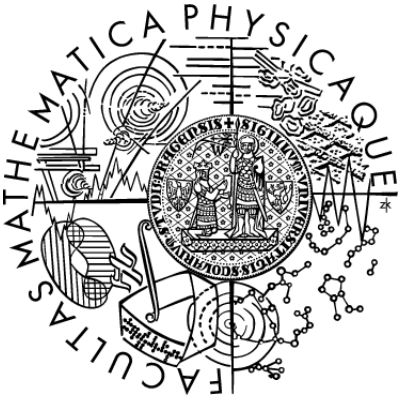
Charles University in Prague
Faculty of Mathematics and Physics

MASTER THESIS



Renáta Ševčíková

NP vyhledávací problémy a redukce mezi nimi

Department of Algebra

Supervisor of the master thesis: prof. RNDr. Jan Krajíček DrSc.

Study programme: Mathematics

Specialization: MMIB

Prague 2012

At this place I would like to sincerely thank my supervisor,
prof. RNDr. Jan Krajíček DrSc., for his valuable advices, comments and interesting insights which were very helpful in writing this thesis.
I thank also my family and friends, who supported me not only during writing the master thesis, but also in all my previous study.

I declare that I carried out this master thesis independently, and only with the cited sources, literature and other professional sources.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular the fact that the Charles University in Prague has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 paragraph 1 of the Copyright Act.

In Prague 13.04.2012 signature

Název práce: NP vyhledávací problémy a redukce mezi nimi

Autor: Renáta Ševčíková

Katedra: Katedra algebry

Vedoucí diplomové práce: prof. RNDr. Jan Krajíček DrSc.

Abstrakt: V předložené práci studujeme třídu Total NP vyhledávacích problémů. Větší pozornost je věnována studiu podtříd této třídy a redukcí mezi nimi. Kombinujeme známé metody: vyhledávací stromy a jejich vztah k redukcím, důkaz sporem pomocí Nullstellensatz důkazového systému a dolní odhad na stupeň tohoto důkazu pomocí designů, abychom ukázali, že dvě třídy relativizovaných NP vyhledávacích problémů založených na Mod-p počítacím principu a Mod-q počítacím principu, kde p a q jsou různá prvočísla, nejsou mezi sebou redukovatelné.
Práce je ukončena novým výsledkem separace pro $p = 2$ a $q = 3$.

Klíčová slova: NP vyhledávací problém, redukce, Modulo-p počítací princip, vyhledávací strom, Nullstellensatz

Title: NP search problems a reductions among them

Author: Renáta Ševčíková

Department: Department of Algebra

Supervisor: prof. RNDr. Jan Krajíček DrSc.

Abstract: In the thesis we study the class of Total NP search problems. More attention is devoted to study the subclasses of Total NP search problems and reductions among them.
We combine some known methods: the search trees and their relation to reductions, the Nullstellensatz refutation and the degree lower bound based on design to show that two classes of relativized NP search problems based on Mod-p counting principle and Mod-q counting principle, where p and q are different primes, are not reducible to each other.
This thesis is finished by a new separation result for $p = 2$ and $q = 3$.

Keywords: NP search problem, reduction, Modulo-p counting principle, search tree, Nullstellensatz

Contents

Introduction

To solve one of the most important questions in complexity theory, the question if P$=^?$NP, we must deal with complexity classes which lie between the classes P and NP and study their properties.

In this thesis we study the class of Total NP search problems, its various subclasses and reductions among them.
Since an unrelativized separation of any two NP search classes implies P$\neq$NP, the relativized separations are currently the best result we can hope for. Hence, we focus on proving some new relativized separations among Total NP search problems.
The needed technical ingredients were developed in proof complexity and are well-known. These include search trees and Nullstellensatz expressions. We use them to show a relativized separation between search classes of the so called Mod-p counting principles. This is summarized in detail in the last chapter.

The thesis is organized as follows:
In Chapter 1 we recall some definitions from complexity theory and define the relativized version of NP search problems. This version leads to reformulating the Turing and many-one reductions in terms of an oracle type-2 Turing machine and definition of classes C_m and C_T.
In Chapter 2 we become familiar with the class of Total NP search problems and its important subclasses and ways how to define some complete problem for each of these classes.
After that in Chapter 3 we look at the Mod-p counting principle in algebraic reformulation, Count_p^N, and show that the Nullstellensatz refutation is sound and complete as a proof system.
We use this fact to obtain a separation between two type-2 Mod-p counting problems in Chapter 4. This separation is based on search tree substitutions and a lower bound on the degree of Nullstellensatz refutation. To prove this lower bound we use a construction of design.
At the and of this thesis we summarize the argument in Chapter 5. In particular, we prove that if there is a Nullstellensatz proof that Count_2^N is total of polylogarithmic (in N) degree and any Nullstellensatz proof that Count_3^M is total requires quasipolynomial (in N) degree, then $\mathrm{Count}_2^N \not\leq_T \mathrm{Count}_3^M$.

1. Definitions

We assume that the reader is familiar with the basics of computational complexity theory. In particular, he knows the Turing machines and polynomial time algorithm, and on that basis can understand the definitions of complexity classes and reductions.

From now on we will use the abbreviation p-time in place of polynomial time. Let us denote by U_n the set $\{0,1\}^n$ of binary strings of length n. We say that a Turing machine runs in p-time if there is a polynomial p such that for each n and all inputs from U_n it runs in time $\leq p(n)$. In this sense n is a size parameter (if the computation time of a Turing machine is polynomial in n, then also the output-length is bounded by some $q(n)$, q polynomial).

The most fundamental complexity class is the class P, also known as PTIME. It contains all decision problems, which can be solved by a deterministic Turing machine in p-time. On the other hand we can define the class NP (NPTIME) as the set of all problems acceptable by a nondeterministic Turing machine in p-time. The most important question in complexity theory is if P $=^?$ NP
(P $\subseteq$ NP is trivial).

Because all definitions in this chapter are cited from [6], [3] and [1], the sources are not mentioned repeatedly in the following sections.

1.1 NP decision and search problems

To see the difference between NP decision and search problems let us define them in terms of a computation of some algorithm.
An NP decision problem is a problem of deciding whether there exists a solution to an instance of this problem. It means: "Given a p-time computable predicate R, polynomial p and some input $x \in U_n$, *decide* whether there exists some output y such that $|y| \leq p(|x|)$ and $R(x,y)$".
An NP search problem is a problem of finding a witness to the given NP property; i.e. the task is to *find* $|y| \leq p(|x|)$ such that $R(x,y)$ holds, if it exists.

NP search problems form the class FNP. They have been studied mostly in terms of their equivalent decision counterparts. For example, the problem SAT of deciding whether a given propositional formula is satisfiable by some 0/1 assignment and the problem SAT-SEARCH of finding some satisfiable assignment are polynomial equivalent (there are Turing reductions to each other in p-time).
But what happens if the solution always exists? In this case the search problem seems to have no polynomially equivalent decision problem. For example,

the problem of factoring some large (of the magnitude of around 300 decimal digits) composite number m is hard but the existence of the prime factors is always guaranteed. This search problem is important in public-key cryptography, namely in RSA cryptographic system; there is a large semiprime[1] and the task is to find its prime factors. Whether breaking RSA encryption is as hard as factoring is an open question known as the RSA problem.

Intuitively, it makes sense to distinguish NP search problems where the existence of a solution is guaranteed by some principle (they will form the class TFNP) from those which can be reduced to their decision counterparts. Now let us give the formal definition of the FNP and TFNP classes (FP $\subseteq$ TFNP $\subseteq$ FNP). The class FP is not important for us, but for completeness: FP is the set of functions computable by a p-time algorithm.

Definition 1.1 (The classes FNP and TFNP)**.**
Let $R(x, y)$ be a p-time predicate such that $R(x, y)$ implies $|y| \leq p(|x|)$, for some polynomial p. (We often suppress p.)
The NP search problem Q_R is: "Given instance x, find y such that $R(x, y)$ holds, if it exists".
The class of all NP search problems is denoted FNP.
Let $Q_R(x) = \{y|R(x, y)\}$. The problem Q_R is total, if $Q_R(x) \neq \emptyset$ for all x. Then TFNP is the set of all Total NP search problems.

In this definition TFNP problem is the so called *type-1* or unrelativized problem; the inputs are only binary strings - the type-0 objects. But it is more useful to work with the relativized version of this problem.
Beame et al. [2] reformulate the search problems in terms of *type-2* problems, denoted as TFNP2 problems. This class contains problems which take also string functions[2] and relations - the type-1 objects (presented as oracles) as part of input.
From now on we will consider the relativized version because it gives us a better view at the class of Total NP search problems and allows us to prove separations between them.

First we define the proper type-2 search problem as a function Q that associates with each set of string functions $\overrightarrow{f} = (f_1, \ldots, f_r)$ and a string $\overrightarrow{x} \in U_n^l$ a set $Q(\overrightarrow{f}, \overrightarrow{x})$ of possible answers to the problem Q on an input instance $(\overrightarrow{f}, \overrightarrow{x})$. Then we consider the relative complexity of the type-1 search problems by studying the relations between their associated type-2 problems.

Definition 1.2 (Type-2 search problem)**.**
Let $\overrightarrow{f} = (f_1, \ldots, f_r)$, $r \geq 0$, be the set of type-1 objects, $\overrightarrow{x} = (x_1, \ldots, x_l)$, $l \geq 1$ and $\overrightarrow{z} = (z_1, \ldots, z_s)$ two sets of type-0 objects, where $\forall i$ $f_i : U_n^{k_i} \to U_n$

[1] Semiprime is number with exactly two prime factors.
[2] String functions are functions whose arguments and values are strings

and $k_i \geq 1$ is the arity of the input of function f_i, $\forall j\ x_j \in U_n$ and $\forall m\ z_m \in U_n$. Assume that the property $\overrightarrow{z} \in Q(\overrightarrow{f}, \overrightarrow{x})$ is decidable by a type-2 Turing machine in p-time (type-2 Turing machines are described below). Then Q is the following type-2 search problem: "Given $\overrightarrow{f}$ and $\overrightarrow{x}$, find $\overrightarrow{z}$ such that $\overrightarrow{z} \in Q(\overrightarrow{f}, \overrightarrow{x})$".
If the set $Q(\overrightarrow{f}, \overrightarrow{x})$ is nonempty for all n, f_i and x_j (defined as above), then Q is total. The value s is the output arity of Q.

If $\overrightarrow{z} \in Q(\overrightarrow{f}, \overrightarrow{x})$ is a solution of Q on an input instance $(\overrightarrow{f}, \overrightarrow{x})$, it must be checkable in p-time and the verifying algorithm is allowed to access to oracles f_i, $1 \leq i \leq r$.

Before we define the many-one and Turing reductions between two type-2 search problems, it is useful to recall informally these reductions for type-1 problems.

We say that the type-1 problem $Q_1 = Q_1(\overrightarrow{x})$ is *Turing reducible* to $Q_2 = Q_2(\overrightarrow{y})$ if there exists a p-time oracle Turing machine M which on input $\overrightarrow{x} \in U_n^l$ produces some solution $\overrightarrow{z} \in Q_1(\overrightarrow{x})$ and is allowed to query an oracle Q_2 repeatedly. That means, M uses an oracle for solving $Q_2(\overrightarrow{y})$ as a subroutine. In this case we will write $Q_1 \leq_T Q_2$.
If M is allowed to query oracle Q_2 only once, we write $Q_1 \leq_m Q_2$ and say that Q_1 is *many-one reducible* to Q_2 by a p-time oracle Turing machine M.

Also in the relativized case a Turing machine uses one of the problems (an oracle for its solutions) as a subroutine to solve the other problem. It is much easier to understand the computation of type-2 Turing machine if we first consider only the type-1 oracle Turing machine M.

Now we describe the way how M uses an oracle $Q_2(\overrightarrow{g}, \overrightarrow{y})$ for solving the problem $Q_1(\overrightarrow{x})$. Such Turing machine M has one tape for its own string inputs $\overrightarrow{x}$, $|x_i| = n, i = 1, \ldots, l$, a special input tape for the string inputs to $Q_2 : \overrightarrow{y}$, $|y_j| = m, j = 1, \ldots, l'$ and tapes for each of the input functions g_k, $k = 1, \ldots, r'$ to Q_2.
M *presents* a query $(g_1, \ldots, g_{r'}, y_1, \ldots, y_{l'})$ to Q_2 where functions g_k, $1 \leq k \leq r'$, are represented by p-time Turing machines for solving g_k and $\overrightarrow{y}$ is simply a bit-sequence.
In the next step M receives an answer $\overrightarrow{t} \in Q_2(\overrightarrow{g}, \overrightarrow{y})$ which it uses to compute $Q_1(\overrightarrow{x})$. Calls to the oracle Q_2 (functions g_k) count as a single time step, so M runs in p-time; i.e. in time $\leq p(n)$, for some fixed polynomial p and n - the length of the input to M.

Let us consider a *type-2 Turing machine* M which solves some problem $Q_1(\overrightarrow{f}, \overrightarrow{x})$ and uses an oracle $Q_2(\overrightarrow{g}, \overrightarrow{y})$. The inputs to M include the strings

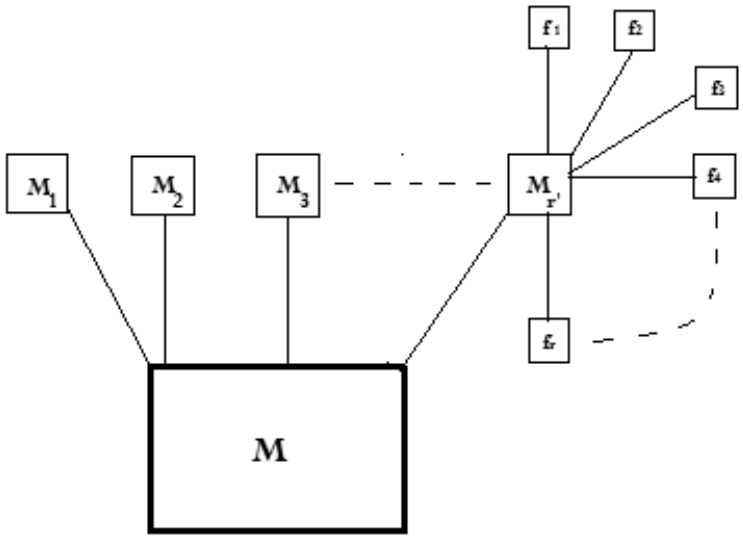


Figure 1.1: Computation of the type-2 Turing machine

$\overrightarrow{x} \in U_n^l$ and functions $\overrightarrow{f} = (f_1, \ldots, f_r)$, each f_i of arity k_i. Each of the function g_k, $k = 1, \ldots, r'$ (form part of the input to $Q_2(\overrightarrow{g}, \overrightarrow{y})$ problem) is described by oracle Turing machine M_k with r function oracles which is allowed to query the functions input to M, see Fig. 1.1. The runtime of M is defined as before. The calls to f_i count as a single time step, meaning f_i is an oracle and does not have a runtime.
Finally, we can define the many-one and Turing reduction in terms of type-2 problems.

Definition 1.3 (Many-one and Turing reduction)**.**
Let $Q_1 = Q_1(\overrightarrow{f}, \overrightarrow{x})$ and $Q_2 = Q_2(\overrightarrow{g}, \overrightarrow{y})$ be type-2 search problems.
We say that Q_1 is Turing reducible to Q_2, write $Q_1 \leq_T Q_2$, if and only if there exists a type-2 oracle Turing machine M that on given $(\overrightarrow{f}, \overrightarrow{x})$, instance of Q_1, outputs some $\overrightarrow{z} \in Q_1(\overrightarrow{f}, \overrightarrow{x})$ in p-time using $f_1, f_2, \ldots, f_r$ and Q_2 as oracles where every query to an oracle Q_2 is of the form $(\overrightarrow{g}, \overrightarrow{y})$, $|y| \leq p(|x|)$ for some polynomial p and with each function/relation $g_k, k = 1, \ldots, r'$, computable in p-time using $f_1, f_2, \ldots, f_r$ as oracles. Turing machines computing the functions $g_1, \ldots, g_{r'}$ are part of the reduction.
We say that Q_1 is many-one reducible to Q_2, write $Q_1 \leq_m Q_2$, if and only if there exists a type-2 oracle Turing machine M that is allowed to query Q_2 at most once.

After this important definition we can define the standard TFNP classes in terms of reductions to their complete type-2 problems. In this case the Turing machine M is type-1, the tapes for function inputs are not needed.

Definition 1.4 (Classes C_m and C_T)**.**
Let Q be a type-2 search problem. The class $C_m(Q)$, respectively $C_T(Q)$, is the set of all type-1 search problems which are many-one, respectively Turing reducible, to Q.
That means:

$$C_m(Q) = \{Q' | Q' \text{ is type-1 \& } Q' \leq_m Q\} \cap TFNP$$

and

$$C_T(Q) = \{Q' | Q' \text{ is type-1 \& } Q' \leq_T Q\} \cap TFNP.$$

In the next section we show how one can present a search problem Q in terms of first-order logic. This special representation of Q together with the definitions of Turing and many-one reduction implies interesting results in proof complexity theory.

1.2 Class TFNP and first-order formulas

Every TFNP2 problem (or TFNP class) can be defined also in terms of first-order logic. Consider the following formula as an example:

$$\Phi : f(0) = 0 \rightarrow \exists x[x \neq f(f(x)) \vee (x \neq 0 \wedge x = f(x))].$$

We interpret this formula expressing the parity principle in a structure with the universe U_n (in particular, it has even size).
It says that if the function f pairs the elements from the set U_n and pairs the element 0 ($=0^n$) with itself, then there exists another element paired with itself or some element x for which $f(f(x)) \neq x$. The universe U_n is given as a part of the input information about function f and there is clearly at least one solution from U_n.
This formula defines the problem Q_Φ, called **Lonely**. Since the set U_n has even cardinality, Lonely is total. This problem will be described in more detail in Section 2.1.

Generally, the first-order formulas will use *uninterpreted* function symbols (the function f above - type-1 input) and also *interpreted* constant symbols ($0 \equiv 0^n$,$1 \equiv 1^n$), relation symbols and function symbols ($f_< \equiv <$ and $f_\leq \equiv \leq$ corresponding to lexicographic ordering). The interpreted symbols depend only on the size parameter n, thus they have a fixed meaning in U_n. They are also called "built-in" symbols.

For simplicity, we may assume that there are no uninterpreted constant symbols and no relation symbols except "=" which always denotes the true equality. An uninterpreted constant symbol c can be replaced by unary function f_c and we set $f_c(0)$ in place of c. Relation symbol R may be replaced with a new function f_R for the graph of R and we set $f_R(\overrightarrow{t}) = 0$ in place of $R(\overrightarrow{t})$. This simplification allows the following

Definition 1.5 (Basic language)**.**
A language L is basic if it is finite and contains only the following symbols:
- built-in constant and function symbols
- one relation symbol: the equality symbol $(=)$
- finite number of non-built-in function symbols $f_1, \ldots, f_r$.

Definition 1.6 (Existential formula, existential sentence)**.**
An $\exists$-formula is a formula of the form $\exists \overrightarrow{x}\, \phi(\overrightarrow{x})$ where ϕ is quantifier-free. If $\exists \overrightarrow{x}\, \phi(\overrightarrow{x})$ has no free variables, then it is an $\exists$-sentence and is said to be total if it is true over all interpretations of the basic language in all U_n.

Definition 1.7 (Type-2 search problem in terms of $\exists$-sentences)**.**
Let Φ be an $\exists$-sentence $\exists \overrightarrow{x}\, \phi(\overrightarrow{x})$ in a basic language. The type-2 search problem Q_Φ is: "Given the string 0^n (n is the the size parameter) and interpretations for non-built-in function symbols f_i's in U_n, $i = 1, \ldots, r$, find $\overrightarrow{u} \in U_n$ such that $\phi(\overrightarrow{u})$ holds." If Φ is total[3], then $Q_\Phi \in$ TFNP2.

For simplicity we will assume that $\exists$-sentence $\Phi = \exists \overrightarrow{x}\, \phi(\overrightarrow{x})$ is in the prenex form, ϕ is in the DNF form:

$$\phi = \bigvee_{j \in J} \phi_j = \bigvee_{j \in J} (\bigwedge_{i=1}^{i_j} l_i)$$

and for all i

$$l_i = \begin{cases} h(\overrightarrow{u}) = v, & h \text{ is a function symbol and } v,\ \overrightarrow{u} \text{ are variables/constants} \\ w = v, & v,\ w \text{ are variables/constants} \\ w \neq v, & v,\ w \text{ are variables/constants} \end{cases}$$

Every literal l can be assumed to be in this form. For example formulas of the form $h(\overrightarrow{u}) \neq v$ can be replaced by $\exists x[h(\overrightarrow{u}) = x \wedge x \neq v]$ where x is new existentially quantified variable.

We follow Buss and Johnson [6] in showing that if $Q_\Phi \leq_T Q_\Psi$, then there is a low degree Nullstellensatz proof of combinatorial principle (Φ) expressing the totality of given search problem Q_Φ from combinatorial principle (Ψ) expressing the totality of the search problem Q_Ψ.

[3] $\exists \overrightarrow{x}\, \phi(\overrightarrow{x})$ is always true

2. The class TFNP and its subclasses

The content of this chapter is based on facts from [1], [3] and [6]. We refer to more specific sources at the beginning of every section.

The problems in the class TFNP have their totality guaranteed typically by some combinatorial principle. Because different combinatorial lemmas are being required for different problems, it seems unlikely that the class TFNP has any complete problem; this would correspond informally to an unintuitive assumption that there is the strongest combinatorial lemma.

There are several studied subclasses of the class TFNP. The exact computational complexity of these classes is unknown. Some problems in these classes are in P, some in NP. But we know that if P = NP, then all problems in TFNP class can be solved in p-time.
Hence, without showing P $\neq$ NP it cannot be established that no reduction between some classes exists. One can instead consider oracle separations, separations between two TFNP2 problems and hope it sheds light on their type-1 parts.
There are some known reductions between these classes, see Fig. 2.1 at the end of this chapter (p. 16).

We introduce three ways how to define a TFNP problem (the second one and the third one were described in previous chapter):

1, *Most intuitive:* by a combinatorial lemma, often from graph theory.

2, *Used in algorithm:* as a type-2 search problem $Q = Q(\overrightarrow{f}, \overrightarrow{x})$ described precisely in previous section (Definition 1.2).

3, *Usually used to obtain reductions:* as a type-1 translation Q of some natural type-2 problem given by first-order formula Φ :
$Q = C_m(Q_\Phi)$ (or $Q = C_T(Q_\Phi)$).

Remark 2.1. *In item 3, we reformulate some type-1 problem complete in a given TFNP subclass in terms of type-2 search problem and then redefine the subclass like in Definition 1.4.*
Every first-order formula (in some basic language) can be put into the form of the $\exists-$sentence/$\exists-$formula.

In the following sections we investigate some interesting subclasses of the class TFNP.

2.1 The classes PPA, PPAD and PPADS

These classes were first introduced by Ch.H.Papadimitriou in [11]. One can find some relations among these classes also in [1] and [3].

We start this section with a description of the class PPA (Polynomial Parity Principle) by a simple combinatorial principle.

Theorem 2.2 (Matoušek and Nešetřil [10])**.**
For every graph G=(V,E) where V is a nonempty set of nodes and $E \subseteq \binom{|V|}{2}$ *is the set of edges in an undirected graph G without loops, the equation*

$$\sum_{v \in V} deg(v) = 2.|E|, \tag{2.1}$$

holds.

Proof.
Each edge is counted into the degree of two nodes: its end-nodes. □

Remark 2.3. *There are some simple consequences of previous theorem:*

a, Parity principle: In every finite graph G the number of odd-degree nodes is even.

b, Every graph G with an odd number of nodes has a node with an even degree.

c, Let G=(V,E) be a graph with nodes of degree less or equal 1 and |V| be odd. There is no perfect matching of V.
A perfect matching is a subset of the edge set E such that each node in G is met by exactly one edge in the subset.

The problems in this class can be often based on special versions of Parity principle. For example the Smith's theorem: "Every graph with odd-degree nodes has an even number of Hamiltonian cycles through some given edge (x, y)."
Then the search problem **Smith** is: Given graph G with all nodes of odd degree and one Hamiltonian cycle, find another one. [11]

Another problem in this class is so called **Chess-player game** problem: Imagine the situation that everybody remembers all games of chess they have played in their life, you have played an odd number of games, and you must find a fellow odd player (known to exist by the Parity principle). It is based on a version of this principle: "All graphs of degree less or equal 2 have an even number of leaves (nodes of degree 1)." [11]

The problem **Leaf** is an example of a complete problem in the whole class PPA. It is based on special version of graph-theoretical lemma in Remark 2.3.

item a, where the graph has all nodes of degree at most 2. We know that there is some standard leaf (let say 0). The problem is to find another leaf in graph G.
For more examples of (complete) problems in class PPA see [11].

Now we show all types of possible definitions of the search problem in TFNP on the total problem called **Lonely** and we will use the first definition ("most intuitive") as the basic one.

1, Lonely is based on combinatorial lemma: Remark 2.3. item c,.
We can define this problem as a set of instances $(\alpha, \overrightarrow{x})$, $|\overrightarrow{x}| = n$, which define an undirected graph $G = (V, E)$, $V = U_n$, $E \subseteq \binom{|V|}{2}$, and an edge $(u, v) \in E$ if and only if $u \neq v$, $u = \alpha(v)$ and $v = \alpha(u)$.
We set $\deg(0^n)$=0, the standard lonely node. Then the set $U_n \setminus \{0^n\}$ has odd number of nodes and by Remark 2.3. item c, there must exist another lonely node v ($\deg(v)$=0). So, the problem Lonely is total in U_n.
The set of solutions to this problem on instance $(\alpha, \overrightarrow{x})$ is
Lonely$(\alpha, \overrightarrow{x})$=$\{v \in V : (v = 0^n$ if 0^n is not a standard lonely node) or $(v \neq 0^n$ if v is lonely node)$\}$

2, Lonely can be also characterized as a type-2 search problem Q. In this case the function α from the previous item is a part of the input to the Turing machine for solving Q. By Definition 1.2 we have Lonely$(\alpha, \overrightarrow{x})$= $Q(\alpha, \overrightarrow{x})$ where α is the only type-1 input with arity 1 and $\overrightarrow{x}$ codes the graph on nodes from the set U_n, $\overrightarrow{x} = (x_0, \ldots, x_l)$, $\forall i = 1, \ldots, l$: $x_i \in U_n$. We know that $\alpha(x_i)$ is either 0 or some x_j, $i \neq j$ and $\alpha(x_0) = 0$ (x_0 is the standard lonely node). We are looking for some node $y \in U_n$ such that $y \in Q(\alpha, \overrightarrow{x})$; i.e. $\alpha(y) = 0$, $y = x_k$ for some $k = 1, \ldots, l$ and this property can be decided by some type-2 Turing machine in p-time in n. The set $Q(\alpha, \overrightarrow{x})$ is nonempty because the cardinality of U_n without element x_0 is odd.

3, Lonely expressed in terms of first-order formula in some basic language with one non-built-in function symbol α was mentioned earlier (Section 1.2):

$$\Phi : \ \alpha(0) = 0 \wedge \forall x(x = \alpha(\alpha(x))) \rightarrow \exists x[x \neq 0 \wedge x = \alpha(x)]$$

and says that if every element is either lonely or is matched with a unique partner and if 0=0^n is lonely, then there exists another lonely element.
This formula defines type-2 search problem Q_Φ, called Lonely and because Lonely is total and complete following the Definition 1.4 we can define the class PPA:

Definition 2.4 (The class PPA)**.**
PPA=C_m(Lonely).

The Remark 2.3. item c, discusses the non-existence of a perfect matching on the set with odd elements. We can say that if the size of set N (e.g. $U_n \setminus \{0^n\}$) is not divisible by 2, then there is no partition of its elements into subsets of size 2. This principle is also called **Mod-2 counting principle** and the problem based on this principle is the **Mod-2 counting problem**, in this case it is the same problem as Lonely.

Following this idea we can define **Mod-p counting principle**for every prime p.

Definition 2.5 (Mod-p counting principle)**.**
Let N be the set of elements and p be some prime such that $|N| \not\equiv 0$ (mod p). Then there is no p-partition of N; i.e. the set N cannot be partitioned into subsets all of size p.

In this thesis we investigate **Mod-3 counting problem** prominently, hence we describe it a more detail.

Let us look at it in terms of graph theory (1, definition). We consider the set $V_n := \{0,1,2\}^n$. Like in the definition of problem Lonely we have some graph $G = (V_n, E)$ but in this case the degree of all nodes is at most 2. The set of instances $(\beta, \overrightarrow{x})$, $|\overrightarrow{x}| = n$, which define the graph G, is described in very similar way: an edge $(u, v) \in E$ if and only if $u \neq v$, $u \in \beta(v)$ and $v \in \beta(u)$ where β is a function assigning to a node u a subset $\beta(u) \subset V_n$ such that $|\beta(u)| \leq 2$[1].
We again set $\deg(0^n) = 0$. Hence, the set $V_n \setminus \{0^n\}$ is not divisible by 3. The problem of finding some non-zero node $u \in V_n$ such that $|\beta(u)| \leq 1$ (analogously $\deg(u) \leq 1$) is total and for illustration we called it **3-Lonely**. Now the set of solutions on an instance $(\beta, \overrightarrow{x})$ is:
3-Lonely$(\beta, \overrightarrow{x})$=$\{v \in V : (v = 0^n$ if $\deg(0^n) \neq 0$ and 0^n is not a standard 3-Lonely node) or ($v \neq 0^n$ if v is lonely node or $\deg(v) = 1)\}$

3-Lonely can be also characterized as a type-2 search problem Q_Ψ (Definition 1.2) or expressed by a first-order formula:

$$\Psi : \ \beta(0) = 0 \wedge \forall x(x = \beta(\beta(\beta(x)))) \rightarrow \exists x[x \neq 0 \wedge (x = \beta(x) \vee x = \beta(\beta(x)))].$$

Another complexity subclasses of TFNP are **PPAD** and **PPADS** - directed versions of PPA. One of the complete problems in the class PPAD is
SOURCEorSINK: Let G be some directed graph on nodes from U_n and for every node is the in-degree and out-degree ≤ 1. The solution of this problem is any sink (a node with in-degree 1 and out-degree 0) or any source (a node with in-degree 0 and out-degree 1) s if $s \neq 0^n$ or 0^n if it is not a source. [1]

[1]The image of β is a subset of V_n with zero, one or two nodes.

The example of complete problem in the class PPADS is called
SINK: We have a graph G like in previous case but now we are looking for some sink $s \neq 0^n$ or we put 0^n as a solution if it is not a source. [1]

It is easy to see ([1]) that

$$\text{SOURCEorSINK} \quad \leq_m \quad \text{Leaf},$$

by ignoring the direction information on the input graph. Also is easy to prove that

$$\text{SOURCEorSINK} \quad \leq_m \quad \text{SINK},$$

and a little harder is to show that

$$\text{SINK} \quad \not\leq_m \quad \text{Lonely}.$$

It holds that FP $\subseteq$ PPAD $\subseteq$ PPA $\subseteq$ TFNP $\subseteq$ FNP and it is not known if PPA = PPAD.

2.2 The class PLS

This class is another subclass of TFNP developed by Ch.H.Papadimitriou and precisely described in [7].
PLS (Polynomial Local Search) consists essentially of those local search problems for which local optimality can be verified in p-time. Every PLS problem can be either maximization or minimization problem.

Consider a typical search problem Q in class PLS. Each instance x (carries an information about some subset of U_n) of Q is associated with a set of *feasible solutions*, every such solution is polynomially bounded in $|x|$, each of them has a *cost* (computable in p-time) and the goal is to find some solution with minimal (maximal) *cost*.

In order to derive a local search algorithm we put on the solution set a so called *neighborhood* structure. This structure defines for each solution s some solution s' which is in some sense close to the given solution s. For example if we consider the graph on U_n nodes and the solution is some node v, its neighbor can be any node v' such that v and v' have the Hamming distance 2.

To make this class meaningful, we must make certain assumptions on the problem at the neighborhood structure:

- given an instance of Q, all solutions must have size bounded by a polynomial in the instance size and we must be able to produce some solution in p-time;

- given an instance and a solution, we must be able to compute the cost of the solution in p-time;
- given an instance and a solution, we must be able to determine in p-time whether that solution is locally optimal and if it is not, to generate a neighboring solution of an improved cost.

Let us now introduce some more useful notations.
For all $x \in U_n$, an input instance to the search problem Q, we have a finite set $F_Q(x)$ of feasible solutions and w.l.o.g. all with the same polynomially bounded length $q(|x|)$; that is for $x \in U_n$ $F_Q(x) \subseteq \{0,1\}^m$, for some $m = n^{O(1)}$.
If $s \in F_Q(x)$ is solution, we define cost function $C_Q(s,x)$ and also the neighbor of s $N_Q(s,x) \in F_Q(x)$ such that $C_Q(N_Q(s,x))$ is the better cost than $C_Q(s,x)$. We will write $C_Q(N_Q(s,x)) < C_Q(s,x)$.

Finding a local optimum for problem Q on instance x is done in three steps:

1, On $x \in U_n$ we produce a particular solution s ($s \in F_Q(x)$). We can always assume that $0 \in F_Q(x)$.

2, We compute its cost $C_Q(s,x)$ and some neighbor $N_Q(s,x)$.

3, Now there are two types of output depending on s:

3a, either there is some $s' \neq s, s' \in F_Q(x)$, $s' = N_Q(s,x)$ such that $C_Q(s',x) < C_Q(s,x)$ and we found the solution of better cost

3b, or we report that no such solution exists and hence that s is locally optimal.

This procedure is repeated starting with some canonical initial solution. The problem is solved by finding a locally optimal solution $s \in F_Q(x)$; i.e. an s such that $N_Q(s,x) = s$.
The local optimal solution must always exists because even when the costs are improving, they never go below zero. So, the problem Q of finding such solution is total and it is based on an iteration principle (we sequentially choose better solution).

The totality of this problem is also assured by the combinatorial principle: "Every directed acyclic graph with at least one edge has a sink".
This lemma expressed by first-order formula

$$\Phi: \; N(0) > 0 \wedge \forall x[C(N(x)) \leq C(x) \wedge C(x) \geq 0] \rightarrow \exists x(C(N(x)) = C(x)),$$

give rise to a complete type-2 search problem $Q_\Phi =$ **ITERATION**. Like in Definition 1.4 we obtain the following:

Definition 2.6 (The class PLS)**.**
PLS=C_m(ITERATION).

The iteration principle expressed by Φ involves an ordering relation and is defined only on finite structure. In [3] and [1] were shown some relativized reductions between complete problems from classes PPA, PPAD and PPADS and problem ITERATION. For example the following facts hold:

$$\begin{aligned} \text{Lonely} &\not\equiv_m \text{ITERATION and also} \\ \text{SOURCEorSINK} &\not\leq_m \text{ITERATION,} \\ \text{SINK} &\not\leq_m \text{ITERATION.} \end{aligned}$$

Again it is easy to see that FP $\subseteq$ PLS $\subseteq$ FNP.

If the reader is interested in more examples of complete problems in the class PLS, we recommend to Johnson, Papadimitriou and Yannakakis [7].

2.3 The class PPP

PPP (Polynomial Pigeonhole Principle) is a subclass of TFNP intuitively relevant to cryptographic hash functions. We do not know if PPP = FP, but if it is so then there exist no one-way permutations.
This class is based on a combinatorial principle: "There is no injective mapping from U_n to $U_n \setminus \{0\}$".

The main problem in the class PPP is the so called **PIGEON**. On the input to some algorithm we have function $f : U_n \to U_n$ where U_n denotes as usual the set of all n-bit strings and a string $\overrightarrow{x}$, $|\overrightarrow{x}| = n$ is the size parameter. The solution to this problem on given instance $(f, \overrightarrow{x})$, is any pair (v,v') such that $v \neq v'$ and $f(v) = f(v') \neq 0^n$ or any u with $f(u) = 0^n$.

This can be expressed by a first-order formula in the form

$$\Phi : \ \forall x(f(x) \neq 0) \Rightarrow \exists x, y[x \neq y \wedge f(x) \neq f(y)]$$

and because of the totality of Q_Φ = PIGEON we can set (using Definition 1.4):

Definition 2.7 (The class PPP)**.**
PPP = C_m(PIGEON).

In Section 2.1 we have shown some relations between classes PPA, PPAD, PPADS. It is not hard to prove that SINK $\leq_m$ PIGEON (see [1]). After showing this simple reduction we can illustrate the relations among all known TFNP subclasses by a set diagram (Fig. 2.1, p. 16).

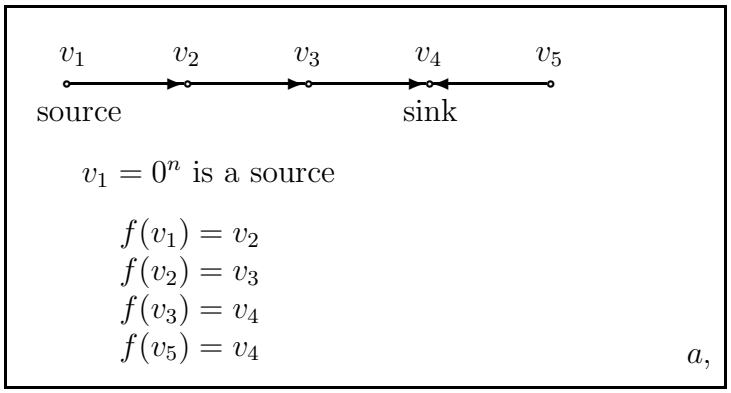
Theorem 2.8 (Beame et. al [1])**.**
SINK $\leq_m$ PIGEON.

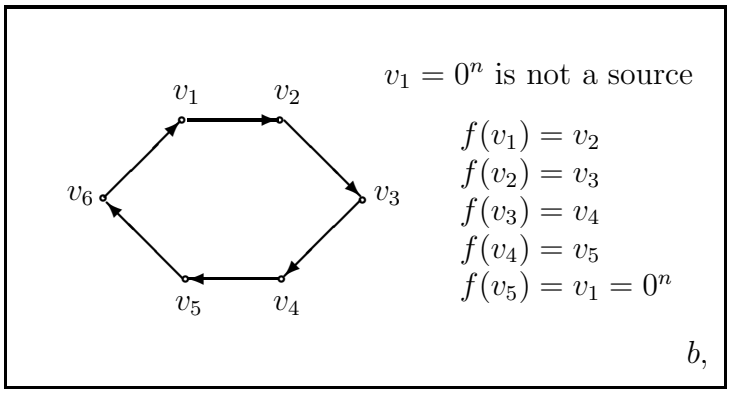
Proof. Let $G = (V, E)$ be a directed graph on $V = U_n$ nodes with in-degree and out-degree ≤ 1 for all $v \in V$. G is an input graph to SINK. We use an algorithm which solves PIGEON (call it P) as a subroutine in algorithm for solving SINK (call it S).

First we construct the function f (input to PIGEON):

- if the node $v \in V$ is a sink, then $f(v) := 0^n$,
- if there is an edge from u to v, then put $f(u) := v$ and
- if the node v is isolated ($\deg(v) = 0$), then put $f(v) := v$.

There are two cases which might happen. We illustrate them on small graphs a, and b, with only 5 (respectively 6) nodes:





In the case a, P responds (v_3, v_5): $f(v_3) = v_4 = f(v_5) \Rightarrow$ S responds node v_4. In the case b, the algorithm P responds v_5: $f(v_5) = 0^n \Rightarrow$ S responds 0^n which is again the right answer. This is made with one query to P. Hence, SINK $\leq_m$ PIGEON. □

Because the relation $\leq_m$ is transitive and we showed that

$$\begin{aligned} \text{SINKorSOURCE} &\leq_m \text{SINK and} \\ \text{SINK} &\leq_m \text{PIGEON in Section 2.1,} \\ \text{SINKorSOURCE} &\leq_m \text{PIGEON also holds.} \end{aligned}$$

It was shown (in [3]) that

$$\begin{aligned} \text{Lonely} &\not\equiv_m \text{PIGEON,} \\ \text{PIGEON} &\not\leq_m \text{ITERATION and} \\ \text{PIGEON} &\not\leq_m \text{SINKorSOURCE.} \end{aligned}$$

Some known relations between the class PLS and the classes PPA, PPAD, PPADS are discussed in the Section 2.2.

From the Definition 1.4 and known reductions between type-2 search problems discussed in previous sections we get interesting relations between the corresponding search classes in relativized world:

PPAD	$\subseteq$	PPA	PPA	$\not\subseteq$	PLS	PPP	$\not\subseteq$	PLS
PPAD	$\subseteq$	PPADS	PPAD	$\not\subseteq$	PLS	PPA	$\neq$	PLS
PPADS	$\subseteq$	PPP	PPADS	$\not\subseteq$	PPA	PPP	$\neq$	PPA
PPAD	$\subseteq$	PPP	PPP	$\not\subseteq$	PPAD			

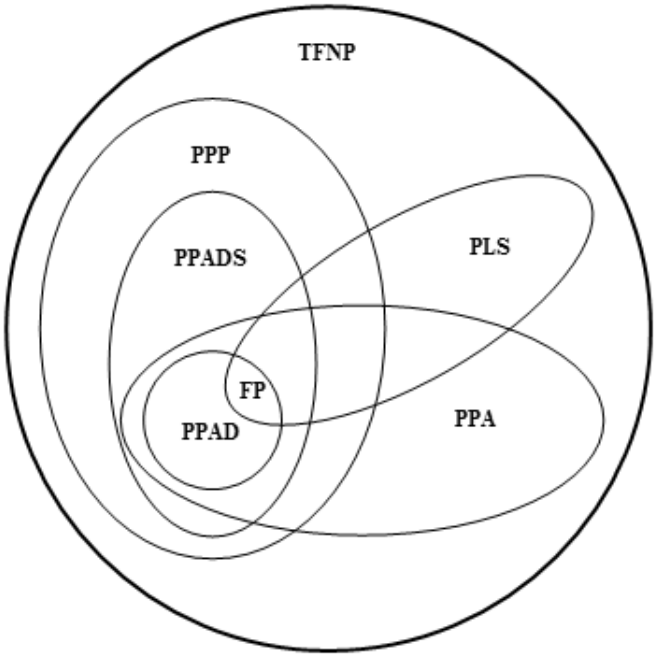


Figure 2.1: Relations between search classes in relativized world

3. Algebraic insight

In this chapter we introduce some algebraic proof systems, in particular, the Nullstellensatz refutations and we reformulate the Mod-p counting principle from an algebraic point of view. We draw on material from [5], [4] and [3].

3.1 Algebraic proof systems

Let R be a fixed commutative ring, usually Z_m for m integer or a prime (which implies Z_m is a field). We try to derive consequences from system of polynomial equations $\mathcal{P} = \{p_i(\overrightarrow{x}) = 0 | i = 1, \ldots, m\}$ using *equational reasoning* over ring R.

Equational reasoning is a useful tool used in automated theorem proving (in rewriting systems as Waldmeister). The ring structure provides some polynomial identities (equations) which always hold. For example, in commutative ring R with two binary operations "+" and ".", the equation $\forall x, y \in R$ $(x.y = y.x)$ holds.
Hence, we use all ring identities and identity laws (the ring axioms) to prove some properties (solvability/unsolvability) of the system of polynomial equations $\mathcal{P}$.

There are two types of algebraic proof systems:

- for proving unsolvability of systems of polynomial identities - Polynomial calculus
- for proving polynomial identities - Equational calculus

A specific example of an algebraic proof system which is contained in Polynomial calculus is the Nullstellensatz proof system discussed in the next section.

From now on we will assume that the ring R is of the form Z_p for p a prime. To prove the main theorem of this thesis it suffices to consider a field structure instead of a more general ring structure.
To enforce the variables having only 0/1 values, the system has always as axioms the equations $x_i^2 - x_i = 0$ for each x_i variable[1].

[1] Such proof system can serve as a propositional proof system.

3.2 The Nullstellensatz refutation

The Nullstellensatz proof system is a proof system for propositional logic based on algebraic identities in some fixed field F.
Boolean values *True* and *False* are identified with algebraic values 1 and 0. Boolean operations are expressed as field operations. We set $1 - x$ instead of $\neg x$, $x \cdot y$ instead of $x \wedge y$ and $x + y - x \cdot y$ instead of $x \vee y$.
The propositional formula $\varphi(\overrightarrow{x})$ is transformed into a polynomial $t(\overrightarrow{x})$ such that both expressions have the same value for all 0/1 assignment to all variables $\overrightarrow{x} = (x_1, \ldots, x_n)$.

Now consider some propositional formula $\varphi(\overrightarrow{x})$. The satisfiability of some formula (SAT) is an *NP-complete* problem in propositional logic. If $t(\overrightarrow{x})$ is the corresponding polynomial which computes the same function as $\varphi(\overrightarrow{x})$ for all Boolean inputs, then

$$\varphi \text{ is SAT } \Leftrightarrow t(\overrightarrow{a}) = 1, \text{ where } \overrightarrow{a} \text{ is some 0/1 assignemt to } \overrightarrow{x}.$$

The polynomial $t(\overrightarrow{x})$ can be w.l.o.g. replaced by several polynomials $t_1(\overrightarrow{x})$,...,$t_m(\overrightarrow{x}) \in F[\overrightarrow{x}]$ such that it holds:

$$\begin{aligned} \varphi \text{ is SAT} \quad \Leftrightarrow \quad & t_i(\overrightarrow{a}) = 1, \text{ for } \overrightarrow{a} \text{ some 0/1 assignment to } \overrightarrow{x} \\ & \text{and for all } i = 1, \ldots, m. \end{aligned}$$

If we recall that the algebraic structure we are working in is a field, then the system of equations $x_i^2 - x_i = 0$ for all $i = 1, .., n$ is satisfiable exactly when the variables x_i are set to 1 or 0. So, φ is unsatisfiable if and only if it is not possible to assign the field elements to the variables $\overrightarrow{x} = (x_1, \ldots, x_n)$ which simultaneously satisfy the $n + m$ equations

$$\begin{aligned} t_j(\overrightarrow{x}) - 1 &= 0 \qquad j = 1, \ldots, m \\ x_i^2 - x_i &= 0 \qquad i = 1, \ldots, n. \end{aligned}$$

Now we reformulate the Hilbert's Nullstellensatz (from [8]) to see the similarity with the above-mentioned SAT problem in terms of polynomials over a field F.

Theorem 3.1 (Hilbert's Nullstellensatz)**.**
Let F be an algebraically closed field and let $f_1, \ldots, f_m \in F[x_1, \ldots, x_n]$ be a list of polynomials of degree at most d.
Let $S = \{\overrightarrow{x} \in F | f_i(\overrightarrow{x}) = 0, i = 1, \ldots, m\}$. The set S is empty if and only if there are polynomials $g_1, \ldots, g_m \in F[x_1, \ldots, x_n]$ such that the following equality

$$1 = f_1 \cdot g_1 + f_2 \cdot g_2 + \cdots + f_m \cdot g_m$$

holds in the ring $F[x_1, \ldots, x_n]$.

In other words: The polynomials $f_1, \ldots, f_m \in F[x_1, \ldots, x_n]$ cannot be simultaneously equal to zero for any 0/1 assignment to variables $x_1, \ldots, x_n$ if and only if $1 \in \langle f_1, \ldots, f_m, x_1^2 - x_1, \ldots, x_n^2 - x_n \rangle$[2].

It is now easy to see that the satisfiability of φ expressed as the system of $n + m$ equations in $F[x_1, \ldots, x_n]$ is equivalent to question whether

$$1 \in \langle t_1, \ldots, t_m, x_1^2 - x_1, \ldots, x_n^2 - x_n \rangle;$$

i.e. whether there are polynomials p_j, $j = 1, \ldots, m$ and q_i, $i = 1, \ldots, n$ such that the polynomial identity

$$\begin{aligned} 1 = t_1(\overrightarrow{x}) \cdot p_1(\overrightarrow{x}) + t_2(\overrightarrow{x}) \cdot p_2(\overrightarrow{x}) + \cdots + t_m(\overrightarrow{x}) p_m \cdot (\overrightarrow{x}) + \\ (x_1^2 - x_1) \cdot q_1(\overrightarrow{x}) + (x_2^2 - x_2) \cdot q_2(\overrightarrow{x}) + \cdots + (x_n^2 - x_n) \cdot q_n(\overrightarrow{x}) \end{aligned} \tag{3.1}$$

holds in $F[\overrightarrow{x}] = F[x_1, \ldots, x_n]$.

Definition 3.2 (The Nullstellensatz proof and refutation)**.**
The Nullstellensatz proof (NP) of polynomial g from polynomials $t_1, \ldots, t_m$ is any set of polynomials p_j, $j = 1, \ldots, m$ and q_i, $i = 1, \ldots, n$ such that the polynomial identity

$$\begin{aligned} g = t_1(\overrightarrow{x}) \cdot p_1(\overrightarrow{x}) + t_2(\overrightarrow{x}) p_2 \cdot (\overrightarrow{x}) + \cdots + t_m(\overrightarrow{x}) p_m(\overrightarrow{x}) + \\ (x_1^2 - x_1) \cdot q_1(\overrightarrow{x}) + (x_2^2 - x_2) \cdot q_2(\overrightarrow{x}) + \cdots + (x_n^2 - x_n) \cdot q_n(\overrightarrow{x}) \end{aligned} \tag{3.2}$$

holds.
The Nullstellensatz refutation (NR) of polynomials $t_1, \ldots, t_m$ is the set of polynomials p_j, $j = 1, \ldots, m$ and q_i, $i = 1, \ldots, n$ such that the polynomial identity (3.1) holds.

In the Nullstellensatz proof system the proof is the set of polynomials $\{p_j, q_i\}$. Each of polynomials p_j, q_i is represented explicitly by its vector of coefficients of all monomials up to the degree of the polynomial. Hence, the size of the proof is determined by the largest degree. We thus concentrate on degrees of polynomials and will use the "degree of NP proof" in the place of proof size for NP proof. It is also more intuitive in the algebraic context.

Definition 3.3 (The degree of NP proof)**.**
The degree of NP proof $\{p_j, q_i\}$ (satisfy (3.2)) is defined as

$$deg_{NP}(\{p_j, q_i\}) = max\{max_{1 \le j \le m}\{deg(p_j) + deg(t_j)\}, max_{1 \le i \le n}\{deg(q_i) + 2\}\}.$$

The following lemma shows that the degree of the Nullstellensatz proof does not depend on the degrees $max_{1 \le i \le n}\{deg(q_i) + 2\}$, so we can define it simply as

$$deg_{NP}(\{p_j, q_i\}) = max_{1 \le j \le m}\{deg(p_j) + deg(t_j)\}$$

[2]Ideal generated by polynomials $f_1, \ldots, f_m, x_1^2 - x_1, \ldots, x_n^2 - x_n$ in $F[\overrightarrow{x}]$

Lemma 3.4 (Buss et al. [5])**.**
Suppose that there is a NR $p_1, \ldots, p_m, q_1, \ldots, q_n$ *of* $t_1, \ldots, t_m$ *and let* $d = max_{1 \leq j \leq m}\{deg(p_j) + deg(t_j)\}$.
Then there is another NR $p_1, \ldots, p_m, q_1', \ldots, q_n'$ *of* $t_1, \ldots, t_m$ *of degree* $\leq d$.

Proof. Consider the eguation (3.1) and denote $Q = 1 - \sum_{j=1}^{m} p_j.t_j$, $\deg(Q) = d$ and suppose $m \neq 0$. (If $m = 0 \Rightarrow 1 = Q$; i.e. $1 = \sum_{i=1}^{n} q_i \cdot (x_i^2 - x_i)$ and $1 \in \langle x_1^2 - x_1, \ldots, x_n^2 - x_n \rangle$ never holds.)
$Q \in \langle x_1^2 - x_1, \ldots, x_n^2 - x_n \rangle$ implies that Q cannot be multilinear, so it must contain some monomial of the form $x_{j_i}^2 \cdot x_{j_2} \cdot \ldots \cdot x_{j_k}$, $\{j_1, \ldots, j_k\} \in \{1, \ldots, n\}$. We can replace this monomial by $x_{j_i} \cdot x_{j_2} \cdot \ldots \cdot x_{j_k}$ and get another polynomial Q', $\deg Q' \leq d$ and the polynomial $Q - Q'$ has a Nullstellensatz derivation (refutation) from $x_{j_1}^2 - x_{j_1}$ of degree $\leq d$. □

Now we prove that the Nullstellensatz proof system is sound and complete as a refutation system following Buss [4].

Theorem 3.5 (Soundness of NR)**.**
If $t_1, \ldots, t_m$ *have a NR, then there is no 0/1 assignment to variables* $x_1, \ldots, x_n$ *that makes* $t_1, \ldots, t_m$ *simultaneously equal to zero.*

Proof. By contradiction:
Suppose $t_1, \ldots, t_m$ have a NR and there is some 0/1 assignment $\overrightarrow{a} \in \{0,1\}^n$ such that $t_1(\overrightarrow{a}) = 0$, $t_2(\overrightarrow{a}) = 0, \ldots, t_m(\overrightarrow{a}) = 0$.
There exist polynomials p_j, $1 \leq j \leq m$ and q_i, $1 \leq i \leq n$, for which the identity (3.1) holds, so we have

$$\begin{aligned} 1 &= \sum_{j=1}^{m} t_j(\overrightarrow{a}) \cdot p_j(\overrightarrow{a}) + \sum_{i=1}^{n} (a_i^2 - a_i) \cdot q_i(\overrightarrow{a}) \\ 1 &= \sum_{j=1}^{m} 0 \cdot p_j(\overrightarrow{a}) + \sum_{i=1}^{n} 0 \cdot q_i(\overrightarrow{a}) \\ 1 &= 0 \qquad \text{which is a contradiction.} \end{aligned}$$

□

Theorem 3.6 (Completeness of NR)**.**
Suppose that there is no 0/1 assignment to variables $x_1, \ldots, x_n$ *that makes* $t_1 = 0, t_2 = 0, \ldots, t_m = 0$. *Then* $t_1, \ldots, t_m$ *have a NR.*

Proof. The completeness of NR is simple consequence of Hilbert's Nullstellensatz. Namely, if there is no 0/1 assignment to variables $x_1, \ldots, x_n$ such that $t_1, \ldots, t_m$ cannot be simultaneously equal to zero, then $1 \in \langle t_1, \ldots, t_m, x_1^2 - x_1, \ldots, x_n^2 - x_n \rangle$. This implies the existence of polynomials p_j, q_i such that (3.1) holds. □

3.3 Algebraic formulation of the Mod-p counting principle

This principle was shortly described in Section 2.1 already. In this section we take a more detail look at it and reformulate this principle in terms of an (N, p)-polynomial system.

Let us fix some prime p and a large number $N \geq p$ such that $N \not\equiv 0 \ (mod\ p)$. The algebraic interpretation of Mod-p counting principle, $Count_p^N$, expresses the following tautology: A set with N elements cannot be partitioned into p-element subsets.

Because we will use the Nullstellensatz refutations to prove the totality of this principle, we will work with its negation and prove that the negation is not satisfiable ($\neg$ Count_p^N is not satisfiable $\Leftrightarrow$ Count_p^N is total).
The next definition specifies the conditions under which the set $[N] = \{1, \ldots, N\}$ has a partition into subsets of size p (shortly: $[N]$ has a p-partition).

Definition 3.7 (The (N,p)-polynomial system)**.**
Assume that $N \geq p$. The (N,p)-polynomial system is the following system of polynomial equations in variables x_e where e ranges over all p-element subsets of the set $[N]$.

$$Q_v: \quad \sum_{v \in e} x_e = 1, \ \textit{one for each } v \in [N] \tag{3.3}$$

$$Q_{e,f}: \quad x_e \cdot x_f = 0, \ \textit{for all } e, f \textit{ such that } e \perp f; \ \textit{i.e. } e \neq f \textit{ and } e \cap f \neq \emptyset \tag{3.4}$$

The equation (3.3) reflects the fact that every v belongs to one subset, the second equation (3.4) says that for every v this subset is exactly one. Then the variable x_e for which $v \in e$ equals 1.

Because the following equation for $v \in e$ holds:

$$\begin{aligned} x_e^2 - x_e &= -1 \cdot \sum_{v \in e \perp f} x_e \cdot x_f + x_e \cdot (\sum_{v \in f} x_f - 1) \\ &= -\sum_{v \in e \perp f} x_e \cdot x_f + x_e^2 + \sum_{v \in e \perp f} x_e \cdot x_f - x_e \end{aligned}$$

the polynomials $x_e^2 - x_e$ have a Nullstellensatz proof from Q_v, $Q_{e,f}$ of degree two[3] (Definition 3.3) and any solution of the (N, p)-polynomial system in any field F must be necessary the 0/1 solution. Then the set $\{e \mid x_e = 1\}$ forms a total p-partition of $[N]$ into p-element subsets. Hence, the (N, p)-polynomial system has no solution in any field F if $N \not\equiv 0 \ (mod\ p)$ ($\Rightarrow$ Count_p^N is total).

[3] In this case the polynomials $x_e^2 - x_e$ do not have to be added to the (N, p)-polynomial system for refutation.

Definition 3.8 (An instance of the (N, p)-polynomial system)**.**
An instance of the (N,p)-polynomial system is any polynomial system obtained from the (N,p)-polynomial system by substituting arbitrary polynomials for the variables x_e.

Beame et al. [2] have noticed a link between the degree of Nullstelensatz refutation of polynomials $Q_v, Q_{e,f}$ over a ring Z_m and constant-depth Frege proofs of Count_p^N from instances of Count_m^M where p is a prime and m some integer not divisible by p. They showed that

Theorem 3.9 (Beame et al. [2])**.**
If there is a NR of $Q_v, Q_{e,f}$ over Z_m of degree $d(N)$, then there is a depth l Frege proof of $Count_p^N$ from instances $\{Count_m^M : M \not\equiv 0 \ (mod\ m)\}$ of size at least $N^{\epsilon.d(N^\epsilon)}$ for $\epsilon > 0$ and $l \geq 1$ constant.

In the next chapter we avoid talking about Frege proof systems and just use the ideas behind the proof of this theorem to demonstrate a non-reduction result between relativized Mod-p and Mod-q counting principles (a separation of corresponding type-2 search problems Count_p^N and Count_q^M).

We will always assume that p and q are two distinct primes. Then the ring Z_q is a field and we can use NR as a sound and complete proof system. Moreover, it is intuitively clear, that the (N, p)-polynomial system is independent from the (M, q)-polynomial system and therefore we do not expect there to be p-time reductions between corresponding problems.

4. Separation of type-2 search problems Count_p^N and Count_q^M

We can prove the main result of this thesis already. To show that there is no reduction between Count_p^N and Count_q^M problems we use the Nullstellensatz refutations and substitutions based on search trees. After that we prove a lower bound for the degree of NR using a combinatorial construction called design; the proof includes an explicit construction of a design. Then in the last chapter we summarize the steps and apply them to the specific case:
a separation of type-2 search problems Count_2^N, $N = 2^n - 1$, and Count_3^M.

Let us start with a theoretical background about search trees. For more details we recommend to [6].

4.1 Search trees

Let M be a Turing machine with a string input x of size n and a function $f : U_n \to U_n$ as its oracle. We define the *search tree* $T_n^f(x)$ as a tree that encodes all possible computations of $M(x)$ in terms of f. Each internal node denotes a query to f with outgoing edges labeled with all 2^n possible ways the query can be answered. Every leaf node is labelled with the output value of $M(x)$ for the corresponding computation.
Analogously would be defined the tree $T_n^f(x)$ for $M(x)$ with more than one oracle function f.

Let denote $P_n^f(x)$ the *path* corresponding to a computation of $M(x)$ for given f. We identify $P_n^f(x)$ with a set of edge labels (the answers to oracle queries) in $T_n^f(x)$ starting in the root of the tree and ending in some leaf of $T_n^f(x)$. The set of all paths in $T_n^f(x)$ (for $M(x)$) is denoted $\mathcal{P}(T_n^f(x))$ and a set of all paths ending in a leaf with some fixed label w is denoted $\mathcal{P}_w(T_n^f(x))$.

The *height* of $T_n^f(x)$ is defined as the length of the longest path from the root to the leaf. One can see that it is equivalent to the maximum number of queries that $M(x)$ asks to f, taken over all functions f.
The *size* of $T_n^f(x)$ is defined as the number of all nodes in the search tree $T_n^f(x)$.

If $M(x)$ runs in p-time, the height of $T_n^f(x)$, denoted $h(T_n^f(x))$, is bounded by a polynomial $p(n)$ and the size of $T_n^f(x)$, denoted $s(T_n^f(x))$, is bounded by $2^{n.p(n)}$. If we set $N = 2^n$, N is called the *maximum branching factor*, then the height is polylogarithmic in N and the size is quasipolynomial in N. We will use the maximum branching factor N to measure the size and the height of tree.

From now on the string x will be some canonical word of given length n and we would not show it explicitly. So, the Turing machine with input x, will be simply denoted M, instead of $M(x)$. The same is true for its search tree T_n^f (in place of $T_n^f(x)$) and the path P_n^f (instead of $P_n^f(x)$).

Now assume that there exists a p-time type-2 oracle Turing machine M and $Q_1 \leq_m Q_2$ by M where $Q_1 = Q_1(\overrightarrow{f}, \overrightarrow{x})$, $|\overrightarrow{x}| = n$ and $Q_2 = Q_2(\overrightarrow{g}, \overrightarrow{y})$, $|\overrightarrow{y}| = m$, $m = n^{O(1)}$.
The computation of type-2 oracle Turing machine M was precisely described in Section 1.1. We can split M into Turing machines M_1 and M_2 such that M_1 simulates the computation of M except that, it terminates when M has produced the query $q = (\overrightarrow{g}, \overrightarrow{y})$ to Q_2. The Turing machine M_2 simulates the computation of M after it receives an answer $w \in Q_2(\overrightarrow{g}, \overrightarrow{y})$. The input to M_2 is of the form $(\overrightarrow{f}, \overrightarrow{x}, w)$.
Then T_n^{f,M_1} is a search tree for M_1 on $(\overrightarrow{f}, \overrightarrow{x})$ with leaves labeled by queries to Q_2 and $T_n^{f,M_2}(q, w)$ is search tree for the machine M_2 on $(\overrightarrow{f}, \overrightarrow{x}, w)$. If P_1 is the path in T_n^{f,M_1} ending with q and P_2 is some path in $T_n^{f,M_2}(q, w)$, then $P_1 \cup P_2$ naturally defines a solution to Q_1.

The computation of oracles $g_1, \ldots, g_{r'}$ can be also represented by search trees $T_m^{g_i(\overrightarrow{w})}$, $i = 1, \ldots, r'$ where $\overrightarrow{w} \in U_m$ is an input to g_i and there is an oracle Turing machine $M_{g_i}(\overrightarrow{w})$ computing $g_i(\overrightarrow{w})$ in p-time using $f_1, \ldots, f_r$ as oracles.
The internal nodes of $T_m^{g_i(\overrightarrow{w})}$ are labeled by queries to $f_1, \ldots, f_r$ and any leaf is labeled with an output value $z \in U_m$ ($g_i(\overrightarrow{w}) = z$, z is produced by $M_{g_i}(\overrightarrow{w})$). If $M_{g_i}(\overrightarrow{w})$ runs in p-time bounded by some polynomial p, then there is a canonical way of constructing a corresponding search tree of height at most p which faithfully represents the computation of $M_{g_i}(\overrightarrow{w})$.

Now we give some precise definitions which will be used in Section 4.2 to obtain a reduction. Before that, we introduce new variables. The fact that the oracle (for function f) answer on input $\overrightarrow{u}$ is v; i.e. $f(\overrightarrow{u}) = v$, will be represented by new variable $x_{\overrightarrow{u},v}$.

Definition 4.1 (Path)**.**
If P is a path in search tree $T_m^{g(\overrightarrow{w})}$ for some $g \in \{g_1, \ldots, g_{r'}\}$ and $\overrightarrow{w} \in U_m$, then we identify P with

$$\bigwedge_{i \in I} x_{\overrightarrow{u_i},v_i}$$

where $\{f(\overrightarrow{u_i}) = v_i\}_{i \in I}$ is set of all f values set by the edge lables of P.

Definition 4.2 (Substitution instance)**.**
The substitution instance $\sigma_{\mathcal{T}}$: $(x_{\overrightarrow{w},z})\sigma_{\mathcal{T}} = \lambda_{\mathcal{T},\overrightarrow{w},z}$ defined from a family of search trees $\mathcal{T} = \{T_m^{g(\overrightarrow{w})} | \overrightarrow{w} \in U_m, g \in \{g_1, \ldots, g_{r'}\}\}$ where $T_m^{g(\overrightarrow{w})}$ corresponds

to computation of $M_g(\overrightarrow{w})$ *is given by formula*

$$\lambda_{\mathcal{T},\overrightarrow{w},z} = \bigvee_{P \in \mathcal{P}_z(T_m^{g(\overrightarrow{w})})} P.$$

Let us define a search tree for the computation of a p-time type-2 oracle Turing machine M which computes a Turing reduction $Q_\Phi \leq_T Q_\Psi$. M makes queries to $f_1, \ldots, f_r$ ($Q_\Phi = Q_\Phi(\overrightarrow{f}, \overrightarrow{x})$, $|\overrightarrow{x}| = n$) and an oracle $Q_\Psi = Q_\Psi(\overrightarrow{g}, \overrightarrow{y})$. For simplicity let us consider there is only one type-1 input to Q_Φ: $f = f_1$ and only one type-1 input to Q_Ψ: $g = g_1$. The search tree T_n^f for computation of M is defined as follows:

a, each internal node is labeled either by some $f(\overrightarrow{u}) \in U_n$, $\overrightarrow{u} \in U_n$, with 2^n children and the outgoing edges labeled with all possible values of $f(\overrightarrow{u})$ or

b, by a query to $Q_\Psi(g, \overrightarrow{y})$ where g is described as p-time oracle Turing machine M_g.
Each such node has $2^{m.s}.J$ children ($\psi = \vee_{j \in J}\psi_j$, s arity of the output of Q_Ψ). The outgoing edges are labeled with all possible values $(j, \overrightarrow{b})$, $\overrightarrow{b} \in U_m$, $1 \leq j \leq J$; i.e. $\overrightarrow{b}$ is a solution to $Q_\Psi(g, \overrightarrow{y})$ such that $\psi_j(\overrightarrow{b})$ is true.
When M calls to the oracle Q_Ψ only the value $\overrightarrow{b}$ is returned but we can w.l.o.g. assume that there is also the value j such that $\psi_j(\overrightarrow{b})$ is true, since M can in p-time determine the value of j by evaluating $\psi(\overrightarrow{b})$.

The leaf nodes of T_n^f are labeled with output values of M - the solutions to Q_Φ. Because M runs in p-time $n^{O(1)}$, the height of T_n^f is also $n^{O(1)}$.

Definition 4.3 (Trace)**.**
Let ν *be a node of* T_n^f. P_ν *denotes the path from the root to the node* ν. *We define the set of traces* τ *to* ν *by induction on length of* P_ν. *Such trace to* ν *will consists of a sequence of literals* $x_{\overrightarrow{u},v}$ *(representing that* $f(\overrightarrow{u}) = v$*).*
The following construction may be used to form traces:

1, If ν *is a root and* P_ν *has length 0* $\Rightarrow$ *there is only one trace to* ν *equal an empty sequence.*

2, If ν *is an internal node labeled by* $f(\overrightarrow{u})$, ν' *is a child of* ν, *the edge* (ν, ν') *is labeled by* $v = f(\overrightarrow{u}) \in U_n$ *and* τ *is a trace to* $\nu \Rightarrow \tau, x_{\overrightarrow{u},v}$ *is a trace to* ν'.

3, If ν *is and internal node labeled with a query to* Q_Ψ *and* ν' *is a child of* ν, (ν, ν') *is labeled by* $(j, \overrightarrow{b})$ *and* $\psi_j(\overrightarrow{b})$ *contains* $h(\overrightarrow{u}) = v$ *or* $w = v$ *or* $w \neq v$ *which is false* $\Rightarrow$ *there is no trace to* ν.

4, Otherwise let us consider $\mathcal{T} = \{T_m^{g(\overrightarrow{w})} | \overrightarrow{w} \in U_m, g \in \{g_1, \ldots, g_{r'}\}\}$ *be a family of search trees for* g *and let* $g(\overrightarrow{w_i}) = z_i$, $i = 1, \ldots, k$ *be the atomic formulas of* ψ_j *that involve* g. *Then for any trace* τ *to* ν *and any choice of paths* $P_i \in \mathcal{P}_{z_i}(T_m^{g(\overrightarrow{w_i})})$ *the sequence* $\tau, P_1^\star, \ldots, P_k^\star$ *is the trace to* ν' *where* $P_i^\star$ *is a sequence of disjuncts contained in* P_i *for all* $i = 1, \ldots, k$. *(*$\psi_j(\overrightarrow{b})$ *is true if each* $g(\overrightarrow{w_i}) = z_i$ *holds.)*

4.2 Reduction by search tree substitutions

Many-one reductions between two TFNP2 problems Q_Φ and Q_Ψ coded as existential first-order formulas give rise to low-degree Nullstellensatz refutations ([6]). In this section we show that in case of Turing reductions, if $Q_\Phi \leq_T Q_\Psi$ and there are polylogarithmic degree Nullstellensatz proofs that Q_Ψ is total, then there are polylogarithmic degree Nullstellensatz proofs that Q_Φ is total.

In Section 3.2 we showed that Nullstelensatz refutation is sound and complete as a proof system, so we will proceed as follows:
In this section we will work in some fixed field F but we would not write it explicitly. Each of the problems $Q_\Phi = Q_\Phi(\overrightarrow{f}, \overrightarrow{x})$, $|\overrightarrow{x}| = n$ and $Q_\Psi = Q_\Psi(\overrightarrow{g}, \overrightarrow{y})$, $|\overrightarrow{y}| = m$ will have w.l.o.g. only one type-1 input.
At the beginning we define a set of polynomials $\mathcal{F}_\phi$ (the values of polynomials are conditional on the structure of a field) which when simultaneously equal 0 encode that the $\exists$-sentence Φ (defined in Section 2.1) is *not* total on n-length inputs. The polynomials in $\mathcal{F}_\phi$ are in variables $x_{\overrightarrow{u},v}$ where $\overrightarrow{u}, v \in U_n$ and think of them defining the graph of function f (the input function to Q_Φ):

$$x_{\overrightarrow{u},v} = \begin{cases} 1, & f(\overrightarrow{u}) = v \\ 0, & f(\overrightarrow{u}) \neq v \end{cases}$$

So, the variables are only 0/1 valued and to ensure this fact we define for all possible variables a set of polynomials

$$\{x_{\overrightarrow{u},v}^2 - x_{\overrightarrow{u},v} | \overrightarrow{u}, v \in U_n\} \tag{4.1}$$

which simultaneously equal 0 only if the variable-assignment is 0 or 1 for all variables. This polynomials form part of the set $\mathcal{F}_\phi$.

The function f must satisfied that it is total and single valued. For this reason it makes sense to add to $\mathcal{F}_\phi$ the sets of polynomials

$$\{\textstyle\sum_{v \in U_n} x_{\overrightarrow{u},v} - 1 | \overrightarrow{u} \in U_n\} \tag{4.2}$$

$$\{x_{\overrightarrow{u},v} \cdot x_{\overrightarrow{u},w} | \overrightarrow{u} \in U_n, v \neq w \in U_n\}. \tag{4.3}$$

If the polynomials in (4.2) equal 0, then there exists some variable $x_{\overrightarrow{u},v} = 1$, for each $\overrightarrow{u} \in U_n$, the function f is total and $f(\overrightarrow{u}) = v$. If the polynomials from the second set (4.3) simultaneously equal 0, then there is exactly one such variable $x_{\overrightarrow{u},v}$, for each $\overrightarrow{u} \in U_n$, the function f is single valued.
The similarity with polynomials Q_v and $Q_{e,f}$ is not coincidental. The set $\mathcal{F}_\phi$ is formed from exactly such polynomials, like those in Definition 3.8 of the (N, p)-polynomial system.

We remind the reader (it was defined in Section 1.2) that the existential first-order formula Φ expressing the totality of some search problem in DNF is

$$\exists \overrightarrow{x} \phi = \exists \overrightarrow{x} \bigvee_{j \in J} \phi_j = \exists \overrightarrow{x} \bigvee_{j \in J} (\bigwedge_{i=1}^{i_j} l_i)$$

and for all i, l_i is of the form $h(\overrightarrow{u}) = v$ or $w = v$ or $w \neq v$ where $\overrightarrow{u}$,w,v are either variables or constant symbols and h is function symbol in some basic language.
To express the fact that this formula is *not* total, we need to encode the fact that for all j, ϕ_j is false for all 0/1 assignment.

Let $\overrightarrow{u} = (u_1, \ldots, u_s)$ be a vector of s variables. Fix some $n \geq 1$ and let $\overrightarrow{a} = (a_1, \ldots, a_s)$ be a vector of fixed values from U_n.
If l_i is of the form $h(\overrightarrow{u}) = v$, then the Nullstellensatz translation under assignment $\overrightarrow{a}$ is $[l_i]_{\overrightarrow{a}} = x_{(\overrightarrow{u})_{\overrightarrow{a}},(v)_{\overrightarrow{a}}}$, meaning that the assignment $\overrightarrow{a}$ was applied to variables $\overrightarrow{u}, v$.
On the other hand if l_i is of the form $w = v$ or $w \neq v$ then the Nullstellensatz translation under the fixed assignment $\overrightarrow{a}$, $[l_i]_{\overrightarrow{a}}$, is either true or false and set to 1 or 0 accordingly.

We want to express that ϕ_j is false for each choice of j and 0/1-assignment $\overrightarrow{a}$ ($\forall j$ $\phi_j(\overrightarrow{a})$ is false $\Leftrightarrow$ $\phi(\overrightarrow{a})$ is false $\Leftrightarrow$ $\neg\phi(\overrightarrow{a})$ is true $\Leftrightarrow$ $\exists j$ $\neg\phi_j(\overrightarrow{a})$ is true).
We look at the negation of this formula and its expression under assignment $\overrightarrow{a}$. Following Buss and Johnson [6] we obtain:

$$[\neg\phi_j]_{\overrightarrow{a}} = \prod_{i=1}^{i_j} [l_{j,i}]_{\overrightarrow{a}}$$

and

$$[l_{j,i}] = \begin{cases} 1, & \text{the Nullstellensatz translation of } l_{j,i} \text{ under assignment } \overrightarrow{a} \text{ is true ,} \\ 0, & \text{the Nullstellensatz translation of } l_{j,i} \text{ under assignment } \overrightarrow{a} \text{ is false} \\ & \text{or } l_{j,i} \text{ is of the form } h(\overrightarrow{u}) \neq v \\ x_{\overrightarrow{u},v}, & l_{j,i} \text{ is of the form } h(\overrightarrow{u}) = v \end{cases}$$

Now we add to polynomials already in $\mathcal{F}_\phi$ also the set

$$\{[\neg\phi_j]_{\overrightarrow{a}}; 1 \leq j \leq J, \overrightarrow{a} \in U_n\}. \tag{4.4}$$

It is easy to see that if the last added polynomials all equal to zero, then some factor $[l_{j,i}]_{\overrightarrow{a}}$ of $[\neg\phi_j]_{\overrightarrow{a}}$ is zero and each solution to Q_Φ is ruled out. Hence, Q_Φ is not total problem.
The union of the polynomials in (4.1), (4.2), (4.3), (4.4), denoted $\mathcal{F}_\phi$, express the fact that Q_Φ is not total and the associated function f is single values and total. The main theorem of this section demonstrate the relation between Turing reductions of two type-2 search problems Q_Φ, Q_Ψ and a Nullstellensatz proof of the polynomial system $\mathcal{F}_\psi$ from the polynomial system $\mathcal{F}_\phi$.

In the proof of the main theorem we will use search trees and substitutions based on this search trees. As it was described in Section 4.1, the path P in a search tree is the set of f-values given by the edge labels in P and we identify it with $\prod_{i\in I} x_{\overrightarrow{u_i},v_i}$ where $x_{\overrightarrow{u_i},v_i}$ is again the variable corresponding to $f(\overrightarrow{u_i}) = v_i$. Similarly, the trace τ is defined as $\prod_{x_{\overrightarrow{u},v}\in\tau} x_{\overrightarrow{u},v}$.

Now we prove several simple lemmas that will be needed in the proof of main theorem:

Theorem 4.4 (Buss and Johnson [6])**.**
Let us have two type-2 search problems $Q_\Phi = Q_\Phi(f, \overrightarrow{x})$, $|\overrightarrow{x}| = n$ and $Q_\Psi = Q_\Psi(g, \overrightarrow{y})$, $|\overrightarrow{y}| = m$.
If there is a Turing reduction $Q_\Phi \leq_T Q_\Psi$ and a Nullstellensatz refutation of $\{\mathcal{F}_\psi\}$ of degree at most $m^{O(1)}$, then there is also a Nullstellensatz refutation of $\{\mathcal{F}_\phi\}$ of degree at most $n^{O(1)}$.

Lemma 4.5 (Buss and Johnson [6])**.**
For any search tree $T_m^{g(\overrightarrow{w})}$ of height at most d, the polynomial

$$\sum_{P\in\mathcal{P}(T_m^{g(\overrightarrow{w})})} P - 1$$

has a Nullstellensatz proof of degree at most d from the polynomials

$$\{\sum_{v\in U_n} x_{\overrightarrow{u},v} - 1 | \overrightarrow{u} \in U_n\}$$

which expressing the totality of function f ($Q_\Phi = (f, \overrightarrow{x})$).

Proof. By induction on the structure of $T = T_m^{g(\overrightarrow{w})}$.
If T is only a single node (the root) $\Rightarrow$ set $P = \{\emptyset\}$ $\Rightarrow$ polynomial $P = 1$ $\Rightarrow$ $\sum_{P\in\mathcal{P}(T)} P - 1 = 0 \in \langle\sum_{v\in U_n} x_{\overrightarrow{u},v} - 1\rangle$, $d = 0$.

Otherwise, pick some node ν labeled $f(\overrightarrow{u})$ whose children are all leaves. Define tree T' as T except the children of ν. Then by induction hypothesis $\sum_{P' \in \mathcal{P}(T')} P' - 1 \in \langle \sum_{v \in U_n} x_{\overrightarrow{u},v} - 1 \rangle$, $d' = h(T') \leq h(T)$.
Let P'_ν be a path in T' that ends in node ν. All paths in T and T' are the same except the P'_ν in T' which is in T replaced by $P'_\nu \cdot x_{\overrightarrow{u},v}$, for each $v \in U_n$ and $x_{\overrightarrow{u},v}$ - all children of node ν labeled $f(\overrightarrow{u})$ $(f(\overrightarrow{u}) = v)$.
Thus

$$\begin{aligned}
\sum_{P \in \mathcal{P}(T)} P - 1 &= \sum_{P' \in \mathcal{P}(T), P' \neq P'_\nu} P' + P'_\nu \cdot \sum_{v \in U_n} x_{\overrightarrow{u},v} - 1 \\
&= (\sum_{P' \in \mathcal{P}(T), P' \neq P'_\nu} P' - 1) + (\sum_{v \in U_n} x_{\overrightarrow{u},v} - 1) \cdot P'_\nu + P'_\nu \\
&= (\sum_{P' \in \mathcal{P}(T)} P' - 1) + (\sum_{v \in U_n} x_{\overrightarrow{u},v} - 1) \cdot P'_\nu
\end{aligned}$$

By induction hypothesis the first member of the right hand side has a Nullstellensatz proof of degree at most d' from $\{\sum_{v \in U_n} x_{\overrightarrow{u},v} - 1 | \overrightarrow{u} \in U_n\}$ and the second member has a Nullstelensatz proof of degree at most $|P'_\nu| + 1 = d' + 1 \leq h(T) = d$ from $\{\sum_{v \in U_n} x_{\overrightarrow{u},v} - 1 | \overrightarrow{u} \in U_n\}$. So the left hand side of the equation has a Nullstellensatz proof of degree at most d from the set $\{\sum_{v \in U_n} x_{\overrightarrow{u},v} - 1 | \overrightarrow{u} \in U_n\}$. □

Lemma 4.6 (Buss and Johnson [6])**.**
Take $\mathcal{T} = \{T_m^{g(\overrightarrow{w})} | \overrightarrow{w} \in U_m,\ h(T_m^{g(\overrightarrow{w})}) \leq d\}$ the family of search trees. Then any polynomial from the set

$$\{\sum_{v \in U_m} x_{\overrightarrow{u},v} - 1 | \overrightarrow{u} \in U_m\}$$

after applying substitution $\sigma_\mathcal{T}$ has a Nullstellensatz proof of degree at most d from polynomials

$$\{\sum_{v \in U_n} x_{\overrightarrow{u},v} - 1 | \overrightarrow{u} \in U_n\}$$

.

Proof. Fix some $\overrightarrow{u} \in U_m$ and set $p = \sum_{v \in U_m} x_{\overrightarrow{u},v} - 1$.
The polynomial p is in the set $\{\sum_{v \in U_m} x_{\overrightarrow{u},v} - 1 | \overrightarrow{u} \in U_m\}$ and by applying the substitution $\sigma_\mathcal{T}$ on it we get $p\sigma_\mathcal{T} = \sum_{v \in U_m} \lambda_{\mathcal{T},\overrightarrow{u},v} - 1$.
This is by definition of a substitution based on search trees (Definition 4.2) equal to $\sum_{v \in U_m} \sum_{P \in \mathcal{P}_v(T_m^{g(\overrightarrow{u})})} P - 1$.
The height of the tree $T_m^{g(\overrightarrow{u})}$ is bounded by d and P is by definition a polynomial in variables $x_{\overrightarrow{u},v}$ where $\overrightarrow{u}, v \in U_n$. □

Lemma 4.7 (Buss and Johnson [6])**.**
Take $\mathcal{T} = \{T_m^{g(\overrightarrow{w})} | w \in U_m\}$ and $m = n^{O(1)}$. Then any polynomial from the set

$$\{x_{\overrightarrow{u},v} \cdot x_{\overrightarrow{u},w} | \overrightarrow{u}, v \neq w \in U_m\}$$

after applying substitution $\sigma_{\mathcal{T}}$ has a Nullstellensatz proof of degree at most $n^{O(1)}$ from polynomials

$$\{x_{\overrightarrow{u},v} \cdot x_{\overrightarrow{u},w} | \overrightarrow{u}, v \neq w \in U_n\}.$$

Proof. Fix $\overrightarrow{u} \in U_m$ and put $X_1 = x_{\overrightarrow{u},v}$ and $X_2 = x_{\overrightarrow{u},w}$, $v \neq w \in U_m$. After applying the substitution $\sigma_{\mathcal{T}}$ on X_1 and X_2 we get $P_1 = X_1\sigma_{\mathcal{T}} = \lambda_{\mathcal{T},\overrightarrow{u},v}$ and $P_2 = X_2\sigma_{\mathcal{T}} = \lambda_{\mathcal{T},\overrightarrow{u},w}$. By the definition of a substitution based on search trees we have $P_1 \in \mathcal{P}_v(T_m^{g(\overrightarrow{u})})$ and analogously $P_2 \in \mathcal{P}_w(T_m^{g(\overrightarrow{u})})$.

Because P_1 and P_2 differ in the outputs, $v \neq w$, there must be a first query where they differ. Let us assume that it is the query $f(\overrightarrow{z})$; in P_1 is the answer y_1 ($f(\overrightarrow{z}) = y_1$) and in P_2 is the answer y_2 ($f(\overrightarrow{z}) = y_2$). Then P_1 include the factor $x_{\overrightarrow{z},y_1}$ and P_2 include the factor $x_{\overrightarrow{z},y_2}$.
Hence, the product $P_1 \cdot P_2 \in \{x_{\overrightarrow{u},v} \cdot x_{\overrightarrow{u},w} | \overrightarrow{u}, v, w \in U_m\}\sigma_{\mathcal{T}}$ and has a Nullstellensatz proof of the size $m = n^{O(1)}$ from the set $\{x_{\overrightarrow{u},v} \cdot x_{\overrightarrow{u},w} | \overrightarrow{u}, v, w \in U_n\}$. □

Lemma 4.8 (Buss and Johnson [6])**.**
Let us have a Turing reduction $Q_\Phi \leq_T Q_\Psi$ and suppose there is a Nullstellensatz refutation of $\mathcal{F}_\psi$ of degree $m^{O(1)}$. If τ is a trace, in the search tree T_n^f, to some general node ν, then the polynomial $\tau = \prod_{x_{\overrightarrow{u},v} \in \tau} x_{\overrightarrow{u},v}$ has a Nullstellensatz proof from $\mathcal{F}_\phi$ of degree at most $n^{O(1)}$.

Proof. At first let us consider the case when the trace τ has two factors $x_{\overrightarrow{u},v}$ and $x_{\overrightarrow{u},w}$. Like in Lemma 4.7 $\tau \in \langle \mathcal{F}_\phi \rangle$ of degree at most $n^{O(1)}$.
Now if τ is some trace in tree $T = T_n^f$ rooted in node ν with height h_ν and τ do not include the contradictory factors $x_{\overrightarrow{u},v}$ and $x_{\overrightarrow{u},w}$ we will proceed by induction on h_ν. The induction progresses from leaves to root. We remind that if the Turing machine associated with the tree T runs in p-time in n, then also the height of the tree is polynomial in n.

The base case is if ν is a leaf node of T. Then the trace τ ends in ν. When the computation of some Turing machine reaches the leaf node ν, it deterministically produces an output $\overrightarrow{a}$ that satisfied a member ϕ_j (from the first-order formula $\phi = \bigvee_{j \in J} \phi_j$). Because $\phi_j(\overrightarrow{a})$ is true, the Nullstellensatz translation $[\neg\phi_j]_{\overrightarrow{a}}$ under this assignment is false, so it is a non-zero polynomial. Each factor of polynomial $[\neg\phi_j]_{\overrightarrow{a}}$ must be also in trace τ and this implies $\tau \in \langle \mathcal{F}_\phi \rangle$ with the degree at most $n^{O(1)}$.

If ν is an internal node and is labeled with a query to $f(\overrightarrow{u})$ (f is part of input to Q_Φ), let ν' be the child of ν and the edge (ν, ν') labeled by answer $f(\overrightarrow{u}) = v$. By induction hypothesis $\tau \cdot x_{\overrightarrow{u},v}$ has a Nullstellensatz proof from $\{\mathcal{F}_\phi\}$ of degree at most $n^{O(1)} \cdot h_{\nu'} (= n^{O(1)})$.
Rewrite the trace τ as

$$\tau = \tau \sum_{v \in U_n} x_{\overrightarrow{u},v} + \tau \cdot (1 - \sum_{v \in U_n} x_{\overrightarrow{u},v}). \tag{4.5}$$

The first member of (4.5) has a Nullstellensatz proof from $\{\mathcal{F}_\phi\}$ of degree at most $max\ \{n^{O(1)} \cdot h_{\nu'}\ |\nu' \text{ is a child of } \nu\}$. The second member has a Nullstellensatz proof from $\{\sum_{v\in U_n} x_{\overrightarrow{u},v} - 1 | \overrightarrow{u} \in U_n\}$ of degree at most $n^{O(1)}$. Thus, τ has a Nullstellensatz proof of $\{\mathcal{F}_\phi\}$ of degree at most $max\{n^{O(1)} \cdot h_{\nu'} | \nu' \text{ is a child of } \nu\} + n^{O(1)} = n^{O(1)} \cdot h_\nu (= n^{O(1)})$.

Now, if the internal node ν is labeled by a query to $Q_\Psi(g, \overrightarrow{y})$ and ν' is again the child of ν with the edge labeled $(j, \overrightarrow{b})$; i.e. the p-time Turing machine M gives an assignment $\overrightarrow{b} \in U_m$ under which the disjunct ψ_j is true. Then the polynomial $[\neg\psi_j]_{\overrightarrow{b}}$ is non-zero and we suppose it is equal to $\prod_{i\in I} x_{\overrightarrow{u_i},v_i}$.
Then $\forall j, \overrightarrow{b}$: $\tau \cdot ([\neg\psi_j]_{\overrightarrow{b}})\sigma_\mathcal{T}$ and assume that $([\neg\psi_j]_{\overrightarrow{b}}) = \prod_{i=1}^{l} x_{\overrightarrow{u_i},v_i}$. After rewriting $\tau \cdot (\prod_{i=1}^{l} x_{\overrightarrow{u_i},v_i})\sigma_\mathcal{T}$ in terms of substitution we have

$$\tau \cdot \prod_{i=1}^{l} \sum_{P_i \in \mathcal{P}_{v_i}(T_m^{g(\overrightarrow{u_i})})} = \sum_{P_1 \in \mathcal{P}_{v_1}(T_m^{g(\overrightarrow{u_1})}),\ldots,P_l \in \mathcal{P}_{v_l}(T_m^{g(\overrightarrow{u_l})})} \tau \cdot \prod_{i=1}^{l} P_i.$$

Each term of this sum lies (by induction hypothesis) in ideal generated by $\mathcal{F}_\phi$ and thus has a Nullstellensatz proof of $\{\mathcal{F}_\phi\}$ of degree at most $n^{O(1)}$.

It is easy to see (for more detail we recommend to [6]) that if there is a Nullstellensatz refutation of $\{\mathcal{F}_\psi\}$ of degree at most $m^{O(1)}$ $(m = n^{O(1)})$, then there is a Nullstellensatz refutation of $\{(\mathcal{F}_\psi)\sigma_\mathcal{T}\}$ of degree at most $m^{O(1)} + n^{O(1)}$. (After the substitution of the $\lambda_{\mathcal{T},\overrightarrow{u},v}$'s for the variables $x_{\overrightarrow{u},v}$'s is the degree of NR of $\{(\mathcal{F}_\psi)\sigma_\mathcal{T}\} \leq m^{O(1)} + n^{O(1)}$.)
If we rewrite this fact we obtain the following equation

$$1 = \sum_{\substack{p_i \in \{\sum_{v\in U_m} x_{\overrightarrow{u},v} - 1 | \overrightarrow{u} \in U_m\} \cup \\ \{x_{\overrightarrow{u},v} \cdot x_{\overrightarrow{u},w} | \overrightarrow{u} \in U_m, v \neq w \in U_m\}}} f_i \cdot (p_i\sigma_\mathcal{T}) + \sum_{q_i \in \{[\neg\psi_j]; 1\leq j\leq J, \overrightarrow{b} \in U_m\}} g_i \cdot (q_i\sigma_\mathcal{T}).$$

By previous lemmas, the first summation has a Nullstellensatz proof from $\{\mathcal{F}_\phi\}$ of degree at most $n^{O(1)}$ and by multiplying both sides by τ the second summation (from already proved) has a Nullstellensatz proof from $\{\mathcal{F}_\phi\}$ of degree $max\{n^{O(1)} \cdot h_{\nu'} |\ \nu' \text{ is a child of } \nu\}$.
Hence, together we get that τ has a Nullstellensatz proof from $\{\mathcal{F}_\phi\}$ of degree $max\{n^{O(1)} \cdot h_{\nu'} | \nu' \text{ is a child of } \nu\} + n^{O(1)} = n^{O(1)} \cdot h_\nu = n^{O(1)}$. □

Applying Lemma 4.8 to the trace τ to the root we get the proof of the main Theorem 4.4.

This theorem implies the following

Corollary 4.9. *If there is a Nullstellensatz proof that Q_Ψ is total of degree at most $n^{O(1)}$ and any Nullstellensatz proof that Q_Φ is total requires degree $m^{O(1)} = 2^{n^{O(1)}}$, then $Q_\Phi \not\leq_T Q_\Psi$.*

4.3 Design based degree lower bound

Designs are collections of subsets with certain intersection properties. The part of mathematics which deals with the existence and construction of such systems is called Combinatorial design theory. This theory has interesting applications in cryptography (authentification codes, threshold schemes), coding theory, group testing, mathematics biology et cetera. At the present time designs are also the most useful tool for proving the lower bounds for the degree of Nullstellensatz refutations.
In this section we draw on material from [4], [2] and [5].

Let us denote by $F[\overrightarrow{x}]_{\leq d}$ the set of all polynomials over some fixed field F of degree less or equal d.

Definition 4.10 (Design)**.**
Let F be some fixed field, $\mathcal{T} = \{t_i\}_{i=1}^{n}$ polynomials from $F[\overrightarrow{x}]_{\leq d}$ and $d \geq 0$. A d-design for $\mathcal{T}$ is a mapping $D : F[\overrightarrow{x}]_{\leq d} \to F$ such that the following conditions hold:

1. *$D(1) = 1$.*
2. *$\forall\ a \in F, \forall\ p,q \in F[\overrightarrow{x}]_{\leq d}$: $D(a \cdot p) = a \cdot D(p)$ and $D(p+q) = D(p)+D(q)$; meaning D is a linear mapping.*
3. *$\forall\ t \in \mathcal{T}, \forall\ p \in F[\overrightarrow{x}]_{\leq d}$ such that $deg(t \cdot p) \leq d$: $D(t \cdot p) = 0$.*
4. *$\forall\ x$ variable, $\forall\ p \in F[\overrightarrow{x}]_{\leq d-2}$: $D(x^2 \cdot p) = D(x \cdot p)$.*

By including polynomials $x_i^2 - x_i$ for all variables x_i into the set $\mathcal{T}$ and because D is a linear mapping, D is completely determined only by values $D(p)$ where p is some multilinear monomial over F of degree less or equal d.

Remark 4.11. *Designs can be also defined on the propositional statements represented by propositional variables. This is easy to see when one realizes the relation between monomials and propositional statements. For example the monomial $x_1 \cdot x_2 \cdot (1 - x_3)$ can be transformed into propositional term $x_1 \wedge x_2 \wedge \neg x_3$.*
As we will see below designs can be also defined on sets with less or equal d elements.

Before formulating and proving the theorem showing a relation between designs and the degree of Nullstellensatz refutation of the set of polynomials $\mathcal{T}$ it is necessary to understand the idea of *Integer linear programming problem.*

Integer linear programming problem (ILP) is the problem of finding the optimal solution (the maximum or the minimum integer value) of some given linear expression with a set of additional constraints. This problem is given by

a rational matrix $\mathcal{A} = \{a_{i,j}\}_1^{n,m}$ and a rational vector $\overrightarrow{B}$. The task is to find a solution, an integer $\overrightarrow{x}$, (or to determine if it exists) to the set of inequalities

$$\sum_{j=1}^{m} a_{i,j}.x_j \leq B_i, i = 1, \ldots, n.$$

By adding an extra condition $0 \leq x_j \leq 1$ for all $j = 1, \ldots, m$ we may restrict the set of solutions only to 0's and 1's.
We will see the connection between ILP problem and NR in the second part of the following theorem's proof.

Theorem 4.12 (Buss [4])**.**
The set of polynomials $\mathcal{T}$ over a field F does not have a NR of degree less or equal d if and only if this set admits a d-design.

Proof. The necessary condition will be proven by contradiction.
Suppose the set $\mathcal{T}$ has a NR of degree $\leq$ d. Then there exist $p_j, q_i \in F[\overrightarrow{x}]$ (Section 3.2) such that

$$\begin{aligned} 1 &= \sum_j p_j \cdot t_j + \sum_i q_i \cdot (x_i^2 - x_i) \\ 1 = D(1) &= D(\sum_j p_j \cdot t_j) + D(\sum_i q_i \cdot (x_i^2 - x_i)) \\ 1 &= \sum_j D(p_j \cdot t_j) + \sum_i D(q_i \cdot x_i^2) - D(q_i \cdot x_i) \\ 1 &= 0 \qquad \text{which is a contradiction.} \end{aligned}$$

To prove the sufficient condition we use the fact that the question whether there exists a NR of degree $\leq d$ turns into a question of finding a solution to a ILP problem. Then the question whether a d-design exists turns out to be the dual problem.

Let $\mathcal{S} = \{p \mid p \text{ monomial, } \deg(p) \leq d\}$ and $s = |\mathcal{S}|$. All this monomials can be ordered by their degree such that the constant monomial will be the last. Then every polynomial q, $\deg(q) \leq d$ can be written as a vector $\overrightarrow{v_q}$ in basis $\mathcal{S}$ and $\dim(\overrightarrow{v_q})$=$s$.

Now let $\mathcal{G} = \{p{\cdot}t \mid p \in F[\overrightarrow{x}],\ t \in \mathcal{T}$ or of the form $x^2{-}x$ and $\deg(p{\cdot}t) \leq d\}$ and denote $g = |\mathcal{G}|$. If we express every $G_i \in \mathcal{G}$ as a column vector in basis $\mathcal{S}$, we obtain a $(s{\times}g)$-matrix M (rows $\overrightarrow{m_p}$, $p \in S$ monomial, columns $\overrightarrow{v_{G_i}}$, $G_i \in \mathcal{G}$).

There is a NR of degree $\leq d$ if and only if $\overrightarrow{v_1} = (0, \ldots, 0, 1)$, $\dim(\overrightarrow{v_1}) = s$, can be expressed as F-linear combination of $\overrightarrow{v_{G_i}}$, $G_i \in \mathcal{G}$. This is equivalent to: $\exists\ \overrightarrow{u} = (u_1, \ldots, u_g)$ from F such that $M \cdot \overrightarrow{u} = \overrightarrow{v_1}$ (ILP).

Let M^- be the matrix which arises from M by cutting out the last row. The vector $\overrightarrow{v_1}$ has non-zero number only on its s-th place, so the vector $\overrightarrow{u}$ must be in the Nullspace of the mapping M^-. (It is easy to see that the mapping and matrix are equivalent in the sense that one can express the mapping by a matrix and the matrix on basis vectors defines a mapping).

Now two cases can occur:

a, the last row of M lies in the linear span of the first $s-1$ rows, then $M^- \cdot \overrightarrow{u} = \overrightarrow{0} \Rightarrow M \cdot \overrightarrow{u} = \overrightarrow{0}$.

b, the last row of M is not in the linear span of the first $s-1$ rows. Denote $U = \{u \mid M^- \cdot \overrightarrow{u} = \overrightarrow{0}\}$. It is possible to pick from U some vector $\overrightarrow{u}$ for which $M \cdot \overrightarrow{u} = \overrightarrow{v_1}$ holds.

So there is no NR of degree $\leq d$ if the last row is F-linear combination of the previous rows. It remains to show that then there exists a d-design. (dual ILP)

Each row is a vector $\overrightarrow{m_p}$ for some monomial p, $\deg(p) \leq d$. In the case a, since the rows in M are linearly dependent we have

$$\sum_{deg(p) \leq d} a_p \cdot \overrightarrow{m_p} = \overrightarrow{0},\ a_p \in F,\ a_1 = 1 \text{ for } \overrightarrow{m_1} \text{ (1 the constant monomial).}$$

Now it is obvious how to define a d-design D : $D(p) = a_p$ for each monomial p over F. ($\deg(p) \leq d$)
It is easy to see that D is correctly defined design ($D(1) = 1$, linear, well defined on minimal monomials of degree less than d). □

We apply the Definition 4.10 of design on the (N, p)-polynomial system over Z_q, p, q different primes which encodes the conditions under which there is a p-partition of $[N]$ over the field Z_q. The place of monomials $\mathcal{T} \subseteq \mathcal{F}_{\leq d}[\overrightarrow{x}]$ is taken by partial p-partitions of $[N]$. Let denote the set of all such partition $\mathcal{P}$.

Every element in $\mathcal{P}$ is some set P of less or equal d p-element subsets of $[N]$, $N > p$. Then it is easy to modify the definition of the design in terms of these sets in the following sense:

- if $P \in \mathcal{P}$, then $D(P) = D(t) \in Z_q$ where $t \in \mathcal{T}$ and $\deg(t)$ = the number of p-element subsets in P,
- if P is empty, then $D(\emptyset) = D(1) = 1$ and
- if P has less than d p-element subsets and v is some element which is not in any of the subset of P we put $D(P) = \sum_{v \in e, e \cap [P] = \emptyset} D(P \cup e)$ where e is p-element subset not in the set P.

The last option says that if there is some partial p-partition P and we can improve it by adding an extra p-element subset to the given p-partition, we just do it and the value of $D(P)$ will not change.

The lower bound for degree of NR by constructing the design was proven for general primes p, q and N ($N \not\equiv 0 \ (mod\ p)$) by Buss et al. in [5]. We apply this proof to a special case of $N = 2^n - 1$ ($n > 1$ integer), $p = 2$ and $q = 3$ and fill in some details of the original proof. At the start of the construction let us introduce some notation:

Let $\mathcal{S}$ be a set of $5N$ elements. Fix a permutation π on $\mathcal{S}$. For a given element $s \in \mathcal{S}$ we put $s^0 = s$, $s^1 = \pi(s), \ldots, s^5 = s^0$. The cycles of π have length 5 and $s^{-i} = s^{5-i}$, $(s^{-i})^i = s^0$. For every such cycle denote $Orbit(s) = \{s^1, s^2, s^3, s^4, s^5\}$ and for $S \subset \mathcal{S}$ we put $Orbit(S) = \bigcup_{s \in S} Orbit(s)$.

Let $\mathcal{R}$ be a set with only N elements. Each element is indexed by cycles from π; i.e. if $s \in \mathcal{S}$, then $r_{(s^1,s^2,s^3,s^4,s^5)} \in \mathcal{R}$. By this construction we get from the set of $5N$ elements a set of N elements : all 2-element subsets $e = (s_1, s_2)$ of S are then 2-element subsets $e' = (r_{Orbit(s_1)}, r_{Orbit(s_2)})$ of $\mathcal{R}$, if $Orbit(s_1) \cap Orbit(s_2) = \emptyset$.

If V is a set created by selecting one element from each cycle of π we have $|V| = N$, $Orbit(V) = \mathcal{S}$ and the number of all such V's is 5^N. Now put $V^i = \{s^i, s \in V\}$. If $i \in \{1, 2, 3\}$ and $e \in V^i$ ($e = (s_1^i, s_2^i)$), then $Orbit(e) = Orbit(s_1) \cup Orbit(s_2)$ has exactly 2 cycles of length 5 and in this case we will call e a $cross - edge$. If $e = \{s^4, s^5\}$ for some $s \in V$, then $Orbit(e) = Orbit(s)$ have only one cycle and we will call such e an $inner-edge$.

To prove the main Theorem 4.16 of this section we first prove lemma in which we construct a design of degree $2d + 1$ on $5N$ elements from design of degree d on N elements.

Lemma 4.13 (Buss et al. [5])**.**
Let N, p and q be as above. If there is a design of degree d on N elements, then there is a design of degree $2d + 1$ on $5N$ elements over the fixed field Z_q.

Proof. Let D be a degree d design on $\mathcal{R}$. If some subset R of $\mathcal{R}$ has more than d elements (2-element cycles) we can w.l.o.g from definition of design put $D(R) = 0$. We begin our construction by creating a design D^V of degree d on $\mathcal{S}$ for the set V defined as before.

First we define a partial mapping $V(S)$ from partial 2-partition on $\mathcal{S}$ to partial 2-partition on $\mathcal{R}$; if for all e in $S \subset \mathcal{S}$ holds that they are either cross-edges ($Orbit(e)$ form 2 cycles in $\mathcal{S}$ and then e' in $\mathcal{R}$ is indexed by these cycles) or inner-edges ($Orbit(e)$ form one cycle in $\mathcal{S}$, then e' in $\mathcal{R}$ has only one element) and the set $\{Orbit(e),\ e \in S\}$ is a partial 2-partition in $\mathcal{R}$ we put

$V(S) = \{Orbit(e),\, e \in S\}$. Only in these cases is $V(S)$ defined.

Now let

$$D^V = \begin{cases} D(V(S)), & \text{if } D(V) \text{ is defined} \\ \\ 0, & \text{otherwise} \end{cases}$$

It is easy to see that $D(V)$ is a degree d design on $\mathcal{S}$ (it is defined only on sets S with less than d 2-element subsets).
Now define

$$D^+(S) = (5^N)^{-1} \cdot \sum_{V \in \mathcal{S}} D^V(S).$$

We are working in field Z_3 so the inverse of 5^N exists and is equal to 2 in Z_3. ($5^{2^n-1} \cdot x \equiv 1 \ (mod\ 3) \Rightarrow 5^{2^n-1} \equiv 2^{2^n} \cdot 2^{-1} \equiv 1 \cdot 2 \equiv 2 \ (mod\ 3)$ and $2 \cdot x \equiv 1 \ (mod\ 3) \Rightarrow x \equiv 2 \ (mod\ 3)$)

By discussing all possibilities which might happen it will be proven that D^+ is degree $2d+1$ design on $\mathcal{S}$.
The condition for the empty set is satisfied:

$$\begin{aligned} D^+(\emptyset) &= (5^{2^n-1})^{-1} \cdot \sum_V D^V(\emptyset) \\ &= (5^{2^n-1})^{-1} \cdot (\#V) \cdot 1 \\ &= (5^{2^n-1})^{-1} \cdot 5^{2^n-1} \\ &= 1 \end{aligned}$$

where $\#V$ is defined as the number of possible $V's$.

Now consider any partial 2-partition S of $\mathcal{S}$ and any element v not in any member of S. We can choose some mapping $V(S)$ (w.l.o.g. is defined at least for one V).

If $|V(S)| < d$, then the same is true for all $W's$ for which $W(S)$ is also defined. The size of the image of such mapping depends only on the number of cross-edges in S.

If v is in a block (cycle of π) which intersects some cross-edge in S; i.e. $v \in e$ for some e cross-edge, then the design condition is satisfied for $D^V(S)$ and also for $D^+(S)$ - is only linear combination of designs satisfying the design condition.

If the block of v intersects some inner-edge in S and hence contains it three cases may occur:

1, v is also in block which intersects some cross-edge in S. In this case is the design condition satisfied by previous observations.

2, The image $|V(S)| < d$ or is undefined and in this case is the design condition again satisfied.

3, In S is an edge which satisfied the condition of following lemma

Lemma 4.14 (Buss et al. [5])**.**
Suppose there is a cross-edge e in S such that no other edge in S intersects any of the blocks that e intersects. Then $D^+(S) = 0$.

Proof. By definition of a cross-edge, there are only 3 possibilities how to make a cross-edge for given nodes v, v' ($\in V^i$, $i = 1, 2, 3$).
Hence, for each V such that $V(S)$ is defined, there are exactly two other sets W_1, W_2 of elements agreeing on the blocks which e does not intersect and such that $W_1(S)$ and $W_2(S)$ are defined. By symmetry we have:
$D^V(S) = D^{W_1}(S) = D^{W_2}(S)$ (they agreeing on images of cross-edges).
Then

$$D^+(S) = 2 \cdot \sum_V D^V(S) = 2 \cdot 3 \cdot D^V(S) \equiv 0 \ (mod\ 3).$$

□

If cases 1, and 2, are false, then S has at least d cross-edges with non-intersecting blocks. Blocks of each other edge intersect with at most one of the d cross-edges (otherwise $V(S)$ is not a partition) and since S has at most $2d$ edges and 1, is false, there must be some cross-edge satisfying the assumption in Lemma 4.14 and the design condition is satisfied.

It remains to consider the case when the block of v does not intersect any edge in S and $|V(S)| = d$ for each V.
Let W range over all selection of elements from blocks not containing v and let $W_i = W \cup v^i$. Because $v \notin S$ we have $D^W(S) = D^{W_i}(S)$ for all $i = 1, \ldots, 5$.
Then

$$D^+(S) = 2 \cdot \sum_W \sum_{i=1}^{5} D^{W_i}(S) = 2 \cdot 5 \cdot \sum_W D^W(S) \equiv \sum_W D^W(S) \ (mod\ 3)$$

Define $e_j = \{v^{-j}, v^{1-j}\}$ for $j = 0, 1$. The edges e_j do not intersect any edge in S ($v \notin S$) and by assumption on the design D these two edges are the only edges for which $D^{W_i}(S \cup e)$ might be non-zero (otherwise $|W_i(S \cup e)| > d$ or is undefined). Moreover, if $i = -4 - j$, $j = 0, 1$, then e_j is an inner-edge of W_i and $D^{W_i}(S \cup e_j) = D^{W_i}(S) = D^W(S)$, otherwise it is zero.

Thus,

$$\begin{aligned}\sum_{j=0}^{1} D^+(S \cup e_j) &= 2 \cdot \sum_W \sum_{i=1}^{5} \sum_{j=0}^{1} D^{W_i}(S \cup e_j) \\ &= 2 \cdot 2 \cdot \sum_W D^W(S) \equiv \sum_W D^W(S) \ (mod\ 3)\end{aligned}$$

which is equal to $D^+(S)$. □

Theorem 4.15 (Lower bound for degree of design, Buss et al. [5])**.**
Let $p = 2$, $q = 3$ and the field Z_3 be as above.
For any $N > 0$ such that $N \not\equiv 0$ $(mod\ 2)$ there is a degree $d = \Omega(N^{1/log_2 5})$ design on set with N elements ($[N]$) over the field Z_3.

Proof.
We prove this in the case of $N = 2^n - 1$, $n \geq 1$.
If $n = 1 \Rightarrow N = 1 < p = 2 \Rightarrow$ w.l.o.g there is a degree 0 design (in the definition of design on sets we require $N > p$).
In other cases where $N > p$ we prove the theorem by induction on d and n (N).

By previous lemma if there is a degree d design on $[N]$, then there is also a degree $2d + 1$ design on $[5N]$. Since

$$\begin{aligned}d &\geq k \cdot N^{1/log_2(5)} \text{ for } k > 0 \text{ a constant and} \\ 2d + 1 > 2d &\geq k \cdot (5N)^{1/log_2(5)} = k \cdot 2 \cdot N^{1/log_2(5)} \qquad (4.6)\end{aligned}$$

dividing both sides of inequality (4.6) by 2 be obtain the same inequality with the same constant $k > 0$.

This holds for N of the special form $(p + q)^i = 5^i$. Also the value of d depends on this form and we can see that it increases doubly:

$$\begin{aligned}N &= 5^i \\ N^{1/log_2 5} &= 5^{i/log_2 5} = (2^{log_2 5})^{i/log_2 5} = 2^i \\ N^{1/log_2 5} &= 2^i = d \text{ for } N = 5^i.\end{aligned}$$

The problem with gaps between $[N]$ and $[5N]$ is solved by restricting the design (is a mapping). For example, if we have set with $N + 1$ elements we use the design D^+ for $[5N]$ and restrict this only on $[N + 1]$. So, we obtain a design $D^+|_{N+1}$ with degree less than $2d + 1$. □

Theorem 4.16 (Lower bound for degree of NR, Buss et al. [5])**.**
The $(N, p)-$polynomial system for $N = 2^n - 1$ and $p = 2$ over the field Z_3 has a Nullstellensatz refutation of degree at least $d = \Omega(N^{1/log_2 5})$.

Proof. The theorem follows from Theorem 4.12 and Theorem 4.15 applying on the $(2^n - 1, 2)$-polynomial system over Z_3. □

At the end of this chapter we formulate and prove the theorem about the $(M, q)-$polynomial system over Z_q, for $q = 3$ and $M \not\equiv 0 \pmod 3$.

Theorem 4.17 (Constant-degree bound of NR)**.**
The $(M, q)-$polynomial system for $q = 3$ over the field Z_3 has a Nullstellensatz refutation of constant-degree.

Proof. Following the Definition 3.7, the $(M, q)-$polynomial system is the system of polynomial equations in variables y_e where e ranges over all q-element subsets of the set $[M]$:

$$\begin{aligned} Q_v: \quad \sum_{v \in e} y_e &= 1, \text{ one for each } v \in [M] \\ Q_{e,f}: \quad y_e \cdot y_f &= 0, \text{ for all e, f such that } e \perp f, \\ &\quad \text{i.e. } e \neq f \text{ and } e \cap f \neq \emptyset \end{aligned}$$

We have

$$\begin{aligned} \sum_{v \in [M]} \sum_{e \ni v} y_e &= \sum_{e \ni v} \sum_{v \in [M]} y_e \\ 1 \equiv \sum_{v \in [M]} 1 &= \sum_{e \ni v} 3 \cdot y_e \equiv 0 \pmod 3 \qquad \text{which is a contradiction.} \end{aligned}$$

Hence, the $(M, 3)-$polynomial system over the field Z_3 has a NR from the system of polynomial equations $\{Q_v | v \in [M]\}$ of constant-degree. □

5. A separation result

By reformulating the Corollary 4.9 in a more specific way we obtain the following

Theorem 5.1.

$$Count_2^N \not\leq_T Count_3^M.$$

Proof. It directly follows from Theorems 4.8, 4.16 and 4.17.
In particular, in Theorem 4.16 we proved the lower bound for degree of Nullstellensatz refutation in the case of $p = 2$, $N = 2^n - 1$ and $q = 3$; i.e. for the $(2^n - 1, 2)$-polynomial system over the field Z_3 and the degree was about $2^{n/log_2(5)}$.
In Theorem 4.17 we proved that the $(M, 3)$-polynomial system has a Nullstellensatz refutation over the field Z_3 of constant-degree.
Hence, the Nullstellensatz proof that $Count_3^M$ over Z_3 is total is of constant-degree and the Nullstellensatz proof that $Count_2^N$ is total requires degree $2^{n^{O(1)}}$ and we use the Corollary 4.10. □

The technical background used to obtain this reduction can be also used in the general case, for p, q distinct primes. Then the lower bound for degree of Nullstellensatz refutation for system $\mathcal{F}_\Phi$ where $Q_\Phi = \text{Count}_p^N$ over the field Z_q is $\Omega(N^{1/log_p(p+q)})$ (Buss et al.[5]).

We can conclude this thesis with some more observations about the structure of the class TFNP and its subclasses (they were described in Chapter 2). The class PPA is based on total and complete problem Lonely. The problem Lonely can be formulated also in the algebraic formalism as Count_2^N problem (see Section 3.2). All Count_p^N problems, for all p primes, can build one subclass in TFNP or we can consider various subclasses based on Mod-p counting principle for every specific prime p.

Let us consider that the problem 3-Lonely (described in Chapter 2) based on Mod-3 counting principle defines a subclass **P3A:= $\mathbf{C}_m$(3-Lonely)** of TFNP. Then the Theorem 5.1 and Definition 1.4 clearly imply the following fact:

$$PPA \not\subseteq P3A$$

in relativized world.

In Section 3.1 we briefly introduced the algebraic proof systems and the connection between equational reasoning and automated theorem proving. If we use the representation of the type-2 search problems Lonely and 3-Lonely in first-order logic (as an $\exists-$sentence in some basic language), we can use a suitable automated theorem prover to conclude some observations about relations between the TFNP subclasses in relativized world. For example,

from the output file of some automated theorem prover one can extract the substitutions which were used to obtain the reductions.
Hence, it would be interesting (maybe my future work) to look at the search problems in terms of first-order logic and prove some reductions among them automatically.

Bibliography

[1] P. Beame, S. Cook, J. Edmonds, R. Impagliazzo and T. Pitassi, *The relative complexity of NP search problems*, Journal of Computer and System Science, 57 (1998), pp. 3-19.

[2] P. Beame, R. Impagliazzo, J. Krajíček, T. Pitassi and P.Pudlák, *Lower bound on Hilbert's Nullstellensatz and propositional proofs*, Proceedings of the London Mathematical Society, 73 (1996), pp. 1-26.

[3] J. Buresh-Oppenheim and T. Morioka, *Relativized NP search problems and propositional proof systems*, in Proc. 19th IEEE Conference on Computational Complexity (CCC), (2004), pp. 54-67.

[4] S. R. Buss, *Lower bounds on Nullstellensatz proofs via designs*, in Proof Complexity and Feasible Arithmetics, P. Beame and S. Buss, eds., American Mathematical Society, (1998), pp. 59-71.

[5] S. R. Buss, R. Impagliazzo, J. Krajíček, P. Pudlák, A. A. Razborov and J. Sgall, *Proof complexity in algebraic systems and bounded depth Frege systems with Modular counting*, Computational Complexity, 6 (1996/1997), pp. 256-298.

[6] S. R. Buss and A. S. Johnson, *Propositional proofs and reductions between NP search problems*, Annals of Pure and Applied Logic, (to appear), pp. 1-36.

[7] D. S. Johnson, Ch. H. Papadimitriou and M. Yannakakis, *How easy is local search?*, Journal of computer and system science, 37 (1988), pp. 79-100.

[8] G. Kemper, *Hilbert's Nullstellensatz*, A Course in Commutative Algebra, 256 (2011), pp. 7-21.

[9] J. Krajíček, *Structured pigeonhole principle,search problems and hard tautologies*, Journal of Symbolic Logic, 70 (2005), pp. 619-630.

[10] J. Matoušek and J. Nešetřil, *Kapitoly z diskrétní matematiky*, (2007), p. 130, Praha Karolinum.

[11] Ch. H. Papadimitriou, *On the complexity of the parity argument and other inefficient proofs of existence*, Journal of computer and system science, 48 (1994), pp. 498-532.