The Isomorphism Problem for One-Time-Only Branching Programs

Thomas Thierauf [*]

Abt. Theoretische Informatik
Universität Ulm
89069 Ulm, Germany

thierauf@informatik.uni-ulm.de

May 21, 1996

Abstract

We investigate the computational complexity of the *isomorphism problem for one-time-only branching programs (BP1-Iso)*: on input of two one-time-only branching programs B_0 and B_1, decide whether there exists a permutation of the variables of B_1 such that it becomes equivalent to B_0.

Our main result is a two-round interactive proof for $\overline{\text{BP1-Iso}}$, the complement of BP1-Iso. The protocol is based on the Schwartz-Zippel Theorem to probabilistically check polynomial idendities. As a consequence, BP1-Iso cannot be NP hard unless the polynomial hierarchy collapses.

We extend the protocol to get an interactive proof to decide the non-isomorphism of multivariate polynomials over an arbitrary field.

Finally, we show that BP1-Iso has a zero-knowledge interactive proof.

1 Introduction

An interesting computational issue is to decide the *equivalence* of two given programs with respect to some computational model as, for example, Boolean circuits, branching programs, or Boolean formulas. A more general problem is to decide the *isomorphism* of two given, say circuits. That is, whether the circuits become equivalent after permuting the input variables of one of the circuits.

The isomorphism problem for Boolean formulas is in fact a very old one dating back to the last century (see [BRS95] for background and early references on this problem). More recently, the problem has been reconsidered in a series of papers with respect to its computational complexity [AT96, BR93, BRS95, CK91].

The equivalence problems for Boolean circuits, branching programs, and Boolean formulas are known to be coNP complete. Asking for isomorphism, puts, roughly speaking, an existential quantifier in front of the problem. Therefore, the corresponding isomorphism problems are in the second level of the polynomial hierarchy, Σ_2^p. An obvious question is whether these isomorphism problems are complete for Σ_2^p. This question was solved by Agrawal and Thierauf [AT96] who showed that *none* of these isomorphism problems is complete for Σ_2^p, unless the polynomial

[*]Supported in part by DAAD, Acciones Integradas 1995, 322-AI-e-dr.

hierarchy collapses. Thus, loosely speaking, the existential quantifier we get by seeking for an isomorphism doesn't seem to add full NP power to the equivalence problem.

There are more examples for the latter observation: the equivalence problem for graphs, which is in fact an equality problem, is trivially solvable in polynomial time. The *graph isomorphism* problem is therefore in NP, but is *not* NP complete, unless the polynomial hierarchy collapses [BHZ87] (see also [Sch89], (comprehensive studies on the graph ismorphism problem can be found in [Hof82, KST93]). The equivalence of two deterministic finite automatas (DFA) can be decided in polynomial time. It is not hard to see that the isomorphism problem for DFA's, where one can permute the states of a DFA, is still solvable in polynomial time.

For a subclass of the branching programs, the *one-time-only branching programs*, where, on each path, each variable is tested at most once (see next section for precise definitions), the equivalence problem is easier then for general ones: it can be efficiently solved by a randomized Las Vegas type algorithm [BCW80], it is in the complexity class coRP. Therefore, putting an existential quantifier in front, the corresponding isomorphism problem is in NP$\cdot$coRP. Motivated by the examples above, we ask for some better bound on its complexity. In this paper we show that the isomorphism problem for one-time-only branching programs, BP1-Iso, is also in the class BP$\cdot$coNP. As a consequence, it is *not* NP hard, unless the polynomial hierarchy collapses.

Our proof is based on a two-round interactive proof for complement of the BP1-Iso. In fact, we solve a more general problem: our interactive proof can decide the non-isomorphism problem for multivariate polynomials over an arbitrary field.

Using these ideas, we also obtain a zero-knowledge interactive proof for BP1-Iso.

2 Preliminaries

We first define some general notions used in the paper. In Section 2.2, we provide definitions and some background on branching programs.

2.1 Basic Definitions

We will use fairly standard notions of complexity theory. We refer the reader to [BDG88, BDG91, HU79] for definitions of complexity classes like P, NP, RP or BPP. Σ_k^p are the levels of the polynomial hierarchy. For any class $\mathcal{C}$, we denote the complement class by co $\mathcal{C}$.

BP $\cdot$ NP is the class of sets L such that there exists a set $A \in$ NP and a polynomial p such that for every x

$$\begin{aligned} x \in L &\implies \mathbf{Pr}[(x,y) \in A] = 1, \\ x \notin L &\implies \mathbf{Pr}[(x,y) \in A] \leq 1/2, \end{aligned}$$

where y is chosen uniformly at random from $\Sigma^{p(|x|)}$.

NP $\cdot$ coRP is the class of sets L such that there exists a set $A \in$ coRP and a polynomial p such that for every x, we have

$$x \in L \iff \exists y \in \Sigma^{p(|x|)} : (x,y) \in A.$$

Interactive proofs were defined in [GMR89]. An *interactive proof system* for a set L consists of a prover P and verifier V. The verifier is a randomized polynomial-time algorithm that can communicate with the prover. The prover can make arbitrary computations. After following some communication protocol, the verifier finally has to accept or reject a given input such that

$$\begin{aligned} x \in L &\implies \exists \text{ prover } P : \mathbf{Pr}[(V,P)(x) \text{ accepts}] = 1, \\ x \notin L &\implies \forall \text{ prover } P : \mathbf{Pr}[(V,P)(x) \text{ accepts}] \leq 1/2, \end{aligned}$$

where the probability is taken over the random choices of the verifier.

IP denotes the class of sets that have an interactive proof system. IP[k] is the subclass of IP where the verifier and the prover exchange at most k messages.

Arthur-Merlin games were introduced in [Bab85]. They are similar to interactive proofs with Arthur corresponding to the verifier and Merlin to the prover. The only difference is that Arthur has to make his random bits available to Merlin, while the prover doesn't know the random bits of the verifier. The classes AM and AM[k] are defined analogously.

In this paper, we are interested in constant round interactive proof systems. It is known that for both, interactive proof systems and Arthur-Merlin games, constantly many rounds are the same as one round: for any $k \geq 2$, IP[k] = AM[k] = AM[2] [Bab85, GS89]. Moreover, it is easy to see that AM[2] = BP $\cdot$ NP.

2.2 Branching Programs

A *branching program* B in n Boolean variables $x_1, \ldots, x_n$ is a directed acyclic graph with the following type of nodes. There is a single node of indegree zero, the *initial node of* B. All nodes have outdegree two or zero. A node with outdegree two is an *internal node of* B. One of its edges is labelled with x_i, the other with $\overline{x}_i$, for some $i \in \{1, \ldots, n\}$. A node with outdegree zero is a *final node of* B. The final nodes are labelled either by *accept* or *reject*.

We call a branching program *one-time-only*, if, on each path from the initial node to a final node, every variable or its complement occurs at most once as an edge label.

A one-time-only branching program is called *ordered*, if the order of occurance of the variables on each path is consistent with some ordering on the set of variables.

Branching programs are also called *binary decision diagrams (BDD)*. Correspondingly, a one-time-only branching program and an ordered branching program is also called *free binary decision diagram (FBDD)* and *ordered binary decision diagram (OBDD)*, respectively.

A branching program B defines an *n-ary Boolean function* from $\{0,1\}^n$ to $\{0,1\}$ as follows. For an *assignment* $\mathbf{a} = (a_1, \ldots, a_n) \in \{0,1\}^n$, we walk through B, starting at the initial node, always following the (unique) edge that evaluates to one under $\mathbf{a}$, until we reach a final node. If the final node is an accepting node, we define $B(\mathbf{a}) = 1$, and $B(\mathbf{a}) = 0$ otherwise.

Two branching programs B and B' in n variables are *equivalent*, $B \equiv B'$ for short, if they define the same Boolean function. By BP-Equ we denote the problem to decide whether two given branching programs are equivalent. That is,

$$\text{BP-Equ} = \{\, (B, B') \mid B \equiv B' \,\}.$$

It is not hard to see that branching programs can compute CNF Boolean formulas. Therefore, the satisfiability problem for branching programs is NP complete, and hence, BP-Equ is coNP complete.

The equivalence problem for one-time-only branching programs, BP1-Equ, is easier because one can *arithmetize* them. That is, one can convert them into an algebraic formula. Note first that a branching program B can be viewed as a compact way of denoting a DNF formula F_B: each path of B can be written as a monom, the conjunction of the literals occuring along that path. Then the function computed by B is simply the disjunction of all monoms coming from accepting paths of B.

We convert F_B into a polynomial p_B over the rational numbers $\mathbf{Q}$ as follows. A variable x_i is kept as x_i. A negated variable $\overline{x}_i$ is replaced by $1 - x_i$. A conjunction is replaced by multiplication and a disjunction is replaced by addition. For each assignment $\mathbf{a} \in \{0,1\}^n$, exactly one path of

B evaluates to true under $\mathbf{a}$. Therefore, at most one product term in p_B will be one on input $\mathbf{a}$. Hence, B and p_B agree on $\{0,1\}^n$. That is,

$$B(\mathbf{a}) = p_B(\mathbf{a}) \quad \text{for all } \mathbf{a} \in \{0,1\}^n.$$

Note that p_B can consist of exponentially many terms even though B has small size. So in general, one cannot write down p_B in polynomial time in $|B|$. However, one can anyway evaluate p_B at a given point in $\mathbf{Q}^n$ in polynomial time by using B.

Since B is one-time-only, p_B is a *multilinear* polynomial. The following theorem is a special case of the Schwartz-Zippel Theorem [S80, Zip79] for multilinear polynomials. It bounds the number of points where two such polynomials can have the same value without being the same function.

Theorem 2.1 *Let $p(x_1, \ldots, x_n)$ be a multilinear polynomial over $\mathbf{Q}$ that is not the zero-polynomial. Let $T \subseteq \mathbf{Q}$ with $|T| > 1$. Then there are at least $(|T|-1)^n$ points $(x_1, \ldots, x_n) \in T$ such that $p(x_1, \ldots, x_n) \neq 0$.*

Setting $T = \{0,1\}$, we get that if two one-time-only branching programs B and B' are equivalent, then the corresponding polynomials p_B and $p_{B'}$ are the same function on $\{0,1\}^n$, and hence, by Theorem 2.1, on $\mathbf{Q}$. However, the equivalence of the polynomials can be tested by a randomized algorithm that simply picks a random point $\mathbf{r} \in \{1, \ldots, 2n\}^n$ and checks whether $p_B(\mathbf{r}) = p_{B'}(\mathbf{r})$. Again by Theorem 2.1 (applied to $p = p_B - p_{B'}$), if p_B and $p_{B'}$ are not equivalent, then they will disagree on a fraction of at least $((2n-1)/2n)^n > 1/2$ of all points in $\{1, \ldots, 2n\}^n$. Thus, the algorithm will detect an inequivalence with probability more than $1/2$. It follows that BP1-Equ $\in$ coRP.

The equivalence problem for ordered branching programs is solvable in polynomial time [MS94, SW92, Bry86]. There are two points to note for this. Let B and B' be ordered branching programs. First, one can transform, say B', in an equivalent ordered branching program that has the same ordering on the variables as B [MS94]. Second, for a fixed ordering, an ordered branching program can be considered as a (nonuniform) finite automaton [GG95]. Hence, there is a unique *minimal* ordered branching program for any Boolean function [Bry86], that is computable even in linear time [SW92]. Thus, via this process, we get a *normal form* for ordered branching programs for any given ordering that can now be used to decide equivalence.

Two branching programs B and B' are *isomorphic*, denoted by $B \cong B'$, if there exists a permutation φ on $\{x_1, \ldots, x_n\}$, such that B becomes equivalent to B' when permuting the variables of B' according to φ. That is $B \equiv B' \circ \varphi$. In this case, we call φ an *isomorphism between B and B'*.

The *isomorphism problem for branching programs* is BP-Iso $= \{ (B, B') \mid B \cong B' \}$. The isomorphism problem for one-time-only branching programs, BP1-Iso, is defined analogously. It follows directly from the definition that BP-Iso $\in \Sigma_2^p$, the second level of the polynomial hierarchy. Agrawal and Thierauf [AT96] showed that $\overline{\text{BP-Iso}}$, the complement of BP-Iso, is in BP $\cdot\, \Sigma_2^p$. By a result of Schöning [Sch89], it follows that BP-Iso *cannot* be complete for Σ_2^p, unless the polynomial hierarchy collapses to its third level, Σ_3^p.

For one-time-only branching programs, we have BP1-Iso $\in$ NP$\cdot$coRP. In this paper, we show that $\overline{\text{BP1-Iso}}$, the complement of BP1-Iso, is in BP $\cdot$ NP. By the result of Boppana, Hastad, and Zachos [BHZ87] (see also Schöning [Sch89]), it follows that BP1-Iso *cannot* be hard for NP, unless the polynomial hierarchy collapses to its second level, Σ_2^p.

This result covers also the case of ordered branching programs. Note however that here, the isomorphism problem is still in NP.

3 An Interactive Proof for the Complement of BP1-Iso

We show that there is a two round interactive proof for the complement of the one-time-only branching program isomorphism problem, $\overline{\text{BP1-Iso}}$.

We start by recalling the idea of the interactive proof for the complement of the graph isomorphism problem [GMR89] (see also [Sch88]). There, on input of two graphs G_0 and G_1, the verifier randomly picks $i \in \{0, 1\}$ and a permutation φ, and sends $H = \varphi(G_i)$ to the prover. Now the prover is asked to find out what the value of i is. The verifier will accept only if the prover gives the right answer.

When the input graphs are *not* isomorphic, then the Prover can find out easily i. However, when the graphs are isomorphic, both could be used by the verifier to compute H, so that no prover can find i. Therefore, the answer of any prover is correct with probability at most 1/2.

First of all note that we cannot directly adapt this protocol to branching programs. The reason for this is that simply the syntactic structure might tell the prover which of two given branching programs was selected by the verifier.

A way out of this would be a normal form for one-time-only branching programs, that is easy to compute. However, such a normal form is not known. At this point, Agrawal and Thierauf [AT96] used a result from learning theorie by Bshouty *et. al.* [BCGKT95]: there is a randomized algorithm that uses an NP oracle and outputs branching programs equivalent to a given one. The important point is that although the algorithm might output (syntactically) different branching programs depending on its random choices, the output does *not* depend on the syntactic structure of its input. However, in our case, the verifier doesn't have an NP oracle available and there is no analog learning result for one-time-only branching programs without an NP oracle.

The idea to get around this problem is as follows. On input of two given one-time-only branching programs B_0 and B_1 with n variables, the verifier randomly chooses one of them and permutes it with a random permutation to obtain a branching program B. Instead of trying to manipulate whole B, *the verifier evaluates B at a randomly choosen point* $\mathbf{r} \in T^n$, where T is some appropriate test domain. The prover is now asked to tell which of B_0, B_1 was used to obtain the point $(\mathbf{r}, B(\mathbf{r}))$. If B_0 and B_1 are isomorphic, then the prover cannot detect and has to guess. So she will fail with probability 1/2. On the other hand, If B_0 and B_1 are *not* isomorphic, then the prover has a good chance of detecting the origin of $(\mathbf{r}, B(\mathbf{r}))$. This is because, by Theorem 2.1, different multilinear polynomials can agree only on a fraction $(|T|^n - (|T| - 1)^n))/|T|^n \leq n/|T|$ of domain T. By choosing T large enough, the origin of $(\mathbf{r}, B(\mathbf{r}))$ is unique with high probability. With an additional round of communication the prover can *always* convince the verifier from the *non-isomorphism* of B_0 and B_1. We give the details below.

Theorem 3.1 $\overline{\text{BP1-Iso}} \in \text{IP}[4]$.

Proof. The following IP-protocol accepts $\overline{\text{BP1-Iso}}$. The input are two one-time-only branching programs B_0 and B_1, both over n variables $x_1, \ldots, x_n$. Let $T = \{1, \ldots, 2n\}$.

V: the verifier randomly picks $i \in \{0,1\}$, a permutation φ, and points $\mathbf{r}_1, \dots, \mathbf{r}_k \in T^n$, where $k = \lceil n \log n \rceil + 2$. Then the verifier permutes the variables of B_i according to φ, computes $y_l = p_{B_i} \circ \varphi(\mathbf{r}_l)$, for $l = 1, \dots, k$, and sends the set of pairs $R = \{ (\mathbf{r}_l, y_l) \mid l = 1, \dots, k \}$ to the prover.

P: the prover is expected to send $j \in \{0,1\}$ and a permutation φ' to the verifier.

V: if $i = j$, then the verifier accepts. If $i \neq j$, the verifier checks whether $p_{B_j} \circ \varphi'$ *matches the set* R, that is, whether $p_{B_j} \circ \varphi'(\mathbf{r}_l) = y_l$, for $l = 1, \dots, k$. If the test fails, the verifier rejects. Otherwise, he sends φ to the prover.

P: the prover is expected to send a point $\mathbf{r}' \in T^n$ to the verifier.

V: finally, the verifier accepts if and only if $p_{B_i} \circ \varphi(\mathbf{r}') \neq p_{B_j} \circ \varphi'(\mathbf{r}')$.

We show that the above protocol works correctly.

Case 1: $B_0 \not\cong B_1$. We show that there is a prover such that the verifier always accepts.

The prover can cycle through all permutations and check for both, p_{B_0} and p_{B_1}, whether it matches with the set R sent by the verifier. Say that polynomial $p_{B_0} \circ \varphi'$ does so. Then the prover sends $j = 0$ and φ' to the verifier.

If *no* permutation of polynomial p_{B_1} matches R as well, then i must have been 0 and therefore the verifier will accept in the first round.

On the other hand, if some permutation of polynomial p_{B_1} matches R, then the prover cannot tell which one was used by the verifier. If the prover is lucky, i has anyway been zero and the verifier accepts. On the other hand, if $i \neq j$, then the verifier will send φ to the prover because $p_{B_j} \circ \varphi'$ matches R. Since $p_{B_j} \circ \varphi' \neq p_{B_i} \circ \varphi$, these polynomials can agree on at most $n|T|^{n-1}$ points on T^n by Theorem 2.1. Note that $n|T|^{n-1} < |T|^n$. Therefore, the prover can find a point $\mathbf{r}' \in T^n$ such that $p_{B_j} \circ \varphi'(\mathbf{r}') \neq p_{B_i} \circ \varphi(\mathbf{r}')$, and send it to the verifier who will accept. In summary, the verifier accepts with probability one.

Case 2: $B_0 \cong B_1$. We show that for any prover, the verifier accepts with probability at most $3/4$. By executing the protocol several times in parallel, the acceptance probability can be made exponentially small.

The prover will always find permutations of p_{B_0} and p_{B_1} that match the set R sent by the verifier. Therefore, with respect to the test $i = j$ made by the verifier, the best the prover can do is to guess j randomly. This will make the verifier accept with probability $1/2$. However, the prover can improve her chances by the condition checked in the second round by the verifier: fix i and φ chosen by the verifier, say $i = 0$. Then there might exist a permutation φ' such that $p_{B_1} \circ \varphi'$ matches R, but $p_{B_0} \circ \varphi \neq p_{B_1} \circ \varphi'$. Now the prover can choose a point $\mathbf{r}'$ such that $p_{B_0} \circ \varphi(\mathbf{r}') \neq p_{B_1} \circ \varphi'(\mathbf{r}')$, and make the verifier accept by sending $j = 1$, φ', and $\mathbf{r}'$. We will give an upper bound on the probability of this event.

By Theorem 2.1, for any φ' such that $p_{B_0} \circ \varphi \neq p_{B_1} \circ \varphi'$ we have

$$\mathbf{Pr}[p_{B_0} \circ \varphi(\mathbf{r}) = p_{B_1} \circ \varphi'(\mathbf{r})] \;\leq\; \frac{n}{|T|} \;=\; \frac{1}{2}, \tag{1}$$

for a randomly chosen $\mathbf{r} \in T^n$. Since points $\mathbf{r}_1, \ldots, \mathbf{r}_k \in T^n$ are chosen independently and uniformly at random from T^n, we have

$$\mathbf{Pr}[p_{B_1} \circ \varphi' \text{ matches } R] \;\leq\; 2^{-k},$$

Therefore, considering all such φ', we get that

$$\mathbf{Pr}[\exists \varphi' \; (p_{B_0} \circ \varphi \neq p_{B_1} \circ \varphi' \text{ and } p_{B_1} \circ \varphi' \text{ matches } R] \;\leq\; n!\, 2^{-k} \;\leq\; \frac{1}{4}$$

by our choice of k. We conclude that the probability that any of the conditions tested by the verifier is satisfied is bounded by $1/2 + 1/4 = 3/4$. That is, the verifier accepts with probability at most $3/4$, irrespective of the prover. □

We can directly come down to a one round interactive proof by choosing T large, for example $T = \{1, \ldots, 2nn!\}$. Then, in case $B_0 \not\cong B_1$, the prover can always find point $\mathbf{r}'$ without knowing φ, and hence can already send it in the first round to the verifier, who can then make all his tests. However, then we get another difficulty: when T has exponential size, the values of the poynomials might be up to double exponential. Then the polynomial time verifier can no longer deal with such numbers. We will show in the next section how the verifier can still manage his task.

As already mentioned in Section 2.1, the class of sets that can be decided by a constant round interactive proof system coincides with the Arthur-Merlin class AM[2] which, in turn, is the same as BP · NP [Bab85, GS89].

Corollary 3.2 $\overline{\text{BP1-Iso}} \in \text{BP} \cdot \text{NP}$.

Schöning [Sch88] gives a direct proof that the graph isomorphism problem is in AM by using hash functions. By combining the technique from Theorem 3.1 and that in [Sch88], we can as well obtain Corollary 3.2 directly.

Note that both classes, BP · NP and NP · coRP can be loosely considered as some, more or less, slight extensions of NP. In this sense, we have shown that BP1-Iso is in a slight extension of NP ∩ coNP.

Corollary 3.3 $\text{BP1-Iso} \in \text{NP} \cdot \text{coRP} \cap \text{BP} \cdot \text{coNP}$.

Boppana, Hastad, and Zachos [BHZ87] (see also Schöning [Sch89]) have shown that a coNP complete set cannot be in BP · NP unless the polynomial hierarchy collapses to the second level, in fact, even to BP · NP. Hence we get the main result of this section.

Corollary 3.4 *If* BP1-Iso *is* NP *hard then* $\text{PH} = \Sigma_2^p$.

4 Extension to Multivariate Polynomials Over an Arbitrary Field

The interactive proof in the previous section decided on the isomorphism of two one-time-only branching programs. However, the important point to make the protocol work was that these branching programs in fact represent multilinear polynomials over an infinite field, say $\mathbf{Q}$: the branching programs are just a compact way of describing them that are easy to evaluate at a given point by the verifier.

We first show how to extend the above protocol to an interactive proof to decide the isomorphism of two multivariate polynomials of higher degree over an *infinite* field, say $\mathbf{Q}$. For this we need the general version of the Schwartz-Zippel Theorem. For a multivariate polynomial $p(x_1, \ldots, x_n)$, the degree of any term is the sum of the exponents of the variables, and the *total degree* of p is the maximum of the degrees of its terms.

Theorem 4.1 **[S80, Zip79]** *Let $p(x_1, \ldots, x_n)$ be a polynomial over some field $\mathbf{F}$ of total degree d that is not the zero-polynomial. Let $T \subseteq \mathbf{F}$ be a finite set, and let $r_1, \ldots, r_n$ be chosen independently and uniformly at random from T. Then*

$$\mathbf{Pr}[p(r_1, \ldots, r_n) = 0] \leq \frac{d}{|T|}.$$

In a multilinear polynomial, the total degree is bounded by n, the number of variables. In the above protocol, we therefore need only to adapt our test domain T to the total degree d of the given polynomials (which must be the same for isomorphic ones). That is, we let $T = \{1, \ldots, 2d\}$. Then we can bound the probability in equation (1) by $d/|T| = 1/2$, so that the verifier accepts again with probability at most $3/4$ for any prover.

Since the verifier has to evaluate the polynomials, the numbers obtained shouldn't get too large. The value of a polynomial on T is bounded by $C(2d)^d d^d = 2^{O(d \log d)}$, where C is the largest coefficient of the polynomial. Thus, if d is some polynomial in n, then the *length* of all occuring values is polynomially bounded.

However, the verifier can in fact deal with larger degrees. Let N be the length of the encoding of our input polynomials and assume that the total degree d is bounded by $2^{O(N)}$. Note that this is always the case as long as the degrees occur as binary numbers in the encoding, so that this is a very weak restriction. Now the polynomials can take on values up to $2^{2^{cN}}$ on T^n, for some constant $c > 0$, so that the polynomial-time verifier can no longer evaluate them at certain points. Instead, the verifier, does now all computations modulo some prime number p.

Let $p_1, p_2, \ldots$ denote an enumeration of all prime numbers. By the Prime Number Theorem, p_i is close to $i/\ln i$. Therefore, the product of the first 2^{cN} primes exceeds $2^{2^{cN}}$, and their length is still polynomially bounded. By the Chinese Remainder Theorem, any number $a \leq 2^{2^{cN}}$ has a unique sequence of remainders when taken modulo p_i for $i = 1, \ldots, 2^{cN}$. Let us even extend the sequence by a factor of 16, i.e., up to length 2^{cN+4}. Consider two numbers $a \neq b \leq 2^{2^{cN}}$. For a randomly chosen prime p_i, where $i \leq 2^{cN+4}$, the probability that a and b will be different modulo p_i is large: $\mathbf{Pr}[a \equiv b \pmod{p_i}] \leq \frac{1}{16}$.

The verifier can use this as follows. He starts by picking uniformly at random numbers $q_1, \ldots, q_b \leq 2^{cN+4}$, for some b we will determine below. Seeking for a prime, the verifier can now either check himself the q_i's using the ZPP algorithm by Adleman and Huang [AH92], or ask the prover to send him a proof whether q_i is prime or not. (Recall that primality is in NP $\cap$ coNP.) With the latter method, we add one round of communication to the above protocol. If all q_i's where non-prime, the verifier stops and accepts immediately. Otherwise, the verifier takes a prime and does all the forthcoming computations modulo that prime.

We can bound the probability that the verifier doesn't find a prime as follows. Let c' be some constant such that $\ln 2^{cN+4} \leq c'N$. Then q_i is prime with probabilty at least $\frac{1}{c'N}$. Therefore, the probability that none of the q_i's is prime is bounded by

$$(1 - \frac{1}{c'N})^b \leq \frac{1}{16}, \quad \text{for } b \geq 4c'N.$$

By the above discussion, the extra error introduced by computing modulo a prime is less than $\frac{1}{16} + \frac{1}{16} = \frac{1}{8}$. In total, the verifier will accept two isomorphic polynomials with probability at most $3/4 + 1/8 = 7/8$. Two non-isomorphic polynomials are still accepted with probability one.

Theorem 4.2 *The non-isomorphism problem for multivariate polynomials over* $\mathbf{Q}$ *of exponentially bounded degree is in* $\mathrm{BP} \cdot \mathrm{NP}$.

If the polynomials are over a *finite* field, say $\mathrm{GF}(q)$, where q is some prime power, we run into the problem that there are not enough elements for our set T in order to make the above protocol work. Instead of $\mathrm{GF}(q)$, we take the extension field $\mathrm{GF}(q^t)$, where t is the smallest integer such that $q^t \geq dn$, so that $t = \lceil \log_q dn \rceil$. Then we can set $T = \mathrm{GF}(q^t)$. Note that two polynomials over $\mathrm{GF}(q)$ are isomorphic if and only if they are isomorphic over any extension field (see for example [vdWae70] for background).

The verifier must be able to evaluate a polynomial at a given point in the extension field. For this, the verifier needs an irreducible polynomial $\phi(x) \in \mathrm{GF}(q)[x]$ of degree t. Note that the degree d of the polynomials in $\mathrm{GF}(q)$ is bounded by $q - 1$ and can therefore be considered as constant. Thus $t = O(\log n)$ and the verifier can cycle through all polynomials in $\mathrm{GF}(q)[x]$ of degree t and check irreducibility in polynomial time using the Berlekamp algorithm [Ber70]. So the verifier will find an irreducible polynomial $\phi(x)$. Then $\mathrm{GF}(q^t)$ is isomorphic to $\mathrm{GF}(q)[x]/\phi(x)$. Therefore, knowing $\phi(x)$, the verifier can do all computation needed in polynomial time. Now, the protocol can proceed as above.

Theorem 4.3 *The non-isomorphism problem for multivariate polynomials over a finite field is in* $\mathrm{BP} \cdot \mathrm{NP}$.

5 A Zero-Knowledge Interactive Proof for BP1-Iso

An IP protocol for a set L is a *zero-knowledge* protocol [GMR89], if it decides L correctly in the usual way and, in addition, there is a prover such that for any $x \in L$ the communication between the prover and the verifier on input x looks random to the verifier in the sense that there is a probabilistic polynomial-time algorithm that outputs some communication sequence of the interactive proof with the same probability as it occurs there.

Using the idea for the interactive proof for the complement of BP1-Iso, we give a zero-knowledge interactive proof for BP1-Iso in this section. The input are two one-time-only branching programs B_0 and B_1, both over n variables $x_1, \ldots, x_n$. Let $T = \{1, \ldots, 2n\}$.

V: the verifier randomly picks points $\mathbf{r}_1, \ldots, \mathbf{r}_k \in T^n$, where $k = \lceil n \log n \rceil + 2$ and sends them to the prover.

P: the prover randomly chooses $i \in \{0, 1\}$, a permutation φ, and sends $y_l = p_{B_i} \circ \varphi(\mathbf{r}_l)$, for $l = 1, \ldots, k$, to the verifier.

V: the verifier randomly picks $j \in \{0, 1\}$ and sends it to the prover.

P: the prover sends a permutation π to the verifier such that $p_{B_j} \circ \pi(\mathbf{r}_l) = y_l$, for $l = 1, \ldots, k$.

V: finally, the verifier accepts if and only if this latter condition about π in fact holds.

If $B_0 \cong B_1$ and the prover behaves as described in the protocol, then the verifier will always accept. Furthermore, consider the following algorithm A that randomly picks $j \in \{0,1\}$, $\mathbf{r}_1, \ldots, \mathbf{r}_k \in T^n$, and a permutation φ and outputs $\mathbf{r}_l$ and $p_{B_j} \circ \varphi(\mathbf{r}_l)$, for $l = 1, \ldots, k$, and furthermore j and φ.

Then any output of A is a possible communication in the above protocol. Moreover, any output of A has the same occurence probability as in the protocol. Therefore, we have the zero-knowledge property.

If $B_0 \not\cong B_1$, then, by arguments similar to those in Section 3, the verifier will accept with probability at most 3/4, no matter what the prover does.

Similar as explained in Section 4, we can derive a zero-knowledge interactive proof from the above one for multivariate polynomials over some field.

Acknowledgements

I want to thank Manindra Agrawal and Uwe Schöning for many helpful discussions. Bernd Borchert gave me some useful pointers to the OBDD-literature.

References

[AH92] L. Adleman and M.-D. Huang. Primality Testeing and Abelian Varieties over Finite Fields. *Lecture Notes in Mathematics 1512*, Springer Verlag, 1992.

[AT96] M. Agrawal and T. Thierauf. The Boolean Isomorphism Problem. ECCC TR96-032, 1996. Available at http://www.eccc.uni-trier.de/eccc/.

[Bab85] L. Babai. Trading group theory for randomness. In *17th ACM Symposium on Theory of Computing*, 421-429, 1985.

[BDG88] J. Balcázar, J. Díaz, and J. Gabarró. Structural Complexity I. EATCS Monographs on Theoretical Computer Science, Springer-Verlag, 1988.

[BDG91] J. Balcázar, J. Díaz, and J. Gabarró. Structural Complexity II. EATCS Monographs on Theoretical Computer Science, Springer-Verlag, 1991.

[Ber70] E. Berlekamp. Factoring Polynomials over Large Finite Fields. *Mathematics of Computation 24(111)*, 713-735, 1970.

[BCGKT95] N. Bshouty, R. Cleve, R. Gavaldà, S. Kannan, C. Tamon. Oracles and queries that are sufficient for exact learning. ECCC TR95-015, 1995. Available at http://www.eccc.uni-trier.de/eccc/.

[BHZ87] R. Boppana, J. Hastad, and S. Zachos. Does co-NP have short interactive proofs? *Information Processing Letters 25*, 27-32, 1987.

[BR93] B. Borchert, D. Ranjan. The Subfunction Relations are Σ_2^p–complete, Technical Report MPI-I-93-121, MPI Saarbrücken, 1993.

[BRS95] B. Borchert, D. Ranjan, F. Stephan. On the Computational Complexity of some Classical Equivalence Relations on Boolean Functions. Forschungsberichte Mathematische Logik, Universität Heidelberg, Bericht Nr. 18, Dezember 1995. Also as ECCC TR96-033, 1996. Available at http://www.eccc.uni-trier.de/eccc/.

[BCW80] M. Blum, A. Chandra, M. Wegman. Equivalence of free Boolean graphs can be decided probabilistically in polynomial time. *Information Processing Letters 10(2)*, 80-82, 1980.

[Bry86] R. Bryant. Graph-Based Algorithms for Boolean Function Manipulation. *IEEE Trans. Comput. C 35(6)*, 677-691, 1986.

[CK91] P. Clote, E. Kranakis. Boolean functions, invariance groups, and parallel complexity. *SIAM Journal on Computing 20(3)*, 553-590, 1991.

[GG95] R. Gavaldà and D. Guijarro. Learning Ordered Binary Decision Diagrams. Manuscript, 1995. Available at http://www-lsi.upc.es/ gavalda.

[GMR89] S. Goldwasser, S. Micali, C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing 18*, 186-208, 1989.

[GS89] S. Goldwasser, M. Sipser. Privat coins versus public coins in interactive proof systems. *Advances in Computing Research. Vol. 5: Randomness and Computation*, S. Micali (Ed.), JAI Press, 73-90, 1989

[Hof82] C. Hoffmann. Group-theoretic algorithms and graph isomorphism. *Lecture Notes in Computer Sience 136*, Springer Verlag, 1982.

[HU79] J. Hopcroft, J. Ullman. Introduction to Automata Theory, Languages, and Computation. Addison-Wesley, 1979.

[KST93] J. Köbler, U. Schöning, J. Toran. The Graph Isomorphism Problem: Its Structural Complexity. Birkhäuser Verlag, 1993.

[MS94] C. Meinel and A. Slobodová. On the Complexity of Constructing Optimal Ordered Binary Decision Diagrams. Proceedings of MFCS '94, Springer Verlag, Lecture Notes in Computer Sience 841, 515-524, 1994.

[Sch88] U. Schöning. Graph isomorphism is in the low hierarchy. *Journal of Computer and System Sciences 37*, 312-323, 1988.

[Sch89] U. Schöning. Probabilistic complexity classes and lowness. *Journal of Computer and System Sciences 39*, 84-100, 1989.

[S80] J. Schwartz. Fast Probabilistic Algorithms for Verification of Polynomial Idendities. *Journal of the ACM 27(4)*, 701-717, 1980.

[SW92] D. Sieling and I. Wegener. Reductions of BDDs in Linear Time. Manuscript, 1992.

[vdWae70] B. van der Waerden. Algebra 1. Heidelberger Taschenbücher 1970.

[Zip79] R. Zippel. Probabilistic Algorithms for Sparse Polynomials. In *Proceedings of EUROSAM 79*, Springer Verlag, Lecture Notes in Computer Sience 72, 1979.