
ACCEPT iff $M_j$ accepts
```

Since M_c runs in time $n^{log^c n}$, the above query to the NP-oracle can be checked to be an NP-computation relative to the length of its input (which is $2^{n^{4ci}}$). So the above simulation can be done in EXP^{NP}. The only thing remaining to show is that the NP oracle correctly decides

[1]Note that $M_c^{z'}(x)$ means that for every query from M_c, a 1 will be written on the ANSWER-tape if it is in z' and a 0 otherwise.

if the z' gives answers which cause the oracle machine M_c to correctly compute SAT. Line $(*)$[2] of the above simulation ensures that this is so.

Note that the z' used in the above simulation may not code exactly the same strings as z but serves the same purpose from the point of view of M_c's computations. $\square$

Note: In the above theorem, since we show that $\Sigma_3^{EXP} \subseteq \Sigma_2^{EXP}$, we actually have that $\Sigma_2^{EXP} = \Pi_2^{EXP}$. This same comment applies to the other results in this section.

Corollary 4.7 *If NP has a $\leq_T^p$ -hard set S, and $\|S^{\leq n}\| \leq 2^{polylog}$, then the EXP hierarchy collapses to $NEXP^{NP}$.*

Proof: The proof of Theorem 4.6 yields the desired conclusion from the hypothesis that the reduction to the almost sparse set S was possible in time $2^{log^k n}$. This is weaker than our assumption here that the reduction is in $PTIME$. $\square$

In Theorem 4.6, no assumption was made concerning the almost sparse set S. Using methods of Kadin[Kad87] we obtain a better collapse of the exponential hierarchy if we assume that the set S is in NP. In the next theorem $EXP^{NP}[poly]$ refers to the class of problems computable by an exponential time oracle Turing machine where, for any input x, the number of oracle queries is bounded by a fixed polynomial in $|x|$. Hemachandra[Hem87] has shown that $EXP^{NP}[poly] = P^{NE}$

Theorem 4.8 *NP has no $\leq_T^p$ -complete set S with $\|S^{\leq n}\| \leq n^{log^l n}$ unless the exponential hierarchy collapses to $EXP^{NP}[poly]$*

Proof: Let A be a set in $NEXP^{NP}$ such that $A = L(M_0, SAT)$, for some $NEXP$ machine M_0. We construct a machine M_1 from M_0 such that $L(M_1, S) = L(M_0, SAT)$. M_1 is basically the same machine as M_0. M_1 nondeterministicly simulates M_0 until M_0 queries a string to SAT. Instead of querying the string to SAT, M_1 uses the $\leq_T^p$ -reduction from SAT to S to answer the query to SAT with queries to S. Note that M_1 only queries S up to strings of length 2^{n^k} for some k.

[2]For a proof of this see Theorem 7.19, chapter 7.

The idea now is to generate two NP machines, M_{no} and M_{yes} which accept $\overline{S}$ and S respectively, up to length 2^{n^k}. Once defined, an $EXP^{NP}[poly]$ machine can generate a machine $M_2(x)$, using oracle substitution, that accepts iff $M_1^S(x)$ accepts. Now the $EXP^{NP}[poly]$ machine queries its NP oracle one more time to decide if $M_2(x)$ accepts.

M_{yes} is the NP machine that accepts S, since S is in NP. M_{no} is a machine that on input $<0^i, 1^j, x>$ guesses i strings of length at most j, verifies that the strings are in S, and accepts iff x is not one of the i guessed strings. If $|x| \leq j$ and $i = (Census)_S(j)$, then M_{no} accepts $<0^i, 1^j, x>$ iff $x \in \overline{S}$.

The $EXP^{NP}[poly]$ machine needs to be able to compute the census of S up to length 2^{n^k}. There are at most $n^{log^l n+1}$ strings of length $\leq n$ in S so the census of S up to 2^{n^k} is bounded by $2^{n^{2lk}}$. Using binary search only n^{2lk} queries are needed to compute the census of S. □

Using a similar, but slightly more complicated, argument we can apply our methods to $PSPACE$.

Theorem 4.9 *PSPACE does not have a $\leq_T^p$ -hard set S with $\|S^{\leq n}\| \leq 2^{polylog}$ unless EXPSPACE $\subseteq$ NEXP^{NP}.*

Proof: Let A be a set in $EXPSPACE$ via a machine M_A using no more than $2^{p(n)}$ space. we have to show that $A \in \Sigma_2^{EXP}$. Consider the set $A' = \{<x, 0^{2^{p(|x|)}-|x|}> \mid x \in A\}$. A' is in $PSPACE$. Under the assumption that $PSPACE$ has a $\leq_T^p$ -hard set S with $\|S^{\leq n}\| \leq 2^{log^k n}$ we can show that A' is in $\Sigma_2^{2^{polylog}}$. To do this we use the same approach as in the previous theorem. The only difference is that we use a different self-reduction. We consider Quantified Boolean Formulas, a $PSPACE$-complete problem. QBF is self-reducible. To show this observe that if we encounter a $\forall x_1 Q_{x_2} \ldots Q_{x_n} \phi(x_1, x_2, \ldots, x_n)$ formula the self-reduction queries $Q_{x_2} \ldots Q_{x_n} \phi(x_1 = 0, x_2, \ldots, x_n)$ and $Q_{x_2} \ldots Q_{x_n} \phi(x_1 = 1, x_2, \ldots, x_n)$ to its oracle and accepts iff both are in QBF. Where Q_x is either a universal or a existential quantifier and $x = 1(0)$ means everywhere in the formula x has been replaced by true (false). If we encounter a existential quantifier the self-reduction becomes $Q_{x_2} \ldots Q_{x_n} \phi(x_1 = 0, x_2, \ldots, x_n)$ or $Q_{x_2} \ldots Q_{x_n} \phi(x_1 = 1, x_2, \ldots, x_n)$ and accepts if at least one of the queries is in QBF. Let M_c be the polynomial time machine witnessing the reduction from QBF to S. A $\Sigma_2^{2^{polylog}}$ machine guesses a string z of appropriate length, uses its NP-oracle to check that it is a correct z and accepts iff M_c^z accepts. Now we have that $A' \in \Sigma_2^{2^{polylog}}$ from this it follows that A is in Σ_2^{EXP}. $\square$

4.3 Small Circuits and the EXP-Time Hierarchy

So far we proved that NP probably does not have sets with $2^{polylog}$ -dense sets. Let's now first take a step back again to polynomial dense sets and try to figure out which sets do *not* reduce $\leq_T^p$ to P-dense sets.

Part of the answer to this question is given by Kannan [Kan82], who proved that any level of the exponential hierarchies above the Δ_1 level (that is E^{NP} and EXP^{NP}) is not in $P/poly$.

Chris Wilson, in [Wil85], constructs an oracle relative to which E^{NP} has polynomial-size circuits. So to extend Kannan's result to E^{NP} and EXP^{NP} is probably quite difficult.

In the following we will show that there is evidence for these classes not being in $P/poly$ (or even in $EXP/poly$). We prove that if there are polynomial-size circuits for EXP^{NP} then the bottom levels of the EXP hierarchy collapse into the polynomial hierarchy. From this the same result for the E-hierarchy and several others results follow.

In light of the results of Section 4.2, it is also reasonable to consider the implications of $2^{polylog}$-size advice for exponential classes as well. We postpone these considerations until the next section, as they involve alternating Turing machines with computations of length $2^{polylog}$ which is studied there.

Let $\{M_e\}$ be an enumeration of the problems in $NEXP$. Define the following set U:

$$U = \{<e, x, t, j, b> | M_e(x) \text{accepts in} \leq\ t \text{ steps and the } j^{th} \text{ bit} \\ \text{of the leftmost accepting path is } b\}$$

We begin with two lemmata which show that U is EXP^{NP} -complete.

Lemma 4.10 *$U \in EXP^{NP}$*

Proof: To determine if $<e, x, t, j, b>$ is in U we use a prefix search to determine the first j bits of the leftmost accepting computation of $M_e(x)$. We then ask if the j^{th} bit is a b. The following algorithm computes the leftmost path via an EXP^{NP}-computation.

```
Input $<e, x, l, j, b>$.
Set $z = \lambda$ (the empty string).
While $|z| \leq j$,
determine if there is an accepting computation of $M_e(x)$ whose first
$|z| + 1$ bits are $z0$. (This is done by a query to the NP oracle.)
If yes, then set $z = z0$. Otherwise, see if there is an accepting compu-
tation of $M_e(x)$ whose first $|z| + 1$ bits are $z1$. If so, set $z = z1$.
```

After concluding this algorithm, z is equal to the first j bits of the leftmost accepting path of $M_e(x)$, if the computation accepts. Otherwise $z = \lambda$. We can now test if the j^{th} bit of z is b. If so, we accept. Clearly, this algorithm can be done in EXP^{NP} as needed. $\square$

Lemma 4.11 *If U is in EXP then $EXP = EXP^{NP}$.*

Proof: Let M be a Turing machine witnessing the computation of a problem L in EXP^{SAT}. (Without loss of generality, we assume the NP-oracle is SAT.) So M runs in exponential time and has access to a SAT-oracle. The idea is to convert M from an oracle machine to an $NEXP$ machine without an oracle. Then, using U we find the leftmost accepting path through this $NEXP$-machine in exponential time. From the leftmost path we are able to read off whether or not the original machine M accepted. Now given an input x to M, consider the following $NEXP$-computation.

Carry out M's computation on x substituting the following nondeterministic procedure for all oracle calls. When a query z to SAT is reached, nondeterministically guess a possible satisfying assignment for the formula z. Consider these guesses as ordered lexicographically. So the leftmost (that is lexicographically least) assignment assigns 0's to each variable and the rightmost assignment assigns 1's to each variable. There are three cases:

1. If the guessed assignment satisfies z then continue on with M's simulation with a 1 written on the ANSWER-tape.

2. If the assignment fails to satisfy z and the assignment is not all 1's (that is, not the rightmost) then the computation halts and rejects.

3. If the guessed assignment fails to satisfy z and is the rightmost assignment then continue on with M's simulation with a 0 written on the ANSWER-tape.

Finally, when M during the above simulation halts either accepting or rejecting, the simulation does one final nondeterministic step (branching either left or right) and then halts. The left (right) branch halts in an accepting state if and only if M halts and rejects (accepts).

Note that the above simulation is in *NEXP* since the length of M's computation is exponential and each oracle query z becomes an *NP*-computation, but on a (possibly) exponential length formula (in $|x|$). For any input x, the *NEXP* computation always has an accepting path and furthermore the leftmost accepting path reflects a computation of M on input x with the correct answers to the oracle queries. (To see this note that when an oracle query is reached, there is always a guessed assignment to the variables of the query which results in the simulation of M continuing along this path. Either the query is satisfiable, in which case (by 1. above) the leftmost path will be a satisfying assignment and M's computation will continue on with a 1 written on the ANSWER-tape, or the query is not satisfiable, in which case (by 3. above) the leftmost path will be the assignment of all 1's to the queries variables and the computation will continue with a 0 written on M's ANSWER-tape.)

Now, in a straightforward manner, we can use an exponential sequence of queries to U to determine the leftmost accepting path of the above *NEXP* simulation of $M^{SAT}(x)$. Since, by assumption U is in *EXP*, this path can be found in exponential time. Moreover, this path reflects the true computation of $M(x)$ with a *SAT*-oracle. From this leftmost path we can immediately read off whether or not M accepts x by seeing if the last bit of the path is a 1, in which case M accepts, or a 0, in which case M rejects. □

Theorem 4.12 *If EXP^{NP} is in $EXP/poly$ then $EXP^{NP} = EXP$.*

Proof: By Lemma 4.11 it is sufficient to prove that $U \in EXP$. The assumption entails the existence of a function h such that $|h(n)| \leq p(n)$ (for some fixed polynomial $p(n)$) and a set B in EXP such that $x \in U$ iff $<x, h(|x|)> \in B$. The idea of the proof is to cycle through all of the possible advice strings for U, using the structure of U to find the correct advice in exponential time. The following algorithm recognizes U in deterministic time 2^{poly}.

```
input$<e, x, t, j, b>$ (let $n = |<e, x, t, j, b>|$)
    foreach string $y'$ with $|y'| \leq p(n)$ do
      let $z = \lambda$
      repeat until $|z| > t$
        if$<<e, x, t, |z|, 0>, y'> \in B$ then $z = z0$
          else $z = z1$
      end repeat
      check whether $z$ is an accepting computation of $M_e(x)$
      if so, store $z$
    end for
    find the smallest $z$ among all the stored $z's$ if one exists
    accept iff the $j^{th}$ bit of this $z = b$
```

□

Recall that the Δ-levels of the EXP-hierarchy are defined to be $\Delta_i^{EXP} = EXP^{NP^i}$, where NP^i is a tower of i NP's. The next corollary says that Theorem 4.12 applies to any Δ-level of the hierarchy.

Corollary 4.13 *If Δ_i^{EXP} in $EXP/poly$ $(i \geq 1)$ then $\Delta_i^{EXP} = EXP$.*

Proof: For $i = 1$ this is a restatement of Theorem 4.12 since $\Delta_1^{EXP} = EXP^{NP}$. We show that it is also true for Δ_2^{EXP}. A straightforward inductive argument (omitted here) completes the proof. The basic observation is that the proof of Theorem 4.12 relativizes. So if $\Delta_1^{EXP^{SAT}}$ is in $EXP^{SAT}/poly$ then $\Delta_1^{EXP^{SAT}} = EXP^{SAT}$. But $\Delta_1^{EXP^{SAT}} = \Delta_2^{EXP}$ and $EXP^{SAT} = EXP^{NP} = \Delta_1^{EXP}$. So if Δ_2^{EXP} is in $EXP/poly$ then

Δ_1^{EXP} is also in $EXP/poly$. Another application of Theorem 4.12 now yields $\Delta_2^{EXP} = EXP$. $\Box$

Next we show that if EXP^{NP} has polynomial size circuits, the EXP hierarchy collapses into the polynomial hierarchy.

Corollary 4.14 *If EXP^{NP} is in $P/poly$ then $EXP^{NP} = \Sigma_2^P \bigcap \Pi_2^P$*

Proof: Since $P/poly \subseteq EXP/poly$, Theorem 4.12 yields $EXP^{NP} = EXP$. From Karp and Lipton[KL80] we know that if $EXP \subseteq P/poly$ then $EXP = \Sigma_2^P \bigcap \Pi_2^P$. $\Box$

This same result can now be proved for the E-hierarchy.

Corollary 4.15 *If E^{NP} is in $P/poly$ then $E^{NP} = \Sigma_2^p \bigcap \Pi_2^p$*

Proof: We will prove that if $E^{NP} \subseteq P/poly$ then $EXP^{NP} \subseteq P/poly$. The result then follows from the previous corollary. So let $A \in EXP^{NP}$, say $A \in DTIME(2^{n^t})$. We will show that $A \in P/poly$. Define $B = \{x0^{(|x|^{n^t} - |x|)} \mid x \in A\}$. It is straightforward to check that $B \in E^{NP}$ and so, by assumption, $B \in P/poly$, say via advice function h and polynomial-time predicate D. Now define the polynomial-length bounded advice function h' by, for any integer n, $h'(n) = h(n^t)$ and define a polynomial-time predicate D' by, on input $<w, n>$, D' carries out the computation of D on input $<w0^{(|w|^t - |w|)}, n>$. Then for any z, $z \in A \leftrightarrow w = z0^{|w|^t - |w|} \in B \leftrightarrow <w, h(|w|)> \in D \leftrightarrow <z, h'(|z|)> \in D'$. So A is in $P/poly$ as claimed. $\Box$

Using our results we can now prove that NE^{NP} does not has P-size circuits. This result was proved earlier by Kannan[Kan82] using different methods.

Corollary 4.16 *NE^{NP} does not have P-size circuits.*

Proof: Suppose NE^{NP} does have P-size circuits. Then by Karp and Lipton [KL80] we have that the PH collapses to $\Sigma_2^P \bigcap \Pi_2^P$. From this an easy padding argument yields that EH collapses to $\Sigma_2^{EXP} \bigcap \Pi_2^{EXP}$. So in particular $\Delta_3^{EXP} \subseteq \Sigma_2^{EXP}$. Together with Corollary 4.13 and Corollary 4.14 this would imply that $\Sigma_2^{EXP} \subseteq \Sigma_2^P$. Since this violates the hierarchy theorem for alternating Turing machines we have a contradiction. □

A consequence of these results is that we have a very strange situation under the assumption that EXP^{NP} has P-size circuits. Namely, the EXP-hierarchy collapses into two levels:

- EXP^{NP} collapses to $EXP = \Sigma_2^P \cap \Pi_2^P$.
- EH collapses to $NEXP^{NP}$.

This seems a strange, yet not contradictory, conclusion.

The next result proves a similar theorem for the E-hierarchy. It is actually a corollary to the proof of Theorem 4.12. Recall that E/lin is the set of problems computable in $DTIME(2^n)$ with a linear size amount of advice. The class E/lin is not very natural, but the result allows us to link our methods to two earlier lines of research.

Corollary 4.17 *If $E^{NP} \subseteq E/lin$ then $E^{NP} = E$.*

Proof: The proof is exactly the same as that of Theorem 4.12 except that EXP is replaced by E and $NEXP$ by NE throughout. Some simple algebra, together with the assumption that the class is contained in E/lin (rather than $EXP/poly$) allows us to achieve the collapse to E. □

The results in this section relate to some of those obtained by Allender and Watanabe in [AW90] . There they considered a property Q which posits that for every honest function $f : \Sigma^* \to 0^*$ there is a polynomial-time computable weak inverse g such that for all $x \in f(\Sigma^*), f(g(x)) = x$. They prove that property Q is equivalent to the property of NE-computations that, "Every NE predicate is E-solvable". Meaning that, given an NE-predicate R, there is an E-computable function which, for any input x to R, computes a witness to $R(x)$, if one exists. Allender and Watanabe (Proposition 2, [AW90]) proved that $E = E^{NP} \Rightarrow$Q$\Rightarrow E = NE$. It is not known if any of these arrows are reversible.

Our results, when looked at in a similar light, can be seen as running parallel to theirs. Consider the property Q' stating that for every honest function $f : \Sigma^* \to 0^*$, the function g which computes, for any $x \in f(\Sigma^*)$, the least y such that $f(y) = x$, is polynomial time computable. Using the techniques in this section one can show that Q' is equivalent to the property that the leftmost path through an NE-computation can be found in E. Given this, Lemmas 4.10 and 4.11 above imply that $E = E^{NP} \Leftrightarrow Q'$. The proof of Theorem 4.12 actually shows that if $E^{NP} \subseteq E/lin$ then Q'. So our results, when stated in this fashion, say that $E^{NP} \subseteq E/lin \Rightarrow Q' \Leftrightarrow E = E^{NP}$.

There is also a relationship between our results and the existence of sparse sets in the polynomial hierarchy. One of the main results in the paper of Hartmanis, Immerman and Sewelson [HIS85] is that $E = NE$ if and only if there are no sparse sets in $NP - P$. This result, together with our theorem, immediately yields,

Corollary 4.18 *If $E^{NP} \subseteq E/lin$ then there are no sparse sets in $\Delta_1^P - P$.*

4.4 The Almost-Polynomial Time Hierarchy

On examining the central proofs of the previous two sections, it becomes apparent that while the results are stated about the exponential hierarchy, they could equally well be applied to the subexponential hierarchy based on $2^{polylog}$-length alternating computations. Results concerning this hierarchy are actually stronger in the sense that they imply the previous results concerning the EXP-hierarchy (usually via a simple padding argument). We chose to state the results for classes in the exponential hierarchy as these classes have previously been extensively studied. In this section we restate our main results for $2^{polylog}$ length computations and obtain from them new results concerning $PSPACE$ and EXP.

Let PL denote the class of $2^{polylog}$ functions. That is $PL = \{2^{p(log n)} \mid p \text{ is a polynomial}\}$. As before, we consider the hierarchy based on alternating PL length computations, whose levels are Σ_i^{PL}, Π_i^{PL} and Δ_i^{PL} (for all i). This hierarchy is not as central or well-studied as the exponential hierarchy. Nonetheless the PL-hierarchy has some interesting properties. First of all there is a natural relationship between this hierarchy and the exponential hierarchy. In essence the *PL*-hierarchy plays the same role for the *EXP*-hierarchy as the polynomial hierarchy does for the *E*-hierarchy. The relationship between the polynomial-time and the *E*-hierarchy was extensively studied by Hartmanis, Immerman and Sewelson in [HIS85] and one can obtain analogous results for the $2^{polylog}$ vs. the *EXP*-hierarchy. For example, there are no sparse sets in $\Sigma_1^{PL} - PL$ if and only if $\Sigma_1^{EXP} = EXP$. Furthermore, possibly unlike the exponential hierarchy, this hierarchy obeys downward separation: if $\Sigma_{i+1}^{PL} \subseteq \Sigma_i^{PL}$ then for all $j > i, \Sigma_j^{PL} \subseteq \Sigma_i^{PL}$.

The following two results concerning the *PL*-hierarchy are proved exactly as the previous results (Theorem 4.6 and Corollary 4.7) in Section 4.2 about the exponential hierarchy. In each case, we use arithmetic properties of $2^{polylog}$ functions in place of those of *EXP*-functions. We omit the proofs here.

Theorem 4.19 *NP -complete sets do not have circuits of size $2^{polylog}$ unless the $2^{polylog}$ hierarchy collapses to Σ_2^{PL}.*

Corollary 4.20 *If NP has a $\leq_T^p$ -hard set S, and $\|S^{\leq n}\| \leq 2^{polylog}$ then the $2^{polylog}$ hierarchy collapses to Σ_2^{PL} and PH $\subseteq \Sigma_2^{PL}$.*

In Section 4.2, we investigated the implications of there being almost-polynomial size circuit families for *NP* and *PSPACE*. In Section 4.3, we considered polynomial size circuits for the *EXP*-hierarchy, and in particular for EXP^{NP}. Having introduced the *PL*-hierarchy, we can now bring these ideas full circle and prove the next result which investigates whether EXP^{NP} has almost-polynomial size circuits. Again, by results of Kannan [Kan82], any class in the *EXP*-hierarchy above EXP^{NP} is known not to have such circuits. As a corollary, our theorem yields a new proof of this fact.

Theorem 4.21 *If EXP^{NP} has circuits of size $2^{polylog}$ then $EXP^{NP} \subseteq \Sigma_2^{PL}$.*

Proof: Without loss of generality we assume that a Turing machine has only 1 tape and 1 head and accepts by entering a unique accepting state. By a configuration we mean a binary string c coding the state,

the tape contents and the place of the head on the tape. For a *NEXP* machine the length of a configuration c is always bounded by $2^{p(n)}$ for some fixed polynomial p. A legal transition from configuration c_i to configuration c_j (with respect to some Turing machine $M_e(x)$) is one where the state, the tape contents and the place of the head change accordingly M_e's transition relation. Note that in order to check one bit of c_j one only needs to consider a constant number of bits of c_j and c_{j-1}. For a more detailed description of this see, for example, [BDG88]. As in the proof of Theorem 4.12 we need to define a complete set for EXP^{NP}. We use almost the same set U as in Theorem 4.12. Define U' to be the following set:
$U' = \{<e, x, t, j, k, b> \mid$ the k^{th} bit of the j^{th} configuration of $M_e(x)$'s leftmost accepting path of length $\leq t$ equals $b\}$

A same sort of argument as used in Lemma 4.10 shows that U' is in EXP^{NP} and a simple $\leq_m^p$ -reduction from U to U' shows that U' is complete for EXP^{NP}. Under the assumption that U' has a circuit family of size $2^{polylog}$ we show that it is in Σ_3^{PL}. Together with Theorem 4.19 we then have that $EXP^{NP} \subseteq \Sigma_3^{PL} \subseteq \Sigma_2^{PL}$.

Claim 4.22 *If U' has a circuit family of size $2^{polylog}$ then U' is in Σ_3^{PL}*

Proof: We provide a Σ_3^{PL} algorithm for U'. We first show how to determine, via an *NP* computation, if a circuit z recognizes the bits in a sequence of configurations comprising an accepting path in a *NEXP*-computation. In the following description of the computation of an *NP*-machine, the input z represents a circuit, possibly recognizing U'. So z takes inputs of the form $<e, x, t, j, k, b>$.
Input z, e, x. Accept if,

1. There is a place in the first configuration, as determined by bits of this configuration which are decided by inputs of z, which incorrectly represents the initial configuration of $M_e(x)$, or

2. there is an integer i (the number of a configuration) and a position k in configuration i, such that the k^{th} bit in configuration i, as determined by z, does not follow from the corresponding bits in the $(i-1)^{th}$ configuration (again as determined by z) via a legal transition of $M_e(x)$, or

3. the last configuration of $M_e(x)$ contains a state other than the accepting state.

Since M_e is a *NEXP*-machine, in 2 above the integers i and k have length polynomial in the length of x. Hence the above computation can be carried out by an *NP*-oracle. This oracle will reject z, e, x iff z correctly determines the bits in a sequence of configurations representing some accepting computation of $M_e(x)$.

Now we have to check for a leftmost computation. The above procedure ensures that the computation is correct but it might not be the leftmost. In order to check that a z codes the leftmost computation we query a NP^{NP}-oracle whether there exists a z' that also codes a correct computation and codes a computation that is to the left of the computation that z codes. More precisely:

```
input $z, e, x$,
    guess a $z' \neq z$
    check that $z'$ codes a correct accepting computation of $M_e(x)$
    (This uses a query to an $NP$-oracle.)
    guess a place in the computation of $M_e(x)$ where $z'$ codes a left
    branch hand $z$ a right. Furthermore check that $z$ and $z'$ code the
    same computation up to this point.
    The last check requires again an $NP$-query.
    If all of this holds accept, else reject.
```

Combining the two algorithms we obtain a Σ_3^{PL}-algorithm for U':

```
input$<e, x, t, j, k, b>$
    guess a $z$
    check that $z$ codes a correct accepting computation of $M_e(x)$
    (This check needs a query to a $NP$-oracle.)
    check that $z$ codes the leftmost accepting computation of $M_e(x)$
    (This check needs a query to a $NP^{NP}$-oracle.)
    Now use $z$ to determine if $<e, x, t, j, k, b> \in U'$ and accept
    accordingly
```

□

Now we apply Theorem 4.19 to obtain the inclusion in Σ_2^{PL}. □

Corollary 4.23 ([Kan82]) *$NEXP^{NP}$ does not have circuits of size $2^{polylog}$.*

Proof: The main observation is again that the above proof relativizes. So if $EXP^{NP^{NP}}$ has circuits of size $2^{polylog}$ then it collapses to Σ_2^{PL}. Furthermore by Theorem 4.6 we have that if $NEXP^{NP}$(and thus NP) has circuits of size $2^{polylog}$ then the exponential hierarchy collapses to $NEXP^{NP}$. So $\Sigma_2^{EXP} = NEXP^{NP} = EXP^{NP^{NP}} \subseteq \Sigma_2^{PL}$. But this is in contradiction with the hierarchy theorem for alternating Turing machines. □

The next two theorems (and their proofs) are analogous to those proved earlier (Theorems 4.8 and 4.9) for the EXP-hierarchy. They will allow us to obtain a new result about *btt* -hard sets for *PSPACE*.

Theorem 4.24 *NP has no $\leq_T^p$ -complete set S with $\|S^{\leq n}\| \leq 2^{polylog}$ unless the $2^{polylog}$ hierarchy collapses to $\Delta_1^{PL}[polylog]$.*

Theorem 4.25 *$PSPACE$ does not have a $\leq_T^p$ -hard set S with $\|S^{\leq n}\| \leq 2^{polylog}$ unless $\bigcup_{k>0} DSPACE(2^{\log^k n}) \subseteq \Sigma_2^{PL}$.*

Combining this last theorem with Proposition 4.5 of Section 4.2 we obtain a new result concerning almost sparse $\leq_{btt}^p$ -hard sets for PSPACE. The results provides strong evidence that such sets do not exist. Of course, a proof that they don't would imply $P \neq PSPACE$.

Theorem 4.26 *If $PSPACE \leq_{btt}^p S$ with $\|S\| \leq 2^{polylog}$ then*

- *$PSPACE \subseteq DTIME(2^{polylog})$ and*
- *$EXPSPACE = EXP$.*

Proof: If an S exists as in the hypothesis, then clearly S is $\leq_{btt}^p$ -hard for NP. So by Proposition 4.5 , $NP \subseteq DTIME(2^{polylog})$. This implies $NTIME(2^{polylog}) = DTIME(2^{polylog})$ by the closure properties of $2^{polylog}$ functions. Since the *PL*-hierarchy obeys downward separation, this implies that the whole *PL*-hierarchy collapses to $DTIME(2^{polylog})$.

Now, by the previous theorem, our hypothesis implies that $PSPACE \subseteq \Sigma_2^{PL} = DTIME(2^{polylog})$. From this inclusion, a straightforward padding argument yields $EXPSPACE = EXP$. $\Box$

4.5 Notes and Open Problems

In this chapter we considered the density of hard and complete sets, under various kinds of reductions. For $\leq_m^p$ (and even $\leq_{btt}^p$) -hard sets for NP the results were motivated by the Berman Hartmanis Conjecture. By varying the density from polynomial dense to subexponential dense we gave evidence for this Conjecture. (c.f. Theorem 4.5)

On the other hand by studying the density of $\leq_T^p$ - hard sets for NP we hope to get some more insight in how computationally (non)complex some sets in NP are. Research along these lines [KL80] indicates that NP -complete sets (for example SAT) are even *nonuniform* computationally hard, in the sense that they probably do not even $\leq_T^p$ -reduce to a sparse set. Again by varying the density to subexponential we gave evidence that these kind of sets are probably even harder to compute.(c.f. Theorem 4.6) There we considered the possibility of $2^{polylog}$ dense sets S, being $\leq_T^p$ -hard for NP. It would be interresting however, to obtain a good upper bound on the complexity of such S. In particular, can S be computed in EXP ?

We have considered reductions to almost-sparse sets and the existence of circuit families of size $2^{polylog}$. From these we have proved collapses within the EXP-hierarchy. What happens if the sets or circuit families are larger than $2^{polylog}$, yet still subexponential ? One would suspect that a collapse in a suitably large superpolynomial hierarchy would result. For example, using the methods here, it is possible to extend our results to some densities larger than $2^{polylog}$. From the assumption that SAT is $\leq_T^P$-reducible to a set S of density $2^{logn^{loglogn}}$ we can prove that the double exponential hierarchy collapses to the second level. We don't know whether there is a corresponding result for any subexponential density.

Corollary 4.14 yielded strong consequences from the existence of polynomial-size circuits for EXP^{NP}. In particular, under this assumption, $EXP^{NP} = EXP = \Sigma_2^p \cap \Pi_2^p$. Similarly, it was previously known that if EXP has polynomial size circuits then $EXP = \Sigma_2^p \cap \Pi_2^p$. But what follows from the intermediate assumption that $NEXP$ has polynomial-size circuits ? The clear expectation is that then $NEXP = EXP$, but this remains an open question. Interestingly, this question seems intertwined with the question of how difficult it is to find an accepting path in an accepting $NEXP$ computation. Analyzing the proof of Theorem 4.12 leads to the conclusion that if one could find such an accepting path in $NEXP$ then the $NEXP = EXP$ conclusion would likely follow from the assumption that $NEXP \subseteq P/poly$. This same question was raised by Babai, Fortnow and Lund in [BFL90] where it arises when considering upper bounds for the power of the prover in multiple prover interactive proofs. If accepting paths in $NEXP$ computations can be found in $NEXP$ then $NEXP$ provers suffice for MIP proofs. The current best upper bound is EXP^{NP}.

As can also be seen in this chapter, research has been roughly moving along the following two lines:

1. What is the density of $\leq_m^p$ -hard sets for this complexity class.

2. What is the density of $\leq_T^p$ -hard sets for this complexity class.

Almost all the results from type 1 are based on the ideas of Mahaney. Whereas the results of type 2 are based on the ideas originating from Karp and Lipton. It would be very nice to see one unifying theorem, using both type of ideas.

Chapter 5

Sparse Sets

5.1 Introduction

In chapter 4 we considered what sets are (probably) not reducible to a sparse set. In this chapter we will study more closely the sets that *do* reduce to a sparse set.

The motivation for this kind of study is two sided:

- Sets that are reducible to a sparse set are in some sense tractable: these sets can be recognized in polynomial time provided a sparse data base is present for the lookup of strings. The data base itself may be hard to compute, but once computed admits polynomial time algorithms. So it is important in a theory of tractable computation to consider these sets as well.

- On the other side the study of sets that are reducible to a sparse set may reveal properties that these sets possess. These properties give a handle on proving sets *not* being reducible to sparse sets, by virtue of lacking this property.

In this chapter we will use a special type of sparse set: a set that is a subset of $\{0\}^*$. These sets are called *tally* sets.

We begin by formalizing the above intuition, by introducing these classes as follows:

Definition 5.1 *Let $\leq_r^p$ be any of the reductions defined in section 2.6. Let SPARSE denote the class of all sparse sets and TALLY the class of all tally sets.*

- $R_r(SPARSE) = \{A \mid A \leq_r^p S, S \in SPARSE\}$.
- $R_r(TALLY) = \{A \mid A \leq_r^p T, T \in TALLY\}$.

This definition allows us to reformulate many of the previous results. For example Mahaney's result about $\leq_m^p$ -hard sets for NP translates to: if $NP \subseteq R_m(SPARSE)$ then $P = NP$.

In this chapter we will study the various classes $R_r(SPARSE)$ and $R_r(TALLY)$ and their relation w.r.t. different kinds of reductions. Research along these lines was pioneered by Book and Ko [BK88], and later by Ko [Ko89]. In these papers almost all relations w.r.t. different kinds of reductions between the $R_r(SPARSE)$ classes are proven. Typical results are of the following form:

Theorem 5.2
1. *[BK88] $R_T(SPARSE) = R_{tt}(SPARSE)$.*
2. *[BK88] $R_m(SPARSE) = R_{bctt}(SPARSE)$*
3. *[BK88] $R_{bctt}(SPARSE) \subset R_{1tt}(SPARSE)$*
4. *[BK88] $R_{k\text{-}tt}(SPARSE) \subset R_{(k+1)\text{-}tt}(SPARSE)$ (for all $k > 0$)*
5. *[Ko89] $R_{k\text{-}dtt}(SPARSE) \subset R_{(k+1)\text{-}dtt}(SPARSE)$ (for all $k > 0$)*
6. *[Ko89] $R_{btt}(SPARSE) \subset R_{dtt}(SPARSE)$*
7. *[Ko89] $R_{ctt}(SPARSE) \not\subseteq R_{dtt}(SPARSE)$*
8. *[GW92] $R_{dtt}(SPARSE) \not\subseteq R_{ctt}(SPARSE)$*

For tally sets the picture is also very well understood and similar results are obtained:

Theorem 5.3
1. *[BK88] $R_m(TALLY) = R_{btt}(TALLY)$.*
2. *[BK88] $R_{tt}(TALLY) = R_T(TALLY) = R_T(SPARSE)$*
3. *[Ko89] $R_{ctt}(TALLY) \not\subseteq R_{dtt}(TALLY)$*

4. *[Ko89]* $R_{dtt}(TALLY) \not\subseteq R_{ctt}(TALLY)$

5. *[Ko89]* $R_m(TALLY) \subset R_{dtt}(TALLY) \subset R_{tt}(TALLY)$

6. *[Ko89]* $R_m(TALLY) \subset R_{ctt}(TALLY) \subset R_{tt}(TALLY)$

For a good overview we direct the interested reader to [HOW92].

In this chapter we will pay close attention to the disjunctive and conjunctive reductions to sparse and tally sets. We will prove, refuting a conjecture of Ko [Ko89], the following relationship between sparse and tally sets. *SPARSE* is in $R_{ctt}(TALLY)$. This result, not true for disjunctive reductions to sparse sets, enables us to derive a handful of other results.

One of the nice things about tally sets is that they are closed, in some sense, under complementation. If A is, for example, $\leq_m^p$ -reducible to a tally set T, then the complement of A, $\overline{A}$, is *also* $\leq_m^p$ -reducible to some other tally set. The reason is that for tally sets one can restrict a reduction to a tally set to query only strings of the form 0^n. The queries that are not of this form are simply not in the tally set and hence do not have to be queried. So the complement of A reduces to the complement of the tally set,$\overline{T}$, which is *not* even sparse. But because of this restriction, only queries of the form 0^n are queried, so $T' = \{0^n \mid 0^n \notin T\}$, will witness the fact that $\overline{A}$ reduces $\leq_m^p$ to a tally set. This complementation trick is not possible for $R_m(SPARSE)$, since from 2 and 3 (Theorem 5.2) it follows that $R_m(SPARSE) \subset R_{1tt}(SPARSE)$, and thus is $R_m(SPARSE)$ not closed under complementation.

We will prove in the next section the main result, and then discuss some corollaries in section 5.3. The new work is taken from cite-BuhrmanLongpreSpaan92.

5.2 Conjunctive Reductions to Tally Sets

Theorem 5.4 *SPARSE* $\subseteq R_{ctt}(TALLY)$.

Proof: Let S be a sparse set of density $d(n)$, where d is a polynomial time computable function, and $d = n^{O(1)}$ We show that $S \in R_{ctt}(TALLY)$.

We have to build a $\leq_{ctt}^{p}$-reduction M_c from S to a tally set T. We can insure that $Q(M_c, x) \cap Q(M_c, y) = \emptyset$ for $|x| \neq |y|$ by building M_c such that every element in $Q(M_c, x)$ is of the form $0^{<n,j>}$ where n is the length of x. In the following, let $x_1, \cdots, x_{2^n}$ be the 2^n strings of length n. Note that if M_c is a reduction from S to T, then $x_i \in S \Leftrightarrow Q(M_c, x_i) \subseteq T$. Since this property holds for all x_i, a $\leq_{ctt}^{p}$-reduction generates a family of 2^n tally sets such that for all $x_i \notin S : Q(M_c, x_i) \not\subseteq \bigcup_{x_j \in S} Q(M_c, x_j)$. Whether the reduction is possible depends on whether we can efficiently construct such a family of sets. The existence of these kinds of families have been studied in [EFF82, EFF85, NW88]. In [NW88], they were used in the construction of pseudo-random number generators. We will construct a family of sets $\mathcal{F} = \{Q_1, \cdots, Q_{2^n}\}$, with the following properties:

1. $Q_i \in \mathit{TALLY}$,
2. Q_i can be generated in polynomial time (in n),
3. For *any* $d(n) + 1$ sets $Q_{i_1}, \cdots, Q_{i_{d(n)}}, Q_k \in \mathcal{F}$ such that $k \notin \{i_1, \cdots, i_{d(n)}\}$, $Q_k \not\subseteq \bigcup_{j=1}^{d(n)} Q_{i_j}$.

If we set the tally set $T = \bigcup_{x_i \in S} Q_i$, then $x_i \in S$ if and only if $Q_i \subseteq T$, since S is of density $d(n)$. If we are able to generate Q_i in polynomial time (in n), then we can define the $\leq_{ctt}^{p}$ -reduction M_c from $S^{=n}$ to T by setting $Q(M_c, x_i) = Q_i$. First we show by the next lemma that property 3 above follows from the following stronger property which is easier to verify.

Lemma 5.5 *Let* $\mathcal{F} = \{Q_1, \cdots, Q_{2^n}\}$, *be a family of sets such that* $\|Q_i\| > r \cdot d(n)$ *and* $\|Q_i \bigcap Q_j\| \leq r$, *for* $i \neq j$. *Then, for* any $d(n) + 1$ *sets* $Q_{i_1}, \cdots, Q_{i_{d(n)}}, Q_k \in \mathcal{F}$ *such that* $k \notin \{i_1, \cdots, i_{d(n)}\}$, $Q_k \not\subseteq \bigcup_{j=1}^{d(n)} Q_{i_j}$.

Proof: Suppose not i.e. there exist $d(n) + 1$ sets $Q_{i_1}, \cdots, Q_{i_{d(n)}}, Q_k \in \mathcal{F}$ such that $k \notin \{i_1, \cdots, i_{d(n)}\}$, and $Q_k \subseteq \bigcup_{j=1}^{d(n)} Q_{i_j}$. Since $\|Q_k\| > r \cdot d(n)$, there must exist a j such that $1 \leq j \leq d(n)$ and $\|Q_k \bigcap Q_{i_j}\| > r$. But this contradicts the fact that the size of the intersection of any two different sets is at most r. $\square$

One way to construct these families is as follows. Let $GF(p)$ be a finite field with a prime number of elements. Note here that we can always find a prime between n and $2n$ [Che52]. More recent research, trying to narrow this gap, indicates that there actually is a prime between n and $n + n^{\alpha}$, where α can be taken as small as $17/31 (= 0.548387...)$ [Pin84].

We are going to consider polynomials over $GF(p)$, for p prime. We need an easy fact about roots of polynomials over finite fields. For more detail see section 6.6 in [Coh74].

Fact 5.6 *Two different polynomials of degree $\leq r$ cannot intersect on more than r points in $GF(p)$.*

We represent a polynomial of degree $\leq r$ by its $r+1$ coefficients. We view each polynomial as a $(r+1)$-digit number in base p. With the i^{th} polynomial, denoted q_i, we mean the polynomial whose representation in base p is the number that represents i. Consider the following family of sets: $Q_i = \{0^{<n,a,q_i(a)>} \mid a \in GF(p)\}$. We will choose r and p such that the conditions of Lemma 5.5 are fulfilled. Observe that Q_i is a tally set of size p, and that for two different polynomials, q_i and q_j, $\|Q_i \bigcap Q_j\| \leq r$. It remains to force the following requirements:

1. $p^{r+1} \geq 2^n$ (We need 2^n different sets)
2. $r{\cdot}d(n) < p$ (to fulfill the requirements of Lemma 5.5)

It is easy to verify that taking $r = \frac{2n}{\log n}$, and p the first prime larger than $r{\cdot}d(n)$ fulfills these two requirements.

The only thing remaining is to show that we can generate the i^{th} set Q_i in polynomial time (in n). First we have to find a suitable prime number p. Since $|r{\cdot}d(n)|$ is $O(\log(n))$ and because there is a prime between $r{\cdot}d(n)$ and $2r{\cdot}d(n)$, we can do a brute force search (or do a more sophisticated sieve method [Pri83]) in polynomial time. Next we have to pick the i^{th} polynomial over $GF(p)$ (can easily be done in polynomial time), and compute Q_i. Since p is a prime number, the operations in $GF(p)$ are simply multiplication and addition modulo p, which also can be done in polynomial time. $\Box$

Recall that the $\leq^p_{ctt}$-reduction M_c from $S^{=n}$ to $\bigcup_{x_i \in S} Q_i$ is defined by $Q(M_c, x_i) = Q_i$. Since $\|Q_i\| = p \leq 2r{\cdot}d(n) \leq \frac{4nd(n)}{\log n}$, we have in fact shown that $S \in R_{O(\frac{nd(n)}{\log n})-ctt}(\mathit{TALLY})$. As recently shown by Saluja [Sal92], this bound is optimal.

Notice that if we consider probabilistic reductions [AM77, CKR91], we can randomly choose exactly one of the strings from $Q(M_c, x)$ and get a many-one reduction with a one-sided error. Furthermore by simply carrying the construction out in a lager Galois Field, i.e. take a bigger prime p for $GF(p)$, we can get the error bounded by $1/p(n)$ for arbitrary polynomial $p(n)$. Schöning noted this in [Sch92].

Corollary 5.7 $R_{ctt}(\mathit{SPARSE}) = R_{ctt}(\mathit{TALLY})$.

Corollary 5.8 *co-SPARSE* $\subseteq R_{dtt}(\mathit{TALLY})$.

Proof: If A is $\leq^p_{ctt}$-reducible to a tally set, then $\overline{A}$ is $\leq^p_{dtt}$-reducible to a tally set. □

The following theorem can be derived using Theorem 5.4. It refutes another conjecture from [Ko89].

Theorem 5.9 $R_{bdtt}(\mathit{SPARSE}) \subseteq R_{ctt}(\mathit{SPARSE})$.

Proof: Let A be $\leq^p_{k\text{-}dtt}$-reducible to some sparse set S via M_d. Using Theorem 5.4 we get that S is $\leq^p_{ctt}$-reducible to some tally set T_S witnessed by M_c. We will construct a tally set T and a reduction M such that $A \leq^p_{ctt} T$ via M. Define

$$T = \{0^{<n_1,\cdots,n_k>} \mid n_j \in I\!N \text{ and } \exists i : 0^{n_i} \in T_S\}.$$

In the following it is convenient to view T as a Cartesian product. For $A_1, \cdots, A_k$ tally sets, let

$$A_1 \times \cdots \times A_k = \{0^{<n_1,\cdots,n_k>} \mid 0^{n_i} \in A_i\}.$$

Define the $\leq^p_{ctt}$ -reduction M as follows: if $Q(M_d, x) = \{y_1, \cdots, y_k\}$, then let $Q(M, x) = Q(M_c, y_1) \times \cdots \times Q(M_c, y_k)$. Note that M works in polynomial time, since both M_d and M_c do. It remains to show that M reduces A conjunctively to T.

$$\begin{array}{rcl}
x \in A & \Rightarrow & \exists i : y_i \in S \\
& \Rightarrow & Q(M_c, y_i) \subseteq T_S \\
& \Rightarrow & Q(M_c, y_1) \times \cdots \times Q(M_c, y_k) \subseteq T. \\
x \notin A & \Rightarrow & \forall i : y_i \notin S \\
& \Rightarrow & \forall i \exists 0^{n_i} : 0^{n_i} \in Q(M_c, y_i) \text{ and } 0^{n_i} \notin T_S \\
& \Rightarrow & 0^{<n_1, \cdots, n_k>} \notin T \\
& \Rightarrow & Q(M_c, y_1) \times \cdots \times Q(M_c, y_k) \not\subseteq T.
\end{array}$$

□

Other applications of Theorem 5.4 can be found in [AKM92]. Their results take advantage of the fact that $R_m(TALLY)$ is closed under complements, and Theorem 5.4 enables one to go "conjunctively" from sparse sets to tally sets. This kind of reasoning will also be used in the next section.

To understand the relationship between sparse and tally sets, it is important to know which reductions are able to differentiate between tally and sparse sets, and which aren't. It is known that $R_{tt}(SPARSE)$ $= R_{tt}(TALLY)$ [HIS85] and our Corollary 5.7 gives the analog for $\leq^p_{ctt}$-reductions. On the other hand, there *do* exist reductions that are more powerful with tally oracles than with sparse oracles: This holds for instance for many-one reductions and for disjunctive truth-table reductions [Ko89].

As the next theorem shows, *positive* truth-table reductions on sparse and tally sets behave like $\leq^p_{ctt}$ -reductions and not like $\leq^p_{dtt}$ -reductions:

Theorem 5.10 $R_{ptt}(SPARSE) = R_{ptt}(TALLY)$.

The result follows immediately from the following theorem that claims that $\leq^p_{ptt}$ -reductions to tally sets capture the class $R_{tt}(TALLY)$.

Theorem 5.11 $R_{tt}(TALLY) = R_{ptt}(TALLY)$.

Proof: Let A be a set in $R_{tt}(TALLY)$, and suppose T is a tally set such that $A \leq^p_{tt} T$ via machine M_{tt}. Assume that machine M_{tt} works in time $p(n)$ where p is a polynomial. We have to show that $A \in R_{ptt}(\mathit{TALLY})$. We define the tally set T', that will witness the fact that $A \in R_{ptt}(\mathit{TALLY})$, as follows:

$$T' = \{0^{<n,0>} \mid 0^n \in T\} \cup \{0^{<n,1>} \mid 0^n \notin T\}$$

We claim that $A \leq_{ptt}^{p} T'$ by the following reduction:

On input x of length n do the following:

1. If there exists an $m \leq p(n)$ such that $0^{<m,0>}$ and $0^{<m,1>}$ are both *not* in the oracle set, then reject;

2. else, if there exists an $m \leq p(n)$ such that $0^{<m,0>}$ and $0^{<m,1>}$ are both *in* the oracle set, then accept;

3. otherwise, simulate the old tt-reduction M_{tt} on input x, replacing each query 0^m on the QUERY-tape by $0^{<m,0>}$.

It is immediate that this reduction reduces A to T', since by definition of T' we are always in case 3, which implies that we just simulate M_{tt}. It remains to show that the reduction is positive. Suppose for a contradiction that it isn't. Then there exist a string x of length n and two oracle sets $X \subset Y$ such that x is accepted with oracle X and rejected with oracle Y. Since x is accepted with oracle X, we cannot be in case 1, that is, it must be the case that for all $m \leq p(n)$ either $0^{<m,0>} \in X$, or $0^{<m,1>} \in X$. Now look at Y. If $Y - X$, does not contain strings of the form $0^{<m,i>}$ for $m \leq p(n), i \in \{0,1\}$, then $M_{tt}^{Y}(x)$ behaves in exactly the same way as $M_{tt}^{X}(x)$. In particular, x is accepted, which contradicts our assumption. Therefore, suppose that for some $m \leq p(n)$ and $i \in \{0,1\}$ it is the case that $0^{<m,i>}$ occurs in Y but not in X. Then it must be the case that $0^{<m,(1-i)>} \in X$, and therefore, since $X \subseteq Y$, both $0^{<m,0>}$ and $0^{<m,1>}$ are in Y. This implies that we are in case 2, and thus, x is accepted contrary to the assumption. $\Box$

Note that by the construction, it is immediate that T' is $\leq_{1-tt}^{p}$ reducible to T.

5.3 Conjunctive and Disjunctive Reductions

In this section we will discuss some consequences of Theorem 5.4.

Gavaldà and Watanabe [GW92] showed that $R_{ctt}(\mathit{SPARSE}) \not\subseteq R_{dtt}(\mathit{SPARSE})$. Combining this result with Theorem 5.4, we can quickly derive the following theorem of Ko:

Theorem 5.12 ([Ko89]) $R_{dtt}(\mathit{SPARSE}) \not\subseteq R_{ctt}(\mathit{SPARSE})$.

Proof: Let A be a set in $R_{ctt}(\mathit{SPARSE})$, that is not in $R_{dtt}(\mathit{SPARSE})$. Consider the set $\overline{A}$. Since $A \in R_{ctt}(\mathit{SPARSE})$ and $R_{ctt}(\mathit{SPARSE}) = R_{ctt}(\mathit{TALLY})$ by Theorem 5.4, we have that $A \in R_{ctt}(\mathit{TALLY})$. By simple complementation, it follows that $\overline{A} \in R_{dtt}(\mathit{TALLY})$ and therefore, $\overline{A} \in R_{dtt}(\mathit{SPARSE})$. $\overline{A}$ cannot be in $R_{ctt}(\mathit{SPARSE})$, for suppose $\overline{A} \in R_{ctt}(\mathit{SPARSE})$. Then using Theorem 5.4, $\overline{A} \in R_{ctt}(\mathit{TALLY})$, so $A \in R_{dtt}(\mathit{TALLY}) \subseteq R_{dtt}(\mathit{SPARSE})$, contradicting our choice of A. □

Gavaldà and Watanabe's proof actually provides something stronger. They show that $R_{f(n)\text{-}ctt}(\mathit{SPARSE})$ is not included in $R_{dtt}(\mathit{SPARSE})$ for arbitrary small polynomial time computable unbounded function f. Ko's proof of Theorem 5.12 does not seem to provide this generalization, and the above proof does not generalize directly, because when we go conjunctively from a sparse set to a tally set, we need a polynomial number of queries. To be able to use the previous argument, while keeping the number of queries small, we need a strengthening of Gavaldá and Watanabe's theorem to tally sets:

Theorem 5.13
For any polynomial time computable unbounded function f:
$R_{f(n)\text{-}ctt}(\mathit{TALLY}) \not\subseteq R_{dtt}(\mathit{SPARSE})$.

Proof: Without loss of generality, assume $f(n) \leq \log n$. Recall that a Kolmogorov random string x of length n is a string such that $K(x) \geq n$. Such strings are called incompressible because the shortest description of the string is the string itself. For every n, let x_n be a Kolmogorov random string of length n. Define

$$\begin{aligned} A = \{ & <0^n, <i_1, b_1>, \cdots, <i_{f(n)}, b_{f(n)}>> \text{ such that} \\ & 1 \le i_1 < i_2 < \cdots < i_{f(n)} \le n \text{ and} \\ & \text{for every } j, \text{ the } i_j\text{-th bit of } x_n \text{ is } b_j\}. \end{aligned}$$

It is immediate that $A \le^p_{f(n)\text{-}ctt} T$, where

$$T = \{0^{<n,i_j,b>} \mid \forall j : \text{ the } i_j\text{-th bit of } x_n \text{ is } b\}.$$

To show that A is not $\le^p_{dtt}$-reducible to any sparse set, leading to a contradiction, assume $A \le^p_{dtt} S$, via reduction M_d, where M_d runs in time n^c and $\|S^{\le n}\| \le n^c$.

Let A_n be the set of all strings of A of the form $<0^n, \cdots>$. We will show that there is a string y_n in S that is queried by many strings from A_n (Lemma 5.14). Suppose that a string $<0^n, <i_1, b_1>, \cdots, <i_{f(n)}, b_{f(n)}>>$ queries the string y_n. Since M_d is a $\le^p_{dtt}$-reduction from A to S and $y_n \in S$, this provides us with the $f(n)$ bits $i_1, i_2, \cdots, i_{f(n)}$ of x_n. By a careful counting argument, we show below that, for n large enough, we get enough bits of x_n from y_n to contradict the randomness of x_n.

Lemma 5.14 *There exist a constant d and for every n a string y_n in S such that:*

$$\|\{z \in A_n \mid y_n \in Q(M_d, z)\}\| \ge n^{\frac{1}{2}f(n)-d}$$

Proof: The number of strings in A_n is $\begin{pmatrix} n \\ f(n) \end{pmatrix} \ge \left(\frac{n}{f(n)}\right)^{f(n)}$. Thus, for $f(n) \le n^{\frac{1}{2}}$, $\|A_n\| \ge n^{\frac{1}{2}f(n)}$. For each string z in A_n, there is a string in $S \cap Q(M_d, z)$. Since strings in A_n are certainly of length less than $2n$, the queried strings are of length at most $(2n)^c$. Thus, there are at most $((2n)^c)^c = (2n)^{c^2}$ strings of S in $\cup_{z \in A_n} Q(M_d, z)$. There must be a string y_n in S that is in $Q(M_d, z)$ for at least $\|A_n\|/(2n)^{c^2}$ many z's. Since $\|A_n\| \ge n^{\frac{1}{2}f(n)}$, $\|A_n\|/(2n)^{c^2} \ge n^{\frac{1}{2}f(n)-d}$ for a suitable d. $\Box$

Given a set $Y \subseteq A_n$, let I_Y be the set of indices i_j that are mentioned in the strings from Y.

Lemma 5.15 *Let $Y \subseteq A_n$, then $\|Y\| \leq \|I_Y\|^{f(n)}$.*

Proof: Each string in Y mentions exactly $f(n)$ bits of I_Y. There are exactly $\left(\begin{array}{c} \|I_Y\| \\ f(n) \end{array} \right)$ ways to select $f(n)$ bits from the set of indices I_Y, so

$$\|Y\| \leq \left(\begin{array}{c} \|I_Y\| \\ f(n) \end{array} \right) \leq \|I_Y\|^{f(n)}.$$

□

Lemma 5.16 *There exists a string $y_n \in S$ such that for the set Y of strings in A_n that query y_n: $\|I_Y\| \geq n^{\frac{1}{2}-d/f(n)}$.*

Proof: Let y_n be given by Lemma 5.14, and let Y be the set of strings $z \in A_n$ such that $y_n \in Q(M_d, z)$. Then, by Lemma 5.15,

$$n^{\frac{1}{2}f(n)-d} \leq \|I_Y\|^{f(n)}.$$

$$\|I_Y\| \geq n^{(\frac{1}{2}f(n)-d)/f(n)} = n^{\frac{1}{2}-d/f(n)}.$$

□

Now, to derive a contradiction, we show how to describe x_n with fewer than n bits. To describe x_n, use the string y_n from Lemma 5.16. To compute y_n, we need one of the strings $z \in A_n$ that query y_n, and the index of y_n in the set of queries. The string z can be described using $O(f(n) \log n)$ bits and the index can be described in $O(\log n)$ bits. It follows that y_n can be described using $O(f(n) \log n)$ bits. Given y_n, we can compute all the bits of x_n that are mentioned in strings from the set Y of strings in A_n that query y_n. Now look at the sequence containing all the bits of x_n that are not mentioned by Y. This requires $n - \|I_Y\| \leq n - n^{\frac{1}{2}-d/f(n)}$ bits. Since the bits described by Y all contain their index, they can be inserted into their respective position. The total number of bits to describe x_n is $n - n^{\frac{1}{2}-d/f(n)} + O(f(n) \log n)$, which is strictly less than n if $f(n)$ is unbounded and $\leq \log n$. □

Now we can derive the wanted theorem.

Theorem 5.17
For any polynomial time computable unbounded function f:
$R_{f(n)\text{-}dtt}(\mathit{TALLY}) \not\subseteq R_{ctt}(\mathit{SPARSE})$.

Proof: Using Theorem 5.13, we can use the same reasoning as in the proof of Theorem 5.12. Since we start from a tally set, we don't have the problem associated with the blow up in number of queries. □

The following corollaries can all be obtained from Theorem 5.13 and Theorem 5.17.

Corollary 5.18
For any polynomial time computable unbounded function f:
$R_{f(n)\text{-}ctt}(\mathit{SPARSE})$ *and* $R_{f(n)\text{-}dtt}(\mathit{SPARSE})$ *are not closed under complementation.*

Corollary 5.19 *For any polynomial time computable unbounded functions f:* $R_{f(n)\text{-}ctt}(\mathit{SPARSE})$ *and* $R_{f(n)\text{-}dtt}(\mathit{SPARSE})$ *are incomparable.*

Corollary 5.20 *For any polynomial time computable unbounded functions f:* $R_{f(n)\text{-}dtt}(\mathit{SPARSE})$ *and* $R_{f(n)\text{-}ctt}(\mathit{SPARSE})$ *are strictly included in* $R_{f(n)\text{-}tt}(\mathit{SPARSE})$.

These results hold for the corresponding $R_r(\mathit{TALLY})$ classes as well. For bounded conjunctive and disjunctive reductions to sparse sets, we get the following analog:

Corollary 5.21 ([Ko89]) $R_{k\text{-}tt}(\mathit{SPARSE})$ *and* $R_{ctt}(\mathit{SPARSE})$ *are incomparable.*

Theorem 5.22 *For all* $k \geq 1$, $R_{k\text{-}ctt}(\mathit{SPARSE})$, $R_{k\text{-}dtt}(\mathit{SPARSE})$, $R_{bdtt}(\mathit{SPARSE})$ *and* $R_{bctt}(\mathit{SPARSE})$ *are not closed under complementation, and therefore strictly included in* $R_{btt}(\mathit{SPARSE})$.

Proof: If $R_{bdtt}(\mathit{SPARSE})$ is closed under complementation, then $R_{1\text{-}tt}(\mathit{SPARSE}) \subseteq R_{bdtt}(\mathit{SPARSE})$. By Theorem 5.9, it follows that $R_{1\text{-}tt}(\mathit{SPARSE}) \subseteq R_{ctt}(\mathit{SPARSE})$, contradicting [Ko89]. For the bounded conjunctive case we can argue in a similar way. □

Note that this theorem does not hold for the corresponding $R_r(\mathit{TALLY})$ classes: It follows from [Ko89] that $R_m(\mathit{TALLY}) = R_{k\text{-}ctt}(\mathit{TALLY}) = R_{k\text{-}dtt}(\mathit{TALLY}) = R_{btt}(\mathit{TALLY})$, and thus all these classes are closed under complementation.

Chapter 6

Structure of Complete Sets

In this chapter we will examine more closely the structure of complete sets. We will try to see whether they have properties in common apart from being complete.

Research along these lines was initiated by Post, as a way to solve his problem whether there exist Turing incomplete sets in $RE - REC$. He proposed to find sets that possess structural properties that prevent them from being complete and recursive.. Here we take the opposite approach and examine, whether complete sets all possess certain structural properties. This gives a handle on proving sets not complete by showing that they lack some structural property. The new work is taken from [BHT93].

The main question then becomes what kind of property are we looking for? Probably the first property considered in this frame work is the density of the complete set, as was discussed in Chapter 4. In this chapter we will turn our attention to other 'structural' set properties of complete sets. These issues have not been as well studied as the density aspects. In this chapter we cannot give a full picture of all possible structural properties, but merely scan through a couple of them.

6.1 Immunity

The first property we touch here is whether computationally complex sets possess infinite subsets that are easy (i.e. polynomial time) to compute. If this is not the case the set is said to be immune.

Definition 6.1 *A set A is p-immune if there is no infinite set $B \in P$, such that $B \subseteq A$.*

So p-immune sets are sets that are not only computationally hard to compute, but also their infinite subsets are complex. Obviously if there are p-immune sets in NP, then $P \neq NP$. But even under the hypotheses that $P \neq NP$ it is not known whether there exist p-immune sets in NP.

So what about NP -complete sets under $\leq_m^p$ -reductions? Again we only have a partial answer. Observe that if A is not p-immune and A is polynomial time isomorphic to B, then B is not p-immune. This observation together with the fact that for example SAT is not p-immune (the set of formula's $(a \vee \sim a), a \vee \sim a \vee \sim a) \wedge (a \vee \sim a)) \ldots$, is an easy to recognize subset of SAT) and Berman and Hartmanis observation about known NP -complete sets being isomorphic [BH77] implies that all known NP -complete sets are not p-immune.

For EXP the situation is somewhat better understood. Berman proved in his thesis [Ber77], in an attempt to prove that all $\leq_m^p$ -complete sets for EXP are p-isomorphic, that they are complete via 1-1 and length increasing reductions. So the only ingredient that was, and still is, missing was the polynomial time invertibility.

Theorem 6.2 ([Ber77]) *All $\leq_m^p$ -complete sets for EXP are $\leq_{m,1\text{-}1,li}^p$ -complete.*

(For an extremely nice proof we refer the interested reader to [GH89])

This, for the isomorphism conjecture for EXP insufficient result, shows however that all $\leq_m^p$ -complete sets for EXP are *not* p-immune. A result that surely would follow from a proof establishing the isomorphism conjecture for EXP.

Theorem 6.3 *All $\leq_m^p$ -complete sets for EXP are not p-immune.*

Proof: Let A be a $\leq_m^p$ -complete set. Consider the set $T = \{0^n \mid n \in \mathbb{N}\}$. By the previous theorem $T \leq_{m,1\text{-}1,li}^p A$ via M_1 say. Consider the set of queries Q, asked by M_1 on input 0^n (for all n): $Q = \{y \mid \exists n : y \in$

$Q(M_1, 0^n)\}$. Obviously $Q \subseteq A$, and because M_1 is length increasing, it follows that $Q \in P$. $\Box$

Unfortunately it is not known whether Theorem 6.2 goes through for *NEXP* and consequently it is an open problem whether there exist *NEXP* $\leq_m^p$ -complete sets that are p-immune.

Can we propagate this non-immunity property to other kind of completeness notions? The answer is a quick and loud no.

Theorem 6.4 *There exists a* $\leq_{2-dtt}^p$ *-complete set B in NEXP such that B is p-immune.*

Proof:(Sketch) Let $M_1, M_2, \ldots$ be an enumeration of polynomial time Turing machines, that accept the languages in P. We have to construct a set that is complete ($\leq_{2-dtt}^p$) on one hand, and on the other we have to make sure that the set is p-immune. We construct the set B such that for every $x \in K$, it is always the case that $<x, 0>$ or $<x, 1>$ is in B. Clearly this will guarantee that B is $\leq_{2-dtt}^p$ -complete. On the other hand we have to make B p-immune. This is ensured by the following requirement: Every machine M_i that recognizes an infinite subset (i.e. in polynomial time), will accept a string in the complement of B. Surely this will make B p-immune. We will use a priority argument to force these requirements. B is constructed in stages:

stage n:

At stage n we put $<i, x>$ $(i = 0, 1)$ in B iff $x \in K$. Next simulate all the machines $M_i (0 < i < n)$ that are not yet satisfied for 2^n steps on all the inputs of the form $<0, x>$ or $<1, x>$ of length n. If there is no M_j accepting a string we go to stage $n + 1$. Otherwise call the machine with the *smallest* index that accepts a string $<i, x>$ satisfied and take $<i, x>$ out of B (i.e. put it in the complement of B). Go to the next stage.

end of stage n

We leave it to reader to check that this construction can be carried out in exponential time. Furthermore it is clear that in stage n at most one string of the form $<i, x> (i = 0, 1)$ is taken out of B, so $K \leq_{2-dtt}^p B$.

It remains to be shown that B is p-immune. Suppose for a contradiction that there exists a polynomial time computable infinite set S

that is a subset of B. Let $S = L(M_m)$. Since $L(M_m)$ is infinite there must be a stage $l \geq m$ such that M_m accepts a string $<i, x> = y$ of length l and m is the smallest index of machines that are not satisfied. At this stage however y is left out of B, contradicting the fact that S is a subset of B. $\Box$

The same statement is true for *EXP*.

6.2 Robustness

When is a complete set robust? The idea is to investigate how many elements need be taken out of a complete set in order to destroy the completeness of the set.

The game goes as follows: Let A be a complete set and S be another set. For what S is the set $A - S$ still complete and for what S is this not the case. Complete sets (under various kinds of reductions) for *EXP* will be studied. Whenever we are able, we will prove the results for *EXP hard* sets as well. From this it follows that the statement just proved is also true for *NEXP* -hard sets. Note that the game will only be restricted to sets S that are not dense since otherwise S can be taken as big as Σ^* and this surely renders A incomplete after removal.

This kind of research was first studied by Schöning [Sch86] who showed that for all complete sets A in *EXP* and every set D in P the set $A \Delta D$[1] is of exponential density. In [TFL91] Tang, Fu en Liu showed that D can be subexponential time computable and that in fact $A \Delta D$ remains exponential time complete. They further show the existence of arbitrarily sparse subexponential time computable sets S such that for *any* exponential time complete set A the set $A - S$ is no longer exponential time hard. The proof hinges again on the fact that for any exponential time computable set B and any exponential time complete set A there exists a *length increasing* reduction from B to A. Then the subexponential time complete set is constructed by choosing a sufficiently sparse polynomial time computable subset of $\{0\}^*$ and defining S as the image of this set varying over all polynomial time

[1]Recall that Δ stands for symmetric set difference and $A \Delta B$ is defined as $(A - B) \cup (B - A)$.

computable functions, i.e. $S = \{0^{b(i)} : |f_i(0^{b(i)})| > i\}$ and the $b(i)$ are chosen sufficiently far apart.

A closer look at the proof reveals that though the theorem just states that S is subexponential time computable there are various ways of making S come arbitrarily close to polynomial time. It therefore seems reasonable to ask if we can also choose S to be polynomial time computable. The answer to this question is no, as [TFL91] observe. From the $\leq^p_{m,1\text{-}1,li}$-reduction of $K' = K \times \Sigma^*$ to the EXP -complete set A we can easily construct a $\leq^p_m$ -reduction to $A - S$ for any polynomial time computable sparse set S.

The set K' itself is of course $\leq^p_m$ -complete for EXP. In fact it is $\leq^p_{dtt}$ -complete for EXP in a special way. For a given string x either all strings $<x, y>$ are in this set, or all are out depending on $x \in K$. Therefore as long as S is $p(n)$-sparse the set $K' - S$ remains $\leq^p_{dtt}$ -complete for EXP. The reduction from K to K' on input x just queries the set $\{<x, y> \mid 0 \leq y \leq p(2n) + 1\}$. Since all these strings have length $\leq 2|x|$ at least one of them is not in S and it is in K' iff $x \in K$. A theorem like "*There exists a sparse set S such that for any $\leq^p_{dtt}$ -complete set A for EXP the set $A - S$ is not $\leq^p_{dtt}$ -complete for EXP*" cannot exist for this reason. The best we can hope for is a theorem for reductions that can query less strings than $\|S^{\leq n}\|$ for each length n. On the other hand we want S to remain arbitrarily sparse, so the best bound we can hope for is some fixed but arbitrary constant. Such reductions are called *bounded truth table reductions*. And for these reductions we can obtain the theorem.

Theorem 6.5 *Given a recursive non-decreasing function $g(n)$ with $\lim_{n\to\infty} g(n) = \infty$. There exists a $g(n)$-sparse subexponential time computable set S such that for any $\leq^p_{btt}$ -complete set A for EXP the set $A - S$ is no longer $\leq^p_{btt}$ -complete.*

Proof: For a given set A we demonstrate the existence of a set L_A such that $L_A \leq^p_{btt} A$ iff there exists a reduction M_i that queries at least one string $y \in A$ with $|y| > b(i)$ on input $0^{b(i)}$. Next we let all such strings in S thereby killing the reduction. This technique was borrowed from Watanabe [Wat87a].

Let $\{M_i\}_i$ be an enumeration of all $\leq^p_{btt}$ -reductions. Without loss of generality we may assume that machine M_i generates $\leq i$ queries on any input. First we define a set of numbers $\{b(i)\}_i$ sufficiently far apart and sufficiently easy to recognize, i.e. we want that for each n the question "$\exists i : n = b(i)$?" can be answered in time $O(n)$, and furthermore we want for each i that $g(b(i)) > i^2$. This means that we can define at least i^2 strings in $S^{\leq b(i)}$ without disturbing the sparseness condition on S. In fact we just put $\leq i$ strings in each interval $S^{\leq b(i+1)} - S^{\leq b(i)}$,

Next we show a property that complete sets, under any kind of reduction posses. This observation was made by Watanabe [Wat87b]. Let $\{M_i\}_i$ be an enumeration of reductions we want to consider. In this case $\leq^p_{btt}$ -reductions.

Lemma 6.6 *Let A be any set in EXP. There exists a set L_A such that if $L_A \leq^p_{btt} A$, via machine M_j, then for almost all n, there exist a q_n in $Q(M_j, 0^{b(<j,n>)}) \bigcap A$, such that $|q_n| > b(<j,n>)$.*

Proof: Let A be any set in EXP, we define the set L_A as follows. Put $0^{b(<i,n>)}$ in L_A iff $M_i^{A^{\leq b(<i,n>)}}(0^{b(<i,n>)})$ rejects. If A is complete then because L_A is easily seen to be in EXP, $L_A \leq^p_{btt} A$. Lets assume that this fact is witnessed by machine M_j. Now M_j has the property that for almost all n it will query on inputs of the form $0^{b(<j,n>)}$ at least one string q such that $|q| > b(<i,n>)$ and $q \in A$. For suppose this is not true then because of the construction of L_A, it follows that $0^{b(<j,n>)}$ in L_A iff $M_i^{A^{\leq b(<i,n>)}}(0^{b(<i,n>)})$ rejects, and because no strings bigger then $b(<j,n>)$ in A are queried, $M_j^{A^{\leq b(<j,n>)}}(0^{b(<j,n>)}) = M_j^{A}(0^{b(<j,n>)})$. Contradicting the fact that M_j is a reduction from L_A to A. □

We let $S = \{y : y \in Q(M_i, 0^{b(i)})$ and $|y| > b(i)\}$ and claim that S is the set searched for.

First $\|S^{\leq n}\| \leq g(n)$ for each n, since for each n the only strings that are in $S^{\leq n}$ are the sets in $\bigcup_{j \leq i} Q(M_j, 0^{b(j)})$ for i the maximal integer such that $b(i) \leq n$. Hence there are at most i^2 strings in $S^{\leq n}$ and $b(i)$ was chosen such that $g(n) \geq g(b(i)) > i^2$ as required.

Second for any $\leq^p_{btt}$ -complete set A the set $A - S$ is not $\leq^p_{btt}$ -complete. If it were, then according to lemma 6.6, L_{A-S} reduces $\leq^p_{btt}$ to $A - S$ via a reduction M_j, that queries on input $0^{b(<j,n>)}$ at least one string bigger than $b(<j,n>)$ in $A - S$, but this cannot happen because of the way S is constructed. □

We observe that for *conjunctive* truth table reductions the statement of the theorem is valid without a constant bound on the number of queries. Consider the i^{th} conjunctive truth table reduction M_i. If $0^{b(i)} \notin L_A$, then for *any* S, M_i cannot be a $\leq^p_{ctt}$ -reductions from L_A to $A - S$. If on the other hand $0^{b(i)} \in L_A$ then $Q(M_i, 0^{b(i)}) \subset A$. Hence in the construction of S it suffices to put $\min\{y : y \in Q(M_i, 0^{b(i)})$ and $|y| > b(i)\}$ in S. From this we obtain.

Corollary 6.7 *Given a recursive non-decreasing function $g(n)$ with $\lim_{n\to\infty} g(n) = \infty$. There exists a $g(n)$-sparse subexponential time computable set S such that for any $\leq^p_{ctt}$ -complete set A for EXP the set $A - S$ is no longer $\leq^p_{ctt}$ -complete.*

Conjunctive truth table reducibilities form an exception in yet another way. For these reductions we can even let the set A be *EXP* -hard instead of *EXP* -complete. We use the fact that for conjunctive truth table reductions we can get a kind of 1-1 behavior for the query sets. A similar result for $\leq^p_m$ -hard sets that uses the fact that these sets are also hard under $\leq^p_{m,1\text{-}1}$ -reductions appears in [TFL91]. We can force the query sets belonging to two distinct inputs to be no subsets of each other.

Lemma 6.8 *If A is EXP -hard under $\leq^p_{ctt}$ -reductions, then for any set B in EXP there exists a $\leq^p_{ctt}$ -reduction M_B, such that $Q(M_B, x) \not\subset Q(M_B, y)$ and $Q(M_B, y) \not\subset Q(M_B, x)$, whenever $x \neq y$*

Proof: Let $\{M_i\}_i$ be an enumeration of $\leq^p_{ctt}$ -reductions. We construct a set W as follows. On input $<i, x>$ compute $Q(M_i, <i, x>)$. If there is a $y < x$ such that either:

1. $Q(M_i, <i, x>) \subset Q(M_i, <i, y>)$ or
2. $Q(M_i, <i, y>) \subset Q(M_i, <i, x>)$.

Then put $<i, x>$ in W iff $<i, y> \notin W$, otherwise put $<i, x>$ in W iff $x \in B$.

It is easy to see that W is in *EXP*, so there exists a $\leq^p_{ctt}$ -reduction from W to A, say M_j. For this reduction it follows that if $x \neq y$ then

$Q(M_j, <j, x>) \not\subset Q(M_j, <j, y>)$ and $Q(M_j, <j, y>) \not\subset Q(M_j, <j, x>)$, so $M'(x) = M_j(<j, x>)$ computes a $\leq_{ctt}^p$ -reduction from B to A with the required property. □

From this lemma we get:

Theorem 6.9 *Given a recursive non-decreasing function $g(n)$ with $\lim_{n\to\infty} g(n) = \infty$. There exists a $g(n)$-sparse set S in EXP such that for any $\leq_{ctt}^p$ -hard set A for EXP the set $A - S$ is no longer $\leq_{ctt}^p$ -complete.*

Proof: We let the numbers $b(i)$ again be sufficiently far apart to guarantee sparseness of S if we put one string in S for each $b(i)$ and such that $0^{b(i)}$ is again easy to recognize. Furthermore we let $2 \times (b(i-1))^{i-1} < b(i)$ to avoid confusion later on. Now we find out whether one of the strings in $\bigcup_{|y|\leq b(i)} Q(M_i, <0^{b(i)}, y>)$ has length $\geq b(i)/2 - 1$. If it has, then we put the least such string in S.

S is exponential time computable since to decide membership of a string x in S we search for a $0^{b(i)}$ such that $b(i) \leq 2\times|x|$ and $|x| < (b(i))^i$. (There can only be one candidate) Now compute the query sets on the at most $2^{2|x|}$ different y in time $\leq 2^{2|x|} \times (2|x|)^i + i$ and see if x is the least string of the right length in the union of these sets.

$A - S$ is not $\leq_{ctt}^p$ -complete since if it were, then one of the $\leq_{ctt}^p$ -reductions of $0^* \times \Sigma^*$ would behave as predicted by Lemma 6.8. However this reduction, say M_i on input $<0^{b(i)}, y>$ finds one of the queried strings not in $A - S$ for some y and hence must reject, a contradiction. □

Conjunctive and disjunctive truth table reducibilities are complementary on *EXP* in that if a set is conjunctive truth table reducible to a set A then it is disjunctive truth table reducible to $\overline{A}$. So we find,

Corollary 6.10 *Given a recursive non-decreasing function $g(n)$ with $\lim_{n\to\infty} g(n) = \infty$. There exists a $g(n)$-sparse subexponential time computable set S such that for any $\leq_{dtt}^p$ -complete set A for EXP the set $A \cup S$ is no longer $\leq_{dtt}^p$ -complete.*

and

Corollary 6.11 *Given a recursive non-decreasing function $g(n)$ with $\lim_{n\to\infty} g(n) = \infty$. There exists a $g(n)$-sparse set S in EXP such that for any $\leq_{dtt}^{p}$ -hard set A for EXP the set $A \cup S$ is no longer $\leq_{dtt}^{p}$ -complete.*

Proof: Let K be the standard $\leq_m^p$ -complete set for *EXP*. If $K \leq_{ctt}^{p} A$ via M_i then:

$$\begin{array}{rlcl} & x \in K & \Leftrightarrow & Q(M_i, x) \subset A \\ \text{or:} & x \notin K & \Leftrightarrow & Q(M_i, x) \cap \overline{A} \neq \emptyset \end{array}$$

Now $\overline{K} \in EXP$ so $\overline{K} \leq_m^p K$ say via f.

$$\begin{array}{rlcccl} \text{so:} & x \in \overline{K} & \Leftrightarrow & f(x) \in K & \Leftrightarrow & Q(M_i, f(x)) \subset A \\ \text{or:} & x \notin \overline{K} & \Leftrightarrow & f(x) \notin K & \Leftrightarrow & Q(M_i, f(x)) \cap \overline{A} \neq \emptyset \\ \text{or:} & x \in K & \Leftrightarrow & f(x) \notin K & \Leftrightarrow & Q(M_i, f(x)) \cap \overline{A} \neq \emptyset \end{array}$$

So $\overline{A}$ is $\leq_{dtt}^{p}$ -hard via $M_i(f(x))$. Along the same lines: If A is $\leq_{dtt}^{p}$ -hard then $\overline{A}$ is $\leq_{ctt}^{p}$ -hard. So A is $\leq_{dtt}^{p}$ -complete(hard) iff $\overline{A}$ is $\leq_{ctt}^{p}$ -complete (hard) for *EXP*. But if $\overline{A}$ is $\leq_{ctt}^{p}$ -complete (hard) then there exists a $g(n)$ sparse subexponential (exponential) time computable set S such that $\overline{A} - S$ is no longer $\leq_{ctt}^{p}$ -complete (hard), and then $A \cup S$ is no longer $\leq_{dtt}^{p}$ -complete (hard). □

As in the case of $\leq_m^p$ -reductions in the case of complete sets we can let S come arbitrarily close to being polynomial time computable, and as in the case of $\leq_m^p$ -reductions polynomial time computability is exactly the cut-off point. But first we need a definition:

Definition 6.12 *Let A be a set. For a string x we let $A^{[x]}$ stand for the* x section of A, *i.e. the set $\{<x, z> \mid <y, z> \in A\}$. In this case z may also be $<z_1, \ldots, z_n>$, so that we get the set $\{<y, z_1, \ldots, z_n> \mid y = x$ and $<y, z_1, \ldots, z_n>$ in A.*

Theorem 6.13 *For any set A that is $\leq_{ctt}^{p}$ -hard for EXP and any $p(n)$-sparse polynomial time computable set S, the set $A - S$ remains $\leq_{ctt}^{p}$ -hard for EXP.*

Proof: We construct a set $W \in EXP$, consisting of pairs $<i, x>$ such that there exists an i for which K reduces $\leq_m^p$ to $W^{[i]}$ and $W^{[i]}$ reduces $\leq_{ctt}^p$ to $A - S$. Then $A - S$ is $\leq_{ctt}^p$ -hard. As before $\{M_i\}_i$ is an enumeration of polynomial time $\leq_{ctt}^p$ -reductions. W is constructed according to the following rules:
On input $<i, x, z>$ with $0 \leq z \leq p(|<i, x, z>|^i + i) + 1$ compute the union of the query sets $U_i(i, x, z) = \bigcup_{z' < z} (Q(M_i, <i, x, z'>) \cap S)$. According to whether $Q(M_i, <i, x, z>) \cap S) \subseteq U_i(i, x, z))$ do one of the following two actions:

1. If $(Q(M_i, <i, x, z>) \cap S) = \emptyset$ or $Q(M_i, <i, x, z>) \cap S) \subseteq U_i(i, x, z))$ then $<i, x, z> \in W$ iff $x \in K$ else
2. $<i, x, z> \in W$.

First W is exponential time computable and hence one of the machines, say M_w is a polynomial time $\leq_{ctt}^p$ -reduction from W to A. Next there is a $\leq_{ctt}^p$ -reduction from $W^{[w]} = \{<y, x, z> \mid y = w \text{ and } <y, x, z> \in W\}$ to $A - S$.
Proof: The reduction works as follows. On input $<w, x, z>$ reject if $z > p(|<w, x, z>|^w + w) + 1$. Otherwise compute $U_w(w, x, z)$ and $Q(M_w, <w, x, z>) \cap S$. If $(Q(M_w, <w, x, z>) \cap S) \not\subseteq U_w(w, x, z)$ then accept, otherwise accept only if $Q(M_w, <w, x, z>) - S \subseteq A - S$. □

Finally we show that there is a many one reduction from K to $W^{[w]}$.
Proof: On input x compute the least z such that $Q(M_w, <w, x, z>) \cap S) = \emptyset$ or $(Q(M_w, <w, x, z>) \cap S) \subseteq U_w(w, x, z)$ and let $f(x) = <w, x, z>$. Because S is a sparse set with density $p(n)$, action 2 in the construction of W above can only happen $p(|<w, x, z>|^w + w)$ many times, so this least z exists and is computable in polynomial time. □
□

In the proof of this theorem we in fact showed the existence of a special set W and a reduction of W to the set $A - S$ that avoids queries about strings in S. By the definition of W we force that queries about strings in S must get a positive answer, and hence if the same queries are asked on larger inputs than the answer to those queries is already known and we can let such a larger input depend only on queries outside S. It is not very difficult now to define a set W that forces a reduction from W to A to consider queries about S as *negatively* answered and so the same strategy yields:

Theorem 6.14 *For any set A that is $\leq^p_{dtt}$ -hard for EXP and any $p(n)$-sparse polynomial time computable set S, the set $A - S$ remains $\leq^p_{dtt}$ -hard for EXP.*

Proof: We now build the set W according to the rules:
On input $<i,x,z>$ with $0 \leq z \leq p(|<i,x,z>|^i + i) + 1$ we again compute the union of the query sets $U_w(i,x,z) = \bigcup_{z'<z} (Q(M_i, <i,x,z'>) \cap S)$. If $(Q(M_i, <i,x,z>) \cap S) = \emptyset$ or $(Q(M_i, <i,x,z>) \cap S) \subseteq U_i$ then $<i,x,z> \in W$ iff $x \in K$ else $<i,x,z> \notin W$. We thus obtain a $\leq^p_{dtt}$ -reduction from K to $A - S$ along the same lines. □

The next logical step would be to prove that the same statement holds for $\leq^p_{btt}$ -reductions. This however seems to require more involved techniques. As things stand we only have a proof for $\leq^p_{2-tt}$ -reductions and it is rather lengthy.

Theorem 6.15 *Let A be $\leq^p_{2-tt}$ -complete for EXP and S a polynomial time computable $p(n)$-sparse set. The set $A - S$ is still $\leq^p_{2-tt}$ -complete for EXP*

Proof: Again we construct an exponential time computable set W. Let $\{M_i\}_i$ be an enumeration of all $\leq^p_{2-tt}$ -reductions. This time we have the following goal in mind for the reduction M_w from W to A. For some fixed polynomial q depending only on p we want for each x that there exists a pair of strings $(<w,x,z>, <w,x,z'>)$ for $0 \leq z \leq q(n) + 1$ such that $\|Q(M_w, <w,x,z>) \cap S\| + \|Q(M_w, <w,x,z'>) \cap S\| < 2$. That is either for one of the sets both queried strings are outside S, or they have at most one string in S in common. Then we will construct W such that one of the two strings $<w,x,z>$, $<w,x,z'>$ is in W iff $x \in K$ and its membership in W can be decided on the basis of membership in A of two strings in $(Q(M_w, <w,x,z>) \cup Q(M_w, <w,x,z'>)) - S$. So $A - S$ remains $\leq^p_{2-tt}$ -complete.

Consider the i^{th} reduction M_i, and a string x. For z an integer less than or equal to $3 \times \binom{p(|<i,x,z>|^i + i)}{2} + p(|<i,x,z>|^i + i) + 1$ we will decide whether to put strings of the form $<i,x,z>$ in or out of W. *All* other strings are not element of W. Without loss of generality we assume that the strings $<i,x,z>$ are of the same length for all z considered.

We treat three different cases according to $\|Q(M_i, <i,x,z>) \cap S\|$.

- First for any of the z considered: If $Q(M_i, <i,x,z>) \cap S = \emptyset$ then $<i,x,z> \in W$ iff $x \in K$.

- Next we make sure that the number of string for which $Q(M_w, <w,x,z>) \subset S$ is limited. There are at most $\binom{p(|<i,x,z>|^i+i)}{2}$ different pairs of strings in S. Hence if we assume $Q(M_i, <i,x,z>) \subset S$ for $0 \leq z \leq 3\binom{p(|<i,x,z>|^i+i)}{2} + 1$, then we must find $z_1 < z_2 < z_3$ such that $Q(M_i, <i,x,z_1>) = Q(M_i, <i,x,z_2>) = Q(M_i, <i,x,z_3>)$. Let $z_1 < z_2 < \ldots$ be such that $Q(M_i, <i,x,z_1>) = Q(M_i, <i,x,z_2>) = \ldots$ and $Q(M_i, <i,x,z_1>) \subset S$.

  - We evaluate M_i on input $<i,x,z_1>$ with all 4 possible strings $(00, 01, 10, 11)$ written on the ANSWER-tape, and let $<i,x,z_1> \in W$ iff the number of strings on which M_i rejects is *greater* than the number of strings on which M_i accepts. This leaves at most two possible answer-strings for the strings in $Q(M_i, <i,x,z_1>)$.
  - We evaluate M_i on input $<i,x,z_2>$ on the remaining two possible answer-strings and let $<i,x,z_2> \in W$ iff M_i rejects with both strings written on the ANSWER-tape. Now there is at most one possible answer-string left.
  - We evaluate M_i on input $<i,x,z_3>$ with the last remaining answer-string written on the ANSWER-tape , and let $<i,x,z> \in W$ iff M_i rejects with this string on the ANSWER-tape.
  - if $j > 3$ then $<i,x,z_j> \notin W$

  We can now safely assume that if M_w is the reduction from W to A then there are *less* than $3 \times \binom{p(|<w,x,z>|^w+w)}{2}$ different z for which $Q(M_w, <w,x,z>) \subset S$.

- The final and most elaborate task is to define membership of $<i, x, z>$ for z such that $\|Q(M_i, <i, x, z>) \cap S\| = 1$. We observe that if $z_1 < z_2 < z_3 < \ldots < z_k$ is a sequence such that $\|Q(M_i, <i, x, z_j>) \cap S\| = 1$ and $Q(M_i, <i, x, z_j>) \cap S \neq Q(M_i, <i, x, z_{j}>) \bigcap S$ for $j \neq j'$ then $k \leq p(|<i, x, z>|^i + i)$.

  So let $<i, x, z>$ be such that $\|Q(M_i, <i, x, z>) \cap S\| = 1$. First we treat the case where $\forall z' < z(Q(M_i, <i, x, z>) \cap S) \neq Q(M_i, <i, x, z'>) \cap S$

  Let $Q(M_i, <i, x, z>) = \{y_1, y_2\}$.

  We simulate $M_i(<i, x, z>)$ with all possible strings a of length 2 written on the ANSWER-tape, and act differently depending on the outcome of these simulations. There are at most sixteen possible outcomes, corresponding to the following four cases.

  1. If $M_i(<i, x, z>)$ always accepts, we let $<i, x, z> \notin W$ and if $M_i(<i, x, z>)$ always rejects, we let $<i, x, z> \in W$. (Hence for the reduction of W to A this case cannot occur. This covers 2 of the 16 cases)
  2. If there is only *one* string of a for which $M_i(<i, x, z>)$ accepts with a on the ANSWER-tape, then $<i, x, z> \in W$, and if there is only *one* a for which $M_i(<i, x, z>)$ rejects, then $<i, x, z> \notin W$ (This way we fix *both* $\chi_A(y_1)$ and $\chi_A(y_2)$. This covers 8 of the 16 cases).
  3. If $M_i(<i, x, z>)$ only depends on one query ($M_i(<i, x, z>)$ is in fact a $\leq^p_{1-tt}$-reduction) then let y_1 be the string on which $M_i(<i, x, z>)$ depends. There are two subcases:
     (a) If y_1 in S then we put $<i, x, z>$ in W iff $M_i(<i, x, z>)$ accepts with 00 written on the ANSWER-tape.
     (b) If $y_1 \notin S$ then we put $<i, x, z> \in W$ iff $x \in K$.

     (this covers 4 of the 16 cases plus the case where $\|Q(M_i, <i, x, z>)\| = 1$.)
  4. In the remaining two cases $M_i(<i, x, z>)$ with $a (= a_1 a_2)$ written on the ANSWER-tape accepts iff $a_1 = a_2$ or $a_1 \neq a_2$. In case $=$ we let $<i, x, z> \in W$ and in case $\neq$ we let

$<i,x,z> \notin W$. (Then in both cases $M_i(<i,x,z>)$ accepts iff $y_1 \in A \Leftrightarrow y_2 \in A$.)

Next assume that $\exists z' < z$ such that $Q(M_i, <i,x,z'>) \cap S = Q(M_i, <i,x,z>)$. Then find the least such z'. We perform the case analysis above for $<i,x,z>$. If this analysis falls into case 3b then we let $<i,x,z> \notin W$. Otherwise we let $<i,x,z> \in W$ iff $x \in K$.

Now we show that for the reduction M_w of W to A the set $W^{[w]}$ reduces $\leq^p_{2-tt}$ to $A - S$ via M_w.

Proof:

On input $<w,x,z>$ reject if $z > 3 \times \binom{p(|<w,x,z>|^w + w)}{2} + p(|<w,x,z>|^w + w) + 1$. Next compute $Q(M_w, <w,x,z>)$. There are three cases.

1. $Q(M_w, <w,x,z>) \cap S = \emptyset$. In this case we can query both strings in $Q(M_w, <w,x,z>)$ and accept iff M_w accepts with the answer-string found.

2. $Q(M_w, <w,x,z>) \subseteq S$. Now by construction of W there can be at most one $z' < z$ such that $Q(M_w, <w,x,z>) = Q(M_w, <w,x,z'>)$. Find out if such a z exists. If it does then find the at most two possible answer-strings for $Q(M_w, <w,x,z>) = Q(M_w, <w,x,z'>)$, and simulate $M_w(<w,x,z>)$ with those strings written on the ANSWER-tape. Accept iff M_w rejects with both strings on the ANSWER-tape.

3. $\|Q(M_w, <w,x,z>) \cap S\| = 1$. Again there are two possibilities

   (a) If there is no $z' < z$ with $Q(M_w, <w,x,z'>) \cap S = Q(M_w, <w,x,z>) \cap S$. Then compute $M_w(<w,x,z>)$ with $A - S$ as oracle and accept according to the answer-string and the rules for W above.

   (b) If there is a $z' < z$ with $Q(M_w, <w,x,z'>) \cap S = Q(M_w, <w,x,z>) \cap S$, then find the least such z'. Perform the case analysis on the pair $<w,x,z'>$. If this falls into case 3b then reject. Otherwise compute $\chi_W(z')$ with the rules for W above and $M_w(<w,x,z'>)$. Query $(Q(M_w, <w,x,z'>) \cup Q(M_w, <w,x,z>)) - S$.

Now either $\chi_W(z)$ fixes $\chi_A(Q(M_w, <w, x, z'>) \cap S) = \chi_A(Q(M_w, <w, x, z>) \cap S)$ or $\chi_A(Q(M_w, <w, x, z'>) \cap S) = \chi_A(Q(M_w, <w, x, z>) \cap S) = \chi_A(Q(M_w, <w, x, z'>) - S)$. Both cases give enough information to compute $\chi_W(<w, x, z>)$.

□

Next we show that K is $\leq_m^p$ -reducible to $W^{[w]}$

Proof: On input x we compute the query sets $Q(M_w, <w, x, z>)$ for $z \leq 3 \times \binom{p(|<w,x,z>|^w + w)}{2} + p(|<w, x, z>|^w + w) + 1$ until we either find a string $<w, x, z>$ for which $Q(M_w, <w, x, z>) \cap S = \emptyset$ or $\exists z' < z : Q(M_w, <w, x, z'>) \cap S = Q(M_w, <w, x, z>) \cap S = y$ and then query $<w, x, z>$ and accept iff a 1 is written on the ANSWER-tape. □

Since $K \leq_m^p W^{[w]}$ and $W^{[w]} \leq_{2-tt}^p A - S$ we have that $A - S$ is $\leq_{2-tt}^p$ -complete. □

For $\leq_{3-tt}^p$ the case analysis quickly blows up. We would have to find a more general property behind the reductions before we could attack the problem for $\leq_{btt}^p$ -reductions. For general truth table reductions such an attack would not even work. The proof method above relativizes, and we can show that there exists an oracle set A relative to which *EXP* has a $\leq_{tt}^p$ -complete tally set T. Then $T - 0^* = \emptyset$ which cannot be complete.

Lemma 6.16 *If EXP* $\subseteq P/poly$ *then there exists a tally set* T *that is* $\leq_{tt}^p$ *-complete for EXP*

Proof: It is well known that A is in $P/poly$ iff $A \leq_{tt}^p T$ for some tally set T. So from our hypothesis this gives us a tally set T, that is truth table *hard* for *EXP*. We will show how to construct a tally set T' (from T) that is complete for *EXP*.

From the fact that there exists a tally set T that is hard for *EXP* we get a truth-table reduction, say by machine M_i from K to T. Now fix n and consider all strings of length n. W.l.o.g. we may assume that M_i queries on input x of length n, always the same strings to T, namely $y_1, \ldots, y_{n^i}$, where $y_i = 0^i$. The idea

is to find the minimal (in some way) setting of the y_j's in T such that $x \in K$ iff $M_i(x)$ accepts with this setting. Let x_j indicate the j^{th} string of length n in lexicographic order. Let $P_j = \{a : |a| = n^i(M_i(x_j)$ with a written on the ANSWER-tape accepts iff $x_j \in K)\}$. P_j codes exactly those tally sets T' that, when used as oracle for $M_i(x_j)$ let M_i compute the correct answer for $x_j \in K$. Note here that $\|P_j\| \leq 2^{n^i}$ for $0 \leq j < 2^n$. Set $P' = \bigcap_{i=1}^{2^n} P_i$ and let p_i be the i^{th} projection of the tuple y, where y is the minimal $y \in P'$. Put $0^{<n,i>}$ in T' iff $p_i = 1$. Obviously T' is tally and from the construction it is clear that T' is computable in exponential time. From the fact that T exists we get that T' exists and that $K \leq_{tt}^{p} T'$
□

Theorem 6.17 *There exists an oracle A such that there exists a $\leq_{tt}^{p^A}$ -complete set B for EXP^A such that $B - S$ where S is sparse and polynomial time computable is not $\leq_{tt}^{p^A}$ -complete for EXP^A.*

Proof: Wilson [Wil85] showed the existence of an oracle A where $EXP^A \subseteq P^A/poly$. Using this oracle together with Lemma 6.16 we get that there exists a tally set T that is complete for EXP^A. Setting $B = T$ and $S = \{0\}^*$, we get that $B - S = \emptyset$ and $\emptyset$ is not $\leq_{tt}^{p^A}$ -complete for EXP^A □

Corollary 6.18 *If for all $\leq_T^p$ -complete sets A for EXP, the set $A - S$, for a sparse set $S \in P$, is still $\leq_T^p$ -complete then $EXP \not\subseteq P/poly$.*

The interresting thing is that this corollary ties together the quest for proving a lower bound for *EXP* and the structure of complete sets. The corollary says that if complete sets are robust, then they are also hard to compute.

6.3 Splittings

In this section we want to investigate to what extend one can *split EXP* -complete sets. A splitting of an r.e. (exptime) set is the construction of

2 r.e. (exptime) sets $A_0, A_1 \subseteq A$ such that $A_0 \bigcap A_1 = \emptyset$ and $A_0 \bigcup A_1 = A$. One of the things to look at is: can this splitting be done so that the subsets both have the same information as A. For complete sets this would mean that the complete set can be split into subsets that are itself again complete. These type of questions have been studied in a recursion theoretical setting by Ladner [Lad73]. He observed that there exist sets that are non splittable or non *mitotic* as he called them. The recursion theoretical definition is as follows:

Definition 6.19 *An r.e. set A is called (m-)mitotic iff there exist r.e. sets A_1 and A_0 such that:*

1. $A_1 \subseteq A,\ A_0 \subseteq A,\ \ A_1 \bigcap A_0 = \emptyset,\ \ A_1 \bigcup A_0 = A$
2. $A \equiv_{(m)T} A_1 \equiv_{(m)T} A_0$

Note here that point 2 in the definition can be weakened, in the case of $\leq_T^p$ -reductions, to $A_1 \equiv_T A_0$. To see this, note that $A \leq_T^p A_1 \oplus A_0$ and $A_1 \oplus A_0 \leq_T^p A_1$. To reduce A_1 to A the reduction queries, on input x, whether x is in A. If this is not the case it rejects straight out, otherwise it starts enumerating A_1 and A_0 since x must be in one of them.

Later, Ambos-Spies [AS84] studied the complexity theoretical variant of mitotic sets and introduced the term *p-mitotic* sets. It is not clear how to define mitoticity in the complexity theoretical setting. Ambos-Spies introduced four definitions; two for the Turing reductions and two for the many-one reductions. One way of doing it would be to change point 2 in Definition 6.19 into $A \equiv_T^p A_1 \equiv_T^p A_0$ and since we are interested in complete sets for complexity classes we could demand that A_0 and A_1 should be in the complexity class under consideration. One problem is that this definition cannot be weakened to $A_1 \equiv_T^p A_0$. (Take for example $\Sigma^* = A \bigcup \overline{A}$ for some Turing complete set A for *EXP* . Now A as well as $\overline{A}$ are both in *EXP* and split Σ^* and are Turing equivalent but obviously do not Turing reduce to Σ^*.) To get around this problem Ambos-Spies defined p-mitotic in the following way:

Definition 6.20 *A recursive set A is p-m(T)-mitotic if there is a set $B \in P$ such that $A \equiv_{m(T)}^p A \bigcap B \equiv_{m(T)}^p A \bigcap \overline{B}$.*

When using this definition the problem of reducing A_1 to A is settled for the Turing case. Namely, x is in A_1 iff x is in B and x is in A. One disadvantage of this definition however is that the requirement that the splitting has to be polynomial time computable seems too strong. In order to capture this feeling, we also want to look at the definition discussed above. Note here also that since our main interest concerns complete sets we will not have the trouble that A_0 (or A_1) does not reduce to A (This because A is complete)

Definition 6.21 *An recursive set A is called weakly-p-m(T) mitotic iff there exist re sets A_1 and A_0 such that:*

1. $A_1 \subseteq A,\ A_0 \subseteq A,\ A_1 \bigcap A_0 = \emptyset,\ A_1 \bigcup A_0 = A$
2. $A \equiv^p_{m(T)} A_1 \equiv^p_{m(T)} A_0$

One of the questions that arises, is: are $\leq^p_m$ -complete sets for *EXP* (weakly) p-m mitotic? In order to answer this question we first take a good look at the complete sets for *RE*. There it is known, due to Myhill, that the $\leq^p_m$ -complete sets are all isomorphic. Now using the fact that K, the standard $\leq^p_m$ -complete set for *RE* is m-mitotic and that this property is preserved under isomorphisms it follows that all $\leq^p_m$ -complete sets are m-mitotic. Unfortunately, it is not known whether the $\leq^p_m$ -complete sets for *EXP* are p-isomorphic but it is known that they are all 1-1, length increasing equivalent [Ber77, GH89, Wat87a]. It turns out that this is sufficient to prove that they are weakly-p-m-mitotic.

Theorem 6.22 *All $\leq^p_m$ -complete sets for EXP are weakly-p-m-mitotic.*

Proof: Let A be a $\leq^p_m$ -complete set for *EXP*. Let $A \oplus A \leq^p_m A$ via f that is 1-1 and length increasing. Set A_0 to be $\{y \mid \exists\ 0x,\ x \in A,\ y = f(0x)\}$. Since f is 1-1 and length increasing it is obvious that A_0 is in *EXP*, furthermore it is also $\leq^p_m$ -complete because $x \in A$ iff $f(0x) \in A_0$, and $A_0 \subseteq A$. Now set $A_1 = A - A_0$. Obviously, $A_0 \bigcup A_1 = A$ and $A_0 \bigcap A_1 = \emptyset$. Now it remains to be shown that A_1 is also $\leq^p_m$ -complete. Let $A^1 = \{1x \mid x \in A\}$. Note that $A^1 \subseteq A \oplus A$ and is $\leq^p_m$

-complete. Since f is 1-1, $1x \in A^1 \Rightarrow f(1x) \in A \Rightarrow f(1x) \in A - A_0$, and $1x \notin A^1 \Rightarrow 1x \notin A \oplus A \Rightarrow f(1x) \notin A \Rightarrow f(1x) \notin A - A_0$. $\square$

So for *EXP* the 1-1 length increasing property seems to be enough in order to get weakly-p-m-mitoticity. For *NEXP* the situation is somewhat different because we do not know whether we have length increasing reductions to complete sets. But we do have the 1-1 property and some sort of honesty namely that the reductions are not more than exponential length decreasing, i.e. $2^{|f(x)|} > |x|$ [GH89]. The main problem however is, that when applying the same proof as above, the set difference used to define $A_1 = A - A_0$ is not known to be in *NEXP* because it is not known if *NEXP* is closed under complementation (and thus under set difference). However we can prove something that at a first glance looks hopeful in order to prove weakly-p-m-mitoticity for *NEXP* -complete sets.

Theorem 6.23 *Every $\leq_m^p$ -complete set A for NEXP can be split into infinitely many disjoint subsets $A_1, A_2 \ldots$ such that $\bigcup_{i=0}^{\infty} A_i = A$, such that for all i, $A_i \subseteq A$ and A_i is $\leq_m^p$ -complete for NEXP.*

Proof: We start the same way as in the *EXP* case. So let $A \oplus A \leq_m^p A$ via f that is 1-1 and exponentially honest. Set $A_0 = \{y \mid \exists\ 0x, y = f(0x), y \in A\}$. Note that it is equivalent, in the definition of A_0, to say that $x \in A$ or $y \in A$, because f is a many-one reduction. Now A_0 is in *NEXP*: on input y guess the $0x$ such that $f(0x) = y$. This can be done in nondeterministic exponential time because $|x| < 2^{|y|}$, by the exponential honesty of f. Now accept y iff $y \in A$. We define A_1 in a similar way: $A_1 = \{y \mid \exists\ 1x, y = f(1x), y \in A\}$. We have now two complete sets A_0 and A_1 and some leftover of A namely $T_0 = A - (A_0 \bigcup A_1)$. At this point we repeat this procedure with A_0 resulting in A_{00} and A_{01} and again some leftover T_1. Repeating this we get an infinite sequence of sets A_{0^i1} and a set $T = \bigcup_{i=0}^{\infty} T_i$ so that $(\bigcup_{i=0}^{\infty} A_0^i 1) \bigcup T = A$. Since T is countable (it is a subset of $I\!N$) we can add the i^th element of T to A_{0^i1} resulting in a sequence A'_{0^i1} satisfying the properties of the theorem. $\square$

Although this looks hopeful, the following example shows that the infinite version of mitoticity and mitoticity can be independent of each other. Ladner [Lad73] showed the existence of *non mitotic* sets, together with the following observation this yields the somewhat bizarre existence of a set that cannot be split into two parts but can be split in infinitely many parts of the same complexity.

Observation 6.24 *every r.e. set A can be split into infinitely many disjoint r.e. subsets $A_1, A_2, \dots$ of A such that they remain in the same Turing degree as A.*

Proof: It is well known that every r.e. set A has an infinite subset B that is recursive. Let B be such an infinite recursive subset of A. Since B is recursive and infinite, it is (recursively) isomorphic to Σ^*. So we can code A into B. I.e. let f be the isomorphism between Σ^* and B, define $A' = \{f(x) \mid x \in A\}$. Obviously A' is r.e and $A' \equiv_T (A - B)$. Furthermore there exists a T_1 such that $A = (A - B) \bigcup A' \bigcup T_1$. Now using the same "divide and split" trick as in the previous theorem we get the desired sequence of subsets. $\Box$

We try to follow the same line now as Ladner [Lad73] and try to prove that there exist non (weakly-p-m) mitotic sets in *EXP*. We succeed in this and can also prove that those sets can be $\leq^p_{3-tt}$ -complete. Note that this also proves that the same result is true for p-m mitoticity.

Theorem 6.25 *There exists a set A in EXP that is non-weakly-p-m-mitotic and $\leq^p_{3-tt}$ -complete.*

Proof: In order to prove this we prove the following: There exists a set A so that for all sets $A_0, A_1 \in EXP$, that split A, $A_0 \not\equiv^p_m A_1$. It is enough to prove this for only exptime computable splittings (i.e. A_i is in *EXP*) in order to get non weakly-p-m-mitoticity. This is because for a weakly-p-m-mitotic set B in *EXP* the parts B_0 and B_1 are in *EXP* because they are $\equiv^p_m$ to B. Let $M_{i'}$ be an enumeration of exponential time machines that run in time $2^{n^i} + i$ and f_j an enumeration of polynomial time many-one reductions.

In order to construct A we have the following set of requirements: For all $i0', i1', i0, i1$, if $L(M_{i0'})$ and $L(M_{i1'})$ split A then either $L(M_{i1'}) \not\leq^p_m L(M_{i0'})$ via f_{i1} or $L(M_{i0'}) \not\leq^p_m L(M_{i1'})$ via f_{i0}.

We introduce a function b in order to have a set of strings to diagonalize over. Let $b(0) = 1$ and $b(i) = (b(i-1)^{i-1}) + 1$. The idea is to put 3 copies of K into A and make sure that from the pairs, $<i,x>$ $(i = 1,2,3)$, always at least one of them is in A if $x \in K$. A becomes then $\leq^p_{3-tt}$ (in fact $\leq^p_{3d}$) -complete by the following reduction from K to A: $x \in K$ iff $\exists i \leq 3$ such that $<i,x>$ in A. On the other hand, we can leave out at most 2 of the pairs $<i,x>$ in order to destroy the mitoticity.
stage 0 $A = \emptyset$

stage n:

Let $n = <i0', i1', i0, i1>$ and let $b = b(i-1)^{i-1}+1$. We will make sure that all strings of length $\leq b$ in A will not be changed by this stage and because $b(i)$ is strictly increasing, not by later stages. Consider $f_{i0}(0^{b(n)}) = y_0$ and $f_{i1}(0^{b(n)}) = y_1$. Set A to be $A \bigcup \{<i,x> \mid i = 1,2,3 \text{ and } x \in K\}$. We assume that the pairing function does not output strings that start with a 0 this in order to be able to let strings of the form $0^{b(i)}$ out of A without disturbing the completeness. We have several cases to consider.

1. $y_0 = 0^{b(n)}$ or $y_1 = 0^{b(n)}$ in this case we put $0^{b(n)}$ into A.

2. $y_0 = y_1$, and not case 1. We have two subcases:

   (a) $|y_0| > b$. Leave y_0 out of A and put $0^{b(n)}$ in A.

   (b) $|y_0| \leq b$. Put $0^{b(n)}$ in A iff $y_0 \notin A$

3. $y_0 \neq y_1$, and not case 1. We have 3 subcases:

   (a) Both $|y_0|$ and $|y_1|$ are bigger than b. Leave both y_0 and y_1 out of A and put $0^{b(n)}$ in A.

   (b) One of the y's is bigger than b and the other is smaller. W.l.o.g. let $|y_0| > b$ and $|y_1| \leq b$. We leave y_0 out of A and have two subcases:

      i. $y_1 \notin A$. Put $0^{b(n)}$ in A.

      ii. $y_1 \in A$. $0^{b(n)} \in A$ iff $M_{i1'}(y_1)$ accepts.

(c) $|y_0| \leq b$ and $|y_1| \leq b$. We have 3 subcases.

i. $y_1 \notin A$ and $y_0 \notin A$. Put $0^{b(n)}$ in A.

ii. $y_1 \in A$ and $y_0 \notin A$. (The other way around is symmetric) $y_1 \in A$ iff $M_{i1'}(y_1)$ accepts.

iii. $y_1 \in A$ and $y_0 \in A$. $0^{b(n)} \in A$ iff $M_{i0'}(y_0)$ accepts *and* $M_{i1'}(y_1)$ accepts.

end of stage n

This ends the construction of A. The correctness of the construction is an analysis of the the cases in the construction. The cases can be distinguished in 2 types: namely the one where y_0 and y_1 are free i.e. $> b$. In those cases we can diagonalize by putting $0^{b(n)}$ in A and leaving both y_1 and y_0 out of A. In these cases we diagonalize against the many one reductions. In the other cases we are forced to leave y_1 and y_0 in A in order not to destroy the the work done at previous stages. But in these cases we are able to compute in exponential time the splittings we want to diagonalize against. We will show the correctness of case 3(c)iii in the construction. The other cases have a similar proof. So both y_0 and y_1 are fixed and in A. By putting in $0^{b(n)}$ we force y_0 and y_1 for any *possible correct* splitting both to be in either in A_0 or A_1. On the other hand by leaving $0^{b(n)}$ out of A, y_1 has to be in A_i and and y_1 in A_{1-i} $(i = 0, 1)$. Since we are able to compute the splittings we can diagonalize in this case. □

The logical next step would be to prove this result for Turing complete sets. We are not yet able to do this but suspect that there exist Turing complete sets that are not weakly-p-T-mitotic.

An other line of splittings in recursion theory is the existence of a splitting of an r.e. set A in A_0 and A_1 that are incomparable. One example of this is the splitting theorem of Sacks [Sac63] and the time bounded versions by [Lad75]. The next theorem is in a way a counterpart of this. In the original splittings one gets the following structure: A_0 and A_1 are Turing (or many one) incomparable but do reduce to A, thus achieving that A does not reduce to A_0 or A_1. I.e. A_0 and A_1 are strictly below A. In the next theorem the sets A_0 and A_1 are strictly below A but are in the same many one degree. Seen in an other light this theorem can be seen as a generalization of the fact that there exists $\leq^p_{2-dtt}$ -complete sets for *EXP* that are not $\leq^p_m$ -complete [Wat87a, GH89].

Theorem 6.26 *If A is $\leq^p_m$ -complete for EXP then A can be split into A_0 and A_1 such that:*

- $A_0 \equiv^p_m A_1$
- *A_0 and A_1 are $\leq^p_{2-dtt}$ -complete for EXP but not $\leq^p_m$ -complete.*

Proof: Let A be $\leq^p_m$ -complete and K be the standard $\leq^p_m$ -complete set. Since the $\leq^p_m$ -complete sets for *EXP* are 1-1 length increasing equivalent we can construct the following length increasing 1-1 function h from A to A. Let f be the 1-1, l.i. reduction from A to K and g the one from K to A. Let $h(x) = f(g(x))$. We say that x is a root if $h^{-1}(x)$ is undefined and x is on a chain if $h^{-1}(x)$ is defined. One possible way to construct A_0 and A_1 is as follows (the real construction follows later): $A_0 = \{x \mid x \in A$ and x is a root $\} \bigcup \{x \mid x \in A$ and x is on a chain and $h^i(x_r) = x$ and x_r is a root and i is even $\}$
and A_1: $\{x \mid x \in A$ and x is on a chain and $h^i(x_r) = x$ and x_r is a root and i is odd $\}$

Clearly A_0 and A_1 split A, are in *EXP* and $A_0 \equiv^p_m A_1$ via h. A_0 and A_1 are $\leq^p_{2-dtt}$ -complete: $x \in A \iff x \in A_0$ or $x \in A_1 \iff x \in A_0$ or $h(x) \in A_0$. The only thing to do now is to show that A_0 and A_1 are not $\leq^p_m$ -complete. Note here that in order to get the above properties (i.e. splitting of A, the $\equiv^p_m$ and the $\leq^p_{2-dtt}$ -completeness) it doesn't matter if the roots are in A_0 or A_1 as long as it holds that: $x \in A_i$ then $h(x) \in A_{1-i}$ This gives enough freedom to diagonalize against $\leq^p_m$ -reductions. So we are going to construct a set $W \in$ *EXP* so that W not $\leq^p_m$ A_0. Note that then A_1 cannot be $\leq^p_m$ -complete either. Let $M_1, M_2, \ldots$ be an enumeration of $\leq^p_m$ -reductions such that M_i runs in time n^i. We also need a function $b(n)$ to denote the set of strings to diagonalize against M_n. Let $b(0) = 1$ and $b(i+1) = b(i)^i + 1$. W is going to be a subset of 0^*. We construct W, A_0 and A_1 in stages. At stage 0, $W = A_0 = A_1 = \emptyset$

stage n:

We have constructed W, A_0 and A_1 up to strings of length $b(n-1)^{n-1}+1$. We simulate $M_n(0^{b(n)}) = y$. We now have three cases for the construction of W:

1. $|y| \leq b(n-1)^{n-1}$. Put $0^{b(n)}$ in W iff $y \notin A_0$.
2. $\exists\ y' \leq b(n-1)^{n-1}$ s.t. $h^i(y') = y$ and i is even. Put $0^{b(n)}$ in W iff $y' \notin A_0$.
3. Otherwise put $0^{b(n)}$ in W.

   This ends the construction of W.

   Construction of A_0 and A_1. Let $b(n-1)^{n-1} < |x| < b(n)^n + 1$ and $x \in A$.

4. $x \neq y$
   (a) x is a root.
      - $\exists\ i$ s.t. $h^i(x) = y$
         - put x in A_0 iff i is odd.
         - put x in A_1 iff i is even.
      - put x in A_0

   (b) x is on a chain, x_r is the root of x and $h^i(x_r) = x$. Put x in A_0 if i is even and $x_r \in A_0$ otherwise put x in A_1.
5. $x = y$. Put x in A_1.

end of stage n

It is not hard to see that W, A_0 and A_1 are all in EXP. Furthermore A_0 and A_1 split A and $A_0 \equiv^p_m A_1$. It remains to be shown that $W \not\leq^p_m A_0$. Suppose it does via machine M_j, then $M_j(0^b(j)) = y$. For cases 1 and 2 in the above construction, $0^{b(j)} \in W$ iff $y \notin A_0$ and from the construction of A_0 and case 3, $0^{b(j)} \in W$ and $y \notin A_0$. □

Chapter 7

Selfreductions

In this chapter we turn our attention to another form of reduction, the selfreduction. This type of reduction reveals more of the inner structure of a *single* set. The reductions introduced so far made precise the notion A is "easier" to compute than B ($A \leq_r^p B$). In this sense it *ordered* the two sets A and B. The first difference with the previous reduction types is that a selfreduction is only defined for one set and thus as such does not imply an ordering between sets. Intuitively a set A is selfreducible if we can recognize A (in polynomial time) with A as an oracle. Since this is trivialy true for *any* A (on input x just simply query if x is in the oracle) there is a restraint on the type of questions to be queried. A first attempt to restrict the access to the oracle is to demand that x itself should not be part of the queries queried by the oracle machine (i.e. $x \notin Q(M, x)$). Sets A that have this property (i.e. can be recognized in polynomial time with A as oracle with the extra property that x is not queried) are called (polynomial) *autoreducible.* This is a direct generalization from the recursion theoretical autoreducibility notion, where the oracle machine has to be recursive. Since in the following we will only talk about polynomial autoreducibility we will drop the adjective polynomial. We would like to note that the *RE* sets that are recursive autoreducible have been characterized as presicely those sets that are mitotic[Lad73]. This chapter is organized as follows: Section 7.1 contains the main definitions and some basic facts. In section 7.2 we will discuss which sets in *NP* are selfreducible and which ones are probably *not* selfreducible. In section 7.3

we will focus on the complexity of sets that are selfreducible and are p-selective. We will end this chapter with section 7.4, where we will study the relationship between the various notions of selfreducibility on *NP*. The new work is taken from [BvHT] and [BTvEB93].

7.1 Definitions and Facts

Before we go on we give a precise definition of autoreducibility due to Ambos-Spies [AS84]:

Definition 7.1 *Let r be any of the previous defined restrictions. Let $\overline{\text{ID}} = \forall y \in Q(M_i, x) : x \neq y$. Let r' be the restriction obtained by adding $\overline{\text{ID}}$ to the* QUERY-TYPE *set of r. (i.e.* QUERY-TYPE = QUERY-TYPE $\bigcup\{\overline{\text{ID}}\}$*).*

- *We say that A is r-autoreducible if $A \leq^p_{r'} A$.*
- *We say that A is autoreducible if A is T-autoreducible.*

It is simple to see that there are autoreducible sets of *arbitrary* complexity. For any set A, $A \oplus A$ is autoreducible. On the other hand it is not hard to see that there are sets in *EXP* that are not autoreducible. For a full discussion of autoreducibility and mitoticity and the structure of autoreducible sets we would like to refer the interested reader to Ambos-Spies [AS84].

It would be interesting to determine which sets in *NP* are autoreducible and which ones (if any) are not. We note immediately that *every* set in P is autoreducible, so the existence of a non-autoreducible set in *NP* would imply $P \neq NP$.

As a partial answer to questions of the latter type we show that every $\leq^p_T$ -complete set A for *NP* is autoreducible. Before we prove this we want to refine the notion of autoreducibility a bit. We want to refine it in the sense that not only x is not allowed in the query tree but also strings that are "bigger" in some ordering. First we define what we mean by "some ordering". The following definition is due to Ko [Ko83]

Definition 7.2 *A partial ordering $\prec$ on Σ^* is polynomially well-founded and length-related (polynomially related) if there is a polynomial p such that:*

1. *$x \prec y$ can be decided in $p(|x| + |y|)$ steps,*
2. *$x \prec y$ implies $|x| \leq p(|y|)$ for all x, y in Σ^* and*
3. *the length of a $\prec$ -decreasing chain is shorter than p of the length of its maximum element.*

As an example of a polynomial related partial ordering we mention the length of the strings, i.e. $x \prec y$ iff $|x| < |y|$. We assume an enumeration of polynomial related partial orderings. Let $\prec_i$ be the i^{th} ordering in such an enumeration. Note we do not demand the enumeration to be effective, although recursive enumerations exists.

Definition 7.3 *Let $i = <k, l>$,* DECREASING $= \forall y \in Q(M_k, x) : y \prec_l x$ *and r be any of the previous defined restrictions. Let r' be the restriction obtained by adding* DECREASING *to the* QUERY-TYPE *set of r.*

- *We say A is r-selfreducible if $A \leq^p_{r'} A$.*
- *We say A is selfreducible if A is T-selfreducible.*

Note that selfreducibility is a form of autoreducibility. Ko observed that selfreducibility does not allow sets of arbitrary complexity as follows from the following theorem:

Theorem 7.4 ([Ko83]) *If A is selfreducible then $A \in PSPACE$.*

Ko observed that disjunctive (and conjunctive) selfreducibility imply further restrictions of the complexity of the set.

Theorem 7.5 ([Ko83])
- *If A is disjunctive-selfreducible then $A \in NP$.*
- *If A is conjunctive-selfreducible then $A \in co\text{-}NP$.*

In a similar vein we show that if a set is parity selfreducible then the set is in $\oplus P$.

Theorem 7.6 *If A is $\oplus$-selfreducible then $A \in \oplus P$.*

Proof: Let the parity selfreduction of A be witnessed by machine M_s. We now construct a nondeterministic Turing machine M_p, with the property that $x \in A$ iff accept$(M_p(x))$ is odd. Let M_p be the machine that simulates M_s on input x guesses nondeterministic a query y and recursively continues on y. An easy inductive argument shows that $x \in A$ iff accept$(M_p(x))$ is odd. $\Box$

An other way of looking at a selfreduction is by seeing it as a *fixed point* of the reducing oracle machine: Let M_{self} be an oracle machine obeying QUERY-TYPE DECREASING. The fixed point of M_{self} is the set that is a solution to the equation: $A = L(M_{self}, A)$. The interesting point is that this fixed point always exists and is unique.

Theorem 7.7 *Let M_i be a an oracle Turing machine obeying a restriction with* QUERY-TYPE *containing* DECREASING. *There exists a unique fixed point for M_i.*

Proof:(existence) Let $\prec_k$ be the polynomially related ordering with polynomial $p(n)$, belonging to M_i. We first define the length$_k$ of strings x ($l_k(x)$) w.r.t. the ordering $\prec_k$, to be the number of strings + 1, preceding x on a maximal descending chain for x. This number is bounded by $p(|x|)$. We construct the fixed point A in stages. A set X containing only strings of length$_k$ $\leq n$ will be suggestively written as X^n. At stage n we put all the strings x with $l_k(x) = n$ in A iff they are in $L(M_i, A^{n-1})$. Since on input x machine $M_i(x)$ only queries strings y, with $l_k(y) < l_k(x)$, we get that A is a fixed point for M_i.
(uniqueness) Suppose that A and B are both fixed points for M_i. (that is satisfy the equation $X = L(M_i, X)$) We prove now by induction on the length$_k$ of strings, that (for all n) $A^n = B^n$. For $n = 1$, M_i is not allowed to query any string on inputs of length$_k$ 1 and since M_i is deterministic, $x \in A^1$ iff $M_i(x)$ accepts iff $M_i^A(x)$ accepts iff $M_i^B(x)$ accepts iff $x \in B^1$. For strings of length$_k$ n, M_i only queries strings of length$_k$ $n-1$ and for those $A^{n-1} = B^{n-1}$ and again since M_i is deterministic, we get that $A^n = B^n$. $\Box$

The last notion of selfreducibility for sets in *NP* we want to discuss here is the one defined by Borodin and Demers [BD76]. They call a set A functional selfreducible if a proof for membership of x can be *generated* in polynomial time, using A as an oracle. We will call this notion, Search Reduces to Decision for A (SRTD for A). To be more precise:

Definition 7.8 ([BD76, BBFG91]) . *Let L be in NP. $x \in L$ can be defined as $\exists y(|y| \leq p(|y|)$ and $R_L(x, y)$ for some polynomial p and some polynomial time computable relation R_L. We say R_L and p define L. Let* $\text{WITNESS}_L(x) = \{y \mid R_L(x, y)\}$. *We say Search Reduces to Decision for L iff there exists a polynomial time computable function relative to L, f^L (with L as oracle) such that for all x: $\exists y \in$* $\text{WITNESS}_L(x)$ *and $f^L(x) = y$, for some relation R_L defining L.*

Thus y in $\text{WITNESS}_L(x)$ is some proof that x is in L. Note that there are many ways to define a set A in *NP*, and that A has SRTD if some proof for $x \in A$ can be generated relative to A.

7.2 Which sets in NP are Selfreducible?

7.2.1 Sets in NP that are Selfreducible

Theorem 7.5 above states that every disjunctive selfreducible set is in *NP*. It is not known if the converse is true, but we will show there is some evidence against this. A relaxation of the previous question would be to look at *NP* -complete sets. The question then becomes : are all $\leq_r^p$ -complete sets for *NP* selfreducible. We do have a partial answer as was observed by Ko [Ko83].

It is easy to see that if A and B are polynomial time isomorphic, and A is r-selfreducible then so is B (just use the induced ordering by the isomorphism to get the ordering for the selfreduction for B). Furthermore SAT, the set of boolean formulas that are satisfiable, is 2dtt-selfreducible. On input $\phi(x_1, \ldots, x_n)$, a formula on n variables, query if either $\phi(x_1 = 0, \ldots, x_n)$ or $\phi(x_1 = 1, \ldots, x_n)$ is in SAT. Here we denote $\phi(x_1 = 0(1), \ldots, x_n)$, with the formula that has False (True) substituted for its first variable. For some suitable coding of formulas the formula with 1 variable less will be strictly smaller in length than the original one and thus selfreducible via the standard ordering on strings with respect to length, i.e. $y \prec x$ iff $|y| < |x|$. This is as we have seen before a polynomially related ordering. Now applying a theorem by Berman and Hartmanis [BH77], that states that all *known* (at that time) *NP* -complete problems are polynomial time isomorphic, we have that all natural *NP* -complete problems are 2-dtt-selfreducible. Whether they are all 2-dtt-selfreducible with respect to some *fixed* polynomially related ordering is a different question. Some work along these lines can be found in [BSS90].

The following theorem shows that for SRTD things are a bit better understood then for selfreducibility, in the sense that some absolute result can be obtained.

Theorem 7.9 *All NP -complete sets w.r.t.* $\leq_T^p$ *-reductions have* SRTD*.*

Proof: Let A be a $\leq_T^p$ -complete set for *NP*. Let R_A and p define A. Define PREFIX$(A) = \{<x, y> \mid \exists z$ such that y is a prefix of z and $R_A(x, z)\}$ (compare with Selman[Sel88]). Clearly PREFIX(A) is in *NP*. Thus PREFIX$(A) \leq_T^p A$. Using PREFIX(A) as an oracle it is easy to generate a proof for x in A:

```
input $x$
Let z = $\lambda$ and DONE = false
repeat
  if $<x, z0>$ in PREFIX($A$) then z=z0
    else if $<x, z1>$ in PREFIX($A$) then z=z1
        else DONE = true
until DONE = true
output z iff R(x,z)
end.
```

Since PREFIX$(A) \leq_T^p A$, the above algorithm can be made computable relative to A, thus establishing that A has SRTD. □

Though we will discuss the relations between the three notions of selfreduction on sets in *NP*in more depth, we give a sneak preview and prove the following lemma:

Lemma 7.10 *If A has* SRTD *then A is autoreducible.*

Proof: Let M_{srtd} witness the fact that A has SRTD for R_A and p, defining A. We are going to alter M_{srtd} to make it an autoreduction for A. On input x simulate $M_{srtd}(x)$. If however x the input itself is written on the QUERY-tape, don't query x, but assume the answer is YES, i.e. write 1 on the ANSWER-tape. Continue the computation of $M_{srtd}(x)$. Let z be the output of the computation. Now accept x iff $R(x, z)$. First note that the above machine obeys the query constraint $\overline{\text{ID}}$. To see that it computes $x \in A$ correctly: if x is in A, the computation is correct (on query x the correct answer is written on the ANSWER-tape) and $M_{srtd}(x)$ should output a witness z such that $R(x, z)$. On the other hand if x not in A the above computation *may* be wrong but nevertheless a witness can never be found because there does not exist one. □

Lemma 7.10 together with Theorem 7.9 give us the following corollary:

Corollary 7.11 *If A is $\leq_T^p$ -complete for NP then A is autoreducible.*

The previous corollary is also interesting seen in the light that such a fact is plainly *not* true for the *RE* sets, since Ladner[Lad73] has shown that there exist $\leq_T^{REC}$ -complete *RE* sets that are not mitotic and thus not autoreducible (in the recursive sense).

7.2.2 Sets in NP that are not Selfreducible

In the previous section we saw that it was possible to show that certain type of sets in *NP* are selfreducible in some sense. In this section we focus on the other possibility: being not selfreducible. As stated before this would imply $P \neq NP$, so at least we have to work under that assumption. Unfortunately no results are known from this assumption, so the next best thing is to assume a stronger assumption like $E \neq NE$ or even $EE \neq NEE$ (here *EE* (and *NEE*) denote (non-deterministic) double exponential time). We note that these assumptions imply $P \neq NP$, but that it is not known whether they are equivalent.

We will need one more definition and some known facts about these assumptions and what they imply. The next notion we want to introduce is again a direct translation of a recursion theoretical one: semi-recursive, proposed by Jockush[Joc68]. The resource bounded version will be called p-selective and is due to Selman.

Definition 7.12 ([Sel79]) *We say that a set A is p-selective iff there exist a polynomial time computable function f: $\Sigma^* \times \Sigma^* \rightarrow \Sigma^*$. Such that:*

1. $f(x, y) = x$ *or* $f(x, y) = y$
2. $(x \in A$ *or* $y \in A) \Rightarrow f(x, y) \in A$.

The function f is called the *selector* function for A. The notion can be seen as a weaker form of decision: of two strings the most likely string of the two to be in A can be computed in polynomial time. Note that a selector function for some set A induce some kind of ordering on two strings: suppose that $f(x, y) = x$, then $y \in A \Rightarrow x \in A$ and because $\Rightarrow$ is a logical implication it follows that $x \notin A \Rightarrow y \notin A$.

The next lemma will be very useful in the following. The lemma states that for any p-selective set A and any n strings, these strings can be ordered in a very special way:

Lemma 7.13 *Let A be a p-selective set. We can order any n strings in the following way: $x_1 \in A \Rightarrow x_2 \in A \Rightarrow \ldots \Rightarrow x_n \in A$.*

Proof: We prove this by induction on the number of strings to be ordered. For any x and y, $f(x, y) = x$ induces the ordering: $y \in A \Rightarrow x \in A$. Let $x_1, \ldots, x_n$ be the strings that have to be ordered. If $f(x_i, x_j) = x_j$ we say that x_i wins from x_j. Now use the p-selector to play a knock-out tournament among the n strings. Let x_0 be the winner of the tournament. If $x_0 \in A$ then for all i, $x_i \in A$ and if there is an i such that $x_i \notin A$, then $x_0 \notin A$. By induction hypotheses the $n-1$ strings ($x_1, \ldots, x_n$ without x_0) can be ordered: $x_1 \in A \Rightarrow \ldots \Rightarrow x_{n-1} \in A$. Together with x_0 this yields the ordering $x_0 \in A \Rightarrow x_1 \in A \ldots \Rightarrow x_{n-1} \in A$. $\square$

In [Sel79], Selman used p-selectivity in order to show that $E \neq NE$ implies that $\leq_T^p$ -reducibility and $\leq_m^p$ -reducibility differ on *NP*. He then asked for constructions (using p-selective sets) to separate the *completeness* notions on NP with respect to these reduction types. (i.e. $\leq_m^p$ and $\leq_T^p$) Ko in [Ko83] then showed that p-selective sets probably are not sufficient to distinguish the completeness notions, for he showed that every p-selective set is $\leq_T^p$ reducible to some sparse[1] set and thus if a p-selective set were complete for NP, then the polynomial time hierarchy would collapse to Σ_2^p.

Theorem 7.14 ([Ko83]) *If A is p-selective then $A \leq_T^p S$, for S a sparse set.*

In [Sel79], Selman proved the following theorem:

Theorem 7.15 ([Sel82]) *For every tally language T there exist a set B such that*

1. *$B \leq_{ptt}^p T$.*
2. *$T \leq_T^p B$.*
3. *B is p-selective.*

Together with the fact [Boo74], that if $E \neq NE$ then there exist a tally set in $NP - P$, this theorem implies the existence of a *p-selective* set in $NP - P$, under the assumption that $E \neq NE$. To see this, observe first that NP is closed under $\leq_{ptt}^p$ -reductions [Sel79]. Now if the tally set T from Theorem 7.15 is in $NP - P$, it follows that the p-selective set B also has to be in $NP - P$.

We will see that this set cannot be selfreducible. It was known that *any* tally set T that is selfreducible is in P[MP79]. Thus the first example of a non-selfreducible set in NP was already given by Book [Boo74].

In section 7.3 we will prove that every set that is both selfreducible and p-selective is in P. Given this theorem, the following corollary is immediate:

Corollary 7.16 *There is a p-selective set in NP that is not selfreducible unless $E = NE$.*

[1]recall that a set S is sparse if for all n, $\|S^{\leq n}\| \leq p(n)$ for some polynomial p.

These results can be viewed as strong evidence that there are non selfreducible sets in *NP*. What about the other types of selfreducibility?

It is at present not known if under the hypotheses that $E \neq NE$, there are sets in *NP* that are non autoreducible or do not have SRTD. But under the stronger hypotheses, that *double* exponential time, *EE*, is not equal to non deterministic *double* exponential time, *NEE*, one can prove that this is indeed the case.

Theorem 7.17 *There exists a set D in NP that is not autoreducible, unless NEE $\neq$ EE.*

Proof: Let A be a set in $NEE - EE$. Define $T(A) = \{0^n \mid n \in A\}$ and $T(T(A)) = \{0^m \mid m \in T(A)\} = \{0^{2^n} \mid n \in A\}$. Let D be $T(T(A))$. First of all if $A \in NEE - EE$ then $D \in NP - P$[2]. Next note that the difference in size of two successive elements in D is exponential. Suppose that D is autoreducible. Consider the autoreduction M_{auto} on input 0^{2^n} for some n. Since M_{auto} is prohibited to query 0^{2^n}, the only strings that are in D and can be queried by M_{auto}, are those that are exponentially smaller than 0^{2^n}. But since D is in *NP*, those queries can be computed in *deterministic* polynomial time and thus D in P would follow, contradicting our assumption. □

Applying Lemma 7.10 we get the following corollary:

Corollary 7.18 ([BBFG91]) *There exists a set A in NP that does not have* SRTD*, unless NEE = EE,*

So far we were able to show that nontrivial (i.e. complete) sets in *NP* posses the property of being selfreducible: all natural $\leq_m^p$ -complete sets are selfreducible and all $\leq_T^p$ -complete sets have SRTD and are autoreducible. Furthermore we could show that other sets did not have these properties. Though non selfreducible sets needed a weaker hypotheses to prove their existence. In section 7.4 we will examine the relationship between the 3 notions of selfreducibility.

[2]Observe that if $x = 0^{2^n}$ is in $T(T(A))$ then $|x| = 2^{2^n}$, so *double* exponential time in n is only polynomial in x, so $T(T(A))$ is in *NP*. On the other hand if $T(T(A))$ is in P then one can show that A has to be in *EE*, contradicting the hypotheses.

In the next section we will prove a characterization of P in terms of sets that are p-selective and selfreducible. This characterization enabled us earlier on to conclude that some sets are not selfreducible. Furthermore it will help us in section 7.4.

7.3 P-selective & Selfreducible Sets.

In order to get a better insight in the sets that live in a certain complexity class, we have already seen that reductions induce some structure in complexity classes. The main goal is to show that certain classes exhibit different structures and thus are different. Another approach to gain some knowledge about certain complexity classes is to characterize them, other than by their definition, by the sets that are contained in them. This approach not only leads to more insight in the sets that are in a certain class, but also gives a handle to show that certain sets are *not* contained in the class under consideration. Well known examples of this approach are the many ways to define the class of sets for which there exist small circuits [Pip79], and the identification of various forms of interactive proof systems with *PSPACE*, *NEXP* and *NP* [Sha90, BFL90, ALM$^+$92]. Other examples are the classification of complexity classes by various logical theories [Imm84, Imm87].

In this section we characterize P in terms of sets that are selfreducible and p-selective. But first we show some other characterizations of P.

7.3.1 Other characterizations of P

The rules of the game leading to characterizations of complexity classes are to describe in some way the sets that are contained in the complexity class. One way to do this is by looking at structural properties of sets that would imply the membership to a certain complexity class.

In this vein it was proven by [GJY87] that all sets that are selfreducible and p-cheatable[3]are in P. In that same paper they prove that

[3]A set is p-cheatable if there is a constant $k \geq 1$) such that for *any* 2^k elements, the question of membership in A for all 2^k elements can be reduced in polynomial time to only k questions about membership in A.

P can also be characterized as those sets that are both p-cheatable and near-testable[4]. Another characterization can be found in [Sel82], where it is shown that every set that is p-selective and positive Turing reducible[5] to its complement is in P.

7.3.2 The P-selective & Selfreducible characterization

In [Sel79] it was shown that all sets that are *disjunctive* selfreducible and p-selective are in P. The proof of this goes as follows. Let A be disjunctive selfreducible, on input x, $y_1, \ldots, y_n$ are generated such that $x \in A$ iff $\exists i$ s.t. $y_i \in A$ and furthermore $y_i \prec_k x$ for some polynomially related ordering $\prec_k$. Now use the p-selector to choose one of the y_i's and proceed recursively on this y_i until a string y_o is encountered for which the selfreduction determines without any queries whether it is in A. Because the nature of the selfreduction and the p-selector, $x \in A$ iff $y_0 \in A$ and because y_0 is reached in at most $p(n)$ many steps, A is recognized in polynomial time.

We want to improve this theorem by generalizing disjunctive selfreducible to selfreducible. First we show what the obvious upper bound is for sets that are p-selective and selfreducible:

Theorem 7.19 ([Ko83]) *Every set that is p-selective and selfreducible is in Σ_2^p.*

Proof: Let A be p-selective and selfreducible by machine M_{self}. Using Theorem 7.14 we get that $A \leq_T^p S$, for some sparse set S, say by M_r. Using techniques from [KL80, Bal90] we show that A is in Σ_2^p. Let $p(n)$ be the polynomial belonging to the polynomially related ordering, $\prec_k$, induced by M_{self}. To see that A is in Σ_2^p, we give a Σ_2^p -algorithm. On input x (of length n), guess $S^{\leq q(n)}$, that part of S, that is needed for M_r to decide membership in A relative to S, for strings of length

[4]A set A is near-testable if there exists a polynomial time computable function for computing the function $\chi_A(x) + \chi_A(x+1) \pmod 2$ where $(x+1)$ denotes the lexicographic successor of x.

[5]Recall that an oracle machine M is positive if it holds for all oracles A, B: if $A \subseteq B \Rightarrow L(M, A) \subseteq L(M, B)$.

$p(n)$. Next universally check the following: $\forall x'(x' \prec_k x'', |x''| \leq n)$: $x' \in L(M_r, S^{\leq q(n)})$ iff $x' \in L(M_{self}, L(M_r, S^{\leq q(n)}))$. Using Theorem 7.7 it follows that if the check does not fail, an initial segment of a fixed point for M_{self} has been found and via uniqueness is the same as an initial segment of A: x is in A iff x is in $L(M_r, S^{\leq q(n)})$. □

We now give the proof of the new characterization of P in terms of sets that are both selfreducible and p-selective. The proof will follow a different strategy than the proof of Theorem 7.19 above.

Theorem 7.20 *A set is in P if and only if it is both selfreducible and p-selective.*

Proof: We will first show how to translate an *adaptive* selfreduction into a *non-adaptive* one using the p-selector function. Then we will show that all non-adaptive selfreducible and p-selective sets are in P. This establishes one side of the theorem. The other side is trivial since every set in P is automatically selfreducible and p-selective.

Lemma 7.21 *Every adaptive-selfreduction for a set A can be transformed into a non-adaptive-selfreduction for A, if A is p-selective.*

Proof: Let M_s be an adaptive selfreduction for some p-selective set A and let f be the p-selector. We will show how to construct a non-adaptive selfreduction, $M_{s'}$ for A. We start by simulating M_s on input x. Let q_1 be the first query queried by M_s. Next compute $f(x, q_1)$.[6] Depending on the outcome of $f(x, q_1)$ either $q_1 \in A \Rightarrow x \in A$, or $q_1 \notin A \Rightarrow x \notin A$. We proceed by writing a 0 (1) on the ANSWER-tape if a 1(0) implies (non-)membership of x in A. (i.e. we explore the branch that does not give an outcome to x yet) We repeat this procedure until an accepting or rejecting state, OUTCOME, has been found. Let $q_1, \ldots, q_{p(n)}$ be the queries encountered in the above procedure. M_s' now simply queries $q_1, \ldots, q_{p(n)}$, and decides x as follows: either the answers to $q_1, \ldots, q_{p(n)}$ are exactly the same as assumed in the above procedure, i.e. the string written on the ANSWER-tape by the oracle is the same as the string written on the ANSWER-tape in the simulation of M_s. In

[6]Recall that a selector function $f(x, y) = x$ for some set A, induces an ordering on x and y: $y \in A \Rightarrow x \in A$ and $x \notin A \Rightarrow y \notin A$.

this case we took the "true" path in the above oracle computation, and $M_{s'}$ accepts iff OUTCOME is an accepting state. In the other case there exists an i such that $\chi_A(q_i) = 0(1)$ and we wrote $1(0)$ on the ANSWER-tape for q_i, during the simulation of M_s. In this case however the p-selector guarantees that $\chi_A(x) = 0(1)$, and $M_{s'}$ accepts accordingly. $M_{s'}$ behaves non-adaptively and since it queries only $q'_i s$, that are in $Q(M_s(x))$, it is clear that M'_s obeys DECREASING and thus is a non-adaptive selfreduction for A. $\Box$

Next we have to show that every non-adaptive selfreducible set is in P. We do this by first reducing the number of queries that the selfreduction queries to a single question. This is enough to show that the set A is in P, since we now proceed recursively on this single question and after polynomially many applications we derive a string y_0 for which the selfreduction is not allowed to query any queries at all and will decide whether $y_0 \in A$ in polynomial time, which on its turn implies (non)membership of x in A.

Lemma 7.22 *If a set A is non-adaptive selfreducible and p-selective, then A is selfreducible via a selfreduction that only queries 1 query. (i.e. A is selfreducible via a machine that obeys a restriction where $N(i,x) = 1$)*

Proof: Let M_s witness the non-adaptive selfreduction for A and let f be the p-selector. We simulate $M_s(x)$ and let $q'_1, \ldots, q'_{p(n)}$, be the queries queried by M_s. By Lemma 7.13 we can order the queries together with x, obtaining a sequence $q_1 \in A \Rightarrow \ldots \Rightarrow q_{i-1} \in A \Rightarrow x \in A \Rightarrow q_i \in A \Rightarrow \ldots \Rightarrow q_{p(n)} \in A$[7]. With the property that if $q_j \in A$, then all the queries to the right of q_j (i.e. $q_i (i > j)$) (including x if it is to the right of q_j) are in A, and all the queries to the left of q_j are not in A if q_j is not in A.

We now claim that the single query that determines completely whether x is in A, is either q_{i-1} or q_i. Thus lets us take a closer look at x, q_{i-1} and q_i in the above ordering. Lets evaluate the possible outcomes of $\chi_A(q_i), \chi_A(q_{i-1})$ and $\chi_A(x)$.

[7]We relabeled the queries according to the ordering

1. $\chi_A(q_{i-1}) = 0$ and $\chi_A(q_i) = 0$. The ordering implies that $\chi_A(x) = 0$.

2. $\chi_A(q_{i-1}) = 1$ and $\chi_A(q_i) = 1$. The ordering implies that $\chi_A(x) = 1$.

3. $\chi_A(q_{i-1}) = 1$ and $\chi_A(q_i) = 0$. This cannot happen, since it is contradictory to the ordering.

4. $\chi_A(q_{i-1}) = 0$ and $\chi_A(q_i) = 1$. The ordering by itself does not force x to be in or out of A immediately. Observe however that for all the strings right of q_i, q_j $(j > i)$, $\chi_A(q_j) = 1$, and for all the strings left of q_{i-1}, q'_j $(j' < (i-1)), \chi_A(q_j) = 0$. So $\chi_A(x)$ can be computed in polynomial time from this information.

From the above evaluation it is clear that $\chi_A(x)$ can be computed from $\chi_A(q_i)$ and $\chi_A(q_{i-1})$. To see that in fact one of the two gives enough information observe that in case 1 and 2, $\chi_A(x)$, $\chi_A(q_{i-1})$ and $\chi_A(q_i)$ are the same. In case 4 one can compute whether $\chi_A(x) = \chi_A(q_{i-1})$ or $\chi_A(x) = \chi_A(q_i)$. Because case 3 does not occur, $\chi_A(x)$ is the same as either $\chi_A(q_{i-1})$ or $\chi_A(q_i)$. $\Box$

This ends the proof of the theorem. $\Box$

Examining the statement of this theorem carefully we observe that it is in fact a statement about selfreducible sets that are many-one reducible to a set that is p-selective. In fact a much stronger statement is true. We capture this in the following theorem.

Theorem 7.23 *If A is $\leq^p_{pos}$ B and B is p-selective then $A \leq^p_m B$ and A is p-selective.*

Proof: We use the same strategy as the proof of Theorem 7.20. First we show how to transform a positive Turing reduction into a positive truth-table reduction, using the p-selector for B. Then we will use a result from Selman [Sel82] that says that any positive truth-table reduction to a p-selective set can be transformed into a many-one reduction. From this it is clear that A is p-selective, since one can use the p-selector for B and the many-one reduction from A to B to construct a p-selector for A.

Let M_T witness the positive Turing reduction from A to B and let f be the p-selector for B. First we define for any z the following two sets: $B^+(z) : \{x \mid f(x,z) = x\}$ and $B^-(z) : B^+(z) - \{z\}$. We will need the following properties of $B^+(z)$ and $B^-(z)$.

Lemma 7.24 *For any z, let $B^+(z)$ and $B^-(z)$ be as above.*

- *$B^+(z)$ and $B^-(z)$ are in P.*
- *$z \in B \Rightarrow B^+(z) \subseteq B$.*
- *$z \notin B \Rightarrow B \subseteq B^-(z)$.*

Proof: Since f can be computed in polynomial time $B^+(z)$ and $B^-(z)$ are in P. ($z \in B \Rightarrow B^+(z) \subseteq B$) Let x be any string in $B^+(z)$, so $f(x,z) = x$ and thus $z \in B \Rightarrow x \in B$.
($z \notin B \Rightarrow B \subseteq B^-(z)$) Let x be any string in B, since $z \notin B, f(x,z) \neq z$ and $f(x,z) = x$ and $x \in B^-(z)$. □

The following procedure will transform the Turing reduction into a non-adaptive computation:
Simulate $M_T(x)$ and let q_1 be the first query queried by M_T. Compute $M_T^{B^+(q_1)}(x)$ and $M_T^{B^-(q_1)}(x)$ there are 4 possible outcomes:

1. $M_T^{B^+(q_1)}(x)$ rejects and $M_T^{B^-(q_1)}(x)$ accepts.
2. $M_T^{B^+(q_1)}(x)$ accepts and $M_T^{B^-(q_1)}(x)$ rejects.
3. $M_T^{B^+(q_1)}(x)$ accepts and $M_T^{B^-(q_1)}(x)$ accepts.
4. $M_T^{B^+(q_1)}(x)$ rejects and $M_T^{B^-(q_1)}(x)$ rejects.

Outcome 1 cannot occur since it violates the positiveness of M_T. We will argue that case 2 will imply that $x \in A$ iff $q_1 \in B$, in which case we already have a many-one reduction and we can stop. In case 3 we will write down the query q_1 and continue the simulation of $M_T(x)$ with 0 written on the ANSWER-tape, and recursively continue on the next query queried. In case 4 we write down q_1 and continue the simulation with 1 written on the ANSWER-tape. This procedure either stops in case a many-one reduction is found or when the simulation of $M_T(x)$ ends in an accepting or rejecting state OUTCOME. Let $q_1, \ldots, q_{p(n)}$ be the queries written down by the above procedure, and y be the string on the ANSWER-tape. The truth-table reduction behaves as follows: It queries $q_1, \ldots, q_{p(n)}$. If the string written on the ANSWER-tape is y, it accepts iff OUTCOME is an accepting state. Otherwise let q_i be the first query such that the i^{th} bit of y is different from the i^{th} bit of the string on the ANSWER-tape. Suppose that for q_i we were in case 3 in the simulation of $M_T(x)$. This means that both the computations with $B^+(q_i)$ and with $B^-(q_i)$ were accepting. We wrote down a 0 on the oracle tape whereas $q_i \in B$. Since $q_i \in B$, it follows from Lemma 7.24 that $B^+(q_i) \subseteq B$. Thus by positiveness of M_T, $M_T^B(x)$ also accepts and we let the truth-table reduction safely accept. On the other hand assume that for q_i we were in case 4. We wrote down a 1 on the ANSWER-tape, but $q_i \notin B$ and thus by Lemma 7.24, $B \subseteq B^-(q_i)$. Again by positiveness, $M_T^B(x)$ rejects and we let the truth-table reduction reject. It is not hard to see that the above truth-table reduction is in fact positive and reduces A to B, as was needed.

It remains to show that case 2 yields a many-one reduction ($x \in A$ iff $q_i \in B$). Let q_i be such that $M_T^{B^+(q_i)}(x)$ accepts and $M_T^{B^-(q_i)}(x)$ rejects. There are two cases:

- $q_i \in B$: $B^+(q_i) \subseteq B$ (Lemma 7.24) and $M_T^{B^+(q_1)}(x)$ accepts, thus via positiveness $M_T^B(x)$ accepts so $x \in A$.

- $q_i \notin B$: $B \subseteq B^-(q_i)$ (Lemma 7.24) and $M_T^{B^-(q_1)}(x)$ rejects, so positiveness guarantees that $M_T^B(x)$ rejects and thus $x \notin A$.

The next lemma ends the proof of this theorem.

Lemma 7.25 ([Sel82]) *If $A \leq_{ptt}^p B$ and B is p-selective, then $A \leq_m^p B$.*

Proof: Let M_p witness the $\leq_{ptt}^p$-reduction. Simulate M_p on input x and order the queries as was done in Lemma 7.13: $q_1 \in B \Rightarrow \ldots \Rightarrow q_{p(n)} \in B$. The only valid, according to the ordering, strings that can be written on the ANSWER-tape are strings that look like $0^i 1^{p(n)-i} (0 \leq i \leq p(n))$. Simulate $M_p(x)$ with $0^i 1^{p(n)-i}$ written on the ANSWER-tape for $i = 0$ up to $i = p(n)$. Let j be such that $M_p(x) = 1$ with $0^j1^{p(n)-j}$ and $M_p(x) = 0$ with $0^{j+1}1^{p(n)-(j+1)}$ written on the ANSWER-tape. (Note that if such a j does not exist, for all $i, 0 \leq i \leq p(n)$, $M_p(x)$ with $0^i1^{p(n)-i}$ written on the ANSWER-tape either always accepts or always rejects. In this case $\chi_A(x)$ can be inferred and a fixed element in B or outside of B can be generated in order to establish a many-one reduction.) Since M_p is a positive oracle machine it follows that for strings $0^i1^{p(n)-i}$ with $i > j$ written on the ANSWER-tape, $M_p(x) = 1$. From this it is clear that $x \in A$ iff $q_j \in B$. $\square$

$\square$

Corollary 7.26 *Let A and B be sets such that $A \leq_{pos}^p B$. A is in P if and only if A is selfreducible and B is p-selective.*

Corollary 7.27 *Let C be any of the following classes: $NP, PP, \oplus P$ or $PSPACE$. $C = P$ if and only if there exists a $\leq_{pos}^p$-hard set for C that is p-selective.*

What about *non*-positive reductions? First we note that if we could generalize Theorem 7.20 to a theorem that states that A is in P iff A is selfreducible and Turing reducible to a p-selective set, then this would imply that EXP is not contained $P/poly$[8]. For suppose it is. Since SAT is in EXP, $SAT \leq_T^p S$, for some sparse set S. Using the fact[HIS85] that every sparse set is $\leq_T^p$ reducible to a tally set T, we have that $SAT \leq_T^p T$, for some tally set T. Applying Theorem 7.15, we get that $SAT \leq_T^p B$, and B is p-selective. Now using a generalized Theorem 7.20 we see that $P = NP$. On the other hand we know from Karp and Lipton [KL80] that $EXP = \Sigma_2^p$ under the hypotheses that EXP in $P/poly$. Putting oneandtwo together this would imply that $EXP = P$, which is clearly in contradiction with the hierarchy theorems. A similar argument yields the following,

Corollary 7.28 *EXP does not have $\leq_{pos}^p$ -hard sets that are p-selective.*

[8] $P/poly$ is the same class as the sets that are $\leq_T^p$ reducible to a sparse set.

The above discussion justifies trying to generalize theorem 7.20. Unfortunately Theorem 7.20 is probably not true for Turing reductions!

Theorem 7.29 *If $E \neq UE$, then there exist sets D and B in $NP - P$, such that:*

1. *D is disjunctive selfreducible.*
2. *B is p-selective.*
3. *$D \leq_T^p B$.*

Proof: The assumption $E \neq UE$ implies the existence of a *tally* set T in $UP - P$. Let R_T and $q(n)$ define T. Since $T \in UP - P$ there is for each $0^n \in T$ exactly one string w such that $R_T(0^n, w)$. We call this string w the witness for 0^n. Construct $D = \text{PREFIX}(T) = \{<0^n, y> \mid y \text{ is a prefix of } w \text{ and } R_T(0^n, w)\}$ as in Theorem 7.9. It is not hard to see that D is disjunctive selfreducible (c.f. Selman [Sel88]). Also $T \leq_m^p D$ because $0^n \in T$ iff $<0^n, \lambda> \in D$ and since $T \in NP - P$ it follows that $D \in NP - P$. Furthermore D is sparse because for each $0^n \in T$ there is only 1 witness of size $q(n)$ and only $q(n)$ prefixes of w. Since D is sparse there is a tally set T' such that $D \leq_T^p T'$ (Theorem 5.4 in Chapter 5 or in [HIS85]). Applying theorem 7.15 (point 1) we get a p-selective set B in $NP - P$. Putting the pieces together we have: $T \leq_m^p D \leq_T^p T' \leq_T^p B$, as was needed. □

The previous theorem prevents us from generalizing Theorem 7.20 upto Turing reductions. Nevertheless can we stretch the theorem up to $\leq_{1-tt}^p$ -reductions, which is an example of a non-positive reduction.

Theorem 7.30 *Let $A \leq_{1-tt}^p B$. A is in P if and only if A is selfreducible and B is p-selective.*

Proof: We follow quite strictly the lines of the proof of Theorem 7.20. First we show how to transform the adaptive selfreduction into a non-adaptive selfreduction, this time using the p-selector for B. We know from [HHO+92] that there are sets that are not p-selective, but that are $\leq_{1-tt}^p$ -reducible to a p-selective set. Although A may not be p-selective, we can still order any pair of strings in a certain way. Let M_1

witness the fact that $A \leq_{1-tt}^{p} B$ and let x and y be pair of strings. Let x' be the query written on the QUERY-tape by $M_1(x)$ and y' the one by $M_1(y)$. The following 4 situations can occur:

1. $x \in A \Longleftrightarrow x' \in B$ and $y \in A \Longleftrightarrow y' \in B$
2. $x \in A \Longleftrightarrow x' \notin B$ and $y \in A \Longleftrightarrow y' \notin B$
3. $x \in A \Longleftrightarrow x' \notin B$ and $y \in A \Longleftrightarrow y' \in B$
4. $x \in A \Longleftrightarrow x' \in B$ and $y \notin A \Longleftrightarrow y' \in B$

In case 1 we order the strings x and y as $x \in A \Rightarrow y \in A$ if $f(x', y') = y'$. In case 2 we do the opposite: $x \in A \Rightarrow y \in A$ if $f(x', y') = x'$. In case 3 we have the following: $x \in A \Rightarrow y \notin A$ if $f(x', y') = x'$. Case 4 is analogous to case 3. It is left to reader to see that this is enough to prove Lemma 7.21 for selfreducible sets that are $\leq_{1-tt}^{p}$-reducible to a p-selective set. The next step is to show that this non-adaptive selfreduction, say M_{self}, can be transformed into one that queries only one query. Let $q_1, \ldots, q_{p(n)}$ be the nonadaptive queries from the selfreduction for A. Let $M_1(x)$ query x' and $M_1(q_i)$ query q_i' $(1 \leq i \leq p(n))$. Observe that M_{self} together with M_1 can be chained together in the sense that $\chi_B(q_1'), \ldots, \chi_B(q_{p(n)}')$ can be used to determine $\chi_B(x')$. Since B is p-selective we can order $q_1' \ldots q_{p(n)}'$ together with x' (Lemma 7.13, yielding the following ordering: $q_1' \in B \Rightarrow \ldots \Rightarrow q_{i-1}' \in B \Rightarrow x' \in B \Rightarrow q_i' \in B \Rightarrow \ldots \Rightarrow q_{p(n)}' \in B$[9]. A similar argument as in Lemma 7.22 shows that $\chi_B(x')$ can be computed from either $\chi_B(q_{i-1}')$ or $\chi_B(q_i')$ and thus that $\chi_A(x)$ can be computed from either $\chi_A(q_{i-1})$ or $\chi_A(q_i)$. $\square$

Corollary 7.31 *EXP does not have $\leq_{1-tt}^{p}$ -hard sets that are p-selective.*

Corollary 7.32 *Let C be any of the following classes: NP, PP, $\oplus$P or PSPACE. $C = P$ if and only if there exists a $\leq_{1-tt}^{p}$-hard set for C that is p-selective.*

[9]We relabeled the queries

The first obvious question that arizes is: does this work for other type of selfreductions as well? Lemma 7.21 and Lemma 7.22 go through with respect to autoreducibility yielding the following:

Corollary 7.33 *If A is both autoreducible and p-selective, then A is autoreducible via a machine that queries only 1 query. (i.e. obeys a restriction with $N(i,x)=1$)*

Unfortunately this is not enough to guarantee that the set is in P, and in the next section we will in fact see that Theorem 7.20 is not true for autoreducible sets and probably neither for sets that have SRTD.

This ends more or less the picture for autoreducible sets. For SRTDhowever we have a possibility of strengthening the definition somewhat: instead of letting the function that computes the witness be adaptive we demand it to be non adaptive.

Definition 7.34 *We say that A has* SRTD|| *(non-adaptively reduces search to decision for A) iff A has* SRTD *via a function that non-adaptively queries A. (i.e. f can be computed via a machine that obeys a restriction that has* COMP *= non-adaptive)*

Very recently it was proven in [NOS] that Theorem 7.20 goes through with respect to sets that have SRTD||.

Theorem 7.35 ([NOS]) *A set is in P if and only if it has* SRTD|| *and is p-selective.*

Proof: Again one side is trivial, so we only show that if A has SRTD|| and is p-selective then it is in P. Let f be the p-selector for A and let $M_{srtd||}$ witness the fact that A has SRTD|| via relation R_A and polynomial $q(n)$. Simulate $M_{srtd||}$ on input x and let $q_1,\ldots,q_{p(n)}$ be the queries queried by $M_{srtd||}$. The idea now is to order the strings as in Lemma 7.13. Let $q_1 \in A \Rightarrow \ldots \Rightarrow q_{p(n)} \in A$ be this ordering. Observe that because of the ordering only $p(n)+1$ values are possible for $\chi_A(q_1),\ldots,\chi_A(q_{p(n)})$. Let $a_1,\ldots,a_{p(n)+1}$ be all the possible values for $\chi_A(q_1),\ldots,\chi_A(q_{p(n)})$. Next for all a_i simulate $M_{srtd||}$ on input x with a_i written on the ANSWER tape of $M_{srtd||}$ and let y_i be the output. Now accept iff there exists an i such that $R_A(x,y_i)$ holds. This algorithm correctly decides A for if $x \in A$, one of the answer strings a_i is correct and the corresponding y_i is a witness for x. On the hand if $x \notin A$, a witness can never be found. $\square$

7.4 Selfreducibility Notions on NP

In this section we want to figure out whether the 3 notions of selfreductions: selfreducible, having SRTD and autoreducible differ on *NP*. First of all we recall that selfreducibility is a special form of autoreducibility, thus selfreducible implies autoreducible. Lemma 7.10 tells us that SRTD implies autoreducible. From the fact that there are autoreducible sets of arbitrary complexity we get that there are sets that are autoreducible but not selfreducible. The following observation shows that there are (probably) autoreducible sets in *NP* that have neither SRTD nor are selfreducible.

Observation 7.36 ([Nai92]) *There is a set B in $NP - P$ that is autoreducible and p-selective, but does not have* SRTD *and is not selfreducible, unless $EE = NEE$.*

Proof: Recall the set D we used to show that there are non-autoreducible sets in *NP* unless $EE = NEE$ Theorem 7.17. Let A be a set in $NEE - EE$. We defined D to be the set $\{0^{2^n} \mid n \in A\}$. Let $D \oplus D$ be the desired set B. Obviously B is autoreducible and in $NP - P$. Moreover B is p-selective. A selector for B works as follows: (let x and y be the input to the selector function)

- $x = 1z, y = 0z, z = 0^{2^n}$. output x.
- x is not of the form $1z$ or $0z$, $z = 0^{2^n}$. output y.
- x is of the form $0z$ or $1z$ and y is of the form $0z'$ or $1z'$, $z = 0^{2^n}$ and $z' = 0^{2^m}$, $(m < n)$. In this case one can compute whether m is in A, and thus whether y is in B, in polynomial time, so output x if $y \notin B$. otherwise output y.
- all of the above with the role of x and y exchanged.

The above procedure works in polynomial time and is a p-selector for A. The next step is to observe that if B has SRTD then B has SRTD$\|$. This is because the only type of questions that have to be queried to B have to be of the form $1z$ or $0z$, z of the form 0^{2^n}, and *all* questions of that type can be queried non-adaptively. Now we apply Theorem 7.35 and Theorem 7.20 and see that B cannot have SRTD and is not selfreducible. □

As promised we get:

Corollary 7.37 *Theorem 7.20 is not true for autoreducible sets in NP nor sets having* SRTD, *unless* $EE = NEE$.

For selfreducible sets and sets that have SRTD the situation is somewhat more complex. In [NOS] it was shown that under the assumption that $NE \bigcap co\text{-}NE \neq E$, there exists a p-selective set A in $NP - P$, that has SRTD. Applying Theorem 7.20 and Theorem 7.35:

Theorem 7.38 *There exists a set in* $NP - P$, *that has* SRTD, *but is not selfreducible and does not have* SRTD$\|$, *unless* $NE \bigcap co\text{-}NE = E$.

Applying Lemma 7.10 we get:

Corollary 7.39 *There exists a set in* $NP - P$, *that is autoreducible, but is not selfreducible, unless* $NE \bigcap co\text{-}NE = E$.

At present it is not known whether the notions selfreducible and having SRTD are incomparable on NP. We note that if A is disjunctive self-reducible, then A has SRTD: Let M_d witness the disjunctive selfreduction for A. Define $R_A(x,y)$ iff $x \in A$ and y codes a sequence $y_1, \ldots, y_k$, such that $y_i \in Q(M_d, y_{i+1})(1 < i \leq k)$ and $M_d(y_k)$ accepts without any oracle queries. Clearly R_A defines A and by doing a prefix search one can generate a sequence y relative to A. Finally we mention that in [NOS] it was proven that under the assumption that $UE \bigcap co\text{-}UE \neq E$, there exists a set in NP that is disjunctive selfreducible but does not have SRTD$\|$. On the other hand they show that if $UE \neq E$, then there exists a set in NP that has SRTD$\|$ but is not selfreducible. Concerning the issue whether every set in NP that is selfreducible is in fact disjunctive selfreducible Naik has informed us that he has constructed a set $A \in NP$ that is selfreducible but not disjunctive selfreducible under the hypotheses that $UE \bigcap co\text{-}UE \neq E$.

Observation 7.40 ([Nai92]) *If $UE \bigcap co\text{-}UE \neq E$ then there exists a set A in NP that is conjunctive selfreducible but not disjunctive selfreducible.*

Proof: Let T be a tally set in $UP \bigcap co\text{-}UP - P$ constructed from the hypotheses. Consider the set $B = \text{PREFIX}(T)$ as in Theorem 7.29. Again B is disjunctive selfreducible and sparse. Set $A = \overline{B}$. To see that A is the appropriate set, observe that B is not conjunctive selfreducible since then it would be in P. This follows from Fortune's [For79] proof of the result stating that if SAT is $\leq_m^p$ -reducible to the complement of a sparse set, then $P = NP$ (See [Loz92]). Thus the complement of B is conjunctive selfreducible and not disjunctive selfreducible. □

The picture on the next page captures the relations between the notions of selfreducibility on NP.

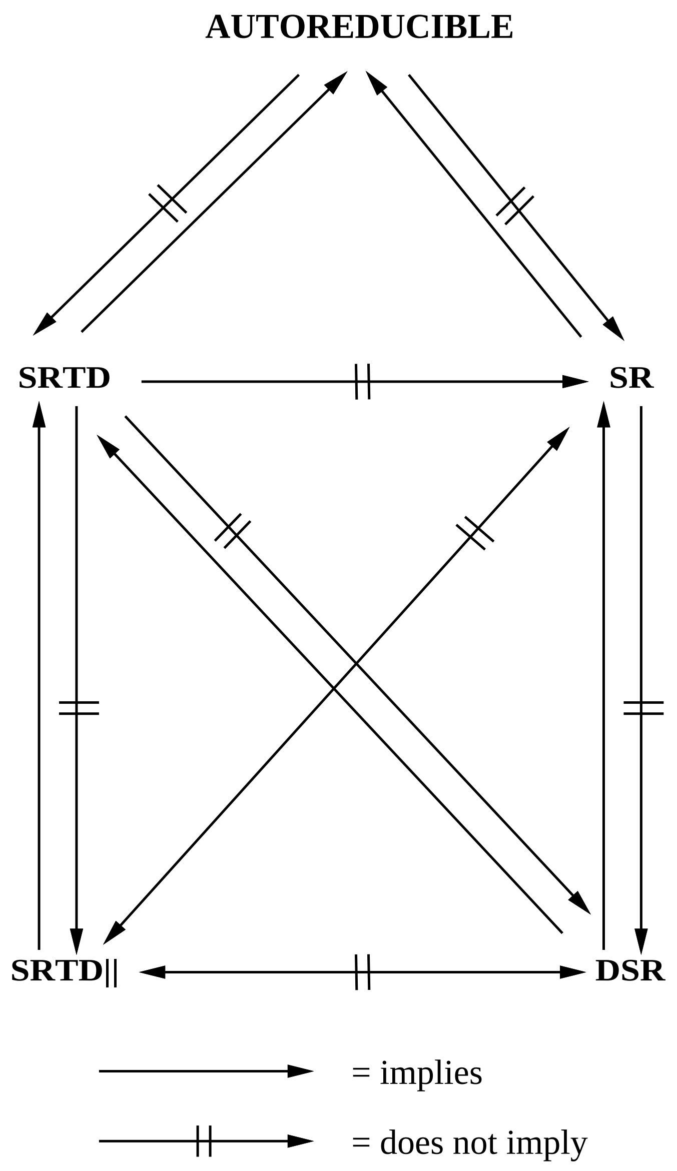


Figure 7.1: Selfreducibility Notions on NP

Bibliography

[ABJ91] C. Àlvarez, J. Balcázar, and B. Jenner. Functional oracle queries as a measure of parallel time. In C. Choffrut and M.Jantzen, editors, *STACS 91, Lecture Notes in Computer Science 480*, pages 422–433. Springer-Verlag, 1991.

[AKM92] V. Arvind, J. Köbler, and M. Mundhenk. On bounded truth-table, conjunctive, and randomized reductions to sparse sets. In R. Shyamasundar, editor, *Proc. 12th Conference on the Foundations of Software Technology & Theoretical Computer Science, Lecture Notes in Computer Science*, pages 140–151. Springer-Verlag, 1992.

[ALM$^+$92] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedi. Proof verification and hardness of approximation problems. In *Proc. 33rd IEEE Symposium Foundations of Computer Science*, 1992.

[AM77] L. Adleman and K. Manders. Reducibility, randomness, and intractability. In *Proc. 9th ACM Symposium Theory of Computing*, pages 151–163, 1977.

[Ars70] M. M. Arslanov. On complete hyperhypersimple sets. *Soviet Mathematics, Dokladi*, 95:30–35, 1970.

[AS84] K. Ambos-Spies. p-mitotic sets. In E. Börger, G. Hasenjäger, and D. Roding, editors, *Logic and Machines, Lecture Notes in Computer Science 177*, pages 1–23. Springer-Verlag, 1984.

[AW90] Eric Allender and Osamu Watanabe. Kolmogorov complexity and degrees of tally sets. *Information and Computation*, 86(2):160–178, June 1990.

[Bal90] J. Balcázar. Self-reducibility. *J. Comput. System Sci*, 41:367–388, 1990.

[BBFG91] R. Beigel, M. Bellare, J. Feigenbaum, and S. Goldwasser. Languages that are easier to verify than their proofs. In *Proc. 32nd IEEE Symposium on Foundations of Computer Science*, pages 19–28, 1991.

[BD76] A. Borodin and A. Demers. Some comments on functional self-reducibility and the NP hierarchy. Technical Report TR76-284, Cornell University, Department of Computer Science, Upson Hall, Ithaca, NY 14853, 1976.

[BDG88] J. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity I.* Springer-Verlag, 1988.

[BDG90] J. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity II.* Springer-Verlag, 1990.

[Ber76] L. Berman. On the structure of complete sets: Almost everywhere complexity and infinitely often speedup. *Proc. 17th IEEE Symposium on Foundations of Computing*, pages 76–80, 1976.

[Ber77] L. Berman. *Polynomial Reducibilities and Complete Sets.* PhD thesis, Cornell University, 1977.

[BFL90] L. Babai, L. Fortnow, and C. Lund. Non-deterministic exponential time has two-prover interactive protocols. In *Proc. 31st IEEE Symposium Foundations of Computer Science*, pages 16–25, 1990.

[BH77] L. Berman and H. Hartmanis. On isomorphisms and density of NP and other complete sets. *SIAM J. Comput.*, 6:305–322, 1977.

[BH92] H. Buhrman and S. Homer. Superpolynomial circuits, almost sparse oracles and the exponential hierarchy. In R. Shyamasundar, editor, *Proc. 12th Conference on the Foundations of Software Technology & Theoretical Computerscience, Lecture Notes in Computer Science*, pages 116–127. Springer Verlag, 1992.

[BHT91] H. Buhrman, S. Homer, and L. Torenvliet. On complete sets for nondeterministic classes. *Math. Systems Theory*, 24:179–200, 1991.

[BHT93] H. Buhrman, A. Hoene, and L. Torenvliet. Splittings, robustness and structure of complete sets. In *STACS 93, Lecture Notes in Computer Science 665*, pages 175–184, 1993.

[BK88] R. Book and K. Ko. On sets truth-table reducible to sparse sets. *SIAM J. Comput.*, 17:903–919, 1988.

[BLS92] H. Buhrman, L. Longpré, and E. Spaan. Sparse reduces conjunctively to tally. Technical Report NU-CCS-92-8, College of Computer Science, Northeastern University, Boston, 02215 MA, 1992. To appear in Proc. Structure in Complexity Theory eighth annual conference 1993.

[Boo74] R. Book. Tally languages and complexity classes. *Information and Control*, 26:186–193, 1974.

[BSS90] H. Buhrman, M. Smid, and E. Spaan. Bounding the number of oracle queries for self-reducible sets. In *Proc. Computing Science in the Netherlands*, pages 79–94, 1990.

[BST91] H. Buhrman, E. Spaan, and L. Torenvliet. Bounded reductions. In *STACS 1991, Lecture Notes in Computer Science 480*, pages 410–421. Springer Verlag, 1991. Will appear in Complexity Theory, Cambridge University press 1993.

[BTvEB93] H. Buhrman, L. Torenvliet, and P. van Emde Boas. Twenty questions to a p-selector. Manuscript, 1993.

[BvHT] H. Buhrman, P. van Helden, and L. Torenvliet. Selective self-reducible sets: A new characterization of P. To appear in Proc. Structure in Complexity Theory eighth annual conference 1993.

[Che52] L. Chebyshev. Mémoire sur les nombres premiers. *Journal de Math.*, 17:366–390, 1852.

[CKR91] R. Chang, J. Kadin, and P. Rohatgi. Connections between the complexity of unique satisfiability and the threshold behaviour of randomized reductions. In *Proc. Structure in Complexity Theory sixth annual conference*, pages 255–270. IEEE computer society press, 1991.

[Coh74] P.M. Cohn. *Algebra Volume 1.* John Wiley & Sons, 1974.

[Coo71] S. Cook. The complexity of theorem-proving procedures. In *Proc. 3rd ACM Symposium Theory of Computing*, pages 151–158, Shaker Heights, Ohio, 1971.

[Deg73] A.N. Degtev. *tt* and *m*-degrees. *Alg. Log.*, 12:143–161, 1973.

[Edm65] J. Edmonds. Paths, trees and flowers. *Canad. J. Math.*, 17:449–467, 1965.

[EFF82] P. Erdös, P. Frankl, and Z. Füredi. Families of finite sets in which no set is covered by the union of two others. *J. Combin. Theory Ser. A*, 33:158–166, 1982.

[EFF85] P. Erdös, P. Frankl, and Z. Füredi. Families of finite sets in which no set is covered by the union of r others. *Israel J. Math.*, 51:79–89, 1985.

[Ers71] Y.L. Ershov. Positive equivalences. *Alg. Log.*, 10:620–650, 1971. transl. 10 (1971) 378–394.

[FHOS93] S. Fenner, S. Homer, M. Ogiwara, and A. Selman. On using oracles that compute values. In *STACS 1993, Lecture Notes in Computer Science 665*, 1993.

[For79] S. Fortune. A note on sparse complete sets. *SIAM J. Comput.*, 8:431–433, 1979.

[Fri57] R.M. Friedberg. Two recursively enumerable sets of incomparable degrees of unsolvability. In *Proc. Nat. Acad. Sci.*, volume 43, pages 236–238, 1957.

[GH89] K. Ganesan and S. Homer. Complete problems and strong polynomial reducibilities. In B. Monien and R. Cori, editors, *STACS 89, Lecture Notes in Computer Science 349*, pages 240–250, 1989.

[GJY87] J. Goldsmith, D. Joseph, and P. Young. Self-reducible, p-selective, near-testable, and p-cheatable sets: The effect of internal structure on the complexity of a set. Technical Report TR #87-06-02, University of Washington, Seattle 98195, June 1987.

[GM74] J. Gill and P. Morris. On subcreative sets and s-reducibility. *J. Symbolic Logic*, 39(4):669–677, 1974.

[GW92] R. Gavaldà and O. Watanabe. On the computational complexity of small descriptions. *SIAM J. Comput.*, 1992. To appear.

[Hem87] L. Hemachandra. *Counting in Structural Complexity Theory.* PhD thesis, Cornell University, 1987.

[HHO$^+$92] L.A. Hemachandra, A. Hoene, M. Ogiwara, A.L. Selman, Th. Thierauf, and J. Wang. Selectivity. Manuscript, 1992.

[HIS85] J. Hartmanis, N. Immerman, and V. Sewelson. Sparse sets in NP-P: EXPTIME versus NEXPTIME. *Information and Control*, 65(2/3):158–181, May/June 1985.

[HKR] S. Homer, S. Kurtz, and J. Royer. A note on many-one and 1-truth table complete sets. *Theoret. Comput. Sci.* to appear.

[HL91] S. Homer and L. Longpré. On reductions of NP sets to sparse sets. In *Proc. Structure in Complexity Theory sixth annual conference*, pages 79–88. IEEE Computer Society Press, 1991.

[HOW92] L. Hemachandra, M. Ogiwara, and O. Watanabe. How hard are sparse sets? In *Proc. Structure in Complexity Theory seventh annual conference*, pages 222–238. IEEE Computer Society Press, 1992.

[Imm84] N. Immerman. Languages that capture complexity classes. *SIAM J. Comput.*, 16:760–778, 1984.

[Imm87] N. Immerman. Expressibility as a complexity measure: results and directions. In *Proc. Structure in Complexity Theory second annual conference*, pages 194–202. IEEE Computer Society Press, 1987.

[Joc68] C.G. Jockusch. Semirecursive sets and positive reducibility. *Trans. of the Amer. Math. Soc.*, 131:420–436, 1968.

[Kad87] J. Kadin. $\mathrm{P}^{\mathrm{NP}[\log n]}$ and Sparse Turing-complete Sets for NP. In *Proc. Structure in Complexity Theory second annual conference*, pages 33–40. IEEE Computer Society Press, 1987.

[Kan82] R. Kannan. Circuit-size lower bounds and non-reducibility to sparse sets. *Information and Control*, 55(1–3):40–56, October/November/December 1982.

[KL80] R. Karp and R. Lipton. Some connections between nonuniform and uniform complexity classes. In *Proc. 12th ACM Symposium on Theory of Computing*, pages 302–309, 1980.

[Ko83] K. Ko. On self-reducibility and weak P-selectivity. *J. Comput. System Sci.*, 26:209–211, 1983.

[Ko89] K. Ko. Distinguishing conjunctive and disjunctive reducibilities by sparse sets. *Information and Computation*, 81:62–87, 1989.

[Lad73] R. Ladner. Mitotic recursively enumerable sets. *J. Symbolic Logic*, 38(2):199–211, 1973.

[Lad75] R. Ladner. On the structure of polynomial time reducibility. *J. Assoc. Comput. Mach.*, 22:155–171, 1975.

[Lev73] L. Levin. Universal sorting problems. *Problemy Peredaci Informacii*, 9:115–116, 1973. in Russian.

[LLS75] R. Ladner, N. Lynch, and A. Selman. A comparison of polynomial time reducibilities. *Theoret. Comput. Sci.*, 1:103–123, 1975.

[Loz92] A. Lozano. *Computational Complexity versus Structural Simplicity.* PhD thesis, Universitat Politècnica de Catalunya, 1992.

[LV90] M. Li and P.M.B. Vitányi. Applications of Kolmogorov complexity in the theory of computation. In A. Selman, editor, *Complexity Theory Retrospective*, pages 147–203. Springer Verlag, 1990.

[Mah82] S. Mahaney. Sparse complete sets for NP: solution of a conjecture of Berman and Hartmanis. *J. Comput. System Sci.*, 25:130–143, 1982.

[Mar76] S.S. Marchenkov. One class of partial sets. *Mat. Zametki*, 20:473–478, 1976.

[Mat70] Y. Matijasevič. Enumerable sets are diophantine. *Soviet Math. Dokl.*, 11:354–357, 1970.

[MP79] A. Meyer and M. Paterson. With what frequency are apparently intractable problems difficult? Technical Report MIT/LCS/TM-126, M.I.T., 1979.

[Muc56] A.A. Muchnik. Negative answer to the problem of reducibility in the theory of algorithms. *Dokl. Acad. Nauk SSSR*, 108:194–197, 1956.

[Nai92] A.V. Naik. Personal communication. e-mail, 1992.

[NOS] A.V. Naik, M. Ogiwara, and A.L. Selman. P-selective sets, and reducing search to decision vs. self-reducibility. To appear in Proc. Structure in Complexity Theory eighth annual conference 1993.

[NW88] N. Nisan and A. Wigderson. Hardness vs. randomness. In *Proc. 29th IEEE Symposium on Foundations of Computer Science*, pages 2–11, 1988.

[Odi89] P. Odifreddi. *Classical Recursion Theory*, volume 125 of *Studies in Logic and the Foundations of Mathematics*. North-Holland, 1989.

[OW90] M. Ogiwara and O. Watanabe. On polynomial time bounded truth-table reducibility of NP sets to sparse sets. In *Proc. 22nd ACM Symposium on Theory of Computing*, pages 457–467, 1990.

[Pin84] J. Pintz. On primes in short intervals. *Studia Sci. Math. Hungar.*, 19:89–96, 1984.

[Pip79] N. Pippenger. On simultaneous resource bounds. In *Proc. 20th IEEE Symposium on Foundations of Computer Science*, pages 307–311, 1979.

[Pos44] E. Post. Recursively enumerable sets of integers and their decision problems. *Bull. Amer. Math. Soc.*, 50:284–316, 1944.

[Pri83] P. Pritchard. Prime number sieves. *J. Algorithms*, 4:332–344, 1983.

[Sac63] G.E. Sacks. On degrees less than $\mathbf{0}'$. *Ann. of Math.*, 2(77):211–231, 1963.

[Sal92] S. Saluja. Relativized limitations of left set technique and closure classes of sparse sets. To appear in proc. 8th structure in complexity theory conf., 1992.

[Sch86] U. Schöning. Complete sets and closeness to complexity classes. *Math. Systems Theory*, 19:24–41, 1986.

[Sch92] U. Schöning. n random reductions from sparse sets to tally sets. Technical report, Fakultät für Informatik, Universität Ulm, 1992.

[Sel79] A. Selman. P-selective sets, tally languages, and the behavior of polynomial time reducibilities on NP. *Math. Systems Theory*, 13:55–65, 1979.

[Sel82] A. L. Selman. Analogues of semicursive sets and effective reducibilities to the study of NP complexity. *Information and Control*, 52(1):36–51, January 1982.

[Sel88] A. L. Selman. Promise problems complete for complexity classes. *Information and Computation*, 78(2):87–97, August 1988.

[Sha90] A. Shamir. IP=PSPACE. In *Proc. 31st IEEE Symposium Foundations of Computer Science*, pages 11–15, 1990.

[Soa87] Robert I. Soare. *Recursively Enumerable Sets and Degrees.* Perspectives in Mathematical Logic. Springer-Verlag, 1987.

[Sol74] V. D. Soloviev. Q-reducibility and hyperhypersimple sets. *Probl. Meth. Cyb.*, 10:121–128, 1974.

[TFL91] S. Tang, B. Fu, and T. Liu. Exponential time and subexponential time sets. In *Proc. Structure in Complexity Theory sixth annual conference*, pages 230–237, 1991.

[Tur36] A.M. Turing. On computable numbers,with an application to the Entscheidungsproblem. *Proc. London Math. Soc.*, 2:230–265, 1936. addendum in 1937.

[Wat87a] O. Watanabe. A comparison of polynomial time completeness notions. *Theoret. Comput. Sci.*, 54:249–265, 1987.

[Wat87b] O. Watanabe. *On the Structure of Intractable Complexity Classes.* PhD thesis, Tokyo Institute of Technology, Jan. 1987.

[Wil85] C.B. Wilson. Relativized circuit complexity. *J. Comput. System Sci.*, 31:169–181, 1985.

[Yat65] C.E.M. Yates. Three theorems on the degrees of r.e. sets. *Duke Math. J.*, 32:461–468, 1965.

Nederlandse Samenvatting

In dit proefschrift bestuderen wij de polynomiale tijd begrensde reducties. Een dergelijke reductie induceert een equivalentie relatie op verzamelingen, zodat de equivalentie klassen, bestaande uit de verzamelingen die tot elkaar reduceren, partieel geordend kunnen worden. Wij bestuderen de structuur die een polynomiale reductie aanbrengt op tijdbegrensde complexiteits klassen.

Het centrale open problem binnen de computationele complexiteits theorie betreft de vraag of polynomiaal begrensde deterministische berekeningen even krachtig zijn als non-deterministische berekeningen, die polynomiaal begrensd zijn. Om een precies beeld te krijgen van de kracht van deterministische en non-deterministische berekeningen, zijn de klassen P en NP geintroduceerd. Het boven genoemde open probleem staat ook wel te boek als het P versus NP probleem.

Aangezien er nog geen duidelijke uitspraak is gedaan of de klassen P en NP verschillen – het probleem is te moeilijk om met de huidige kennis op te lossen – is de aandacht verschoven naar de klassen E en NE, die in de buurt van P en NP liggen. Ook voor deze klassen blijven veel problemen onopgelost. Desalniettemin is er enige vooruitgang te melden over de opheldering van de structuur, aangebracht door de polynomiale tijd reductie, op deze klassen. Er bestaat de hoop, dat de methoden en ideeën ontwikkeld voor het in kaart brengen van de structuur van deze klassen, enig licht zullen werpen op het centrale probleem.

In dit proefschrift zetten wij een kleine stap op de voorgestelde weg. Wij bewijzen in hoofdstuk 3 dat de structuur van de volledige verzamelingen voor verschillende type reducties voor NE gelijk is aan die van E. Opvallend is hier dat de 1-truth-table reductie dezelfde

volledigheids notie induceert als de many-one reductie, temeer daar niet bekend is of NE gesloten is onder complementatie.

In hoofdstuk 4 nemen wij de volledige verzamelingen voor E, NE en NP onder de loep. Wij proberen hier duidelijkheid te verkrijgen over de gevolgen van het hebben van redelijk dunne, subexponentieel dichte, volledige verzamelingen voor deze klassen. Wij hebben sterke aanwijzingen dat deze klassen niet anders dan dichte volledige verzamelingen kunnen bevatten. Dit generaliseert eerdere resultaten die behaalt zijn op dit gebied.

In hoofdstuk 5 gebruiken wij een complementaire strategie. Wij bekijken hier welke verzamelingen reduceren naar dunne, polynomiaal dichte, verzamelingen. Wij lossen twee open problemen op en laten o.a. zien dat elke polynomiaal dichte verzameling op zeer simpele wijze te coderen is in een zogenaamde 'tally' verzameling. Dit mag een verrassend resultaat genoemd worden daar algemeen vermoed werd dat een dergelijke codering onmogelijk was.

Hoofdstuk 6 kan gezien worden als een vervolg van hoofdstuk 4. In dit hoofdstuk echter bekijken wij andere – dan dichtheid – structurele eigenschappen van volledige verzamelingen. Wij bekijken hoe robuust volledige verzamelingen zijn tegen het wegnemen en toevoegen van een relatief gering aantal elementen. De resultaten geven aan dat sommige elementen cruciaal zijn voor de volledigheid van de verzameling. Aan de andere kant blijkt echter dat het moeilijk is om zulke 'hoekstenen' te vinden. Wij laten namelijk zien dat als geëist wordt dat de elementen in polynomiale tijd berekend moeten worden, dat dan voor een aantal volledigheids noties, volledigheid bewaard kan blijven. Interessant is dat een verder gaande generalisatie van de hier behaalde resultaten een ander open probleem zou oplossen: exponentiële tijd heeft geen polynomiale circuits. In dit hoofdstuk wordt verder aandacht besteed of het mogelijk is verzamelingen, in het bijzonder volledige, te splitsen in twee delen die een gelijke hoeveelheid computationele informatie bevatten. In het geval van een splitsing voor volledige verzamelingen betekent dit dat de delen wederom volledig zijn. Wij laten zien dat voor many-one volledige verzamelingen in E een dergelijke splitsing *altijd* mogelijk is, resulterend in het feit dat een volledige verzameling in oneindig veel stukjes verdeeld kan worden en dat ieder van de stukjes wederom volledig is. Interessant zijn deze splitsings resultaten met het oog op de eerder aangetoonde 'hoekstenen' in een volledige verzameling.

Hoofdstuk 7 benadert de structurele eigenschappen van een verzameling op een andere wijze met behulp van de zelfreductie. Een zelfreductie ordent niet twee verzamelingen maar geeft, voor één verzameling, een interne structuur aan. Wij bekijken welke verzamelingen in NP waarschijnlijk – een absoluut resultaat levert $P \neq NP$ – zelfreduceerbaar zijn. Verder geven wij een sterke karakterisering van de klasse P in termen van verzamelingen, die zowel zelfreduceerbaar als p-selectief zijn. Deze laatste notie is een polynomiale tijdbegrensde versie van de uit de recursie theorie bekende notie van semirecursiviteit. Vervolgens laten wij zien dat de klasse van p-selectieve verzamelingen gesloten is onder positieve Turing reducties. Tot slot geven wij, onder redelijke aannamen, een gedetailleerd beeld van de verschillende zelfreduceerbaarheids noties op NP.