Instance Compression for the Polynomial Hierarchy and beyond

Citation for published version:
Chakraborty, C & Santhanam, R 2012, Instance Compression for the Polynomial Hierarchy and beyond. in Parameterized and Exact Computation: 7th International Symposium, IPEC 2012, Ljubljana, Slovenia, September 12-14, 2012. Proceedings. vol. 7535, Springer Berlin Heidelberg, pp. 120-134. DOI: 10.1007/978-3-642-33293-7_13

Digital Object Identifier (DOI):
10.1007/978-3-642-33293-7_13

Link:
Link to publication record in Edinburgh Research Explorer

Document Version:
Peer reviewed version

Published In:
Parameterized and Exact Computation

Instance Compression for the Polynomial Hierarchy and Beyond

Chiranjit Chakraborty
Rahul Santhanam

School of Informatics, University of Edinburgh, UK

Abstract. We define instance compressibility ([1], [7], [5], [6]) for parametric problems in PH and $PSPACE$. We observe that the problem $\Sigma_i CircuitSAT$ 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 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. *CircuitSAT* is non-uniformly compressible within NP implies Σ_i*CircuitSAT* 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 $Succinct\ IP$ and show that $QBFormulaSAT$ (Quantified Boolean Formula Satisfiability) is in $Succinct\ IP$.

1 Introduction

An NP problem is said to be instance compressible if there is a polynomial-time reduction mapping instances of size m and witness length 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 ([1]) motivated by applications in cryptography. This notion is closely related to the notion of polynomial kernelizability in parametrized complexity ([7], [5], [6]), which is motivated by algorithmic applications. Fortnow and Santhanam showed ([2]) that the compressibility of the satisfiability problem for Boolean formulae (even non-uniformly) is unlikely, since it would imply that the Polynomial Hierarchy collapses. Since then, there's been a very active stream of research extending this negative result to other problems in NP ([7], [8], [9] etc.). Instance compressibility is a useful notion from the point of view of complexity theory as well - Buhrman and Hitchcock [10] 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 extended 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 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 characterisation 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 characterisation to extend the definition of compressibility in a natural way to parametric problems in PH and $PSPACE$.

There have been proposals made in the parametrized complexity setting ([12] [11]) for defining in general the parametrized complexity analogue of a classical complexity class. Our definition seems similar in spirit, but there are important differences. All the problems we consider *are* in fact fixed-parameter tractable. What we're interested in is whether they are instance-compressible, or equivalently whether they have polynomial-size kernels. The theory developed so far has dealt with problems which are in NP - we'd like to extend it to the Polynomial Hierarchy and beyond.

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 of interest to study what kinds of analogues to classical results hold for the new notion. The result of Fortnow and Santhanam [2] can be thought of as an analogue of the Karp-Lipton theorem, 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 $PSPACE$. 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 $\Sigma_i CircuitSAT$, which is in some sense complete for parametric problems in Σ_i^p. Note that we need a stronger assumption that in the Fortnow-Santhanam result: they need only to assume that SAT is compressible. This reflects the fact that the result is technically much harder - it relies on the Fortnow-Santhanam result 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 $\Sigma_i CircuitSAT$, 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 make the observation that under the assumption that $\Sigma_3 CircuitSAT$ is 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), it follows that all of the Polynomial Hierarchy is 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 is an adaptation of the Fortnow-Santhanam technique to $PSPACE$. As they do, we consider an "OR" version of the problem, 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 trivial; however, this is not the case for $PSPACE$ and this is where we need to work a little harder.

In the next section, we have defined $Succinct\ IP$. This is actually the extension of the complexity class IP. Then we have shown that, not just $QBCNFSAT$, $QBFormulaSAT$ (Quantified Boolean Formula Satisfiability) is in $Succinct\ IP$.

There are many open problems in the compressibility theory for NP such as whether there any unlikely consequences of SAT being probabilistically compressible, and whether the problem $AND\text{-}SAT$ 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, and that if we manage to settle these questions for the larger classes the techniques used can then be translated back to NP.

2 Some Complexity Theory Notions

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

NP problems in parametric form: Let's consider some well known NP problems in parametric form.
SAT = $\{\langle \varphi, 1^n \rangle \mid \varphi$ is a satisfiable formula, 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 has size at most $n\}$.

In our work, we insist on the parameter being interpretable as the *witness size* for a 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.

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* $n \in N$, $|f(\langle x, 1^n \rangle)| \leq p(n)$ *and* $|(\langle x, 1^n \rangle)| \in L$ *iff* $|f(\langle x, 1^n \rangle)| \in L_A$ *for some language* L_A *in the complexity class* A.

Definition 3. ***Non-uniform Compression:*** *A language* 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.* 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 language in the preceding definition takes advice in case of *Non-uniform Compression*.

Definition 4. ***W-Reduction:*** *[1] 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* $|y| \leq p_1(n_1 + |x|)$ *and* $n_2 \leq p_2$ (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's 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 easily prove that $OR\text{-}SAT \leq_w SAT$.
As we have already mentioned, our primary objective is to extend the idea of compression to higher classes, namely $Polynomial\ Hierarchy$ and $PSPACE$. So, we have considered the standard definitions of the complexity classes Σ_i^p and Π_i^p [15] from $Polynomial\ Hierarchy$ and the class $PSPACE$. Let us now take 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** = $\{\langle\, \varphi,\, 1^n\,\rangle \mid \varphi$ *is a satisfiable quantified boolean formula where 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 where 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 $\Pi_i SAT$ *and* $\Pi_i CircuitSAT$ *in parametric form.*

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

QBFormulaSAT = $\{\langle\, \varphi,\, 1^n\,\rangle \mid \varphi$ *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 *has size at most* $n\}$.

Here we would like to mention that Σ_i*SAT* and Σ_i*CircuitSAT* are complete for Σ_i^p according to Cook-Levin reduction. Similarly *QBCNFSAT*, *QBFormulaSAT* and $QBCSAT$ are complete for $PSPACE$.

Now, We can define a parametric problem corresponding to any language L in 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_R,*we can write* $L_R = \{\langle\, x,\, 1^n\,\rangle \mid \exists\ u_1 \in \{0,1\}^{n_1}\ \forall\ u_2 \in \{0,1\}^{n_2} \ldots Q_i\ u_i \in \{0,1\}^{n_i}\ R\ (x,\, u_1\,,\, \ldots\,,\, u_i) = 1$ *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 $L \in PSPACE$.

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 $PSPACE$ languages where the "witness length" or "guess length" is the parameter of the problem.

Proposition 1. $\Sigma_i CircuitSAT$ *is a complete language with respect to* W*-reduction for* i_{th} *level of Polynomial Hierarchy.*

Proof. Let $L \in \Sigma_i^p$. 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 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. Let C_m be the circuit on inputs of length m - we can generate C_m from 1^m in polynomial time. Hence, $x \in L \Leftrightarrow \exists\, u_1 \in \{0,1\}^{n_1}\ \forall\, u_2 \in \{0,1\}^{n_2} \ldots Q_i\ u_i \in \{0,1\}^{n_i}\ C\ (x, u_1\ , \ldots , u_i) = 1$, 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 corresponding to L to $\Sigma_i CircuitSAT$, since the length of the parameter is preserved. ♣

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

Proposition 2. *$QBCSAT$ is a complete language for $PSPACE$ with respect to W-reductions.*

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 3. *$QBCSAT$ 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 $\Sigma_2 CircuitSAT$ 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.

Theorem 1. *CircuitSAT is non-uniformly compressible within the class NP implies Σ_2CircuitSAT is non-uniformly compressible within the class NP.*

Proof. Let's consider the parametric problem $\Sigma_2 CircuitSAT$ first. For the sake of convenience, we often omit the parameter when talking about an instance of this problem. According to the definition,
$C \in \Sigma_2 CircuitSAT \Leftrightarrow \exists\, u \in \{0,1\}^{n_1}\ \forall\, v \in \{0,1\}^{n_2}\ C\ (u, v) = 1$
$C \notin \Sigma_2 CircuitSAT \Leftrightarrow \forall\, u \in \{0,1\}^{n_1}\ \exists\, v \in \{0,1\}^{n_2}\ C\ (u, v) = 0$
where m is the length of the description of the circuit C and $n = (n_1 + n_2)$ is the number of variables of C.

Let us now fix a specific $u = u_1$. Now, we can define a new language $L^{'}$ as follows,
$\langle C, u_1\rangle \in L^{'} \Leftrightarrow \forall\, v \in \{0,1\}^{n_2}\ C\ (u_1, v) = 1$
$\langle C, u_1\rangle \notin L^{'} \Leftrightarrow \exists\, v \in \{0,1\}^{n_2}\ C\ (u_1, v) = 0$

It's clear from the above definition that $L^{'}$ is a $CoNP$ language (of instance size $\leq\ O(m + n_1)$) and any instance of $L^{'}$ can be polynomial-time reduced to an instance of $Circuit\text{-}UnSAT$, say $C^{'}$ (because $Circuit\text{-}UnSAT$, the language of all unsatisfiable circuits, is a $CONP$-Complete language). As shown in Proposition 1, 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 $\langle C, u_1 \rangle \in L^{'} \Leftrightarrow C^{'} \in Circuit\text{-}UnSAT$. 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 compression algorithm we can now non-uniformly compress the instance $C^{'}$ to an instance of size $poly(n_2)$ of another new language. As $CircuitSAT$ is compressible within NP, clearly the new language will be a $CoNP$ language (as $C^{'}$ is an instance of a $CoNP$ language). Without loss of generality, we can assume this compressed instance $C^{''}$ is an instance of complete language $Circuit\text{-}UnSAT$.

Therefore, $C^{'} \in Circuit\text{-}UnSAT \Leftrightarrow C^{''} = f_1(C^{'}, w_1) = f_1(g(C, u_1), w_1) \in Circuit\text{-}UnSAT$, 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 running in polynomial(in m) time.

Now, if $CircuitSAT$ is non-uniformly compressible within NP so is SAT as SAT is a special case of $CircuitSAT$. Now, $OR\text{-}SAT$ is also non-uniformly compressible as $OR\text{-}SAT$ W-reduces to SAT. It can be proved [2] that if $OR\text{-}SAT$ is non-uniformly compressible then $CoNP \subseteq NP/poly$.

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 convert our $CoNP$ language (here $Circuit\text{-}UnSAT$) instance $C^{''}$ into a NP language instance using polynomial size advice. Let's consider that NP language instance to be a $CircuitSAT$ instance $C^{'''}$. 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,
$C^{'} \in Circuit\text{-}UnSAT \Leftrightarrow C^{'''} = f_2(C^{''}, w_2) = f_2(f_1(g(C, u_1), w_1), w_2) \in CircuitSAT$, 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 [2]. 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
$C \in \Sigma_2 CircuitSAT \Leftrightarrow \exists\, u \in \{0,1\}^{n_1}\ \exists\, v \in \{0,1\}^{n_2}\ C_1\ (u, v) = 1$
$C \notin \Sigma_2 CircuitSAT \Leftrightarrow \forall\, u \in \{0,1\}^{n_1}\ \forall\, v \in \{0,1\}^{n_2}\ C_1\ (u, v) = 0$

The key point is that we have reduced our original $\Sigma_2 CircuitSAT$ 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 NP language instance C_2 of size $poly(n_1 + n_2)$. 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 $\Sigma_2 CircuitSAT$ instance is a valid non-uniform instance compression. Thus it is shown that if $CircuitSAT$ is non-uniformly compressible within NP, $\Sigma_2 CircuitSAT$ is also non-uniformly compressible within NP. ♣

3.2 Instance Compression for higher level

Now we are going to extend the idea for higher classes. It's not difficult to see, if $\Sigma_2 CircuitSAT$ is non-uniformly compressible within NP, $\Pi_2 CircuitSAT$ is non-uniformly compressible within $CoNP$. We will use this in the following theorem.

Theorem 2. *CircuitSAT is non-uniformly compressible within the class NP implies Σ_iCircuitSAT is non-uniformly compressible within the class NP for all $i > 1$.*

$Proof\ Outline$: We are going use induction here. Let's consider $CircuitSAT$ is non-uniformly compressible within NP. To prove $\Sigma_i CircuitSAT$ is compressible for all $i > 1$, the base case i = 2, directly follows from Theorem 1. 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 we are now assuming that $\Sigma_i CircuitSAT$ is non-uniformly compressible within NP for all $i \leq k$ and going to prove that $\Sigma_{k+1} CircuitSAT$ is also non-uniformly compressible within NP.
Now, fixing the first variable, u_1 to $u^{'}$ of $\Sigma_{k+1} CircuitSAT$ instance C as before, we can define a new language similarly as we did in the proof of Theorem 1. Using similar argument we can introduce a circuit C_1 as well. The key point is that we have reduced our original $\Sigma_{k+1} CircuitSAT$ 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. Next, using the compressibility assumption for $CircuitSAT$, we can non-uniformly compress C_1 to an NP language instance C_2 of size $poly(n_1 + n^{'})$ i.e. $poly(n_1 + \ldots + n_{k+1})$. So using mathematical induction we can say if $CircuitSAT$ is non-uniformly compressible within NP, $\Sigma_i CircuitSAT$ is also non-uniformly compressible within NP for all $i > 1$. (detailed proof is mentioned in the Appendix) ♣

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

As $\Pi_i CircuitSAT$ W-reduces to $\Sigma_{i+1} CircuitSAT$, Corollary 1 is trivial. Theorems 1 and 2 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 4. *If $\Sigma_3 CircuitSAT$ is compressible, then $\Sigma_i CircuitSAT$ is compressible for any* $i > 3$.

The above proposition follows from the fact that $\Sigma_3 CircuitSAT$ being compressible implies that SAT is compressible, which implies by the result of Fortnow and Santhanam that PH collapses to Σ_3^p, and hence that every parametric problem in Σ_i^p W-reduces to $\Sigma_3 CircuitSAT$.

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 [2] where it shows, compressibility of SAT implies $NP \subseteq coNP/poly$. To show their result, they used the OR-SAT problem, which is trivially W-reducible to SAT. 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. Unlike in the case of OR-SAT, it is not trivial that the language OR-$QBCNFSAT$ W-reduces to $QBCNFSAT$. There are a couple of different issues. First the quantifier patterns for the formulae $\{\phi_i\}, i = 1 \ldots m$ 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're forced to adopt a different strategy as explained below. Later we have found similar strategy is used in [13], 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 instance of OR-$QBCNFSAT$. 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 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 $y_1, y_2 \ldots y_k$. 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 $\psi_i = z_1 \wedge z_2 \ldots \wedge z_k \wedge \phi_i$, 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, $i = 1 \ldots m$. It is not hard to check that ψ is valid iff one of the ϕ_i's is. ♣

Theorem 3. *If QBCNFSAT is compressible, then $PSPACE \subseteq NP/poly$, and hence $PSPACE = \Sigma_3^p$.*

Proof. Let φ be any $OR\text{-}QBCNFSAT$ instance of size m consisting of the disjunction of Quantified Boolean Formula (QBF) in CNF, each of size at most n. Using Lemma 1, if $QBCNFSAT$ is compressible, $OR\text{-}QBCNFSAT$ is also compressible. So φ is compressible. Rest of the proof follows the similar technique used by Fortnow and Santhanam [2], which more generally shows that any language L for which $OR\text{-}L$ is compressible lies in $coNP/poly$. Thus, since $QBCNFSAT$ is $PSPACE$-complete and $PSPACE$ is closed under complementation, a compression for it 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 $PSPACE$, we have that $PSPACE = \Sigma_3^p$. ♣

5 Succinct IP and PSPACE

IP 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 language 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 $\tilde{\text{P}}$ and string w,

1. $w \in L$ implies Pr[$V \leftrightarrow P$] $\geq$ 2/3, and
2. $w \notin L$ implies Pr[$V \leftrightarrow$ P̃] $\leq$ 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 = $PSPACE$. But we can even prove something more. Not only for $QBCNFSAT$, we can construct *Succinct IP* protocol for $QBFormulaSAT$ as well.

Proposition 5. ***QBFormulaSAT*** $\in$ ***Succinct IP***

$Proof\ Outline$: The key idea is to take an algebraic view of boolean formulae by representing them as polynomials as follows (for 0/1 values).
$x \wedge y \leftrightarrow X\ .\ Y$, $\bar{x} \leftrightarrow$ 1 - X and $x \vee y \leftrightarrow X$*$Y$ = 1 - (1 - X)(1 - Y)
We are considering the inputs are from some finite field $\mathbb{F}$. 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's consider the given a quantified Boolean formula is
Ψ = $Q_1\ x_1\ Q_2\ x_2\ Q_3\ x_3 \ldots\ Q_n\ x_n\ \phi(x_1, \ldots\ , x_n)$, where the size of Ψ is m. ϕ is any boolean formula over n variables.

But because of multiplication, exponent of a variable may grow exponentially. So, to arithmetize Ψ we introduce some new terms in quantification to as follows,
Ψ' = $Q_1\ x_1\ R\ x_1\ Q_2\ x_2\ R\ x_1\ R\ x_2\ Q_3\ x_3\ R\ x_1\ R\ x_2\ R\ x_3 \ldots\ Q_n\ x_n\ R\ x_1\ R\ x_2 \ldots$ $R\ x_n\ \phi(x_1, \ldots\ , x_n)$.
Then we arithmetize the quantifiers as well in standard way [14]. We can actually follow the same IP protocol [14] for $QBCNFSAT$ and see that the degree of the polynomial exchanged at each stage between P and V is atmost 2. Coefficients of the polynomials are from the field $\mathbb{F}$ which is in $poly(n)$. So $O(log(poly(n)))$ size messages are sent in any phase. Number of such phases k are bounded by $O(n^2)$. So it's succinct.

Besides, we can prove, for 'yes' instance, there is no error. Otherwise, the probability $Pr[V\ rejects] \geq (1 - 2/|\mathbb{F}|)^{k-1}$ which is very close to 1 for sufficiently large values of $|\mathbb{F}|$. Even, it will be sufficient for us if $|\mathbb{F}|$ is bounded by a large enough polynomial in n. So we can construct a *Succinct* Interactive proof protocol for $QBFormulaSAT$. (detailed proof is mentioned in the Appendix) ♣

Problem 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. As a result, for polynomial (in m) size field $\mathbb{F}$, the error bound will be much higher. 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. $poly(m)$. So, it's not succinct any more. So we can construct IP protocol for $QBCSAT$, but still don't know how to make it 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 [2] that the immediate consequence is the collapse of *Polynomial Hierarchy* at third level. But it's 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 $OR\text{-}SAT$ or $OR\text{-}CircuitSAT$ 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 can even try to find a $Succinct\ IP$ protocol for $QBCSAT$ to show $Succinct\ IP$ = $PSPACE$ or try to find some negative implications of such protocol existing for $QBCSAT$.

References

1. 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.
2. 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.
3. Chee-Keng Yap. Some consequences of non-uniform conditions on uniform classes. Theoretical Computer Science, 26: 287-300, 1983.
4. Leonard Adleman. Two theorems on random polynomial time. In Proceedings of the 20th Annual IEEE Symposium on the Foundations of Computer Science, pages 75-83, 1978.
5. R. Niedermeier. Invitation to Fixed Parameter Algorithms. Oxford University Press, 2006.
6. J. Flum and M. Grohe. Parameterized Complexity Theory. Springer, 2006.
7. Hans L. Bodlaender, Rodney G. Downey, Michael R. Fellows, Danny Hermelin: On problems without polynomial kernels. J. Comput. Syst. Sci. 75(8): 423-434 (2009)
8. Stefan Kratsch, Magnus Wahlstrom: Preprocessing of Min Ones Problems: A Dichotomy CoRR abs/0910.4518: (2009)
9. Holger Dell, Dieter van Melkebeek: Satisfiability allows no nontrivial sparsification unless the polynomial-time hierarchy collapses. STOC 2010: 251-260
10. 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).
11. 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.
12. J. Flum and M. Grohe. Describing parameterized complexity classes. Information and Computation 187, 291-319 (2003)
13. Y. Chen, J. Flum, M. Mller. Lower bounds for kernelizations. CRM Publications, Nov. 2008.
14. M. Sipser. Introduction to the Theory of Computation.Course Technology, 2nd edition, 2005.
15. S. Arora and B. Barak. Computational Complexity: A Modern Approach. Cambridge University Press, 2009.

Appendix:

Theorem 2: *CircuitSAT is non-uniformly compressible within NP implies that Σ_iCircuitSAT is non-uniformly compressible within NP for all $i > 1$.*

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

Now, suppose $CircuitSAT$ is compressible. To prove $\Sigma_i CircuitSAT$ 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, $\Sigma_2 CircuitSAT$ 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 $\Sigma_i CircuitSAT$ is non-uniformly compressible within NP for all $i \leq k$, we have to prove that $\Sigma_{k+1} CircuitSAT$ is also non-uniformly compressible within NP.

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

So it's clear from the above definition that $L^{'}$ is a Π_k^p language (of instance size $\leq O(m + n_1)$) and any instance of $L^{'}$ can be polynomially reduced to an instance of $\Pi_k CircuitSAT$ (because $\Pi_k CircuitSAT$ is a Π_k^p-Complete language). As shown in Proposition 1, the size of the witness will be preserved in this reduction. So this reduction is essentially a W-reduction. Suppose this $\Pi_k^p CircuitSAT$ 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, $\Sigma_k CircuitSAT$ is non-uniformly compressible within NP. So any $\Pi_k CircuitSAT$ instance, say $C^{'}$ is non-uniformly compressible to a $CoNP$ instance as $\Pi_k CircuitSAT$ = co$\Sigma_k CircuitSAT$. After compression suppose the instance is $C^{''}$ which, without loss of generality, we can take as a $Circuit$-$UnSAT$ instance. Here $|C^{''}| = poly(n^{'})$ where $n^{'} = (n_2 + n_3 +$

$\ldots + n_{k+1})$
So, $C^{'} \in \Pi_k CircuitSAT \Leftrightarrow C^{''} \in Circuit\text{-}UnSAT$.

So from the above arguments we can say,
$C^{'} \in \Pi_k CircuitSAT \Leftrightarrow C^{''} = f_1(\ C^{'}, w_1\ ) = f_1(\ g(C, u^{'}), w_1\ ) \in Circuit\text{-}UnSAT$, 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 a special case of $CircuitSAT$. Now, $OR\text{-}SAT$ is also non-uniformly compressible as $OR\text{-}SAT$ is W-reduced to SAT.

It can be proved that [2] , if $OR\text{-}SAT$ is non-uniformly compressible then $CoNP \subseteq NP/poly$.

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 convert our $CoNP$ language (here $Circuit\text{-}UnSAT$) instance $C^{''}$ into a NP language instance using polynomial size advice. Let's consider that NP language instance to be a $CircuitSAT$ instance $C^{'''}$. In the above procedure, the length of the instance definitely will not increase. So clearly $|C^{'''}| = poly(n^{'})$.

So from the above arguments we can say that,
$C^{'} \in \Pi_k CircuitSAT \Leftrightarrow C^{'''} = f_2(\ C^{''},\ w_2\ ) = f_2(\ f_1(\ g(C,\ u^{'}),\ w_1\ ),\ w_2\ ) \in CircuitSAT$, 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 [2]. 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,
$C \in \Sigma_{k+1} CircuitSAT \Leftrightarrow \exists\, u_1 \in \{0,1\}^{n_1}\ \exists\, v \in \{0,1\}^{n^{'}}\ C_1\ (u_1, v) = 1$
$C \notin \Sigma_{k+1} CircuitSAT \Leftrightarrow \forall\, u_1 \in \{0,1\}^{n_1}\ \forall\, v \in \{0,1\}^{n^{'}}\ C_1\ (u_1, v) = 0$

The key point is that we have reduced our original $\Sigma_{k+1} CircuitSAT$ question to a *CircuitSAT* question, *without* a super-polynomial blowup 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.

Next, using the compressibility assumption for $CircuitSAT$, we can non-uniformly compress C_1 to an NP language instance C_2 of size $poly(n_1 + n^{'})$ i.e. $poly(n_1 + n_2 + \ldots + n_{k+1})$. 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 $\Sigma_{k+1} CircuitSAT$ instance is a valid non-uniform instance compression.

So using mathematical induction we can say if $CircuitSAT$ is non-uniformly compressible within NP, $\Sigma_i CircuitSAT$ is also non-uniformly compressible within NP for all $i > 1$. ♣

Proposition 5: QBFormulaSAT $\in$ Succinct IP

We are now basically going to scrutinize the formal proof of the part, $PSPACE \subseteq IP$ [14]. So we are going to use the same arithmetization technique. Interestingly, not only for $CNF\ SAT$, $Formula\ SAT$ version (Quantified) has Succinct IP as well.

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's consider the given a quantified Boolean formula is
$\Psi = Q_1\ x_1\ Q_2\ x_2\ Q_3\ x_3 \ldots\ Q_n\ x_n\ \phi(x_1, \ldots\ , x_n)$,
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:
$\Psi' = Q_1\, x_1\, R\, x_1\, Q_2\, x_2\, R\, x_1\, R\, x_2\, Q_3\, x_3\, R\, x_1\, R\, x_2\, R\, x_3 \ldots\ Q_n\, x_n\, R\, x_1\, R\, x_2\ \ldots\ R\, x_n$
$\phi(x_1, \ldots\ , x_n)$,
We now rewrite this Ψ' as follows : Ψ' = $S_1\ x_1\ S_2\ x_2\ S_3\ x_3 \ldots\ S_k\ x_k$ [ϕ],
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 atmost $O(n^2)$.

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 arithmetizing ϕ. For $i < k$ we define f_i in terms of f_{i+1}:

$S_{i+1} = \forall$: $f_i(\ldots) = f_{i+1}(\ldots, 0).f_{i+1}(\ldots, 1)$;

$S_{i+1} = \exists$: $f_i(\ldots) = f_{i+1}(\ldots, 0){*}f_{i+1}(\ldots, 1)$;

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

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. Here "$\ldots$" can be replace by a_1 through a_j for appropriate values of j.

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 $Q_i x_i$ in Ψ' in order to reduce the degree of each variable to 1 prior to the squaring due to arithmetizing 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.
.
.
.

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, $\ldots$)
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.
.
.
.

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 by following ways. We can simply replace all the three or more input gates (nodes) in the formula ϕ, by equivalent two input nodes. This will introduce some extra gates (nodes), but now the number of gates in the formula is polynomially bounded in m.

Now for the evaluation of the polynomial p for $r_1, \ldots, r_n$, we will consider the modified 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:

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

where d is the highest degree of the polynomial sent in each stage. We can see that value of k can be atmost $O(n^2)$. 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,

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

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. By the induction hypothesis, 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 (1). Thus we have,

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

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 atmost 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 $QBFormulaSAT$. ♣