Electronic Colloquium on Computational Complexity, Report No. 77 (2012)

Instance Compression for the Polynomial Hierarchy and Beyond

Chiranjit Chakraborty
Rahul Santhanam

School of Informatics, University of Edinburgh, UK

Abstract. We define instance compressibility ([5, 7]) for parametric problems in PH and PSPACE. We observe that the problem Σ_iCIRCUITSAT of deciding satisfiability of a quantified Boolean circuit with $i-1$ alternations of quantifiers starting with an existential quantifier is complete for parametric problems in the class Σ_i^p with respect to W-reductions, and that analogously the problem QBCSAT (Quantified Boolean Circuit Satisfiability) is complete for parametric problems in PSPACE with respect to W-reductions. We show the following results about these problems:

1. If CIRCUITSAT is non-uniformly compressible within NP, then Σ_iCIRCUITSAT is non-uniformly compressible within NP, for any $i \geq 1$.
2. If QBCSAT is non-uniformly compressible (or even if satisfiability of quantified Boolean CNF formulae is non-uniformly compressible), then PSPACE $\subseteq$ NP/poly and PH collapses to the third level.

Next, we define Succinct Interactive Proof (Succinct IP) and by adapting the proof of IP = PSPACE ([4, 2]), we show that QBFORMULASAT (Quantified Boolean Formula Satisfiability) is in Succinct IP. On the contrary if QBFORMULASAT has Succinct PCPs ([12]), Polynomial Hierarchy (PH) collapses.

1 Introduction

An NP problem is said to be instance compressible if there is a polynomial-time reduction mapping instances of size m and parameter n to instances of size $poly(n)$ (possibly of a different problem). The notion of instance compressibility for NP problems was defined by Harnik and Naor ([5]) motivated by applications in cryptography. This notion is closely related to the notion of polynomial kernelizability in parametrized complexity ([7, 15, 9]), which is motivated by algorithmic applications. Fortnow and Santhanam showed ([12], Theorem 3.1) that the compressibility of the satisfiability problem for Boolean formulae (even non-uniformly) is unlikely, since it would imply that the Polynomial Hierarchy (PH) collapses. Since then, there's been a very active stream of research extending this negative result to other problems in NP ([7, 18] etc.). Instance compressibility is a useful notion for complexity theory as well - Buhrman and Hitchcock [6] use it to study the question of whether NP has sub-exponentially-sparse complete sets.

Given different possibilities of application of this notion, it is a natural question whether we can extend it to other complexity classes, such as PH and PSPACE. Our first contribution here is to define such an extension. The key to defining instance

ISSN 1433-8092

compressibility for NP problems is that there is a notion of "witness" for instances of NP problems, and in general the witness size can be much smaller than the instance size. We observe that the characterization of PH and PSPACE using alternating time Turing machines yields a natural notion of "guess size" - namely the total number of non-deterministic or co-non-deterministic bits used during the computation. We use this characterization to extend the definition of compressibility to parametric problems in PH and PSPACE in a natural way.

Some proposals ([8, 10]) have already been made in the parametrized complexity setting for defining in general the parametrized complexity analogue of a classical complexity class. Our definition seems similar in spirit, but all the problems we consider *are* in fact fixed-parameter tractable. What we are interested in is whether they are instance-compressible, or equivalently whether they have polynomial-size kernels.

One of our main motivations is to provide a structural theory of compressibility, analogous to the theory in the classical setting. Intuitively, instance compressibility provides a different, more relaxed notion of "solvability" than the traditional notion. So it is interesting to study what kinds of analogues to classical results hold for the new notion. The result of Fortnow and Santhanam ([12]) can be thought of as an analogue of the Karp-Lipton theorem ([14], Theorem 6.1), since non-uniform compressibility is a weakening of the notion of non-uniform solvability. Other well-known theorems in the classical setting are that NP has polynomial-size circuits iff all of PH does, as well as the Karp-Lipton theorem for PSPACE ([14], Theorem 4.1). The main results we prove here are analogues of these results for instance compressibility.

Our first main result is, if the language CIRCUITSAT is non-uniformly compressible within NP (i.e., the reduction is to an NP problem), then so is the language Σ_iCIRCUITSAT, which is in some sense complete for parametric problems in the class Σ_i^p. Note that we need a stronger assumption here compared to that in the Fortnow-Santhanam result ([12]): they need only to assume that SAT is compressible. This reflects the fact that the proof is more involved - it relies on the Fortnow-Santhanam result ([12]) as well as on the techniques used in the classical case. In addition, the code used by the hypothetical compression for CIRCUITSAT shows up not just in the resulting compression *algorithm* for Σ_iCIRCUITSAT, but also in the *instance* generated - this is why we need to work with circuits, as they can simulate any polynomial-time computation. This ability to interpret code as data is essential to our proof. We give more intuition about the proof in Section 3, where the detailed proof can also be found. We also observe that under the assumption of Σ_3CIRCUITSAT being compressible (we make no assumption about the complexity of the set we are reducing to, nor do we require the compression to be non-uniform), all of the PH is compressible as well.

Our second main result is that if QBCNFSAT is non-uniformly compressible, the Polynomial Hierarchy collapses to the third level. The proof of this is easier and an adaptation of the Fortnow-Santhanam technique ([12]) to PSPACE. Here we consider an "OR" version of the problem as they do, and derive the collapse of the hierarchy from the assumption that the OR version is compressible. In the case of NP, showing that compressing the OR version is at least as easy as compressing SAT is easier as there are no quantifiers; however, this is not the case for PSPACE and this is where we need to work a little harder.

Our third result is an analogue of the IP = PSPACE ([4, 2]) result in the parametric world. We define the class Succinct IP, which consists of parametric problems with interactive protocols where the total amount of communication is polynomial in the size of the parameter. We observe that the traditional proof of IP = PSPACE ([4, 2]) can be adapted to show that the problem of determining whether a quantified Boolean formula is valid, has succinct interactive proofs. This demonstrates a difference between succinctness in an interactive setting and succinctness in a non-interactive setting - it is shown in [12] that if SAT has succinct probabilistically checkable proofs, then PH collapses.

There are many open problems in the compressibility theory for NP, such as, whether there are any unlikely consequences of SAT being probabilistically compressible, and whether the problem AND-SAT is deterministically compressible. Our hope is that extending the theory to larger classes such as PH and PSPACE will give us more "room" to work with. Besides, if we manage to settle these questions for the larger classes, the techniques can be translated back to NP.

2 Some Complexity Theory Notions

Definition 1. ***Parametric problem:*** *A parametric problem is a subset of* $\{ \langle x, 1^n \rangle \mid x \in \{0,1\}^*, n \in \mathbb{N} \}$. *The term* n *is known as the parameter of the problem.*

NP problems in parametric form: Now consider some popular NP languages in parametric form.
SAT = $\{\langle \varphi, 1^n \rangle \mid \varphi$ is a satisfiable formula in CNF, and n is the number of variables in $\varphi\}$.
VC = $\{\langle G, 1^{k\ log(m)} \rangle \mid G$ has a vertex cover of size at most k, where $m = |G|\}$.
CLIQUE = $\{\langle G, 1^{k\ log(m)} \rangle \mid G$ has a clique of size at least k, where $m = |G|\}$.
DOMINATINGSET = $\{\langle G, 1^{k\ log(m)} \rangle \mid G$ has a dominating set of size at most k, where $m = |G|\}$.
OR-SAT = $\{\langle \{\varphi_i\}, 1^n \rangle \mid$ At least one φ_i is satisfiable, and each φ_i is of bit-length at most $n\}$.

For the parametric problems above in NP, the parameter can be interpreted as the *witness size* for some natural NTM deciding the language. For example in SAT, the number of variables, which captures the witness of satisfiability problem, is taken as the parameter. Note that in the definitions of the CLIQUE, VC and DOMINATINGSET problems, the parameter is $k\ log(m)$ rather than k as in the typical parametrized setting. So we can say that any parametric problem is in NP if there exists a polynomial time computable NTM to solve it.

Definition 2. ***Compression of parametric problem:*** *Suppose here* L *is a parametric problem.* L *is said to be compressible within a complexity class* A *if there is a polynomial p(.), and a polynomial-time computable function* f *, such that for each* $x \in \{0, 1\}^*$ *and* $n \in \mathbb{N}$, $|f(\langle x, 1^n \rangle)| \leq p(n)$ *and* $\langle x, 1^n \rangle \in L$ *iff* $f(\langle x, 1^n \rangle) \in L_A$ *for some parametric problem* L_A *in the complexity class* A.

Definition 3. ***Non-uniform Compression:*** *A parametric problem* L *is said to be compressible with advice s(., .) if the compression function is computable in deterministic* *polynomial time when given access to an advice string of size $s(m, n)$ which depends only on m and n but not on the actual instance. Here m is the instance length and n is the parameter. L is non-uniformly compressible if s is polynomially bounded in m and n.*

In other words, we can say that the machine compressing the parametric problem in the preceding definition takes advice in case of *Non-uniform Compression*.

Definition 4. ***W-Reduction:*** *[5] Given parametric problems L_1 and L_2 , L_1 W-reduces to L_2 (denoted $L_1 \leq_w L_2$) if there is a polynomial-time computable function f and polynomials p_1 and p_2 such that:*
1.$f(\langle x, 1^{n_1} \rangle)$ is of the form $\langle y, 1^{n_2} \rangle$ where $n_2 \leq p_2(n_1)$.
2.$f(\langle x, 1^{n_1} \rangle) \in L_2$ iff $\langle x, 1^{n_1} \rangle \in L_1$.

The semantics of a W-reduction is that if L_1 W-reduces to L_2 , it is at least as hard to compress L_2 as it is to compress L_1 . If $L_1 \leq_w L_2$ and L_2 is compressible, then L_1 is compressible. One can prove that OR-SAT $\leq_w$ SAT ([19]).
As we have already mentioned, our primary objective is to extend the idea of compression to higher classes, namely Polynomial Hierarchy (PH) and PSPACE [16]. In our work, by a quantified Boolean formula, we mean a Boolean formula in prenex normal form where the quantifiers are in the beginning as follows, $\psi = Q_1\, x_1\, Q_2\, x_2 \ldots Q_n\, x_n\, \phi$, for any Boolean formula ϕ. Similarly we can consider quantified Boolean circuits. Let us now consider some standard PH and PSPACE languages but in parametric form.

CIRCUITSAT = $\{\langle C, 1^n \rangle \mid$ *C is a satisfiable circuit, and n is the number of variables in C*$\}$

Σ_i**SAT** = $\{\langle \varphi, 1^n \rangle \mid$ *φ is a satisfiable quantified Boolean formula in CNF with $i-1$ alternations where alternately odd position quantifiers are $\exists$ and even position quantifiers are $\forall$, and $n = (n_1 + n_2 + \ldots + n_i)$ where n_i is the number of the variables corresponding to i^{th} quantifier*$\}$

Σ_i**CIRCUITSAT** = $\{\langle C, 1^n \rangle \mid$ *C is a satisfiable quantified circuit with $i-1$ alternations where alternately odd position quantifiers are $\exists$ and even position quantifiers are $\forall$, and $n = (n_1 + n_2 + \ldots + n_i)$ where n_i is the number of the variables corresponding to i^{th} quantifier*$\}$
Similarly we can define Π_iSAT and Π_iCIRCUITSAT in parametric form.

QBCNFSAT = $\{\langle \varphi, 1^n \rangle \mid$ *φ is a satisfiable quantified Boolean formula in CNF, and n is the number of variables*$\}$

QBFORMULASAT = $\{\langle \varphi, 1^n \rangle \mid$ *φ is a satisfiable quantified Boolean formula (not necessarily in CNF), and n is the number of variables*$\}$
If φ is replaced by the circuit C, then similarly we can define **QBCSAT**.

OR-QBCNFSAT = $\{\langle \{\varphi_i\}, 1^n \rangle \mid$ *Each φ_i is a quantified Boolean formula in CNF and at least one φ_i is satisfiable, and each φ_i is of bit-length at most n*$\}$.

Now we can try to generalize. For any language L we can define, **OR-L** = $\{\langle \{x_i\}, 1^n \rangle \mid$ At least one $x_i \in$ L, and each x_i is of bit-length at most $n\}$.

Here we would like to mention that the non-parametric versions of Σ_iSAT and Σ_iCIRCUITSAT are complete for the class Σ_i^p according to Cook-Levin reduction, and similarly the non-parametric versions of QBCNFSAT, QBFORMULASAT and QBCSAT are complete for PSPACE.

We can define a parametric problem corresponding to any language L in the class Σ_i^p, or more precisely to the $i+1$-ary polynomial-time computable relation R defining L, as follows.

Definition 5. *For any Σ_i^p language L, we can define a parametric problem in Σ_i^p, L_R = $\{\langle\, x,\, 1^n\, \rangle \mid \exists\, u_1 \in \{0,1\}^{n_1}\ \forall\, u_2 \in \{0,1\}^{n_2} \ldots Q_i\ u_i \in \{0,1\}^{n_i}\ R\ (x,\, u_1\,,\, \ldots,\, u_i)$ = 1 and n = $(n_1 + n_2 + \ldots + n_i)$ where n_i is the parameter corresponding to i^{th} quantifier$\}$*

We can do essentially the same thing for any language $L \in$ PSPACE using the characterization of PSPACE as alternating polynomial time ([1], Corollary 3.6) as follows:

Proposition 1. *Any language L is in PSPACE if and only if it is decidable by an Alternating Turing machine in polynomial time.*

Now we can define,

Definition 6. *For any PSPACE language L, we can define a parametric problem in PSPACE, L_R = $\{\langle\, x,\, 1^n\, \rangle \mid Q_1\, u_1 \in \{0,1\}^{n_1}\ Q_2\, u_2 \in \{0,1\}^{n_2} \ldots Q_i\, u_i \in \{0,1\}^{n_i}\ R$ $(x,\, u_1\,,\, \ldots,\, u_i)$ = 1 and n = $(n_1 + n_2 + \ldots + n_i)$ where all the Q variables denote $\exists$ or $\forall$ alternately, depending on whether its suffix is odd or even, i is polynomially bounded with respect to the size of x and n_i is the parameter corresponding to i^{th} quantifier$\}$*

So using the general definition of compression of any language in parametric form given above, we can define the compression for all the PH and PSPACE parametric problems where the "witness length" or "guess length" is the parameter of the problem.

Proposition 2. *Σ_iCIRCUITSAT is a complete parametric problem with respect to W-reduction for the class of parametric problems in Σ_i^p.*

Proof. Firstly we can observe that Σ_iCIRCUITSAT is among the parametric problems in the class Σ_i^p as there is an Alternating Turing Machine accepting this language with $i-1$ alternations, starting with existential guesses. Let us now consider the language $L \in \Sigma_i^p$. Then there exists a polynomial-time computable relation R such that,
$x \in L \Leftrightarrow \exists\, u_1 \in \{0,1\}^{n_1}\ \forall\, u_2 \in \{0,1\}^{n_2} \ldots Q_i\ u_i \in \{0,1\}^{n_i}\ R\ (x, u_1\,, \ldots, u_i) = 1$,
where Q_i denotes $\exists$ or $\forall$ depending on whether i is odd or even respectively.

Now consider the parametric problem L_R corresponding to L where the parameter is the number of guess bits used by R. We know that any polynomial time computable relation has uniform polynomial size circuits ([16], Theorem 6.7). Let C_m be the circuit on inputs of length m - we can generate C_m from 1^m in polynomial time. Hence, $\langle x$, $1^n\rangle \in L_R \Leftrightarrow \exists\, u_1 \in \{0,1\}^{n_1}\ \forall\, u_2 \in \{0,1\}^{n_2} \ldots Q_i\ u_i \in \{0,1\}^{n_i}\ C_m\ (x, u_1\,, \ldots, u_i)$ = 1, where Q_i denotes $\exists$ or $\forall$ depending on whether i is odd or even respectively. This gives a W-reduction from the parametric problem L_R to Σ_iCIRCUITSAT, since the length of the parameter is preserved. □

A similar proposition holds for Π_iCIRCUITSAT as well. We can also show, using essentially the same proof, a completeness result for PSPACE.

Proposition 3. QBCSAT *is a complete parametric problem for the class of parametric problems in* PSPACE *with respect to W-reduction.*

Proof. Firstly we can observe that QBCSAT is among the parametric problems in the class PSPACE as there is an Alternating Turing Machine accepting this language with at most n alternations, where n is the number of variables of the QBCSAT instance. Let us now consider the language $L \in$ PSPACE. Then from Proposition 1 there exists a polynomial-time computable relation R such that,
$x \in L \Leftrightarrow Q_1\ u_1 \in \{0,1\}^{n_1}\ Q_2\ u_2 \in \{0,1\}^{n_2} \ldots Q_i\ u_i \in \{0,1\}^{n_i}\ R\,(x, u_1\ , \ldots , u_i)$ = 1, where all the Q variables denote $\exists$ or $\forall$ alternately, depending on whether its suffix is odd or even. i is polynomially bounded with respect to the size of x.

Now consider the parametric problem L_R corresponding to L where the parameter is the number of guess bits used by R. We know that any polynomial time computable relation has uniform polynomial size circuits ([16], Theorem 6.7). Let C_m be the circuit on inputs of length m - we can generate C_m from 1^m in polynomial time. Hence, $\langle x, 1^n\rangle \in L_R \Leftrightarrow Q_1\ u_1 \in \{0,1\}^{n_1}\ Q_2\ u_2 \in \{0,1\}^{n_2} \ldots Q_i\ u_i \in \{0,1\}^{n_i}\ C_m\,(x, u_1\ , \ldots , u_i)$ = 1, where all the Q variables denote $\exists$ or $\forall$ alternately, depending on whether its suffix is odd or even. This gives a W-reduction from the parametric problem L_R to QBCSAT, since the length of the parameter is preserved. □

We note that all the parametric problems we have defined so far are in fact fixed-parameter tractable, simply by using brute force search.

Proposition 4. QBCSAT *is solvable in time $O(2^n poly(m))$ by brute force enumeration.*

3 Instance Compression for Polynomial Hierarchy

3.1 Instance Compression in second level

In this section, we are going to show that non-uniform compression of CIRCUITSAT within NP implies the non-uniform compression of Σ_2CIRCUITSAT within NP as well. In the next subsection, essentially by using induction and relating this consequence, we show how to extend this to the entire Polynomial Hierarchy. We have used the following result by Fortnow and Santhanam ([12], Theorem 3.1):

Theorem 1. *If* OR-SAT *is compressible, then* CoNP $\subseteq$ NP/poly*, and hence* PH *collapses.*

The same technique actually shows that, for any parametric problem L for which OR-L (section 2) is compressible, lies within CoNP/poly.

Theorem 2. *If* CIRCUITSAT *is non-uniformly compressible within* NP*, then* Σ_2CIRCUITSAT *is non-uniformly compressible within* NP.

Proof. Let us consider the parametric problem Σ_2CIRCUITSAT first. For the sake of convenience, we often omit the parameter when talking about an instance of this problem. According to the definition,

$$C \in \Sigma_2\text{CIRCUITSAT} \Leftrightarrow \exists u \in \{0,1\}^{n_1} \forall v \in \{0,1\}^{n_2} C(u,v) = 1 \quad (1)$$

$$C \notin \Sigma_2\text{CIRCUITSAT} \Leftrightarrow \forall u \in \{0,1\}^{n_1} \exists v \in \{0,1\}^{n_2} C(u,v) = 0 \qquad (2)$$

Let m be the length of the description of the circuit C and $n = (n_1 + n_2)$ to be the number of variables of C.

Let us now fix a specific $u = u_1$. Now, we can define a new parametric problem $L^{'}$ as follows,

$$\langle C, u_1, 1^{n_2} \rangle \in L^{'} \Leftrightarrow \forall v \in \{0,1\}^{n_2} C(u_1, v) = 1 \qquad (3)$$

$$\langle C, u_1, 1^{n_2} \rangle \notin L^{'} \Leftrightarrow \exists v \in \{0,1\}^{n_2} C(u_1, v) = 0 \qquad (4)$$

It is clear from the above definition that $L^{'}$ is a parametric problem in CoNP (of instance size $\leq O(m+n_1)$) and any instance of $L^{'}$ can be polynomial-time reduced to an instance of CIRCUIT-UNSAT, say $C^{'}$ (because CIRCUIT-UNSAT, the parametric problem of all unsatisfiable circuits, is CoNP-Complete with respect to W-reduction). As shown in Proposition 2, the size of the witness will be preserved in this reduction.

$C \in \Sigma_2$CIRCUITSAT $\Leftrightarrow \exists u_1 \langle C, u_1 \rangle \in L^{'}$ and $\langle C, u_1 \rangle \in L^{'} \Leftrightarrow C^{'} \in$ CIRCUIT-UNSAT. Here the instance length $|C| = m$ and $|C^{'}| = poly(m)$. $poly(.)$ is denoting just an arbitrary polynomial function.

Let g be the polynomial-time reduction used to obtain $C^{'}$ from C and u_1. Namely, $C^{'} = g(C, u_1)$. If CIRCUITSAT is non-uniformly compressible within NP, using the same reduction we can non-uniformly compress CIRCUIT-UNSAT within CoNP. That means we can reduce a CIRCUIT-UNSAT instance into another CIRCUIT-UNSAT instance in polynomial time, as CIRCUIT-UNSAT is CoNP-complete with respect to W-reduction. Assume this polynomial time compression function be f_1 with polynomial size advice. So we will use f_1 to compress CIRCUIT-UNSAT instance $C^{'}$ to another CIRCUIT-UNSAT instance, say $C^{''}$, of size $poly(n_2)$.

Therefore, $C^{'} \in$ CIRCUIT-UNSAT $\Leftrightarrow C^{''} = f_1(\, C^{'}, w_1\,) = f_1(\, g(C, u_1), w_1\,) \in$ CIRCUIT-UNSAT, where $|C^{''}| = poly(n_2)$ and the string w_1 (of size at most $poly(m)$) is capturing the notion of polynomial size advice. Here the compression function f_1 is computable in polynomial (in m) time.

Now, if CIRCUITSAT is non-uniformly compressible within NP so is SAT as SAT is W-reducible to CIRCUITSAT. Now, OR-SAT is also non-uniformly compressible as OR-SAT W-reduces to SAT. It can be proved from Theorem 1 that if OR-SAT is non-uniformly compressible then CoNP $\subseteq$ NP/poly, as mentioned in the beginning of this section.

Now combining the statements in the above paragraph we can say that if CIRCUIT-SAT is non-uniformly compressible within NP then CoNP $\subseteq$ NP/poly. So we can now reduce our parametric problem in CoNP (here CIRCUIT-UNSAT) instance $C^{''}$ to a NP-complete parametric problem instance using polynomial size advice. As CIRCUITSAT is a NP-complete with respect to W-reduction, we can reduce $C^{''}$ to a CIRCUITSAT instance, say $C^{'''}$, using a polynomial time computable function f_2 with advice w_2. In the above procedure, the length of the instance definitely will not increase by more than a polynomial factor. So clearly $|C^{'''}| = poly(n_2)$.

So from the above arguments we can say that,
$C^{'} \in$ CIRCUIT-UNSAT $\Leftrightarrow C^{'''} = f_2(\, C^{''}, w_2\,) = f_2(\, f_1(\, g(C, u_1), w_1\,), w_2\,) \in$ CIRCUITSAT, where $|C^{'''}| = poly(n_2)$ and the string w_2 (of size at most $poly(n_2)$) is capturing the notion of polynomial size advice which arises in the proof of Theorem 1. Here the function f_2 is computable in polynomial (in n_2) time.

Now we define a new circuit C_1 as follows. C_1 is a non-deterministic circuit whose non-deterministic input is divided into two strings: u of length n_1 and v of length $poly(n_2)$. Given its non-deterministic input, C_1 first computes $C''' = f_2(\ (f_1(g(C, u),$ $w_1), w_2)$. This can be done in polynomial size in m since the functions f_2, f_1 and g are all polynomial-time computable and C, w_1 and w_2 are all fixed strings of size polynomial in m. It then uses its input v as non-deterministic input to C''' and checks if v satisfies C'''. This can be done in polynomial-size since the computation of a polynomial-size circuit can be simulated in polynomial time. If so, it outputs 1, else it outputs 0. Now we have

$$C \in \Sigma_2\text{CIRCUITSAT} \Leftrightarrow \exists u \in \{0,1\}^{n_1} \exists v \in \{0,1\}^{n_2} C_1(u,v) = 1 \tag{5}$$

$$C \notin \Sigma_2\text{CIRCUITSAT} \Leftrightarrow \forall u \in \{0,1\}^{n_1} \forall v \in \{0,1\}^{n_2} C_1(u,v) = 0 \tag{6}$$

The key point is that we have reduced our original Σ_2CIRCUITSAT question to a CIRCUITSAT question, *without* a super-polynomial blow-up in the witness size. This allows us to apply the compressibility hypothesis again. Also, note that C_1 is computable from C in polynomial size.

After that, using the compressibility assumption for CIRCUITSAT, we can non-uniformly compress C_1 to an instance C_2 of size $poly(n_1 + n_2)$ of a parametric problem in NP. Our final compression procedure just composes the procedures deriving C_1 from C and C_2 from C_1, and since each of these can be implemented in polynomial size, our compression of the original Σ_2CIRCUITSAT instance is a valid non-uniform instance compression. Thus it is shown that if CIRCUITSAT is non-uniformly compressible within NP, Σ_2CIRCUITSAT is also non-uniformly compressible within NP. □

3.2 Instance Compression for higher levels

Now we are going to extend the idea for higher classes. It is not difficult to see, if Σ_2CIRCUITSAT is non-uniformly compressible within NP, Π_2CIRCUITSAT is non-uniformly compressible within CoNP. We will use this in the following theorem.

Theorem 3. *If* CIRCUITSAT *is non-uniformly compressible within* NP*, then* Σ_iCIRCUITSAT *is non-uniformly compressible within* NP *for all* $i > 1$*.*

Proof. Suppose C is a Σ_iCIRCUITSAT instance. So from the definition we can say that, $C \in \Sigma_i\text{CIRCUITSAT} \Leftrightarrow \exists\ u_1 \in \{0,1\}^{n_1}\ \forall\ u_2 \in \{0,1\}^{n_2} \ldots Q_i\ u_i \in \{0,1\}^{n_i}\ C\ (u_1\ ,$ $\ldots, u_i) = 1$,
where Q_i denotes $\exists$ or $\forall$ depending on whether i is odd or even respectively.

Now, suppose CIRCUITSAT is compressible. To prove Σ_iCIRCUITSAT is compressible for all $i > 1$, we have to check the base case at the first place, that is for the case when $i = 2$. From the Theorem 1, we can say that if CIRCUITSAT is non-uniformly compressible within NP, Σ_2CIRCUITSAT is also non-uniformly compressible within NP. So the statement is true for base case.

Now suppose the statement is true for all $i \leq k$. We have to prove that the statement is true for $i = k + 1$ as well. So, assuming CIRCUITSAT is non-uniformly compressible within NP implies Σ_iCIRCUITSAT is non-uniformly compressible within NP for all

$i \leq k$, we have to prove that Σ_{k+1}CIRCUITSAT is also non-uniformly compressible within NP.

Suppose C is a Σ_{k+1}CIRCUITSAT instance of size m. So from the definition we can say that,
$C \in \Sigma_{k+1}$CIRCUITSAT
$\Leftrightarrow \exists\, u_1 \in \{0,1\}^{n_1}\ \forall\, u_2 \in \{0,1\}^{n_2} \ldots Q_{k+1}\ u_{k+1} \in \{0,1\}^{n_{k+1}}\ C\,(u_1\,, \ldots, u_{k+1}) =$ 1,
where Q_{k+1} denotes $\exists$ or $\forall$ depending on whether $(k+1)$ is odd or even respectively.

Now, let us fix u_1 to $u^{'}$. So now we can define a new parametric problem as follows,
$\langle C, u^{'}, 1^{n_2+n_3+\ldots+(n_k+1)}\rangle \in L^{'} \Leftrightarrow \forall\, u_2 \in \{0,1\}^{n_2} \ldots Q_{k+1}\ u_{k+1} \in \{0,1\}^{n_{k+1}}\ C\,(u^{'},$ $u_2\,, \ldots, u_{k+1}) = 1$,
where Q_{k+1} denotes $\exists$ or $\forall$ depending on whether $(k+1)$ is odd or even respectively.

So it is clear from the above definition that $L^{'}$ is a parametric problem in Π_k^p (of instance size $\leq O(m+n_1)$) and any instance of $L^{'}$ can be polynomially reduced to an instance of Π_kCIRCUITSAT (because Π_kCIRCUITSAT is Π_k^p-Complete with respect to W-reduction). As shown in Proposition 2, the size of the witness will be preserved in this reduction. So this reduction is essentially a W-reduction. Suppose this Π_k^pCIRCUITSAT instance is $C^{'}$.

So from the above arguments,
$C \in \Sigma_{k+1}$CIRCUITSAT
$\Leftrightarrow \exists u^{'} \langle C, u^{'}\rangle \in L^{'}$ and $\langle C, u^{'}\rangle \in L^{'}$
$\Leftrightarrow C^{'} \in \Pi_k$CIRCUITSAT
Here the instance length $|C| = m$ and $|C^{'}| = poly(m)$. $poly(.)$ is denoting just an arbitrary polynomial function.

Suppose g is the function to obtain $C^{'}$ from C, running in polynomial (in m) time. Namely, $C^{'} = g(C, u^{'})$.

From the induction hypothesis we can say, Σ_kCIRCUITSAT is non-uniformly compressible within NP. So any Π_kCIRCUITSAT instance, say $C^{'}$ is non-uniformly compressible to a CoNP instance as Π_kCIRCUITSAT = CoΣ_kCIRCUITSAT. After compression suppose the instance is $C^{''}$ which, without loss of generality, can be taken as a CIRCUIT-UNSAT instance as it is a complete for CoNP with respect to W-reduction. Here $|C^{''}| = poly(n^{'})$ where $n^{'} = (n_2 + n_3 + \ldots + n_{k+1})$
So, $C^{'} \in \Pi_k$CIRCUITSAT $\Leftrightarrow C^{''} \in$ CIRCUIT-UNSAT.

Assume f_1 to be the above compression function. So from the above arguments we can say,
$C^{'} \in \Pi_k$CIRCUITSAT $\Leftrightarrow C^{''} = f_1(\,C^{'}, w_1\,) = f_1(\,g(C, u^{'}), w_1\,) \in$ CIRCUIT-UNSAT, where $|C^{''}| = poly(n^{'})$ and the string w_1 (of size at most $poly(m)$) is capturing the notion of polynomial size advice. Here the compression function f_1 is running in polynomial (in m) time.

Now, if CIRCUITSAT is non-uniformly compressible within NP so is SAT as SAT is W-reduced to CIRCUITSAT. Now, OR-SAT is also non-uniformly compressible as OR-SAT W-reduces to SAT. It can be proved from Theorem 1 that if OR-SAT is non-uniformly compressible then CoNP $\subseteq$ NP/poly, as mentioned in the beginning of this section.

Now combining the above statements we can say that if CIRCUITSAT is non-uniformly compressible within NP then CoNP $\subseteq$ NP/poly. So we can now reduce our instance of parametric problem in CoNP (here CIRCUIT-UNSAT), $C^{''}$ into an instance of a parametric problem which is NP-complete, using polynomial size advice. As CIRCUITSAT is a NP-complete with respect to W-reduction, we can reduce $C^{''}$ to a CIRCUITSAT instance, say $C^{'''}$, using a polynomial time computable function f_2 with advice w_2. In the above procedure, the length of the instance definitely will not increase by more than a polynomial factor. So clearly $|C^{'''}| = poly(n_{'})$.

So from the above arguments we can say that,
$C^{'} \in \Pi_k$CIRCUITSAT $\Leftrightarrow C^{'''} = f_2(C^{''}, w_2) = f_2(f_1(g(C, u^{'}), w_1), w_2) \in$ CIRCUIT-SAT, where $|C^{'''}| = poly(n^{'})$ and the string w_2 (of size at most $poly(n^{'})$) is capturing the notion of polynomial size advice which arises in the proof of Theorem 1. Here the compression function f_2 is running in polynomial (in $n^{'}$) time.

Now we define a new circuit C_1 as follows. C_1 is a non-deterministic circuit whose non-deterministic input is divided into two strings: u_1 of length n_1 and v of length $poly(n^{'})$. Given its non-deterministic input, C_1 first computes $C^{'''} = f_2((f_1(g(C, u_1), w_1), w_2)$. This can be done in polynomial size in m since the functions f_2, f_1 and g are all polynomial-time computable and C, w_1 and w_2 are all fixed strings of size polynomial in m. It then uses its input v as non-deterministic input to C''' and checks if v satisfies C'''. This can be done in polynomial-size since the computation of a polynomial-size circuit can be simulated in polynomial time. If so, it outputs 1, else it outputs 0.

Now we have,

$$C \in \Sigma_{k+1}\text{CIRCUITSAT} \Leftrightarrow \exists u_1 \in \{0,1\}^{n_1}\, \exists v \in \{0,1\}^{n^{'}}\, C_1\,(u_1, v) = 1 \quad (7)$$

$$C \notin \Sigma_{k+1}\text{CIRCUITSAT} \Leftrightarrow \forall u_1 \in \{0,1\}^{n_1}\, \forall v \in \{0,1\}^{n^{'}}\, C_1\,(u_1, v) = 0 \quad (8)$$

The key point is that we have reduced our original Σ_{k+1}CIRCUITSAT question to a CIRCUITSAT question, *without* a super-polynomial blow-up in the witness size. This allows us to apply the compressibility hypothesis again. Also, note that C_1 is computable from C in polynomial size.

After that, using the compressibility assumption for CIRCUITSAT, we can non-uniformly compress C_1 to an instance C_2 of size $poly(n_1 + n^{'})$ i.e. $poly(n_1 + n_2 + \ldots + n_{k+1})$ of a parametric problem in NP. Our final compression procedure just composes the procedures deriving C_1 from C and C_2 from C_1, and since each of these can be implemented in polynomial size, our compression of the original Σ_{k+1}CIRCUITSAT instance is a valid non-uniform instance compression.

So using mathematical induction we can say if CIRCUITSAT is non-uniformly compressible within NP, Σ_iCIRCUITSAT is also non-uniformly compressible within NP for all $i > 1$. □

Corollary 1. *If* CIRCUITSAT *is compressible within* NP*,* Π_iCIRCUITSAT *is also non-uniformly compressible within* NP *for all* $i \geq 1$*.*

As Π_iCIRCUITSAT W-reduces to Σ_{i+1}CIRCUITSAT, the above Corollary is trivial. Theorems 2 and 3 require an assumption on non-uniform compressibility in NP. But we don't need this for compressibility of a problem higher in the hierarchy.

Proposition 5. *If* Σ_3CIRCUITSAT *is compressible, then* Σ_iCIRCUITSAT *is compressible for any* $i > 3$.

Proof. This proposition follows from the fact that Σ_3CIRCUITSAT being compressible implies that SAT is compressible. So, by the result of Fortnow and Santhanam (Theorem 1), PH collapses to Σ_3^p. As a result, every parametric problem in the class Σ_i^p W-reduces to Σ_3CIRCUITSAT, as Σ_3CIRCUITSAT is complete for the class Σ_3^p with respect to W-reduction. Hence, Σ_3CIRCUITSAT being compressible, Σ_iCIRCUITSAT is compressible for any $i > 3$. □

4 Instance Compression for PSPACE

In this section, we show that QBCNFSAT is unlikely to be compressible, even non-uniformly - compressibility of QBCNFSAT implies that PSPACE collapses to the third level of the Polynomial Hierarchy. The strategy we adopt is similar to that in Theorem 1 where it is shown that compressibility of SAT implies NP $\subseteq$ CoNP/poly. To show their result, they used the OR-SAT problem, which is W-reducible to SAT ([19]). Thus an incompressibility result for the OR-SAT problem translates directly to a corresponding result for SAT.

We similarly defined OR-QBCNFSAT problem in Section 2. But it is not that easy to show that OR-QBCNFSAT W-reduces to QBCNFSAT. There are a couple of different issues. First the quantifier patterns for the formulae $\{\phi_i\}, i = 1 \dots m$ might all be different. This is easily taken care of, because we can assume quantifiers alternate between existential and universal - this just blows up the number of variables for any formula by a factor of at most 2. The more critical issue is that nothing as simple as the OR works for combining formulae. $\exists x \forall y \phi_1(x, y) \vee \exists x \forall y \phi_2(x, y)$ is not equivalent to $\exists x \forall y (\phi_1(x, y) \vee \phi_2(x, y))$. We are forced to adopt a different strategy as explained below. Later we have found similar strategy is used in [19], though it was in the context of OR-SAT, not OR-QBCNFSAT.

Lemma 1. OR-QBCNFSAT *is W-reducible to* QBCNFSAT

Proof. Let $\langle \{\phi_i\}, 1^n \rangle$ be an OR-QBCNFSAT instance of length m. Assume without loss of generality that each ϕ_i has exactly n variables and that the quantifiers alternate starting with existential quantification over x_1, continuing with quantification over x_2, x_3 etc. We construct in polynomial time in m an equivalent instance of QBCNFSAT with at most $poly(n)$ variables and of size $poly(m)$. We first take care of quantifications. The quantifier patterns for the formulae $\{\phi_i\}, i = 1 \dots m$ might all be different. But we can assume quantifiers alternate between existential and universal - this just blows up the number of variables for any formula by a factor of at most 2. Then we check if the number of input formulae is greater than 2^n or not. If yes, we solve the original instance by brute force search and output either a trivial true formula or a trivial false formula depending on the result of the search. If not, then we define a new formula with $\lceil log(m) \rceil$ additional variables $y_1, y_2 \dots y_k$. We identify each number between 1 and m uniquely with a string in $\{0, 1\}^k$. Now we define the formula ψ_i corresponding to ϕ_i as follows. Let the string $w_i \in \{0, 1\}^k$ correspond to the number i. Then $\psi_i = z_1 \wedge z_2 \dots \wedge z_k \wedge \phi_i$,

where $z_r = y_r$ if $w_r = 1$ and the complement of y_r otherwise. The output formula ψ starts with existential quantification over the y variables followed by the standard pattern of quantification over the x variables followed by the formula which is the OR of all ψ_i's, $i = 1 \ldots m$. So ψ will be as follows:
$\psi = \exists\, y_1\, \exists\, y_2 \ldots \exists\, y_k\, Q_1\, x_1\, Q_2\, x_2 \ldots Q_n\, x_n\, (\psi_1 \vee \psi_2 \vee \ldots \vee \psi_m)$.
Where Q_i's are the quantifications of the x_i's as before. It is not that hard to check that ψ is valid iff one of the ϕ_i's is. □

Theorem 4. *If* QBCNFSAT *is compressible, then* PSPACE $\subseteq$ NP/poly*, and hence* PSPACE = Σ_3^p.

Proof. Using Lemma 1, if QBCNFSAT is compressible, OR-QBCNFSAT is also compressible. From the proof of Theorem 1 we can say for any parametric problem L for which OR-L (section 2) is compressible, lies in CoNP/poly. Thus, since the parametric problem QBCNFSAT is PSPACE-complete and PSPACE is closed under complementation, a compression for OR-QBCNFSAT implies PSPACE is in NP/poly. Hence by the result of Yap [3], it follows that PH collapses to the third level. Combining this with the Karp-Lipton theorem for PSPACE ([14], Theorem 4.1), we have that PSPACE = Σ_3^p. □

5 Succinct IP and PSPACE

IP ([17, 11]) is the class of problems solvable by an interactive proof system. An interactive proof system consists of two machines, a $Prover$, P, which presents a proof that a input string is a member of some language, and a $Verifier$, V, that checks that the presented proof is correct. Now we are extending this idea of IP to Succinct IP, where the total number of bits communicated between $prover$ and the $verifier$ is polynomially bounded in parameter length.

We define $Verifier$ to be a function V that computes its next transmission to the $Prover$ from the message history sent so far. The function V has three inputs:
(1) **Input String**, (2) **Random input** and (3) **Partial message history**

$m_1\#m_2\# \ldots \#m_i$ is used to represent the exchange of messages m_1 through m_i between P and V. The Verifier's output is either the next message m_{i+1} in the sequence or *accept* or *reject*, designating the conclusion of the interaction. Thus V has the function from V: $\Sigma^* \times \Sigma^* \times \Sigma^* \rightarrow \Sigma^* \cup \{$ accept, reject $\}$.
The $Prover$ is a party with unlimited computational ability. We define it to be a function P with two inputs:
(1) **Input String** and (2) **Partial message history**
The Prover's output is the next message to the Verifier. Formally, P: $\Sigma^* \times \Sigma^* \rightarrow \Sigma^*$.
Next we define the interaction between Prover and the Verifier. For particular input string w and random string r, we write $(V \leftrightarrow P)(w, r)$ = *accept* if a message sequence m_1 to m_k exists for some k whereby

1. for $0 \leq i < k$, where i is an even number, $V(w, r, m_1\#m_2\# \ldots \#m_i) = m_{i+1}$;
2. $0 < i < k$, where i is an odd number, $P(w, m_1\#m_2\# \ldots \#m_i) = m_{i+1}$; and
3. the final message m_k in the message history is $accept$.

In the definition of the class Succinct IP, the lengths of the Verifier's random input and each of the messages exchanged are $p(n)$ for some polynomial p that depends only on the Verifier. Here n is the parameter length of input instance. Besides, total bits of messages exchanged is at most $p(n)$ as well.

Succinct IP: A parametric problem L ($\subseteq \{\langle x, 1^n\rangle | x \in \{0,1\}^*, n \in \mathbb{N}\}$) is in Succinct IP if there exist some polynomial time function V and arbitrary function P, with total $poly(n)$ many bits of messages communicated between them and for every function $\tilde{P}$ and string w,

1. $w \in L$ implies Pr[$V \leftrightarrow P$] $\geq$ 2/3, and
2. $w \notin L$ implies Pr[$V \leftrightarrow \tilde{P}$] $\leq$ 1/3.

Here $poly(n)$ denotes some polynomial that depends only on the Verifier and n is the parameter length of input instance w.

We know that QBFormulaSAT is in IP, as IP = PSPACE ([4, 2]). But we can even prove something more. Not only for QBCNFSAT, we can construct Succinct IP protocol for QBFormulaSAT as well. To prove that we are basically going to adapt the formal proof of the part, PSPACE $\subseteq$ IP ([4, 2, 13]).

Proposition 6. QBFormulaSAT $\in$ Succinct IP

Proof. The key idea is to take an algebraic view of Boolean formulae by representing them as polynomials. We are considering the inputs are from some finite field $\mathbb{F}$. We can see that 0, 1 can be thought of both as truth values and as elements of $\mathbb{F}$. Thus we have the following correspondence between formulas and polynomials when the variables take 0/1 values:
$x \wedge y \leftrightarrow X\,.\,Y$
$\bar{x} \leftrightarrow 1 - X$
$x \vee y \leftrightarrow X{*}Y = 1 - (1 - X)(1 - Y)$
So, if there is a Boolean formula $\phi(x_1, x_2, \ldots, x_n)$ of length m, we can easily convert that into a polynomial p of degree at most m following the rules described above.

Let the given formula be,
$\Psi = Q_1\, x_1\, Q_2\, x_2\, Q_3\, x_3 \ldots\, Q_n\, x_n\, \phi(x_1, \ldots, x_n)$,
where the size of Ψ is m. ϕ is any Boolean formula over n variables.

To arithmetize Ψ we introduce some new terms in quantification and rewrite the expression in the following manner:
$\Psi' = Q_1\, x_1\, R\, x_1\, Q_2\, x_2\, R\, x_1\, R\, x_2\, Q_3\, x_3\, R\, x_1\, R\, x_2\, R\, x_3 \ldots\, Q_n\, x_n\, R\, x_1\, R\, x_2 \ldots$ $R\, x_n\, \phi(x_1, \ldots, x_n)$,
Here R is introduced to enable linearize operation on the polynomial as explained later. We now rewrite this Ψ' as follows : $\Psi' = S_1\, x_1\, S_2\, x_2\, S_3\, x_3 \ldots\, S_k\, x_k\, [\phi]$,
where each $S_i \in \{\exists, \forall, R\}$. We are going to define R very soon. We can see that value of k can be at most $O(n^2)$.

For each $i \leq k$ we define the function f_i. We define $f_k(x_1, x_2, \ldots, x_n)$ to be the polynomial p [i.e. $p(x_1, x_2, \ldots, x_n)$] obtained by arithmetization of ϕ. For $i < k$ we define f_i in terms of f_{i+1}:
$S_{i+1} = \forall$: $f_i(\ldots) = f_{i+1}(\ldots, 0).f_{i+1}(\ldots, 1)$;
$S_{i+1} = \exists$: $f_i(\ldots) = f_{i+1}(\ldots, 0){*}f_{i+1}(\ldots, 1)$;

S_{i+1} =R: $f_i(\ldots, a)$ = (1-a)$f_{i+1}(\ldots, 0)$ + $af_{i+1}(\ldots, 1)$.

Here we reorder the inputs of the functions in such a way that variable y_{i+1} is always the last argument. If S is $\exists$ or $\forall$, f_i has one fewer input variable than f_{i+1} does. But if S is R, both of them have same number of arguments. To avoid complicated subscripts, we use "..." which can be replaced by a_1 through a_j for appropriate values of j after the reordering of the inputs.

We can observe that operation R on polynomial doesn't change their values for Boolean inputs. So $f_0()$ is still the truth value of Ψ. Now we can observe that these Rx operation produces a result that is linear in x. We added Rx_1 Rx_2 $\ldots Rx_i$ after Q_ix_i in Ψ' in order to reduce the degree of each variable to 1 prior to the squaring due to arithmetization of Q_i.

We are now ready to describe the protocol. Here P is denoted to be the prover and V to be the verifier as we always use.

Phase 0: [P sends $f_0()$]
$P \rightarrow V$: P sends $f_0()$ to V. V checks that $f_0()$ = 1 and *rejects* if not.

Progressing similarly,

Phase i: [P persuades V that $f_{i-1}(r_1, \ldots)$ is correct if $f_i(r_1, \ldots, r)$ is correct]
$P \rightarrow V$: P sends the coefficients of $f_i(r_1, \ldots, z)$ as a polynomial in z. (Here $r_1 \ldots$ denotes a setting of the variables to the previously selected random values r_1, r_2, ...)
V uses these coefficients to evaluate $f_i(r_1, \ldots, 0)$ and $f_i(r_1, \ldots, 1)$. Then it checks that the polynomial degree is at most 2 and that these identities hold:

$$f_{i-1}(r_1, \ldots) = \begin{cases} f_i(r_1, \ldots, 0).f_i(r_1, \ldots, 1) & \text{if } S_i = \forall \\ f_i(r_1, \ldots, 0) * f_i(r_1, \ldots, 1) & \text{if } S_i = \exists \end{cases}$$

and

$$f_{i-1}(r_1, \ldots, r) = (1 - r)f_i(r_1, \ldots, 0) + rf_i(r_1, \ldots, 1) \text{ if } S_i = R$$

If either fails, V *rejects*.
$V \rightarrow P$: V picks a random Boolean value r from $\mathbb{F}$ and sends it to P. If S_i = R, this r replaces the previous r
Then it goes to phase i+1, where P must persuade V that $f_i(r_1, \ldots, r)$ is correct.

Progressing similarly,

Phase k**+1**: [V checks directly that $f_k(r_1, \ldots, r_n)$ is correct]
V evaluates $p(r_1, \ldots, r_n)$ to compare with the value V has for $f_k(r_1, \ldots, r_n)$. If they are equal, V *accepts*, otherwise V *rejects*. That completes the description of the protocol.

Here polynomial p is nothing but the arithmetization of the formula ϕ, as we have already seen. It can be shown that the evaluation of this polynomial can be done in polynomial time.

For the evaluation of the polynomial p for $r_1, \ldots, r_n$, we will consider ϕ and apply the arithmetization for the nodes individually. We will evaluate the nodes from lower level. Before we evaluate for any node, corresponding inputs are already evaluated and ready to use. Evaluation for each node will take constant amount of time. So total evaluation of p for $r_1, \ldots, r_n$ through modified ϕ will take $poly(m)$ time.

Now we can try to prove that the probability of error is bounded within the limit. If the prover P always returns the correct polynomial, it will always convince V . If P is not honest then we are going to prove that V rejects with high probability:

$$Pr[V\ rejects] \geq (1 - d/|\mathbb{F}|)^k \tag{9}$$

where d is the highest degree of the polynomial sent in each stage. We can see that value of k can be at most $O(n^2)$. As the value of d is 2 in our case, the right hand side of the above expression is at least (1 - $2k/|\mathbb{F}|$), which is very close to 1 for sufficiently large values of $|\mathbb{F}|$. It will be sufficient for us if $|\mathbb{F}|$ is bounded by a large enough polynomial in n.

Now we are going to see how the proof works when the proves is trying to cheat for "no" instance. In the first round, the prover P should send $f_0()$ which must be 1. Then P is supposed to return the polynomial f_1. If it indeed returns f_1 then since f_1(0) + f_1(1) $\neq f_0$() by assumption, V will immediately reject (i.e., with probability 1). So assume that the prover returns some $s(X_1)$, different from $f_1(X_1\ )$. Since the degree d non-zero polynomial $s(X_1)$ - $f_1(X_1\ )$ has at most d roots, there are at most d values r such that $s(r)$ = $f_1(r)$. Thus when V picks a random r,

$$Pr_r[s(r)\ \neq\ f_1(r)]\ \geq\ (1 - d/|\mathbb{F}|) \tag{10}$$

Then the prover is left with an incorrect claim to prove in all the phases. So prover should lie continuously. If P is lucky, V will not understand the lie. To prove equation (9), we will use induction here. We assume the induction hypothesis to be true for $k-1$ steps, that is, the prover fails to prove this false claim with probability at least $\geq (1-d/|\mathbb{F}|)^{k-1}$. Base case is easy to see from equation (10). Thus we have,

$$Pr[V\ rejects]\ \geq\ (1 - d/|\mathbb{F}|).(1 - d/|\mathbb{F}|)^{k-1}\ = (1 - d/|\mathbb{F}|)^k \tag{11}$$

If P is not lucky, as the verifier is evaluating $p()$ explicitly in the last stage, V will anyway detect the lie.

Here in the description of the protocol, we can see that the degree of the polynomial at each stage is at most 2. So we need just constant number of coefficients for encoding such polynomials. coefficients are from the field $\mathbb{F}$ which is of size $poly(m)$. So $O(log(poly(m)))$ i.e. $O(poly(n))$ size messages are sent in any phase. Even, it will be sufficient for us if $|\mathbb{F}|$ is bounded by a large enough polynomial in n. Number of such phases are bounded by (k+1) which is $O(n^2)$. So we have constructed a *Succinct* Interactive proof protocol for QBFORMULASAT. □

Issue in finding Succinct IP protocol for QBCSAT: In case of QBCSAT, similar arithmetization technique will give polynomial of degree much larger size, actually exponential in m. Now, to reduce the error, we have to use Field $\mathbb{F}$ of larger size, basically exponential in m. This will give us each coefficients of the polynomials exchanged between $prover$ and $verifier$ to be of size $log(e^{poly(m)})$, i.e. $poly(m)$, which means the protocol is not succinct.

6 Future Directions

There are various possible directions. Suppose CIRCUITSAT is compressible within a class C. Here we have considered C to be the class NP and got some interesting results.

For any general class C we know from Theorem 1 that the immediate consequence is the collapse of PH at third level. But it is still not known how our results for compression at second level of Polynomial Hierarchy will be affected for compression into an arbitrary class C. Besides, one could try to work under the weaker assumption that SAT or OR-SAT or OR-CIRCUITSAT is compressible instead of CIRCUITSAT. We also don't know whether there are similar implications for probabilistic compression where we allow certain amount of error in compression. One could also try to find a Succinct IP protocol for QBCSAT to show Succinct IP = PSPACE or try to find some negative implications of such a protocol existing for QBCSAT.

References

1. A. K. Chandra, D. C. Kozen, and L. J. Stockmeyer, 'Alternation', Journal of the ACM, Volume 28, Issue 1, pp. 114-133, 1981.
2. A. Shamir. IP = PSPACE. Journal of the ACM, 39(4):869-877, 1992.
3. C. K. Yap. Some consequences of non-uniform conditions on uniform classes. Theoretical Computer Science, 26: 287-300, 1983.
4. C. Lund, L. Fortnow, H. Karloff, and N. Nisan. Algebraic methods for interactive proof systems. Journal of the ACM, 39(4):859-868, 1992.
5. D. Harnik and M. Naor. On the compressibility of NP instances and cryptographic applications. In Proceedings if the 47th Annual IEEE Symposium on Foundations of Computer Science, pages 719-728, 2006.
6. H. Buhrman, J. M. Hitchcock: NP-Hard Sets are Exponentially Dense Unless NP is contained in coNP/poly. Elect. Colloq. Comput. Complex. (ECCC) 15(022): 2008.
7. H. L. Bodlaender, R. G. Downey, M. R. Fellows, D. Hermelin: On problems without polynomial kernels. J. Comput. Syst. Sci. 75(8): 423-434 2009.
8. J. Flum and M. Grohe. Describing parameterized complexity classes. Information and Computation 187, 291-319 2003.
9. J. Flum and M. Grohe. Parameterized Complexity Theory. Springer, 2006.
10. K.A. Abrahamson, R.G. Downey, and M.R. Fellows. Fixed-parameter tractability and completeness IV: On completeness for W[P] and PSPACE analogs. Annals of pure and applied logic, 73:235-276, 1995.
11. L. Babai and S. Moran. Arthur-Merlin games: a randomized proof system, and a hierarchy of complexity classes. Journal of Computer and System Sciences, 36: p.254-276. 1988.
12. L. Fortnow and R. Santhanam. Infeasibility of instance compression and succinct PCPs for NP. Journal of Computer and System Sciences, 77(1):91-106, January 2011. Special issues celebrating Karp's Kyoto Prize.
13. M. Sipser. Introduction to the Theory of Computation.Course Technology, 2nd edition, 2005.
14. R. M. Karp and R. J. Lipton. Some connections between nonuniform and uniform complexity classes. Proceedings of the Twelfth Annual ACM Symposium on Theory of Computing, pp. 302-309, doi:10.1145/800141.804678, 1980.
15. R. Niedermeier. Invitation to Fixed Parameter Algorithms. Oxford University Press, 2006.
16. S. Arora and B. Barak. Computational Complexity: A Modern Approach. Cambridge University Press, 2009.
17. S. Goldwasser, S. Micali, and C. Rackoff. The Knowledge complexity of interactive proof-systems. Proceedings of 17th ACM Symposium on the Theory of Computation, Providence, Rhode Island. 1985, pp. 291-304.
18. S. Kratsch, M. Wahlstrom: Preprocessing of Min Ones Problems: A Dichotomy. ICALP (1) 2010: 653-665.
19. Y. Chen, J. Flum, M. Muller. Lower bounds for kernelizations. CRM Publications, Nov. 2008.